Informatics 1 – Introduction to Computation
Computation and Logic

Julian Bradfield
based on materials by
Michael P. Fourman

Logic and Binary Data

Logic – the language of reasoning

Reasoning, even talking, about the world is hard:

Logic – the language of reasoning

Reasoning, even talking, about the world is hard:

- Nature is fuzzy: we try to classify (species, star types, languages) but categories are not sharp.
  - fuzzy logic

Logic – the language of reasoning

Reasoning, even talking, about the world is hard:

- Nature is fuzzy: we try to classify (species, star types, languages) but categories are not sharp.
- There's a lot of chance in the world.

- fuzzy logic
- probabilistic logic

Logic – the language of reasoning

Reasoning, even talking, about the world is hard:

- Nature is fuzzy: we try to classify (species, star types, languages) but categories are not sharp.
- There's a lot of chance in the world.
- We don't always have all the information.

- fuzzy logic
- probabilistic logic
- logics of imperfect information

Logic – the language of reasoning

Reasoning, even talking, about the world is hard:

- Nature is fuzzy: we try to classify (species, star types, languages) but categories are not sharp.
- There's a lot of chance in the world.
- We don't always have all the information.
- The calculations are too hard.

- fuzzy logic
- probabilistic logic
- logics of imperfect information
- (numerical or logical)

Logic – the language of reasoning

Reasoning, even talking, about the world is hard:

- Nature is fuzzy: we try to classify (species, star types, languages) but categories are not sharp.
- There's a lot of chance in the world.
- We don't always have all the information.
- The calculations are too hard.
- We might need to reason about what people know, or believe, or feel.

- fuzzy logic
- probabilistic logic
- logics of imperfect information
- (numerical or logical)
- epistemic logic

Logic – the language of reasoning

Reasoning, even talking, about the world is hard:

- Nature is fuzzy: we try to classify (species, star types, languages) but categories are not sharp.
- There's a lot of chance in the world.
- We don't always have all the information.
- The calculations are too hard.
- We might need to reason about what people know, or believe, or feel.

In this course, we sweep all that under the carpet, and think only about sharp, certain, and apparently simple statements.

How can we simplify the world?

- fuzzy logic
- probabilistic logic
- logics of imperfect information
- (numerical or logical)
- epistemic logic

Information?

Informatics is ‘the study of systems that store, process, and communicate information’.

What is **information?**

The OED says (among many sub-definitions):

> *Knowledge communicated concerning some particular fact, subject, or event; that of which one is apprised or told; intelligence, news.*

Earliest use of ‘information’ in OED is in Scots in 1390: *Robert..through his wrang informatioune has gert skaith the said abbot.*

Too much information

Part of terms of use of ACX (Audible's audiobook networking site):

Examples of the *information* we collect and analyze include the Internet protocol (IP) address used to connect your computer to the Internet; login; e-mail address; password; computer and connection information such as browser type, version, and time zone setting, browser plug-in types and versions, operating system, and platform; the full Uniform Resource Locator (URL) clickstream to, through, and from our Web site, including date and time; cookie number; products and services you viewed or searched for; and the phone number you used to call our 800 number. We may also use browser data such as cookies, Flash cookies (also known as Flash Local Shared Objects), or similar data on certain parts of our Web site for fraud prevention and other purposes. During some visits we may use software tools such as JavaScript to measure and collect session information, including page response times, download errors, length of visits to certain pages, page interaction information (such as scrolling, clicks, and mouse-overs), and methods used to browse away from the page.

Several hundred million emails are sent every minute. Five hundred hours of video are uploaded to Youtube every minute.

Keep It Simple, S——!

The KISS principle is that simplicity is a key design goal to build working (and repairable) systems.

KISS is a good principle in maths as well as engineering!

To control the complexity of 'information' we assume:

KISS was coined by Kelly Johnson, lead engineer at the Lockheed Skunkworks, in 1960.

Keep It Simple, S——!

The KISS principle is that simplicity is a key design goal to build working (and repairable) systems.

KISS is a good principle in maths as well as engineering!

To control the complexity of 'information' we assume:

- Each observation/sensor/question always gives an answer
- For each observation/sensor/question there are only finitely many possible answers
- In the simplest case for each observation/sensor/question there are only two possible answers

KISS was coined by Kelly Johnson, lead engineer at the Lockheed Skunkworks, in 1960.


$\top$ 1

$\bot$ 0

Keep It Simple, S——!

The KISS principle is that simplicity is a key design goal to build working (and repairable) systems.

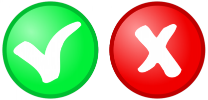
KISS is a good principle in maths as well as engineering!

To control the complexity of ‘information’ we assume:

- Each observation/sensor/question always gives an answer
- For each observation/sensor/question there are only finitely many possible answers
- In the simplest case for each observation/sensor/question there are only two possible answers

This is how we arrive at **Binary Data**.

KISS was coined by Kelly Johnson, lead engineer at the Lockheed Skunkworks, in 1960.



$\top$ 1

$\bot$ 0

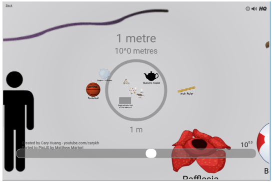
Universes

Our general setting for thinking about logic and computation is a universe:

- A universe is a finite **set** of things.
- We don't care what *things* are – we just need names for them.

If you haven't seen this wonderful visualization, check it out:

https://htwins.net/scale2/



Universes

Our general setting for thinking about logic and computation is a universe:

- A universe is a finite **set** of things.
- We don't care what *things* are – we just need names for them.

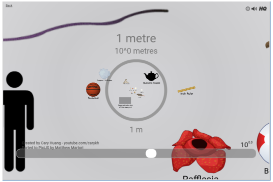
A universe could be tiny, or huge:

- $\{\top, \bot\}$
- all the people in the world
- my emails to the class

We will study binary (yes/no) questions about universes.

If you haven't seen this wonderful visualization, check it out:

`https://htwins.net/scale2/`



A universe of chess pieces

♖ ♘ ♗ ♕ ♔ ♗ ♘ ♖

♙ ♙ ♙ ♙ ♙ ♙ ♙ ♙

Ignoring the colour, there are six kinds of chess piece.

‘*What kind of piece is that?*’ has 6 answers.

As in the game ‘Twenty Questions’, we reduce the question to a series of yes/no questions.

If you are a chess player, the following questions will seem natural. If you are not a chess player, what questions seem natural to you?

Question 1

♙ ♙ ♙ ♙ ♙ ♙ ♙ ♙

Is it a pawn or not a pawn?

♖ ♘ ♗ ♕ ♔ ♗ ♘ ♖

‘pawn’ derives from an Old French word for pedestrian, foot-soldier. (Compare Spanish ‘peón’.)

Question 2

♘ ♗ ♗ ♘

Is it minor or major?

♖ ♕ ♔ ♖

We're cheating, because in real chess terminology, ♔ is neither major nor minor.

Question 3(1,2)

If it is minor,

♘ ♘

Is it a knight or a bishop?

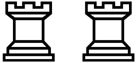
♗ ♗

Question 3(1,2)

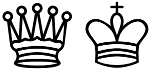
If it is minor,

Is it a knight or a bishop?

If it is major,



Is it a rook or a royal?



Question 3(1,2)

If it is minor,

♘ ♘

Is it a knight or a bishop?

♗ ♗

If it is major,

♖ ♖

Is it a rook or a royal?

♕ ♔

Question 4

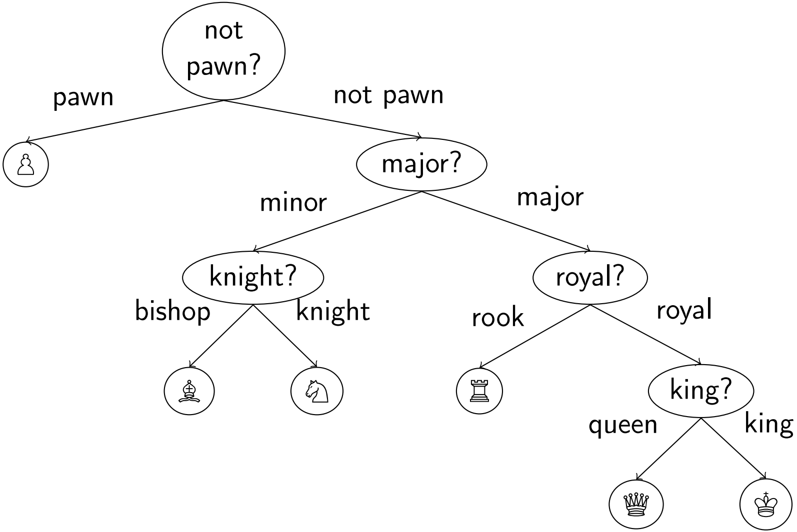
If it's a royal,

♕

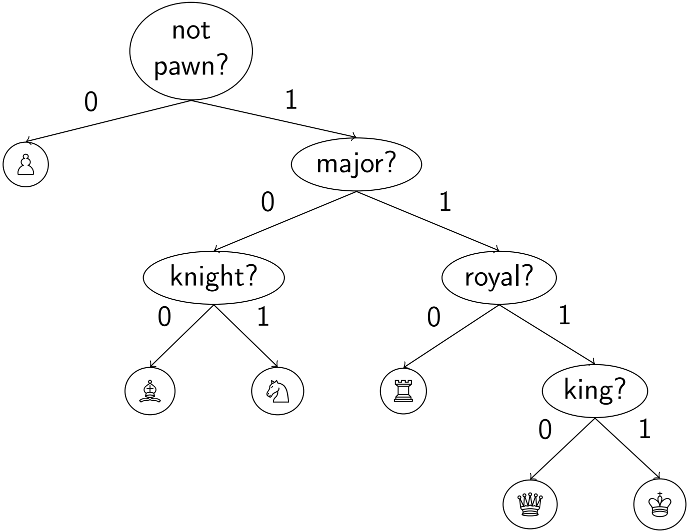
Is it a queen or a king?

♔

Decision Tree



Decision Tree

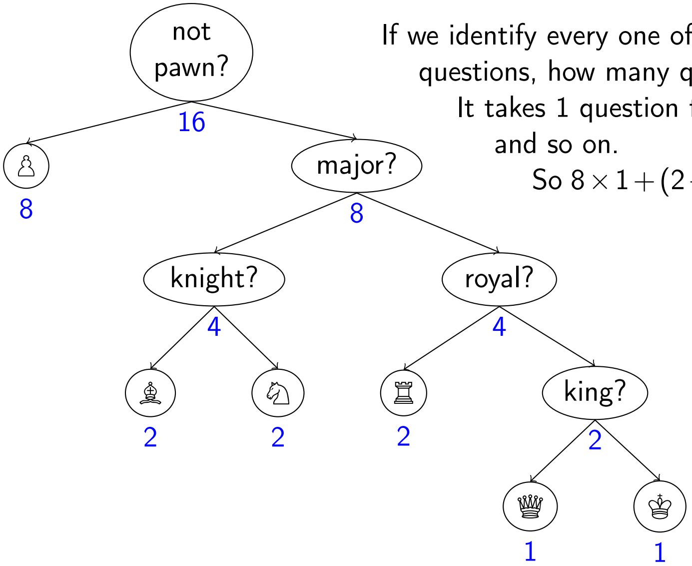


Binary encoding of piece types:

♙	0
♗	100
♘	101
♖	110
♕	1110
♔	1111

This is a *variable-length* encoding: 0 rather than 0000.

How many questions?

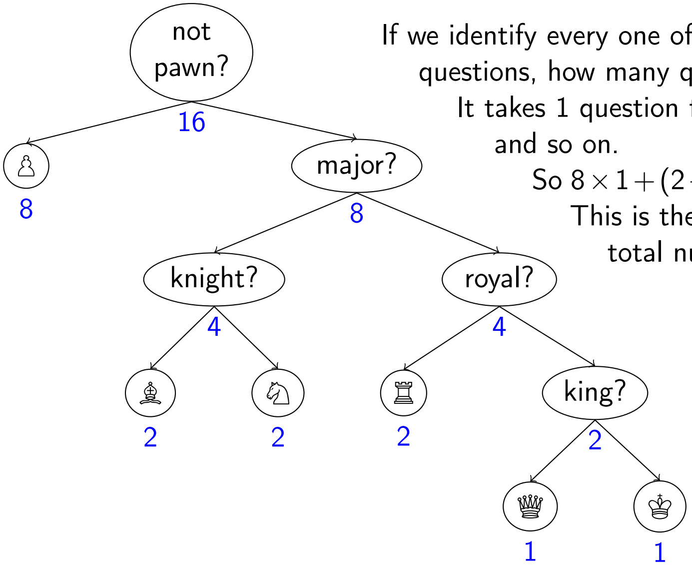


If we identify every one of the 16 pieces by these questions, how many questions do we ask?

It takes 1 question for a pawn, 3 for a knight, and so on.

So $8 \times 1 + (2+2+2) \times 3 + (1+1) \times 4 = 34$.

How many questions?



If we identify every one of the 16 pieces by these questions, how many questions do we ask?

It takes 1 question for a pawn, 3 for a knight, and so on.

So $8 \times 1 + (2+2+2) \times 3 + (1+1) \times 4 = 34.$

This is the total number of **bits** in our encoding:

$8 \times$	♙	0
$2 \times$	♗	100
$2 \times$	♘	101
$2 \times$	♖	110
$1 \times$	♕	1110
$1 \times$	♔	1111

From encodings to decision trees

Here is another encoding:

♙	000
♘	001
♗	010
♖	100
♕	110
♔	111

From encodings to decision trees

Here is another encoding:

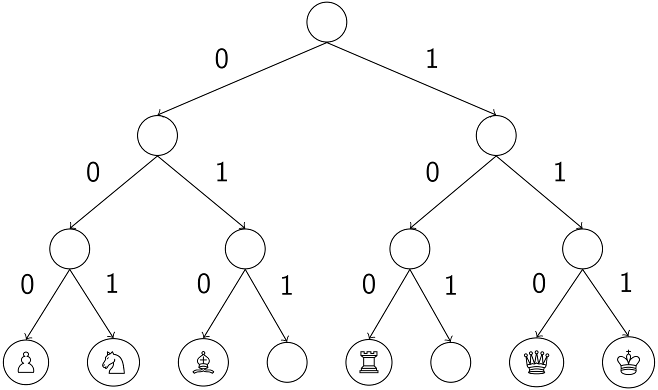
♙	000	Here each piece
♘	001	needs 3
♗	010	bits/questions.
♖	100	What are the
♕	110	questions that
♔	111	produce it?

From encodings to decision trees

Here is another encoding:

♙	000
♘	001
♗	010
♖	100
♕	110
♔	111

Here each piece needs 3 bits/questions. What are the questions that produce it?

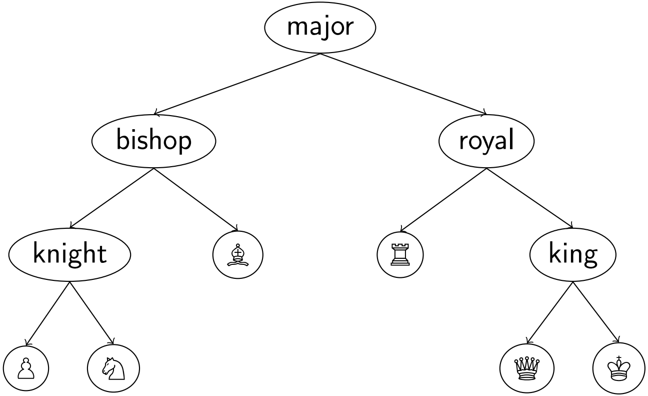


From encodings to decision trees

Here is another encoding:

♙	000
♘	001
♗	010
♖	100
♕	110
♔	111

Here each piece needs 3 bits/questions. What are the questions that produce it?



How many questions to identify all pieces? You should count 44.
But our representation uses 48 bits for all pieces.

Bits

Our chess piece encodings used 1 to 4 bits (variable), or 3 bits (fixed) to encode 6 types.

Mark the following notational convention (used by computer scientists):

$\log n$ means $\log_{10} n$

$\ln n$ means $\log_e n$

$\lg n$ means $\log_2 n$

Bits

Our chess piece encodings used 1 to 4 bits (variable), or 3 bits (fixed) to encode 6 types.

In general, with m bits we can encode 2^m values.

To encode n values, we need $\lceil \lg n \rceil$ bits.

How many different 3-bit encodings of 6 values are there?

Mark the following notational convention (used by computer scientists):

$\log n$ means $\log_{10} n$

$\ln n$ means $\log_e n$

$\lg n$ means $\log_2 n$

Bits

Our chess piece encodings used 1 to 4 bits (variable), or 3 bits (fixed) to encode 6 types.

In general, with m bits we can encode 2^m values.

To encode n values, we need $\lceil \lg n \rceil$ bits.

How many different 3-bit encodings of 6 values are there?

Exercise: in theory, how many possible 1-hour HD digital movies are there? Do a bit of calculation, come up with some answers, and discuss with your colleagues in the tutorial next week.

Mark the following notational convention (used by computer scientists):

$\log n$ means $\log_{10} n$

$\ln n$ means $\log_e n$

$\lg n$ means $\log_2 n$

Logic and Language

Normal human language is often ambiguous, imprecise, or verbose. Even when people try very hard not to be – which is why lawyers exist!

Informatics has mathematics and logic as its foundation: this both enables and requires clear, precise, and concise communication.

We now turn to logic as a language to achieve such communication.

‘Logic’ is from the Greek λόγος (logos) ‘word, oration, reasoning, reason’. It’s short for ἡ λογικὴ τέχνη (hē logikē tekhnē) ‘the art of reasoning’.

Propositional/Boolean Logic

We've seen how to reduce the description and classification of things to yes/no questions. Every question comes from a statement.

Propositional/Boolean Logic

We've seen how to reduce the description and classification of things to yes/no questions. Every question comes from a statement.

A proposition is a simple statement that is either true or false:

- 'the moon is round'
- 'it is raining (here and now)'
- 'I like mooncakes'
- 'that book is yellow'

These 'simple' statements contain a lot of complexity. What is 'the moon'? What does 'round' mean? Where is 'here and now'? Who is 'I'? Which book? But the complexity is not *logical*.

Propositional/Boolean Logic

We've seen how to reduce the description and classification of things to yes/no questions. Every question comes from a statement.

A proposition is a simple statement that is either true or false:

- 'the moon is round'
- 'it is raining (here and now)'
- 'I like mooncakes'
- 'that book is yellow'

We'll use letters such as $P, Q, \ldots$ to stand for propositions.

These 'simple' statements contain a lot of complexity. What is 'the moon'? What does 'round' mean? Where is 'here and now'? Who is 'I'? Which book? But the complexity is not *logical*.

Combining propositions

We can combine propositions to form compound propositions.

- 'and'. The 'and' ('conjunction') of P and Q is true exactly when both P and Q are true.

Combining propositions

We can combine propositions to form compound propositions.

- ‘and’. The ‘and’ (‘conjunction’) of P and Q is true exactly when both P and Q are true.
  There are many symbols: $P \wedge Q$, $P \,\&\, Q$, $P \cdot Q$ and others. We use $P \wedge Q$.

Combining propositions

We can combine propositions to form compound propositions.

- 'and'. The 'and' ('conjunction') of P and Q is true exactly when both P and Q are true.
  There are many symbols: $P \wedge Q$, $P \,\&\, Q$, $P \cdot Q$ and others. We use $P \wedge Q$.
  We can write a *truth table* to show how $\wedge$ works:

		Q	
	$\wedge$	F	T
P	F	F	F
	T	F	T

Combining propositions

We can combine propositions to form compound propositions.

- ‘or’. The ‘or’ (‘disjunction’) of P and Q is true exactly when at least one of P and Q is true.

Combining propositions

We can combine propositions to form compound propositions.

- 'or'. The 'or' ('disjunction') of P and Q is true exactly when at least one of P and Q is true.
  There are many symbols: $P \vee Q$, $P \mid Q$, $P + Q$ and others. We use $P \vee Q$.

Combining propositions

We can combine propositions to form compound propositions.

- 'or'. The 'or' ('disjunction') of P and Q is true exactly when at least one of P and Q is true.
  There are many symbols: $P \vee Q$, $P \mid Q$, $P + Q$ and others. We use $P \vee Q$.
  We can write a *truth table* to show how $\vee$ works:

	$\vee$	F (Q)	T (Q)
P	F	F	T
P	T	T	T

Combining propositions

We can combine propositions to form compound propositions.

- 'or'. The 'or' ('disjunction') of P and Q is true exactly when at least one of P and Q is true.
  There are many symbols: $P \vee Q$, $P \mid Q$, $P + Q$ and others. We use $P \vee Q$.
  We can write a *truth table* to show how $\vee$ works:

	$\vee$	F (Q)	T (Q)
P	F	F	T
P	T	T	T

	$\wedge$	F (Q)	T (Q)
P	F	F	F
P	T	F	T

Exercise: Compare the truth tables for $\wedge$ and $\vee$. What do you observe about them?

Combining propositions

We can combine propositions to form compound propositions.

- ‘not’. The ‘not’ (‘negation’) of P is true exactly when P is false.

Combining propositions

We can combine propositions to form compound propositions.

- 'not'. The 'not' ('negation') of P is true exactly when P is false.
  There are many symbols: $\neg P, \sim P, \overline{P}$ and others. We use $\neg P$.

Combining propositions

We can combine propositions to form compound propositions.

- 'not'. The 'not' ('negation') of P is true exactly when P is false.
  There are many symbols: $\neg P, \sim P, \overline{P}$ and others. We use $\neg P$.
  We can write a *truth table* to show how $\neg$ works:

P		
$\neg$	F	T
	T	F

Combining propositions

We can combine propositions to form compound propositions.

- 'not'. The 'not' ('negation') of P is true exactly when P is false.
  There are many symbols: $\neg P, \sim P, \overline{P}$ and others. We use $\neg P$.
  We can write a *truth table* to show how $\neg$ works:

	P	
$\neg$	F	T
	T	F

We can build up complex propositions:

$$(P \wedge Q) \vee (\neg(R \wedge S))$$

using parentheses in the usual mathematical way.

Combining propositions

$\wedge, \vee, \neg$ are enough for all possible combinations (check for yourself!). But we use one combination a lot.

- 'if-then'. The 'if-then' ('implication') of P and Q is true exactly if whenever P is true then Q is true.

Combining propositions

$\wedge, \vee, \neg$ are enough for all possible combinations (check for yourself!). But we use one combination a lot.

- 'if-then'. The 'if-then' ('implication') of P and Q is true exactly if whenever P is true then Q is true.
  There are many symbols: $P \rightarrow Q$, $P \Rightarrow Q$, $P \supset Q$ and others.
  We use $P \rightarrow Q$.

Combining propositions

$\wedge, \vee, \neg$ are enough for all possible combinations (check for yourself!). But we use one combination a lot.

- 'if-then'. The 'if-then' ('implication') of P and Q is true exactly if whenever P is true then Q is true.
  There are many symbols: $P \rightarrow Q$, $P \Rightarrow Q$, $P \supset Q$ and others.
  We use $P \rightarrow Q$.
  We can write a *truth table* to show how $\rightarrow$ works:

		Q	
	$\rightarrow$	F	T
P	F	T	T
	T	F	T

Combining propositions

$\wedge, \vee, \neg$ are enough for all possible combinations (check for yourself!). But we use one combination a lot.

- 'if-then'. The 'if-then' ('implication') of P and Q is true exactly if whenever P is true then Q is true.
  There are many symbols: $P \to Q$, $P \Rightarrow Q$, $P \supset Q$ and others.
  We use $P \to Q$.
  We can write a *truth table* to show how $\to$ works:

		Q	
	$\to$	F	T
P	F	T	T
	T	F	T

Think carefully about the first row . . .

Combining propositions

$\wedge, \vee, \neg$ are enough for all possible combinations (check for yourself!). But we use one combination a lot.

- 'if-then'. The 'if-then' ('implication') of P and Q is true exactly if whenever P is true then Q is true.
  There are many symbols: $P \rightarrow Q$, $P \Rightarrow Q$, $P \supset Q$ and others.
  We use $P \rightarrow Q$.
  We can write a *truth table* to show how $\rightarrow$ works:

		Q	
	$\rightarrow$	F	T
P	F	T	T
	T	F	T

Note that false implies anything! Later we'll see that this is true in proofs, too: *ex falsum quodlibet*

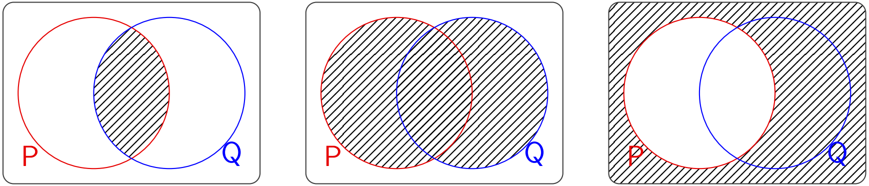
Think carefully about the first row ...

$P \rightarrow Q$ is the same as $Q \vee \neg P$.

Venn diagrams

You (should) know Venn diagrams. Our boolean combinators are just like set-theoretic combinators:

$P \wedge Q$ $\qquad$ $P \vee Q$ $\qquad$ $\neg P$



$P \cap Q$ $\qquad$ $P \cup Q$ $\qquad$ $\overline{P}$

This is not, of course, a coincidence.

When we're being really precise, we define the *meaning of P* to be $\|P\|$, the set $\{x : P(x)\}$, and then we define the meaning of $\wedge$ by $\|P \wedge Q\| = \|P\| \cap \|Q\|$.

Predicates

A proposition is just true or false, on its own.

A predicate is a proposition *about* things:

- The moon is round
- The sun is round
- I like mooncakes

Predicates

A proposition is just true or false, on its own.

A predicate is a proposition *about* things:

- The moon is round
- The sun is round
- I like mooncakes

Individual things have names, and we use variables $x, y, \ldots$ to represent arbitary things.

We represent predicates by $P, Q, \ldots$ as well, but apply them to arguments:

- $P(x)$: the (unary) predicate P is true of x
- $Likes(x, y)$: the (binary) predicate *Likes* is true of x, y.

Predicates

A proposition is just true or false, on its own.

A predicate is a proposition *about* things:

- The moon is round
- The sun is round
- I like mooncakes

Individual things have names, and we use variables $x, y, \ldots$ to represent arbitary things.

We represent predicates by $P, Q, \ldots$ as well, but apply them to arguments:

- $P(x)$: the (unary) predicate P is true of x
- $Likes(x, y)$: the (binary) predicate $Likes$ is true of x, y.

A special binary predicate is equality, which we write $x = y$.

First-order logic, or predicate logic

The usual next step beyond propositional logic is first-order logic.

First-order logic, or predicate logic

The usual next step beyond propositional logic is first-order logic.
This lets us make statements about ‘all’ or ‘some’ of something:

- $\forall x.P(x)$ (‘for all x, $P(x)$’): P is true about x whatever x is
- $\exists x.P(x)$ (‘there exists x such that $P(x)$’): there is some x of which P is true

First-order logic, or predicate logic

The usual next step beyond propositional logic is first-order logic. This lets us make statements about 'all' or 'some' of something:

- $\forall x.P(x)$ ('for all x, $P(x)$'): P is true about x whatever x is
- $\exists x.P(x)$ ('there exists x such that $P(x)$'): there is some x of which P is true

Now we can say much more. E.g. you may see the definition of $f : \mathbb{R} \to \mathbb{R}$ being *everywhere continuous* as:

$$\forall x.\forall \epsilon > 0.\exists \delta > 0.\forall x'.(|x' - x| < \delta) \to (|f(x') - f(x)| < \epsilon)$$

First-order logic, or predicate logic

The usual next step beyond propositional logic is first-order logic.
This lets us make statements about 'all' or 'some' of something:

- $\forall x.P(x)$ ('for all x, $P(x)$'): P is true about x whatever x is
- $\exists x.P(x)$ ('there exists x such that $P(x)$'): there is some x of which P is true

Now we can say much more. E.g. you may see the definition of $f : \mathbb{R} \to \mathbb{R}$ being *everywhere continuous* as:

$$\forall x.\forall \epsilon > 0.\exists \delta > 0.\forall x'.(|x' - x| < \delta) \to (|f(x') - f(x)| < \epsilon)$$

First-order logic is hard! In two senses:

- long formulae are hard for humans to understand;
- we cannot work out whether arbitrary formulae are true.

First-order logic, or predicate logic

The usual next step beyond propositional logic is first-order logic. This lets us make statements about 'all' or 'some' of something:

- $\forall x.P(x)$ ('for all x, $P(x)$'): P is true about x whatever x is
- $\exists x.P(x)$ ('there exists x such that $P(x)$'): there is some x of which P is true

Now we can say much more. E.g. you may see the definition of $f : \mathbb{R} \to \mathbb{R}$ being *everywhere continuous* as:

$$\forall x.\forall \epsilon > 0.\exists \delta > 0.\forall x'.(|x' - x| < \delta) \to (|f(x') - f(x)| < \epsilon)$$

First-order logic is hard! In two senses:

- long formulae are hard for humans to understand;
- we cannot work out whether arbitrary formulae are true.

So we're going to start with something a bit easier than FOL.

FOL is the language of mathematics, and of much other reasoning. It was only invented/ discovered 140 years ago. Two millennia earlier ...