MATH70132: Mathematical Logic

Lecturer: David Evans

Notes written by 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 (1.0.1) 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

Broadly speaking, the study of propositional logic involves studying its two major components: **syntax** and **semantics**. While the most formal approach to the study of propositional logic is to study them in that order, in this module, we study semantics before syntax, because while syntax must precede semantics, semantics can serve as motivation for the syntactic choices we make when defining a 'formal system' for propositional logic (see Section 1.2). In this section, we will study the semantics of propositional logic.

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 ($\to$)
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 $\to$ 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 $\to$ 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 ϕ and ψ 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 ϕ determines the truth value for ϕ in a **unique** manner, using the exhaustive definitions of the connectives given in Definition 1.1.2. We often express all possible values of a propositional formula in a **truth table**, much like we did in Definition 1.1.2 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 ϕ 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 ϕ 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 ϕ be a propositional formula whose variables are $p_1, \ldots, p_n$. We can associate to ϕ 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 ϕ 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** ϕ, denoted F_ϕ.

We can now construct a truth function for the example we saw at the very beginning involving Mr and Mrs Jones (cf. (1.0.1)).

Example 1.1.8. `sorry`

We see something quite remarkable here: the truth function of the propositional formula defined in (1.0.1) maps every possible input to **T**! We have a special term for this.

Definition 1.1.9 (Tautology). A propositional formula ϕ is a **tautology** if its truth function F_ϕ 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 ψ and χ 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 ψ and χ be formulae in $p_1, \ldots, p_n$. Then, ψ and χ 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 ϕ is a propositional formula with variables $p_1, \ldots, p_n$. Let $\phi_1, \ldots, \phi_n$ be propositional formulae. Denote by ϑ the result of substituting each p_i with ϕ_i in ϕ. Then,*

(i) ϑ is a propositional formula.

(ii) if ϕ is a tautology, so is ϑ.

(iii) the truth function of ϑ is the result of composing the truth function of ϕ 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 ϕ_1 and ϕ_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 ϕ' such that $(\phi_1 \to (\neg \phi_1))$ is a tautology: for example, we can define ϕ' 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. Definition 1.1.10).

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_ϕ 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.

[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.

Assume that G is not identically $\mathbf{T}$. 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 $v_1, \dots, v_r$. For each $1 \leq i \leq r$, denote

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

where $v_{ij} \in \{\mathbf{T}, \mathbf{F}\}$ is the jth component of v_i. Let $p_1, \dots, p_n$ 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 $(p_1, \dots, p_n)$ is one of the v_i.

First, we formalise the notion of the $(p_1, \dots, p_n)$ 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

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

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

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

Next, we combine these ψ_i so that the truth function of the resulting formula is $\mathbf{T}$ if and only if one of the ψ_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

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

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 Lemma 1.1.6, 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 Theorem 1.1.18 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 Theorem 1.1.18 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 χ 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 Theorem 1.1.18. 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 (i), it suffices to show that $p_1 \vee p_2$ can be expressed in terms of $\neg$ and $\to$. Indeed,

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

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 ϕ 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 ϕ, that $F_\phi(\mathbf{T}, \ldots, \mathbf{T}) = \mathbf{T}$. Since this is true of any ϕ, 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 [**LecNotes2018**].

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** Σ 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 Σ is a finite sequence of formulae $\phi_1, \ldots, \phi_n \in \mathcal{F}$ such that each ϕ_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 ϕ_i that is contained in a proof $\phi_1, \ldots, \phi_n$ is called a **theorem** of Σ. We write $\vdash_\Sigma \phi$ to denote that ϕ is a theorem of Σ.

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 Σ is a **proof of a theorem** ϕ if ϕ 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 ϕ is a finite string of formulae consisting of a single formula, namely, itself. Therefore, the proof consisting only of ϕ is a proof of ϕ. □

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, Σ 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 Σ 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 Definition 1.2.8, 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 Definition 1.1.9) 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 ϕ is a formula, then $(\neg\phi)$ is a formula.
3. If ϕ and ψ 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 ϕ, ψ, χ be formulae in propositional logic. The **axioms** of propositional logic in ϕ, ψ and (where present) χ 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.

(A1) If ϕ is true, then ϕ is implied by any statement.

(A2) If $(\psi \to \chi)$ is true (under some assumption ϕ), then to prove χ (assuming ϕ), it suffices to prove ψ (assuming ϕ).

(A3) 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 (A1) and (A2) are axioms *in* ϕ, ψ, χ and (A3) is an axiom *in* ϕ, ψ. There are therefore infinitely many axioms, one for each choice of ϕ, ψ, and, where applicable, χ. In fact, (A1)-(A3) 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 ϕ and ψ. The **deduction rule** on ϕ and ψ states:

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

This rule is known as **modus ponens**. We will abbreviate this to (MP).

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 $\mathbf{L} = (A, \mathcal{F}, \mathcal{A}, \mathcal{D})$, where

1. A is the alphabet for propositional logic consisting of countably many propositional variables, as defined in Definition 1.2.11.
2. $\mathcal{F}$ is the set of formulae of propositional logic constructed in terms of the connectives $\neg$ and $\rightarrow$, as defined in Definition 1.2.12.
3. $\mathcal{A}$ is the set of axioms (A1)-(A3) of propositional logic defined in Definition 1.2.13.
4. $\mathcal{D}$ is the deduction rule (MP) defined in Definition 1.2.14.

We are now ready to write formal proofs in **L**.

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

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

It is not obvious how to prove this statement using only (A1)-(A3) and (MP). 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 **L**.

1. $(\phi \rightarrow ((\phi \rightarrow \phi) \rightarrow \phi))$

   Justification: *We apply (A1) 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 (A2) to* $\phi, (\phi \rightarrow \phi), \phi$.
3. $((\phi \rightarrow (\phi \rightarrow \phi)) \rightarrow (\phi \rightarrow \phi))$

   Justification: *We apply (MP) to steps 2 and 1.*
4. $(\phi \rightarrow (\phi \rightarrow \phi))$

   Justification: *We apply (A1) to* ϕ, ϕ, ϕ.
5. $(\phi \rightarrow \phi)$

   Justification: *We apply (MP) to steps 3 and 4.*

□

We need a little more machinery before we can prove that theorems in **L** are precisely the tautologies.

We will develop this in the next subsection.

1.2.3 Deductions in 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 **L**-formulas Γ.

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

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

Essentially, deductions capture the notion of proofs, with an additional layer of flexibility coming from the fact that we are allowed to have assumptions in Γ that go beyond merely the axioms of **L**. Just as proofs end in theorems, deductions end in consequences.

Definition 1.2.18 (Consequence). We say a formula ϕ is a **consequence** of Γ if there exists a deduction from Γ ending in ϕ. In this case, we write $\Gamma \vdash_{\mathbf{L}} \phi$. If ϕ is not a consequence of Γ, we write $\Gamma \nvdash_{\mathbf{L}} \phi$.

The relationship between proofs and deductions/theorems and consequences is that the former correspond to the case where Γ is empty.

Convention. Instead of writing $\emptyset \vdash_{\mathbf{L}} \phi$, we write $\vdash_{\mathbf{L}} \phi$.

Therefore, the theorems in **L** are precisely those formulae that are not consequences of other formulae.

We state a rather trivial fact.

Lemma 1.2.19. *Let $\Delta \subseteq \Gamma$. Then, for all* **L***-formulae ϕ, if $\Delta \vdash_{\mathbf{L}} \phi$, then $\Gamma \vdash_{\mathbf{L}} \phi$.*

We do not prove this fact. In fact, we will often use it without mentioning it explicitly.

There is an important theorem that relates the notion of consequence with that of implication and provides us with a framework of reasoning with statements about consequence.

Theorem 1.2.20 (The Deduction Theorem)**.** *Let ϕ, ψ be* **L***-formulae. Then,*

$$\Gamma \cup \{\psi\} \vdash_{\mathbf{L}} \phi \text{ if and only if } \Gamma \vdash_{\mathbf{L}} (\psi \to \phi)$$

Proof. We begin by showing that $\Gamma \cup \{\psi\} \vdash_{\mathbf{L}} \phi$ implies $\Gamma \vdash_{\mathbf{L}} (\psi \to \phi)$.

Assume that $\Gamma \cup \{\psi\} \vdash_{\mathbf{L}} \phi$. This states that there is a deduction from $\Gamma \cup \{\psi\}$ to ϕ in the formal system **L**. By Definition 1.2.17, we know that such a deduction must be finite. We prove that for all $n \in \mathbb{N}$, if $\Gamma \cup \{\psi\} \vdash_{\mathbf{L}} \phi$ is a deduction of length n, then $\Gamma \vdash_{\mathbf{L}} (\psi \to \phi)$. We proceed by induction on n.

Base Case: $n = 1$. We know that ϕ is either an axiom or is lies in Γ or is ψ. In the first two cases, we have

$$\Gamma \vdash_{\mathbf{L}} \phi$$

in which case the axiom (A1) guarantees that

$$\Gamma \vdash_{\mathbf{L}} (\phi \to (\psi \to \phi))$$

Applying (MP) to the deductions above then gives us that

$$\Gamma \vdash_{\mathbf{L}} (\psi \to \phi)$$

as required. In the third case, where ϕ is ψ, we have

$$\Gamma \vdash_{\mathbf{L}} (\psi \to \psi)$$

by the same argument as in Example 1.2.16.

Inductive Case. sorry

□

From this theorem, we can deduce that the implication connective $\rightarrow$ is transitive.

Corollary 1.2.21 (Hypothetical Syllogism). *Suppose* ϕ, ψ, χ *are* **L**-*formulae such that* $\Gamma \vdash_{\mathbf{L}} (\phi \rightarrow \psi)$ *and* $\Gamma \vdash_{\mathbf{L}} (\psi \rightarrow \chi)$. *Then,* $\Gamma \vdash_{\mathbf{L}} (\phi \rightarrow \chi)$.

Proof. We show that there is a deduction of χ from $\Gamma \cup \{\phi\}$. We give a formal proof in **L**.

1. $\Gamma \vdash_{\mathbf{L}} \phi \rightarrow \psi$

   Justification: *Assumption.*

2. $\Gamma \vdash_{\mathbf{L}} \psi \rightarrow \chi$

   Justification: *Assumption.*

3. $\Gamma \cup \{\phi\} \vdash_{\mathbf{L}} \psi$

   Justification: *Applying the Deduction Theorem to Step 1.*

4. $\Gamma \cup \{\phi\} \vdash_{\mathbf{L}} \psi \rightarrow \chi$

   Justification: *Applying Lemma 1.2.19 to Step 2.*

5. $\Gamma \cup \{\phi\} \vdash_{\mathbf{L}} \chi$

   Justification: *Applying (MP) to Step 4 and Step 3.*

6. $\Gamma \vdash_{\mathbf{L}} \phi \rightarrow \chi$

   Justification: *Applying the Deduction Theorem to Step 5.*

The last step is the desired deduction. □

Example 1.2.22 (A Few Theorems in **L**). Suppose ϕ and ψ are **L**-formulae. Then,

(a) $\vdash_{\mathbf{L}} ((\neg\psi) \rightarrow (\psi \rightarrow \phi))$

(b) $\{(\neg\psi), \psi\} \vdash_{\mathbf{L}} \phi$

(c) $\vdash_{\mathbf{L}} (((\neg\phi) \rightarrow \phi) \rightarrow \phi)$

are all theorems in **L**.

Proof.

(a) Problem Sheet 1, Question 6. sorry

(b) We apply (MP) to (a) twice, the first time using our assumption $(\neg\psi)$ and the second time using our assumption ψ.

(c) Suppose χ is any formula. Then, from (b), we can see that

$$\{(\neg\phi), ((\neg\phi) \to \phi)\} \vdash_{\mathbf{L}} \chi \tag{1.2.2}$$

because we would be able to apply (MP) to the assumptions to deduce both ϕ and $(\neg\phi)$.

Now, let α be an axiom and let χ be $(\neg\alpha)$. Then, (1.2.2) tells us that

$$\{(\neg\phi), ((\neg\phi) \to \phi)\} \vdash_{\mathbf{L}} (\neg\alpha)$$

sorry

□

1.3 Important Properties of L

In this section, we prove important properties about **L**, with our first major goal being to prove that the theorems in **L** are precisely the tautologies.

1.3.1 Propositional Valuations

Definition 1.3.1 (Propositional Valuation). A **propositional valuation v** is an assignment of truth values to propositions $p_1, p_2, \ldots$, so that

$$\mathbf{v}(p_i) \in \{\mathbf{T}, \mathbf{F}\}$$

for all $i \in \mathbb{N}$.

Using the truth table rules by which we defined the connectives $\to$ and $\neg$ of **L** (cf. Definition 1.1.2), we can assign a truth value $\mathbf{v}(\phi)$ to any **L**-formula ϕ.

Lemma 1.3.2 (Behaviour of Propositional Valuations). *Let ϕ, ψ be* **L***-formulae. Then,*

1. $\mathbf{v}((\neg\phi)) \neq \mathbf{v}(\phi)$
2. $\mathbf{v}(\phi \to \psi) = \mathbf{F}$ if and only if $\mathbf{v}(\phi) = \mathbf{T}$ and $\mathbf{v}(\psi) = \mathbf{F}$

We do not prove these results.

We can now be precise about what a tautology is.

Definition 1.3.3 (Tautology). An **L**-formula ϕ is a **tautology** if $\mathbf{v}(\phi) = \mathbf{T}$ for all propositional valuations $\mathbf{v}$.

We are now ready to prove that **L** is sound.

1.3.2 Soundness

Definition 1.3.4 (Soundness). We say a formal system is **sound** if every theorem in it is a tautology.

Now, the much-awaited result.

Theorem 1.3.5 (Soundness of **L**). *The formal system* **L** *of propositional logic is sound.*

Proof. Let ϕ be a theorem in **L**. We prove that ϕ is a tautology by performing induction on the length of ϕ. The base case involves proving that the axioms (A1)-(A3) are tautologies, and the inductive case involves proving that the modus ponens deduction rule (MP) preserves tautologies.

While these proofs can be done manually, there is a rather clever trick that can be used in the case of (A2). This is given in [**LecNotes2018**]. `sorry` □

We have a more general version of this result.

Theorem 1.3.6 (A Generalisation of Soundness). *Let Γ be a set of* **L***-formulae, and let ϕ be an* **L***-formula such that $\Gamma \vdash_{\mathbf{L}} \phi$. If $\mathbf{v}(\psi) = \mathbf{T}$ for all $\psi \in \Gamma$, then $\mathbf{v}(\phi) = \mathbf{T}$.*

Proof. The proof is actually identical to that of Theorem 1.3.5, but with Γ being a part of the base case. In this case, the proof follows from the assumption that **v** is true on Γ. $\square$

Now that we have shown that **L** is sound, we show that **L** has other important properties that will enable us to reason in **L** the way we would like to.

1.3.3 Consistency

For the remainder of this section, fix a set Γ of **L**-formulae.

Definition 1.3.7 (Consistency). Γ is **consistent** if there is no **L**-formula ϕ such that $\Gamma \vdash_{\mathbf{L}} \phi$ and $\Gamma \vdash_{\mathbf{L}} (\neg\phi)$.

Soundness tells us something about the consistency of the empty set in **L**, which we will refer to more generally as the **consistency of L**.

Theorem 1.3.8 (Consistency of **L**). *There is no* **L***-formula ϕ such that $\vdash_{\mathbf{L}} \phi$ and $\vdash_{\mathbf{L}} (\neg\phi)$.*

Proof. This follows from Theorem 1.3.5: if such an **L**-formula ϕ did exist, both ϕ and $(\neg\phi)$ would need to be tautologies, violating Lemma 1.3.2. Therefore, such an **L**-formula cannot exist. $\square$

We have a more general result.

Proposition 1.3.9. *Suppose Γ is a consistent set of* **L***-formulae. Let ϕ be an* **L***-formula that is* not *a consequence of Γ. Then, $\Gamma \cup \{(\neg\phi)\}$ is consistent.*

Proof. Suppose that $\Gamma \cup \{(\ neg\phi)\}$ is not consistent. Then, there exists a formula ψ such that

$$\Gamma \cup \{(\neg\phi)\} \vdash_{\mathbf{L}} \psi \tag{1.3.1}$$

$$\Gamma \cup \{(\neg\phi)\} \vdash_{\mathbf{L}} (\neg\psi) \tag{1.3.2}$$

We can apply the Deduction Theorem (Theorem 1.2.20) to (1.3.2) to deduce that

$$\Gamma \vdash_{\mathbf{L}} ((\neg\phi) \to (\neg\psi)) \tag{1.3.3}$$

Then, applying (A3) to (1.3.3), we have

$$\Gamma \vdash_{\mathbf{L}} (\psi \to \phi) \tag{1.3.4}$$

In similar fashion, we can apply the Deduction Theorem followed by (A3) to (1.3.1) to deduce that

$$\Gamma \vdash_{\mathbf{L}} ((\neg\psi) \to \phi) \tag{1.3.5}$$

sorry □

1.3.4 Completeness

We begin by making an observation about **L**: in general, it is not true that for any given **L**-formula, either it or its negation is a theorem.

Definition 1.3.10 (Completeness)**.** Let Γ be a consistent set of **L**-formulae. We say Γ is **complete** if for all **L**-formulae ϕ, either $\Gamma \vdash_{\mathbf{L}} \phi$ or $\Gamma \vdash_{\mathbf{L}} (\neg\phi)$.

It turns out that all consistent sets of **L**-formulae are contained in complete sets of formulae.

Theorem 1.3.11 (Lindenbaum's Lemma)**.** *Suppose Γ is a consistent set of* **L***-formulae. Then, there exists a complete set of* **L***-formulae Γ^* such that $\Gamma \subseteq \Gamma^*$.*

Proof. The idea is to construct Γ^* by brute force. We know that the alphabet of **L** is countable. Therefore, so is the set of all possible **L**-formulae. We can list them as $\phi_0, \phi_1, \ldots$. We perform the fomllowing inductive construction: let $\Gamma_0 := \Gamma$ and for all $n \geq 0$, define

$$\Gamma_{n+1} := \begin{cases} \Gamma_n & \text{if } \Gamma_n \vdash_{\mathbf{L}} \phi_n \\ \Gamma_n \cup \{(\neg\phi_n)\} & \text{if } \Gamma_n \nvdash_{\mathbf{L}} \phi_n \end{cases} \tag{1.3.6}$$

One can perform induction on n to show that for all $n \geq 0$,

- $\Gamma_n \subseteq \Gamma_{n+1}$ (using purely (1.3.6))
- Γ_n is consistent (using Proposition 1.3.9)

Define

$$\Gamma^* := \bigcup_{n=0}^{\infty} \Gamma_n$$

Before we can show that Γ^* is complete, we need to show that it is consistent: see Definition 1.3.10. To that end, suppose there exists some **L**-formula ϕ such that $\Gamma^* \vdash_{\mathbf{L}} \phi$ and $\Gamma^* \vdash_{\mathbf{L}} (\neg\phi)$. This means there is a finite sequence of deductions from Γ^* and the axioms of **L** using only (MP) that ends in ϕ and another one that ends in $(\neg\phi)$. There exist $i, j \in \mathbb{N}$ such that these sequences are contained in Γ_i and Γ_j respectively, because the Γ_ns represent all possible deductions that can be made from Γ (and the axioms). Since we have a chain of inclusions, we have either $\Gamma_i \subseteq \Gamma_j$ or $\Gamma_j \subseteq \Gamma_i$. In either case, we have a contradiction, because Γ_i and Γ_j are both consistent, but we would be able to deduce both ϕ and $(\neg\phi)$ from one of them. Therefore, Γ^* is consistent too.

Finally, we show that Γ^* is complete. Let ϕ be a formula. By the enumeration above, we know that $\phi = \phi_n$ for some $n \in \mathbb{N}$. We know either $\Gamma_n \vdash_{\mathbf{L}} \phi_n$ or $\Gamma_n \nvdash_{\mathbf{L}} \phi_n$. If the former is true, we have that $\Gamma^* \vdash_{\mathbf{L}} \phi_n$ and we are done. Else, by our construction in (1.3.6), we have that $\Gamma_{n+1} \vdash_{\mathbf{L}} (\neg\phi)$, and therefore, $\Gamma^* \vdash_{\mathbf{L}} (\neg\phi)$. Therefore, from Γ^*, we can deduce either ϕ or $(\neg\phi)$, making Γ^* complete, as required. □

Going forward, we will adopt the following notation.

Convention. Let Γ be a consistent set of **L**-formulae. We will denote by Γ^* the complete set of **L**-formulae containing Γ as given in Theorem 1.3.11.

It turns out that the construction allows us to prove the existence of an important kind of valuation.

Proposition 1.3.12. *Let Γ be a consistent set of* **L***-formulae. Then, there exists a propositional valuation* $\mathbf{v}$ *such that*

$$\mathbf{v}(\phi) = \mathbf{T} \text{ if and only if } \Gamma^* \vdash_{\mathbf{L}} \phi$$

Proof. sorry □

Chapter 2

First-Order Logic

The plan for this part of the module is to examine Predicate Logic, or First-Order Logic. We will examine the following, though the distinction between semantics and syntax shall not be as pronounced as it is below:

1. Semantics:
   (a) First-order structures
   (b) First-order languages and the corresponding formulae
2. Syntax:
   (a) A formal system for first-order logic
   (b) Gödel's Completeness Theorem, which tells us that the theorems of the formal system are the "logically valid formulae"

Before we can begin the study of first-order logic, we need to make several important definitions and introduce the notation we will use for the remainder of this chapter. We will begin by studying structures and first-order languages. We will then relate ideas from propositional logic to ideas in first-order logic. Finally, we will build a formal system for first-order logic and give a rigorous syntactic foundation for the ideas we discuss.

2.1 Languages, Structures and Interpretations

In this section, we discuss the notion of first-order languages and their interpretations in first-order structures. While this is primarily a study of semantics, the definition of languages is syntactic in nature. Yet, we consider this section to be a study of semantics because the purpose is to give some sort of meaning to the syntactic expressions we have in first-order languages.

We begin by studying first-order structures in an abstract sense. We then take a syntactic detour into the study of first-order languages, before examining what it means for structures to exist inside (and give interpretations for) first-order languages.

2.1.1 First-Order Structures

We begin by discussing relations and functions of a given arity.

Definition 2.1.1 (n-ary Relation on a Set)**.** Suppose A is a set and $n \geq 1$ is a natural number. An **n-ary relation on A** is a subset

$$\overline{R} \subseteq \{(a_1, \dots, a_n) \mid a_1, \dots, a_n \in A\}$$

We have a similar notion for functions, with the key fact being that n-ary functions take in n inputs and return a single output, and all inputs and outputs must come from the set in question.

Definition 2.1.2 (n-ary Function on a Set)**.** Given a set A, an **n-ary function on A** is a function

$$\overline{f} : A^n \to A$$

We make a subtle distinction between functions and relations in formal and informal language. This is something that will get clearer as we progress.

Convention. The reason why we put bars on top of the symbols is to distinguish functions and relations as they appear in formulae from the way that discuss them.

We have special terms when $n = 1, 2, 3$.

Convention.

1. A 1-ary relation is commonly called a **unary relation**.
2. A 2-ary relation is commonly called a **binary relation**.
3. A 3-ary relation is commonly called a **ternary relation**.

These notions are not new to us.

Example 2.1.3 (Some Familiar n-ary Relations).

1. Equality is a binary relation on any set.
2. $\leq$ is a binary relation on $\mathbb{R}$.
3. $\{x \in \mathbb{Z} \mid x \text{ is even}\}$ is a unary relation on $\mathbb{Z}$.

Admittedly, the fact that the third example is precisely a set is a little unusual to see. This is because in practice, the following convention is used.

Convention. Let $\overline{R} \subseteq A^n$ be a relation on some set A. For all $(a_1, \ldots, a_n) \in A^n$, when we write

$$\overline{R}(a_1, \ldots, a_n)$$

or say that

$$\overline{R}(a_1, \ldots, a_n) \text{ holds}$$

we mean that $(a_1, \ldots, a_n) \in \overline{R}$.

We are now ready for the most important definition of this chapter.

Definition 2.1.4 (First-Order Structure). A **first-order structure** is the following data:

1. a non-empty set A called the **domain** of $\mathcal{A}$.
2. a set of **relations** on A

$$\left\{\overline{R_i} \subseteq A^{n_i} \;\middle|\; i \in I\right\}$$

3. a set of **functions** on A

$$\left\{\overline{f_j} : A^{m_j} \to A \,\middle|\, j \in J\right\}$$

4. a set of **constants** that are elements of A

$$\{\overline{c_k} \in A \mid k \in K\}$$

where I, J, K are index sets that can be empty.

Usually, the index sets of a first-order structure are subsets of $\mathbb{N}$, but in principle, they could be any set. We package the information about the constants and the arity of the functions and relations together in the following manner.

Definition 2.1.5 (Signature). Let $\mathcal{A}$ be a first-order structure. The **signature** of $\mathcal{A}$ is the information

$$\{n_i \mid i \in I\} \qquad \{m_j \mid j \in J\} \qquad K$$

with the respective sets describing the arity of the relations on A, the arity of the functions on A, and the index set of the constants in A.

We use the following notation for first-order structures.

Convention. For a first-order structure $\mathcal{A}$ given as above, we wil deonte

$$\mathcal{A} = \left\langle A; \left\{\overline{R_i}\right\}_{i\in I}, \left\{\overline{f_j}\right\}_{j\in J}, \left\{\overline{c_k}\right\}_{k\in K}\right\rangle$$

More generally, we use the notation

$$\text{Structure} = \langle \text{Domain}; \text{Relations}, \text{Functions}, \text{Constants}\rangle$$

Often, we even drop the { } when describing first-order structures: for instance, we would simply write

$$\langle \mathbb{Z}; =, +, -, 0\rangle$$

as opposed to

$$\langle \mathbb{Z}; \{=\}, \{+, -\}, \{0\} \rangle$$

to describe the group of integers along with the relation of equality, the binary function of addition, the unary function of inversion by sign chance, and the identity 0.

We have encountered any number of first-order structures so far. Here are a few examples.

The first is a very basic example.

Example 2.1.6 (Orderings). We can take A to be one of the sets $\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}$. We can define a first-order structure on A with only one unary relation—that of ordering—and no functions or constants.

It is important to note that while the sets $\mathbb{N}, \mathbb{Z}, \mathbb{Q}, \mathbb{R}$ all admit richer structures on them, they are not needed to define ordering. We don't even include equality in this description because a formula that contains an equality symbol is not about ordering.

In the next example, we look at an algebraic structure.

Example 2.1.7 (Groups). Every group is a first-order structure with the following data.

1. The domain is the set of elements of the group.
2. The sole relation is the binary relation of equality.
3. There is a binary function for the group operation and a unary function for inversion.
4. There is a constant for the identity element.

We can make a similar definition for rings.

We do not even need to talk about objects that we usually deal with as sets. We can also talk about graphs, which, while defined in terms of sets, are usually studied visually.

Example 2.1.8 (Graphs). Graphs (or, more precisely, their vertices), along with two binary relations—equality and adjacency—and no functions or constants, form a first-order structure.

2.1.2 First-Order Languages

We are now ready to formally define the notion of first-order languages.

Definition 2.1.9 (First-Order Language). A **first-order language** $\mathcal{L}$ consists of the following data.

1. Index sets I, J, K where I is non-empty but J and K can be empty.
2. An **alphabet** of symbols, consisting of
   (a) **Variables** $x_0\ x_1\ x_2\ \cdots$
   (b) **Connectives** $\neg\ \rightarrow$
   (c) **Punctuation** ( ) ,
   (d) The **Quantifier** $\forall$
   (e) **Relation symbols** R_i for $i \in I$
   (f) **Function symbols** f_j for $j \in J$
   (g) **Constant symbols** c_k for $k \in K$
3. An **arity** for every relation and function symbol.

The arity and cardinality information of a first-order language is encoded in the following manner.

Definition 2.1.10 (Signature). Let $\mathcal{L}$ be a first-order language with index sets I, J, K such that I is non-empty and

1. The relations $\{R_i \mid i \in I\}$ have arities $\{n_i \mid i \in I\}$
2. The functions $\{f_j \mid j \in J\}$ have arities $\{m_j \mid j \in J\}$
3. The constants are given by $\{c_k \mid k \in K\}$

The information

$$\{n_i \mid i \in I\} \qquad \{m_j \mid j \in J\} \qquad K$$

is called the **signature** of $\mathcal{L}$.

In principle, one should very precisely define punctuation rules and the general notation for expressing formulae in a first-order language. Indeed, what this means is that one should define what it means for a string of symbols to be 'well-formed'. This is somewhat laborious, so we simply adopt the following convention.

Convention. We use the punctuation symbols

$$(\quad) \quad ,$$

in the following manner.

- We enclose all expressions involving connectives in parentheses, barring those involving a single variable or a single constant.
- We enclose statements of the form "$\forall x$" in parentheses.
- We denote applications functions f by $f(\cdots)$ and relations R by $R(\cdots)$.
- We use commas to separate the arguments of functions and relations.

There are numerous symbols in first-order languages. It makes sense to isolate the ones that form the 'objects' with which we 'reason' in this language.

Definition 2.1.11 (Terms). Let $\mathcal{L}$ be a first-order language. The set of **terms** of $\mathcal{L}$ is the smallest set such that

1. Every variable is a term.
2. Every constant is a term.
3. If $t_1, \ldots, t_n$ are terms and f is an n-ary function symbol, then $f(t_1, \ldots, t_n)$ is a term.

Moreover, we shall stipulate that every term arises in this manner.

We can define very basic first order languages.

Example 2.1.12. Let $\mathcal{L}$ be a first-order language such that

1. There are no relations
2. There is a binary function f
3. There are two constants c_1, c_2

Some terms of $\mathcal{L}$ are

$$c_1 \quad c_2 \quad x_1 \quad f(c_1, c_2) \quad f(x_1, c_1) \quad f(x_1, f(c_1, c_2)) \quad f(f(c_1, x_1), c_2)$$

There are many other terms. Note that we automatically assumed the existence of variables $x_1, x_2, \ldots$, which is consistent with Definition 2.1.9.

Non-Example 2.1.13. Let $\mathcal{L}$ be the first-order language given in Example 2.1.12. A string of symbols from the alphabet of $\mathcal{L}$ that is **not** a term is

$$ffx_1$$

The reason for this is that x_1 is not applied to f, and even if we were to ignore the punctuation convention of writing function arguments inside parentheses, we would have that the arity of f is violated, because a function of two variables is being applied (twice) to a single input (or the leftmost f is being applied to both f and x_1, which contradicts the fact that f takes inputs that are both terms, and f alone is not a term). It is precisely to avoid ambiguities of this sort that we have punctuation conventions; either way, in this case, there are too many errors for ffx_1 to be a term in $\mathcal{L}$.

We now define a way of using the quantifiers and connectives of first-order language s to build *formulae*. The idea is to define a fundamental notion of formulae using purely the relations of the language and then define how more complex formulae can be built from them.

Definition 2.1.14 (Atomic Formula). Let $\mathcal{L}$ be a first-order language. An **atomic formula** of $\mathcal{L}$, or an $\mathcal{L}$-**atomic formula**, is an expression of the form

$$R(t_1, \dots, t_n)$$

where R is an n-ary relation symbol in $\mathcal{L}$ and $t_1, \dots, t_n$ are terms.

Note that in the above definition, we do not require the inputs $t_1, \dots, t_n$ to be variables or constants. We merely require them to be *terms*. This means they could be variables, constants, or outputs of functions applied to other terms. Being 'atomic' has only to do with being the output of a relation symbol. We can now define what a formula is in a broader sense.

Definition 2.1.15 (Formula). Let $\mathcal{L}$ be a first-order language. The **formulae** of $\mathcal{L}$, or the $\mathcal{L}$-**formulae**, are defined as follows.

1. Every atomic formula is a formula.
2. If ϕ is a formula, then so is $(\neg\phi)$.

3. If ϕ and ψ are formulae, then so is $(\phi \to \psi)$.
4. If ϕ is a formula, then so is $(\forall x)\,\phi$.

Moreover, we stipulate that every formula arises in this manner.

We can define very simple formulae in very simple first-order language s.

Example 2.1.16. Let $\mathcal{L}$ be a first-order language with

1. One unary relation symbol P and one binary relation symbol R
2. One binary function symbol f
3. Two constants c_1 and c_2

Then, the following are all atomic formulae:

$$P(x_1) \quad R(c_1, x_1) \quad R(f(x_1, c_1), c_2)$$

Similarly, the following are all formulae:

$$\neg P(x_1) \quad (P(x_1) \to R(c_1, x_1)) \quad (\forall x)\, R(x, c_1)$$

There is a reason why we only allowed first-order language s to have connectives $\to$ and $\neg$ and quantifier $\forall$: we can build the other connectives $(\wedge, \vee, \leftrightarrow, \ldots)$ and the other quantifier $(\exists)$ from these.

Definition 2.1.17 (The Existential Quantifier)**.** Let $\mathcal{L}$ be a first-order language and let ϕ be an $\mathcal{L}$-formula. Then, we define

$$(\exists x)\,\phi$$

to be shorthand for the formula

$$(\neg\,(\forall x)\,(\neg\phi))$$

We also define the other connectives as in propositional logic.

Definition 2.1.18 (Connectives). Let $\mathcal{L}$ be a first-order language. Let ϕ and ψ be $\mathcal{L}$-formulae. We define the connectives $\wedge, \vee, \leftrightarrow, \uparrow, \downarrow$ as follows:

$$(\phi \wedge \psi) \text{ is shorthand for } (\neg(\phi \to \neg\psi))$$
$$(\phi \vee \psi) \text{ is shorthand for } ((\neg\phi) \to \psi)$$
$$(\phi \leftrightarrow \psi) \text{ is shorthand for } ((\phi \to \psi) \wedge (\psi \to \phi))$$
$$(\phi \uparrow \psi) \text{ is shorthand for } (\neg(\phi \wedge \psi))$$
$$(\phi \downarrow \psi) \text{ is shorthand for } (\neg(\phi \vee \psi))$$

We are now ready to explore the utility of first-order logic in the study of mathematics, where we relate the study of first-order structures to first-order languages.

2.1.3 First-Order Structures Revisited

Throughout this subsection, we will fix a first-order language $\mathcal{L}$ with signature

$$\{n_i \mid i \in I\} \qquad \{m_j \mid j \in J\} \qquad K$$

where the n_is are the arities of the relation symbols R_i, the m_j are the arities of the function symbols f_j, and K is the index set of the constants c_k.

Definition 2.1.19 (First-Order Structures in First-Order Languages). A **structure** in $\mathcal{L}$, or an $\mathcal{L}$-**structure**, is a first-order structure

$$\mathcal{A} = \left\langle A; \left\{\overline{R_i}\right\}_{i\in I}, \left\{\overline{f_j}\right\}_{j\in J}, \left\{\overline{c_k}\right\}_{k\in K}\right\rangle \tag{2.1.1}$$

such that the signature of $\mathcal{A}$ is the same as that of $\mathcal{L}$, ie, the arities of the relations and functions in $\mathcal{A}$ are the same as those in $\mathcal{L}$.

If one takes a closer look at the definitions of first-order structures and first-order languages (resp. Definition 2.1.4 and Definition 2.1.9), one notices that the former involves **actual** relations and functions, which are defined as subsets of certain sets, whereas the latter merely involves function and relation **symbols**. This is a crucial distinction.

Convention. The reason why we put bars on top of the symbols is to distinguish functions and relations as they appear in first-order structures—*with* bars on top—from the way we express them in the ambient first-order language.

For the remainder of this subsection, fix a first-order structure $\mathcal{A}$. Denote its domain, relations, functions and constants as in (2.1.1), with bars above as per our convention for this module. We now discuss the significance of first-order structures and underscore the power and versatility of first-order languages.

When we say, in Definition 2.1.19, that the signature of an $\mathcal{L}$-structure is the same as that of the first-order language $\mathcal{L}$, we mean that the arities of the relations in the structure must match up with the arities of the relation symbols in the language, and similarly for functions and constants. More precisely, we have the following.

Definition 2.1.20 (Interpretation). Let $\mathcal{A}$ be an $\mathcal{L}$-structure. The correspondence

$$\overline{R_i} \leftrightsquigarrow R_i \qquad \overline{f_j} \leftrightsquigarrow f_j \qquad \overline{c_k} \leftrightsquigarrow c_k$$

between relations, functions and constants in $\mathcal{A}$ and relation symbols, function symbols and constant symbols in $\mathcal{L}$ is called an **interpretation of** $\mathcal{L}$.

We have a special term for the "$\mathcal{L}$ to $\mathcal{A}$" direction of this correspondence.

Definition 2.1.21 (Valuation). Let $\mathcal{A}$ be an $\mathcal{L}$-structure. A **valuation** in $\mathcal{A}$ is a function

$$\mathbf{v} : \{\text{Terms of } \mathcal{L}\} \to A$$

that assigns terms of $\mathcal{L}$ to their interpretations in $\mathcal{A}$ in the following manner.

1. For all constant symbols c_k in $\mathcal{L}$, we have

$$\mathbf{v}(c_k) = \overline{c_k}$$

2. For all terms $t_1, \ldots, t_m$ and m-ary function symbols f in $\mathcal{L}$, we have

$$\mathbf{v}(f(t_1, \ldots, t_m)) = \overline{f}(\mathbf{v}(t_1), \ldots, \mathbf{v}(t_m))$$

where $\overline{f}$ is the interpretation of f in $\mathcal{A}$.

The idea is that a first-order language gives a purely symbolic way of expressing relationships between objects and their properties. When studying statements expressed in first-order languages, one must purely view them symbolically, as formal expressions that do not carry any *meaning* per se. The 'meaning' comes from valuations that allow us to interpret symbols in first-order languages as ideas in first-order structures.

We have an existence and uniqueness result.

Lemma 2.1.22. *Let $\mathcal{A}$ be an $\mathcal{L}$-structure with domain A. Fix elements $a_0, a_1, a_2, \dots \in A$. There exists a unique valuation $\mathbf{v}$ of $\mathcal{L}$ in $\mathcal{A}$ such that $\mathbf{v}(x_i) = a_i$ for all $i \in \mathbb{N}$.*

Proof. We begin by showing existence. We can define $\mathbf{v}$ explicitly for all terms of $\mathcal{L}$ by performing recursion on their length. For terms of length 1, we deal with the variables and the distinguished constants separately.

1. $\mathbf{v}(x_i) := a_i$ for all $i \in \mathbb{N}$

2. $\mathbf{v}(c_k) := \overline{c_k}$ for all $k \in K$

We can then define $\mathbf{v}$ for terms arising from functions and relations of arity ≥ 2 by setting

$$\mathbf{v}(f(t_1, \dots, t_m)) := \overline{f}(\mathbf{v}(t_1), \dots, \mathbf{v}(t_m))$$

for all terms $t_1, \dots, t_m$ and m-ary function symbols f in $\mathcal{L}$ with interpretation $\overline{f}$ in $\mathcal{A}$. Such a valuation is unique because if there are two valuations satisfying the condition on the variables, then they agree on <u>all</u> terms of length 1 (because they must agree on all constants—see Definition 2.1.21). The fact that they obey the recursion relation by definition then gives us the result. □

Example 2.1.23 (Interpreting a Term in Groups). Let $\mathcal{L}$ have the following data.

1. Function symbols: a binary symbol m and a unary symbol i.
2. Relation symbols: a binary symbol R.
3. Constant symbols: a constant symbol e.

The following is an $\mathcal{L}$-formula:

$$m(m(x_0, x_1), i(x_0))$$

Consider the first-order structure $\mathcal{A}$, meant to represent a group, with the underlying data.

1. Domain: G, a set.
2. Functions: the binary function $\overline{m}$ of multplication (the group operation) and the unary function $\overline{i}$ of inversion.
3. Relations: the binary relation $\overline{R}$ of equality.
4. Constants: the identity element $\overline{e}$.

Fix arbitrary group elements $g, h \in G$. We know, from Lemma 2.1.22, that there exists a unique valuation $\mathbf{v}$ of $\mathcal{L}$ in $\mathcal{A}$ such that $\mathbf{v}(x_0) = g$ and $\mathbf{v}(x_1) = h$. This valuation will allow us to interpret the formula above as a statement about the group elements g and h:

$$\begin{aligned}
\mathbf{v}(m(m(x_0, x_1), i(x_0))) &= \overline{m}(\mathbf{v}(m(x_0, x_1)), \mathbf{v}(i(x_0))) \\
&= \overline{m}\big(\overline{m}(\mathbf{v}(x_0), \mathbf{v}(x_1)), \overline{i}(\mathbf{v}(x_0))\big) \\
&= \overline{m}\big(\overline{m}(g, h), \overline{i}(g)\big) \\
&= \overline{m}\big(g \cdot h, g^{-1}\big) \\
&= ghg^{-1}
\end{aligned}$$

If we had defined the structure $\mathcal{A}$ a little differently—for example, if we had defined $\overline{m}(g, h) := hg$ instead—then the interpretation of the formula in $\mathcal{A}$ would have been different, despite the formula itself being exactly the same in $\mathcal{L}$. This illustrates the role of valuations and interpretations in moving between purely formal, syntactic expressions in first-order languages to meaningful expressions or statements in first-order structures.

Remark. It is worth remarking that in the above example, we use the $=$ symbol somewhat frivolously. On the one hand, left- and right-hand sides of each equation lie in G, and $=$ can be understood as equality in $\mathcal{A}$. However, one can also view these as being a *syntactic* equalities in $\mathcal{A}$, since the properties we use to manipulate the above expressions are independent of the group structure of $\mathcal{A}$. The only simplifications we make come from the *definitions* of $\mathbf{v}$, $\overline{m}$, and $\overline{i}$ (for that matter, we can even view $\cdot$ and $^{-1}$ as notation for $\overline{m}$ and $\overline{i}$ instead of viewing $\overline{m}$ and $\overline{i}$ as notation for $\cdot$ and $^{-1}$). We sidestep these syntactic nuances in this module, but we do mention that a more syntax-heavy treatment is necessary for a computer-scientific study of first-order logic.

With this, we have studied first-order languages and structures in sufficient detail to be able to talk about how we can express and implement ideas from propositional logic in first-order logic.

2.2 A Bridge between Propositional and First-Order Logic

The purpose of this section is to discuss how ideas in propositional logic can be expressed in first-order logic. As we might expect from Definition 2.1.9, first-order logic is a generalisation of propositional logic. We will use this section to make this precise.

We will begin by making rigorous the notion of **logical validity**. We will see a connection with tautologies, which, as we know, are precisely the theorems of propositional logic. We will explore notions like satisfaction and substitution to make this precise.

2.2.1 Valuations Satisfying Formulae

For the remainder of this subection, fix some first-order language $\mathcal{L}$ and some $\mathcal{L}$-structure $\mathcal{A}$. We begin by defining a notion of equivalence for valuations that is weaker than equality.

> **Definition 2.2.1** (Equivalence with respect to a Variable)**.** Suppose x_l is a variable in $\mathcal{L}$ and valuations $\mathbf{v}, \mathbf{w}$ of $\mathcal{L}$ in $\mathcal{A}$. We say that $\mathbf{v}$ and $\mathbf{w}$ are x_l-**equivalent** if for all $i \in \mathbb{N} \setminus \{l\}$, we have $\mathbf{v}(x_i) = \mathbf{w}(x_i)$.

One way to see that this notion is weaker than equality of valuations is that if valuations agree on all variables, then by Lemma 2.1.22, they must be equal. The fact that there is one variable on which they *may* differ is what makes this notion weaker. We also emphasise that we only stipulate that x_l-equivalent valuations *need not* agree on x_l, not that they *must not* agree on x_l. In particular, *equal valuations are equivalent with respect to all variables*.

We are now ready to define precisely what it means for a valuation to satisfy a formula. The idea is that in a general setting—ie, in a first-order language—formulae do not express any properties, and without an interpretation in a first-order structure, it is not very meaningful to talk about what it means for a formula to be 'true'. A valuation translates formulae into 'meaningful statements', and therefore, informally, we can think of a valuation satisfying a formula as meaning that it translates a general formula into some interpretation that we know to be 'true' in some sense.

Definition 2.2.2 (Satisfaction). Let ϕ be an $\mathcal{L}$-formula and $\mathbf{v}$ a valuation of $\mathcal{L}$ in $\mathcal{A}$. We can define what it means for **v satisfies ϕ in $\mathcal{A}$** recursively on the number of connectives or quantifiers in ϕ.

1. If ϕ is an atomic formula, ie, has no connectives or quantifiers, then we know ϕ is of the form $R(t_1, \ldots, t_n)$ for some n-ary relation R and terms $t_1, \ldots, t_n$ in $\mathcal{L}$. In this case, we say that **v satisfies ϕ in $\mathcal{A}$** if the relation

$$\overline{R}(\mathbf{v}(t_1), \ldots, \mathbf{v}(t_n))$$

holds in $\mathcal{A}$, ie, if $(\mathbf{v}(t_1), \ldots, \mathbf{v}(t_n)) \in \overline{R} \subseteq A^n$.

2. If ϕ is not an atomic formula in $\mathcal{L}$, then we know ϕ is of one of the following forms.
   (a) If ϕ is of the form $(\neg\psi)$ for some $\mathcal{L}$-formula ψ, then we say that **v satisfies ϕ in $\mathcal{A}$** if $\mathbf{v}$ does not satisfy ψ in $\mathcal{A}$.
   (b) If ϕ is of the form $(\psi \to \chi)$ for $\mathcal{L}$-formulae ψ and χ, then we say that **v satisfies ϕ in $\mathcal{A}$** if it is not the case that $\mathbf{v}$ satisfies ψ in $\mathcal{A}$ and $\mathbf{v}$ does not satisfy χ in $\mathcal{A}$.
   (c) If ϕ is of the form $(\forall x)\,\psi$ for some $\mathcal{L}$-formula ψ, we say **v satisfies ϕ in $\mathcal{A}$** if for all variables x, if $\mathbf{w}$ is a valuation of $\mathcal{L}$ in $\mathcal{A}$ that is x-equivalent to $\mathbf{v}$, then $\mathbf{w}$ satisfies ψ in $\mathcal{A}$.

   In all three cases, we have define what it means for $\mathbf{v}$ to satisfy ϕ in terms of what it means for $\mathbf{v}$ to satisfy formulae with one fewer connectives or quantifiers than ϕ, making the recursion well-defined.

If $\mathbf{v}$ satisfies ϕ in $\mathcal{A}$, we write $\mathbf{v}[\phi] = \mathbf{T}$, and if $\mathbf{v}$ does not satisfy ϕ in $\mathcal{A}$, we write $\mathbf{v}[\phi] = \mathbf{F}$.

We remark that the notation $\mathbf{v}[\phi] = \mathbf{T}$ is merely shorthand for a statement of fact. It does not actually represent an equality in some setting where both $\mathbf{T}$ and $\mathbf{v}[\phi]$ are defined. The same goes for $\mathbf{v}[\phi] = \mathbf{F}$. However, it is often convenient to abuse notation.

Convention. Given valuations $\mathbf{v}, \mathbf{w}$ of $\mathcal{L}$ in $\mathcal{A}$, we write

$$\mathbf{v}[\phi] = \mathbf{w}[\phi]$$

to denote the condition that

$$\mathbf{v}[\phi] = \mathbf{T} \text{ if and only if } \mathbf{w}[\phi] = \mathbf{T}$$

or the equivalent condition that

$$\mathbf{v}[\phi] = \mathbf{F} \text{ if and only if } \mathbf{w}[\phi] = \mathbf{F}$$

This notion of a valuation satisfying a formula tells us what it means for a formula to hold true in a structure for a *given* assignment of meaning to the first-order language symbols in the formula. We can define a more absolute notion of truth.

Definition 2.2.3 (Truth of a First-Order Formula in a First-Order Structure). Let ϕ be an $\mathcal{L}$-formula. We say **ϕ is true in $\mathcal{A}$**, or that **$\mathcal{A}$ is a model of ϕ**, if $\mathbf{v}[\phi] = \mathbf{T}$ for all valuations $\mathbf{v}$ of $\mathcal{L}$ in $\mathcal{A}$. We denote this by $\mathcal{A} \models \phi$.

A first-order formula true in one structure might not be true in another.

Example 2.2.4. Let $\mathcal{L}$ have one binary relation. Consider the atomic formula

$$(\forall x_1)\,(\exists x_2)\,R(x_1, x_2) \tag{2.2.1}$$

This formula is true in structures like $\langle \mathbb{N}; < \rangle$, $\langle \mathbb{Z}, < \rangle$, and $\langle \mathbb{Z}, > \rangle$.

Non-Example 2.2.5. Let $\mathcal{L}$ be as above. The formula (2.2.1) above is NOT true in $\langle \mathbb{N}; > \rangle$, because 0 does not have a predecessor in $\mathbb{N}$. The reason for this is that relation inputs are *ordered* (ie, relations are not, in general, symmetric).

Despite being valuation-indepentent, given that the truth of formulae is structure-dependent, it is still not the strongest notion of truth we can define in first-order logic. We want theorems to be stronger and more 'absolute' notions of truth. To that end, we define logical validity, a *structure-independent notion of truth*.

Definition 2.2.6 (Logical Validity). Let ϕ be an $\mathcal{L}$-formula. We say that ϕ is logically valid if $\mathcal{A} \models \phi$ for all $\mathcal{L}$-structures $\mathcal{A}$.

Logically valid formulae in first-order logic are meant to be analogues of tautologies in propositional logic. An important difference, though, is that in propositional logic, there is an algorithm that *decides* whether a given formula is a tautology. In first-order logic, this is not usually the case, as we shall see when we study Gödel's Incompleteness Theorem.

Example 2.2.7. For all $\mathcal{L}$-formulae ϕ, the formula

$$((\exists x_1)\,(\forall x_2)\,\phi \to (\forall x_2)\,(\exists x_1)\,\phi)$$

is logically valid.

Proof. sorry □

Non-Example 2.2.8. Given an $\mathcal{L}$-formulae ϕ, the formula

$$((\forall x_1)\,(\exists x_2)\,\phi \to (\exists x_2)\,(\forall x_1)\,\phi)$$

is not necessarily logically valid.

Proof. sorry □

2.2.2 Substitution

In this section, we will discuss how replacing propositional variables in propositional formulae with first-order formulae gives a bridge between the two kinds of logic. For the remainder of this subection, fix

- a natural number n
- a first-order language $\mathcal{L}$
- $\mathcal{L}$-formulae $\phi_1, \ldots, \phi_n$
- propositional variables $p_1, \ldots, p_n$

- a propositional formula χ with propositional variables $p_1, \ldots, p_n$ (cf. Definition 1.1.3)

We have a name for replacing the p_i χ with ϕ_i.

Definition 2.2.9 (Substitution Instance). The **substitution instance of χ with** $\phi_1, \ldots, \phi_n$ is the $\mathcal{L}$-formula obtained by replacing each propositional variable p_i in χ with the $\mathcal{L}$-formula ϕ_i.

Remark. We underscore here that a substitution instance is a formula in $\mathcal{L}$, not a propositional formula, because the terms of which it is made are all $\mathcal{L}$-formulae rather than propositional variables.

For the remainder of this subsection, denote by θ the substitution instance of χ with $\phi_1, \ldots, \phi_n$.

Substitution is our gateway for 'embedding' propositional logic into first-order logic. This 'embedding' preserves 'truth' in the following manner.

Theorem 2.2.10. *If χ is a tautology, then θ is logically valid.*

Proof. Assume that χ is a tautology (cf. Definition 1.1.9). To show that θ is logically valid, we need to show that $\mathcal{A} \models \theta$ for all $\mathcal{L}$-structures $\mathcal{A}$. To that end, let $\mathcal{A}$ be an $\mathcal{L}$-structure and let $\mathbf{v}$ be a valuation of $\mathcal{L}$ in $\mathcal{A}$. We need to show that $\mathbf{v}[\theta] = \mathbf{T}$. `sorry` □

It is tempting to ask whether all logically valid formulae are substitution instances of tautologies. Such a fact would truly make the above result a two-way bridge. Unfortunately, this is not true, as shown by the following counterexample.

Counterexample 2.2.11. For all $\mathcal{L}$-formulae ϕ, the $\mathcal{L}$-formula

$$((\exists x_2)(\forall x_1)\,\phi \to (\forall x_1)(\exists x_2)\,\phi)$$

is logically valid. However, it is not a substitution instance of any propositional formula, because there is no notion of quantification in propositional logic.

This tells us that first-order logic is not only stronger than propositional logic, it is **strictly** stronger.

2.3 Variables and the Universal Quantifier

The purpose of this section is to understand how to work with variables and the universal quantifier in first-order languages. Throughout this section, fix a first-order language $\mathcal{L}$.

2.3.1 Bound and Free Variables

Consider the formula

$$\psi_1 : (R_1(x_1, x_2) \rightarrow (\forall x_3)\, R_2(x_1, x_3)) \tag{2.3.1}$$

where R_1, R_2 are relation symbols in $\mathcal{L}$ and x_1, x_2, x_3 are variables in $\mathcal{L}$. Intuitively, we do not want x_3 to 'exist', or be 'known' or 'accessible', outside of the sub-formula[1] $(\forall x_3)\, R_2(x_1, x_3)$. We can mae this notion precise.

Definition 2.3.1 (Scope of a Quantifier). Suppose ϕ and ψ are $\mathcal{L}$-formulae. If $(\forall x_i)\, \phi$ occurs as a sub-formula of ψ, we say that ϕ is the **scope of** $(\forall x_i)$ **in** ψ.

Example 2.3.2. In (2.3.1), $R_2(x_1, x_3)$ is the scope of the quantifier $(\forall x_3)$.

We use the existence of a scope to characterise variables.

Definition 2.3.3 (Bound Variables). Let ψ be an $\mathcal{L}$-formula and let x_j be a variable that appears in ψ. We say that x_j is **bound in** ψ if it is within the scope of a quantifier.

Example 2.3.4. In (2.3.1), the variable x_3 is bound, because it lies in the scope $R_2(x_1, x_3)$ of the quantifier $(\forall x_3)$.

We also have a special name for variables that are not bound.

[1]When we say sub-formula, we mean exactly what it sounds like: since formulae are built from smaller formulae using connectives and the quantifier, when we say sub-formula, we mean a formula that arose at an intermediate step in this inductive construction of the formula of which it is a sub-formula.

Definition 2.3.5 (Free Variables)**.** Let ψ be an $\mathcal{L}$-formula and let x_j be a variable that appears in ψ. We say that x_j is **free in** ψ if it is not bound in ψ.

Example 2.3.6. In (2.3.1), the variable x_1 is bound, because it does not lie in the scope of any quantifiers, meaning it is not bound.

The same variable can have both free and bound occurrences within a given formula.

Example 2.3.7. In the formula

$$((\forall x_1)\, R_1(x_1, x_2) \rightarrow R_2(x_1, x_2))$$

the variable x_1 is bound in the expression

$$(\forall x_1)\, \underbrace{R_1(x_1, x_2)}_{\text{scope of } (\forall x_1)}$$

but free in the expression

$$R_2(x_1, x_2)$$

We can also show that existentially quantified variables are bound.

Lemma 2.3.8. *Let ψ and ϕ be an $\mathcal{L}$-formulae. If ψ contains*

$$(\exists x_1)\, \phi$$

as a sub-formula, then x_1 is bound in ψ.

Proof. This follows immediately from the definition of the existential quantifier. `sorry` □

The occurrence of free variables inside a formula means that it is very general, as the variables in it are purely formal symbols that we have no machinery to reason with. We give a special term to, and have a special interest in, formulae with no free variables.

Definition 2.3.9 (Closed Formulae). An $\mathcal{L}$-formula with no free variables is called a **closed formula** or a **sentence**.

Formulae that are not closed can be thought of as being 'dependent' on their free variables, in the sense that we would intuitively want to only define a notion of substitution only for free variables. To that end, we adopt some notation.

Convention. Let ψ be an $\mathcal{L}$-formula. If $\mathcal{L}$ has free variables $x_1, \ldots, x_n$, then we denote ψ by

$$\psi(x_1, \ldots, x_n)$$

when we wish to underscore the dependence of ψ on $x_1, \ldots, x_n$ or the fact that $x_1, \ldots x_n$ are free in ψ or perform a substitution.

We now define substitution.

Definition 2.3.10 (Substitution). Let $t_1, \ldots, t_n$ be terms in $\mathcal{L}$ and $\psi(x_1, \ldots, x_n)$ an $\mathcal{L}$-formula with free variables $x_1, \ldots, x_n$[a]. We define the formula obtained by **substituting the t_i for the x_i** to be the $\mathcal{L}$-formula obtained by replacing each occurrence of x_i in ψ with an occurrence of the corresponding t_i. We denote this formula

$$\psi(t_1, \ldots, t_n)$$

[a]By the aforementioned convention, we would ordinarily not mention that $x_1, \ldots, x_n$ are free in ψ.

2.3.2 An Analogue of Completeness

Throughout this subsection, fix an $\mathcal{L}$-formula ϕ and an $\mathcal{L}$-structure $\mathcal{A}$.

We begin by mentioning a result about valuations.

Theorem 2.3.11. *Fix $n \in \mathbb{N} \cup \{0\}$. If ϕ has free variables $x_1, \ldots, x_n$ and $\mathbf{v}, \mathbf{w}$ are valuations* *of $\mathcal{L}$ in $\mathcal{A}$ with $\mathbf{v}(x_i) = \mathbf{w}(x_i)$ for all $1 \leq i \leq n$, then*

$$\mathbf{v}[\phi] = \mathbf{T} \text{ if and only if } \mathbf{w}[\phi] = \mathbf{T}$$

In other words, we have $\mathbf{v}[\phi] = \mathbf{w}[\phi]$.

Proof. Fix valuations $\mathbf{v}$ and $\mathbf{w}$ of $\mathcal{L}$ in $\mathcal{A}$ that agree on all the free variables of ϕ. We argue that $\mathbf{v}[\phi] = \mathbf{w}[\phi]$ by performing induction on the total number connectives and quantifiers in ϕ.

The base case is when ϕ has no connectives or quantifiers, ie, when ϕ is atomic in terms that contain no quantifiers. In this case, we have that ϕ is of the form

$$R(t_1, \ldots, t_m)$$

where $t_1, \ldots, t_m$ are terms that contain no quantifiers. In this case, each t_i is either a constant or a variable. Since ϕ is assumed not to contain any quantifiers, the t_i that are variables must be free variables, and the rest must be constants. Since $\mathbf{v}$ and $\mathbf{w}$ agree on all constants by definition and on all free variables by assumption, we must have $\mathbf{v}(t_i) = \mathbf{w}(t_i)$ for all i. Then, `sorry` □

This gives us a completeness-like result about closed formulae.

Corollary 2.3.12. *If ϕ is closed, then either $\mathcal{A} \models \phi$ or $\mathcal{A} \models (\neg\phi)$.*

Proof. If ϕ is closed, ϕ has no free variables. Then, any two valuations of $\mathcal{L}$ in $\mathcal{A}$ agree vacuously on the set of all free variables of ϕ. Therefore, there cannot be two distinct valuations such that one satisfies ϕ and one does not. In other words, either ϕ is true in $\mathcal{A}$ or ϕ is not true in $\mathcal{A}$. Equivalently, either ϕ or $(\neg\phi)$ is true in $\mathcal{A}$. □

2.3.3 Understanding the Universal Quantifier

The purpose of this subsection is to discuss the relationship between the syntax and the semantics of the universal quantifier $\forall$. We have not been too precise so far about what it *means* to write an expression like

$$(\forall x_1)\, \phi$$

for some formula ϕ with a bound variable x_1. We particularly delve into nuances that arise due to the fact that in our strict syntactic definition of $\mathcal{L}$-formulae (Definition 2.1.15, we only allow the quantifier to be succeeded by a **variable** in $\mathcal{L}$, which must be one of $x_1, x_2, \ldots$. The really confusing thing about this syntactic definition of the universal quantifier is the fact that it is not obvious that this quantifier means "for all": instead of saying "for all variables x_1, we have ϕ", on a strictly syntactic level, the expression $(\forall x_1)\,\phi$ is using the *specific variable* x_1.

What we would want—say, in an interactive theorem prover like Lean—is a mechanism to be able to *remove* the quantifier and instead *introduce* a free variable into our formula and label that variable differently from all other free and bound variables in our formula.

In this subsection, we will make this notion precise. Throughout, fix a first-order language $\mathcal{L}$ and a first-order structure $\mathcal{A}$ in $\mathcal{L}$. Denote the domain of $\mathcal{A}$ by A.

We begin by describing what it means for a structure to model a formula under a valuation.

Definition 2.3.13. Let $\psi(x_1, \ldots, x_n)$ be an $\mathcal{L}$-formula whose free variables are $x_1, \ldots, x_n$. Let $a_1, \ldots, a_n \in A$ be constants. We say that $\mathcal{A}$ **models** $\psi(a_1, \ldots, a_n)$, denoted

$$\mathcal{A} \models \psi(a_1, \ldots, a_n)$$

if $\mathbf{v}[\psi] = \mathbf{T}$ for every valuation $\mathbf{v}$ of $\mathcal{L}$ in $\mathcal{A}$.

Note that by Lemma 2.1.22, to prove that $\mathcal{A} \models \psi(a_1, \ldots, a_n)$, it suffices to prove that $\mathbf{v}[\psi] = \mathbf{T}$ for *some* valuation $\mathbf{v}$ of $\mathcal{L}$ in $\mathcal{A}$.

We would want to be able to use the quantifier $\forall$ in the way that we are used to using it. In other words, from a hypothesis

$$(\forall x_1)\,\phi(x_1)$$

we would want to be able to deduce

$$\phi(t)$$

for any term t in $\mathcal{L}$. Unfortunately, given that x_1 is *itself* a term in $\mathcal{L}$, and $(\forall x_1)\,\phi(x_1)$ is does not involve some 'general variable' but the very concrete variable x_1, this deduction is simply not true![2]

Counterexample 2.3.14 (Indiscriminate specialisation is not a good idea)**.** We provide a counterexample that disproves the claim that we can indiscriminately specialise a formula $(\forall x_1)\,\phi(x_1)$ to *any* $\mathcal{L}$-term t to obtain a true statement $\phi(t)$. More precisely, we give

- a first-order language $\mathcal{L}$
- an $\mathcal{L}$-structure $\mathcal{A}$
- an $\mathcal{L}$-formula $\phi(x_1)$ with a free variable x_1
- an $\mathcal{L}$-term t
- a valuation $\mathbf{v}$ of $\mathcal{L}$ in $\mathcal{A}$

such that

$$\mathcal{A} \models (\forall x_1)\,\phi(x_1)$$

but

$$\mathbf{v}[\phi(t)] = \mathbf{F}$$

Let $\mathcal{L}$ have one binary relation R and one unary relation S. Let

$$\phi(x_1) : ((\forall x_2)\,R(x_1, x_2) \to S(x_1))$$

We can specialise this formula ϕ via a **substitution instance** (cf. Definition 2.3.10). Let t be the term x_2. Since Definition 2.3.10 imposes **no conditions** on what can be substituted for a free variable, we can substitute t for x_1 in ϕ. Then, we have that $\phi(t)$ is the formula

$$(\forall x_2)\,R(x_2, x_2) \to S(x_2)$$

The problem with this substitution is that the x_1 that served as the first argument of R prior to substitution was free, whereas the x_2 was bound to the quantifier $(\forall x_2)$ that preceded

[2] It is at times like this that it is particularly important to distinguish semantics and syntax. When we say "simply not true", that is an inherently semantic statement: syntactic reasoning is not a reasoning of *truth*, but one of *provability*. What we are really attempting to do is establish *motivation* for the axioms of the formal deduction system for a first-order language. We will see that in our formal system, we will not allow this type of deduction. This section underscores that it is actually *sensible* to disallow this explicitly, because even in an 'intuitive' sense of reasoning, this would not 'make sense'.

$R(x_1, x_2)$. After substitution, both arguments of R are now bound to the quantifier $(\forall x_2)$. We can see this in the following model.

Let $\mathcal{A} = \langle \mathbb{N}; \leq, (=0) \rangle$. That is, $\mathcal{A}$ is the $\mathcal{L}$-structure with the following data.

- The domain of $\mathcal{A}$ is $\mathbb{N}$.
- The binary relation in $\mathcal{A}$ is the ordering $\leq$.
- The unary relation on $\mathcal{A}$ is the condition that the argument be equal to zero.

In this case, we can certainly see that

$$\mathcal{A} \models (\forall x_1)\, \phi(x_1)$$

because for any natural number x_1, if $x_1 \leq x_2$ for all natural numbers x_2, then $x_1 = 0$.

sorry

sorry

2.4 A Formal System for First-Order Logic

The purpose of this section is to define, for any first-order language $\mathcal{L}$, a **formal deduction system $\mathbf{K}_{\mathcal{L}}$** in which we can do **first-order logic**.

Throughout this section, fix a first-order language $\mathcal{L}$.

2.4.1 The Formal Deduction System $\mathbf{K}_{\mathcal{L}}$

Recall Definition 1.2.6, in which we define a **formal deduction system**. Informally, this is meant to be a system in which we can perform **deductions**. The purpose of this subsection is to define a **formal deduction system for $\mathcal{L}$**.

Definition 2.4.1 (A Formal Deduction System for First-Order Logic)**.** Define $\mathbf{K}_{\mathcal{L}}$ to be the **formal deduction system** consisting of the following alphabet, formulae, axioms and deduction rules.

1. **Alphabet.** The **alphabet** of $\mathbf{K}_{\mathcal{L}}$ is the alphabet of $\mathcal{L}$ (cf. Definition 2.1.9).

2. **Formulae.** The **formulae** of $\mathbf{K}_{\mathcal{L}}$ are the formulae of $\mathcal{L}$ (cf. Definition 2.1.15).
3. **Axioms.** For any $\mathcal{L}$-formulae ϕ, ψ, χ and $i \in \mathbb{N}$, the **axioms** of $\mathbf{K}_{\mathcal{L}}$ in ϕ, ψ, χ and x_i are the following distinguished $\mathcal{L}$-formulae:

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

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

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

   (K1) $((\forall x_i)\, \phi(x_i) \to \phi(t))$ where t is a term free for x_i in ϕ (cf. sorry) and ϕ can have other free variables.

   (K2) $((\forall x_i)\, (\phi \to \psi) \to (\phi \to (\forall x_i)\, \psi))$ if x_i is not free in ϕ.
4. **Deduction Rules.** For any $\mathcal{L}$-formulae ϕ, ψ and $i \in \mathbb{N}$, the **deduction rules** for $\mathbf{K}_{\mathcal{L}}$ in ϕ, ψ, x_i are

   (MP) Modus Ponens: From ϕ and $(\phi \to \psi)$, deduce ψ.

   (Gen) Generalisation: From ϕ, deduce $(\forall x_i)\, \phi$.

We define deductions identically to how we did for arbitrary formal deduction systems.

Definition 2.4.2 (Deductions in $\mathbf{K}_{\mathcal{L}}$). Let Σ be a set of $\mathbf{K}_{\mathcal{L}}$-formulae. A **deduction in $\mathbf{K}_{\mathcal{L}}$ from Σ** is a finite sequence of $\mathbf{K}_{\mathcal{L}}$-formulae such that each is one of the following:

- an axiom of $\mathbf{K}_{\mathcal{L}}$.
- a formula in Σ.
- deduced from a previous formula in the sequence via the deduction rule (MP).
- deduced from a previous formula in the sequence via the deduction rule (Gen) with the restriction that when (Gen) is applied to deduce $(\forall x_i)\, \phi$ from ϕ, x_i does not appear as a free variable in any formula of Σ that is used in the deduction of ϕ.

We write

$$\Sigma \vdash_{\mathbf{K}_{\mathcal{L}}} \psi$$

to mean that a $\mathbf{K}_{\mathcal{L}}$-formula ψ occurs at the end of a deduction in $\mathbf{K}_{\mathcal{L}}$ from Σ.

We can similarly define proofs.

Definition 2.4.3 (Proofs in $\mathbf{K}_{\mathcal{L}}$). A **proof** in $\mathbf{K}_{\mathcal{L}}$ is a finite sequence of deductions made from the empty set.

Theorems are the results deduced in proofs.

Definition 2.4.4 (Theorems in $\mathbf{K}_{\mathcal{L}}$). A **theorem** of $\mathbf{K}_{\mathcal{L}}$ is a $\mathbf{K}_{\mathcal{L}}$-formula that occurs at the end of a proof. We write

$$\vdash_{\mathbf{K}_{\mathcal{L}}} \psi$$

to mean that ψ is a theorem of $\mathbf{K}_{\mathcal{L}}$.

There is a very deep reason why we restrict the use of (Gen) to deduce $(\forall x_i)\,\phi$ from ϕ in Definition 2.4.2 and the subsequent definitions of proofs and theorems in $\mathbf{K}_{\mathcal{L}}$. Unfortunately, we are not yet in a position to discuss it. But we will in due course: see Counterexample 2.4.9.

2.4.2 Tools for Deduction

An important objective of our discussion on the formal system $\mathbf{K}_{\mathcal{L}}$ is to show that the theorems of $\mathbf{K}_{\mathcal{L}}$ are precisely the logically valid formulae. In Section 2.2, we established a bridge between Propositional and First-Order Logic. In particular, we showed that the theorems in the formal system $\mathbf{L}$—that is, the tautologies—are all logically valid (see Theorem 2.2.10). We can actually show something stronger.

Theorem 2.4.5. *Let ϕ be an $\mathbf{K}_{\mathcal{L}}$-formula that is a substitution instance of a tautology in propositional logic. Then, $\vdash_{\mathbf{K}_{\mathcal{L}}} \phi$.*

Proof. `sorry` □

While the above gives us a way of dealing with formulae that come from first-order logic, we also have a way of getting access to the first of two formulae linked by a $\rightarrow$ connective.

Theorem 2.4.6 (Deduction Theorem for $\mathbf{K}_{\mathcal{L}}$). *Let Σ be a set of $\mathcal{L}$-formulae and let ϕ and ψ be $\mathcal{L}$-formulae. If $\Sigma \cup \{\phi\} \vdash_{\mathbf{K}_{\mathcal{L}}} \psi$, then $\Sigma \vdash_{\mathbf{K}_{\mathcal{L}}} (\phi \to \psi)$.*

Proof. sorry □

2.4.3 Soundness

In this subsection, we show one direction of our claim that the theorems of $\mathbf{K}_{\mathcal{L}}$ are precisely the logically valid formulae (cf. Definition 2.2.6).

Theorem 2.4.7 (Soundness Theorem for $\mathbf{K}_{\mathcal{L}}$). *Let ϕ be a $\mathbf{K}_{\mathcal{L}}$-formula. If $\vdash_{\mathbf{K}_{\mathcal{L}}} \phi$, then $\models \phi$. In other words, if ϕ is a theorem of $\mathbf{K}_{\mathcal{L}}$, then ϕ is logically valid.*

Proof. As with the proof of the Deduction Theorem (Theorem 2.4.6), we follow the proof of the Soundness Theorem for **L** (Theorem 1.3.5). Our strategy for showing that all formulae deduced from the empty set are logically valid will be to prove the following.

1. The axioms of $\mathbf{K}_{\mathcal{L}}$ are logically valid.

   We do not need to worry about the axioms (A1), (A2) and (A3), because they are substitution instances of propositional tautologies, making them logically valid by Theorem 2.2.10. Furthermore, the axiom (K1) is logically valid by sorryFinally, the axiom (K2) is logically valid by sorry

2. Deductions preserve logical validity.

   sorry

□

We have an important corollary that mirrors sorry

Before we can generalise the Soundness Theorem for first-order logic (just like we did the Soundness Theorem for propositional logic), we define the following notation.

Convention. Let Σ be a set of $\mathcal{L}$-formulae and ϕ an $\mathcal{L}$-formula. We say

$$\Sigma \models \phi$$

if for every $\mathcal{L}$-structure $\mathcal{A}$ and every valuation $\mathbf{v}$ of $\mathcal{L}$ in $\mathcal{A}$, if $\mathbf{v}[\sigma] = \mathbf{T}$ for every $\sigma \in \Sigma$, then $\mathbf{v}[\phi] = \mathbf{T}$.

We can now generalise the Soundness Theorem, and express it in a similar notation to what we used in Theorem 2.4.7.

Corollary 2.4.8. *Let Σ be a set of $\mathbf{K}_{\mathcal{L}}$-formulae and let ψ be a $\mathbf{K}_{\mathcal{L}}$-formula. If $\Sigma \vdash_{\mathbf{K}_{\mathcal{L}}} \psi$, then $\Sigma \models \phi$.*

Proof. sorry □

We are now in a position to comment on the restriction we impose on the use of the deduction rule (Gen) in Definition 2.4.2 and the subsequent definitions of proofs and theorems in $\mathbf{K}_{\mathcal{L}}$. We want our formal system to be **sound**, but it turns out that if we did not have this restriction, we would be able to construct a counterexample disproving soundness.

Counterexample 2.4.9 (Disproving soundness when we allow unrestricted use of (Gen)). If we allowed (Gen) to deduce $(\forall x_i)\,\phi$ from formulae in which x_i occurs freely, in a language $\mathcal{L}$ with a unary relation R, we would have

$$\{R(x_1)\} \vdash_{\mathbf{K}_{\mathcal{L}}} (\forall x_1)\, R(x_1)$$

In structures, such a statement would make no sense, as it would say that we can deduce a fact about all constants from a fact about a single constant. Indeed, by the Deduction Theorem (Theorem 2.4.6), if $\{R(x_1)\} \vdash_{\mathbf{K}_{\mathcal{L}}} (\forall x_1)\, R(x_1)$, then we would have

$$\vdash_{\mathbf{K}_{\mathcal{L}}} (R(x_1) \to (\forall x_1)\, R(x_1))$$

However, in an $\mathcal{L}$-structure like $\mathcal{A} = \langle \mathbb{N}; (=0) \rangle$, with domain $\mathbb{N}$ and unary relation being the condition that an element be equal to 0, we have

$$\mathcal{A} \not\models (R(x_1) \rightarrow (\forall x_1)\, R(x_1))$$

because if you take the valuation $\mathbf{v}$ that maps x_1 to $0 \in \mathbb{N}$, we have that

$$\mathbf{v}((R(x_1) \rightarrow (\forall x_1)\, R(x_1))) = \mathbf{F}$$

because indeed, it is the case that $x_1 = 0$ and $\mathbf{v}[(\forall x_1)\, R(x_1)] = \mathbf{F}$ for any valuation that maps x_1 to 0. In particular, we would have that

$$\not\models (R(x_1) \rightarrow (\forall x_1)\, R(x_1))$$

In other words, despite having the deduction

$$\vdash_{\mathbf{K}_{\mathcal{L}}} (R(x_1) \rightarrow (\forall x_1)\, R(x_1))$$

the formula

$$(R(x_1) \rightarrow (\forall x_1)\, R(x_1))$$

would not be a valid theorem in $\mathcal{L}$. This is a direct contradiction of the Soundness Theorem (Theorem 2.4.7). Therefore, the restriction on the use of (Gen) **is necessary** for soundness.

Apart from offering an explanation for the restriction on the use of the deduction rule (Gen) in first-order logic, the Soundness Theorem gives us more properties we can use to make deductions. It brings us an important step closer to proving that the theorems of first-order logic are precisely the logically valid formulae.

2.4.4 Consistency

Again, there is some overlap with propositional logic.

Definition 2.4.10 (Consistency). A set Σ of $\mathbf{K}_{\mathcal{L}}$-formulae is **consistent** if there is no formula ϕ such that

$$\Sigma \vdash_{\mathbf{K}_{\mathcal{L}}} \phi \qquad \text{and} \qquad \Sigma \vdash_{\mathbf{K}_{\mathcal{L}}} (\neg\phi)$$

The Soundness Theorem allows us to prove an important result.

Theorem 2.4.11 (The Consistency Theorem). *The empty set $\emptyset$ of $\mathbf{K}_{\mathcal{L}}$-formulae is consistent.*

Proof. sorry □

Convention. We say $\mathbf{K}_{\mathcal{L}}$ **is consistent**.

For the remainder of this subsection, fix a set Σ of $\mathbf{K}_{\mathcal{L}}$-formulae. We mention an analogue of the intuitive idea that 'a proof of False yields anything'.

Proposition 2.4.12. *If Σ is inconsistent, then for all $\mathbf{K}_{\mathcal{L}}$-formulae χ, we have*

$$\Sigma \vdash_{\mathbf{K}_{\mathcal{L}}} \chi$$

Proof. sorry □

We state a few more consistency results that are analogous to propositional logic. Give cross references!

Proposition 2.4.13. *Suppose Σ is consistent and consists entirely of closed $\mathbf{K}_{\mathcal{L}}$formulae. Let ϕ be a closed $\mathbf{K}_{\mathcal{L}}$-formula. If $\Sigma \nvdash_{\mathbf{K}_{\mathcal{L}}} \phi$, then $\Sigma \cup \{(\neg\phi)\}$ is consistent.*

Proof. sorry □

Proposition 2.4.14 (Analogue of Lindenbaum's Lemma). *Suppose Σ consists entirely of closed $\mathbf{K}_{\mathcal{L}}$-formulae. There is a consistent set Σ^* of closed $\mathbf{K}_{\mathcal{L}}$-formulae such that $\Sigma \subseteq \Sigma^*$.*

Proof. sorry □

The above results have important consequences that we will discuss in the next section. These will be instrumental in proving a converse result for Theorem 2.4.7.

2.4.5 Model Existence

In this subsection, we state and sketch the proof of an important result that will serve as a stepping-stone towards proving converse of the Soundness Theorem for $\mathbf{K}_{\mathcal{L}}$, known as the Completeness Theorem.

First, we introduce some notation.

Convention. Let $\mathcal{A}$ be an $\mathcal{L}$-structure and Σ a set of $\mathcal{L}$-formulae. We write

$$\mathcal{A} \models \Sigma$$

if for all $\sigma \in \Sigma$, $\mathcal{A} \models \sigma$.

We have a rather surprising existence result for a model for any consistent set of closed formulae.

Theorem 2.4.15 (Model Existence Theorem). *Suppose $\mathcal{L}$ is a countable first-order language and Σ is a consistent set of $\mathcal{L}$-formulae. Then, there is a countable $\mathcal{L}$-structure $\mathcal{A}$ such that $\mathcal{A} \models \Sigma$.*

Proof Sketch. This is quite a lengthy proof, so we do not give more than a sketch here. The details can be found in [**LecNotes2018**]. There are several steps.

Step 1. Extending our Language.

We define a new language $\mathcal{L}^+$ that is an extension of $\mathcal{L}$ with countably many constant symbols. That is, the constants in $\mathcal{L}^+$ are the constants in $\mathcal{L}$ together with countably many new constants $b_0, b_1, b_2, \ldots$. Note that $\mathcal{L}^+$ is also countable because $\mathcal{L}$ is, and we only add countably many new constants.

Step 2. Adding Witnesses.

We can regard Σ as a set of closed $\mathcal{L}^+$-formulae. It is possible to prove that Σ is still consistent as a set of $\mathcal{L}^+$-formulae. In fact, we can extend Σ to a set $\Sigma_\infty \supseteq \Sigma$ such that

- every formula in Σ_∞ is closed

- for every $\mathcal{L}^+$-formula $\theta(x_i)$ with one free variable x_i, there exists some constant b_j in $\mathcal{L}^+$ such that

$$\Sigma_\infty \vdash_{\mathbf{K}_{\mathcal{L}^+}} ((\neg (\forall x_i)\, \theta(x_i)) \to (\neg \theta(b_j)))$$

This formula essentially says

$$\Sigma_\infty \vdash_{\mathbf{K}_{\mathcal{L}^+}} (((\exists x_i)\, (\neg \theta(x_i))) \to (\neg \theta(b_j)))$$

and we say "b_j **witnesses** the existence of x_i satisfying $(\neg \theta(x_i))$".

We say that this process of constructing Σ_∞ "**adds witnesses.**"

Step 3. Constructing a Complete Set of Formulae.

We can now apply Lindenbaum's Lemma (Proposition 2.4.14) to Σ_∞: there exists a consistent set Σ^* of closed $\mathcal{L}^+$-formulae such that for every closed L^+-formula ϕ, we have either $\Sigma^* \vdash_{\mathbf{K}_{\mathcal{L}^+}} \phi$ or $\Sigma^* \vdash_{\mathbf{K}_{\mathcal{L}^+}} (\neg \phi)$. We will not use Σ^* right away, but it will soon become clear what the relevance of this set is.

Step 4. Constructing an $\mathcal{L}^+$-structure.

sorry

Step 5. A Result on L^+.

sorry

Step 6. Restricting Step 5. to $\mathcal{L}$.

sorry

□

What's more surprising than the actual existence result is the accompanying countability result: the model we constructed above is countable!

It turns out that the Model Existence Theorem has a very important consequence. So important is this consequence that we shall give it its own subsection.

2.4.6 Compactness

We now state and prove one of the most important theorems of first-order logic, known as the Compactness Theorem.

Theorem 2.4.16 (The Compactness Theorem for First-Order Logic). *Assume that $\mathcal{L}$ is a countable first-order language. Let Σ be a set of closed $\mathcal{L}$-formulae. If every finite subset of Σ has a model, then Σ has a model.*

Proof. Suppose Σ has no model. Then, by the Model Existence Theorem (Theorem 2.4.15), Σ must be inconsistent (because if it is consistent, Theorem 2.4.15 tells us it must have a model).

If Σ is inconsistent, there exists an $\mathcal{L}$-formula χ such that $\Sigma \vdash_{\mathbf{K}_\mathcal{L}} \chi$ and $\Sigma \vdash_{\mathbf{K}_\mathcal{L}} (\neg\chi)$. Since deductions in $\mathbf{K}_\mathcal{L}$ are finite, there is a finite subset $\Sigma_0 \subseteq \Sigma$ such that $\Sigma_0 \vdash_{\mathbf{K}_\mathcal{L}} \chi$ and $\Sigma_0 \vdash_{\mathbf{K}_\mathcal{L}} (\neg\chi)$. This makes Σ_0 inconsistent. However, by our assumption that every finite subset of Σ has a model, we have that Σ_0 has a model. This contradicts the Soundness Theorem (Theorem 2.4.7), because it tells us that we then have a model of both χ and $(\neg\chi)$, which the Soundness Theorem does not allow. Therefore, Σ must have a model. □

prove this as a corollary in the soundness section

We are now ready for the converse of the Soundness Theorem.

2.4.7 Completeness

Our strategy will be to first prove the result for closed formulae and then prove it in general.

We begin by introducing some notation.

Convention. Let Σ a set of $\mathcal{L}$-formulae and ϕ an $\mathcal{L}$-formula. We write

$$\Sigma \models \phi$$

if every model of Σ is a model of ϕ. That is, $\Sigma \models \phi$ if for every $\mathcal{L}$-structure $\mathcal{A}$, it holds that if $\mathcal{A} \models \Sigma$, then $\mathcal{A} \models \phi$.

We are now in a position to prove the converse of the Soundness Theorem for $\mathbf{K}_{\mathcal{L}}$ for closed formulae. It turns out to be a consequence of the following important theorem.

Theorem 2.4.17. *Let Σ be a set of closed $\mathcal{L}$-formulae and ϕ a closed $\mathcal{L}$-formula. If $\Sigma \models \phi$, then $\Sigma \vdash_{\mathbf{K}_{\mathcal{L}}} \phi$.*

Proof. sorry □

The case when $\Sigma = \emptyset$ gives us a converse of the Soundness Theorem for $\mathbf{K}_{\mathcal{L}}$ for closed formulae.

Corollary 2.4.18 (A Partial Converse of Soundness)**.** *Let ϕ be a closed $\mathcal{L}$-formula. If $\models \phi$, then $\vdash_{\mathbf{K}_{\mathcal{L}}} \phi$.*

Proof. When $\Sigma = \emptyset$, we do, indeed have that $\Sigma \models \phi$ is the same as $\models \phi$: for any $\mathcal{L}$-structure $\mathcal{A}$, since $\mathcal{A}$ vacuously models every element of Σ, the condition that if $\mathcal{A} \models \sigma$ for all $\sigma \in \Sigma$ then $\mathcal{A} \models \phi$ is precisely the condition that $\mathcal{A} \models \phi$. Theorem 2.4.17 then gives us that $\vdash_{\mathbf{K}_{\mathcal{L}}} \phi$. □

It turns out that Corollary 2.4.18, a converse of the Soundness Theorem for closed formulae, gives us everything we need to prove the result in general.

Theorem 2.4.19 (Gödel's Completeness Theorem for $\mathbf{K}_{\mathcal{L}}$)**.** *Let ϕ be any $\mathcal{L}$-formula. If ϕ is logically valid, then ϕ is a theorem of $\mathcal{L}$. That is, if $\models \phi$, then $\vdash_{\mathbf{K}_{\mathcal{L}}} \phi$.*

Proof. By Corollary 2.4.18, it suffices to consider the case where ϕ is not closed, that is, where ϕ has free variables. sorry □

2.5 First-Order Languages with Equality

In this section, we address a special type of first-order language: one that expresses the notion of equality.

Convention. Denote by $\mathcal{L}^E$ a first-order language with a distinguished binary relation symbol E.

We begin with a discussion on the concept of equality.

2.5.1 The Axioms of Equality

We begin by syntactically defining the axioms we want our symbol E to obey for it to be considered as representing a notion of equality.

Definition 2.5.1 (The Axioms of Equality). The **Axioms of Equality** are collected in a set Σ_E consisting of the following $\mathcal{L}^E$-formulae.

(E1) $(\forall x_1)\, E(x_1, x_1)$

(E2) $(\forall x_1)\,(\forall x_2)\,(E(x_1, x_2) \to E(x_2, x_1))$

(E3) $(\forall x_1)\,(\forall x_2)\,(\forall x_3)\,(E(x_1, x_2) \to (E(x_2, x_3) \to E(x_1, x_3)))$

(E4) For each n-ary relation symbol R in $\mathcal{L}^E$,

$$(\forall x_1) \cdots (\forall x_{2n})\,(R(x_1, \ldots, x_n) \wedge E(x_1, x_{n+1}) \wedge \cdots \wedge E(x_n, x_{2n})) \to (R(x_{n+1}, \ldots, x_{2n}))$$

(E5) For each n-ary function symbol f in $\mathcal{L}^E$,

$$(\forall x_1) \cdots (\forall x_{2n})\,((E(x_1, x_{n+1}) \wedge \cdots \wedge E(x_n, x_{2n})) \to E(f(x_1, \ldots, x_n), f(x_{n+1}, \ldots, x_{2n})))$$

We know these axioms by more familiar names: (E1) is called **reflexivity**; (E2) is called **symmetry**; (E3) is called **transitivity**; (E4) is called **congruence of relations**; and (E5) is called **congruence of functions**.

We now discuss models of equality.

2.5.2 Normal Structures

We have a special name for first-order structures that have a notion of equality.

Definition 2.5.2 (Normal $\mathcal{L}^E$-Structures)**.** An $\mathcal{L}^E$-structure $\mathcal{A}$ is **normal** if $\mathcal{A} \models \sigma$ for all $\sigma \in \Sigma_E$.

In other words, an $\mathcal{L}^E$-structure is **normal** if the symbol E is interpreted in it as equality.

sorry

2.5.3 Normal Models

In this subsection, we state and prove important results closed $\mathcal{L}^E$-formulae. Throughout, fix a set Δ of closed $\mathcal{L}^E$-formulae.

Definition 2.5.3 (Normal Models)**.** A **normal model** of Δ is a normal $\mathcal{L}^E$-structure $\mathcal{B}$ such that $\mathcal{B} \models \sigma$ for all $\sigma \in \Delta$.

In other words, a normal model of Δ is precisely a model of Δ that is normal in $\mathcal{L}^E$.

Recall that Σ_E is the set of $\mathcal{L}^E$-formulae consisting of the Axioms of Equality (cf. Definition 2.5.1). We have an equivalent condition for a model to be normal.

Lemma 2.5.4. *Δ has a normal model if and only if $\Sigma_E \cup \Delta$ has a model.*

Proof. sorry □

We have a compactness result for normal structures.

Theorem 2.5.5 (The Compactness Theorem for Normal Structures)**.** *Assume that $\mathcal{L}^E$ is countable. If every finite subset of Δ has a normal model, then Δ has a normal model.*

Proof. By definition, every normal $\mathcal{L}^E$-sturcture is a model of Σ_E. Assuming that every finite subset of Δ admits a normal model, then every finite subset of $\Delta \cup \Sigma_E$ has a model, because any normal model of every finite subset of Δ is both a model of Δ and a normal $\mathcal{L}^E$-structure, and therefore, model of Σ_E. Then, by the standard version of the Compactness Theorem (Theorem 2.4.16), $\Delta \cup \Sigma_E$ has a model. Then, by Lemma 2.5.4, Δ has a normal model. □

sorry

2.6 Linear Orders

In this section, we discuss results on ordered sets. Throughout, denote by $\mathcal{L}^E$ a first-order language with a binary relation symbol $=$ (for 'equality') and another binary symbol $\leq$ (for 'comparisons').

We begin by defining what a linear order is.

Definition 2.6.1 (Linear Order). A **linear order** is a normal $\mathcal{L}^E$-structure $\mathcal{A} = \langle A; \leq_A \rangle$ that models the following $\mathcal{L}^E$-formulae:

$$\phi_1 : (\forall x_1)\,(\forall x_2)\,(((x_1 \leq x_2) \wedge (x_2 \leq x_1)) \leftrightarrow (x_1 = x_2))$$
$$\phi_2 : (\forall x_1)\,(\forall x_2)\,(\forall x_3)\,(((x_1 \leq x_2) \wedge (x_2 \leq x_3)) \to (x_1 \to x_3))$$
$$\phi_3 : (\forall x_1)\,(\forall x_2)\,((x_1 \leq x_2) \vee (x_2 \leq x_1))$$

Chapter 3

Set Theory

We now come to the final component of this module: the theory of sets. The purpose of this chapter is to motivate and discuss the results that have led to the development of the modern theory of sets.

The structure of this chapter will be as follows. We will begin by discussing naïve set theory and the motivations for the axiomatisation of set theory by Zermelo and Fraenkel. We will then state and explain some of the Zermelo-Fraenkel axioms. We will then develop some theory and prove them using *naïve* set-theoretic arguments, and mention what breaks down in the axiomatic sense when we use the axioms developed up to that point. This will motivate the addition of an appropriate axiom to make the naïve arguments valid in the axiomatic sense, all the while steering clear of the paradoxes that come with a non-axiomatic treatment of set theory.

3.1 Naïve Set Theory

We begin by discussing the reasons why set theory was axiomatised, as well as a few basic axioms that will already allow us to do a non-trivial amount of mathematics, following which we will motivate, state and explore the more nuanced axioms.

3.1.1 Familiar Set-Theoretic Constructions

There are five important notions from naïve set theory with which we are already quite familiar:

1. Extensionality
2. The Natural Numbers
3. sorry
4. Ordered Pairs
5. Functions

3.1.2 The Concept of Cardinality

Informally, the cardinality of a set is the number of elements it has. We briefly explore this notion in more detail in this section.

Definition 3.1.1 (Equinumerosity). We say sets A and B are **equinumerous**, or that A and B **have the same cardniality**, if there is a bijection between A and B.

There are two ways by which we will denote this.

Convention. If A and B are equinumerous, we write

$$A \approx B \text{ or } |A| = |B|$$

(regardless of the actual cardinalities of A and B).

We also define finiteness and countability.

Definition 3.1.2 (Finiteness). A set A is **finite** if it is equinumerous with some element of $\mathbb{N}$.

The idea is that we view any element $n \in \mathbb{N}$ as a set with b elements. Ie, for all $n \in \mathbb{N}$,

$$n = \{0, 1, \ldots, n-1\}$$

with $0 = \emptyset$, $1 = \{\emptyset\}$, and so on.

Admittedly, this is not the soundest thing to do, since we have yet to define things like $\emptyset$, $\cup$, and

$\mathbb{N}$. However, since the purpose of this section is to offer motivation, we do not make anything of it. We now define countable infiniteness.

Definition 3.1.3 (Countable Infiniteness)**.** A set A is **countably infinite** if it is equinumerous with all of $\mathbb{N}$.

We adopt the following definition for countability.

Definition 3.1.4 (Countability and Uncountability)**.** A set A is **countable** if it is either finite or countably infinite; similarly, we say A **uncountable** if it is neither (ie, not countable).

We have many conditions for countability. We recall some of them.

Proposition 3.1.5. *Let A and B be countable sets.*

1. Every subset of A (and of B) is countable.
2. $A \times B$ is countable.
3. $A \cup B$ is countable.
4. $A \sqcup B$ is countable.

The above basic facts do not warrant proving in this module.

We even have a somewhat surprising result about countable unions of countable sets.

Proposition 3.1.6. *Let $A_0, A_1, A_2, \ldots$ all be countable sets. Then, the countable union*

$$\bigcup_{n \in \mathbb{N}} A_n$$

is countable.

We do not prove this fact either, but we do mention that **every proof of it uses the Axiom of Chocie**.

add reference

Example 3.1.7.

3.2 The Zermelo-Fraenkel Axioms

In this section, we define the **Zermelo-Fraenkel Axioms** on which most of modern mathematics is built.[1]

We will begin by mentioning what we *mean* be a Zerlemo-Fraenkel Axiom. This is not a formal definition per se, but a convention we adopt for our own convenience.

Convention. The Zermelo-Fraenkel Axioms are first-order formulae in some first-order language $\mathcal{L}$ that contains

- Variable symbols representing sets
- A binary relation symbol $=$ representing equality
- A binary relation symbol $\in$ representing membership

The Zermelo-Fraenkel Axioms are contained in a set of formulae Δ_{ZF} that is modelled by all modern mathematics that does not use the Axiom of Choice. When we discuss the Axiom of Choice, we will consider the set Δ_{ZFC} instead, which consists of all elements of Δ_{ZF} as well as the Axiom of Choice. All structures in modern mathematics are models of Δ_{ZFC}.

We also now

We are now ready to begin stating the Zermelo-Fraenkel Axioms.

ZF Axiom 1. `sorry`

ZF Axiom 2. `sorry`

ZF Axiom 3. `sorry`

ZF Axiom 4. `sorry`

ZF Axiom 5. `sorry`

[1] I personally prefer the Calculus of Inductive Constructions, but what do I know...

ZF Axiom 6. `sorry`

`sorry`

3.2.1 The Axiom of Infinity

The seventh axiom posits the existence of a notion of infinity. Before we can state it, we need to define a notion of *inductive sets*.

Definition 3.2.1 (The Successor). Let a be an arbitrary set. The **successor** of a is the set

$$a^{\dagger} := a \cup \{a\}$$

The construction of the natural numbers, with which we expect the reader to be familiar, involves applying the successor function repeatedly to the empty set.

We can now define what it means for a set to be inductive. There is an important difference between the definition below and the notion of inductive types in programming languages like Lean that are built using the Calculus of Inductive Constructions: here, we have *specified* a successor *operation* that takes in a *set* and gives out another *set*, whereas when constructing $\mathbb{N}$ in Lean, we have two *constructors*: the distinguished number 0 and the *function* `succ` $: \mathbb{N} \to \mathbb{N}$

Definition 3.2.2 (Inductivity). A set A is said to be **inductive** if

$$\text{Ind}(A) : \left((\emptyset \in A) \wedge (\forall x)\left((x \in A) \to \left(x^{\dagger} \in A\right)\right)\right)$$

holds in A.

The idea behind the Axiom of Infinity is that by our informal understanding of infinity, an inductive set cannot be finite.

ZF Axiom 7 (The Axiom of Infinity). The Axiom of Infinity posits the existence of an inductive set. Ie, it is the first-order formula

$$(\exists A)\,(\mathrm{Ind}(A))$$

3.2.2 The Axiom of Replacement

The primary motivation for this axiom comes from the study of ordinals. In particular, we will see it used in the proof of sorry, where without the axiom of replacement, the constructions in the argument become dubious. **It is not a bad idea to read sorry before reading this section, because that is where we find the motivation.** Indeed, in lectures, too, the material on ordinals was convered **before** the material in this section.

We begin by defining an operation on sets.

Definition 3.2.3 (Operation on Sets). We say an $\mathcal{L}$-formula $F(x, y, z_1, z_2, z_3, \ldots, z_r)$ is an **operation on sets** if it satisfies the property that whenever $s_1, \ldots, s_r$ are sets and a is a set, there is a *unique* set b such that $F(a, b, s_1, \ldots, s_r)$ holds.

We can think of an operation of sets as mapping the set a to the set b given **parameters** $s_1, \ldots, s_r$. Indeed, the variables $z_1, \ldots, z_r$ are sometimes referred to as the **parameter variables of** F.

We are no stranger to the concept of an operation on sets. Indeed, endofunctors in the category of sets are all operations on sets, though these are not the only examples.

Example 3.2.4 (Familiar Operations on Sets). We give two examples, one without parameters and one with one parameter.

1. Define $F(a, b)$ to mean 'b is the power set of a.'
2. Define $F(a, b, s)$, with parameter s, to mean 'b is the set of functions from a to s.'

We are now ready to state the Axiom of Replacement. Technically, we don't have an *axiom* but an *axiom scheme*, which means we have *several axioms indexed by formulae*.

ZF Axiom 8 (The Axiom Scheme of Replacemenet). Suppose $F(x, y, z_1, \ldots, z_r)$ is an operation on sets. Let $A, s_1, \ldots, s_r$ be sets. Then, there is a set B given by

$$B = \{b \mid (\exists a)((a \in A) \wedge F(a, b, s_1, \ldots, s_r))\}$$

This axiom scheme gets its name from the fact that B is constructed by "replacing" every $a \in A$ with the corresponding set b given by the operation F and parameters $s_1, \ldots, s_r$.

3.3 Well-Ordered Sets

3.3.1 Products and Sums

3.3.2 Segments

3.4 The Theory of Ordinals

As children, we are introduced to two notions of counting: the *cardinal numbers* and the *ordinal numbers*. We are taught that the *cardinal numbers* are the numbers of *quantity*—that is, "one", "two", "three", and so on—whereas the *ordinal numbers* are the numbers of *ordering*—that is, "first", "second", "third", and so on. It turns out there is a reason for this. While the cardinal numbers represent different possible *cardinalities* of finite sets, it turns out that the ordinal numbers represent a broader notion of ordering that is not limited to the set of natural numbers.

Broadly speaking, ordinals are nice well-ordered sets that mimic the properties of the natural numbers. The idea is to try and replicate the property in $\mathbb{N}$—at least, when $\mathbb{N}$ is defined using the successor † (cf. Definition 3.2.1)—that the ordering comes from both membership and inclusion.

3.4.1 Transitive Sets

We begin by defining the notion of transitive sets.

Definition 3.4.1 (Transitivity). A set X is **transitive** if every element of X is also a subset of X, that is, if

$$(\forall y)\left((y \subset x) \to (y \in X)\right)$$

holds in X.

We have examples of transitive sets that come from the set of natural numbers.

Example 3.4.2. The set $3 = \{0, 1, 2\} = \{\emptyset, \{\emptyset\}, \{\emptyset, \{\emptyset\}\}\}$ is transitive. For example, $2 \in 3$ and $2 \subset 3$ both hold.

However, not all subsets of the naturals are transitive.

Non-Example 3.4.3. $\{0, 2\}$ is not a transitive set. $2 \in \{0, 2\}$ but $2 \not\subset \{0, 2\}$.

The reason why Example 3.4.2 works is the following.

Lemma 3.4.4. *If X is a transitive set, then so is $X^\dagger = X \cup \{X\}$.*

Proof. sorry □

It does, of course, remain to show that 2 (and by recursion, 1 and 0) are all transitive to show that 3 is, indeed, transitive, but this is easily done. In fact, we can go one step further.

Theorem 3.4.5. *$\mathbb{N}$ is transitive.*

Proof. sorry □

We are now ready to define ordinals.

3.4.2 Ordinals: The Fundamentals

The idea is for an ordinal to be a transitive set in which the ordering and membership relations are related. This is true in the natural numbers, for example, provided we define the successor function the way we have defined the † operation (cf. Definition 3.2.1).

Definition 3.4.6 (Ordinal). A set α is an **ordinal** if

1. α is a transitive set
2. The relation $<$ on α given by

$$x < y \iff x \in y$$

   for all $x, y \in \alpha$ is a strict well-ordering on α.

Here, we pause to make an important observation. In Definition 3.4.6, we **explicitly disallow** $\alpha \in \alpha$. The reason for this is that if we had $\alpha \in \alpha$, we would have $\alpha < \alpha$, which contradicts strictness. It is therefore supremely important that the strictness of the well-ordering be included in the definition of an ordinal.

We now relate ordinals and the successor †.

Lemma 3.4.7. *If α is an ordinal, then $\alpha^\dagger = \alpha \cup \{\alpha\}$ is also an ordinal.*

Proof. `sorry` □

We can, in fact, show that $\mathbb{N}$ is an ordinal. However, we are not yet ready to do this. We need more machinery.

We have the following properties of ordinals that follow from the definition.

Proposition 3.4.8 (Relationships between Ordinals). *Let α and β be sets.*

1. If α is an ordinal and $\beta \in \alpha$, then β is an ordinal.
2. If α and β are both ordinals and $\alpha \subsetneq \beta$, then $\alpha \in \beta$.

Proof.

1. `sorry`

2. `sorry`

□

It turns out the above properties allow us to compare ordinals and define a notion of ordering between them.

3.4.3 Ordering Ordinals

While there are too many ordinals for there to be a *set* of all ordinals, we can reason collectively about the **class of all ordinals**, where the word "class" is used in the category theoretic sense. In fact, one can show that this class is well-ordered.

sorry

3.4.4 Transfinite Induction

In this subsection, we explore a proof strategy that resembles mathematical induction but that works on ordinals instead of the natural numbers. Note that at first glance, it appears that we have ignored the "base case" analogue for ordinals, but observe that (☎) includes the case where $\alpha = \emptyset$.

Theorem 3.4.9 (The Principle of Transfinite Induction). *Suppose $P(\cdot)$ is a first-order formula in the language $\mathcal{L}$ of sets with a free variable. If, for all ordinals α, we have the property that*

If $P(\beta)$ holds for all ordinals $\beta < \alpha$, then $P(\alpha)$ holds. (☎)

Proof. sorry □

There are many results that can be proven using transfinite induction. Here is one example.

Theorem 3.4.10. *If α is an infinite ordinal, then $|\alpha \times \alpha| = |\alpha|$. That is, there is a bijection between $\alpha \times \alpha$ and α.*

Proof. First, observe that if α is countably infinite, then the result holds, because we know that the Cartesian product of countably infinite sets is countably infinite. We focus our attention on the case where α is uncountably infinite.

sorry ☐

Corollary 3.4.11. *If $(A; \leq)$ is an infinite well-ordered set, then $|A \times A| = |A|$.*

Proof. By sorry, we know that there is an ordinal α with $(A; \leq) \simeq (\alpha; \in)$. Therefore,

$$|A \times A| = |\alpha \times \alpha| = |\alpha| = |A|$$

☐

3.4.5 Transfinite Recursion

In the natural numbers, there is more than one use of Peano's Fifth Axiom: it is used not only to prove results by induction, but it is also used to construct things inductively. In this subsection, we explore the notion of transfinite recursion, which provides us with a similar framework that works over ordinals instead of just natural numbers.

The objective of transfinite recursion is to allow us to construct, for ordinals α, sets $G(\alpha$ that are obtained from $G(\beta)$ (for $\beta < \alpha$) by applying some operation F (cf. Definition 3.2.3). In other words, we want to find an operation on sets F that allows us to construct a family of sets $G(\cdot)$, indexed by ordinals, such that F, when applied to the sets $G(0), G(1), \ldots, G(\beta), \ldots$ indexed by ordinals that are less than α, gives $G(\alpha)$.

Theorem 3.4.12 (The Principle of Transfinite Recursion). *Let F be an operation on sets. Then, there is an operation G such that for all ordinals α, we have*

$$G(\alpha) = F(G \restriction \alpha)$$

Furthermore, G is unique on ordinals, in the sense that if G' is another operation such that

$$G'(\alpha) = F(G' \restriction \alpha)$$

for all ordinals α, then for all ordinals α,

$$G(\alpha) = G'(\alpha)$$

The proof is quite technical, and was omitted from lectures. It can be found in [**LecNotes2025**].

Note that the operation G given in the statement of Theorem 3.4.12 is not unique in an absolute sense: we have no control over how G behaves on sets that are not ordinals. Theorem 3.4.12 only tells us that G is unique on ordinals.

An interesting application of the principle of transfinite recursion is the following proof of the Lindenbaum Lemma (cf. Proposition 2.4.14) for languages whose alphabet is well-ordered.

Corollary 3.4.13 (Lindenbaum Lemma for Well-Ordered Languages). *Let $\mathcal{L}$ be a first-order language whose alphabet of symbols is well-ordered (cf. sorry). If Σ is a set of closed $\mathbf{K}_{\mathcal{L}}$-formulae, then there is a consistent set Σ^* of closed $\mathbf{K}_{\mathcal{L}}$-formulae such that $\Sigma \subseteq \Sigma^*$.*

Proof. sorry □

3.5 The Theory of Cardinals

3.5.1 The Axiom of Choice

sorry

Lemma 3.5.1. *Assume all the ZFC Axioms, that is, sorry. Suppose A and B are non-empty sets. Then, there is an injective function from A to B if and only if there is a surjective function from B to A.*

Proof. We need the Axiom of Choice to construct an injection given a surjection, but we do not need it to construct a surjection given an injection.

($\implies$) Assume that there exists an injection $f : A \hookrightarrow B$. Since A is non-empty, there exists some element $a_0 \in A$. Define $h : B \to A$ in the following manner:

$$\forall b \in B, \qquad h(b) = \begin{cases} f^{-1}(b) & \text{if } b \in f(A) \\ a_0 & \text{if } b \notin f(A) \end{cases}$$

where $f^{-1}(b)$ is the unique $a \in A$ such that $f(a) = b$, as guaranteed by the injectivity of f. It is easily seen that h is surjective, because every $a \in A$ is the image of $f(a)$ in h.

($\Longleftarrow$) sorry

□

3.5.2 The Notion of Cardinality

Throughout this subsection, assume **all** the ZFC Axioms.

We remarked, at the beginning of our discussion of ordinals, that the cardinal numbers are the numbers of size and the ordinal numbers are the numbers of ordering. In this section, we will discuss what the 'numbers of size' actually represent.

We begin by defining our primary object of study in this section.

Definition 3.5.2 (Cardinal). An ordinal α is a **cardinal** if for all ordinals $\beta < \alpha$, α is not equinumerous with β.

Informally, cardinals are merely sets with a well-understood notion of cardinality. There are many examples.

Example 3.5.3 (Familiar Cardinals). Every natural number is a cardinal. Furthermore, the set of all natural numbers is a cardinal.

We can also see that infinite ordinals do not behave nicely with the notion of cardinality, in the sense that they might be too large.

Non-Example 3.5.4. If γ is an infinite ordinal, then $\gamma^\dagger$ is *not* a cardinal, because $\gamma^\dagger$ and γ are equinumerous, but $\gamma < \gamma^\dagger$.

3.5.3 The Sequence of Alephs

In this subsection, we define the cardinals $\aleph_\alpha$, indexed by ordinals α, using the Principle of Transfinite Recursion (cf. Theorem 3.4.12).

Definition 3.5.5 (The Sequence of Alephs). Define $\aleph_0 := \omega$, and for all sorry

sorry

3.5.4 Cardinal Arithmetic

In this section, we essentially generalise the operations of addition, subtraction and exponentiation in $\mathbb{N}$ to arbitrary cardinals.

Definition 3.5.6 (Cardinal Arithmetic). Suppose A and B are disjoint sets. Let them be cardinals, with $|A| = \alpha$ and $|B| = \beta$ for ordinals α and β. We define the operations of addition, multiplication and exponentiation in the following manner: sorry

sorry

Below, we give an example where we use sorry result to compute cardinalities.

we really gotta complete our notes

Example 3.5.7. Suppose X is an infinite set. That is, suppose $|X| = \lambda$ for some ordinal $\lambda \geq \omega$. Let S be the set of finite sequences of elements of X. That is,

$$S = \bigcup_{n \in \Omega} X^n$$

We claim that the cardinality of S is also λ.

We have more concrete applications as well.

Example 3.5.8 (Showing that $\dim_{\mathbb{Q}}(\mathbb{R}) = |\mathbb{R}|$). Consider $\mathbb{R}$ as a $\mathbb{Q}$-vector space. Suppose $X \subseteq \mathbb{R}$ spans $\mathbb{R}$: that is, suppose that for all $r \in \mathbb{R}$, there exists some $s \in \mathbb{N}$, $q_1, \ldots, q_s \in \mathbb{Q}$, and $x_1, \ldots, x_s \in X$ such that

$$r = q_1 x_1 + \cdots + q_s x_s$$

We claim that $|X| = |\mathbb{R}|$. We know already that $|X| \leq |\mathbb{R}|$, because $X \subseteq \mathbb{R}$, so it only remains to show that $|\mathbb{R}| \leq |X|$. We do this by applying transitivity.

Let P be the set of pairs

$$\begin{aligned} P &= \left\{ \Big((q_1, \ldots, q_s), (x_1, \ldots, x_s) \Big) \;\Big|\; s \in \mathbb{N},\ q_i \in \mathbb{Q},\ x_i \in X \right\} \\ &\subseteq \left(\bigcup_{n \in \omega} \mathbb{Q}^n \right) \times \left(\bigcup_{n \in \omega} X^n \right) \end{aligned}$$

Then, by sorry, we have that

$$|P| \leq |\mathbb{Q}|\, |X| = \omega \cdot |X| = |X|$$

where the $\cdot$ is the notion of multiplication seen in Definition 3.5.6. If we can show that $|\mathbb{R}| \leq |P|$, then we can show that $|\mathbb{R}| \leq |X|$, from which it will follow, as argued earlier, that $|\mathbb{R}| = |X|$. sorry

3.5.5 Zorn's Lemma

We have already seen the power of the Axiom of Choice. But as the reader might be aware, the mysterious assumption about the nature of mathematics that underlies it takes many equivalent forms, the best known of which are the Axiom of Choice itself, the Well-Ordering Principle, and Zorn's Lemma. Indeed, there is the following well-known mathematical adage:

> *The Axiom of Choice is obviously true; the Well-Ordering Principle is obviously false; and as for Zorn's Lemma, who can say?*

In this subsection, we explore Zorn's Lemma and some of its consequences. Most readers would have encountered Zorn's Lemma when studying other areas of mathematics—particularly algebraic ones—but we will not assume any familiarity with these applications of this strange form of the Axiom of Choice.

sorry

sorry

We now come to the most important theorem of this section.

Theorem 3.5.9 (An Equivalence between the Axiom of Choice and Zorn's Lemma). *Assume all the Zermelo-Fraenkel Axioms. Then, the Axiom of Choice is true if and only if Zorn's Lemma is true. That is,*

$$\mathbf{ZF} \vdash_{\mathsf{K}_{\mathcal{L}_{\in}^{=}}} (\textit{Axiom of Choice} \leftrightarrow \textit{Zorn's Lemma})$$

Proof. Assume that all the Zermelo-Freankel Axioms hold. □

sorry

We now come to an important consequence of Zorn's Lemma: every vector space has a basis. In particular, this applies to infinite-dimensional vector spaces as well.

Theorem 3.5.10. *Let F be a field and V an F-vector space. Then, V has an F-basis.*

Proof. Let $\mathscr{A}$ be the set of all linearly independent subsets of V, partially ordered by inclusion $\subseteq$. The idea is to apply Zorn's Lemma to show that $\mathscr{A}$ contains a maximal element $\mathcal{B}$, which we will then show to be an F-basis of V.

In order to apply Zorn's Lemma, we first need to show that if $(C_i)_{i \in \mathcal{I}} \subseteq \mathscr{A}$ is a chain, indexed by some set $\mathcal{I}$ ordered by inclusions, then $\bigcup_{i \in \mathcal{I}} C_i \in \mathscr{A}$ as well.

sorry □

Exercises

Here, we provide solutions to the exercises discussed in problems classes. We do not include *all* exercises, only the ones that are 'interesting'.

Problems Class 1

Exercise 1.1. *Decide whether the following are true or false, giving reasons. Here, Γ is a set of $\mathbf{L}$-formulae, as are ϕ and ψ.*

1. In every $\mathbf{L}$-formula, the number of opening parentheses (is equal to the number of connectives.
2. If $\Gamma \vdash_{\mathbf{L}} \phi$ and $\Gamma \vdash_{\mathbf{L}} (\phi \to \psi)$, then $\Gamma \vdash_{\mathbf{L}} \psi$.
3. Suppose $\mathbf{v}$ is a propositional valuation and Γ is such that $\mathbf{v}(\Gamma) = \mathbf{F}$. Then, for all ϕ such that $\Gamma \vdash_{\mathbf{L}} \phi$, we have $\mathbf{v}(\phi) = \mathbf{F}$.
4. Suppose $\mathbf{v}$ is a propositional valuation and $\Delta_{\mathbf{v}} = \{\phi \mid \mathbf{v}(\phi) = \mathbf{F}\}$. Then, Δ is consistent.
5. Suppose $\mathbf{v}$ is a propositional valuation and $\Delta_{\mathbf{v}} = \{\phi \mid \mathbf{v}(\phi) = \mathbf{F}\}$. Then, Δ is complete.

Solution.

1. TRUE. Every connective in an $\mathbf{L}$-formula is associated with precisely one sub-$\mathbf{L}$-formula that is not a variable, and the number of pairs of parentheses (which is the number of opening parentheses) is precisely the number of sub-$\mathbf{L}$-formulae that are not variables, because every such sub-$\mathbf{L}$-formula is enclosed in parentheses.

2. TRUE. This is simply modus ponens generalised to arbitrary (potentially nonempty) Γ. While the deduction rule (MP) is technically not something we defined over arbitrary contexts Γ,

we can easily prove that it holds by induction on the length of Γ.

3. FALSE. Let $\mathbf{v}$ be any propositional valuation and ϕ and axiom of $\mathbf{L}$.

4. FALSE. Let $\mathbf{v}$ be any propositional valuation and p a propositional variable. Then, we know $(p \to (\neg p)) \in \Delta_{\mathbf{v}}$. We also know that $(p \wedge (\neg p)) \in \Delta_{\mathbf{v}}$. But the latter is logically equivalent to the negation of the former, making $\Delta_{\mathbf{v}}$ inconsistent.[2]

5. TRUE. Let $\mathbf{v}$ be sorry

Exercise 1.2. *Suppose ϕ is an $\mathbf{L}$-formula and Γ is a set of $\mathbf{L}$-formulae. Do the following **syntactically**, ie, without using the Completeness Theorem. You may use theorems of $\mathbf{L}$ we have proved in lectures or the weekly problem sheets.*

(i) Express the 'Law of the Excluded Middle' $(\phi \vee (\neg\phi))$ as an $\mathbf{L}$-formula and say why this is a theorem of $\mathbf{L}$.

(ii) sorry

Solution.

(i) Recall that any formula of the form $(a \vee b)$ is expressed as $((\neg a) \to b)$. Thus, we have that $(\phi \vee (\neg\phi))$ is expressible as

$$((\neg\phi) \to (\neg\phi))$$

Indeed, we proved in lectures that in $\mathbf{L}$, any formula implies itself. Therefore, the Law of the Excluded Middle is a theorem of $\mathbf{L}$.

(ii) sorry

Exercise 1.3. *Show that the set of connectives $\{\neg, \leftrightarrow\}$ is not adequate.*

Hint. See EdStem.

Solution. The idea is to lok at all possible truth functions of two variables p and q that are obtained using formulae involving $\neg$ and $\leftrightarrow$. sorry

[2] We have actually done something stronger than simply provide a counterexample: we have proved that the statement is false for *any* propositional valuation, not just for *some* propositional valuation

Problems Class 2

Exercise 2.4. *In Example 2.1.23, we introduced a first-order language $\mathcal{L}$ that is appropriate for groups. In this question, we will use the notation defined in Example 2.1.23. Define the $\mathcal{L}$-structures*

$$\mathcal{A} := \langle \mathbb{Z}; =, +, -, 0 \rangle$$
$$\mathcal{B} := \langle \mathbb{Q}; =, +, -, 0 \rangle$$

1. True or false? Give reasons.
   (a) Every $\mathcal{L}$-structure is a group.
   (b) $(\neg m(x_2, m(e, x_1)))$ is a term of $\mathcal{L}$.
   (c) $(\neg m(x_2, m(e, x_1)))$ is a formula of $\mathcal{L}$.
   (d) $R(e, m(x_1, i(x_1)))$ is a formula of $\mathcal{L}$.
   (e) $(\exists x_1)(\neg R(e, m(x_1, i(x_1))))$ is a formula of $\mathcal{L}$.
   (f) $(\exists x_1)((\neg R(e, m(x_1, i(x_1)))))$ is a formula of $\mathcal{L}$.
2. Suppose $\mathbf{v}$ is a valuation of $\mathcal{L}$ in $\mathcal{A}$ such that

$$\mathbf{v}(x_1) = 2$$
$$\mathbf{v}(x_2) = 4$$
$$\mathbf{v}(x_j) = 0 \text{ for } j \neq 1, 2$$

   (a) Compute

$$\mathbf{v}(m(x_2, m(e, x_1)))$$

   (b) Find a term t of $\mathcal{L}$ such that $\mathbf{v}(t) = -6$.
   (c) Is there a term t' of $\mathcal{L}$ such that $\mathbf{v}(t') = 7$?
3. Find a closed $\mathcal{L}$-formula ϕ such that $\mathcal{A} \models \phi$ but $\mathcal{B} \not\models \phi$.

Solution.

1. (a) FALSE. The first-order language $\mathcal{L}$ is syntactic. There is nothing that tells us that the unary function symbol for inversion must actually correspond to a function that undoes

the binary function to get the constant. For example,

$$\langle \mathbb{Z}; \times, -, 0 \rangle$$

is, as per Definition 2.1.19, an $\mathcal{L}$-structure, because it has the same signature as $\mathcal{L}$.

(b) FALSE. As per Definition 2.1.11, a term cannot have a connective. Since the expression

$$(\neg m(x_2, m(e, x_1)))$$

has a connective, it cannot be a term.

(c) FALSE. As per Definition 2.1.15, any formula must contain an atomic formula, which contains a relation symbol. Since the expression

$$(\neg m(x_2, m(e, x_1)))$$

has no relation symbols, it cannot be a formula.

(d) TRUE. Both e and $m(x_1, i(x_1))$ are terms, as they satisfy Definition 2.1.11. Then, since R is a relation symbol, we have that the expression

$$R(e, m(x_1, i(x_1)))$$

is, in fact, an *atomic* formula (cf. Definition 2.1.14). Therefore, by Definition 2.1.15, it is also a formula.

(e) TRUE. This is easily checked.

(f) FALSE. It looks the same as the previous one, but there are two brackets too many! So, the formula is **not well-formed**. (In practice, we only use brackets to disambiguate. In this module, though, we've defined them as an integral part of our syntax. Therefore, in this module, we have to be careful when dealing with brackets!)

2. (a) In $\mathcal{A}$, the (additive) group of integers, this corresponds to the expression

$$4 + 2$$

which we can evaluate to 6.

(b) The following is such a term.

$$m(i(2)\,,m(i(2)\,,i(2)))$$

It corresponds to the expression

$$(-2)+((-2)+(-2))$$

which we can simplify, using the properties of $\mathcal{A}$, to -6.

(c) We cannot, because as per our definition of **v**, the valuation of any term must be an even number.

Problems Class 3

Exercise 3.5. *Let $\mathcal{L}^E$ be the usual first-order language for rings, with a binary relation symbol for equality ($=$), three binary function symbols for addition ($+$), subtraction ($-$) and multiplication ($\cdot$), and two constant symbols for the additive and multiplicative identities (0, 1). Let Φ contain the axioms of a field, expressed in $\mathcal{L}^E$. So, any field is a normal model of Φ.*

Use the compactness theorem for noaml models to prove that for all closed $\mathcal{L}^E$-formulae ϕ, if there are infinitely many primes p such that $\mathbb{F}_p \models \phi$, then there is a field F of characteristic 0 such that $F \models \phi$.

Solution. For all $n \in \mathbb{N}$, define the $\mathcal{L}^E$-formula

$$\sigma_n : (\exists x_1)\cdots(\exists x_n)\left(\bigwedge_{1\le i<j\le n}(x_i \neq x_j)\right)$$

$$\tau_n : \left(\neg\left(\underbrace{1+\cdots+1}_{n \text{ times}}=0\right)\right)$$

Let

$$\Sigma := \{\phi\}\cup\Phi\cup\bigcup_{n\in\mathbb{N}}\{\sigma_n,\tau_n\}$$

Then, applying the compactness theorem for normal models, if there is a normal model for every finite subset of Σ, then there is a normal model for Σ.

Let Δ be a finite subset of Σ. If $\phi \notin \Delta$, then $\mathbb{Q} \models \Delta$, because $\mathbb{Q} \models \Sigma \setminus \{\phi\}$. So, we can assume $\phi \in \Delta$. Then, let n be the largest natural number such that $\sigma_n, \tau_n \in \Delta$. Such an n exists because Δ is finite. In this case, since there are infinitely many primes p such that $\mathbb{F}_p \models \phi$, pick some p such that $p > n$. In this case, we have $\mathbb{F}_p \models \phi$.

Therefore, by the Compactness Theorem for Normal Models (Theorem 2.5.5), there exists a field F of characteristic 0 such that $F \models \phi$. □

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