Part I

Propositional logic

Chapter 1

Syntax and Semantics of Propositional Logic

Syntax is a set of formal rules for creating well-formed sentences consisting of words (in the case of natural languages) or formal expressions consisting of symbols (e.g., statements in a programming language). In contrast, *semantics* describes the meaning of such expressions. The relationship between syntax and semantics is fundamental to all of logic and is therefore the key to its understanding.

1.1 Syntax of Propositional Logic

First, we define the formal 'statements' with which we will work in propositional logic.

1.1.1 Language

The *language* of propositional logic is determined by a non-empty set of *propositional variables* $\mathbb{P}$ (also called *atomic propositions* or *atoms*). This set can be finite or infinite, but it will usually be countable[1] (unless otherwise specified), and it will have a fixed ordering. For propositional variables, we will typically use the notation p_i (from the word "proposition"), but for better readability, especially if $\mathbb{P}$ is finite, we will also use $p, q, r, \ldots$. For example:

$$\mathbb{P}_1 = \{p, q, r\}$$
$$\mathbb{P}_2 = \{p_0, p_1, p_2, p_3, \ldots\} = \{p_i \mid i \in \mathbb{N}\}$$

In addition to propositional variables, the language also includes *logical symbols*: symbols for logical connectives $\neg, \wedge, \vee, \rightarrow, \leftrightarrow$ and parentheses $(,)$. However, for simplicity, we will talk about the "language $\mathbb{P}$".

Remark 1.1.1. If we need to formally express the ordering of the propositional variables in $\mathbb{P}$, we imagine it as a bijection $\iota\colon \{0, 1, \ldots, n-1\} \to \mathbb{P}$ (for a finite, n-element language) or $\iota\colon \mathbb{N} \to \mathbb{P}$ (if $\mathbb{P}$ is countably infinite). In our examples, $\iota_1(0) = p$, $\iota_1(1) = q$, $\iota_1(2) = r$, and $\iota_2(i) = p_i$ for all $i \in \mathbb{N}$.[2]

[1] This is important in many applications in computer science, as uncountable sets cannot fit into a (even infinite) computer.

[2] The set of natural numbers $\mathbb{N}$ includes zero, see standard ISO 80000-2:2019.

1.1.2 Proposition

The basic building block of propositional logic is a *proposition*, also called *propositional formula*. It is a finite string composed of propositional variables and logical symbols according to certain rules. Atomic propositions are propositions, and we can further create propositions from simpler propositions and logical symbols: for example, for the logical connective $\wedge$, we write first the symbol '(', then the first proposition, the symbol '$\wedge$', the second proposition, and finally the symbol ')'.

Definition 1.1.2 (Proposition)**.** A *proposition* (*propositional formula*) in the language $\mathbb{P}$ is an element of the set $\mathrm{PF}_{\mathbb{P}}$ defined as follows: $\mathrm{PF}_{\mathbb{P}}$ is the smallest set satisfying[3]

- for every atomic proposition $p \in \mathbb{P}$, $p \in \mathrm{PF}_{\mathbb{P}}$,
- for every proposition $\varphi \in \mathrm{PF}_{\mathbb{P}}$, $(\neg\varphi)$ is also an element of $\mathrm{PF}_{\mathbb{P}}$,
- for every $\varphi, \psi \in \mathrm{PF}_{\mathbb{P}}$, $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$, $(\varphi \to \psi)$, and $(\varphi \leftrightarrow \psi)$ are also elements of $\mathrm{PF}_{\mathbb{P}}$.

Propositions are usually denoted by Greek letters φ, ψ, χ (φ from the word "formula"). To avoid listing all four binary logical connectives, we sometimes use a placeholder symbol $\square$. Thus, the third point of the definition could be expressed as:

- for every $\varphi, \psi \in \mathrm{PF}_{\mathbb{P}}$ and $\square \in \{\wedge, \vee, \to, \leftrightarrow\}$, $(\varphi \square \psi)$ is also an element of $\mathrm{PF}_{\mathbb{P}}$.

A *subproposition* (*subformula*) is a substring that is itself a proposition. Note that all propositions are necessarily finite strings, created by applying a finite number of steps from the definition to their subpropositions.

Example 1.1.3. The proposition $\varphi = ((p \vee (\neg q)) \leftrightarrow (r \to (p \wedge q)))$ has the following subpropositions: $p, q, (\neg q), (p \vee (\neg q)), r, (p \wedge q), (r \to (p \wedge q)), \varphi$.

A proposition in the language $\mathbb{P}$ does not have to contain all atomic propositions from $\mathbb{P}$ (and it cannot if $\mathbb{P}$ is an infinite set). Therefore, it will be useful to denote by $\mathrm{Var}(\varphi)$ the set of atomic propositions occurring in φ.[4] In our example, $\mathrm{Var}(\varphi) = \{p, q, r\}$.

We introduce abbreviations for two special propositions: $\top = (p \vee (\neg p))$ (*tautology*) and $\bot = (p \wedge (\neg p))$ (*contradiction*), where $p \in \mathbb{P}$ is fixed (e.g., the first atomic proposition from $\mathbb{P}$). Thus, the proposition $\top$ is always true, and the proposition $\bot$ is always false.

When writing propositions, we may omit some parentheses for better readability. For example, the proposition φ from Example 1.1.3 can be represented by the string $p \vee \neg q \leftrightarrow (r \to p \wedge q)$. We omit outer parentheses and use priority of operators: $\neg$ has the highest priority, followed by $\wedge, \vee$, and finally $\to, \leftrightarrow$ have the lowest priority. Furthermore, the notation $p \wedge q \wedge r \wedge s$ means the proposition $(p \wedge (q \wedge (r \wedge s)))$, and similarly for $\vee$.[5][6]

[3] This kind of definition is called *inductive*. It can also be naturally expressed using a *formal grammar*, see the course NTIN071 Automata and Grammars.

[4] If we do not specify the language of a proposition (and if it is not clear from the context), we mean that it is in the language $\mathrm{Var}(\varphi)$.

[5] Due to the associativity of $\wedge, \vee$, the placement of parentheses does not matter.

[6] Sometimes finer priorities are introduced, $\wedge$ often has higher priority than $\vee$, and $\to$ has higher priority than $\leftrightarrow$. Also, sometimes $p \to q \to r$ is written instead of $(p \to (r \to q))$, although $\to$ is not associative and so here the placement of parentheses does matter. We prefer to avoid both of those conventions.

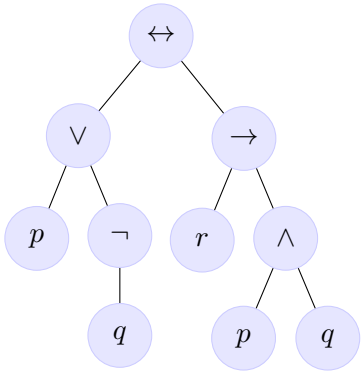


Figure 1.1: Tree of the proposition $\varphi = ((p \vee (\neg q)) \leftrightarrow (r \rightarrow (p \wedge q)))$

1.1.3 Tree of a Proposition

In the definition of a proposition, we chose *infix* notation (with parentheses) purely for human readability. Nothing would prevent us from using *prefix* ("Polish") notation, i.e., defining propositions as follows:

- every atomic proposition is a proposition, and
- if φ, ψ are propositions, then $\neg\varphi$, $\wedge\varphi\psi$, $\vee\varphi\psi$, $\rightarrow\varphi\psi$, and $\leftrightarrow\varphi\psi$ are also propositions.

The proposition $\varphi = ((p \vee (\neg q)) \leftrightarrow (r \rightarrow (p \wedge q)))$ would then be written as $\varphi = \leftrightarrow\vee p\neg q\rightarrow r\wedge pq$. We could also use *postfix* notation and write $\varphi = pq\neg\vee rpq\wedge\rightarrow\leftrightarrow$. The essential information about a proposition is actually contained in its tree structure, which captures how it is composed of simpler propositions, similar to the tree of an arithmetic expression.

Example 1.1.4. The tree of the proposition $\varphi = ((p \vee (\neg q)) \leftrightarrow (r \rightarrow (p \wedge q)))$ is illustrated in Figure 1.1. Notice also how the subpropositions of φ correspond to subtrees. The proposition φ is obtained by traversing the tree from the root, and at each node:

- if the label is an atomic proposition, write it out,
- if the label is a negation: write '(¬', recursively call the child, write ')',
- otherwise (for binary logical connectives): write '(', call the left child, write the label, call the right child, write ')'.[7]

Now we will define the tree of a proposition formally, *by induction on the structure of the proposition*:[8]

Definition 1.1.5 (Tree of a Proposition)**.** The *tree of a proposition* φ, denoted $\mathrm{Tree}(\varphi)$, is a rooted ordered tree, defined inductively as follows:

[7] Prefix and postfix notations would be obtained similarly, but we do not write parentheses, and the label is written immediately upon entering or just before leaving the node.

[8] Once we have the tree of a proposition defined, we can understand induction on the structure of the proposition as induction on the depth of the tree. For now, understand it as induction on the number of steps in Definition 1.1.2 by which the proposition was created. Alternatively, induction on the length of the proposition or the number of logical connectives would work as well.

- If φ is an atomic proposition p, Tree(φ) contains a single node, and its label is p.
- If φ is of the form $(\neg\varphi')$, Tree(φ) has a root labeled $\neg$, and its single child is the root of Tree(φ').
- If φ is of the form $(\varphi' \square \varphi'')$ for $\square \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$, Tree($\varphi$) has a root labeled $\square$ with two children: the left child is the root of the tree Tree(φ'), and the right child is the root of Tree(φ'').

Exercise 1.1. Prove that every proposition has a uniquely determined proposition tree, and vice versa.

1.1.4 Theory

In practical applications, we do not express the desired properties with a single proposition — it would have to be very long and complex and difficult to work with — but with many simpler propositions.

Definition 1.1.6 (Theory)**.** A *theory* in the language $\mathbb{P}$ is any set of propositions in $\mathbb{P}$, that is, any subset $T \subseteq \mathrm{PF}_{\mathbb{P}}$. The individual propositions $\varphi \in T$ are also called *axioms*.

Finite theories could be replaced by a single proposition: the conjunction of all their axioms. But that would not be practical. Moreover, we also allow infinite theories (a trivial example is the theory $T = \mathrm{PF}_{\mathbb{P}}$), and the empty theory $T = \emptyset$.[9]

1.2 Semantics of Propositional Logic

In our logic, the semantics are given by one of two possible values: *True* or *False*. (In other logical systems, semantics can be more interesting.)

1.2.1 Truth Value

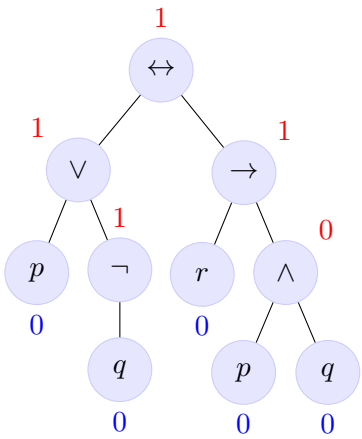
Propositions can be assigned one of two possible truth values: *True* (1) or *False* (0). Atomic propositions represent simple, indivisible statements (hence the name '*atomic*'); their truth value must be assigned to correspond to what we want to model (that is why we call them *propositional variables*). Once we *assign* truth values to the atomic propositions, the truth value of any compound proposition is uniquely determined and can be easily calculated according to the tree of the proposition:

Example 1.2.1. Let us calculate the truth value of the proposition $\varphi = ((p \vee (\neg q)) \leftrightarrow (r \rightarrow (p \wedge q)))$ for the assignments (a) $p = 0$, $q = 0$, $r = 0$ and (b) $p = 1$, $q = 0$, $r = 1$. We proceed from the leaves towards the root, similarly to evaluating an arithmetic expression. The proposition φ is *true* under assignment (a) and *false* under assignment (b). See Figure 1.2.

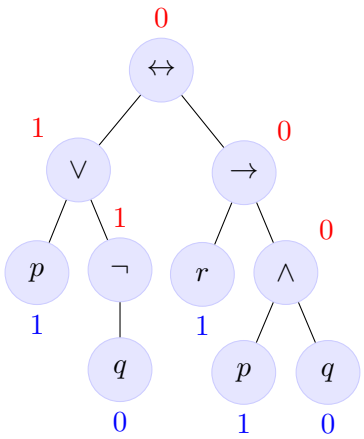
Logical connectives in the inner nodes are evaluated according to their *truth tables*, see Table 1.1.[10]

[9] Infinite theories are useful, for example, for describing the development of a system over (discrete) time steps $t = 0, 1, 2, \ldots$. The empty theory is not useful for anything, but it would be awkward to formulate statements about logic if theories had to be non-empty.

[10] Let us recall once again that disjunction is not exclusive, i.e., $p \vee q$ is true even if both p and q are true, and that implication is purely logical, i.e., $p \rightarrow q$ is true whenever p is false.



(a) $p = 0$, $q = 0$, $r = 0$



(b) $p = 1$, $q = 0$, $r = 1$

Figure 1.2: Truth value of a proposition

p	q	$\neg p$	$p \wedge q$	$p \vee q$	$p \to q$	$p \leftrightarrow q$
0	0	1	0	0	1	1
0	1	1	0	1	1	0
1	0	0	0	1	0	0
1	1	0	1	1	1	1

Table 1.1: Truth tables of logical connectives.

1.2.2 Propositions and Boolean Functions

To formalize the truth value of a proposition, we first look at the relationship between propositions and Boolean functions.

A *Boolean function* is a function $f\colon \{0,1\}^n \to \{0,1\}$, meaning that the input is an n-tuple of zeros and ones, and the output is 0 or 1. Each logical connective represents a Boolean function. In the case of negation, it is a unary function $f_\neg(x) = 1 - x$, while the other logical connectives correspond to binary functions described in Table 1.2.

Definition 1.2.2 (Truth Function)**.** The *truth function* of a proposition φ in a finite language $\mathbb{P}$ is the function $f_{\varphi,\mathbb{P}}\colon \{0,1\}^{|\mathbb{P}|} \to \{0,1\}$ defined inductively:

- If φ is the i-th atomic proposition from $\mathbb{P}$, then $f_{\varphi,\mathbb{P}}(x_0, \ldots, x_{n-1}) = x_i$,
- If $\varphi = (\neg\varphi')$, then

$$f_{\varphi,\mathbb{P}}(x_0, \ldots, x_{n-1}) = f_\neg(f_{\varphi',\mathbb{P}}(x_0, \ldots, x_{n-1})),$$

$f_\wedge(x,y)$:

	0	1
0	0	0
1	0	1

$f_\vee(x,y)$:

	0	1
0	0	1
1	1	1

$f_\to(x,y)$:

	0	1
0	1	1
1	0	1

$f_\leftrightarrow(x,y)$:

	0	1
0	1	0
1	0	1

Table 1.2: Boolean functions of logical connectives

- If $(\varphi' \square \varphi'')$ where $\square \in \{\wedge, \vee, \to, \leftrightarrow\}$, then

$$f_{\varphi,\mathbb{P}}(x_0, \dots, x_{n-1}) = f_{\square}(f_{\varphi',\mathbb{P}}(x_0, \dots, x_{n-1}), f_{\varphi'',\mathbb{P}}(x_0, \dots, x_{n-1})).$$

Example 1.2.3. Let us calculate the truth function of the proposition $\varphi = ((p \vee (\neg q)) \leftrightarrow (r \to (p \wedge q)))$ in the language $\mathbb{P}' = \{p, q, r, s\}$:

$$f_{\varphi,\mathbb{P}'}(x_0, x_1, x_2, x_3) = f_{\leftrightarrow}(f_{\vee}(x_0, f_{\neg}(x_1)), f_{\to}(x_2, f_{\wedge}(x_0, x_1)))$$

Truth value of the proposition φ for the truth assignment $p = 1$, $q = 0$, $r = 1$, $s = 1$ is calculated as follows (compare with Figure 1.2(b)):

$$\begin{aligned} f_{\varphi,\mathbb{P}'}(1,0,1,1) &= f_{\leftrightarrow}(f_{\vee}(1, f_{\neg}(0)), f_{\to}(1, f_{\wedge}(1,0))) \\ &= f_{\leftrightarrow}(f_{\vee}(1,1), f_{\to}(1,0)) \\ &= f_{\leftrightarrow}(1,0) \\ &= 0 \end{aligned}$$

Observation 1.2.4. *The truth function of a proposition φ over $\mathbb{P}$ depends only on the variables corresponding to the atomic propositions in* $\mathrm{Var}(\varphi) \subseteq \mathbb{P}$.

Thus, even if we have a proposition φ in an *infinite* language $\mathbb{P}$, we can restrict ourselves to the language $\mathrm{Var}(\varphi)$ (which is finite) and consider the truth function over this language.

1.2.3 Models

A given truth assignment of propositional variables is a representation of the 'real world' (system) in our chosen 'formal world,' hence it is also called a *model*.

Definition 1.2.5 (Model of a Language)**.** A *model* of a language $\mathbb{P}$ is any truth assignment $v\colon \mathbb{P} \to \{0,1\}$. The *set of (all) models of a language* $\mathbb{P}$ is denoted by $\mathrm{M}_{\mathbb{P}}$:

$$\mathrm{M}_{\mathbb{P}} = \{v \mid v\colon \mathbb{P} \to \{0,1\}\} = \{0,1\}^{\mathbb{P}}$$

Models will be denoted by letters v, u, w, etc. (v from the word 'valuation'). A model of a language is therefore a function, formally a set of pairs (input, output). For example, for the language $\mathbb{P} = \{p, q, r\}$ and the truth assignment where p is true, q false, and r true, we have the model

$$v = \{(p,1), (q,0), (r,1)\}.$$

For simplicity, we will write it as $v = (1,0,1)$. For the language $\mathbb{P} = \{p, q, r\}$, we have $2^3 = 8$ models:

$$\mathrm{M}_{\mathbb{P}} = \{(0,0,0), (0,0,1), (0,1,0), (0,1,1), (1,0,0), (1,0,1), (1,1,0), (1,1,1)\}$$

Remark 1.2.6. Formally speaking, we identify the set $\{0,1\}^{\mathbb{P}}$ with the set $\{0,1\}^{|\mathbb{P}|}$ using the ordering ι of the language $\mathbb{P}$ (see Remark 1.1.1). Specifically, instead of the element $v = \{(p,1), (q,0), (r,1)\} \in \{0,1\}^{\mathbb{P}}$, we write $(1,0,1) = (v \circ \iota)(0,1,2) = (v(\iota(0)), v(\iota(1)), v(\iota(2))) \in \{0,1\}^{|\mathbb{P}|}$ (where we allow the functions v, ι to act 'component-wise').[11] If this seems confusing, imagine the model v as a set of atomic propositions that are true, i.e., $\{p, r\} \subseteq \mathbb{P}$, our notation $v = (1,0,1)$ is then the characteristic vector of this set. This identification will be used henceforth without further notice.

[11] Alternatively, we could require (at least for countable languages) that the language *be* $\mathbb{P} = \{0, 1, 2, \dots\}$ and use symbols p_0, p_1, p, q, r only for readability.