# Introduction to Logic

**Course notes by Richard Baron**

**This document is available at www.rbphilo.com/coursenotes**

## *Contents*

**Page**

**Methods – propositional logic**

**Methods – first-order logic**

**Ideas**

Version of April 2014

## *What is logic about?*

Logic covers:

a set of techniques for formalizing and testing arguments;

the study of those techniques, and the powers and limits of different techniques;

a range of philosophical questions about topics like truth and necessity.

The arguments that we formalize may look too simple to be worth the effort. But the process of formalizing arguments forces us to be precise. That is often helpful when we try to construct arguments that use abstract concepts. And the techniques can be used to formalize more complicated arguments. Furthermore, unless we learn the techniques, we cannot understand the philosophical questions that arise out of logic.

### Reading

You can rely on these notes alone. But you may like to do some further reading. There are plenty of textbooks available to choose from. They have different styles, and some will be more to your taste than others. So if you are going to buy a book, have a look at several before you choose which one to buy. You might like to consider the following books.

Samuel Guttenplan, *The Languages of Logic*, second edition. Wiley-Blackwell, 1997.

Colin Allen and Michael Hand, *Logic Primer*, second edition. Bradford Books (MIT Press), 2001. Insist on the 2010 reprint, which corrects some errors. There is a website for the book at http://logic.tamu.edu/Primer/

Paul Tomassi, *Logic*. Routledge, 1999.

## *Propositional logic: formalizing arguments*

In propositional logic, we look at whole propositions, without looking at what is within them, and we consider the consequences of each one being true, or false. We can have a proposition, like “All foxes are greedy”, and just label it true, or false, without worrying about foxes. We just want to play around with its truth value (true, or false).

When we come on to first-order logic, we will start to look at the internal structure of propositions, and what makes them true or false. So if we look at “All foxes are greedy”, we will think about what would make it true (each fox being greedy), and what would make it false (any fox, even just one, not being greedy).

Going back to propositional logic, we might have the following propositions.

| | |
|---|---|
| p: | Porcupines hide in long grass. |
| q: | Quills of porcupines hurt. |
| s: | Shoes are a good idea when walking through long grass. |

Common sense tells us that if p and q are true, then s is true too:

If p and q, then s.

Suppose that someone gives us evidence that p is true, and that q is true. Then we can put the following argument together. The first three lines are the premises, and the last line is the conclusion.

If p and q, then s<br>
p<br>
q

Conclusion: s

Propositional logic tells us that any argument with this form is *valid*: whenever the premises are all true, the conclusion is true too. It does not matter what p, q and s actually are. The premises might be false, but the argument would still be valid: it has got a form that guarantees that the conclusion is true whenever all of the premises are true.

An argument is *sound* if it is valid *and* the premises are all true. So if an argument is sound, the conclusion is true.

A conclusion can still be true if the argument is unsound, or even if it is invalid (that is, not valid). The conclusion can be true for a different reason. Maybe porcupines don’t hide in long grass (so that p is false and the argument is unsound). But shoes could still be a good idea. Maybe thistles grow in long grass.

## Rules for formalizing arguments

We must identify the different propositions. We must identify the smallest units that are propositions. For example:

> “If books are boring, then publishers are cruel” is two propositions related by “if … then”.
>
> “John is tall and brave” is two propositions related by “and”: John is tall and John is brave.
>
> “Bruno is clever or playful” is two propositions related by “or”: Bruno is clever or Bruno is playful. “Or” is inclusive, unless we say otherwise. So Bruno is clever or playful or both.
>
> “Mary is not Scottish” is a smaller proposition with “not” attached: “not (Mary is Scottish)”. We always try to pull the negatives out to the front like this. The brackets show that the “not” applies to the whole proposition, Mary is Scottish. If we use “p” for “Mary is Scottish”, we can write “not p” for “Mary is not Scottish”.

We give the propositions letters such as p, q, r, … . It does not matter which letter we use for which proposition, so long as we use the same letter every time the same proposition comes up.

Now we formalize everything using the letters and these extra elements: if … then, and, or, not. We can also use brackets to group things together. So if we want to say that it is not true that p and q, without saying whether p, q or both are false, we can write “not (p and q)”.

## Exercise: formalizing arguments

Here are some arguments. Formalize each one. Decide which arguments are valid. (It does not normally matter which letter you use for which proposition, but it will help here if you use p for the first one you come to within each argument, q for the second, and so on. This is because we will re-use these arguments later. Start again with p when you start each new argument.)

If it is raining, the grass will grow. It is raining. So the grass will grow.

Either Mary is Scottish or John is Welsh. If John is Welsh, then Peter is Irish. Mary is not Scottish. So Peter is Irish.

If the train is slow, Susan will be late for work. The train is not slow. So Susan will not be late for work.

Either the butler or the gardener was in the house. The maid was in the house. If the maid was not in the house, then the gardener was not in the house. So the butler was in the house.

The moon is made of green cheese. If the moon is made of green cheese, then elephants dance the tango. So elephants dance the tango.

If the weather is warm or today is Saturday, I shall go to the beach today. The weather is not warm. Today is not Saturday. So I shall not go to the beach today.

## *Propositional logic: the connectives*

We can formalize arguments using words, as above, or we can formalize them using symbols called connectives. We use them to replace the words “and”, “or”, “if … then” and “not”. This makes it quicker to write down arguments, and easier to use methods that will show us which arguments are valid.

| In words | In symbols | Alternative symbols |
|---|---|---|
| | | |
| p or q | p v q | |
| p and q | p & q | p.q pq p ∧ q |
| if p then q | p → q | p ⊃ q |
| not p | ¬p | ~p $\overline{p}$ |

**Exercise: using symbols**

Go back to the exercise in the previous section, and re-write the formalized versions of the arguments using these symbols.

**Truth**

If we just say or write “p”, we claim that whatever proposition p stands for is true. If we just say or write “p & q”, we claim that both of the propositions are true (the one that p stands for and the one that q stands for), and so on.

We can use brackets as much as we like to group things together and make it clear what we mean:

p v q & r

can be set out as (p v q) & r

if we mean that at least one of p and q is true, and r is true;

or as p v (q & r)

if we mean that either p is true, or both of q and r are true.

**Truth values**

Instead of just writing “p”, or “p & q”, or whatever, we can play around with the truth and falsity of the different propositions. We can see what happens if p is true, and what happens if it is false, and the same for the other propositions. We can, for example, see what happens if p is true, q is false, r is false and s is true.

We do this by assigning truth values to the different propositions. We use two truth values:

True, which we write as T.

False, which we write as F (some books use ⊥).

Now we can set out the rules for the connectives, by showing how different combinations of truth values for p and q give rise to different truth values for “p v q”, “p & q”, “p → q” and “¬p”. We do this using truth tables, which run through all of the possibilities.

**Or (disjunction: p and q are the disjuncts)**

| **p** | **q** | **p v q** |
|---|---|---|
| | | |
| T | T | T |
| T | F | T |
| F | T | T |
| F | F | F |

The value of T for p v q in the first row means that v stands for inclusive or: p or q or both.

**And (conjunction: p and q are the conjuncts)**

| **p** | **q** | **p & q** |
|---|---|---|
| | | |
| T | T | T |
| T | F | F |
| F | T | F |
| F | F | F |

### If … then (material implication: p is the antecedent, and q is the consequent)

| p | q | p → q |
|---|---|---|
| | | |
| T | T | T |
| T | F | F |
| F | T | T |
| F | F | T |

The last two rows of the truth table for if … then may look a bit odd. If p is false, how can we know whether q would follow from p? So how can we assign a truth value to p → q? But in order for propositional logic to work, we need to get a truth value for p → q for every possible combination of truth values of p and of q. Putting T when p is false is a way of saying that we have no evidence that q would not follow from p. It is also very helpful to put T when p is false. It allows us to build our logical system in a very straightforward way.

### Not (negation)

| p | ¬p |
|---|---|
| | |
| T | F |
| F | T |

### Truth tables for long expressions

We can build up long expressions, and write truth tables for them. We want to do this because we can turn arguments into long expressions, and then test for validity by seeing whether those long expressions always come out true. We will see how this works later.

Here are two examples.

**(p v q) & (q →¬p)**

| p | q | ¬p | p v q | q →¬p | (p v q) & (q →¬p) |
|---|---|---|---|---|---|
| | | | | | |
| T | T | F | T | F | F |
| T | F | F | T | T | T |
| F | T | T | T | T | T |
| F | F | T | F | T | F |

**(p v q) → (¬p v r)**

| **p** | **q** | **r** | **¬p** | **(p v q)** | **(¬p v r)** | **(p v q) → (¬p v r)** |
|---|---|---|---|---|---|---|
| | | | | | | |
| T | T | T | F | T | T | T |
| T | T | F | F | T | F | F |
| T | F | T | F | T | T | T |
| T | F | F | F | T | F | F |
| F | T | T | T | T | T | T |
| F | T | F | T | T | T | T |
| F | F | T | T | F | T | T |
| F | F | F | T | F | T | T |

Note the following points.

- We start with columns for the basic propositions p, q and r, then draw up columns for the parts that build up to the whole expression.

- If we have p and q, we need four rows of truth values: p can be T or F, and q can be T or F, so there are 2 x 2 = 4 possibilities. If we have p, q and r, we need eight rows because there are 2 x 2 x 2 = 8 possibilities. If we have p, q, r and s, we need 2 x 2 x 2 x 2 = 16 rows. And so on.

- We can cover all possibilities as follows. Start with the p column, and divide it in half (half the rows above the line, then half below). Start with the top half, fill it with Ts, and then fill the bottom half with Fs. Then move to the q column, fill the top half of the “p is T” rows with Ts, the bottom half of that section with Fs, the top half of the “p is F” rows with Ts, and the bottom half of that section with Fs. Then move on to the r column. The q column is divided into four chunks (T, F, T, F). Fill the top half of the first chunk with Ts, the bottom half with Fs, the top half of the second chunk with Ts, the bottom half with Fs, and so on. If there is an s column, repeat the process. Fill the top half of each of the eight r chunks with Ts, and the bottom half of each of those eight chunks with Fs.

**Exercise: truth tables**

Draw up truth tables for the following expressions.

p → (q & r)

(q v r) & (p v q)

**Tautologies and contradictions**

Sometimes, we will get T in every row in the final column of a truth table. This means that the expression has to come out true, whatever the truth values of the basic propositions p, q and r. Then we call the expression a tautology.

Sometimes, we will get F in every row in the final column of a truth table. This means that the expression has to come out false, whatever the truth values of the basic propositions p, q and r. Then we call the expression a contradiction.

**Exercise: tautologies and contradictions**

Draw up truth tables for the following expressions. Identify any tautologies, and any contradictions.

p & (¬p v q)

(p → q) & (p & ¬q)

p & ¬p

> (In this example, the truth table only needs two rows of truth values, one for p is T, and one for p is F.)

[ p → (q → r) ] v p

> (In this example, we have two different types of brackets just to make the expression easier to read. We must evaluate q → r first, then p → (q → r), then the whole expression.)

## *Propositional logic: testing what follows from what*

An argument in propositional logic is valid if, and only if, it satisfies this condition:

> Whenever all the premises are true, the conclusion is also true.

Put another way, if we made a list of all the valid arguments, and a list of all the arguments that satisfied the condition, our two lists would be the same. That is the force of "if, and only if".

Now suppose that we have an argument with premises A, B, C and conclusion D.

> (The capital letters stand for whatever expressions using p, q, r, … we have. So for example, A might be (p & q) → s.)

Suppose that the argument is valid.

> Then it will satisfy the condition.
>
> So (A & B & C) → D will be a tautology, for the following reason.
>
> > Whenever all of the premises are true, (A & B & C) will be T, and D will be T as well, so the whole expression will be T.
> >
> > Whenever any premise is false, (A & B & C) will be F, so the whole expression will be T. It won't matter whether D is T or F.

Now suppose that the argument is not valid.

> Then it will fail the condition.
>
> So (A & B & C) → D will not be a tautology, for the following reason.
>
> > There will be some circumstance in which all of the premises are true, (A & B & C) will be T, but D will be F, so the whole expression will be F.

So if the argument is valid, (A & B & C) → D will be a tautology.

And if the argument is not valid, (A & B & C) → D will not be a tautology.

So we can test whether an argument is valid by constructing an expression like (A & B & C) → D, drawing up its truth table, and seeing whether it is a tautology.

(We have been casual in stringing lots of things together in a conjunction. Strictly speaking, it is naughty to write A & B & C, or p & q & r. We have only given a truth table for "&" for two propositions. So we should only apply "&" to two things at a time, and write (A & B) & C, or (p & q) & r. But this only matters when we get very formal about the rules of a logical language. We won't go wrong in practice if we string lots of things together: p & q & r & s is T when all of p, q, r and s are true, and F if even one of them is false. Likewise, p v q v r v s is T so long as at least one of p, q, r and s is true. It is only F if they are all false.)

**Example: testing an argument for validity**

Here is an argument:

p
p → (q v r)
¬q

Conclusion: r

We can see whether the argument is valid by constructing a conditional, with the premises together as the antecedent (the bit before the arrow) and the conclusion as the consequent (the bit after the arrow). We then draw up a truth table for the conditional, and see whether we get T in every row for the whole expression.

[ p & ( p → (q v r) ) & ¬q ] → r

| **p** | **q** | **r** | **¬q** | **q v r** | **p→(q v r)** | **p & (p→(q v r)) & ¬q** | **[p & ( p→(q v r)) & ¬q] → r** |
|---|---|---|---|---|---|---|---|
| | | | | | | | |
| T | T | T | F | T | T | F | T |
| T | T | F | F | T | T | F | T |
| T | F | T | T | T | T | T | T |
| T | F | F | T | F | F | F | T |
| F | T | T | F | T | T | F | T |
| F | T | F | F | T | T | F | T |
| F | F | T | T | T | T | F | T |
| F | F | F | T | F | T | F | T |

We do get T in every row, so the argument is valid.

### Exercise: testing arguments for validity

Here are two arguments. Test them for validity in the same way.

p<br>
q<br>
(p & q) → (r v ¬p)

Conclusion: r

p<br>
¬q<br>
q → (p & r)

Conclusion: r

### Another approach: negating the conclusion

There is another test for validity in propositional logic, which is given in some books. It comes to exactly the same thing, so it does not matter which test we use.

For a valid argument, with premises A, B and C, and conclusion D, (A & B & C) →D is a tautology. It is always true. This means that we never get (A & B & C) true, and D false. So we will find that (A & B & C & ¬D) is a contradiction.

For an invalid argument, (A & B & C) →D is not a tautology. It is sometimes false. This means that we sometimes get (A & B & C) true, and D false. So (A & B & C & ¬D) is not a contradiction.

So the alternative test is this:

> Build a conjunction of all of the premises, and the negation of the conclusion.
>
> Draw up its truth table.
>
> If it is a contradiction (F in every row), then the argument is valid.
>
> If it is not a contradiction (T in at least one row), then the argument is not valid.

### If and only if

We used this expression above. Philosophers use it a lot. “X if and only if Y” means that X and Y are always both true, or both false. Their truth values never get out of step.

The expression is so common, that it has a short form: iff (with two letters f).

We can also use a connective, a double-headed arrow:

“p iff q” can be written “p ↔ q”. It can also be written “p ≡ q”. (There are three bars, not two.)

It has a truth table, as follows.

### Iff (material equivalence, or biconditional)

| **p** | **q** | **p ↔ q** |
|---|---|---|
| | | |
| T | T | T |
| T | F | F |
| F | T | F |
| F | F | T |

We can add the double-headed arrow to our list of symbols if we like, but we don’t need to do so. We can say the same thing using the other connectives. p ↔ q is equivalent to (p → q) & (q → p).

We won’t use the double-headed arrow in the rest of these notes.

## *Propositional logic: a formal language, a system and a theory*

We have been quite casual in stringing propositions together, with or without brackets, and in using different types of bracket. As noted above, this does not normally matter. But we can be more formal. We can lay down rules that say which expressions are allowed. This is useful when we want to prove things about our logical systems. Sometimes, we can base proofs on the fact that only certain expressions are permitted. It is also useful when we know that an argument is long and complicated, and we want to make sure that we do not draw any mistaken conclusions. If we are casual about how we write expressions, we can slip up and write something that we should not. Finally, strict rules about what is permitted make it much easier to write computer programs that can check logical proofs. Computers are not very good at knowing what we mean, if we don't say it in precisely the way they expect.

For everyday purposes, we don't need to follow the strict rules that we set out here. But we do need to know that these rules exist, what they look like, and when they are useful.

### The symbols we can use

Propositional letters: p, q, r, … . We have an infinite supply. There are only 26 letters, but we can use $p_1$, $p_2$, $p_3$, … to get round this.

Connectives: v, &, →, ¬

Brackets: (, ). Only one type of bracket is allowed. We cannot use [, ].

We are not allowed to use any other symbols.

### Well-formed formulae (wffs)

A string of symbols is called a formula. A well-formed formula, or wff, is a formula that we are allowed to write. We must write each wff on a new line. We are not allowed to write anything else.

> Each propositional letter is a wff.
>
> If A is a wff, then so is ¬A.
>
> If A and B are wffs, then so are (A v B), (A & B) and (A → B). The brackets are compulsory.
>
> Nothing else is a wff.

Blank spaces between characters have no significance, so they can be included just to make formulae easier to read.

These rules allow us to build up long formulae. For example:

p, q and r are wffs.

So (p v q), (q & r) and ¬q are wffs.

So ( (p v q) v (q & r) ) is a wff.

So ( ( (p v q) v (q & r) ) → ¬q ) is a wff.

### A logical system

The rules about symbols and wffs give us a formal language. They tell us what we can say, without being ungrammatical. But we also want to know what we can usefully say. In particular, we want to know what follows from what. For that, we need a logical system. It will have three components.

There is a language: a set of symbols and rules that determine what is a wff.

There are some axioms. These are wffs that we just accept as starting-points in arguments. We set out a few axioms below. Some systems manage without axioms. We will see such a system when we look at natural deduction.

There are rules that allow us to move through arguments. Here we will have only one rule, modus ponens, which we set out below.

### A theory

Once we have a logical system, we can use it to construct a theory. A theory is a set of wffs. The theories that we normally find interesting are ones that include:

the axioms (unless we are managing without axioms);

the theorems. These are all the wffs that we can get to as the conclusions of arguments, using the rules that allow us to move through arguments. Axioms also count as theorems;

no other wffs.

It makes sense to pick axioms so that:

they look plausible in themselves;

we can use them to prove everything that ought to be provable, and nothing else. We want to be able to prove all tautologies, and we don’t want to be able to prove contradictions, such as (p & ¬p). Usually, we want axioms that will only allow us to prove tautologies, otherwise we would be able to prove things that might not be true. Sometimes (but not here), we add extra axioms to create a theory that will reflect the nature of some special subject matter, such as a part of mathematics. But then we can only rely on the things that we prove when we are discussing that subject matter.

## A set of axioms

If we stated axioms using p and q, we would have a problem: they would work when we were using p and q, but we would not be able to plug in r and s instead. So we give axiom schemata (singular, axiom schema) instead. If A, B and C are wffs, however simple or complicated, then the following are axioms.

1. $( A \rightarrow (B \rightarrow A) )$

2. $( (A \rightarrow (B \rightarrow C) ) \rightarrow ( (A \rightarrow B) \rightarrow (A \rightarrow C) ) )$

3. $( (\neg A \rightarrow \neg B) \rightarrow (B \rightarrow A) )$

Are these plausible? Yes. We can take them in turn, and construct truth tables for them, giving values T and F to A, B and C as if they were the p, q and r of the type of table we prepared above. We will get T at the end of every row.

Does everything we want follow from them, and nothing we don't want? Yes. This is a tried and tested set of axioms. People have shown that they can be used to prove all the tautologies, and nothing else. We will see how to get things to follow from them in the next section.

## A rule of inference

We need one other tool to allow us to get things to follow from our axioms. This is a rule of inference. It takes us from some wffs to others. If we didn't have it, we would just sit and stare at our axioms, because we would not know how to move from one wff to another.

The rule we use is a very simple one, with a Latin name, modus ponens (the method of affirming):

> From A and $(A \rightarrow B)$, infer B.

Another rule, which is not part of our logical system, is modus tollens (the method of denying):

> From $(A \rightarrow B)$ and $\neg B$, infer $\neg A$.
>
> > Example of modus tollens:
> >
> > > If the tumble dryer is working properly (A), then the washing is dry (B)
> > >
> > > The washing is not dry (¬B)
> > >
> > > So the tumble dryer is not working properly (¬A)

There's nothing wrong with modus tollens, but we want to construct a system with as few rules as possible. So we don't add modus tollens, because we can manage without it.

## *Propositional logic: proofs using axioms*

### A proof of a theorem

A proof of a theorem is a sequence of wffs. The last one is the theorem that we are proving.

Each wff must be one of the following:

> An axiom.
>
> Something that we have already shown is a theorem.
>
> Something that follows from earlier wffs using a rule of inference. The earlier wffs don't have to be the most recent ones. They can be anywhere further back in the proof.
>
> A repetition of an earlier wff, or a re-written version of an earlier wff under this rule: we can insert, or delete, two negation signs in a row, " ¬ ¬ ", whenever we like.

It is worth being precise about what counts as a proof, because that allows us to program computers to check proofs and to find new proofs. It also allows logicians to check that we will be able to prove everything that we should be able to prove, and nothing else.

### A proof that a conclusion follows from certain premises

We may want to show that a conclusion follows from certain premises. Perhaps we cannot show that some conclusion D is a theorem, because it does not follow from the axioms alone, but we can show that it follows from premises A, B and C.

A proof that a conclusion follows from certain premises is a sequence of wffs. The last one is the conclusion that we are proving.

Each wff must be one of the following.

> Something that would be allowed in a proof of a theorem (see above).
>
> One of the premises.

Note the difference from the definition of a proof of a theorem. When we are proving that a conclusion follows from premises, we can help ourselves to those premises when writing down the proof, as well as helping ourselves to axioms and established theorems.

## Example: proof of a theorem

Prove: $( p \to p )$

### *The proof*

1. Axiom 1: $( p \to ( ( q \to p ) \to p ) )$

2. Axiom 2: $( ( p \to ( ( q \to p ) \to p ) ) \to ( ( p \to ( q \to p ) ) \to ( p \to p ) ) )$

3. MP on 1, 2: $( ( p \to ( q \to p ) ) \to ( p \to p ) )$

4. Axiom 1: $( p \to ( q \to p ) )$

5. MP on 4, 3: $( p \to p )$

### *Notes*

In line 1, we use axiom 1 with $A = p$ and $B = ( q \to p )$.

In line 2, we use axiom 2 with $A = p$, $B = ( q \to p )$ and $C = p$.

In line 3, “MP” is short for “modus ponens”. We apply the rule with A = line 1 and B = the second part of line 2, so that $(A \to B)$ = line 2.

In line 4, we use axiom 1 with $A = p$ and $B = q$.

In line 5, we apply modus ponens. A = line 4 and B = the second part of line 3. $(A \to B)$ = line 3.

### Example: proof that a conclusion follows from premises

Given the premises: ( p → q ), ( q → r ), derive the conclusion ( p → r ).

1. ( q → r )

2. ( ( q → r ) → ( p → ( q → r ) ) )

3. ( p → ( q → r ) )

4. ( ( p → ( q → r ) ) → ( ( p → q ) → ( p → r ) ) )

5. ( ( p → q ) → ( p → r ) )

6. ( p → q )

7. ( p → r )

### Exercise: understanding a proof

Prepare notes that explain each line in the above proof, just like the notes that were given against each line, and underneath the proof, in the proof of ( p → p ).

### The link between proofs from premises and conditionals: the deduction theorem and the resolution theorem

Suppose that we have some premises, A and B, and a conclusion C that follows from them.

Then we can move a premise over to the conclusion by turning the conclusion into a conditional, like this:

If C follows from A and B, then ( B → C ) follows from A.

We can repeat the move: ( A → ( B → C ) ) follows from no premises at all, so it is a theorem.

But ( A → ( B → C ) ) is equivalent to ( ( A & B ) → C ) ). We can check that they are equivalent, by using truth tables.

So we have this rule, called the **deduction theorem**:

> Suppose that we have some premises, and a conclusion that follows from them. Then we can take some or all of the premises, make a conjunction out of them, and create a conditional with that conjunction before the arrow and the original conclusion after it. Then the new conditional follows from the remaining premises. If we take all of the premises across to the conditional, the conditional is a theorem.

We can go the other way too, taking the antecedent out of a conditional and making it into a premise. This is called the **resolution theorem:**

> If we have a conditional, which follows from some premises or from none, then we can do the following. Take the antecedent off the conditional, delete the arrow, and make the antecedent into a new premise. Then the consequent follows from the existing premises (if any), plus the new premise.
>
> For example, if ( B → C ) follows from premise A, then C follows from premises A and B.

**Why bother with proofs when we have truth tables?**

As the examples show, even proofs of very simple wffs can get long and complicated. And we need imagination to find the right proof.

Truth tables, on the other hand, allow us to find tautologies in a mechanical way, and reasonably quickly. And we chose the axioms to allow us to prove all the tautologies, and only the tautologies. So we can test whether something is a theorem, by constructing its truth table and seeing whether it is a tautology.

Moreover, the deduction theorem allows us to convert a question about whether some conclusion follows from premises, into a question about whether a conditional is a tautology. So we can check whether a conclusion follows from premises by constructing the conditional, drawing up its truth table and seeing whether the conditional is a tautology.

But despite all these points, there are good reasons for using the concept of a proof from axioms, even if we don’t often bother to construct proofs in propositional logic.

> The first reason is that it is interesting to see how much we can get out of a very small set of axiom schemata and a single rule.
>
> The second reason is that when we want to establish what our logical systems can do, we have to use proofs. Someone had to prove the deduction theorem, before we could use it to say that we often don’t need to bother with proofs. So we need to understand what a proof is, if we are going to understand the foundations that support our day-to-day use of logic.
>
> The third reason relates to the distinction between:
>
> > syntax, the rules that tell us how we can arrange and use symbols;
> >
> > semantics, the rules that tell us what the symbols mean and relate them to the world.
>
> Proofs are defined in syntactic terms. They are governed by rules within a formal language. We say what counts as a wff, say which wffs count as axioms, and define a proof as a certain type of sequence of wffs. We can do all this without thinking about whether our wffs are true, or even what they mean. We are just playing a game with symbols and rules about how we can arrange them.

Truth tables bring in semantics. Sometimes we have particular propositions in mind. For example, “p” might be the proposition that the Government will lose the next election. That brings in meanings, and the real world. At other times, we may not know, or care, what propositions are represented by p, q and r. But even then, when we draw up truth tables, we assume that p, q and r are true, or false. So we assume that they have some meaning, even if we don’t say what it is. If a string of words does not mean anything, we cannot think of it as true, or as false.

We use the notion of proof a lot anyway, in mathematics. So we would like to know how proof, the syntactic game that is detached from the real world, and truth in the real world, are connected. In particular, if we have a specific logical system, we would love to know whether the following are correct:

> if we can prove something, it must be true (soundness);
>
> if something must be true, we can prove it (completeness).

Both of these look like pretty desirable properties. But we can only establish whether a system has them if we have a very clear idea of what a proof is, and a very clear idea of what we mean when we say that something must be true. We get the former by developing a system that allows us to prove theorems. We get the latter by developing things like truth tables. So we need to do both, before we can start to explore the relationship between the syntactic game and truth in the world.

The propositional logic and the first-order logic that are presented here are sound and complete, and the deduction and resolution theorems apply. But it is a serious question, whether a logical system has the two nice properties of soundness and completeness. Not all systems have these properties. Plenty of systems are incomplete. Systems can also be unsound, although that reduces their usefulness.

It is also a serious question, whether the deduction theorem and the resolution theorem apply. There are systems in which the deduction theorem does not apply. The resolution theorem applies whenever modus ponens applies, so it only fails to apply in some fairly strange logical systems.

## *Propositional logic: proofs using natural deduction*

We do not have to use axioms in order to generate proofs. We can use a method that relies on several rules, instead of just the one rule of inference that we need when we rely on axioms. This method is called natural deduction. Of course, we need to make sure that our rules look plausible, that they are adequate to prove as much as we want (all the tautologies), and that they will not allow us to prove other things.

Here is a set of rules that will do the job, split into two tables. There are quite a lot of rules, and we could manage with fewer of them. But having lots of rules allows us to make proofs shorter. We can use one rule, instead of achieving the same effect by using two or three rules drawn from a smaller set of rules, one after the other. And there is no need to learn the rules by heart, except for examinations. Outside examinations, we can always refer to the tables.

### First table: rules of inference

These rules show how we can infer a new line from an earlier line, or from two earlier lines. Each row of the table gives a rule. Take the first rule, modus ponens, as an example. The rule says that if we already have (A → B) and A in a proof, we can write the line B.

A, B, C and D can be any wffs. So we could use modus ponens on (p → q) and p to get to q. Or we could use it on:

( (r & s) → ¬t )
(r & s)

to get to ¬t.

The first column gives the name of each rule. Sometimes the same name appears on two rows. One row gives us a result that involves A, and the other row gives us a result that involves B. We can use the same name, whichever row we use.

The second and third columns show what we need to have, earlier in a proof, in order to apply a rule. For most of the table, we need two things. But for some rules, we only need one thing, which is shown in the second column. And for a few rules at the end, we don’t need anything at all earlier in the proof: we can use the rule whenever we feel like it. These rules have nothing in either the second column or the third column.

The last column shows what we are allowed to write, as a fresh line in a proof, when we can apply a rule.

Note that A, B, C and D do not have to be different from one another. We could, for example, use and-elimination to go from (A & A) to A.

## Rules of inference

| Name of rule | What we need:<br>one line | What we need:<br>the other line | What we can write |
|---|---|---|---|
| | | | |
| Modus ponens (MP) | (A → B) | A | B |
| Modus tollens (MT) | (A → B) | ¬B | ¬A |
| Hypothetical syllogism (HS) | (A → B) | (B → C) | (A → C) |
| Disjunctive syllogism (DS) | (A v B) | ¬A | B |
| Disjunctive syllogism (DS) | (A v B) | ¬B | A |
| Constructive dilemma (CD) | ( (A → B) & (C → D) ) | (A v C) | (B v D) |
| Destructive dilemma (DD) | ( (A → B) & (C → D) ) | (¬B v ¬D) | (¬A v ¬C) |
| Bidirectional dilemma (BD) | ( (A → B) & (C → D) ) | (A v ¬D) | (B v ¬C) |
| And-introduction (AI) | A | B | (A & B) |
| And-elimination (AE) | (A & B) | | A |
| And-elimination (AE) | (A & B) | | B |
| Or-introduction (OI) | A | | (A v B) |
| Or-introduction (OI) | B | | (A v B) |
| Or-elimination (OE) | (A v A) | | A |
| Excluded middle (XM) | | | (A v ¬A) |
| Non-contradiction (NC) | | | ¬ (A & ¬A) |

**Second table: rules of equivalence**

These rules show how we can replace one wff with another. Each row states a separate rule. The first column gives the name of the rule. Wffs that match the entries in the second and third columns can replace each other. A, B and C can be any wffs. Of course, they have to stand for the same wffs in the second-column formula and in the third-column formula.

We can use each rule in either of two ways.

> **Inference:** if we have a whole line earlier in our proof that matches an entry in the second column, we can write a new line that matches the entry in the third column. We can also go the other way: if we have a line that matches something in the third column, we can write a new line that matches the entry in the second column.
>
> Take the first rule, De Morgan-and, as an example.
>
> If we had ¬(p & q) earlier in a proof, we could write (¬p v ¬q).
>
> Going the other way (third column to second column), and with more complicated wffs:
>
> if we had (¬ (q & r) v ¬ (s v t) ) earlier in a proof,
>
> we would have A = (q & r) and B = (s v t),
>
> so we could write ¬ ( (q & r) & (s v t) ).
>
> **Substitution:** if we have a little wff that is part of a big wff, and the little wff matches an entry in the second column, we can write a new line which consists of the big wff, with the little wff replaced by the corresponding entry in the third column. We can also go the other way: if the little wff matches an entry in the third column, we can replace it with the corresponding entry in the second column. If the little wff occurs more than once in the big wff, then we can replace it every time, or just some of the times that it occurs.
>
> Take the second rule, De Morgan-or, as an example.
>
> Suppose that earlier in a proof, we had ( ¬ (p v q) & (r → s) ).
>
> Then we could write a new line, ( (¬p & ¬q) & (r → s) ).

## Rules of equivalence

| Name of rule | One wff | The other wff |
|---|---|---|
| De Morgan-and (DMA) | ¬ (A & B) | ( ¬A v ¬B ) |
| De Morgan-or (DMO) | ¬ (A v B) | (¬A & ¬B) |
| Commutation-and (CA) | (A & B) | (B & A) |
| Commutation-or (CO) | (A v B) | (B v A) |
| Association-and (AA) | ( (A & B) & C ) | (A & (B & C) ) |
| Association-or (AO) | ( (A v B) v C ) | (A v (B v C) ) |
| Distribution-and (DA) | ( A & (B v C) ) | ( (A & B) v (A & C) ) |
| Distribution-or (DO) | ( A v (B & C) ) | ( (A v B) & (A v C) ) |
| Double negation (NN) | A | ¬ ¬ A |
| Material implication (MI) | (A → B) | ( ¬A v B ) |
| Transposition (TR) | (A → B) | ( ¬B → ¬A ) |
| Exportation (EX) | ( (A & B) → C ) | ( A → (B → C) ) |

## The deduction theorem and the resolution theorem

We can go on using these helpful theorems.

**Example: natural deduction**

Prove that $( ( ( ( p \to q ) \,\&\, ( q \to r ) ) \,\&\, \neg r ) \to \neg p )$

We can prove that $\neg p$ follows from $( ( ( p \to q ) \,\&\, ( q \to r ) ) \,\&\, \neg r )$, then rely on the deduction theorem to generate the conditional.

***The proof***

1. Premise: $( ( ( p \to q ) \,\&\, ( q \to r ) ) \,\&\, \neg r )$

2. AE on 1: $( ( p \to q ) \,\&\, ( q \to r ) )$

3. AE on 1: $\neg r$

4. AE on 2: $( p \to q )$

5. AE on 2: $( q \to r )$

6. MT on 5, 3: $\neg q$

7. MT on 4, 6: $\neg p$

**Exercise: natural deduction**

Show that q follows from $( p \,\&\, \neg p )$.

The premise, $( p \,\&\, \neg p )$, is a contradiction. So it should not crop up in real life. Note that we have not specified what “q” stands for. It could stand for “Paris is the capital of France”, or it could stand for “The Earth is flat”. So the proof will show that if we have a contradiction, we can prove anything at all. This is one reason why contradictions are dangerous.

**Making assumptions**

We can assume things in order to prove other things. We do this in ordinary argument, when we interrupt the main flow of the argument, make an assumption, and see where we get. We can do this in either of the following two ways.

> We say “assume for the sake of argument that B”. Then we show that C follows. We cannot conclude that C, but we can conclude that if B, then C. And we can go on to use “If B, then C” in our main argument.
>
> We say “assume for the sake of argument that B”. Then we show that this assumption leads to a contradiction, for example that D is both true and false. Then we can conclude that B must be false. We conclude that not B. And we can go on to use “not B” in our main argument.

When we do logic, we describe this as making assumptions, then discharging them. In this context, “discharging” means “no longer relying on”. We use the word because we discharge our responsibility to show that B, the assumption we started with, is true, because we no longer assert that B. In the first type of argument, we only assert that if B, then C. In the second type, we have shown that B cannot be true.

We fit this technique of making and discharging assumptions into natural deduction by using sub-arguments. Within a sub-argument, we make an assumption, then see where it takes us. But we don’t feed the assumption back into the main argument. We only feed back what we get once we discharge the assumption: “if B, then C”, or “not B”, in the above examples.

### Saying precisely what counts as a proof

It is helpful to say precisely what counts as a proof in natural deduction. If we do that, then we can program computers to check whether proofs work and to find new proofs. And if we are precise, logicians can check that we will be able to prove everything that we should be able to prove, and nothing else.

### A proof of a theorem in natural deduction

A proof of a theorem is a sequence of items, each of which is either a wff or a sub-argument. The proof must end with a wff that is the theorem we are proving.

Each wff must be one of the following.

> Something that we have already shown is a theorem.
>
> Something that we are allowed to write under a rule, or that is a repetition of an earlier wff. Earlier wffs that we use when applying rules, or that we repeat, don’t have to be the most recent wffs. They can be anywhere further back in the proof. But they cannot be wffs within sub-arguments.
>
> A wff that comes straight after a sub-argument, under the rules for sub-arguments.

Note the differences from our earlier definition of a proof, when we were using axioms.

> There are no axioms, so we cannot use them.
>
> We have a new type of wff, the wff that comes straight after a sub-argument.

**A proof that a conclusion follows from certain premises in natural deduction**

A proof that a conclusion follows from certain premises is a sequence of items, each of which is either a wff or a sub-argument. The proof must end with a wff that is the conclusion we are proving.

Each wff must be one of the following:

> Something that would be allowed in a proof of a theorem (see above).
>
> One of the premises.

**Sub-arguments**

The definition of a sub-argument is the same for proofs of theorems, and for proofs that conclusions follow from premises.

A sub-argument is a sequence of items, each of which is either a wff or a sub-argument. That is, we can have sub-arguments within sub-arguments. They would be sub-sub-arguments, or sub-sub-sub-arguments, and so on, of the main argument.

> Each wff must be one of the following:
>
> > The assumption of the sub-argument. We will call it B. We will only allow one assumption, in order to keep the rules about what can come after a sub-argument simple. But the assumption might be a single propositional letter, like p, or it might be some more complicated wff, like (p & ¬q). The permission to use complicated wffs means that only allowing one assumption is not really a restriction.
> >
> > Something that we have already shown is a theorem.
> >
> > One of the premises of the main argument (if it has premises).
> >
> > Something that we are allowed to write under a rule, or that is a repetition of an earlier wff. Earlier wffs that we use when applying rules, or that we repeat, don't have to be the most recent wffs. They can be anywhere further back in the sub-argument. They can also be anywhere before the sub-argument, subject to this restriction:
> >
> > > We cannot use anything that is inside a sub-argument that has already ended.
> > >
> > > This rule applies, whatever the levels of the sub-arguments that we are in or that have ended. Suppose that we start the main argument, then we enter a sub-argument X, then it ends and we resume the main argument. Later, we enter another sub-argument Y. When we are within it, we go down another level, into a sub-sub argument Z. While we are in Z, we can use wffs from the part of Y that comes before Z (not the part after Z), and we can use wffs from the main argument. But we cannot use wffs from inside X, because it has ended. And when we are in a part of Y after the end of Z, we cannot use wffs from inside Z.

### The wff straight after a sub-argument

The last restriction means that we cannot go into sub-arguments after they have ended, and take wffs out of them to use later. We also cannot take wffs out of them to use in the main argument, because our definitions of proofs forbid that. So what is the point of sub-arguments?

The point is that once we have constructed a sub-argument, we can write a wff straight after it, and use that wff elsewhere.

> If the assumption is B, and the sub-argument includes the wff C, we can write (B → C) straight after the sub-argument. We can do this several times, so that we get several wffs straight after the same sub-argument. Thus if the sub-argument includes the wffs C, E and F, we can write (B → C), (B → E) and (B → F).
>
> If the assumption is B and the sub-argument includes a contradiction, like (D & ¬D), then we can write ¬B straight after the sub-argument. Often, the contradiction we get is (B & ¬B). That is, we show that if B is true, it is also false. Then it has to be false.

If our sub-argument is within another sub-argument, the wff we write straight after it is also within the larger sub-argument. This is important. We might have made assumptions within the larger sub-argument and relied on them, or on things that followed from them, within the smaller one. That means that we cannot allow our wff after the smaller one to escape to outside the larger one. It depends on the assumptions that we made in the larger one. If our smaller sub-argument does not rely on assumptions that are made within the larger sub-argument, or on things that follow from them, then we can write it as a separate sub-argument, outside the larger one. Then the wff that comes after it will not be trapped within the larger sub-argument.

### Sub-arguments as boxes

Think of each sub-argument as a box. We can make what assumptions we like within the box. But the only things that can come out from the box are wffs that we can write straight after the sub-argument, like (B → C) or ¬B.

If we have a later sub-argument within the same main argument, we cannot go back into earlier boxes that have already closed, in order to fish things out of them.

And we can have boxes within boxes.

When we write a proof that uses sub-arguments, it is helpful to mark them off by drawing boxes round them.

**Example: natural deduction using sub-arguments**

Show that the conclusion $(q \rightarrow r)$ follows from the two premises $(q \rightarrow p)$ and $(\neg r \rightarrow \neg p)$.

1. Premise: $(q \rightarrow p)$

2. Premise: $(\neg r \rightarrow \neg p)$

3. Assumption: $q$

4. MP on 1, 3: $p$

5. NN on 4: $\neg \neg p$

6. MT on 2, 5: $\neg \neg r$

7. NN on 6: $r$

8. After sub-argument: $q \rightarrow r$

**Exercise: natural deduction using sub-arguments**

Show that $\neg q$ follows from the three premises $(q \rightarrow p)$, $(p \rightarrow \neg r)$ and $(\neg r \rightarrow \neg q)$. Use a sub-argument that makes the assumption q, and that ends in a contradiction.

## *First-order logic: formalizing statements*

In propositional logic, we look at propositions as wholes. We just assign truth-values to them. We don't care what they are about.

In first-order logic, we look at the internal structure of propositions, and what makes them true or false. So if we look at "All foxes are greedy", we think about what would make it true (each fox being greedy), and what would make it false (any fox, even just one, not being greedy).

We fix the two properties, being a fox and being greedy, then look at each object in turn and see whether it has these properties.

The phrase "first-order" is used because this logic deals with sentences in which the properties are fixed. We choose the properties, and then look at various objects to see whether they have those properties. If we want to deal with sentences in which the properties are not fixed, like "All properties of Fred are properties of Joe", we have to do second-order logic.

First-order logic is a type of predicate logic. This name means that it deals with predicates, as well as names. In "Albert is a fox", "Albert" is a name, but "is a fox" is a predicate.

### Checking whether all foxes are greedy

There are two ways to find out whether all foxes are greedy.

> We can go through the whole universe, and round up all the foxes. Having done that, we can check each fox to see whether it is greedy.
>
> We can go through the whole universe, pick up each object and inspect it. If it is not a fox, we just put it back. If it is a fox, we check whether it is greedy, and then put it back. Then we go on to the next object.

These two approaches will get us to the same answer. But first-order logic adopts the second approach. Every object in the universe is considered, and we consider each object once. We do not round up all the objects with one property before checking them for the other.

> So "All foxes are greedy" gets treated as:
>
> For all objects, (the object is a fox → the object is greedy)

Suppose that there are only four objects in the universe, as follows:

| Name | Type | Greedy or not |
|---|---|---|
| | | |
| Albert | Fox | Greedy |
| Beatrice | Fox | Not greedy |
| Charlie | Hippo | Not greedy |
| Deborah | Koala | Greedy |

When we say that these are the only objects in the universe, we don’t mean that they float around in empty space. We mean that these are the only objects we are going to talk about. They can live in a forest, have things to eat, and so on. But when we say “all objects”, we mean all objects in the set of objects that we have chosen as our universe. And when we say “there exists”, we mean “there exists within that set of objects”. We ignore all the surroundings of the objects in our chosen universe.

In this simple universe, “For all objects (the object is a fox → the object is greedy)” amounts to the following conjunction. (We are going back to being casual about stringing together conjuncts.)

(Albert is a fox → Albert is greedy) &

(Beatrice is a fox → Beatrice is greedy) &

(Charlie is a fox → Charlie is greedy) &

(Deborah is a fox → Deborah is greedy)

The first conjunct has a true antecedent, so we must check whether it has a true consequent. (We have picked up a fox, so we must check whether it is greedy.) It does have a true consequent, so it is true. The second conjunct has a true antecedent, but a false consequent, so it is false. The third and fourth conjuncts have false antecedents, so they are bound to be true. (We have picked up non-foxes, so we can put them back without checking whether they are greedy.)

The overall result is that the conjunction is false, because there is a false conjunct, the second one. So in this universe, it is not true that all foxes are greedy. This is the right result, because there is a fox that is not greedy, Beatrice.

We can state this result formally by putting the sign for “not” in front of the formal statement that we have found to be false, like this:

¬ (For all objects (the object is a fox → the object is greedy) )

### Does everything have a certain property?

We are not limited to conditionals. We can also ask whether all things have some given property. First-order logic sees us as going through the entire universe, picking up each object in turn and checking whether it has the desired property. For example, we can consider “All objects are mammals”.

This is treated as: “For all objects, the object is a mammal”.

In our four-object universe, this amounts to “Albert is a mammal & Beatrice is a mammal & Charlie is a mammal & Deborah is a mammal”.

Each conjunct is true, so the whole conjunction is true, so it is true that all objects are mammals.

If we had considered “All objects are foxes”, we would have tested the statement “For all objects, the object is a fox”. But as soon as we got to Charlie or Deborah, we would have found out that this was false. So we would write:

> ¬ (For all objects, the object is a fox)

### Are there any hippos? Are there any tigers?

In first-order logic, we don’t just ask whether all objects have a certain property, or whether they all make a conditional true. We can also ask whether there exists an object that has a certain property.

Take the question “Are there any hippos?”. The answer is yes, so long as there is at least one hippo. There may be one, two or a million hippos: the answer will still be yes. The answer is only no, if there are no hippos at all.

In order to answer the question, we will pick up each object in the universe and check it to see whether it is a hippo. We may not have to go through all the objects. We can stop as soon as we find a hippo.

To answer the question, we therefore need to test whether it is true that:

> There exists an object such that the object is a hippo

This will be true in our four-object universe so long as this disjunction is true:

> Albert is a hippo v Beatrice is a hippo v Charlie is a hippo v Deborah is a hippo

One of the disjuncts is true (Charlie is a hippo), so the whole disjunction is true, so the answer to “Are there any hippos?” is “Yes”.

If the question is “Are there any tigers?”, we must test whether it is true that:

> There exists an object such that the object is a tiger

This will be true in our four-object universe so long as this disjunction is true:

> Albert is a tiger v Beatrice is a tiger v Charlie is a tiger v Deborah is a tiger

Every disjunct is false, so the whole disjunction is false, so the answer to “Are there any tigers?” is “No”. Informally, we can say “There are no tigers”. Formally, we write:

> ¬ (There exists an object such that the object is a tiger)

**Are there any greedy koalas?**

We can also ask whether there are objects with certain combinations of properties.

Take the question “Are there any greedy koalas?”. The answer is yes, so long as there is at least one object that is both a koala and greedy.

In order to answer the question, we will pick up each object in the universe and check it to see whether it is both a koala and greedy. We may not have to go through all the objects. We can stop as soon as we find one greedy koala.

To answer the question, we therefore need to test whether it is true that:

> There exists an object such that ( the object is a koala & the object is greedy )

This will be true in our four-object universe so long as this disjunction of conjunctions is true:

> (Albert is a koala & Albert is greedy) v
>
> (Beatrice is a koala & Beatrice is greedy) v
>
> (Charlie is a koala & Charlie is greedy) v
>
> (Deborah is a koala & Deborah is greedy)

The first three disjuncts are false. In each of these first three, the object fails the first conjunct: it is not a koala. In the fourth disjunct, the object is a koala, and it is greedy. So the fourth disjunct is true, so the whole disjunction is true. So there does exist an object such that it is a koala and it is greedy, so there are greedy koalas.

If we had asked whether there were any greedy hippos, we would have found that there were not. Informally, we could say “There are no greedy hippos”. Formally, we would write:

> ¬ (There exists an object such that (the object is a hippo & the object is greedy) )

### “Some” and “at least one”

The sort of logic we are doing is quite simple-minded. It considers whether there is at least one thing that has a given property, or whether everything has a given property. In ordinary language, we are more subtle. We can say “a few”, “some”, “several”, and so on.

When we translate sentences like that into first-order logic, we take “some” to mean “at least one”.

> So “Some hippos are muddy” becomes “At least one hippo is muddy”,
>
> that is, “There exists an object such that (the object is a hippo & the object is muddy)”.

When we translate back from first-order logic into ordinary English, it helps to use “there exists”, or “there is at least one”. If we use “some”, people may assume that we mean rather more than one.

### Some - not

We have seen how to say that not everything has a certain property. If we want to say that not everything is a fox, we say:

> ¬ (For all objects, the object is a fox)

But if it is not true that everything is a fox, then there must be something that is not a fox. We can put this as follows:

> There exists an object such that ¬ (the object is a fox)

This is the formal translation of “some things are not foxes”. But again, when we translate back from first-order logic into ordinary English, it helps to use “there exists”, or “there is at least one”, so as not to suggest that there are several things that are not foxes. There may be several, but the sentence in first-order logic does not guarantee this.

Note the position of “¬” in “¬ (the object is a fox)”. We write “not (the object is a fox)”. We don’t write “the object is a non-fox”. The reason for this choice is that it avoids creating a whole extra set of predicates. We have the predicate “is a fox”. We can manage with that. There is no need to invent predicates like “is a non-fox”.

We also found that in our four-object universe, not all foxes were greedy:

> ¬ (For all objects (the object is a fox → the object is greedy) )

Informally, we can say either “not all foxes are greedy” or “some foxes are not greedy”. Formally, there is some object that does not satisfy the conditional. So we can re-write the above formal statement, “¬ (For all objects (the object is a fox → the object is greedy) )”, as:

> There exists an object such that ¬ (the object is a fox → the object is greedy)

So there must be an object that makes the conditional false. A conditional is only false if the antecedent is true and the consequent is false. So there must be an object that is a fox (making the antecedent true) but that is not greedy (making the consequent false). We need to have an object such that:

the object is a fox & ¬ (the object is greedy)

So we can re-write the formal statement:

There exists an object such that ¬ (the object is a fox → the object is greedy)

as:

There exists an object such that ( the object is a fox & ¬ (the object is greedy) )

It is made true by there being at least one fox that is not greedy. That is why the formal statement translates "some foxes are not greedy". In our four-object universe, Beatrice is the fox we need.

**Exercise: formalizing statements**

Formalize each of the following statements. Your answer for each one should start "For all objects", or "There exists an object such that", or it should start with "¬" and then one of those two phrases. Use the symbols &, v, → and ¬ whenever you can.

Some people like football.

All tigers eat meat.

Astronauts exist.

Some philosophers are vegetarian.

Some people do not drive.

There are no Martians.

No cockatoos smoke.

Some dragons are friendly.

Not all dragons are friendly.

All unicorns are white.

## *First-order logic: predicate letters, constants, variables and quantifiers*

When we conduct a long argument, it gets tedious to write out sentences in full. We use symbols instead, just as we replace sentences with p, q and r in propositional logic. In this section, we will see how to do this. Strings of symbols are called formulae.

### Predicate letters

We use a capital letter for each predicate. We normally use F, G, H and so on. If we need letters for lots of predicates, we can use $F_1$, $F_2$, $F_3$, and so on. Then we won't run out of letters.

We might, for example, have:

| | |
|---|---|
| F: | is a fox |
| G: | is greedy |
| H: | is a hippo |
| K: | is a koala |
| M: | is muddy |

The next time we set out an argument, we might assign different meanings to F, G, H, K and M. We assign the meanings that we need for a given argument. But we keep the meaning of each letter fixed throughout the argument.

### Constants

We use lower-case letters a, b, c, and so on, for the names of objects. If we need lots of names, we can use $a_1$, $a_2$, $a_3$, and so on. Every object in the universe must have a constant to label it. We should also give each object only one label. This avoids confusion. It also matters in formal theories.

We might, for example, have:

| | |
|---|---|
| a: | Albert |
| b: | Beatrice |
| c: | Charlie |
| d: | Deborah |

The next time we set out an argument, we might assign different objects to a, b, c and d. We assign the objects that we need for a given argument. But we keep the object that is named by each letter fixed throughout the argument.

Once we have predicate letters and constants,we can write simple sentences. For example:

| | |
|---|---|
| "Albert is a fox" can be written as: | $Fa$ |
| "Beatrice is a fox but not greedy" can be written as: | $Fb \mathbin{\&} \neg Gb$ |
| "If Charlie is a hippo, then Albert is greedy" can be written as: | $Hc \rightarrow Ga$ |

## Variables

We use lower-case letters x, y, z, w for variables. If we need lots of variables, we can use $x_1$, $x_2$, $x_3$, and so on.

A variable is a letter that can stand for any object. So we do not start by specifying which objects our variables stand for. We have a universe of objects, perhaps just our friends Albert, Beatrice, Charlie and Deborah, or perhaps all of the objects in the actual universe, and the variables may stand for any one of them.

We use variables to handle sentences that start “For all”, or “there exists”. They take the place of references to objects, as follows.

| | |
|---|---|
| For all objects, (the object is a fox → the object is greedy): | For all x (Fx → Gx) |
| There exists an object such that the object is a hippo: | There exists x (Hx) |
| ¬ (For all objects, the object is a hippo): | ¬ For all x (Hx) |

## Free and bound variables

“Fx” on its own does not say anything in particular. If x happens to stand for Albert, then it says that Albert is a fox, which is true. If x happens to stand for Deborah, then it says that Deborah is a fox, which is false. But we don’t know what x stands for.

When x occurs in “Fx”, with nothing else to help us understand what x stands for, it is called a free variable. It is free to stand for what it likes, we cannot control it, and we don’t know what to make of a formula that contains it. A formula that contains a free variable, like the formula “Fx” on its own, is called an open formula. It is open to lots of interpretations, and we don’t know which one to choose.

When we put “For all x” or “there exists x” on the front, we say something in particular. We do not tie x down to a particular object. Instead, we use it to represent the fact that we are going to look at every object in the universe, to see whether they all have some specific property, or to see whether at least one object has that property. When we have done that, so that x is not a free variable, we say that x is a bound variable. If there are no free variables in a formula, we say it is a closed formula.

Since we use variables to stand for nothing in particular, or to show that we are going to look at every object, it does not matter which letter we use. These two are interchangeable:

For all x (Fx → Gx)

For all y (Fy → Gy)

They remain interchangeable when we use special symbols instead of “For all” and “There exists”. We will go on to do that now. But “For all x (Fx → Gy)” would not do instead of either of the above, because the x in “For all x” would only connect with the x after F, and not with the y after G.

## Quantifiers

The last step is to replace the phrases "for all" and "there exists" with symbols that are quicker to write. We use an upside-down A for "for all", and a backwards E for "there exists". We put the variable straight after this symbol, and put brackets round the symbol and the variable together.

For all x (Fx → Gx): $(\forall x)$ (Fx → Gx)

There exists x (Hx): $(\exists x)$ (Hx)

¬ For all x (Hx): ¬ $(\forall x)$ (Hx)

∀ is used to create universal quantifiers: we use it when we want to say something about every object in the universe. ∃ is used to create existential quantifiers: we use it when we want to say that something with a given property exists, or that nothing with that property exists.

Note that we put brackets around the bit that comes after the brackets around the quantifier, for example the "Fx → Gx" and the "Hx" above. This is important. This second pair of brackets shows how far the quantifier reaches. The quantifier binds every instance of the quantifier's variable ( "x" in the examples above) within the brackets. We say that everything within the second pair of brackets is within the scope of the quantifier. Anything that comes later, is not within the scope. (The scope also includes the quantifier itself, and any negation signs straight after the quantifier, but not negation signs before the quantifier. We are allowed to have negation signs before the first bracket that follows the quantifier.)

We might for example have:

(∀x) (Fx → Gx) & Hx

Then the x in "Fx" and the x in "Gx" are both within the scope of the quantifier. They are bound variables. But the x in "Hx" is outside the scope of the quantifier. It is a free variable, and "Hx" is an open formula.

We might go on to bind the x in "Hx", for example as follows:

(∀x) (Fx → Gx) & (∃x) (Hx)

That is fine. The x in the first quantifier and the x in the second quantifier do not have anything to do with each other. The first one runs through the whole universe, when we check whether everything that is F, is also G. The second one runs through the universe in a separate operation, when we check whether there is at least one object that is H. But if we want to help people to follow our formulae, we can use different letters for the two bound variables, without changing our meaning, as follows:

(∀x) (Fx → Gx) & (∃y) (Hy)

Some writers do not bother with ∀. They just put the variable in brackets. So (∀x) (Fx → Gx) would be written as (x) (Fx → Gx).

**Exercise: using the symbols of predicate logic**

Put all of the following into the symbols of predicate logic. Use the meanings of a, b, c, d, F, G, H, K and M that were given above.

(Albert is a fox → Albert is greedy)

¬ ( For all objects (the object is a fox → the object is greedy) )

¬ (For all objects, the object is a fox)

There exists an object such that (the object is a koala & the object is greedy)

¬ ( There exists an object such that (the object is a hippo & the object is greedy) )

There exists an object such that (the object is a hippo & the object is muddy)

There exists an object such that ¬ (the object is a fox)

There exists an object such that ¬ (the object is a fox → the object is greedy)

There exists an object such that ( the object is a fox & ¬ (the object is greedy) )

## Relations

So far, we have only used predicates that apply to one object at a time: "is a fox", "is muddy", and so on.

We also want to be able to handle relations between two objects, such as "… loves …".

There are also relations between three, four or more objects that we should be able to handle, such as:

> "… makes toast for … with butter from …":
>
> > Albert makes toast for Charlie with butter from Deborah.

We use capital letters for relations, just like the letters that we use for predicates. As with predicate letters, we must start each argument by saying what each letter stands for, and each letter must stand for the same relation throughout the argument.

Each predicate letter is followed by a single constant like "a", or a single variable like "x". But a relation letter needs to be followed by constants or variables for all of the items that it relates, and they have to be in the right order: "Albert loves Beatrice" does not mean the same thing as "Beatrice loves Albert".

We will stick to our existing set of meanings for letters (a for Albert, F for being a fox, and so on), and add the following.

| | |
|---|---|
| Lxy: | x loves y |
| Txyz: | x makes toast for y with butter from z |

Then we can turn English sentences into symbols as follows.

| | |
|---|---|
| Albert loves Beatrice: | Lab |
| Beatrice loves Albert: | Lba |
| Albert makes toast for Charlie with butter from Deborah: | Tacd |
| Deborah makes toast for Beatrice with butter from Albert: | Tdba |

We have defined the relation letter T as standing for a relation between three objects. This means that we cannot leave any of the objects out. We cannot use T to say that Albert makes toast for Charlie, without mentioning a supplier of butter. If we want to be able to say that, we need another relation letter. Thus we might use Uxy to stand for "x makes toast for y".

### Binding variables in relations

“Fx” on its own, and “Lxy” on its own, do not say anything in particular. If we want them to say things, we must bind the variables by putting quantifiers on the front. We need a separate quantifier for each variable: one for x and one for y in “Lxy”. Here are some examples.

| | |
|---|---|
| $(\forall x)(\forall y)(Lxy)$ | Everybody loves everybody (including themselves) |
| $(\exists x)(\exists y)(Lxy)$ | Somebody loves somebody (maybe someone else, or maybe himself or herself) |
| $(\forall x)(\exists y)(Lxy)$ | Everybody loves somebody or other (not necessarily the same beloved) |
| $(\exists y)(\forall x)(Lxy)$ | There is at least one person in particular whom everybody loves |
| $(\forall x)(Lxx)$ | Everybody loves himself or herself |
| $(\exists x)(Lxx)$ | Somebody loves himself or herself |
| $(\forall x)(Lxc)$ | Everybody (including Charlie) loves Charlie |
| $(\exists x)(Lcx)$ | Charlie loves somebody |

The order of quantifiers matters, when there is a mixture of universal and existential quantifiers.

> $(\forall x)(\exists y)(Lxy)$ means that everybody has found someone or other to love.
>
> $(\exists y)(\forall x)(Lxy)$ means that there is at least one mega-star, whom everybody loves.

The order does not matter within we have a string of quantifiers, all of the same type (all universal or all existential), where there is absolutely nothing else in the string. Thus $(\exists x)(\exists y)$ and $(\exists y)(\exists x)$ have the same effect. So do $(\forall x)(\forall y)$ and $(\forall y)(\forall x)$.

When we have a string of quantifiers, with nothing else in between them apart from negation signs (see below), they all have the same scope. It stretches from the first quantifier (but not any negation sign before it), to the right hand bracket that matches the first left hand bracket after the string.

We should pick variable letters so as to keep everything clear, and avoid tangles.

> When a string of quantifiers has the same scope, we must make sure that each quantifier has a different variable next to it. $(\forall x)(\exists y)(\exists x)$ is wrong. We should change it to $(\forall x)(\exists y)(\exists z)$, and put z instead of x in the formula after the quantifiers where we want to connect with the last quantifier.
>
> We should not duplicate variables when one quantifier is within the scope of another. If we have $(\forall x)(\ Fxc\ \&\ (\exists x)(Gx)\ )$, we should re-write it as $(\forall x)(\ Fxc\ \&\ (\exists y)(Gy)\ )$.
>
> Remember that we can always change one variable letter to another within a formula, so long as we don’t change the meaning by making too few or too many changes.

If there is only one variable within the scope, we only need one quantifier to bind it. We can see this in the examples $(\forall x)(Lxx)$ and $(\exists x)(Lxx)$.

We can mix variables and constants, as we do in the examples $(\forall x)(Lxc)$ and $(\exists x)(Lcx)$

Two different variables, like x and y, can refer to the same thing. Thus if everybody loves everybody, that means that each one loves himself or herself, as well as loving everybody else.

## Negation signs with relations

We can use negation signs, but where we put them affects the meaning. Here are some examples.

| | |
|---|---|
| $\neg (\exists x)(\exists y)(Lxy)$ | Nobody loves anybody (there are no loving relationships at all) |
| $(\exists x) \neg (\exists y)(Lxy)$ | There is at least one person who does not love anybody |
| $(\exists x)(\exists y) \neg (Lxy)$ | There is at least one pair of people (possibly the same person each time) such that the first does not love the second. So it is not true that everybody loves everybody |
| $(\exists y) \neg (\exists x)(Lxy)$ | There is at least one person who is not loved by anybody |
| $\neg (\forall x)(\forall y)(Lxy)$ | It is not true that everybody loves everybody |
| $(\forall x) \neg (\forall y)(Lxy)$ | Everybody is such that he or she does not love everybody. That is, there is nobody who loves everybody (but each person might still love several people) |
| $(\forall x)(\forall y) \neg (Lxy)$ | Nobody loves anybody (there are no loving relationships at all) |
| $(\exists y) \neg (\forall x)(Lxy)$ | There is at least one person who is not loved by everybody |
| $(\forall x) \neg (\exists y)(Lxy)$ | Everybody is such that there does not exist anybody that he or she loves. That is, nobody loves anybody |
| $(\forall x)(\exists y) \neg (Lxy)$ | Everybody is such that there is at least one person that he or she does not love |
| $\neg (\forall x)(Lxc)$ | Not everybody loves Charlie |
| $(\exists x) \neg (Lxc)$ | There is at least one person who does not love Charlie. That is, not everybody loves Charlie |

**Moving negation signs**

Look at the last two examples, ¬ (∀x)(Lxc) and (∃x) ¬ (Lxc). They mean the same thing. There is a general rule here.

> When we have a negation sign on one side of a quantifier (left or right), we can move it to the other side, so long as we change the quantifier from universal to existential, or the other way round.

The rule makes sense because "there does not exist an object such that …" means the same thing as "all objects are such that it is not the case that …".

The negation sign must be right next to the quantifier, with nothing else in between. So if we want to move a negation sign along a string of quantifiers, we have to do it one step at a time. The following example shows how we can do this. It takes us from the first example above to the ninth, and then to the seventh.

> ¬ (∃x)(∃y)(Lxy)
>
> (∀x) ¬ (∃y)(Lxy)
>
> (∀x)(∀y) ¬ (Lxy)

If we want to change some quantifiers, for example because we want to use only universal quantifiers, or only existential ones, but we don't have any negation signs in the formula, that is not a problem. We can always introduce negation signs wherever we want, so long as we introduce them in pairs, ¬ ¬. Putting in a pair of negation signs, or taking a pair out, will not change the meaning of the formula, so long as the two signs are right next to each other with nothing in between.

It is more likely that we will want to get negation signs out of formulae, because they can make them hard to read. We can do that by moving negation signs around, changing quantifiers as we do so, in order to bring the negation signs together. Then whenever we get a pair of negation signs next to each other, we can just delete them.

Suppose, for example, that someone gave us ¬ (∀x) ¬ (∀y) ¬ (∀z) ¬ (Txyz). We could clear out the negation signs, as follows.

¬ (∀x) ¬ (∀y) ¬ (∀z) ¬ (Txyz)

¬ (∀x) ¬ (∀y) ¬ ¬ (∃z) (Txyz)

¬ (∀x) ¬ (∀y) (∃z) (Txyz)

¬ ¬ (∃x)(∀y)(∃z) (Txyz)

(∃x)(∀y)(∃z) (Txyz)

This means that there is someone in particular who makes toast for everyone with butter from someone (possibly a different supplier of butter for each eater of toast).

### Exercise: putting relations into symbols

Use letters with the meanings given above to put the following into symbols.

Charlie loves everybody.

It is not true that Beatrice makes toast for everybody with butter from Deborah.

Everybody makes toast for everybody with butter from everybody.

Somebody makes toast for somebody with butter from Albert.

Nobody makes toast for anybody with butter from Deborah.

### Assigning truth values

So far, it has been pretty clear what makes things true. In our little four-object universe, we can see that it is not true that everything is a fox (because we also have a hippo and a koala). And if we want to know whether it is true that everybody loves everybody, we just go round asking each animal some personal questions.

Now we want to be more formal about this. We want a systematic procedure that will allow us to work out the truth value (true or false) of any closed formula, once we have been told what the constants, predicate letters and relation letters all mean, and once we have been told what objects are in the universe we are considering, what properties each object has and which objects have which relations to which other objects.

### Defining the universe

It is important to be clear about the universe. Are we concerned with just our four animals, or with all human beings, all physical objects, all numbers, all laws passed in France since 1 January 1900, or what? This matters because our variables will range over the items in the chosen universe. If it is all human beings, for example, then when we want to check whether an assertion is true, we will need to pick up each human being and look at him or her. We will ignore other objects.

### Interpreting the constants, predicate letters and relation letters

One approach would be to start with the objects, stick names on them, and set up letters for properties and relations depending on what properties the objects had, and how they related to one another. If we had a universe of human beings, we might for example set up letters for relations like “is the sister of”, and “loves”, and then see which pairs of human beings fitted those relations.

If we did that, we would end up with names for objects, statements of which properties those objects had, and statements of which relations held between which pairs, triples, and so on of objects. That looks like a good place to end up.

We do end up there, but we get there by working the other way round. We start with our constants, predicate letters and relation letters. Then we interpret these symbols by assigning objects and groups of objects to them. We work this way round because we are interested in the logical structures of formulae and of arguments that use them. Which arguments always work, regardless of the universe and the meanings of the constants, predicate letters and relation letters? To find that out, we need to be able to change the interpretations without changing the symbols. So the symbols are the main thing. Specific interpretations are secondary. Think about valid arguments in propositional logic. They work whatever the truth values of the letters p, q and r, so they work whatever meanings we give to those letters.

Here are two possible interpretations of the set of constants and of predicate letters that we have been using. They are both interpretations in our four-object universe.

| **Letter** | **Interpretation 1** | **Interpretation 2** |
|---|---|---|
| | | |
| a | Albert | Albert |
| b | Beatrice | Beatrice |
| c | Charlie | Charlie |
| d | Deborah | Deborah |
| F | { Albert, Beatrice } | { Albert } |
| G | { Albert, Deborah } | { Deborah} |
| H | { Charlie } | { Beatrice, Charlie} |
| K | { Deborah } | { Deborah } |
| L | { (Albert, Albert), (Albert, Beatrice), (Albert, Charlie), (Albert, Deborah), (Deborah, Albert) } | { (Charlie, Albert), (Charlie, Beatrice), (Charlie, Charlie), (Charlie, Deborah), (Deborah, Albert)} |
| M | { Charlie } | { Deborah} |
| T | { (Beatrice, Charlie, Albert) } | Ø |

The curly brackets { } just show that we are dealing with sets. What is between a pair of curly brackets is a set. We can think of a set as a single thing, a collection of objects.

Ø (without curly brackets) is the empty set, the set with no members at all.

An interpretation assigns:

objects to constants

sets of objects to predicate letters

sets of ordered pairs of objects to two-object relation letters

sets of ordered triples of objects to three-object relation letters, and so on for four-object, etc

> The word “ordered” just means that we say what the order is, and that different orders make for different pairs. So the pair (Albert, Beatrice) is not the same as the pair (Beatrice, Albert). If the first pair, but not the second, is assigned to L, and if Lxy means that x loves y, then Albert loves Beatrice, but Beatrice does not love Albert.

An interpretation must not leave gaps. Each object must have a constant, and each constant must have just one object assigned to it. Each predicate or relation letter must have one set, and only one set, assigned to it.

The first interpretation tells us the following.

Albert and Beatrice are foxes, Charlie is a hippo and Deborah is a koala
Only Albert and Deborah are greedy
Albert loves all four of them (including himself)
Deborah loves only Albert
None of the others loves anybody
Only Charlie is muddy
Beatrice makes toast for Charlie with butter from Albert
No other toast-making with butter supplied by anyone goes on

The second interpretation gives us a different story.

Albert is a fox, Beatrice and Charlie are hippos, and Deborah is a koala
Only Deborah is greedy
Charlie loves all four of them (including himself)
Deborah loves only Albert
None of the others loves anybody
Only Deborah is muddy
No toast-making with butter supplied by anyone goes on

Of course, at least one of these interpretations must get the facts wrong, and perhaps both of them do. We would have to go and look at the four animals to find out who was really which species, who was greedy, who loved whom, who was muddy and what toast-making went on. But the important point for us is that we can interpret the constants, predicate letters and relation letters in different ways.

## Assigning truth values to atomic formulae

Atomic formulae are the simplest formulae we have. They are true, or false, under an interpretation, not true or false absolutely. There are two sorts of atomic formula that we need to consider here.

> There are predicate letters and relation letters, followed by the right number of constants.
>
> > Fb is an atomic formula. It is true under interpretation 1 above, because b refers to Beatrice, and Beatrice is one of the objects assigned to F under interpretation 1. But Fb is false under interpretation 2 above, because b refers to Beatrice, and Beatrice is not assigned to F under interpretation 2.
> >
> > Lcd is an atomic formula. In both of our interpretations, c refers to Charlie and d refers to Deborah. Under interpretation 1, (Charlie, Deborah) is not assigned to L, so Lcd is false. But under interpretation 2, (Charlie, Deborah) is assigned to L, so Lcd is true.
>
> There are whole propositions, where we do not ask what they are about, but just ask about their truth values. These are like the p, q and r of propositional logic. In predicate logic, they tend to get represented by capital letters. So the letters look like predicate letters or relation letters, but we can tell that they are not predicate letters or relation letters because they do not have constants or variables straight after them. An interpretation may assign each one the value T, or the value F. That is its truth value under that interpretation.

## Assigning truth values to other formulae

We can now build up from the truth values of the atomic formulae. Those formulae only have truth values under an interpretation, so the more complex formulae will also only have truth values under an interpretation.

If we have two things with truth values, connected by &, v or →, or one thing with a truth value, with ¬ in front of it, we apply the rules that we learnt for propositional logic to work out the truth value of the whole compound.

Now we can turn to the variables that are bound by quantifiers, where there is only one quantifier to deal with at a time. If there are two or more quantifiers together, or together except for negation signs between them, we need to amend our method. We cover that below. And if there is another quantifier within the scope of the quantifier we are working on (but after the string of quantifiers that includes the quantifier we are working on), we need to deal with that in-scope quantifier first, using the method set out here or the method for strings of two or more quantifiers.

The idea is to give values to the variables, so that formulae with them get turned into formulae that can be given truth values using the rules for atomic formulae and for &, v, → and ¬. The values that we give them are the constants, rather than the objects that are assigned to those constants, so that we can apply the rules on atomic formulae.

If there is a negation sign before the quantifier, we start by ignoring it. We get a truth value for the formula that consists of the quantifier and the rest of its scope. Once we have that, we apply negation to turn T into F, or to turn F into T.

**Examples: assigning truth values to formulae with single quantifiers**

If the constants that had objects assigned to them were a, b, c and d, we might take (∃x)(Fx), change the x in Fx to Fa, and discover that our interpretation made Fa true. Then (∃x)(Fx) would be true under our interpretation, because it only says that there is something that is F, so one example will do.

If the formula was (∀x)(Fx), we would have to try out Fa, Fb, Fc and Fd. (∀x)(Fx) says that everything is F, so our interpretation would only make (∀x)(Fx) true if every one of Fa, Fb, Fc and Fd was true under it. That is, the interpretation would have to assign all four objects to F.

If the formula was (∃x)(Fx & Gx), we would have to try out the following four options until we found at least one that was true (making the whole formula true), or we found out that none of them was true (making the whole formula false).

Fa & Ga
Fb & Gb
Fc & Gc
Fd & Gd

If the formula was (∃x) ¬ (Fx & Gx), at least one of the four options would have to be false for the formula to be true.

If the formula was (∀x)(Fx & Gx), we would have to try all four of the options. The formula would be true if all of them were true. Otherwise, it would be false.

If the formula was (∀x) ¬ (Fx & Gx), all of them would have to be false for the formula to be true.

If the formula was (∃x) (Fx → Gx), we would have to try out the following options until we found at least one that was true (making the whole formula true), or we found out that none of them was true (making the whole formula false).

Fa → Ga
Fb → Gb
Fc → Gc
Fd → Gd

If the formula was (∃x) ¬ (Fx → Gx), at least one of the four options would have to be false for the formula to be true.

If the formula was (∀x)(Fx → Gx), we would have to try all four of the options. The formula would be true if all of them were true. Otherwise, it would be false.

If the formula was (∀x) ¬ (Fx → Gx), all four of them would have to be false for the formula to be true.

### Exercise: assigning truth values to formulae with single quantifiers

Go back to our four-object universe, and apply interpretation 1. Which of these formulae are true, and which ones are false? Set out all the formulae that you test, for example $Fa$, or $Fa \rightarrow Ga$, in order to reach your conclusion.

$(\exists x)(Fx \,\&\, Gx)$

$(\exists x) \neg (Fx \,\&\, Hx)$

$(\exists x)( Gx \vee (Fx \,\&\, Hx) )$

$(\forall x)(Gx \rightarrow Lxa)$

$(\forall x)(Gx \vee Hx \vee Kx)$

$\neg (\exists x)(Txca)$

### Assigning truth values when there are strings of quantifiers

If we have a string of two or more quantifiers, with or without some $\neg$ signs between them, then we proceed as follows. (This method also works where there is only one quantifier, but we can use the simpler method described above instead.)

> We identify the scope of the string of quantifiers, and make sure that we have included in the string all quantifiers with that same scope.
>
> We make sure that there are no quantifiers within the scope but outside the string, which are still waiting for this procedure to be applied. If there are, we apply this procedure to them first, then re-start the whole procedure for the string of quantifiers we thought we were going to tackle first.
>
> We re-write the formula so as to get all of the negation signs from within the string of quantifiers out to the left of the string. (It does not matter whether we include any negation sign straight after the string of quantifiers in this procedure.) Remember that a negation sign can move from one side of a quantifier to the other, so long as we change a universal quantifier to an existential one, or vice versa. We then get rid of pairs of negation signs, so that we are left with just one, or none at all. If there is one, we do not take it into account in this procedure. It is not part of the string, and we will deal with it after we have given a truth value to the string of quantifiers and the rest of its scope.

We play with values of the variables in the quantifiers. The formula that consists of the quantifiers and the rest of their scope will be true, if we can find enough values for the variables to make the formula that starts just after the quantifiers true. If we cannot find values like that, it will be false. The values must be drawn from our universe of objects.

> We need to find a single value for each variable attached to an existential quantifier.
>
> We need to be able to give variables that are attached to universal quantifiers all possible values, that is, the constants that are attached to every object in our universe. The formula that starts just after the string of quantifiers must be true every time, when the variables that are attached to existential quantifiers have the values that we have given to them. If there are several universal quantifiers, we must allow their variables to vary independently. That is, every possible value for each variable, in combination with every possible value for each other variable, must make the formula true.
>
> As we change the value of the variable that is attached to any universal quantifier, we can change the values of the variables that are attached to any existential quantifiers that come after it, in order to try to secure truth. But we cannot change the values of variables that are attached to existential quantifiers that come before the universal quantifier.

The idea of playing with values until we find enough that make the formula that comes after the string of quantifiers true is a bit vague. We will give some examples to show what it means. At the end of the section, we give a more precise statement of a procedure.

**Examples: strings of quantifiers**

We will assign truth values to each of the following formulae, under interpretation 1 in our four-object universe.

(∀x)(∃y)(Lxy)

We need to try every possible value for x, and see whether there is, for each value, a value for y that we can drop in so that Lxy is true. It can be a different value of y for each value of x.

If x = a, there is no problem. We can, for example, have y = a (or b, or c, or d, because Albert is so loving). And if x = d, we have no problem, because we can use y = a (Deborah loves Albert).

If x = b, we cannot find a value for y, because Beatrice does not love anyone. We have the same problem if x = c.

So the entire formula is false, because x is in a universal quantifier and there are values of x (b and c) for which we cannot find a value of y to make Lxy true.

$(\exists x)(\forall y)(Lxy)$

We need to find one value for x, such that Lxy will be true for every value of y. We only need one value of x, but it has to be the same one throughout.

x = a will do, because Albert loves everybody, so Laa, Lab, Lac and Lad are all true.

So the entire formula is true.

$(\exists y)(\forall x)(Lxy)$

We need to find one value for y, such that Lxy will be true for every value of x. We only need one value of y, but it has to be the same one throughout.

There is no value of y that will do, because nobody is loved by all of Albert, Beatrice, Charlie and Deborah. Albert is loved by two of them (Albert and Deborah), but that is as good as it gets.

So the entire formula is false.

$(\forall x)(\forall y)(Lxy)$

We need to show that every possible value of x, combined with every possible value of y, leads to Lxy coming out true. We would have to try 16 combinations in our four-object universe, Laa, Lab, … Lba, Lbb, … Ldc, Ldd.

Interpretation 1 does not make all 16 of them true. We only need to find one combination that does not work, and then we can stop. Lba will do, because Beatrice does not love Albert.

So the entire formula is false.

$(\forall x)(\forall y)( ( Fx \,\&\, Mx ) \rightarrow Lxy)$

We have two universal quantifiers at the start, so we start by thinking about all 16 possible combinations of x and y.

But we only need to show that if (Fx & Mx) is true, then Lxy is true. So we only need to worry about combinations that have values of x for which (Fx & Mx) is true. When it is not true, the conditional will have a false antecedent, so it will be true whether or not Lxy is true.

(Fx & Mx) is not true for any value of x. Fx is true when x = a or when x = b, but Mx is only true when x = c.

So the conditional will be true for all 16 of the combinations that we have to try, because its antecedent will be false for every combination.

So the whole formula is true.

(∀x)(∀y)( ( Fx & Lxx ) → Lxy)

> We have two universal quantifiers at the start, so we start by thinking about all 16 possible combinations of x and y.
>
> But we only need to show that if (Fx & Lxx) is true, then Lxy is true. So we only need to worry about combinations that have values of x for which (Fx & Lxx) is true. When it is not true, the conditional will have a false antecedent, so it will be true whether or not Lxy is true.
>
> (Fx & Lxx) is true when x = a. It is false when x = b, because Lbb is false. It is also false when x = c and when x = d, because Fc and Fd are false.
>
> So we need to test the truth of Lxy for the four combinations when x = a. These are Laa, Lab, Lac and Lad. These are all true.
>
> In the other 12 combinations, x = b, c or d. So the antecedent of the conditional is false, so the whole conditional is true. We don’t need to ask whether Lxy is true.
>
> So the conditional is true for all 16 of the combinations that we have to try.
>
> So the whole formula is true.

**Assigning truth values where there are quantifiers – a precise statement of a method**

The procedure that was described above, of finding enough values that would make a formula true, was not very precise. Here, we give a precise statement of a procedure that gives us the same results.

First, we need the idea of a sequence of values. It is is a possible set of values for the variables that appear in our string of quantifiers, in the order that they have in the string. So if the quantifiers were (∃x)(∀y)(∃z)(∀w), and the constants that had objects assigned to them were a, b, c, and d, one sequence would be x = a, y =b, z = a, w = d. Another sequence would be x = c, y = d, z = b, w = b. And so on.

A partial sequence is the first section of a sequence. So for the above example, x = a would be a partial sequence. x = a, y = b, would be another one. A partial sequence must include a value for only the first variable in the string of quantifiers, or values only for the first and second, or only for the first, second and third, and so on.

Now we need the idea of a comprehensive set of sequences of values. It is a set that is big enough to cover all the options that we need to cover, in order to test whether the whole formula, the quantifiers plus the rest of their scope, is true. If we can find one comprehensive set, and every sequence of values within that set makes the formula that comes after the quantifiers true, then the whole formula is true. If we cannot find a comprehensive set with that property, of making the formula that comes after the quantifiers true every time, then the whole formula is false. We only need one comprehensive set with the property, so we can pick any one that works.

A set is comprehensive if it includes every sequence of values that we would get by applying the following procedure.

> We start by getting rid of duplicate constants that name the same object. If, for example, b, c and f all name the same object under the interpretation, we delete all but one of them. It does not matter which ones we delete, so long as we keep only one. If we have been sensible, we won't have duplicate constants in the first place.
>
> The set starts off with no sequence in it.
>
> We start at the left hand end of the string of quantifiers, and work steadily towards the right.
>
> If the first quantifier is a universal quantifier, then we introduce into the set all possible partial sequences that only give a value for its variable. If, for example, there are ten constants that have objects assigned to them, then there will be ten partial sequences like that.
>
> If the first quantifier is an existential quantifier, then we only introduce one partial sequence. It is the partial sequence that gives the value that we choose to the variable. We can choose any possible value, but we must only choose one.
>
> We then move through the rest of the string of quantifiers.
>
> When we reach a universal quantifier, we replace each partial sequence that is already in the set with n copies of itself, where n is the number of constants that have objects assigned to them. We then extend each of the n copies of each partial sequence with a different value for the variable of the quantifier that we are working on. Then we move on to the next quantifier.
>
> When we reach an existential quantifier, we extend each partial sequence that is already in the set by adding a value for the variable of the quantifier that we are working on. We can choose different values for the extensions to different partial sequences, if we want. But we can only add one value to each partial sequence. We do not create any copies of partial sequences.
>
> When we have been through all the quantifiers in the string, we stop.

Note the importance of our rule that every object in the universe must have a constant to label it. This means that when we come to universal quantifiers, working through the constants will cover all the possibilities.

Also note that this procedure only replaces the stage of playing with the values of the variables. We must first carry out the preceding stages: identify the scope, deal with other quantifiers within the scope but outside the string, and get rid of any negation signs within the string.

## *First-order logic: some valid arguments*

If we have an interpretation, we can apply it and see which formulae are true, and which ones are false. But that can get tedious, especially with formulae that have strings of quantifiers in them. If we can find out which arguments from premises to conclusions always work, we can save ourselves a lot of effort. Instead of having to apply an interpretation to each formula separately, we can check the truth of some sentences that we can use as premises of arguments, then deduce conclusions using types of argument that we know are valid. Then we can be sure that our conclusions are true. We won’t have to apply an interpretation to each conclusion.

Another reason for wanting to identify valid arguments is that they allow us to discover things that are always true, regardless of which universe we are studying and regardless of the interpretation we are using. We discovered tautologies when doing propositional logic. In first-order logic, we get closed formulae that are logically valid. Like tautologies, they are always true. We can generate logically valid closed formulae from valid arguments by using the deduction theorem. If a conclusion follows from some premises, then the conditional that has the conjunction of the premises as antecedent, and the conclusion as consequent, will be logically valid.

In this section, we will look at one very useful class of valid argument, the class of valid syllogisms. We will learn several types of argument, that we can use over and over again to produce valid arguments.

### Syllogisms

Here is an example of a syllogism, set out in symbols and then in normal English.

| | | |
|---|---|---|
| Major premise: | $(\forall x)(Mx \rightarrow Px)$ | All monkeys are primates |
| Minor premise: | $(\forall x)(Sx \rightarrow Mx)$ | All sakis are monkeys |
| Conclusion: | $(\forall x)(Sx \rightarrow Px)$ | All sakis are primates |

It should be clear that the argument in symbols is valid. If the premises are true, then the conclusion will be true, whatever the universe, and whatever interpretation we give to the predicate letters S, M and P. So the version in symbols gives us a type of valid argument. We can turn it into a particular argument by interpreting the symbols, or by substituting a version in normal English that uses words which have accepted meanings, and which therefore give us an interpretation.

All syllogisms follow this pattern of a single bound variable, two premises and a conclusion. (Sometimes we need to add an extra premise to the two usual ones.) There are always three predicate letters, one of which appears in both of the usual premises. But the quantifier can be universal or existential, the connective can be “→” or “&”, and negation signs can be used. So there are quite a few types of valid syllogism.

We will shortly give a complete table of the types of valid syllogism. These are not the only valid ways of arguing in predicate logic, as we shall see when we come on to deductions from axioms and natural deduction.

**Traditional terminology**

Syllogisms originated with Aristotle (384 - 322 BC) and with the Stoic tradition in logic, and they were the main focus of work in logic for over 2,000 years. So a terminology has developed around them. Now that first-order logic has become well-established, this terminology is only really of interest to historians. But it is still used, so it is worth being aware that it exists. Here, we outline the terminology. The table of types of valid syllogism that follows includes the appropriate terminology. But this material is only for reference. There is no need to learn it by heart.

There are two premises, and a conclusion. The two premises are called the major premise and the minor premise. They are traditionally given in that order, but any valid argument is still valid if we give them in the reverse order.

Sometimes, we need an extra premise to guarantee that there are some objects of the required types. It is needed when both of the usual premises begin with $(\forall x)$ or with the word "all", and the conclusion begins with $(\exists x)$ or with the word "some". These are the only times when it is needed.

Traditionally, people assumed that objects of the required types existed, so they did not state the extra premise. But the rules of first-order logic mean that we really ought to spell out the extra premise. Without it, the syllogisms that need it would be invalid. We spell it out in the table, wherever it is needed.

There are three terms: the minor term, the middle term and the major term.

The minor term is the subject in the conclusion. That is why we used "S" in the above example. It also appears in the minor premise.

We can have syllogisms in which the minor term refers to an individual, rather than a group of objects with a given property. In the following example, "Denning" is the minor term:

| | | |
|---|---|---|
| Major premise: | $(\forall x)(Jx \rightarrow Ux)$ | All judges are unbiased |
| Minor premise: | $Jd$ | Denning is a judge |
| Conclusion: | $Ud$ | Denning is unbiased |

(If we use the name of an individual as the minor term, we have to amend both the way that the syllogism is expressed in symbols, and the way that it is expressed in English.)

The major term is the predicate in the conclusion. That is why we used "P" in the first example. It also appears in the major premise. In the examples so far, the major terms have been "primate" and "unbiased".

The middle term appears in both the major premise and the minor premise, but not in the conclusion. In the examples so far, the middle terms have been "monkey" and "judge". In the first example, we used "M" for it.

We can build up several different types of syllogism, using “all”, “some”, and negation. There are four types of sentence that can appear, either as premises or as conclusions. They are referred to by letters and they have descriptive names, as follows.

| **Letter** | **In symbols** | **In words** | **Name** |
|---|---|---|---|
| | | | |
| A | $(\forall x)(Fx \rightarrow Gx)$ | All F are G | Universal affirmative |
| E | $(\forall x)(Fx \rightarrow \neg Gx)$ | No F are G | Universal negative |
| I | $(\exists x)(Fx \,\&\, Gx)$ | Some F are G | Particular affirmative |
| O | $(\exists x)(Fx \,\&\, \neg Gx)$ | Some F are not G | Particular negative |

Any type of sentence might be either a premise or a conclusion. So ether F or G might correspond to the major term, the minor term or the middle term, depending on which line in the syllogism we are looking at.

The letters A, E, I and O have been used to give names to the different types of valid syllogism, so as to make it easier to remember the types. (Not just any random mix of premises and a conclusion will give us a valid argument.) We no longer need to worry about learning the types by heart, but we give the names in the table below. The names are formed by using the letters for the two premises and the conclusion, then inserting other letters to make names that are easy to pronounce. They are especially easy to pronounce if you are used to speaking Latin. The earliest source we have for these names is the work of William of Sherwood (1190 - 1249), who did write in Latin.

The examples above both have the form A-A-A, so that type of syllogism has the name Barbara.

Here is another example.

| | | |
|---|---|---|
| Major premise: | $(\forall x)(Vx \rightarrow Lx)$ | All violins are less than 800 years old |
| Minor premise: | $(\exists x)(Vx \,\&\, Bx)$ | Some violins are beautiful |
| Conclusion: | $(\exists x)(Bx \,\&\, Lx)$ | Some beautiful things are less than 800 years old |

The letters here are A-I-I, so this type of syllogism has the name Datisi.

Where an extra premise is needed, that is not reflected in the name. This is because when the names were invented, syllogisms were set out without extra premises. But the name can still tell us if an extra premise is needed. It is needed if the first two vowels are both “a”, both “e” or one of each, and the third vowel is “i” or “o”.

The middle term may be either the F or the G, in both the major premise and the minor premise. This gives rise to a classification of syllogisms into four figures, as follows. “M” is the middle term, “P” is the major term (the predicate in the conclusion) and “S” is the minor term (the subject in the conclusion).

| Figure | Form of major premise | Form of minor premise |
|---|---|---|
| | | |
| First | M - P | S - M |
| Second | P - M | S - M |
| Third | M - P | M - S |
| Fourth | P - M | M - S |

So the first two examples above, about sakis and about Denning, are in the first figure: in the major premise, we have the middle term before the predicate. In the minor premise, we have the subject before the middle term. The third example, about violins, is in the third figure. The middle term, “violins”, comes first in both premises.

**A table of all types of valid syllogism**

In the table, “M” is the middle term, “P” is the major term and “S” is the minor term. The lines are given in the order major premise, minor premise, conclusion. Where we need an extra premise to assert that something exists, this is given on the third line, and the conclusion is the fourth line. The table is split over two pages.

In working through the table:

> think through how each description in symbols matches up with the description in words;
>
> think up examples of each type of syllogism in English, with true premises and true conclusions (like the examples about sakis and about violins);
>
> look carefully at the types of syllogism with extra existence premises. Why wouldn’t they be valid without those premises? And why don’t we need extra existence premises in the other types of syllogism?

| In symbols | In words | Name and figure |
|---|---|---|
| $(\forall x)(Mx \rightarrow Px)$<br>$(\forall x)(Sx \rightarrow Mx)$<br>$(\forall x)(Sx \rightarrow Px)$ | All M are P<br>All S are M<br>All S are P | Barbara<br>First figure |
| $(\forall x)(Mx \rightarrow \neg Px)$<br>$(\forall x)(Sx \rightarrow Mx)$<br>$(\forall x)(Sx \rightarrow \neg Px)$ | No M is P<br>All S are M<br>No S is P | Celarent<br>First figure |
| $(\forall x)(Mx \rightarrow Px)$<br>$(\exists x)(Sx \,\&\, Mx)$<br>$(\exists x)(Sx \,\&\, Px)$ | All M are P<br>Some S are M<br>Some S are P | Darii<br>First figure |
| $(\forall x)(Mx \rightarrow \neg Px)$<br>$(\exists x)(Sx \,\&\, Mx)$<br>$(\exists x)(Sx \,\&\, \neg Px)$ | No M is P<br>Some S are M<br>Some S are not P | Ferio<br>First figure |
| $(\forall x)(Mx \rightarrow Px)$<br>$(\forall x)(Sx \rightarrow Mx)$<br>$(\exists x)(Sx)$<br>$(\exists x)(Sx \,\&\, Px)$ | All M are P<br>All S are M<br>There are some S<br>Some S are P | Barbari<br>First figure |
| $(\forall x)(Mx \rightarrow \neg Px)$<br>$(\forall x)(Sx \rightarrow Mx)$<br>$(\exists x)(Sx)$<br>$(\exists x)(Sx \,\&\, \neg Px)$ | No M is P<br>All S are M<br>There are some S<br>Some S are not P | Celaront<br>First figure |
| | | |
| $(\forall x)(Px \rightarrow \neg Mx)$<br>$(\forall x)(Sx \rightarrow Mx)$<br>$(\forall x)(Sx \rightarrow \neg Px)$ | No P is M<br>All S are M<br>No S is P | Cesare<br>Second figure |
| $(\forall x)(Px \rightarrow Mx)$<br>$(\forall x)(Sx \rightarrow \neg Mx)$<br>$(\forall x)(Sx \rightarrow \neg Px)$ | All P are M<br>No S is M<br>No S is P | Camestres<br>Second figure |
| $(\forall x)(Px \rightarrow \neg Mx)$<br>$(\exists x)(Sx \,\&\, Mx)$<br>$(\exists x)(Sx \,\&\, \neg Px)$ | No P is M<br>Some S are M<br>Some S are not P | Festino<br>Second figure |
| $(\forall x)(Px \rightarrow Mx)$<br>$(\exists x)(Sx \,\&\, \neg Mx)$<br>$(\exists x)(Sx \,\&\, \neg Px)$ | All P are M<br>Some S are not M<br>Some S are not P | Baroco<br>Second figure |
| $(\forall x)(Px \rightarrow \neg Mx)$<br>$(\forall x)(Sx \rightarrow Mx)$<br>$(\exists x)(Sx)$<br>$(\exists x)(Sx \,\&\, \neg Px)$ | No P is M<br>All S are M<br>There are some S<br>Some S are not P | Cesaro<br>Second figure |
| $(\forall x)(Px \rightarrow Mx)$<br>$(\forall x)(Sx \rightarrow \neg Mx)$<br>$(\exists x)(Sx)$<br>$(\exists x)(Sx \,\&\, \neg Px)$ | All P are M<br>No S is M<br>There are some S<br>Some S are not P | Camestros<br>Second figure |

| In symbols | In words | Name and figure |
|---|---|---|
| (∀x)(Mx → Px)<br>(∀x)(Mx → Sx)<br>(∃x)(Mx)<br>(∃x)(Sx & Px) | All M are P<br>All M are S<br>There are some M<br>Some S are P | Darapti<br>Third figure |
| (∃x)(Mx & Px)<br>(∀x)(Mx → Sx)<br>(∃x)(Sx & Px) | Some M are P<br>All M are S<br>Some S are P | Disamis<br>Third figure |
| (∀x)(Mx → Px)<br>(∃x)(Mx & Sx)<br>(∃x)(Sx & Px) | All M are P<br>Some M are S<br>Some S are P | Datisi<br>Third figure |
| (∀x)(Mx → ¬Px)<br>(∀x)(Mx → Sx)<br>(∃x)(Mx)<br>(∃x)(Sx & ¬Px) | No M is P<br>All M are S<br>There are some M<br>Some S are not P | Felapton<br>Third figure |
| (∃x)(Mx & ¬Px)<br>(∀x)(Mx → Sx)<br>(∃x)(Sx & ¬Px) | Some M are not P<br>All M are S<br>Some S are not P | Bocardo<br>Third figure |
| (∀x)(Mx → ¬Px)<br>(∃x)(Mx & Sx)<br>(∃x)(Sx & ¬Px) | No M is P<br>Some M are S<br>Some S are not P | Ferison<br>Third figure |
| | | |
| (∀x)(Px → Mx)<br>(∀x)(Mx → Sx)<br>(∃x)(Px)<br>(∃x)(Sx & Px) | All P are M<br>All M are S<br>There are some P<br>Some S are P | Bramantip<br>Fourth figure |
| (∀x)(Px → Mx)<br>(∀x)(Mx → ¬Sx)<br>(∀x)(Sx → ¬Px) | All P are M<br>No M is S<br>No S is P | Camenes<br>Fourth figure |
| (∃x)(Px & Mx)<br>(∀x)(Mx → Sx)<br>(∃x)(Sx & Px) | Some P are M<br>All M are S<br>Some S are P | Dimaris<br>Fourth figure |
| (∀x)(Px → ¬Mx)<br>(∀x)(Mx → Sx)<br>(∃x)(Mx)<br>(∃x)(Sx & ¬Px) | No P is M<br>All M are S<br>There are some M<br>Some S are not P | Fesapo<br>Fourth figure |
| (∀x)(Px → ¬Mx)<br>(∃x)(Mx & Sx)<br>(∃x)(Sx & ¬Px) | No P is M<br>Some M are S<br>Some S are not P | Fresison<br>Fourth figure |
| (∀x)(Px → Mx)<br>(∀x)(Mx → ¬Sx)<br>(∃x)(Sx)<br>(∃x)(Sx & ¬Px) | All P are M<br>No M is S<br>There are some S<br>Some S are not P | Calemos<br>Fourth figure |

## *First-order logic: wffs and axioms*

We can be formal, and say which expressions are allowed. These are the well-formed formulae, or wffs. We can also set up some axioms. But first, we set out which symbols we can use.

### The logical symbols

Logical symbols always mean the same thing, regardless of the interpretation. They are as follows.

Connectives: &, v, →, ¬

Brackets: (, )

Variables: x, y, z, w and so on. If we need lots of variables, we can use $x_1$, $x_2$, $x_3$, and so on.

Symbols to use when writing quantifiers: ∀, ∃

Equality: =. We have not used this in formulae so far. It says that two terms name the same thing. We can do first-order logic with equality. Then = is a logical symbol. Or we can do first-order logic without equality. Then we cannot use =. Take care: = is not the same as the material equivalence symbol ≡, also written ↔, which only says that two things have the same truth value.

### The non-logical symbols

Non-logical symbols change their meaning, according to the interpretation. For example, "F" might mean "is a fox" under one interpretation, and "is a falcon" under another interpretation. We have the following non-logical symbols.

**Predicate and relation letters,** F, G, H and so on. If we need lots of them, we can use $F_1$, $F_2$, $F_3$, and so on.

> Sometimes it is useful to show how many items come after such a symbol (zero for one that is being used for a proposition that just has a truth value, one for a predicate, two for a two-place relation, and so on). We can do this with superscripts.
>
> So if $F_1$ and $F_2$ were predicate letters but $F_3$ was a two-place relation letter and $F_4$ was being used for a proposition that just had a truth value, we could write $F_1^1$, $F_2^1$, $F_3^2$ and $F_4^0$.
>
> Sometimes people re-use the same subscripts for different letters with different superscripts. So there would be a series of letters that we would use for propositions that just had truth values: $F_1^0$, $F_2^0$, $F_3^0$ and so on, a series for predicates, $F_1^1$, $F_2^1$, $F_3^1$ and so on, a series for two-place relations, $F_1^2$, $F_2^2$, $F_3^2$ and so on, and so on for relations with more and more variables. $F_1^0$, $F_1^1$ and $F_1^2$ would be unconnected with one another, despite having the same subscript.

**Constants,** a, b, c, and so on. If we need lots of them, we can use $a_1$, $a_2$, $a_3$, and so on. Some presentations of logic don't use constants. Instead they use free variables. But if we use constants, that makes it easier for us to understand what we are doing.

**Function letters,** f, g, h (lower case letters), and so on. If we need lots of them, we can use $f_1$, $f_2$, $f_3$, and so on.

> We have not used function letters so far. They are followed by constants or variables, and they pick out specific objects. Thus under a given interpretation, f(b) might mean "the husband of b", and g(c, d) might mean "the number of days between the birthdays of c and d within a single leap year". The result is called the value of the function: b's husband, or a number of days. The husband will be represented by a constant that names him under that interpretation. The number of days will be represented by a numeral like "75", which is also a constant.
>
> It is important to define functions carefully, so that there is no doubt about their values. If a function has given arguments (the b, or the c and the d, in the above examples), it must yield a single value. This is why we had to specify "within a single year" in the example of g(c, d). If c's birthday is in January and d's is in March, or vice versa, we want an answer in the range 29 to 89, counting through February. We don't want an answer in the range 275 to 335, counting from March round to the following January. And we had to specify leap years, because if the end of February falls between the two birthdays, 29 February makes a difference.
>
> The object that a function picks out under a given interpretation must be an object in the universe we have specified. But a function may not have a value for every possible argument.
>
>> Think back to our four-object universe of Albert, Beatrice, Charlie and Deborah. Suppose that Albert and Beatrice are married to each other, but that Charlie and Deborah are both single. And suppose that we want to define f(x) to mean "the husband of x".
>>
>> Then our interpretation will have to include the information that f(Beatrice) = Albert, and that f(Albert), f(Charlie) and f(Deborah) are undefined.
>>
>> Functions that are not always defined make things messy. They require us to add "so long as the function is defined", or similar conditions, to various rules. And a lot of the time, it is obvious that a function is always defined. (Sometimes, we define the universe so as to make it big enough to make all the functions defined for all arguments.) So from here on, we will require all functions to be defined for all arguments.
>
> It can be useful to show how many arguments a function takes. We do this with superscripts. If $f_1$ and $f_2$ represent one-argument functions like "the husband of", but $f_3$ represents a two-argument function like "days between the birthdays of", we can write $f_1^1$, $f_2^1$ and $f_3^2$.
>
> Sometimes people re-use the same subscripts for different function letters with different superscripts. So we could have $f_1^1$, $f_2^1$, $f_3^1$ and so on for one-argument functions, $f_1^2$, $f_2^2$, $f_3^2$ and so on for two-argument functions, and so on for functions with more and more arguments. $f_1^1$ and $f_1^2$ would be unconnected with each other, despite having the same subscript.
>
> We can think of constants as zero-argument functions ($f_1^0$, $f_2^0$ and so on). They always have the same value. Throwing possible arguments at them to try to change their values makes no difference, because they have no places for arguments.

## Terms

We need to say what a term is, before we define wffs.

Any variable or constant is a term.

If $t_1$, $t_2$, $t_3$, … are terms, then any function $f(t_1, t_2, t_3, \dots)$ is a term.

The number of the $t_1$, $t_2$, $t_3$, … must be the right number of arguments for the function.

Nothing else is a term.

We can apply these rules repeatedly. Using the example of f and g above, this would be a term:

g ( b, f(c) )

This would be the number of days between the birthdays of b and of the husband of c.

## Well-formed formulae (wffs)

As with propositional logic, a well-formed formula, or wff, is something that we are allowed to write. We must write each wff on a new line. We are not allowed to write anything else.

There are two classes of wff: atomic and non-atomic. The first two of the following rules give us atomic wffs, but the other rules give us non-atomic wffs.

If F is a predicate or relation letter and $t_1$, $t_2$, … are terms, then $F(t_1, t_2, \dots)$ is an atomic wff.

There must be the right number of terms for F (one for a predicate like “is a fox”, two for a two-place relation like “loves”, and so on).

So far we have written things like $Ft_1t_2t_3$, without brackets or commas. But that was just a casual short form. Now that we have function letters, we can have complicated terms, so we should use commas to separate the terms, and brackets to group them together.

If $t_1$ and $t_2$ are terms, and we are doing logic with equality, then $( t_1 = t_2 )$ is an atomic wff. The brackets are compulsory.

If A is a wff, then so is ¬A.

If A and B are wffs, then so are (A v B), (A & B) and (A → B). The brackets are compulsory.

If B is a wff and x is a variable, then (∀x)(B) and (∃x)(B) are wffs. If B starts with a bracket, or with one or more negation signs then a bracket, then (∀x)B and (∃x)B are also wffs.

Nothing else is a wff.

Blank spaces between characters have no significance, so they can be included just to make formulae easier to read.

There are a few points to note about the rules on wffs.

As in propositional logic, we can apply the rules over and over again to build up complicated wffs.

A wff may have free variables. Then if we want it to say something in particular, we will need to use quantifiers to bind those free variables.

The wff rules show us how to do the binding. We simply pick the right variables and use them when applying this rule: if B is a wff and x is a variable, then $(\forall x)B$ and $(\exists x)B$ are wffs.

We have already seen examples of this with predicate or relation letters, like Fx or Gxy.

We can do the same with function letters. Here is an example.

| | |
|---|---|
| $f(x)$ | x’s favourite composer |
| $Rz$ | z is Russian |
| $(\forall x)( R ( f(x) ) )$ | each person’s favourite composer is Russian |
| $(\forall x) (\forall y)( f(x) = f(y) )$ | everybody has the same favourite composer |

The wff rules do not in themselves require us to observe the rules on constants and variables that we set out earlier:

When a string of quantifiers has the same scope, we must make sure that each quantifier has a different variable next to it. $(\forall x)(\exists y)(\exists x)$ is wrong. We should change it to $(\forall x)(\exists y)(\exists z)$, and put z instead of x in the formula after the quantifiers where we want to connect with the last quantifier.

We should not duplicate variables when one quantifier is within the scope of another. If we have $(\forall x)( Fxc \ \& \ (\exists x)(Gx) )$, we should re-write it as
$(\forall x)( Fxc \ \& \ (\exists y)(Gy) )$.

This does not mean that these rules no longer matter. If we observe the rules, we will make life a lot easier for ourselves. In addition, we will be able to use short-cuts safely, when they would be dangerous and liable to lead us into error if we broke the rules. The wff rules do not include these rules because their goal is to set out what is allowed as briefly as possible, so as to allow people to derive general results, particularly results about logical systems as wholes. Their goal is not to make life easy for people who want to conduct practical arguments.

**Axioms**

As with propositional logic, we can pick out some wffs that we will regard as axioms. We can then construct a theory, which is the set of theorems. The axioms count as theorems, and all the wffs that follow from the axioms also count as theorems. And as with propositional logic, we want axioms that look plausible in themselves, that allow us to prove all the things that we ought to be able to prove, and that do not allow us to prove things that we should not be able to prove.

We have two sorts of axiom.

> The **logical axioms** are basic to the system, and we always have them. They create a framework in which functions, predicates, relations, quantifiers and connectives do the jobs we expect them to do.
>
> Once we have set up that framework, we can add some **non-logical axioms** so as to create a theory that will do what we want. For example, we can add some axioms to create a theory that will express some part of mathematics. These axioms will change from theory to theory.
>
> Sometimes, the framework that is set up by the rules on wffs and by the logical axioms is not enough. For example, we cannot use that framework to write a finite list of all of the non-logical axioms for the arithmetic of the non-negative integers (the numbers 0, 1, 2, 3, … ). We must either say "everything of the following form is an axiom", or move up to second-order logic. But using second-order logic makes life more complicated.

Here, we will only look at logical axioms. There are several possible sets of logical axioms, any of which will do the job. We will present just one set, to show what a set looks like and how axioms can be used in proofs.

As with propositional logic, we actually give axiom schemata. Anything that fits the pattern of a schema is an axiom. Here are the schemata. Numbers 1 to 3 are the ones that we used in propositional logic. Numbers 8 to 12 only apply if we are doing first-order logic with equality.

We can add strings of universal quantifiers on the front of these axioms, and still have axioms.

1. $( A \rightarrow (B \rightarrow A) )$

2. $( (A \rightarrow (B \rightarrow C) ) \rightarrow ( (A \rightarrow B) \rightarrow (A \rightarrow C) ) )$

3. $( (\neg A \rightarrow \neg B) \rightarrow (B \rightarrow A) )$

4. $( A \rightarrow (\forall x) (A) )$

   x must be a variable that does not occur free in A. Then putting the universal quantifier on the front makes no difference to what A says.

5. $( (\forall x)(A \rightarrow B) \rightarrow ( (\forall x)(A) \rightarrow (\forall x)(B) ) )$

x must be a variable. An example can show why this schema is acceptable. If all gorillas laugh, then it follows that: if everything is a gorilla, then everything laughs.

6. $( (\forall x)( \dots x \dots ) \rightarrow ( \dots t \dots ) )$

x must be a variable and t must be a term. And t must be free for x in the consequent (see below).

In this schema, the antecedent says something about everything in the universe, and the consequent says the same thing about a member of the universe. We replace x with t, wherever x is free once we have taken the "(∀x)" off the antecedent.

So if F means "is a fox", and e is a constant that stands for Edward, we can have as an axiom $( (\forall x)( F(x) ) \rightarrow F(e) )$. This says that if everything is a fox, Edward is a fox. And that looks like an acceptable axiom.

If x appears more than once, we have to replace it with t in every single place that x would be free in the antecedent with its "(∀x)" taken off:

$( (\forall x)( L(x, x) ) \rightarrow L(e, e) )$ is fine.

$( (\forall x)( L(x, x) ) \rightarrow L(e, x) )$ is not. We cannot tell what the consequent says.

We do not replace x with t in places where x is still bound, even after we have taken the "(∀x)" off the front of the antecedent. This could happen because there was another "(∀x)" inside the antecedent.

Now we can look at the condition that t must be free for x in the consequent. The problem is that t could be any term. It does not have to be a constant. So it could be a variable, or it could include variables. Then a quantifier that was lurking around could bind a variable and change the meaning. Here is an example of the sort of thing that we must not allow to be an axiom.

$( (\forall x) \neg(\forall y) ( L (x, y) ) \rightarrow \neg(\forall y) (L (y, y) ) )$

This follows the basic rule. Generate the consequent by stripping off the initial quantifier, and replacing the corresponding variable, x, with a term, the variable y. But the resulting conditional does not have to be true. There are universes in which the antecedent is true, because everyone has someone they do not love, but the consequent is false, because everyone loves himself or herself. So we must not let the whole conditional, with its quantifiers, be an axiom.

We block this sort of thing by saying that t must be free for x in the consequent. This means that when we replace the free instances of x with t, no free variable in any instance of t may get bound by any quantifier in the consequent.

7. $(\forall x)(A^{x/t})$, whenever A is a logical axiom that involves some term t.

x is a variable, and t is a term that occurs one or more times in some logical axiom A. $A^{x/t}$ is the same as A, except that all instances of t are replaced by instances of x. We must choose a variable that does not occur anywhere in A. So if x and y occur in A, we could choose z and form the new axiom $(\forall z)(A^{z/t})$. Furthermore, t must not include any variable that is bound in any instance of t in A.

This schema says that if we have a logical axiom that talks about some term, we can conclude that it would be true if it talked about everything. That might look unsafe. After all, we only know that A is true when it talks about t. How can we conclude that it would be true if it talked about all x? In fact, the rule is safe to use. A is a logical axiom. It is not something that just happens to be true. And it is not a non-logical axiom that has been specified for a particular subject matter. Its truth does not depend on what it is talking about, the term t, but on its structure. So we can exchange t for something else, in fact, for anything else.

8. $(\forall x)( x = x )$. x must be a variable.

9. $(\forall x)(\forall y)( ( x = y ) \rightarrow ( y = x ) )$. x and y must be variables.

10. $(\forall x)(\forall y)(\forall z) ( ( ( x = y ) \& ( y = z ) ) \rightarrow ( x = z) )$. x, y and z must be variables.

11. $(\forall x)(\forall y) ( ( x = y ) \rightarrow ( A \rightarrow A^{y\text{-}x} ) )$

x and y are variables. A is a wff. $A^{y\text{-}x}$ is a wff that we get from A by changing x to y in some or all of the places where x occurs free in A. So if a wff A talks about something, it implies the same wff tweaked to talk about the same thing in a different way. But we can only use this axiom schema to change instances of x that are free in A to instances of y, and we must make sure that the new instances of y are also free in $A^{y\text{-}x}$. (“Free in A” means free when we just look at A, and ignore the quantifiers on the front of the axiom schema.)

12. $(\forall x)(\forall y) ( ( x = y ) \rightarrow ( f( \dots x \dots ) = f( \dots y \dots ) ) )$

x and y are variables, and f is any function. So if x and y refer to the same thing, a given function of x (and, perhaps, some other terms in the areas marked … ) has the same value as the same function of y (and the same other terms).

### A rule of inference

As with propositional logic, axioms are not enough. We need a rule of inference, in order to get our arguments going. Our axiom schemata are such that we only need the same rule, modus ponens:

From A and (A → B), infer B.

### The re-writing rules

We can change (∀x) to (∃x), and vice versa, by passing a negation sign from one side of the quantifier to the other.

We can insert, or delete, two negation signs in a row, ¬ ¬, whenever we like. So we can always change (∀x) to ¬(∃x)¬, and vice versa, and (∃x) to ¬(∀x)¬, and vice versa. We just insert ¬ ¬ on one side of the quantifier, and pass one of the two negation signs to the other side.)

If we have (B), and B on its own is a wff, we can write B. That is, we can discard surplus outer brackets, so long as we are left with a wff. But the occurrence of (B) that we use must be a complete formula on its own line, not part of some larger wff.

When we have a quantifier binding one or more instances of a variable, we can change the variable letter used, so long as we do so in the quantifier and throughout its scope, and do not pick a letter that will get bound by any other quantifier.

For example, we can change (∀x)(∀y) (L (x, y) ) to (∀x)(∀z) (L (x, z) ).

When we use any of these re-writing rules, we should set out the re-written wff on a new line in a proof, and treat it as derived from the line that we have re-written.

### A proof of a theorem, and a proof of a conclusion from certain premises, using axioms

A proof is a sequence of wffs. The last one is the theorem or conclusion that we are proving.

Each wff must be one of the following:

An axiom.

Something that we have already shown is a theorem.

One of the premises (this is only allowed when we are proving that a conclusion follows from some premises, not when we are proving a theorem).

Something that follows from earlier wffs using a rule of inference. The earlier wffs don’t have to be the most recent wffs. They can be anywhere further back in the proof.

A repetition of an earlier wff, or a re-written version of an earlier wff under the re-writing rules.

### Example: proof of a conclusion from premises in first-order logic

Premise 1: $(\forall x) ( F(x) \rightarrow G(x) )$

Premise 2: $(\forall x) ( F(x) )$

Conclusion: $G(a)$

#### *The proof*

1. Premise 1: $(\forall x) ( F(x) \rightarrow G(x) )$

2. Axiom 5: $( (\forall x) ( F(x) \rightarrow G(x) ) \rightarrow ( (\forall x)( F(x) ) \rightarrow (\forall x)( G(x) ) ) )$

3. MP on 1, 2: $( (\forall x)( F(x) ) \rightarrow (\forall x)( G(x) ) )$

4. Premise 2: $(\forall x) ( F(x) )$

5. MP on 4, 3: $(\forall x)( G(x) )$

6. Axiom 6: $( (\forall x)( G(x) ) \rightarrow G(a) )$

7: MP on 5, 6: $G(a)$

### The deduction theorem and the resolution theorem

We might want to change a request to prove that a conclusion follows from some premises into a request to prove a conditional, or the other way round. Fortunately, we can do this, because we can use the deduction theorem and the resolution theorem.

### The empty universe

When we studied syllogisms, we saw that we sometimes needed an extra premise, to tell us that something of a particular type existed.

When we do first-order logic with axioms, we have a similar worry. We must look out for cases where there is nothing in the universe. We must also look out for cases where we have constants that do not label anything. Both would create problems for formulae like "$( (\forall x) ( F(x) ) \rightarrow F(a) )$".

There is a special form of logic, called free logic, which can cope with these problems. But it is much simpler just to forbid them. We require at least one object in the universe, and we require every constant to name some object in the universe. We will assume that these same requirements are met when we look at natural deduction, in the next section.

## *First-order logic: natural deduction*

Proofs from axioms can get rather long. Fortunately, there is an alternative. Just as with propositional logic, we can use natural deduction. Instead of a set of axioms and one, or a few, rules of inference, we have lots of rules. We use the rules of propositional logic, and add some new rules that allow us to work with quantifiers.

### Proofs in natural deduction

We can use natural deduction both to prove theorems, and to prove that a given conclusion follows from some given premises.

A proof is a sequence of items, each of which is either a wff or a sub-argument. We give the formal definition below, but it is very similar to the definition for natural deduction in propositional logic.

### Arguments and sub-arguments

We have arguments and sub-arguments, just as we had for natural deduction in propositional logic. The rules that we had there about sub-arguments, what we can use within them and what we can write straight after them, apply here too. There are some additional ways to use sub-arguments, and additional things we can write straight after sub-arguments. These are set out below, in connection with the rules that are called “universal generalization” and “existential instantiation”. For those ways too, the rules that allow the use of earlier material, but not material from inside sub-arguments that have already ended, apply.

### A set of rules to use

We can use all of the rules that we had for natural deduction in propositional logic. These are the rules that were set out in the two tables, “Rules of inference” and “Rules of equivalence”.

> We must take care when we use a rule of equivalence to replace a little wff that is part of a big wff. If there are quantifiers in the big wff, but outside the little wff, they may bind variables within the little wff. So we have to make sure that substitution of one little wff for another does not change the meaning.

We have the following four new rules.

#### Universal instantiation (UI)

If we have a wff $(\forall x)A$, we can write the wff $A^{t/x}$.

> x is a variable and t is a term. $A^{t/x}$ is the same as A, except that all instances of x that are free in A (after we take off the $(\forall x)$ at the start), are replaced by instances of t. t must be free for x in A. This means that when we replace the free instances of x with t, no free variable in t may get bound by any quantifier in the new wff.
>
> The rule says that if we know that a formula is true for all values of x, we can put in a particular value and say something true.

The rule is sometimes called universal elimination.

#### Existential generalization (EG)

If we have a wff A, we can write the wff $(\exists x)(A^{x/t})$ (or $(\exists x)A^{x/t}$ if that is a wff).

> x is a variable, and t is a term that occurs one or more times in A. $A^{x/t}$ is the same as A, except that all instances of t are replaced by instances of x.
>
> We must choose a variable that does not occur anywhere in A. So if x and y occur in A, we could choose z and write the new wff $(\exists z)(A^{z/t})$. And t must not include any variable that is bound in any instance of t in A. That is, t must be free for x in A.
>
> The rule says that if we know that a formula is true for some particular thing, we can say that there exists something, for which it is true.
>
> The rule is only safe to use if all constants refer to objects and all functions are defined for all arguments. Even then, there can be a problem if t is, or includes, a variable that is free in any instance of t in A. So t should be either a constant, or a function with constants as arguments. (This problem goes away when we do first-order logic with variables only, and no constants, but then we have to amend other rules.)

The rule is sometimes called existential introduction.

**Universal generalization (UG)**

If we have a wff A, we can write the wff (∀x)($A^{x/e}$) (or (∀x)$A^{x/e}$ if that is a wff).

> x is a variable, and e is a constant. $A^{x/e}$ is the same as A, except that all instances of e are replaced by instances of x. We must choose a variable that does not occur anywhere in A. So if x and y occur in A, we can choose z and write (∀z)($A^{z/e}$).
>
> There must be a sub-argument that gets us to A. We cannot just introduce A, as an assumption. We must argue our way to it, in the usual way. Then if we want to use universal generalization, we write (∀x)($A^{x/e}$) straight after the sub-argument, just outside its box.
>
> e must be a new constant. e may only appear in the sub-argument, not before it or after it. We include a note at the start of the sub-argument to say that we are introducing e. If there is a sub-sub argument within the sub-argument that also introduces a new constant, it must be different from e. But there can be a sub-sub argument that carries on part of the sub-argument about e, so long as we do not try to write (∀x)($A^{x/e}$) straight after that sub-sub argument. (∀x)($A^{x/e}$) can only come straight after the end of the sub-argument that opened with the introduction of e.
>
> The point of working in this way is that by the end of the sub-argument, we can say that we have proved A for an arbitrary thing e. e could have been anything. So we could have proved A for each thing in the universe separately. So we are entitled to generalize, and write (∀x)($A^{x/e}$).
>
> A sub-argument often proceeds by assuming some wff B and deriving C. That allows us to write (B → C) straight after the sub-argument. But we would also be entitled to write (B → C) within the sub-argument, by applying the deduction theorem to the lines “B” and “C”. When we introduce a new constant, we can also make an assumption B at the start of the same sub-argument, and deduce C. Then writing (B → C) within the sub-argument is usually the best thing to do. Then the formula that we write straight after the sub-argument (just outside its box) is (∀x)$(B \rightarrow C)^{x/e}$.

The rule is sometimes called universal introduction.

If we do first-order logic using variables instead of constants, e is a fresh variable instead of a constant. (We can use letters from near the start of the alphabet to name variables, if we do not need them for constants. Some books prefer to show the fresh variable as an x with a special mark, for example $x_0$.) Again, x must be a variable that does not occur anywhere in A. And e (or $x_0$) must not be bound anywhere in A.

### Existential instantiation (EI)

If we have a wff (∃x)A, we can write the wff $A^{c/x}$.

x is a variable, and c is a constant. $A^{c/x}$ is the same as A, except that all instances of x that are free in A (after we take off the (∃x) at the start), are replaced by instances of c.

The rule says that if we know that a formula is true for some value of x, we can put in a particular value and say something true.

We can only use existential instantiation if c is an acceptable constant to substitute for x. After all, if we had (∃x)( F(x) ), where F meant "is a fox", we could substitute a constant that named a fox, but not any other constant.

The trick is to start with a new constant that we have not used before in the argument, and will not use again. We just take it that the constant names something that makes A come out true. But we only use A, with c in place, in order to prove something else, which we will call K, for the purposes of the rest of the argument.

Once we have finished the sub-argument, we must not go on using c. If we did, we might say something else about it, which might not be true. All we know about c is that it is something that makes A come out true. If we want to prove something else starting with (∃x)A, we can, but we should start again with another constant.

We achieve all this by using a sub-argument with a box round it. The sub-argument must start somewhere after we have had (∃x)A, and the (∃x)A that we use must not be trapped within a sub-argument that has already ended. At the start of the sub-argument, we say that we are introducing the constant c, then write the wff $A^{c/x}$ (that is, A with c put in place of x).

We then argue down to K, applying the usual rules on what we can use within sub-arguments (everything before the sub-argument, except things that are within sub-arguments that have already ended). We can have sub-sub-arguments, but a sub-sub-argument must not re-apply EI to (∃x)A, and if it applies EI to some other wff, (∃x)B, then it must use a different arbitrary constant. K must occur at the level of the sub-argument that introduced c, and not just in any sub-sub-argument within it.

When we have got to K, we can end the sub-argument, and re-write K straight after it. We can then carry on using K.

K must not contain c, because the constant c must have no life after the sub-argument.

We can use EI twice within the same sub-argument, but we need to be careful. We must use different constants for the two uses, but we must not conclude that the two objects that they name are different, or that they are the same. They might be either.

The rule is sometimes called existential elimination.

If we do first-order logic with variables instead of constants, c is a fresh variable instead of a constant (some writers use something like $x_0$). c must not be bound anywhere in A.

## Additional rules

We can use the re-writing rules that we have already seen.

> We can change (∀x) to (∃x), and vice versa, by passing a negation sign from one side of the quantifier to the other.
>
> We can insert, or delete, two negation signs in a row, ¬ ¬, whenever we like.
>
> If we have (B), and B on its own is a wff, we can write B. That is, we can discard surplus outer brackets, so long as we are left with a wff. But the occurrence of (B) that we use must be a complete formula on its own line, not part of some larger wff.
>
> When we have a quantifier binding one or more instances of a variable, we can change the variable letter used, so long as we do so in the quantifier and throughout its scope, and do not pick a letter that will get bound by any other quantifier.

We can use the deduction theorem and the resolution theorem.

## A proof of a theorem, and a proof of a conclusion from premises, in natural deduction

A proof is a sequence of items, each of which is either a wff or a sub-argument. The proof must end with the wff that we are proving.

Each wff must be one of the following.

> Something that we have already shown is a theorem.
>
> One of the premises.
>
>> We can only use premises when we prove that a conclusion follows from some premises. If we want to prove a theorem, we are not allowed any premises. But we still have ways to get started. We can use a sub-argument with an assumption that gets discharged, or we can rely on the deduction theorem to turn the problem of proving a conditional into the problem of showing that its consequent follows from its antecedent, or we can use the rules of excluded middle and non-contradiction.
>
> Something that we are allowed to write under a rule. Earlier wffs that we use when applying rules don’t have to be the most recent wffs. They can be anywhere further back in the proof. But they cannot be wffs within sub-arguments.
>
> A wff that comes straight after a sub-argument, under the rules for sub-arguments.
>
> A repetition of an earlier wff, or a re-written version of an earlier wff under the re-writing rules. The earlier wff must be available for use, under the rules that govern the use of wffs that are in sub-arguments.

## Sub-arguments

A sub-argument is a sequence of items, each of which is either a wff or a sub-argument. That is, we can have sub-arguments within sub-arguments. They would be sub-sub-arguments, or sub-sub-sub-arguments, and so on, of the main argument.

Each wff must be one of the following:

> The assumption of the sub-argument, if there is one. We will call it B. We will only allow one assumption, in order to keep the rules about what can come after a sub-argument simple. But the assumption might be a complicated wff.
>
> Something that we have already shown is a theorem.
>
> One of the premises of the main argument (if it has premises).
>
> Something that we are allowed to write under a rule. Earlier wffs that we use when applying rules don’t have to be the most recent wffs. They can be anywhere further back in the sub-argument. They can also be anywhere before the sub-argument, subject to this restriction:
>
> > We cannot use anything that is inside a sub-argument that has already ended. This rule applies, whatever the levels of the sub-arguments that we are in or that have ended (sub-sub-arguments, etc).
>
> A repetition of an earlier wff, or a re-written version of an earlier wff under the re-writing rules. The earlier wff must be available for use, under the rules that govern the use of wffs that are in sub-arguments.

## The wff straight after a sub-argument

If there is an assumption B and the sub-argument includes the wff C, we can write (B → C) straight after the sub-argument. We can do this several times, so that we get several wffs straight after the same sub-argument. Thus if the sub-argument includes the wffs C, E and F, we can write (B → C), (B → E) and (B → F).

If there is an assumption B and the sub-argument includes a contradiction, like (D & ¬D), then we can write ¬B straight after the sub-argument.

If we used the sub-argument to apply UG, we can write $(\forall x)A^{x/e}$ straight after it.

If we used the sub-argument to apply EI, we can write a wff K from it straight after it, so long as K does not include the arbitrary constant that we used.

If our sub-argument is within another sub-argument, the wff that we write straight after it is also within the larger sub-argument.

### Example: universal instantiation and existential generalization

Premise: $(\forall x)(\ F(x)\ )$

Conclusion: $(\exists x)(\ F(x)\ )$

***The proof***

1. Premise: $(\forall x)(\ F(x)\ )$

2. UI on 1: $F(c)$

3. EG on 2: $(\exists x)(\ F(x)\ )$

### Example: universal generalization

Premise 1: $(\forall x)(\ F(x) \rightarrow G(x)\ )$

Premise 2: $(\forall x)(\ F(x)\ )$

Conclusion: $(\forall x)(\ G(x)\ )$

***The proof***

1. Premise 1: $(\forall x)(\ F(x) \rightarrow G(x)\ )$

2. Premise 2: $(\forall x)(\ F(x)\ )$

UG with constant e

3. UI on 1: $(\ F(e) \rightarrow G(e)\ )$

4. UI on 2: $F(e)$

5. MP on 4, 3: $G(e)$

6. UG on 5: $(\forall x)(\ G(x)\ )$

### Example: existential instantiation and universal instantiation

Premise 1: $(\forall x)( M(x) \rightarrow P(x) )$

Premise 2: $(\exists x)( S(x) \& M(x) )$

Conclusion: $(\exists x)( S(x) \& P(x) )$

***The proof***

1. Premise 1: $(\forall x)( M(x) \rightarrow P(x) )$

2. Premise 2: $(\exists x)( S(x) \& M(x) )$

EI with constant c

3. EI on 2: $( S(c) \& M(c) )$

4. UI on 1: $( M(c) \rightarrow P(c) )$

5. AE on 3: $M(c)$

6. MP on 5, 4: $P(c)$

7. AE on 3: $S(c)$

8. AI on 7, 6: $( S(c) \& P (c) )$

9. EG on 8: $(\exists x)( S(x) \& P(x) )$

10. Repeat 9: $(\exists x)( S(x) \& P(x) )$

***Notes***

Line 9 is the wff that we called K when we described existential generalization.

We have just proved that syllogisms of the type Darii are valid.

We have used modus ponens, and two other rules taken from the table of rules of inference for propositional logic: and-elimination (AE) and and-introduction (AI). This illustrates why existential and universal instantiation are useful. They get rid of quantifiers, and leave us with complete propositions that we can easily use with the rules for natural deduction in propositional logic.

**Exercise: natural deduction in first-order logic**

Construct natural deduction proofs to show that syllogisms of the following types are valid.

*Ferio*

| | |
|---|---|
| Premise: | $(\forall x)( M(x) \rightarrow \neg P(x) )$ |
| Premise: | $(\exists x)( S(x) \,\&\, M(x) )$ |
| Conclusion: | $(\exists x)( S(x) \,\&\, \neg P(x) )$ |

*Disamis*

| | |
|---|---|
| Premise: | $(\exists x)( M(x) \,\&\, P(x) )$ |
| Premise: | $(\forall x)( M(x) \rightarrow S(x) )$ |
| Conclusion: | $(\exists x)( S(x) \,\&\, P(x) )$ |

*Fresison*

| | |
|---|---|
| Premise: | $(\forall x)( P(x) \rightarrow \neg M(x) )$ |
| Premise: | $(\exists x)( M(x) \,\&\, S(x) )$ |
| Conclusion: | $(\exists x)( S(x) \,\&\, \neg P(x) )$ |

*Camestres*

| | |
|---|---|
| Premise: | $(\forall x)( P(x) \rightarrow M(x) )$ |
| Premise: | $(\forall x)( S(x) \rightarrow \neg M(x) )$ |
| Conclusion: | $(\forall x)( S(x) \rightarrow \neg P(x) )$ |

## *The history of logic*

Aristotle (384 - 322 BC) gave the first extensive treatment of logic that has survived. A book called the *Prior Analytics* discusses syllogisms. Another book, the *Posterior Analytics,* covers the qualities of a demonstration: we should start from known principles, we should not argue in a circle, and we should ideally show why our conclusions are true. Aristotle’s student Theophrastus (c.371 - c.287 BC) added to Aristotle’s work.

There was a parallel school, Stoic logic, whose major figures included Euclid of Megara (c.435 - c.365 BC, not the geometer), Diodorus Cronus (died c.284 BC), Philo the Dialectician (active c.300 BC) and Chrysippus of Soli (c.279 - c.206 BC), but we only have secondhand accounts of their works. They focused on necessity, possibility, conditionals and the meanings of whole propositions.

Logic developed in India from about the sixth century BC onwards. Leading figures were the Buddhists Dignaga (c.480 - 540 AD) and Dharmakirti (seventh century AD). Indian logic influenced George Boole in the nineteenth century.

In the Arabic world, Al-Farabi (c.872 - 950/951) developed Aristotle’s ideas, and Avicenna (Ibn Sina) (c.980 - 1037) went further. He sought to work out a logic that was suitable for scientific research. He also worked on the relationship between language and the world.

In Europe, from the thirteenth century onwards, Aristotle’s logic was studied and developed. One major figure was William of Ockham (c.1288 - c.1348), who formulated some logical laws to do with “and” and “or”.

In the seventeenth century, a lot of earlier work got pulled together in textbooks such as the Port-Royal Logic (1662) by Antoine Arnauld (1612 - 1694) and Pierre Nicole (1625 - 1695). Francis Bacon (1561 - 1626) set out his proposals for scientific method in the *Novum Organum* (1620). The tradition of setting out thoughts in textbook form was taken up by John Stuart Mill (1806 - 1873) with his *A System of Logic* (1843).

The nineteenth and twentieth centuries saw huge progress, involving the replacement of words by symbols. Formal logic came to be seen as a part of mathematics, and some philosophical questions came to be stated much more precisely. These questions included the question of the relationship between the meanings of terms and the items to which they apply, and the question of how to handle talk about things that do not exist. Leading figures were George Boole (1815 - 1864), Gottlob Frege (1848 - 1925), David Hilbert (1862 - 1943), Bertrand Russell (1872 - 1970), Alfred North Whitehead (1861 - 1947) and Kurt Gödel (1906 - 1978).

## *Fallacies*

There are plenty of fallacies, that is, bad ways to argue. They are bad ways to argue because they can either lead us to false conclusions, or lead us to conclusions that happen to be true, but in a way that could just as easily have led us to false conclusions. So we should make sure that we do not use them, and watch out in case other people use them. There is no standardized complete list of fallacies, but here are some of them, with examples.

### Hasty generalization

> Economic policies b, c and d have failed, so all economic policies are useless.

### Not noting exceptions to general rules

> Knocking someone over is unacceptable, so a tackle in a game of rugby is unacceptable.

### Appeal to authority

> Professor P says that this theory is correct, so it is.

### Argumentum ad hominem

> Q argues for that view, but he is vain/self-interested/disgraceful, so that view is false.

### Affirming the consequent

> Vegetarians like carrots. He likes carrots. So he is a vegetarian.

### Denying the antecedent

> Smokers cough. She is not a smoker. So she does not cough.

### Begging the question (petitio principii)

> People should not want to get rich,  because it is bad to want that.

### Assuming causation (post hoc ergo propter hoc)

> I drank whisky and felt ill the next day. So the whisky made me feel ill.

### Straw man

Logicians claim that only speech in the form of logical argument is worthwhile, but that devalues poetry, so logicians talk nonsense. (Logicians do not make any such claim. The straw man is the imagined logician who makes the claim.)

### Equivocation

This is using the same word in different senses, for example the word “man” in the following.

> Rex is a man-eating tiger. A women is not a man. So Rex is not a danger to women.

### Amphiboly

This is using a grammatical structure that creates ambiguity, then drawing a conclusion that relies on just one reading. It is also called amphibology.

Thus if we saw the following sentence at the top of an escalator, we might conclude that people without dogs were not allowed on the escalator.

> Dogs must be carried.

### Composition

It is good for Brighton if I go there tomorrow and spend money there. The same is true of each person. So it would be good for Brighton if we all went there tomorrow. (It would not: the town could not cope with the crowds.)

### Division

The army can hold back the enemy. So each soldier can hold back the enemy.

## *Paradoxes*

A paradox is something that looks as though it cannot be right, but at the same time looks as though we ought to be able to say it. We ought to be able to say it either because it is just a variant on something that we definitely can say, or because it is the conclusion of an argument in which the premises look true, and each step looks perfectly legitimate. But we also feel that it has to be wrong.

Paradoxes are fun. They can also point the way to useful developments in logic. Here is a small selection.

### The liar

> "This sentence is false." If it is true, then it is false. If it is false, then it is true.
>
> One response is to forbid sentences that talk about their own truth or falsity. But we can say "This sentence is true". Why should that be forbidden?

### The liar in two sentences

> "The next sentence is false. The previous sentence is true."
>
> This one does not rely on a sentence talking about its own truth or falsity.

### Euathlus

> Protagoras (c.490 - 420 BC) took on Euathlus as a law student. Euathlus would have to pay the tuition fee if, and only if, he won his first case. He did not pay and did not take on any cases. So Protagoras sued him for the fee. According to Protagoras, Euathlus would have to pay if Protagoras won (because Protagoras would have won), or if Euathlus won (because he would have won his first case). According to Euathlus, he would not have to pay if Protagoras won (because he would have lost his first case) or if he won (because Protagoras would have lost his action for the fee).

### The barber

> A town has only one barber, who is a man. He shaves all the men in the town who do not shave themselves, and only those men. Does he shave himself?

## The Berry paradox

We can name positive integers (1, 2, 3, … ) using a certain number of words in English. For example, 112 takes four words: “a hundred and twelve”. Some numbers take lots of words: 720,450 takes nine words: “seven hundred and twenty thousand, four hundred and fifty”. Eventually, we will need eleven words or more. So we can identify “the smallest positive integer not nameable in under eleven words”. But that is a ten-word name, so we have named it in under eleven words.

One response is to say that we have to specify our naming system. There is the ordinary one, using words like “seven” and “thousand”. Then there is the system where we talk about how many words are needed. And the phrase “the smallest positive integer not nameable in under eleven words” should be treated as meaning “the smallest positive integer not nameable in under eleven words using the ordinary naming system”.

## The sorites paradox (the paradox of the heap)

Three grains of sand do not make a heap. If you have a collection of grains of sand that do not make a heap, adding one more grain will not turn it into a heap. Start with three, and add one at a time. Eventually, you will have a million grains of sand. Can that be a heap?

This paradox leads us to develop ways to handle vague terms. There is no particular number of grains of sand that you need for a heap, but there are still numbers that are definitely heaps (if you gather all the grains together), and numbers that are definitely non-heaps.

## The friendship paradox

Most people find that their friends have more friends than they do themselves.

The reason is that each of us is more likely to be friends with someone who has lots of friends, than with someone who has few friends.

## The infinite hotel

A hotel has an infinite number of rooms, and all of them are occupied. A tour bus with an infinite number of people turns up, but we can fit them all in. The person already in room 1 goes to room 2. The person already in room 2 goes to room 4. The person already in room 3 goes to room 6. And so on. Then all the odd-numbered rooms will be free, and the new guests can occupy them.

The message is that infinite numbers do not always behave like finite numbers. The development of the arithmetic of infinite numbers was the great achievement of Georg Cantor (1845 - 1918).

## *Deduction, induction and abduction*

When we want to find out the truth about something, and we have some evidence to go on, there are several different ways in which we can reason.

### Deduction

This is the style of argument that we cover in these notes. Propositional logic and first-order logic are both methods of deductive reasoning.

#### *Advantages of deduction*

If we have a certain degree of confidence in our premises, we can have exactly the same degree of confidence in our conclusions.

The rules tell us exactly what we can infer, and what we cannot infer. There is no scope for cheating in order to reach the conclusions we would like.

The rules of logic have been studied extensively. Their scope and limits are very well-understood.

#### *Disadvantages of deduction*

Deduction is very rarely enough on its own. We need to go out into the world and gather evidence, then knock it into a sufficiently precise form, before we can start to use deduction.

There are limits to what logical systems can do, even within their own field. We will see one such limit when we look at the work of Kurt Gödel.

## Induction

Using induction, we observe a number of events and draw general conclusions. This does not have to be as crude as counting: “20 per cent of the people I have met are fair haired, so 20 per cent of the population is fair haired”. We can use statistical methods to say how confident we are about our conclusions.

We can also have more confidence in our conclusions if we can see that they fit into some background theory, or make sense in the light of observations of something else. Thus we may have a lot of confidence that all ravens are black if we have seen lots of black ravens. But we can have more confidence, once we understand how the genetics of feather-colours work.

(We must not confuse induction in this sense with mathematical induction, where we establish that the first item in a series has some property, then show that if each one does, the next one does, then conclude that they all do. That is a form of deduction.)

### *Advantages of induction*

It seems to work pretty well.

It is practical. We can only gather a certain amount of evidence. It we want to draw general conclusions, we have to work from that limited evidence.

Indeed, going with the evidence we have is the only rational thing to do, apart from refusing to draw general conclusions. It would be crazy to go against the evidence, and draw conclusions that opposed what it suggested.

We can use mathematically precise statistical methods, which do not allow us to cheat and reach conclusions we would like. We can also use Bayesian logic to update our beliefs in the light  of new evidence in a systematic way (Thomas Bayes, c.1702 - 1761).

### *Disadvantages of induction*

It is not wholly reliable. We thought that all swans were white, because all observed swans were white, until we went to Australia and discovered black swans.

Any claim that since induction has mostly worked so far, we can expect it to go on working, looks as though it is circular. We only have inductive evidence that induction works. We can only conclude that it will go on working, if it is indeed reliable.

We must face up to the “grue” problem, proposed by Nelson Goodman (1906 - 1998). Something is grue if it is green when observed before 1 January 2150, or blue when observed from that date onwards. We have plenty of evidence that all emeralds are green. But this is equally good evidence that all emeralds are grue.

We must face up to the paradox of the ravens, proposed by Carl Hempel (1905 - 1997). “All ravens are black” is logically equivalent to “all non-black things are non-ravens”. So a green apple should be evidence that all ravens are black.

## Abduction

This is inference to the best explanation. We observe something. Then we identify something else that makes the observation very likely, fits with other theories that we want to go on accepting, and is reasonably straightforward.

For example, there was a tree in the park yesterday, but it has vanished today. A good explanation is that human beings came along in the night and cut it down. A bad explanation is that Martians landed in the night and stole it.

To take another example, we can observe the changing positions of planets against the background of the stars. A very good, straightforward, explanation is that the Earth and the other planets orbit the Sun. At least, that is a good explanation if we can also explain why we feel that the Earth is still, and not hurtling through space.

### *Advantages of abduction*

If we require an explanation to be integrated with our existing theories, that forces us to bring our existing knowledge into our reasoning. That should usually (not always) help us to avoid mistakes.

We can use statistical methods to work out which properties vary with other properties. (For example, does crop growth vary most with moisture or with temperature?) This can make abduction more precise.

### *Disadvantages of abduction*

There are always many possible explanations for any observation. How can we choose the best one? And won’t our choice be guided by our existing theories, which could be wrong?

It is not clear what makes an explanation the best one. What do we do if there are rival explanations? Do we accept the one that makes our observations most likely, even if it is complicated, or do we accept one that makes our observations a bit less likely, but that has the advantage of being more straightforward?

We might not have observed the important things. For example, Gregor Mendel (1822 - 1884) cross-bred tall and short pea plants to see what would happen. If he had only looked at the first generation, the child plants, he would have concluded that tallness wipes out shortness (they were all tall). He had to go on to the second generation, the grandchild plants, to see that shortness could re-appear. Only that observation allowed him to lay the foundations for genetics.

# *Theories, models, soundness and completeness*

## Theories and models

We defined theories when looking at propositional logic. The definition applies to first-order logic, and to other types of logic, too.

A theory is a set of wffs. The theories that we normally find interesting are ones that include all, and only, the following wffs.

> A few wffs that we call axioms. We just accept these as starting-points in arguments.
>
> All the other wffs that follow from these axioms. We call these theorems. The axioms also count as theorems.

We have looked at theories in which we have some logical axioms that always apply (although what we actually stated were axiom schemata). But we can go further, and add some non-logical axioms for a specific purpose.

Suppose, for example, that we want axioms that reflect interpretation 1 of our theory with four constants, a, b, c and d. This is the interpretation in which “a” stands for Albert, in which Albert and Beatrice are foxes, in which Albert loves everybody, and so on. We can use the following set of non-logical axioms. This is not the only possible set, but it will do.

$(\forall x)(\,(\,(\,(x = a) \vee (x = b)\,) \vee (x = c)\,) \vee (x = d)\,)$

$F(a), F(b), \neg F(c), \neg F(d), G(a), \neg G(b), \neg G(c), G(d)$

$\neg H(a), \neg H(b), H(c), \neg H(d), \neg K(a), \neg K(b), \neg K(c), K(d)$

$L\,(d, a)$

$(\forall y)\,(\,L(a, y)\,)$

$(\forall x)(\forall y)\,(\,L(x, y) \rightarrow (\,(x = a) \vee (\,(\,x = d\,) \,\&\, (\,y = a)\,)\,)\,)$

$(\forall x)(\,(\,H(x) \rightarrow M(x)\,) \,\&\, (\,M(x) \rightarrow H(x)\,)\,)$

$T\,(b, c, a)$

$(\forall x)(\forall y)(\forall z)\,(\,T(x, y, z) \rightarrow (\,(\,(x = b) \,\&\, (\,y = c\,)\,) \,\&\, (\,z = a)\,)\,)$

On the first line, we state that there no more than four objects. On the second and third lines, we have written eight axioms on each line, separated by commas. These axioms allocate properties. They are also enough to ensure that there are at least four different objects.

We need all three of the axioms about L (lines 4, 5 and 6). The first two tell us where L applies, and the third one tells us that those are the only places where it applies. The same goes for the two axioms about T.

Now we can think about what is going on here.

We have a set of axioms, the logical axioms that we get from the basic axiom schemata of first-order logic with equality plus the non-logical axioms. That set, plus everything that follows from its members using rules, constitutes a theory. It is not, in itself, about anything in particular.

We have a population of real animals, Albert, Beatrice, Charlie and Deborah, with their various properties and relations as given under interpretation 1.

This population, with those properties and relations, is a model for the theory. It fits the theory. That is, if we interpret any theorem in the theory, using interpretation 1, we get a statement that is true in the model.

But this population is not the only model for the theory. We could have a different population, say four people. We could re-interpret F, H and K to refer to properties of people, for example “is fast”, “is handsome” and “is knowledgeable”. (We could re-interpret G, L, M and T too, but we would not have to: people can be greedy, love, be muddy and make toast.) Then if the people had the properties and relations that interpretation 1 said they had, the population would be a model for the theory.

However, not just any population would do as a model. A population of five objects would not do for our theory, because that would violate the first axiom that we added to the logical axiom schemata. And a population of four objects with the properties and relations that interpretation 2 gave would not do, because the axiom “T (b, c, a)”, and some other axioms, would be violated.

So we can expect there to be lots of models for each theory. A model will fit a theory so long as it has the right number of objects, with the right pattern of properties and relations. One image that may help is to say that a model must have the right shape for a theory. If, for example, a theory requires that precisely three objects share some property or other, there must be a set of precisely three objects in the model that have a property in common. (The technical term for having the same shape is “isomorphic”.)

It is an interesting question, whether all of the models of a given theory have the same shape. For some theories, they do. For others, they do not. Theories for which all models do have the same shape are called categorical theories.

Logicians who are interested in the general properties of logical systems do not, in practice, think of properties like “is greedy”, and then go round looking for populations that fit their theories. They just imagine populations of whatever size they need. They then allocate imaginary properties and relations to objects, in order the get the patterns that interest them.

## Soundness and completeness

Suppose we have a logical system that consists of a language (a set of symbols and rules that determine what is a wff), some axioms and one or more rules. We use it to construct a theory, and the theory has models.

Take any theorem. We would like it to be true in every model that fits the axioms alone. If all theorems are true in all models for the axioms, then we say that the logical system is sound.

Soundness is a very good property for a system to have. If a system is unsound, we can get into trouble. We may pick a model for the axioms, rely on our theorems, and find that we have said something that is not true in the model.

Suppose that some proposition is true in every model that fits the axioms alone. Then we would like that proposition to be a theorem. If all such propositions are theorems, then we say that the logical system is complete.

We sometimes call this semantic completeness, so as to distinguish it from the negation completeness that we will see in the next section.

## *Kurt Gödel*

Kurt Gödel is one of the most significant logicians of all time. He was born in 1906 in Brno, and died in 1978 in Princeton. He studied at the University of Vienna, where he proved his completeness theorem in 1929. He went on to publish his two incompleteness theorems in 1931. He went on to do further work in mathematical logic and in general relativity. Gödel’s work was powerful and subtle. What follows is only a sketch, but it does show why he was so important.

### A logical theory for arithmetic

We can add non-logical axioms to our first-order logical axiom schemata, so as to get a theory that will capture some specific content. Here we will create a theory that will capture the content of arithmetic with the natural numbers (0, 1, 2 and so on) and with addition and multiplication.

Here is a set of axioms that will do the job, when added to our first-order logic with equality. They were devised by Giuseppe Peano (1858 - 1932). The notion of “number” is just taken for granted. One way to read this is to say that we are only interested in things that are numbers. The implication is that we will only allow our variables to range over things that are numbers.

1. There is a number that we will call 0.

2. There is a function s(x), the successor function. It is defined for all numbers, and the successor of a number is a number.

3. $\neg(\exists x)\ (0 = s(x)\ )$.

   This says that 0 does not come after anything, so 0 is the first number.

   It also rules out a set of numbers with only 0 in the set, as its own successor. We must have at least one more number after 0, so that 0 is not its own successor.

4. $(\forall x)(\forall y)(\ (\ s(x) = s(y)\ ) \rightarrow (\ x = y\ )\ )$

   This rules out chains of numbers that come to an end, with the last member being its own successor. If a chain did end, the last number would be its own successor, and the successor of the number before it, so it would be the same as the number before it. We could repeat this, making the chain shorter each time, until we got back to 0 as the only number left in the chain. But axiom 3 rules out that.

   Axiom 4 also rules out side-stream of numbers, starting somewhere other than 0, that feed into the main stream of numbers. The point in the main stream at which a side-stream joined would be the successor of two numbers, so they would be the same, so the side-stream would get sucked into the main stream, bit by bit.

5. If it is true that:

0 has a given property; and

if any number has that property, so does its successor

then every number has that property.

> Axiom 5 is not really an axiom. It is an axiom schema. We can use it to generate specific axioms by dropping in specific properties.

These few axioms, plus our logical axiom schemata for first-order logic, are enough to allow us to generate arithmetic using the numbers 0, 1, 2 and so on, and addition and multiplication.

We generate the numbers as follows. 1 is s(0), 2 is s(1), and so on.

We deal with addition as follows.

We first define x + 0 as x, and x + s(y) as s (x + y).

Suppose we want to work out 4 + 3.

3 = s(2), so we want s (4 + 2).

But 2 = s(1), so we want s (s (4 + 1) ).

But 1 = s(0), so we want s ( s (s (4) ) ).

To get s (s (s (4) ) ), we can count three steps on from 4, to get 7.

Multiplication is defined as follows.

z x 0 = 0

z x s(y) = z x y + z

So, for example, 4 x 0 = 0, 4 x 1 = 0 + 4, and 4 x 2 = (0 + 4) + 4.

**Gödel’s first incompleteness theorem**

Set up a logical system that can formulate arithmetic (the numbers 0, 1, 2, … , and addition and multiplication). We can use the Peano axioms, or something else. But whatever we use, it must be powerful enough to do this job.

Assume that there is a mechanical procedure that we can use on any wff, to establish, after a finite number of steps, either “it is an axiom” or “it is not an axiom”. This is easy if we have a finite list of axioms. It may be possible if there are infinitely many axioms, but we must check whether it is possible.

Assume that the system is consistent. That is, it does not allow us to prove contradictions.

Then we can produce a sentence G in the system which is such that:

> there is no proof of G within the system;
> there is no proof of not G within the system.
>
> Gödel showed how to construct G: “No proof of G in the specified system exists”.
>
> > We would have a contradiction if we could prove G.
> >
> > We cannot prove not G, because:
> >
> > > if we could, we would be able to prove: not (G has no proof)
> > >
> > > so we would be able to prove: G has a proof
> > >
> > > but if G has a proof, we have a contradiction
> > >
> > > so to avoid inconsistency, we must not be able to prove that G has a proof
> > >
> > > so we must not be able to prove not G.

So the system we started with is negation-incomplete. That is, there are some sentences which are such that we cannot prove or disprove them.

> Negation-completeness is not the same as semantic completeness. Negation-completeness and negation-incompleteness are properties of logical systems. We define them without mentioning models at all.

So what? Construct a more powerful system, which can prove more. This will not, however, solve the problem. If the system meets the conditions (it is powerful enough, we can identify the axioms and the non-axioms, and it is consistent), it will have its sentence G. And there would be a contradiction if we could prove G within the system, or if we could prove not G within the system.

### What Gödel’s first incompleteness theorem shows

It is not surprising that a system can include sentences that it cannot prove and cannot disprove. We can easily construct a system that does not have enough axioms to do the jobs we want. But what is surprising is that we go on having the problem, however much we strengthen the system by adding axioms.

The theorem can be used to show that there are things that are true in arithmetic, but that we cannot prove.

These unprovable truths are truths in our standard model of the axioms of arithmetic, with a single sequence 0, 1, 2, … . But there are some non-standard models of the first-order axioms of arithmetic, in which these unprovable truths are in fact false. The first-order axioms of arithmetic give us a system in which not all models of the theory have the same shape: the theory is not categorical.

Non-standard models have the usual sequence, 0, 1, 2, … , and then an infinite number of extra blocks of numbers, with each block being like a complete run through all the integers, negative, then 0, then positive.

### What Gödel’s first incompleteness theorem does not show

The theorem is easily misinterpreted. People use it to try to prove all sorts of things. So be warned, the theorem does not show:

> that there is something wrong with arithmetic;<br>
> that there are things that we cannot do, but that superhuman intelligences could do;<br>
> that the human mind is not a computer;<br>
> that we have free will;<br>
> that God exists.

## Gödel’s second incompleteness theorem

We can establish that a logical system like the one we have been discussing is powerful enough to formulate arithmetic, and that we could identify its axioms mechanically. Then we are left with consistency.

> If our logical system is consistent, then G has no proof.
>
> But G is “G has no proof”.
>
> So if our logical system is consistent, then G.
>
> This means that if the system could prove that it was consistent, it could prove G.
>
> But we know that if it is consistent, it cannot prove G.
>
> So if the system is consistent, it cannot prove that it is consistent.

This is Gödel’s second incompleteness theorem. Our system cannot prove its own consistency.

Any proof by a system of its own consistency would not be worth much anyway. An inconsistent system can prove anything, including its own consistency, because we can derive anything we like from a contradiction.

## Gödel’s completeness theorem

If a formula is true in every model of a first-order logical system, then there is a proof of it within that system.

We can relate this back to the point above, about truths of arithmetic that cannot be proved. Such unprovable truths only exist because they come out false in some non-standard models of the first-order logical system that we use for arithmetic. If they came out true in all models, they would be provable. (Note that this theorem only holds for first-order logical systems.)