Propositional Logic

Main Features

Consider the following example:

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

- Basic sentences (propositional atoms), ‘count>0’, found
  - They are true or false depending on circumstances
- Connectives (and, or, not, etc.)
  - They are used to build more and more complex test sentences from the atoms
- The final complex sentence evaluates to true or false

Propositional logic is not very expressive.

Why do propositional logic?

All logics are based on propositional logic in some form

Syntax

First, we need to fix a precise definition of the formal language, that is the syntax of propositional logic. It has three ingredients:

- Propositional atoms
- Boolean connectives
- Punctuation

Propositional atoms

We are not concerned about which facts are being represented ('$count > 0$', $found$). So long as they can get a truth-value ($true$ or $false$), that's enough. Quotation marks are used to encapsulate a single expression.

Therefore, we fix a collection of algebraic symbols to stand for these statements.

These symbols are called **propositional atoms**. For short, we will call them **atoms**.

They are like variables $x, y, z, \ldots$. But because they are Propositional, we use the letters $p, p', p'' \ldots p_0, p_1, p_2, \ldots$, and also $p, q, r, s, \ldots$

Avoid mixing conventions.

Boolean Connectives

We are interested in the following Boolean connectives (a.k.a. operator or operations):

- not: written as $\neg$
- and: written as $\wedge$
- or: written as $\vee$
- $if - then$, or $implies$: written as $\rightarrow$ (but not as $\implies$)
- $if - and - only - if$: written $\leftrightarrow$ (but not as $\iff$ or $\equiv$)
- $truth$ and $falsity$: written as $\top$, $\bot$

Our test `count>0 and not found` would be expressed as `'count>0'` $\wedge$ `¬found`, or better $p \wedge \neg q$.

Boolean connectives $and, or, implies, if - and - only - if$ take two arguments and are written in infix form

Negation (not) takes one argument, written to the right of it: e.g., $\neg p$

Truth and falsity, $\top$, $\bot$, take no arguments. They are logical constants

Punctuation

Consider the following if-test:

$$\text{`count>0'} \wedge \neg\text{found} \vee \text{`count>10'}$$

We need brackets to disambiguate it.

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

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

and the difference matters.

So we start by **putting all brackets in**, and then **deleting those that are not needed**

Propositional formulas

What we have called if-tests are called **formulas**.

Informally, a **propositional formula** is a string of symbols made from propositional atoms, Boolean connectives, and brackets, in the appropriate way.

Definition 1.1 (Propositional formula)

- Any propositional atom (p, q, r, etc) is a propositional formula.
- $\top$ and $\bot$ are formulas.
- If ϕ is a formula then so is $(\neg\phi)$. *Note the brackets!*
- If ϕ, ψ are formulas then so are $(\phi \wedge \psi)$, $(\phi \vee \psi)$, $(\phi \to \psi)$, and $(\phi \leftrightarrow \psi)$.
- That's it: nothing is a formula unless built by these rules.

The first two in the above are called **atomic formulas**.

Greek letters (ϕ, ψ, ...) are often used as meta-variables, representing formulas. They are not themselves expressions of the language.

Examples

The following are formulas:

- p
- $(\neg p)$
- $((\neg p) \wedge \top)$
- $(\neg((\neg s) \wedge \bot))$
- $((\neg p) \rightarrow (\neg((\neg q \vee r) \wedge \top)))$

The following are not:

- $\wedge\ p\ q$ NO ($\wedge$ takes two arguments on either side.)
- $(\neg\bot) \wedge (r \rightarrow q)$ NO (missing brackets)
- $(p\ \vee q\ \neg s)$ NO
- $\neg r)$ NO

Take the formula $((\neg p) \rightarrow (\neg((\neg q \vee r) \wedge \top)))$

The brackets are a pain already. We need some conventions to get rid of (some of) them.

What we'll get are **abbreviations** of genuine formulas

We can always omit the final, outermost brackets.

Binding conventions

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

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

But don't take the conventions too far, e.g., $p \to \neg q \vee \neg\neg r \wedge s \leftrightarrow t$ is a mess!

Use brackets if it helps readability, even if they are not strictly needed.

Repeated connectives

What about $p \to q \to r$? The binding conventions are no help now.

Always put brackets in such a formula. However, $p \wedge q \wedge r$ and $p \vee q \vee r$ are OK as they are since their logical meaning is the same, however we bracket them.

$(p \wedge q) \wedge r$ and $p \wedge (q \wedge r)$ are different formulas, but they always have the same truth value in any situation: they are **logically equivalent**.

Usually, $p \wedge q \wedge r$ disambiguates to $(p \wedge q) \wedge r$: the connectives are **left-associative**

Repeated negations, as with $(\neg(\neg(\neg p)))$, can create no ambiguity: their brackets can always be removed, e.g., to $\neg\neg\neg p$

Special Cases

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

For example, if I saw $p \to r \wedge q \to r$, I'd probably read it as $(p \to r) \wedge (q \to r)$

Why? $\wedge$ is stronger than $\to$. Shouldn't it be $p \to (r \wedge q) \to r$?

Read according to a plausible convention about repeated $\to$, this is $(p \to (r \wedge q)) \to r$. But few people would write it in this way.

So $p \to r \wedge q \to r$ seems more likely to mean $(p \to r) \wedge (q \to r)$

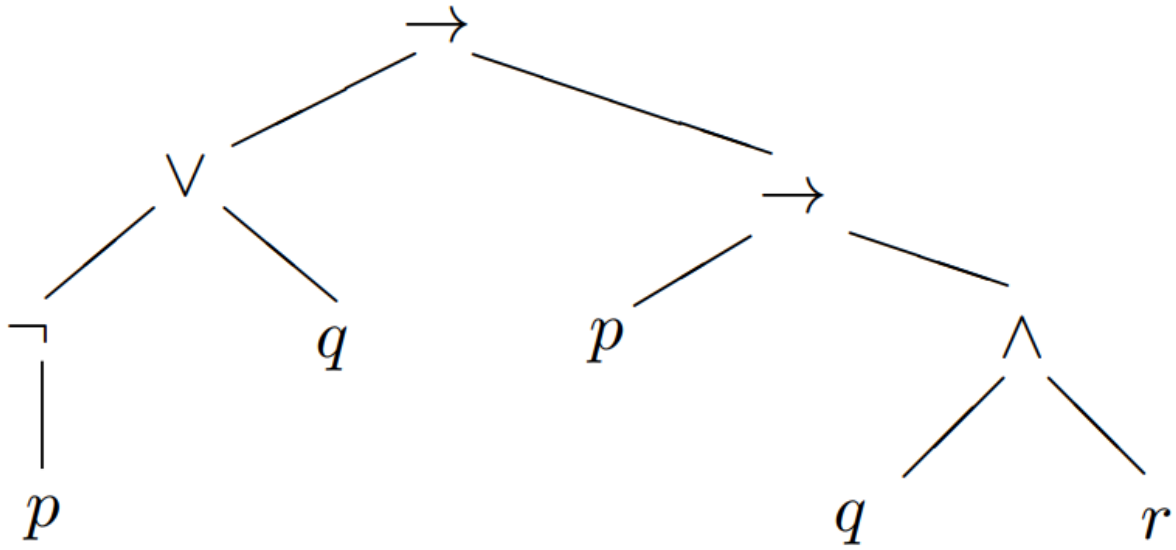
Formulas like $p \to r \wedge q \to r$ can be misinterpreted without brackets, even if they are not strictly needed. **Don't write such formulas without brackets.**

Parsing: formation tree, logical form

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

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

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



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

Principal Connective

The connective at the root of the tree is $\to$. This is the **principal connective** of $\neg p \vee q \to (p \to q \wedge r)$.

This formula has the **overall logical form** $\phi \to \psi$

Every non-atomic formula has a principal connective, which determines its overall logical form.

Subformulas

The subformulas of a formula ϕ are the formulas built in the stages on the way to building ϕ as in Definition 1.1 Propositional Logic > ^definition-1-1

They correspond to the subtrees, of the formation tree of ϕ. The subformulas of $\neg p \vee q \to (p \to q \wedge r)$ are:

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

$$\neg p \lor q \qquad\qquad p \to q \land r$$

$$\neg p \qquad q \qquad p \qquad q \land r$$

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

There are two different subformulas p. And $p \lor q$ and $p \to q$ are substrings but not subformulas

Abbreviations

- $\bigwedge_{1 \le i \le n} \phi_i, \quad \bigwedge_{i=1}^{n} \phi_i,$
  These all abbreviate $\phi_1 \land \phi_2 \land \ldots \land \phi_n$.
  Example: $\bigwedge_{1 \le i \le n} p_i \to q$ abbreviates $p_1 \land \ldots \land p_n \to q$.
- $\bigvee_{1 \le i \le n} \phi_i, \quad \bigvee_{i=1}^{n} \phi_i, \quad \bigvee_{i=1}^{n} \phi_i, \quad \bigvee_{1 \le i \le n} \phi_i, \quad \bigvee\!\!\bigvee_{1 \le i \le n} \phi_i.$
  These abbreviate $\phi_1 \lor \phi_2 \lor \ldots \lor \phi_n$.

Technical terms for logical forms

Definition 1.2

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

Technical terms Continued — Literals and Clauses

Definition 1.3 (Literals and clauses)

- A formula that is either atomic or negated-atomic is called a *literal*.
- A *clause* is a disjunction ($\vee$) of one or more literals.

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

The following are all clauses:

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

Semantics — The meaning

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

We need a precise idea of the meaning of formulas:

- Especially as there are infinitely many of them
- Because we may want to implement it

In propositional logic, our concern is to study good ('valid') inferences which depend just on:

- The truth or falsity of the propositional atoms
- The logical form in terms of Boolean connectives

Atomic Evaluation Functions

A **situation** is something that determines truth-values, true (tt) or false (ff), to each propositional atom in the language. It is formally given by an **atomic evaluation function**

Definition 1.4 (Atomic evaluation function)

Let $\mathcal{A}$ be a set of propositional atoms. An *atomic evaluation function* $v : \mathcal{A} \to \{\mathrm{tt}, \mathrm{ff}\}$ assigns truth-values to each propositional atom in $\mathcal{A}$.

There is more than one situation. In a different situation, the truth-values may be different

Let v be an atomic evaluation function. The situation in which the atom p is true is one where we would use v with $v(p) = tt$

Evaluation Functions

Knowing the situation, we can work out the truth-value of any given propositional formula:

Definition 1.5 (Evaluation function)

Let $\mathcal{A}$ be a set of propositional atoms, and v an atomic evaluation function for $\mathcal{A}$. The *evaluation function* $|\ldots|_v$ assigns the truth-value *true* (tt) or *false* (ff) to formulas as follows.

- If ϕ is an atom $p \in \mathcal{A}$: $|p|_v = \mathsf{tt}$ *iff* $v(p) = \mathsf{tt}$
- $|\neg\phi|_v = \mathsf{tt}$ *iff* $|\phi|_v = \mathsf{ff}$
- $|\phi \wedge \psi|_v = \mathsf{tt}$ *iff* $|\phi|_v = \mathsf{tt}$ and $|\psi|_v = \mathsf{tt}$
- $|\phi \vee \psi|_v = \mathsf{tt}$ *iff* $|\phi|_v = \mathsf{tt}$ or $|\psi|_v = \mathsf{tt}$
- $|\phi \rightarrow \psi|_v = \mathsf{tt}$ *iff* $|\phi|_v = \mathsf{ff}$ or $|\psi|_v = \mathsf{tt}$
- $|\phi \leftrightarrow \psi|_v = \mathsf{tt}$ *iff* $|\phi|_v = |\psi|_v$
- $|\bot|_v = \mathsf{ff}$
- $|\top|_v = \mathsf{tt}$

A formula is assigned $ff \leftrightarrow$ it is not assigned the truth-value tt

Do not confuse tt, ff with $\top$, $\bot$; the latter are formulas, not truth-values.

We don't ask whether a formula is true. (It's like asking 'is $3x = 7$ true?') We ask if it is true **in a given situation**

Understanding evaluation functions

Note that '$\vee$' means inclusive or: if both ϕ and ψ are true, then so is $\phi \vee \psi$. If you want 'exclusive' or, write: $(\phi \vee \psi) \wedge \neg(\phi \wedge \psi)$

The semantics of '$\rightarrow$' we gave does make $\phi \rightarrow \psi$ true where ϕ is false, regardless of ψ

'$\rightarrow$' can often be used to model if-then claims; but all it asserts is the relationship between the truth-values of p and q as stated in Definition 1.5 Propositional Logic > ^definition-1-5

The evaluation function makes the Boolean connectives **truth functional**: the truth-value of the whole formula is functionally determined by the truth-values of the "connected" subformulas

Truth Tables

Our connectives can also have their semantics given using **truth tables**. These give the same information as Definition 1.5 Propositional Logic > ^definition-1-5, for **every possible** v

Truth tables show how the truth-values of complex formulas depend on the truth-values of their subformulas.

p	$\neg p$
tt	ff
ff	tt

ϕ	ψ	$\phi \wedge \psi$	$\phi \vee \psi$	$\phi \rightarrow \psi$	$\phi \leftrightarrow \psi$
tt	tt	tt	tt	tt	tt
tt	ff	ff	tt	ff	ff
ff	tt	ff	tt	tt	ff
ff	ff	ff	ff	tt	tt

$\top$	$\bot$
tt	ff

Consider the situation where p is true (i.e., $v(p) = tt$) and q is false (i.e., $v(q) = ff$); this fixes what the atomic evaluation v must be.

In this situation, the formula $p \rightarrow q$ is assigned the truth-value ff

p	q	$p \wedge q$	$p \vee q$	$p \rightarrow q$	$p \leftrightarrow q$
.	.	.	.	.	.
tt	ff	.	.	ff	.
.	.	.	.	.	.

Let's revisit Definition 1.5:

$$|p \rightarrow q|_v = \text{tt} \quad \textit{iff} \quad |p|_v = \text{ff} \;\; \textit{or} \;\; |q|_v = \text{tt}$$

$$\textit{iff} \quad \text{tt} = \text{ff} \;\; \textit{or} \;\; \text{ff} = \text{tt} \quad \textit{(in our case)}$$

The RHS is false, so $|p \rightarrow q|_v = \text{ff}$, as the truth table said.

Meanings of other connectives

Any truth table determines a connective. For instance:

ϕ	ψ	$\phi \uparrow \psi$
tt	tt	ff
tt	ff	tt
ff	tt	tt
ff	ff	tt

$\uparrow$, as defined here, is called the **Sheffer stroke**. It has a special property: **all our other Boolean connectives can be defined in terms of it.**

It is often useful to know how many truth-functionally, distinct Boolean connectives of a given arity can be constructed.

There are **two 0-ary connectives**, and **four 1-ary connectives**.

One way to see that there are four 1-ary connectives is to list the two truth assignments for an atom p, tt and ff. Let Θ represent an arbitrary 1-ary connective. It can have any of the following truth tables:

p	Θp
tt	tt
ff	tt

p	Θp
tt	ff
ff	ff

p	Θp
tt	tt
ff	ff

p	Θp
tt	ff
ff	tt

In general, there are 2^{2^n} distinct n-ary Boolean connectives.

Functional Completeness

Definition 1.6 (Functional completeness)

Let $\mathcal{C}$ be a set of Boolean connectives. $\mathcal{C}$ is functionally complete (for propositional logic) if any connective of any arity can be defined just in terms of the connectives in $\mathcal{C}$.

The Sheffer stroke is functionally complete for propositional logic.

The set $\{\neg, \wedge\}$ is functionally complete for propositional logic

$\phi \uparrow \psi$ can be expressed as $\neg(\phi \wedge \psi)$

English Correspondence

Translating from English to logic is difficult. But we have to do it for applications.

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

I am king of the world	('I am king of the world')
I bought milk and cookies	('I bought milk' $\wedge$ 'I bought cookies')
There's no crying in baseball	($\neg$ 'There is crying in baseball')
The answer is yes or no	('The answer is yes' $\vee$ 'the answer is no')

Let us consider the following sentence:

'*The train is delayed and we don't have a car*'

The train is delayed **and** we don't have a car

(The train is delayed **and** we don't have a car)

('The train is delayed' **and** we don't have a car)

('The train is delayed' **and it's not the case that** we have a car)

('The train is delayed' **and** (**it's not the case that** we have a car))

('The train is delayed' **and** (**it's not the case that** 'we have a car'))

Take d to represent '*The train is delayed*' and c for '*we have a car*', then the above English sentence is translated into the following propositional logic formula:

$(d \wedge (\neg c))$

English Variants

- '*But*' means '*and*', as do *yet*, *although*, *though*
  - E.g., *I will go out, but it is raining*
  - Let g be an atom for '*I will go out*' and r for '*it is raining*': We get $(g \wedge r)$
- '*Unless*' generally means '*or*'
  - E.g., *I will go out unless it rains*
  - Let g be an atom for '*I will go out*' and w for '*it will rain*' (Note the extra 'will'.)
  - We get $(g \vee w)$. You could also use $((\neg w) \rightarrow g)$

But you may think that '*I will go out unless it rains*' implies that *if it does rain then I won't go out*. This is a stronger reading of '*unless*'

- Strong '*Unless*' (also called "exclusive or")
  - '*I will go out unless it rains*' becomes $(g \leftrightarrow (\neg w))$

– *'Only if'*

You will pass only if your average is at least forty	('You will pass' → 'your average is at least forty')

– *'Is necessary for'*

James' attending the course is necessary for his obtaining a certificate of attendance	('James obtains a certificate of attendance' → 'James attends the course')

– *'Is sufficient for'*

Layla's timely arrival for rehearsals is sufficient for her taking part in the play	('Layla arrives on time for rehearsals' → 'Layla will take part in the play')

Other variants: *provided that*, *on the condition that*, *in case*, ...

Use symbols for propositional atoms rather than strings in quotes

From Logic to English

Let p be '*It is raining*' and q be '*I go out*',

- $p \wedge q$ translates to *It is raining and I go out*
- $\neg p$ translates to *It is not the case that it is raining*
- $p \rightarrow q$ translates to *If it is raining then I go out*

Complex formulas are hard to render naturally in English.

'$\rightarrow$' is a nightmare to translate.
The semantics of '$\rightarrow$' works superbly for logic. But in English we use 'if-then' in many different ways, and not always carefully.

Don't read '$\rightarrow$' as 'causes': this is too strong.

E.g., '*I play in league 2 football*' '$\rightarrow$' '*I play in the Premiere league*' is true

But saying '*If I play in league 2 football, then I play in the Premiere league*' is false

Modalities (pitfalls)

A modality can be seen as shifting the context of evaluation of an utterance.

Time:

- '*I will be rich and famous*'
- Can't be translated as '*I will be rich*' and '*I will be famous*', since this would be true if I get famous only after I lose all my money.
- The original sentence suggests I'll be rich and famous at the same time.

Permission:

- '*You can have chicken or fish*'
- It usually means '*You can have chicken*' $\vee$ '*you can have fish*'

Obligation:

- '*I must do topics or a foreign language*'
- Can't be translated as '*I must do topics*' $\vee$ '*I must do a language*' (was false — you could choose which to take)

A really Vicious Example

This one deserves careful thought. Consider the formulas:

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

Let p be 'I throw the rock at the bottle', q be 'you throw the rock at the bottle', and r be 'the bottle shatters'.

> Then (1) says that **If** we both throw the rock at the bottle, **then** the bottle shatters.
>
> And (2) says If I throw the rock at the bottle **or** you throw it then the bottles shatters.

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

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

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

1. 'Arron and Russell are students'.

   This could be rewritten as: 'Arron is a student' **and** 'Russell is a student'.

How about

2. 'Arron and Russell are friends'.

   Is this the same as: 'Arron is a friend' **and** 'Russell is a friend'?

We might argue our way to represent this by rephrasing the English sentence as

'Arron is a friend of Russell' **and** 'Russell is a friend of Arron'.