Propositional logic

Michael Franke

Syntax & semantics of propositional logic; truth-tables; tautologies vs. contraditions vs. contingencies; translations from natural language into propositional logic; argument schemas & logical validity.

1 *The language of propositional logic*

Propositional logic (PropLog) studies how propositions are combined by logical operators, which closely correspond to certain sentential connectives in natural language (such as *and*, *or*, *if*, or *not*). A *proposition* in the sense of PropLog is a minimal unit of thought which can be evaluated as true or false independently of other propositions.[1] For example, the logical structure of the following sentence:

$$\underbrace{\text{The earth is round}}_{p}\ \underbrace{\text{and}}_{\wedge}\ \underbrace{\text{the moon is made of cheese.}}_{q}$$

could be analyzed as composed of two propositions, viz., the proposition denoted here with *proposition letter* p that the earth is round and the proposition denoted by q that the moon is made of cheese. These two propositions are connected by a logical operator "and", for which we write $\wedge$ in PropLog. The logical structure of the complex sentence above can therefore be written as $p \wedge q$ in PropLog.

[1] The notion of a proposition in this sense is not unproblematic. For example, a case like "*This pixel is red.*" seems like a minimal unit of truth-evaluable information about the color of a particular pixel, but it is not independent of another statement like "*This pixel is blue.*". (Historically, this problem related to color was a problem brought up famously by Frank Ramsey in response to the early logical work of Ludwig Wittgenstein.)

1.1 *Proposition letters & sentential connectives*

The language of PropLog is formed by:

(i) a set of *proposition letters* $\mathfrak{P} = \{p, q, r, s, p_1, q_{27}, \dots\}$, and

(ii) a set of *sentential connectives* $\{\neg, \wedge, \vee, \rightarrow, \leftrightarrow\}$[2]

The sentential connectives have names and are intended to correspond (approximately) to natural language paraphrases:[3]

name	paraphrase	symbol
negation	"not"	$\neg$
conjunction	"and"	$\wedge$
disjunction	"or"	$\vee$
implication	"if ..., then ..."	$\rightarrow$
equivalence	"if and only of"	$\leftrightarrow$

[2] You will also find terminology like *logical connectives* or *logical operators*. There may be additional connectives used by some logicians or textbooks, and you might find slightly different symbols for the same notions in some places

[3] The names and the (approximate) correspondence are earned by the *semantics* which we will give to these symbols later on.

1.2 *Formulas*

The language $\mathfrak{L}$ of PropLog is the set of all *formulas* which are recursively defined as follows:[4]

[4] We will make consistent use of Greek letters $\varphi, \psi, \chi, \dots$ as variables for formulas.

(i) Every proposition letter is a formula.

(ii) If φ is a formula, so is $\neg\varphi$.

(iii) If φ and ψ are formulas, so are:
a. $(\varphi \wedge \psi)$ b. $(\varphi \vee \psi)$ c. $(\varphi \rightarrow \psi)$ d. $(\varphi \leftrightarrow \psi)$

(iv) Anything that cannot be constructed by (i)–(iii) is not a formula.

Examples for formulas of PropLog are:[5]

$$pp \wedge p$$
$$p \rightarrow \neg q(p \vee q) \leftrightarrow r$$

[5] We conventionally omit the outermost parentheses of a formula.

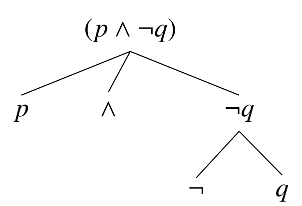
Examples of strings made of proposition letters and logical connectives which are *not* formulas of PropLog are:

$$(p) \qquad \neg pq$$
$$p\neg \rightarrow \neg q \qquad p \vee q \leftrightarrow r$$

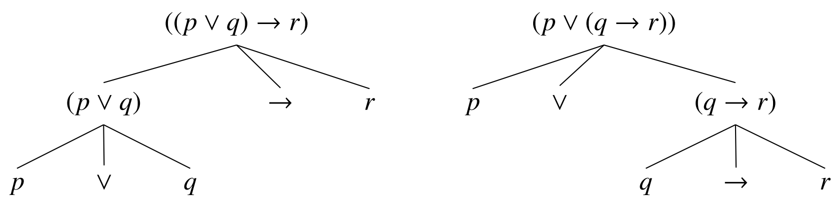
1.3 *Syntactic trees*

The recursive definition for formulas of PropLog gives an internal structure to each formula. Take the example $p \wedge \neg q$. There is only one way in which this formulas could have been generated by a constructive process that follows the recursive definition above. In the last step of that process, the two subformulas $\varphi = p$ and $\psi = \neg q$ have been combined to form an expression of the form $\varphi \wedge \psi$ using the rule (iii) part a. The first subformula $\varphi = p$ is constructed by rule (i). The second formula can only be constructed by first using rule (i) to introduce q and then using rule (ii) to introduce the negation sign.

A *syntactic tree* is a useful visual illustration of the internal structure of a formula. The syntactic tree of formula $p \wedge \neg q$ is this:



The construction of a complex formula, and therefore its syntactic tree, is always recoverable by following the introduction of the parentheses in step (iii). This is illustrated by the minimal pair in Figure 1.



(a) Tree for $((p \vee q) \to r)$ (b) Tree for $(p \vee (q \to r))$

Figure 1: Examples of syntactic trees

1.4 *Atomic & complex formulas*

A formula of PropLog which consist of a single proposition letter is also called *atomic formula.* Any formula of PropLog which is not atomic is also called a *complex formula*. Each complex formula has a *main connective*. The main connective is the last sentential connector introduced during the construction of the formula. A complex formula is also often called by the name of its main connective. For example, the formula $p \wedge \neg q$ has a conjunction as its main operator and could therefore be called a conjunction. The formula $(p \vee q) \to r$ from Figure 1(a) is an implication, while the formula $p \vee (q \to r)$ from Figure 1(b) is a disjunction.

Exercise 1. Determine which of the following strings are formulas of propositional logic. For any formula, determine its main operator.

a. q_{12}

b. $p, q \wedge r$

c. $(p) \wedge q$

d. $p \to (p \wedge p)$

e. $(p \to)(p \wedge p)$

f. $(p \vee \neg q) \leftrightarrow (r \to (\neg(p \vee \neg p)))$

Exercise 2. Draw the syntactic tree for each of the following formulas.

a. $p \leftrightarrow q$

b. $\neg p \wedge p$

c. $p \to \neg(q \wedge r)$

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