Propositional Logic

We now present a simple but powerful logic called **propositional logic.** We cover the syntax of propositional logic and its semantics. The way in which the truth of sentence is determined. Then we look at **entailment**. The relation between a sentence and another sentence that follows from it and see how this leads to a simple algorithm for logical inference.

Syntax

The **syntax** of propositional logic defines the allowable sentences. The **atomic sentences** consist of a single **proposition symbol**. Each such symbol stands for a proposition that can be true or false. We use symbols that start with an uppercase letter and may contain other letters or subscripts, for example: P, Q, R, $W_{1,3}$ and North. The names are arbitrary but are often chosen to have some mnemonic value. There are two proposition symbols with fixed meanings: True is always-true proposition and False is always-false proposition. **Complex sentences** are constructed from simpler sentences, using parentheses and **logical connectives**. There are five connectives in common use:

- ¬ (not). A sentence such as $\neg W_{1,3}$ is called the **negation** of $W_{1,3}$. A **literal** is either an atomic sentence (**a positive literal**) or a negated atomic sentence (**a negative literal**).
- ∧(and). A sentence whose main connective is ∧, such as $W_{1,3} \wedge P_{3,1}$, is called a **conjunction**. its parts are the **conjuncts**. (The ∧ looks like an “A” for “And”)
- ∨ (or). A sentence using ∨, such as $(W_{1,3} \wedge P_{3,1}) \vee W_{2,2}$, is a **disjunction** the **disjuncts** $(W_{1,3} \wedge P_{3,1})$ and $W_{2,2}$.
- ⇒ (implies). A sentence such as $(W_{1,3} \wedge P_{3,1}) \Rightarrow \neg W_{2,2}$ is called an **implication** (or conditional). Its **premise** or **antecedent** is $(W_{1,3} \wedge P_{3,1})$, and its **conclusion** or **consequent** is $\neg W_{2,2}$**.** Implications are also known as **rules** or **if-then** statements.
- ⇔ (if and only if). The sentence $W_{1,3} \Leftrightarrow \neg W_{2,2}$ is a **biconditional**. Some other books write this as ≡.

E-mail: hasanein.y.m.alhrabi@uomus.edu.iq

Operator precedence

Precedence	Operators
First Precedence	Parenthesis
Second Precedence	Negation
Third Precedence	Conjunction (AND)
Fourth Precedence	Disjunction (OR)
Fifth Precedence	Implication
Six Precedence	Biconditional

Semantics

Having specified the syntax of propositional logic, we now specify its semantics. The semantics defines the rules for determining the truth of sentence with respect to a particular model. In propositional logic, a model simply fixes the **truth value (true or false)**. For every proposition symbol. For example, if the sentences in the knowledge base make use of the proposition symbols $P_{1,2}$, $P_{2,2}$ and $P_{3,1}$, then one possible model is:

$$m1 = \{P_{1,2} = \text{false}, P_{2,2} = \text{false}, P_{3,1} = \text{true}\}.$$

With three proposition symbols, there are $2^3 = 8$ possible models, the semantics for propositional logic must specify how to compute the truth value of any sentence, given a model. This is done recursively. All sentences are constructed from atomic sentences and five connectives; therefore, we need to specify how to compute the truth of atomic sentences and how to compute the truth of sentences formed with each of the five connectives. Atomic sentences are easy:

- True is true in every model and false is false in every model.
- $\neg P$ is true iff P is false in m.
- $P \wedge Q$ is true iff both P and Q are true in m.

E-mail: hasanein.y.m.alhrabi@uomus.edu.iq

- $P \vee Q$ is true iff either *P* or *Q* is true in m.
- $P \Rightarrow Q$ is true unless *P* is true and *Q* is false in m.
- $P \Leftrightarrow Q$ is true iff *P* and *Q* are both true or both false in m.

The rules can also be expressed with **truth tables** that specify the truth value of a complex sentence for each possible assignment of truth values to its components. The truth tables for the five connectives are given in figure below.

P	Q	$\neg P$	$P \wedge Q$	$P \vee Q$	$P \Rightarrow Q$	$P \Leftrightarrow Q$
false	*false*	*true*	*false*	*false*	*true*	*true*
false	*true*	*true*	*false*	*true*	*true*	*false*
true	*false*	*false*	*false*	*true*	*false*	*false*
true	*true*	*false*	*true*	*true*	*true*	*true*

Logical Equivalence

Two sentences α and β are logically equivalent if they are true in the same set of models. We write this $\alpha \equiv \beta$. For example, we can easily show (using truth tables) that $P \wedge Q$ *and* $Q \wedge P$ are logically equivalent. Other equivalences are shown in figure below.

$$
\begin{aligned}
(\alpha \wedge \beta) &\equiv (\beta \wedge \alpha) && \text{commutativity of } \wedge \\
(\alpha \vee \beta) &\equiv (\beta \vee \alpha) && \text{commutativity of } \vee \\
((\alpha \wedge \beta) \wedge \gamma) &\equiv (\alpha \wedge (\beta \wedge \gamma)) && \text{associativity of } \wedge \\
((\alpha \vee \beta) \vee \gamma) &\equiv (\alpha \vee (\beta \vee \gamma)) && \text{associativity of } \vee \\
\neg(\neg\alpha) &\equiv \alpha && \text{double-negation elimination} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\beta \Rightarrow \neg\alpha) && \text{contraposition} \\
(\alpha \Rightarrow \beta) &\equiv (\neg\alpha \vee \beta) && \text{implication elimination} \\
(\alpha \Leftrightarrow \beta) &\equiv ((\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)) && \text{biconditional elimination} \\
\neg(\alpha \wedge \beta) &\equiv (\neg\alpha \vee \neg\beta) && \text{De Morgan} \\
\neg(\alpha \vee \beta) &\equiv (\neg\alpha \wedge \neg\beta) && \text{De Morgan} \\
(\alpha \wedge (\beta \vee \gamma)) &\equiv ((\alpha \wedge \beta) \vee (\alpha \wedge \gamma)) && \text{distributivity of } \wedge \text{ over } \vee \\
(\alpha \vee (\beta \wedge \gamma)) &\equiv ((\alpha \vee \beta) \wedge (\alpha \vee \gamma)) && \text{distributivity of } \vee \text{ over } \wedge
\end{aligned}
$$

E-mail: hasanein.y.m.alhrabi@uomus.edu.iq

Validity

The second concept we will need is **validity**. A sentence is valid if it is true in **all models**. For example, the sentence $P \vee \neg P$ is valid. Valid sentence is also known as **tautologies**. They are necessarily true because the sentence is true in all models. On the other hand, a **contradiction** is a statement that is always false. For example, $P \wedge \neg P$.

Satisfiability

The final concept we will need **satisfiability**. A sentence is satisfiable if it is true in, or satisfied by, some model. Satisfiability can be checked by enumerating the possible models until one is found that satisfies the sentence.

Inference and proofs

Proofs using truth table: one the method for determining the formal validity of an argument, the truth table method is the simplest. Consider the following basic example.

If it is raining, it is wet outside.

It is raining

It is wet outside.

Translating it into symbolic form using propositional variables and logical connectives, we get the following:

$P \Rightarrow Q$

P

Q

E-mail: hasanein.y.m.alhrabi@uomus.edu.iq

The argument is valid if the conclusion follows logically from the two premises. To use a truth table to determine the validity of the argument, we first construct a truth table that includes all of the premises and the conclusion. Testing for validity, we then verify that there is no case in which the ***premises are true and the conclusion false***.

First, list all possible truth values for the component propositions in the left-most columns. Then in the next column, work out the possible truth values for the compound proposition. For the final column, add the conclusion with its possible truth values. The truth table looks as follows:

P	Q	P ⇒ Q	Q
T	T	T	T
T	F	F	F
F	T	T	T
F	F	T	F

To determine validity, look at every row in which both premises ('P ⟹ Q', 'P') are true. For those rows (the first row only, in this case), is there a case in which the conclusion ('Q') is not true? There is not. **Therefore, the argument is valid**.

Inference rules

This section covers the inference rules that can be applied to derive a proof. A chain of conclusions that leads to the desired goal.

1. **Modus Ponens:**

$$\frac{\alpha \Rightarrow \beta\ ,\ \alpha}{\beta}$$

   The notation means that, whenever any sentence of the form $\alpha \Rightarrow \beta$ and α are given, then the sentence β can be inferred.

2. **And-Elimination:**

$$\frac{\alpha \wedge \beta}{\alpha}$$

E-mail: hasanein.y.m.alhrabi@uomus.edu.iq

Another useful inference rule is And-Elimination, which says that, from a conjunction, any of the conjuncts can be inferred.

By considering the possible truth values of α and β, on can show easily that Modus Ponens and And-Elimination are sound once and for all. These rules can then be used in any particular instances where then apply, generating sound inferences without the need for enumerating models. Additionally, All the logical equivalences above, can be used as inference rules.

Let us see how their inference rules can be used. Let start with $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$ to infer $\neg P_{1,2}$.

1	$B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$	Apply biconditional elimination
2	$\left(B_{1,1} \Rightarrow (P_{1,2} \vee P_{2,1})\right) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$	Apply And-Elimination
3	$((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$	Apply Logical equivalence for contrapositives
4	$((\neg B_{1,1} \Rightarrow \neg(P_{1,2} \vee P_{2,1}))$	Apply Modus Ponens
5	$\neg (P_{1,2} \vee P_{2,2})$	Apply De Morgan's rule
6	$\neg P_{1,2} \wedge \neg P_{2,1}$	Apply commutativity of AND
7	$\neg P_{2,1} \wedge \neg P_{1,2}$	Apply And-Elimination
8	$\neg P_{1,2}$	

Proof by resolution: the inference rules covered so far are sound, but we have not discussed the question of completeness for the inference algorithms that use them. Algorithm using inference rules are complete in the sense that they will find any reachable goal, but if the available inference rules are inadequate, then the goal is not reachable. For example, if we removed the biconditional elimination rule, the proof in the preceding section would not got through. This section introduces a single inference rule, **resolution**, that yields a complete inference algorithm.

The resolution rule applies only to clauses (**that is, disjunction of literals**), so it would seem to be relevant only knowledge bases and queries consisting of clauses. How, then can it lead to a complete inference procedure for all of propositional logic? The answer is that every sentence of propositional logic is logically equivalent to a conjunction of clauses. A sentence expressed as a conjunction of clauses is said to be in **conjunctive normal form or CNF.** We now describe a procedure for converting to CNF. We illustrate the procedure by converting the sentence $B_{1,1} \Leftrightarrow (P_{1,2} \vee P_{2,1})$ into CNF. The steps are as follows:

1. Eliminate ⇔, replacing $\alpha \Leftrightarrow \beta$ with $(\alpha \Rightarrow \beta) \wedge (\beta \Rightarrow \alpha)$.
   $(B_{1,1} \Longrightarrow (P_{1,2} \vee P_{2,1})) \wedge ((P_{1,2} \vee P_{2,1}) \Rightarrow B_{1,1})$
2. Eliminate ⇒, replacing $\alpha \Rightarrow \beta$ with $\neg\alpha \vee \beta$:
   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg(P_{1,2} \vee P_{2,1}) \vee B_{1,1})$
3. CNF requires ¬ to appear only in literals, so we "move ¬ inwards" by repeated application of the following equivalences.

   $\neg(\neg\alpha) \equiv \alpha$ (double-negation elimination)
   $\neg(\alpha \wedge \beta) \equiv (\neg\alpha \vee \neg\beta)$ (De Morgan)
   $\neg(\alpha \vee \beta) \equiv (\neg\alpha \wedge \neg\beta)$ (De Morgan)

   In the example, we require just one application of the last rule:

   $(\neg B_{11} \vee P_{1,2} \vee P_{2,1}) \wedge ((\neg P_{1,2} \wedge \neg P_{2,1}) \vee B_{1,1})$

4. Now we have a sentence containing nested ∧ and ∨ operators applied to literals. We apply the distributivity law, distributing ∨ over ∧ wherever possible.
   $(\neg B_{1,1} \vee P_{1,2} \vee P_{2,1}) \wedge (\neg P_{1,2} \vee B_{1,1}) \wedge (\neg P_{2,1} \vee B_{1,1})$

The original sentence is now in CNF, as a conjunction of three clauses.

Resolution Algorithm

E-mail: hasanein.y.m.alhrabi@uomus.edu.iq

Inference procedures based on resolution work by using the principle of proof by contradiction. That is, to show that $KB \models \alpha$, we show that $(KB \wedge \neg\alpha)$ is unsatisfiable.

A resolution algorithm first is convert $(KB \wedge \neg\alpha)$ into CNF. Then, the resolution rule is applied to the resulting clauses. Each pair that contains complementary literals is resolved to produce a new clause, which is added to the set if it is not already present. The process continues until one of two things happens.

- There are no new clauses that can be added.
- Two clauses resolve to yield the empty clause.

E-mail: hasanein.y.m.alhrabi@uomus.edu.iq