# Propositional logic

Note Title

We deal with some fixed set of variables that describe a situation we're interested in.

e.g. $x \in \{1,2,3\}$
$y \in \{7,8\}$
$z \in \{3,5,7\}$

Begin with some motivation. See e.g. last page of these notes. Also mention the book's Wumpus world

A <u>model</u> assigns a value to each variable
e.g. $x=1,\ y=7,\ z=7$. is a model of the above

A sentence is a statement about the variables
e.g. "$x=2$", "if $y=z$ then $z=7$"

Sentence $\alpha$ <u>entails</u> sentence $\beta$, written $\alpha \models \beta$ if $\beta$ is true in all models where $\alpha$ is true.

e.g. "$y=z$" $\models$ "$z=7$"

In <u>propositional logic</u>, all variables have values in $\{\text{True}, \text{False}\}$

We can think of the variables $P, Q, R, \ldots$ representing statements (or 'propositions') like "I'm tall", "AI is fun"

Formally, symbols $P, Q, R, \ldots$ stand for <u>atomic sentences</u>.

We can combine them to produce <u>complex sentences</u> using <u>logical connectives</u>:

$\wedge$ — and  e.g. $P \wedge Q$

$\vee$ — or  $P \vee Q$

$\neg$ — not  $\neg P$

$\Rightarrow$ — implies  $P \Rightarrow Q$

$\Leftrightarrow$ — if & only if  $P \Leftrightarrow Q$

We can draw truth tables for these (see book fig 7.8)

e.g.

| $P$ | $Q$ | $P \Rightarrow Q$ |
| --- | --- | --- |
| F | F | T |
| F | T | T |
| T | F | F |
| T | T | T |

exercise: draw truth table for $\Leftrightarrow$

A legitimate combination of symbols is called a sentence

e.g. $P \vee (Q \wedge \neg P) \Rightarrow R$

Sentences are logically equivalent if they're true in the same set of models.

denoted $\equiv$

Important equivalences include:

de Morgan:

$$\neg(P \wedge Q) \equiv \neg P \vee \neg Q$$

$$\neg(P \vee Q) \equiv \neg P \wedge \neg Q$$

Implication elimination: $P \Rightarrow Q \equiv \neg P \vee Q$

(check truth table)

Biconditional elimination: $P \Leftrightarrow Q \equiv (P \Rightarrow Q) \wedge (Q \Rightarrow P)$

Distributivity:

$$P \wedge (Q \vee R) \equiv (P \wedge Q) \vee (P \wedge R)$$
$$P \vee (Q \wedge R) \equiv (P \vee Q) \wedge (P \vee R)$$

A sentence is

- <u>valid</u> (or a <u>tautology</u>) if it's true in all models

  e.g. $P \vee \neg P$ and $(P \wedge (P \Rightarrow Q)) \Rightarrow Q$ — both are valid

- <u>satisfiable</u> if it's true in some model

  e.g. $P \wedge Q$ - satisfiable but not valid

- <u>unsatisfiable</u> if it's not true in any model

  e.g. - $P \wedge \neg P$
  - $P \wedge \neg Q \wedge (P \Rightarrow Q)$

  both are unsatisfiable

Note that negating a valid sentence yields an unsatisfiable

e.g. $$\underbrace{P \vee \neg P}_{\text{valid}} \xrightarrow[\text{negate}]{} \neg(P \vee \neg P) \equiv \underbrace{\neg P \wedge P}_{\text{unsatisfiable}}$$

exercise: specify which are valid, satisfiable, unsatisfiable:

(a) $(P \Leftrightarrow Q) \wedge (P \Rightarrow \neg Q)$

(b) $(P \vee Q) \wedge \neg Q$

A <u>clause</u> is the OR of some atomic sentences (with negations allowed

e.g. $P \vee Q$

$P \vee \neg Q \vee \neg R$

$Q$

A sentence is in <u>conjunctive normal form</u> (CNF) if

it is the AND of some clauses

e.g. $(P \vee Q) \wedge (\neg P \vee R \vee S) \wedge T$

Any sentence can be converted to CNF with the following technique:

1. Eliminate $\Leftrightarrow$
2. Eliminate $\Rightarrow$
3. Move $\neg$ inwards
4. Distribute $\vee$ over $\wedge$

e.g. $P \Leftrightarrow Q \wedge R$

1. $(P \Rightarrow Q \wedge R) \wedge (Q \wedge R \Rightarrow P)$
2. $(\neg P \vee (Q \wedge R)) \wedge (\neg(Q \wedge R) \vee P)$
3. $(\neg P \vee (Q \wedge R)) \wedge (\neg Q \vee \neg R \vee P)$
4. $(\neg P \vee Q) \wedge (\neg P \vee R) \wedge (\neg Q \vee \neg R \vee P)$

exercise: Convert to CNF: $\neg P \Rightarrow (Q \Rightarrow (R \wedge S))$

A knowledge base (KB) is a set of sentences that are known to be true. (OR can combine KB into a single sentence using $\wedge$).

e.g. $KB = \{ P, \; P \Rightarrow Q, \; \neg R, \; S \wedge T \}$

[same as $KB = \{ P \wedge P \Rightarrow Q \wedge \neg R \wedge S \wedge T \}$]

Can add to KB using inference rules:

e.g. "and elimination": if $\alpha \wedge \beta \in KB$, can add $\alpha$ to KB.

Notation: $\boxed{\dfrac{\alpha \wedge \beta}{\alpha}}$

exercise: apply to above KB

"modus ponens": if $\alpha \in KB$ and $\alpha \Rightarrow \beta \in KB$, can add $\beta$ to KB

Notation: $\boxed{\dfrac{\alpha, \; \alpha \Rightarrow \beta}{\beta}}$

exercise: apply to above KB

"resolution": see next section

**Resolution** is an important inference rule.

Basic idea is that opposite literals in separate clauses cancel each other out, yielding a combined clause.

e.g. $P \vee Q$, $R \vee \neg Q$ yields $P \vee R$

cancel

combine

$P \vee Q \vee \neg R \vee S$, $Q \vee R \vee T \vee \neg U$

cancel

combine

yields $P \vee Q \vee S \vee T \vee \neg U$

Exercise: Apply resolution to the KB
$\{P \vee \neg Q, \neg P \vee R \vee \neg S, S \vee T\}$

The resolution rule is important because it can be used as part of an algorithm that infers entailment.
i.e. it decides whether $KB \models \alpha$ for any $KB, \alpha$.

We study this next.

## Our Simple resolution algorithm for entailment

We want to determine whether $KB \models \alpha$.
Equivalently, is $KB \Rightarrow \alpha$ valid?
Equivalently, is $KB \wedge \neg\alpha$ unsatisfiable?

Algorithm:
- Convert $KB \wedge \neg\alpha$ to CNF
- Apply resolution repeatedly
- If you ever get an empty clause, conclude that $KB \models \alpha$.
- If can't make any more clauses, conclude that $KB \not\models \alpha$.

Why? Because you've derived the empty clause, equiv to 'False', meaning $KB \wedge \neg\alpha$ is unsatisfiable

Why? Because you can now satisfy $KB \wedge \neg\alpha$. Detailed proof in book (not required) but basically just fill in the values

Exercise:

$$KB = \{ P \Rightarrow Q,\ Q \vee R \vee S,\ S \Rightarrow P \vee Q \}$$

(i) Does KB entail $Q \vee R$?
(ii) Does KB entail $\neg Q \wedge S$?

## Why, why, why?

One application among very many is proving correctness of computer programs.

e.g.

Java program — infer KB from preceding statements

```
int x = y/z ;
```

let $P$ be the proposition "$z \neq 0$".
Can we prove that the $KB \models P$ ?
i.e. prove that we will <u>never</u> get a divide-by-zero exception when running this program?