# MATH70132: Mathematical Logic

Lecturer: David Evans

Scribe: Sidharth Hariharan

Imperial College London - Spring 2025

# Contents

# Chapter 1

# Propositional Logic

Propositional logic is the logic of reasoning and proof. Before we get started with anything formal, here's a motivating example.

Consider the following statement:

> If Mr Jones is happy, then Mrs Jones is unhappy, and if Mrs Jones is unhappy, then Mr Jones is unhappy. Therefore, Mr Jones is unhappy.

One can ask ourselves whether it is logically valid to conclude that Mr Jones is unhappy based on the relationship between the happiness of Mr Jones and that of Mrs Jones expressed in the sentence preceding it.

Putting this into symbols, let $P$ denote the statement that Mr Jones is happy, and let $Q$ denote the statement that Mrs Jones is unhappy. We can express the statement as follows:

$$((P \implies Q) \wedge (Q \implies \neg P)) \implies (\neg P) \tag{1.0.1}$$

This disambiguation, by removing any question of marital harmony from what is otherwise a purely logical question, allows us to manually check whether (??) is a valid statement by constructing a **truth table**.

We will begin by developing some machinery to reason about these sorts of statements more formally.

## 1.1 Propositional Formulae

### 1.1.1 Propositions and Connectives

We begin by defining the notion of a proposition.

**Definition 1.1.1** (Proposition)**.** A **proposition** is a statement that is either true or false.

**Convention.** We will denote the state of being **true** by **T** and that of being **false** by **F**.

Propositions can be connected to each other using tools known as **connectives**. These can be thought of as **truth table rules**.

**Convention.** Before we define the actual connectives we shall use, we list them down, along with notation.

1. Conjunction ($\wedge$)
2. Disjunction ($\vee$)
3. Negation ($\neg$)
4. Implication ($\rightarrow$)
5. The Biconditional ($\leftrightarrow$)

In particular, we will only use the $\implies$ and $\iff$ symbols when reasoning **informally**. For **formal** use, we will stick to the $\rightarrow$ and $\leftrightarrow$ symbols. In more precise terms, we will use $\implies$ and $\iff$ when reasoning **about** the language we are constructing, whereas we will use $\rightarrow$ and $\leftrightarrow$ when reasoning **within** the language. As we shall see, it will be of paramount importance to distinguish between these two modes of reasoning.

We define them exhaustively as follows.

**Definition 1.1.2** (Connectives)**.** Let $p$ and $q$ be true/false variables. We define each of the connectives listed above to take on truth values depending on those of $p$ and $q$ as follows.

| $p$ | $q$ | $(\neg p)$ | $(\neg q)$ | $(p \wedge q)$ | $(p \vee q)$ | $(p \to q)$ | $(p \leftrightarrow q)$ |
|---|---|---|---|---|---|---|---|
| **T** | **T** | **F** | **F** | **T** | **T** | **T** | **T** |
| **T** | **F** | **F** | **T** | **F** | **T** | **F** | **F** |
| **F** | **T** | **T** | **F** | **F** | **T** | **T** | **F** |
| **F** | **F** | **T** | **T** | **F** | **F** | **T** | **T** |

We are now ready to define the main object of study in this section: propositional formulae.

**Definition 1.1.3** (Propositional Formula)**.** A **propositional formula** is obtained from propositional variables and connectives via the following rules:

(i) Any propositional variable is a propositional formula.

(ii) If $\phi$ and $\psi$ are formulae, then so are $(\neg\phi)$, $(\neg\psi)$, $(\phi \wedge \psi)$, $(\phi \vee \psi)$, $(\phi \to \psi)$, $(\psi \to \phi)$, and $(\phi \leftrightarrow \psi)$.

(iii) Any formula arises in this manner after a finite number of steps.

What this means is that a propositional formula is a string of symbols consisting of

1. variables that take on true/false values,
2. connectors that express the relationship between these variables, and
3. parentheses/brackets that separate formulae within formulae and specify the order in which they must be evaluated when the constituent variables are assigned specific values.

In particular, every propositional formula is either a propositional variable or is built from 'shorter' formulae, where by 'shorter' we mean consisting of fewer symbols.

**Convention.** Throughout this module, we will adopt two important conventions when dealing with propositional formulae.

1. All propositional formulae, barring those consisting of a single variable, shall be enclosed in parentheses.
2. When we want to denote a propositional formula by a certain symbol, we will use the notation "symobol : formula".

As a concluding remark on the nature of propositional formulae, we will note that just as we use trees to evaluate expressions on the computer when performing arithmetic, we can use them to

express and evaluate propositional formulae as well. We will not usually do this, however, as it takes up a lot of space. In any event, we would first need to make precise the notion of *evaluating* a propositional formula. For this, we will turn to the concept of a truth function.

### 1.1.2 Truth Functions

Any assignment of truth values to the propositional variables in a formula $\phi$ determines the truth value for $\phi$ in a **unique** manner, using the exhaustive definitions of the connectives given in **??**. We often express all possible values of a propositional formula in a **truth table**, much like we did in **??** when defining the connectives.

**Example 1.1.4.** Consider the formula $\phi : ((p \to (\neg q)) \to p)$, where $p$ and $q$ are propositional variables. We construct a truth table as follows.

| $p$ | q | $(\neg q)$ | $(p \to (\neg q))$ | $((p \to (\neg q)) \to p)$ |
|---|---|---|---|---|
| **T** | **T** | **F** | **F** | **T** |
| **T** | **F** | **T** | **T** | **T** |
| **F** | **T** | **F** | **T** | **F** |
| **F** | **F** | **F** | **T** | **F** |

From this table, it is clear that the truth value of $\phi$ depends on the truth values of $p$ and $q$ in some manner (to be perfectly precise, it only depends on the truth value of $p$, and is, in fact, biconditionally equivalent to $p$). We would like to have a formal notion of navigating this dependence to 'compute a value for $\phi$ given values of $p$ and $q$'.

Throughout this subsection, $n$ will denote an arbitrary natural number.

**Definition 1.1.5** (Truth Function)**.** A **truth function** of $n$ variables is a function

$$f : \{\mathbf{T}, \mathbf{F}\}^n \to \{\mathbf{T}, \mathbf{F}\}$$

Before discussing the relevance of truth functions, we will mention a very natural fact.

**Lemma 1.1.6.** *To show two truth functions are equal, it suffices that they take the value* **T** *on precisely the same inputs or that they take the value* **F** *on precisely the same inputs.*

*Proof.* This is obvious, because any truth function can only take one of two values. If they take one value on precisely the same inputs, they must take the other value on the other inputs. This precisely corresponds to what it means for functions to be equal by extensionality. □

These are very directly related to propositional formulae.

**Definition 1.1.7** (Truth Function of a Propositional Formula)**.** Let $\phi$ be a propositional formula whose variables are $p_1, \ldots, p_n$. We can associate to $\phi$ a truth function whose truth value at any $(x_1, \ldots, x_n) \in \{\mathbf{T}, \mathbf{F}\}^n$ corresponds to the truth value of $\phi$ that arises from setting $p_i$ to $x_i$ for all $1 \leq i \leq n$. We define this truth function to be the **truth function of** $\phi$, denoted $F_\phi$.

We can now construct a truth function for the example we saw at the very beginning involving Mr and Mrs Jones (cf. (??)).

**Example 1.1.8.** `sorry`

We see something quite remarkable here: the truth function of the propositional formula defined in (??) maps every possible input to **T**! We have a special term for this.

**Definition 1.1.9** (Tautology)**.** A propositional formula $\phi$ is a **tautology** if its truth function $F_\phi$ maps every possible input to **T**.

We can also be more precise about what the biconditional connective actually tells us.

**Definition 1.1.10** (Logical Equivalence)**.** The propositional formulae $\psi$ and $\chi$ are **logically equivalent** if the truth function $F_{\psi \leftrightarrow \chi}$ of their biconditional is a tautology.

We have a fairly basic result about logical equivalence.

**Lemma 1.1.11.** *Let $p_1, \ldots, p_n$ be propositional variables and let $\psi$ and $\chi$ be formulae in $p_1, \ldots, p_n$. Then, $\psi$ and $\chi$ are logically equivalent if and only if $F_\psi = F_\chi$.*

We omit the proof of this result as it merely involves checking things manually. A computer should

be able to do this almost instantaneously.

We can also say something about composing formulae together.

**Lemma 1.1.12.** *Suppose that $\phi$ is a propositional formula with variables $p_1, \ldots, p_n$. Let $\phi_1, \ldots, \phi_n$ be propositional formulae. Denote by $\vartheta$ the result of substituting each $p_i$ with $\phi_i$ in $\phi$. Then,*

(i) *$\vartheta$ is a propositional formula.*

(ii) *if $\phi$ is a tautology, so is $\vartheta$.*

(iii) *the truth function of $\vartheta$ is the result of composing the truth function of $\phi$ with the Cartesian product of the truth functions of $\phi_1, \ldots, \phi_n$.*

We do not prove this result either, as it merely involves manual verification.

**Example 1.1.13.** For propositional variables $p_1, p_2$, the statement $(((\neg p_2) \to (\neg p_1)) \to (p_1 \to p_2))$ is a tautology. Therefore, if $\phi_1$ and $\phi_2$ are propositional formulae, then $(((\neg \phi_2) \to (\neg \phi_1)) \to (\phi_1 \to \phi_2))$ is a tautology as well.

We will also mention that a composition being a tautology does not mean the outermost proposition of the composition is a tautology.

**Non-Example 1.1.14.** Let $p$ be a propositional variable. The formula $\phi : (p \to (\neg p))$ is not a tautology. However, we can find a propositional formula $\phi'$ such that $(\phi_1 \to (\neg \phi_1))$ is a tautology: for example, we can define $\phi'$ to be identically **F**.

There are numerous propositional formulae that we know to be logically equivalent. Here is a (non-exhaustive) list.

**Example 1.1.15** (Logically Equivalent Formulae)**.** Let $p_1, p_2, p_3$ be logically equivalent formulae. Then, the following equivalences hold.

1. $(p_1 \wedge (p_2 \wedge p_3))$ is logically equivalent to $((p_1 \wedge p_2) \wedge p_3)$.
2. $(p_1 \vee (p_2 \vee p_3))$ is logically equivalent to $((p_1 \vee p_2) \vee p_3)$.
3. $(p_1 \vee (p_2 \wedge p_3))$ is logically equivalent to $((p_1 \vee p_2) \wedge (p_1 \wedge p_3))$.

4. $(\neg(\neg p_1))$ is logically equivalent to $p_1$.
5. $(\neg(p_1 \wedge p_2))$ is logically equivalent to $((\neg p_1) \vee (\neg p_2))$.
6. $(\neg(p_1 \vee p_2))$ is logically equivalent to $((\neg p_1) \wedge (\neg p_2))$.

Upon inspection, one can find algebraic patterns in the above logical equivalences. There are similarities to the axioms of a **boolean algebra**. We will not explore this further in this module, but we will adopt the convention used in algebra where parentheses are dropped when dealing with associative operations.

**Convention.** We will denote both $(p_1 \wedge (p_2 \wedge p_3))$ and $((p_1 \wedge p_2) \wedge p_3)$ by $(p_1 \wedge p_2 \wedge p_3)$. Similarly, we will denote both $(p_1 \vee (p_2 \vee p_3))$ and $((p_1 \vee p_2) \vee p_3)$ by $(p_1 \vee p_2 \vee p_3)$.

We will end with a combinatorial fact about truth functions.

**Lemma 1.1.16.** *There are $2^{2^n}$ possible truth functions on $n$ variables.*

*Proof.* A truth function is any function from a the set $\{\mathbf{T}, \mathbf{F}\}^n$ to the set $\{\mathbf{T}, \mathbf{F}\}$, with no further restrictions. The former set has $2^n$ elements and the latter set has 2 elements. Therefore, there are $2^{2^n}$ possible truth functions. □

### 1.1.3 Adequacy

We have defined several connectives so far, but we have yet to say anything about whether we will be defining any more connectives going forward. To begin, we will state an important definition.

**Definition 1.1.17** (Adequacy). We say that a set $S$ of connectives is **adequate** if for every $n \geq 1$, every truth function on $n$ variables can be expressed as the truth function as a propositional formula which only involves connectives from $S$ (and $n$ propositional variables).

The idea that this definition seeks to express is that a set is adequate if and only if for evern $n$, every propositional formula in $n$ variables is logically equivalent to a propositional formula that only contains those $n$ variables and connectives from the set in question. In other words, every propositional formula should admit an equivalent expression that does not contain any connectives

apart from those in the set in question. The reason this is expressed in terms of truth functions is that that is how logical equivalence is *defined* (cf. **??**).

We now have the first theorem of this module.

**Theorem 1.1.18.** *The set* $\{\neg, \wedge, \vee\}$ *is adequate.*

*Proof.* Fix some $n \geq 1$, and let $G : \{\mathbf{T}, \mathbf{F}\}^n \to \{\mathbf{T}, \mathbf{F}\}$ be a truth function. We have two cases.

**Case 1.** The first case is a trivial case. There are two trivial truth functions on $n$ variables, namely, the constrant truth functions that take the values **T** and **F** for all inputs. Truth is not something encoded 'naturally' into the connectives $\{\neg, \wedge, \vee\}$, but falsity is: the $\neg$ connective directly has to do with expressing falsity. Therefore, the trivial truth function that we will show can always be expressed in terms of the desired connectives is the one that is always false. We show this rigorously.

Assume that $G$ is identically **F**. Then, define the propositional formula $\phi : (p_1 \wedge (\neg p_1))$. Even defining it as a formula on $n$ variables, it is clear to see that its truth function $F_\phi$ is identically **F**. Therefore, $G = F_\phi$.[1]

**Case 2.** The second case will be the nontrivial case of when a truth function can take on both values **T** and **F**. The way we will show that $\{\neg, \wedge, \vee\}$ is adequate is by constructing a propositional formula in $n$ variables whose truth function is **T** whenever the one in question is **T**. We will do this by isolating the inputs that yield **T** and manipulating propositional variables in a way that corresponds to these inputs.

Assume that $G$ is not identically **T**. Then, list all $v \in \{\mathbf{T}, \mathbf{F}\}^n$ such that $G(v) = \mathbf{T}$. Since $\{\mathbf{T}, \mathbf{F}\}^n$ is a finite set, this list is finite, and we can number these $v_1, \ldots, v_r$. For each $1 \leq i \leq r$, denote

$$v_i = (v_{i1}, \ldots, v_{ir})$$

[1] Admittedly, we are using the Axiom of Extensionality here to define what it means for the two functions to be equal. We will ignore this technicality for now.

where $v_{ij} \in \{\mathbf{T}, \mathbf{F}\}$ is the $j$th component of $v_i$. Let $p_1, \ldots, p_n$ be propositional variables. Define propositional formulae $(q_{ij})_{1 \leq i \leq r, 1 \leq j \leq n}$ by

$$q_{ij} : \begin{cases} p_j & \text{if } v_{ij} = \mathbf{T} \\ (\neg p_j) & \text{if } v_{ij} = \mathbf{F} \end{cases}$$

Then, $q_{ij}$ has value $\mathbf{T}$ if and only if $p_j$ has value $v_{ij}$. The idea is to now construct a propositional formula that has value $\mathbf{T}$ if and only if $(p_1, \ldots, p_n)$ is one of the $v_i$.

First, we formalise the notion of the $(p_1, \ldots, p_n)$ taking the value of one of the $v_i$. The idea is to combine them using the $\wedge$ connective. Define propositional formulae $(\psi_i)_{1 \leq i \leq r}$ by

$$\psi_i : (q_{i1} \wedge \cdots \wedge q_{in})$$

Then, we have that for all $1 \leq i \leq r$ and $v \in \{\mathbf{T}, \mathbf{F}\}^n$,

$$F_{\psi_i}(v) = \mathbf{T} \iff q_{i1}, \ldots, q_{in} \text{ all have value } \mathbf{T} \iff \text{Each } p_j \text{ has value } v_{ij} \iff v = v_i$$

Next, we combine these $\psi_i$ so that the truth function of the resulting formula is $\mathbf{T}$ if and only if one of the $\psi_i$ is true, a fact that would be equivalent to the input of the truth function being precisely one of the $v_i$. We do this using the $\vee$ connective. Define the propositional formula

$$\vartheta : (\psi_1 \vee \cdots \vee \psi_r)$$

Then, for all $v \in \{\mathbf{T}, \mathbf{F}\}^n$, we have that

$$F_\vartheta(v) = \mathbf{T} \iff \text{One of the } \psi_i \text{ is true} \iff v \text{ is precisely equal to one of the } v_i$$

In partiular, we have that $F_\vartheta(v) = \mathbf{T}$ if and only if $G(v) = \mathbf{T}$ for all $v \in \{\mathbf{T}, \mathbf{F}\}^n$. Then, by **??**, we are done. □

Before illustrating the point of the above theorem, we make an important definition.

**Definition 1.1.19** (Disjunctive Normal Form)**.** When a propositional formula is expressed only in terms of propositional variables and the set $\{\neg, \wedge, \vee\}$ of connectives, it is said to be in **disjunctive normal form**, which we abbreviate to **DNF**.

What **??** then tells us is that every propositional formula is expressible in DNF.

**Corollary 1.1.20.** *For every propositional formula in $n$ variables, there exists a logically equivalent propositional formula in $n$ variables that is in DNF.*

*Proof.* We know that every propositional formula admits a truth function. For any propositional formula in $n$ variables, we can apply **??** to its truth function. Then, unfolding the definition of adequacy yields the desired result. □

**Example 1.1.21.** Let $p_1$ and $p_2$ be propositional variables. Consider the propositional formula $\chi : ((p_1 \to p_2) \to (\neg p_2))$. We can see that $F_\chi(v) = \mathbf{T}$ only if $v = (\mathbf{T}, \mathbf{F})$ or $v = (\mathbf{F}, \mathbf{F})$. Therefore, the DNF of $\chi$ is

$$((p_1 \wedge (\neg p_2)) \vee ((\neg p_1) \wedge (\neg p_2)))$$

It turns out that $\{\neg, \wedge, \vee\}$ is not the only adequate set of connectives.

**Example 1.1.22** (Adequate Sets)**.** The following sets of connectives are adequate.

(i) $\{\neg, \vee\}$

(ii) $\{\neg, \wedge\}$

(iii) $\{\neg, \to\}$

The way we can prove this is by simplifying each case using **??**. Fix propositional variables $p_1, p_2$.

(i) It suffices to show that $p_1 \wedge p_2$ can be expressed using $\neg$ and $\vee$. Indeed,

$$(p_1 \wedge p_2) \text{ is logically equivalent to } (\neg((\neg p_1) \vee (\neg p_2)))$$

(ii) It suffices to show that $p_1 \vee p_2$ can be expressed using $\neg$ and $\wedge$. Indeed,

$$(p_1 \vee p_2) \text{ is logically equivalent to } (\neg((\neg p_1) \wedge (\neg p_2)))$$

(iii) By Case **??**, it suffices to show that $p_1 \vee p_2$ can be expressed in terms of $\neg$ and $\rightarrow$. Indeed,

$$(p_1 \vee p_2) \text{ is logically equivalent to } ((\neg p_1) \rightarrow p_2)$$

There are also sets of connectives that are not adequate.

**Non-Example 1.1.23** (Inadequate Sets). The following sets are not adequate.

(i) $\{\wedge, \vee\}$

(ii) $\{\neg, \leftrightarrow\}$

The way we can prove this is by constructing truth functions that cannot be realised by combining propositional variables using only the connectives in the above sets.

(i) No truth function that is identically false can be realised. For that matter, no truth function that maps an input whose every component is **T** to **F** can be realised. Formally, consider any propositional formula $\phi$ built exclusively using a finite set of propositional variables and the connectives $\wedge$ and $\vee$. One can show, by induction on the number of connectives in $\phi$, that $F_\phi(\mathbf{T}, \ldots, \mathbf{T}) = \mathbf{T}$. Since this is true of any $\phi$, a truth function mapping an input of the form $(\mathbf{T}, \ldots, \mathbf{T})$ to **F** is not the truth function of a propositional formula that only includes $\wedge$ and $\vee$.

(ii) No truth function that is identcally true can be realised.

It turns out that there is one connective with a rather astounding adequacy property.

**Definition 1.1.24** (The NOR Connective). Define the **NOR connective**, denoted $\downarrow$, via the following truth table in propositional variables $p$ and $q$.

| $p$ | $q$ | $(p \downarrow q)$ |
|---|---|---|
| **T** | **T** | **F** |
| **T** | **F** | **F** |
| **F** | **T** | **F** |
| **F** | **F** | **T** |

Informally, NOR corresponds to "neither ... nor ...". Formally, we have the following.

**Lemma 1.1.25.** *For all propositional variables $p$ and $q$, the DNF of $(p \downarrow q)$ is given by $((\neg p) \wedge (\neg q))$. In particular, we have that $(p \downarrow q)$ is logically equivalent to $((\neg p) \wedge (\neg q))$.*

We do not write out a proof, as it merely involves comparing truth tables.

**Example 1.1.26** (An Adequate Set with One Connective)**.** It turns out that $\{\downarrow\}$ is connective. Indeed, for propositional variables $p$ and $q$, we have

1. $(p \downarrow p)$ is logically equivalent to $(\neg p)$.
2. $((p \downarrow p) \downarrow (q \downarrow q))$ is logically equivalent to $(p \wedge q)$.

So far, we have been studying *meaning*, in the form of truth functions, but have yet to formally define *what* we are allowed to express that *has* meaning within the propositional paradigm. In other words, we have been studying **semantics** but have yet to define the **syntax** of propositional logic. We will do this in the next section.

## 1.2 A Formal System for Propositional Logic

The motivating idea for everything we shall do in this section is to try and generate *all tautologies* from certain 'basic assumptions', known as **axioms**, using certain **deduction rules**. Together, these will form a **formal system for propositional logic**.

### 1.2.1 Formal Deduction Systems

This subsection gives a very general definition of a formal deduction system, which allows us to construct 'proofs' in a sense more general than propositional logic. We will then specialise this to propositional logic. The material here was **not covered in lectures**, and is based on [**2018LecNotes**].

The first ingredient of a formal system is the set of symbols we are allowed to use within it.

**Definition 1.2.1** (Alphabet)**.** An **alphabet** is a nonempty list of symbols.

For the remainder of this subsection, fix an alphabet $A$. $A$ is not useful on its own: we need to be

able to combine elements of $A$ with each other.

**Definition 1.2.2** (Strings)**.** A **string** is any finite sequence of of elements of $A$.

We do not always want all strings to be useful to us, as we shall see in the case of propositional logic. Our next ingredient in the construction of a formal system is the precise set of strings we are allowed to use.

**Definition 1.2.3** (Formulae)**.** A set of **formulae** is a non-empty subset of a set of strings in $A$.

For the remainder of this subsection, fix a set $\mathcal{F}$ of formulae. It will be important that to distinguish the formulae that will serve as our most basic assumptions and those that will be derived from them.

**Definition 1.2.4** (Axioms)**.** A set of **axioms** is a subset of $\mathcal{F}$.

For the remainder of this subsection, fix a set $\mathcal{A}$ of axioms. We now need to be able to generate formulae from the distinguished ones (ie, axioms).

**Definition 1.2.5** (Deduction Rules)**.** A **deduction rule** is a function that takes in a finite list of formulae in $\mathcal{F}$ and outputs a formula in $\mathcal{F}$.

Together, these ingredients form a **formal deduction system**.

**Definition 1.2.6** (Formal Deduction System)**.** A **formal deduction system** $\Sigma$ is a tuple $(A, \mathcal{F}, \mathcal{A}, \mathcal{D})$, where

1. $A$ is an alphabet
2. $\mathcal{F}$ is a set of formulae in $A$
3. $\mathcal{A}$ is a set of axioms contained in $\mathcal{F}$
4. $\mathcal{D}$ is a set of deduction rules on $\mathcal{F}$

For the remainder of this subsection, fix a formal deduction system $\Sigma = (A, \mathcal{F}, \mathcal{A}, \mathcal{D})$. We can now define what it means to reason in this system.

**Definition 1.2.7** (Proof)**.** A **proof** in $\Sigma$ is a finite sequence of formulae $\phi_1, \ldots, \phi_n \in \mathcal{F}$ such that each $\phi_i$ is either an axiom in $\mathcal{A}$ or is obtained from $\phi_1, \ldots, \phi_{i-1}$ using a deduction rule in $\mathcal{D}$.

We want to be able to isolate the formulae that are 'proven' in this manner.

**Definition 1.2.8** (Theorem)**.** The last formula $\phi_i$ that is contained in a proof $\phi_1, \ldots, \phi_n$ is called a **theorem** of $\Sigma$. We write $\vdash_\Sigma \phi$ to denote that $\phi$ is a theorem of $\Sigma$.

There is a direct correspondence between the way we intuitively think about the proofs and the way we view them here.

**Convention.** We say that proof $P$ in $\Sigma$ is a **proof of a theorem** $\phi$ if $\phi$ is the last formula in the sequence of formulae that make up $P$.

Note the following trivial result.

**Lemma 1.2.9.** *Any axiom is a theorem.*

*Proof.* An axiom $\phi$ is a finite string of formulae consisting of a single formula, namely, itself. Therefore, the proof consisting only of $\phi$ is a proof of $\phi$. □

Finally, we mention a condition on formal deduction systems that lends them to computer-based verification.

**Definition 1.2.10** (Recursive Formal Deduction Systems)**.** Suppose there exists an algorithm that can test whether a string is a formula and whether it is an axiom. Then, $\Sigma$ is called a **recursive formal deduction system**.

The point of the above definition is that in recursive systems, a computer can systematically generate all possible proofs in $\Sigma$ by checking whether every possible formula is an axiom or is derived from axioms using deduction rules. In this case, the validity of any proof can be checked, because each formula in its constituent sequence can be checked.

We now close our discussion on general formal deduction systems. We will now specialise to propositional logic.

### 1.2.2 Constructing a Formal System Propositional Logic

The objective of this subsection is to define the formal system for propositional logic that we will use in this module. We want this formal system to correspond to our intuition in a very specific manner: we would like the theorems in this system to be precisely the tautologies. In other words, we want to construct a system in which theorems are precisely 'statements that are true'.

If we re-examine **??**, we observe something rather interesting: *the definition does not mention truth*! The *validity* of a theorem only has to do with the *deduction rules* of the formal system in which it lives. Therefore, we want to define axioms and deduction rules for propositional logic such that the axioms are 'true' (ie, are *tautologies*, as defined in **??**) and the deduction rules preserve truth. This is an important motivation not only for the choice of deduction rule but also the chocie of (adequate) set of connectives we use in our formal system.

We begin by defining the alphabet.

**Definition 1.2.11** (Alphabet for Propositional Logic)**.** The **alphabet** for propositional logic consists of the following symbols:

1. **Variables** $p_1, p_2, p_3, \ldots$
2. **Connectives** $\neg, \rightarrow$
3. **Punctuation** $(,)$

Next, we define the formulae of propositional logic.

**Definition 1.2.12** (Formulae of Propositional Logic)**.** We define the **formulae** of propositional logic in the following manner.

1. Any variable $p_i$ is a formula.
2. If $\phi$ is a formula, then $(\neg\phi)$ is a formula.
3. If $\phi$ and $\psi$ are formulae, then $(\phi \rightarrow \psi)$ is a formula.
4. Any formula arises from a finite number of applications of the above rules.

Next, we define the axioms of propositional logic. These do appear to be a bit unusual when expressed purely in symbols, but we can interpret them in plain English as well.

**Definition 1.2.13** (Axioms of Propositional Logic)**.** Let $\phi, \psi, \chi$ be formulae in propositional logic. The **axioms** of propositional logic in $\phi$, $\psi$ and (where present) $\chi$ are the following.

(A1) $(\phi \to (\psi \to \phi))$

(A2) $((\phi \to (\psi \to \chi)) \to ((\phi \to \psi) \to (\phi \to \chi)))$

(A3) $(((\neg\phi) \to (\neg\psi)) \to (\psi \to \phi))$

We can interpret these axioms as follows.

?? If $\phi$ is true, then $\phi$ is implied by any statement.

?? If $(\psi \to \chi)$ is true (under some assumption $\phi$), then to prove $\chi$ (assuming $\phi$), it suffices to prove $\psi$ (assuming $\phi$).

?? An implication $(\phi \to \psi)$ is implied by its contrapositive $((\neg\phi) \to (\neg\psi))$.

Indeed, we can see that these axioms are all tautologies: they are always true. This is exactly what we want from axioms.

*Remark.* It must be stressed that ?? and ?? are axioms *in* $\phi, \psi, \chi$ and ?? is an axiom *in* $\phi, \psi$. There are therefore infinitely many axioms, one for each choice of $\phi$, $\psi$, and, where applicable, $\chi$. In fact, ??-?? are sometimes referred to as **axiom schemes** instead of just **axioms** to underscore this.

Finally, we define the sole deduction rule of propositional logic.

**Definition 1.2.14** (Deduction Rule of Propositional Logic)**.** Fix formulae $\phi$ and $\psi$. The **deduction rule** on $\phi$ and $\psi$ states:

(MP) From $\phi$ and $(\phi \to \psi)$, we can deduce $\psi$.

This rule is known as **modus ponens**. We will abbreviate this to ??.

We can now define the formal deduction system for propositional logic.

**Definition 1.2.15** (Formal Deduction System for Propositional Logic)**.** The **formal deduction system for propositional logic** is the tuple $\mathcal{L} = (A, \mathcal{F}, \mathcal{A}, \mathcal{D})$, where

1. $A$ is the alphabet for propositional logic consisting of countably many propositional variables, as defined in Definition ??.
2. $\mathcal{F}$ is the set of formulae of propositional logic constructed in terms of the connectives $\neg$ and $\rightarrow$, as defined in Definition ??.
3. $\mathcal{A}$ is the set of axioms ??-?? of propositional logic defined in Definition ??.
4. $\mathcal{D}$ is the deduction rule ?? defined in Definition ??.

We are now ready to write formal proofs in $\mathcal{L}$.

**Example 1.2.16.** Suppose $\phi$ is any $\mathcal{L}$-formula. Then, $(\phi \rightarrow \phi)$ is a theorem of $\mathcal{L}$, ie,

$$\vdash_{\mathcal{L}} (\phi \rightarrow \phi) \tag{1.2.1}$$

It is not obvious how to prove this statement using only ??-?? and ??. But it is possible. We present a formal proof that consists of a sequence of five formulas, each either axiomatic or deduced from two previous ones using modus ponens.

*Formal proof in $\mathcal{L}$.*

1. $(\phi \rightarrow ((\phi \rightarrow \phi) \rightarrow \phi))$
   *Justification: We apply ?? to $\phi, (\phi \rightarrow \phi), \phi$.*
2. $((\phi \rightarrow ((\phi \rightarrow \phi) \rightarrow \phi)) \rightarrow ((\phi \rightarrow (\phi \rightarrow \phi)) \rightarrow (\phi \rightarrow \phi)))$
   *Justification: We apply ?? to $\phi, (\phi \rightarrow \phi), \phi$.*
3. $((\phi \rightarrow (\phi \rightarrow \phi)) \rightarrow (\phi \rightarrow \phi))$
   *Justification: We apply ?? to steps 2 and 1.*
4. $(\phi \rightarrow (\phi \rightarrow \phi))$
   *Justification: We apply ?? to $\phi, \phi, \phi$.*
5. $(\phi \rightarrow \phi)$
   *Justification: We apply ?? to steps 3 and 4.*

□

We need a little more machinery before we can prove that theorems in $\mathcal{L}$ are precisely the tautologies. We will develop this in the next subsection.

### 1.2.3 Deductions in $\mathcal{L}$

In this subsection, we will develop some machinery as well as notation to deal with the concept of deduction. However, before we proceed any further, we will give a precise definition of what a deduction is.

Throughout this subsection, fix a set of $\mathcal{L}$-formulas $\Gamma$.

**Definition 1.2.17** (Deduction). A **deduction** from $\Gamma$ is a finite sequence of $\mathcal{L}$-formulae $\phi_1, \ldots, \phi_n$ such that for all $1 \leq i \leq n$, either

- $\phi_i$ is an axiom of $\mathcal{L}$,
- $\phi_i$ is a formula in $\Gamma$, or
- $\phi_i$ is obtained from $\phi_1, \ldots, \phi_{i-1}$ using modus ponens.

Deductions end in consequences.

**Definition 1.2.18** (Consequence). We say a formula $\phi$ is a **consequence** of $\Gamma$ if there exists a deduction from $\Gamma$ ending in $\phi$. In this case, we write $\Gamma \vdash_{\mathcal{L}} \phi$.

Note that $\Gamma$ can be empty. In this case, instead of writing $\emptyset \vdash_{\mathcal{L}} \phi$, we write $\vdash_{\mathcal{L}} \phi$. Therefore, the theorems in $\mathcal{L}$ are precisely those formulae that are not consequences of other formulae.

For the latest version of these notes, visit https://thefundamentaltheor3m.github.io/LogicNotes/LastLocallyCompiled.pdf. For any suggestions or corrections, please feel free to fork and make a pull request to my repository.