Introduction to Logic

Notes from the Stanford Course

Amir H. Ebrahimnezhad

University of Tehran

July 10, 2023

Contents

1 Propositional Logic

Propositional Logic deals with the relationships between propositions. The exact definition of a proposition is not possible, but generally, it refers to a potential state of the world that can be either true or false. Examples of propositions include the possibility of rain or the possibility of clouds. It's important to note that a proposition doesn't have to be true; it can also be false, depending on the truth value of another proposition.

In this chapter, we start by examining the syntax rules that establish the language of Propositional Logic. Then, we introduce the concept of a truth assignment, which helps us define the meaning of sentences in Propositional Logic. Next, we provide a systematic approach for evaluating sentences based on a given truth assignment and outline a method for finding truth assignments that satisfy sentences. Finally, we demonstrate how Propositional Logic can be used to formalize Natural Language and Digital Circuits through a series of examples.

1.1 Syntax

In Propositional Logic, we have two types of sentences, simple and compound. Simple sentences are the ones that express a fact about the world. Compound sentences are the ones that are made out of simple sentences and logical relationships.
Simple Sentences in Propositional Logic are often called proposition constants or, sometimes, logical constants.

Callout — In what follows we would write proposition constant as strings of letters, digits and underscores, which begin only by a small leter.

§ **Compound Sentences:** These are made out of simple sentences and logical relaitonships.

- *Negation:* Consists of negation operator and a simepl sentence, called the target. For a statement p it's negate is written as:
$$\neg p \tag{1}$$
- *Conjunction:* This is a sequence of sentences separated by occurrences of the $\wedge$ operator and enclosed by parantheses. The constituent sentences are called *conjuncts.*
$$(p \wedge q) \tag{2}$$
- *Disjunction:* This is a sequence of sentences separated by occurrences of $\vee$ operator and enclosed in parentheses. This constituent sentences are called *disjuncts.*
$$(p \vee q) \tag{3}$$
- *Implication:* This one consists of a pair of sentences separated by the $\Rightarrow$ or $\rightarrow$and enclosed in parentheses. The sentence to the left of the operator is called antecedent, and the sentece to the right is called the consequent. The implication of p and q is shown below:
$$p \rightarrow q \tag{4}$$
- *Biconditional:* A biconditional is a combination of an implication and a reverse implication:
$$p \leftrightarrow q \tag{5}$$

Callout — Note That the constituent sentences within any compund sentence can be either simple sentences or compound sentences or a mixture of the two.

§ **Operator Precedence:** Since more complex compund statements would eventually becom ambiguity, we have the following precedence:

$$\begin{array}{c} \neg \\ \wedge \\ \vee \\ \rightarrow \\ \leftrightarrow \end{array}$$

Callout — Note that just because precedence allows us to delete parentheses in some cases does not mean that we can dispense with parentheses entirely. Consider the example shown earlier. Precedence eliminates the ambiguity by dictating that the sentence without parentheses is an implication with a disjunction as antecedent. However, this makes for a problem for those cases when we want to express a disjunction with an implication as a disjunct. In such cases, we must retain at least one pair of parentheses.

We end the section with two simple definitions that are useful in discussing Propositional Logic. *A propositional vocabulary* is a set of proposition constants. *A propositional language* is the set of all propositional sentences that can be formed from a propositional vocabulary.

1.2 Semantics

The approach to semantics in Logic bears resemblance to that of Algebra. Algebra is not concerned with the real-world meaning of variables; rather, it focuses on the relationships between the values of variables as expressed in equations. Algebraic methods are designed to uphold these relationships, regardless of the interpretation of the variables.

Similarly, Logic does not concern itself with the real-world significance of proposition constants. Instead, it emphasizes the relationships between the truth values of simple sentences and compound sentences that contain them. Like Algebra, logical reasoning methods are detached from the interpretation of proposition constants; what truly matters is the structure and form of the sentences. Although the values assigned to propositions constants are not crucial in the sense just described, we sometimes would like to assign various values to them. Such assignment is called *truth assignment*.

With such truth assignment we would see for each compound sentence there are different results by assigning different truth values.

Callout — Note that for the value *true* we use 1, while for *false* we use 0.

- *Negation:*

ϕ	$\neg\phi$
0	1
1	0

- *Conjunction:*

ϕ	ψ	$\phi \wedge \psi$
1	1	1
1	0	0
0	1	0
0	0	0

- *Disjunction:*

ϕ	ψ	$\phi \vee \psi$
1	1	1
1	0	1
0	1	1
0	0	0

- *Implication:*

ϕ	ψ	$\phi \rightarrow \psi$
1	1	1
1	0	0
0	1	1
0	0	1

- *Biconditional:*

ϕ	ψ	$\phi \leftrightarrow \psi$
1	1	1
1	0	0
0	1	0
0	0	1

1.3 Evaluation & Satisfaction

Evaluation is when we got the truth values of the simple statements and then try to find the truth value of a compound statement made out of those simple ones. *Satisfaction* is the opposite, we try to find the truth values that woudl make a statement false or true, in other sense we consider all the possible truth value combinations for the simple sentences in the compound sentence and evaluate each one.

2 Propositional Analysis

Satisfaction is a relationship between specific sentences and specific truth assignments. In Logic, we are usually more interested in properties and relationships of sentences that hold across all truth assignement. In this section we will look at logical propertirs such as valudity, contingency, and unsatisfiability, then we look at three types of logical relationship between sentences, 1. Logical Entailment, Logical Equivalence and Logical Consistency.

2.1 Logical Properties

As there are sentences that are true or false, there are sentences that are sometimes true and sometimes false. This would lead us to categorized sentences into three types:

> ***Definition*** 1 –*A Sentence is **valid** if and only if it is satisfied by every truth assignment.*

> ***Definition*** 2 –*A sentence is **unsatifiable** if and only if it is not satisfied by any truth assignement.*

> ***Definition*** 3 –*A sentence is **contingent** if and only if there is some truth assignment that satisfies it and some truth assignement that falsifies it.*

Callout — As you might have guessed, in some sense the valid and unsatifiable sentences are useless because they do not rule out any truth assignment as one accepts and the other denies all.

Callout — For many purposes, it is useful to group valid dn continengency into one group called *satisfiable*.

2.2 Logical Equivalence

We say that a sentence ϕ is logically equivalent to a sentence ψ if and only if for every truth assignment that satisfies ϕ, ψ is also satisfied and vice versa. One way of determining whether or not two sentences are logically equivalent is to check the truth table for the proposition constants in the language. This is called the truth table method:

1. We forma truth table for the propositiona constants and add a column for each of the sentences,
2. we Then evaluate the two expressions,
3. At the end we compare the results if the values for the two sentences are the same in every case, then the two sentences are logically equivalent, otherwise, they are not.

2.3 Logical Entailment

We say that a sentence ϕ logically entails a sentence ψ, written $(\phi \vDash \psi)$, if and only if truth assignment that satisfies ϕ also satisfies ψ. More generally, we say that a set of sentences Δ logically entails a sentence ψ, written $\Sigma \vDash \psi$ if and only if the truth assignment that satisfies all of the sentences in Δ also satisfies ψ.

Callout — Note that the relationship of logical entailment is purely logical one. Even if the premises of a problem do not logically entail the conclusion this does not mean that the conclusion is necessarily false, even if the premises are tru. It just means that ist is possible that the conclusion is false.

As with logical equibalence, we can use truth tables to determine whether or note a set of premises logically entails a possible conclusion by checking the truth table for the proposition constants in the language:

1. We form atruth table for thhe proposition constants and add a column for the premisses and a columns for the conclusion.
2. We then evaluate the premises.
3. We evaluate the conclusion,
4. Finally, we compare the results. If every row that satisfies the premises also satisfies the conclusion, then the premises logically entail the conclusion.

2.4 Logical Consistency

A sentence ϕ is consistent with a sentence ψ if and only if there is a truth assignment that satisfies both ϕ and ψ. A sentence ψ is consistent with a set of sentences Δ if and only if there is a truth assignment that satisfies both Δ and ψ.

2.5 Connections Between Properties and Relationships

Before we end this chapter, it is worth noting that there are some strong connections between logical properties like validity and satisfiability and the logical relationships introduced in the preceding sections.

First of all, there is a connection between the logical equivalence of two sentences and the validityof the iconditional sentence built from the two sentences. In particular, we have the following theorem expressing this connection.

Theorem 1 — ***Equibalence Theorem:*** *A sentence ϕ and a sentence ψ are logically equivalent if and only if the sentence $\psi \leftrightarrow \phi$ is valid.*

Why is this true? Consider the definition of logical equivalence. Two sentences are logically equivalent if and only if they are satisfied by the same set of truth assignement. Now recall the semantics of sentences the biconditional operator. Clearly, if two sentences are logically equivalent, they are satisfied by the same truth assignment, and so the corresponding biconditional must be valid. Conversely, if a biconditional is valid the two component sentences must be satisfied by the same truth assignments and so they are logically equivalent.

There is a similar connection between logical entailment between two sentences and the validityof the corresponding implication. And there is a natural externsion to cases of logical entailment involving finite set of sentences.

Theorem 2 — ***Deduction Theorem:*** *A sentence ϕ logically entails a sentence ψ if and only if $\phi \rightarrow \psi$ is valid. More generally, a finite set of sentences $\{\phi_1, \phi_2, \ldots, \phi_n\}$ logically entails ϕ if and only if the compound sentence $\phi_1 \wedge \cdots \wedge \phi_n \rightarrow \phi$ is valid.*

If a dentence ϕ logically entails a sentence ψ, it means that any truth assignment that satisfies ϕ also satisfies ψ. Looking at the semantics of implications, we see that an implication is true if and only if every truth assignmentthat makes the antecedent true also makes the consequent true. Consequently, logical entailment holds exactly when the corresponding implication is valid.

There's also a connection betwenn logical entailment and unsatisfiability. In particular, if a set Δ of sentences logically entails a sentence ϕ, then Δ together with the negation of ϕ myst be unsatifiable. The reverse is also true.

Theorem 3 — ***Unsatisfiability Theorem:*** *A set Δ of sentences logically entails a sentence ϕ if and only if the set of sentences $\Delta \cup \{\neg\phi\}$ is unsatisfiable.*

An interesting consequence of this result is that we can determine logical entailment by checking for unsatisfiability. This turns out to be useful in various logical proof methods.

Finally, consider the definition of logical consistency. A sentence ϕ is logically consistent with a sentence ψ if and only if there is a truth assignment that satisfies both ϕ and ψ. This is equivalent to sayin that the sentence $\phi \wedge \psi$ is satisfiable.

Theorem 4 — ***Consistency Theorem:*** *A sentence ϕ is logically consistent with a sentence ψ if and only if the sentence $\psi \wedge \phi$ is satisfiable. More generally, a sentence ϕ is logically consistent with a finite set of sentences $\{\phi_1, \ldots, \phi_n\}$ if and only if $(\phi_1 \wedge \cdots \wedge \phi_n)$*

The connections described in the preceding section are useful in solving logical problems because they allow us to transform problems of one type into problems of another type. In thinking about these various connections, the main thing to keep in mind is that logical properties and logical relationships are metalevel. They are things we assert in talking about logical sentences; they are not sentences withinour formal language; By contrast, implications and biconditionals and conjuctions are statements within our forma laguage; they are not metalevel statements. In a sense we can implicitly express some ligical relationships within our formal language by writing the corresponding biconditionals and implications and conjunctions and checking for the logical properties of these statements.

3 Direct Proofs

Checking logical entailment with truth tables has merit of being conveptually simple. However, it is not very much practical in the sense that for each logical constants the process doubles. Proof methods provide an alternatice way of checking logical entailment that addresses this problem. In many cases, it is possible to create a proof of a conclusion from a set of premises that is much smaller than the truth table for the language; moreover, it is often possible to find such proofs with less wotk than is neccessary to check the entire truth table.

3.1 Axiom Schemas

Definition 4 –*An **Axiom Schema (Schema)** is an expression satisfying the grammatical rules of our language except for the occurrence of **metavariables** (written as greek letters) in place of various subparts of the expression.*

As an example the following expression is a Schema:

$$\phi \Rightarrow (\psi \Rightarrow \phi) \tag{6}$$

Definition 5 –*An instance of an axiom schema is the sentence obtained by consistently substituting sentences for the metavariables in the rule.*

Definition 6 –*An Axiom Schema is valid of and only if every instance of the schema is valid.*

These schemas are all valid:

$$\text{Reflexivity:} \quad \phi \Rightarrow \phi \tag{7}$$
$$\text{Negation Elimination:} \quad \neg\neg\phi \Rightarrow \phi \tag{8}$$
$$\text{Negation Introduction:} \quad \phi \Rightarrow \neg\neg\phi \tag{9}$$
$$\text{Tautology:} \quad \phi \vee \neg\phi \tag{10}$$

We use both valid and non-valid shcemas. Non-valid schemas are used in defining rules of inference and valid schemas are used as components of deductive proof systems.

3.2 Rules of Inference

> ***Definition*** 7 *–A **Rule of Inference** is a pattern of reasoning consisting of some schemas, called premises, and one or more additional achemas, called conclusions.*

Rules of Iference are often written as below. The schemas above the line are the premises, and the schemas below the line are the conclusions. Here are some rules of inference:

1. This rule of inference is called the *Implication Elimination*, because it eliminates the implication from the first premise.

$$\begin{array}{l} \phi \Rightarrow \psi \\ \phi \\ \hline \therefore\ \psi \end{array}$$

2. *Implication Creation*, tells us that, if a sentence ψ is true, we can infer $\phi \Rightarrow \psi$ for any ϕ whatsoever.

$$\begin{array}{l} \psi \\ \hline \therefore\ \phi \Rightarrow \psi \end{array}$$

3. *Implication Distribution* tells us that implication can be distributed over other implications:

$$\begin{array}{l} \phi \Rightarrow (\psi \Rightarrow \gamma) \\ \hline \therefore\ (\phi \Rightarrow \psi) \Rightarrow (\phi \Rightarrow \gamma) \end{array}$$

4. *Implication Reversal* allows us to infer an implication if we have an implication with the arguments reversed and negated:

$$\begin{array}{l} \neg\psi \Rightarrow \neg\phi \\ \hline \therefore\ \phi \Rightarrow \psi \end{array}$$

> Callout — An instance of a rule of inference is the rule obtained by consistently substituting sentences for the metavariables in the rule. If a metavariable occurs more than once, the same expression must be used for every occurrence.

Remember that there are infinitely many sentences in out anguage. Even though we start with finitely many propositional constants and finitely many operators, we can combine them in arbitrary many ways. The upshot is that there are infinitely many instances of any rule of inference involving metavariables.

> Callout — A rule *applies* to a set of sentences if and only if there is an instance of the rule in which all of the premises are in the set. In this case, the conclusions of the instance are the results of the rule application. Also in using rules of inference, it is important to remember that they apply only to top-level sentences, not to components of sentences. While applying to components sometimes works, it can also lead to incorrect results.

3.3 Direct Proofs

Bz writing down premises, writing instances of axiom schemas, and applzing rules of inference, it is possible to derive conclusions that cannot be derived from the premises in a single step. This idea of stringing things together in this waz leads to the notion of a direct proof.

> ***Definition*** 8 *–A **Direct Proof** of a conclusion from a set of premises in a sequence of sentences terminating in the conclusion in which each item is either (1) a premise, (2) an instance of an axiom schema, (3) or the result of applying a rule of inference to earlier items in sequence.*

> ***Definition*** 9 *–Let R be a set of rules of inference. If there exists a prof of a sentence ϕ from a set Δ of premises using the rules of inference in R, we say that ϕ is provable from Δ using R. We usually write this as $\Delta \vdash_R \phi$, using the provability operator $\vdash$ (which is sometimes called **single turnstile**). If the set of rules is clear from context, we usually drop the subscript, writing just $\Delta \vdash \phi$.*

3.4 Proof Systems

> ***Definition*** 10 *–A **Proof System** is a finite set of axiom schemas and rules of inference. Although it is interesting to consider proof systems with non-valid aciom schemata or unsound rules of inference, in this book we concentrate exclusively on proof systems with valid axiom schemata and sound rules of inference.*

The **Hilbert System** is a well-known proof system for Propositional Logiv. It has one rule of inference, viz. *Implication Elimination.* In addition, the Hilbert systems has three axiom schemas. *Implication Creation, Implication Distribution, Implication Reversal.*

3.5 Soundness and Completeness

We say that a proof szstem is *sound* if and only if every provable conclusion is logically entailed. In other words:

$$\Delta \vdash \phi \Rightarrow \Delta \vDash \phi \tag{11}$$

We say that a proof system is *complete* if and only if every logical conclusion is provable. In other words:

$$\Delta \vDash \phi \Rightarrow \Delta \vdash \phi \tag{12}$$

The Hilbert system is sound and complete for propositional logic. In other words, for this system, logical entailment and provability are identical.

Callout — The upshot of this result is significant. On large problems, the proof method often takes fewer steps than the truth table method. (Disclaimer: In the worst case, the proof method may take just as many or more steps to find an answer as the truth table method.) Moreover, proofs are usually much smaller than the corresponding truth tables. So writing an argument to convince others does not take as much space.