Propositional and Predicate Logic

Benjamin Cosman, Patrick Lin and Mahesh Viswanathan

Fall 2020

TAKE-AWAYS

- A proposition is a statement that can either be true (denoted T) or false (denoted F).
- Propositions can be combined using logical operators *not* ($\neg$), *or* ($\vee$), *and* ($\wedge$), *implies* ($\rightarrow$), and *if and only if* ($\leftrightarrow$), to create new propositions.
- Two formulas/propositions P and Q are *logically equivalent* (denoted $P \equiv Q$) if they have the same meaning. That is, P and Q evaluate to the same value in all rows of a truth table, or the formula $P \leftrightarrow Q$ evaluates to T in all rows of a truth table.
- The *contrapositive* of an implication $P \rightarrow Q$ is the formula $(\neg Q) \rightarrow (\neg P)$. The contrapositive $(\neg Q) \rightarrow (\neg P)$ is logically equivalent to $P \rightarrow Q$. The *converse* of an implication $P \rightarrow Q$ is the formula $Q \rightarrow P$.
- Predicates can either be *universally* quantified ($\forall x\ P(x)$) or *existentially* quantified ($\exists x\ P(x)$) to create propositions from predicates. Formulas can have multiple quantifiers and the order in which they appear can influence their meaning.
- A *domain of discourse* identifies the set over which predicate variables take values and the meaning of predicates. It plays an important role in determining the truth of propositions.

Propositional Logic

In mathematics, our goal is to establish mathematical truths by proving statements that hold. The statements we try to prove are called *propositions*.

Definition 1 (Propositions)**.** A *proposition* is a statement that can either be *true* or *false*. We will denote true as T and false as F.

Let us look at some examples and non-examples of propositions.

Example 2. The following statements are all propositions because they are either true or false: "5 is prime"; "Champaign is the capital of Illinois". On the other hand, questions like "When is this going to end?" or commands like "Read these notes!" are not propositions.

The truth of a proposition is not evident from the statement, and may depend on how terms are interpreted. For example, the statement "Champaign is the capital of Illinois" (Example 2) is false if we are referring to the political capital of the state. On the other hand, it is true if we are referring to the intellectual capital of Illinois. We will often find it convenient to abstract propositions and represent them by variables. A *propositional or Boolean variable* is one that takes values either true or false.

Logical Operators

Propositions can be combined using logical operators to create more complex propositions. We will start with the logical operator *not*. The *negation* of a proposition P is written $\neg P$ and read "not P". The meaning of proposition $\neg P$ depends on whether P is T or F. This is often best described using a table that is called a *truth table*. A truth table has columns corresponding to different propositions/formulas, and has a row corresponding to each possible assignment to the propositions that are being combined. Such a table describes how the truth of $\neg P$ depends on P. Figure 1 shows that truth table for $\neg P$, which shows that $\neg P$ takes the value T when P is F, and takes the value F when P is T. The *disjunction* of propositions P and Q

P	$\neg P$
F	T
T	F

Figure 1: Truth table for $\neg P$

Figure 2: Truth table for $P \vee Q$

P	Q	$P \vee Q$
F	F	F
F	T	T
T	F	T
T	T	T

Figure 3: Truth table for $P \wedge Q$

P	Q	$P \wedge Q$
F	F	F
F	T	F
T	F	F
T	T	T

is written $P \vee Q$ and read "P or Q". The meaning of $P \vee Q$ is given by truth table shown in Figure 2 — $P \vee Q$ evaluates to F if both P and Q are both F, and it evaluates to T in all other cases. It is worth noting that $P \vee Q$ evaluates to T when both P and Q are T. Thus it corresponds to an "inclusive or". Another important logical opertor

is *conjunction*, written as $P \wedge Q$ and read "P and Q". The truth table for $P \wedge Q$ (Figure 3) shows that this is T only in the case when both P and Q are T. In all other cases, $P \wedge Q$ is F.

P	Q	$P \to Q$
F	F	T
F	T	T
T	F	F
T	T	T

Figure 4: Truth table for $P \to Q$

The next important logical operator is implication, which is written as $P \to Q$ and read as "P implies Q". P is the *antecedent* of the implication, and Q is the *consequent*. The truth table for $P \to Q$ is shown in Figure 4. It is F only when P is T but Q is F. In all other cases it is T. It is important to observe that $P \to Q$ is T whenever P is F. In mathematics, we are often confronted with proving that a statement of the form $P \to Q$ is true. In such cases, if P can be shown to be never true, then the implication is guaranteed to be true no matter what Q is. When this happens we say that $P \to Q$ holds *vaccuously*. Thus, $P \to Q$ doesn't indicate a "causal" relationship between P and Q. For example, the statement "If $1 = 2$ then every Python program is correct" is a vaccuously true statement since "$1 = 2$" is not true. The fact that "If $1 = 2$ then every Python program is correct" is true says nothing about whether Python programs are correct.

An implication $P \to Q$ has two closely related propositions, called the *contrapositive* and *converse*.

Definition 3 (Contrapositive and Converse)**.** The *contrapositive* of an implication $P \to Q$ is the formula $(\neg Q) \to (\neg P)$.

The *converse* of an implication $P \to Q$ is the formula $Q \to P$.

P	Q	$P \to Q$	$Q \to P$	$P \leftrightarrow Q$
F	F	T	T	T
F	T	T	F	F
T	F	F	T	F
T	T	T	T	T

Figure 5: Truth table for $P \leftrightarrow Q$

Next, the proposition "P if and only if Q" is written as $P \leftrightarrow Q$ and is true if both $P \to Q$ and its converse $Q \to P$ are true. Using the notation we have just developed, we could say $P \leftrightarrow Q \stackrel{\Delta}{=} (P \to Q) \wedge (Q \to P)$. Based on this interpretation, we can draw a truth table for $P \leftrightarrow Q$ as shown in Figure 5. First we will evaluate $P \to Q$ for each assignment of truth values to propositions P and Q. Next we

will evaluate $Q \to P$. It is worth observing that $P \to Q$ and $Q \to P$ do not evaluate to the same value in all cases. $P \leftrightarrow Q$ is a conjunction of $P \to Q$ and $Q \to P$, which gives us the final result by conjuncting the columns for $P \to Q$ and $Q \to P$. Observe that $P \leftrightarrow Q$ is true exactly in the two cases when P and Q have the same values.

Logical Equivalence

LOGICAL PROPOSITIONS THAT LOOK DIFFERENT MAY HAVE THE SAME MEANING. This leads to the notion of *logical equivalence*.

Definition 4 (Logical Equivalence)**.** Two formulas or propositions P and Q are said to be *logically equivalent* (denoted $P \equiv Q$) if they have the same meaning. That is, P and Q evaluate to the same value in all rows of a truth table, or the formula $P \leftrightarrow Q$ evaluated to T in all rows of a truth table.

Let us look at some examples.

P	Q	$P \to Q$	$\neg P$	$(\neg P) \vee Q$
F	F	T	T	T
F	T	T	T	T
T	F	F	F	F
T	T	T	F	T

Figure 6: Truth table for $P \to Q$ and $(\neg P) \vee Q$

Example 5. The formula $P \to Q$ has the same meaning, or is logically equivalent, to the formula $(\neg P) \vee Q$. To see why this is true, we need to look at a truth table that evaluates these two formulas (see Figure 6). Our truth table has a row corresponding to each evaluation of propositions P and Q. The column $P \to Q$ evaluates its truth in each case. To compute $(\neg P) \vee Q$ we need to first compute $\neg P$ and then disjunct it with Q. Computing $\neg P$ requires one to flip each value in column P. $\neg P \vee Q$ is based on the columns for $\neg P$ and Q, and taking their disjunction. Observe that since the columns for $P \to Q$ and $(\neg P) \vee Q$ evaluate to the same value in each row of the truth table, we can conclude that they are logically equivalent.

The contrapositive of an implication $P \to Q$ was defined to be $(\neg Q) \to (\neg P)$ (see Definition 3). The significance of this definition arises from the fact that $P \to Q \equiv (\neg Q) \to (\neg Q)$. This is shown in Example 6

Example 6. $P \to Q \equiv (\neg Q) \to (\neg P)$. This can be seen through the truth table in Figure 7, where we compute $\neg P$, $\neg Q$, and using these columns compute $P \to Q$ and $(\neg Q) \to (\neg P)$.

P	Q	$\neg P$	$\neg Q$	$P \to Q$	$(\neg Q) \to (\neg P)$
F	F	T	T	T	T
F	T	T	F	T	T
T	F	F	T	F	F
T	T	F	F	T	T

Figure 7: Truth table for $P \to Q$ and $(\neg Q) \to (\neg P)$

The equivalence shown in Example 5 and Example 6 are two examples of logical equivalences that are often used in mathematical reasoning. We list some of the other equivalences that are commonly used.

Commonly used Equivalences

Equivalences between formulas are quite often used in simplifying mathematical statements, and in writing proofs. We list some commonly used equivalences. Constructing truth tables to demonstrate these equivalences, is left for the reader as an exercise. In each of the equivalences below P, Q, R are any propositions.

1. **Double Negation:** $\neg\neg P \equiv P$
2. The following two equivalences hold
$$P \vee (\neg P) \equiv \mathsf{T} \qquad P \wedge (\neg P) \equiv \mathsf{F}$$
3. **Identity:** The following hold.
$$P \vee \mathsf{F} \equiv P \qquad P \wedge \mathsf{T} \equiv P$$
4. **Commutativity:** The logical operators $\vee$ and $\wedge$ are *commutative*. That is,
$$P \vee Q \equiv Q \vee P \qquad P \wedge Q \equiv Q \wedge P$$
5. **Idempotence:** The logical operators $\wedge$ and $\vee$ are *idempotent*. That is,
$$P \vee P \equiv P \qquad P \wedge P \equiv P$$
6. **Associativity:** The logical operators $\vee$ and $\wedge$ are *associative*. That is,
$$P \vee (Q \vee R) \equiv (P \vee Q) \vee R \qquad P \wedge (Q \wedge R) \equiv (P \wedge Q) \wedge R$$
7. **Distributivity:** The logical operators $\vee$ and $\wedge$ *distribute* over each other.
$$P \vee (Q \wedge R) \equiv (P \vee Q) \wedge (P \vee R)$$
$$P \wedge (Q \vee R) \equiv (P \wedge Q) \vee (P \wedge R)$$

8. **de Morgan's Laws:** These are two of the most important equivalences that relate negation, conjunction, and disjunction.
$$\neg(P \vee Q) \equiv (\neg P) \wedge (\neg Q)$$
$$\neg(P \wedge Q) \equiv (\neg P) \vee (\neg Q)$$
Combining this with the law of double negation, de Morgan's Laws can be seen as a way to define conjunction in terms of disjunction and negation (or disjunction using conjunction and negation). Thus, we could say,
$$P \vee Q \equiv \neg((\neg P) \wedge (\neg Q))$$
$$P \wedge Q \equiv \neg((\neg P) \vee (\neg Q))$$

9. **Implication:** Implication can be defined in terms of disjunction and negation. That is, $P \rightarrow Q \equiv (\neg P) \vee Q$. Thus, using de Morgan's laws and the law of double negation, we could say
$$\neg(P \rightarrow Q) \equiv P \wedge (\neg Q).$$

10. **Contrapositive:** The contrapositive of an implication is equivalent to the implication. That is $P \rightarrow Q \equiv (\neg Q) \rightarrow (\neg P)$.

The Logical Equivalence Problem and its central role in computer science

Logical equivalence plays an extremely important role in mathematics and in practical algorithms used in computer science. There are two special forms of logical equivalence (or inequivalence), namely validity and satisfiability, that play a central role in computer science. A formula/proposition P is said to be *valid* if $P \equiv \mathsf{T}$, i.e., P evaluates to T in every row of the truth table. On the other hand, a formula/proposition P is said to be *satisfiable* if $P \not\equiv \mathsf{F}$, i.e., P evaluates to T in at least one row of the truth table.

Algorithms for many computational tasks in different areas of computer science essentially reduce to constructing a special logical formula and checking if it is satisfiable. Examples include reasoning in AI systems, path planning for robots and autonomous cars/vehicles, circuit design and synthesis, and verfication/testing of circuits and programs. Due to this reason, significant energies have been devoted to developing heuristics and tools to solve the logical equivalence (or more specifically, the satisfiability) problem. Satisfiability and equivalence engines are one of the main economic drivers in the Electronic Design and Automation (EDA) industry.

What is the best way to solve logical inequivalence? That is, given two formulas P and Q, how can we determine if P and Q are inequivalent (or mean different things)? We have seen one approach to solving this problem — build a truth table and see if P and Q have different answers in some row of the truth table! If P and Q depend on n propositional variables, the truth table will have 2^n rows. Note that to convince someone that P and Q are not equivalent, all we need to do is to share a row where P and Q differ. Is there an algorithm that will discover this distinguishing row *without* building all the 2^n rows of the truth table? There are special cases of P and Q for which this can be done. But is there an algorithm that avoids building the entire truth table *no matter* which two formulas P and Q are given?

This has remained a stubbornly difficult problem to answer since the 1970s. In fact, it is among the most important scientific questions known to mankind, and has a million dollar prize from the Clay Mathematics Institute associated with it (`https://www.claymath.org/millennium-problems/p-vs-np-problem`). This is sometimes called the P *versus* NP question.

Predicate Logic

PROPOSITIONS IN MATHEMATICS OFTEN ARISE THROUGH PREDICATES. A *predicate* is a proposition that depends on the value of some variables. For example, the statement "x is prime" will either be true or false depending on the value of x. Taking the standard interpretation for a prime number, when $x = 2$, the statement "x is prime" is true; and when $x = 4$ the statement "x is prime" is not true. Predicates may depend on more than one variable. For example, the predicate $Q(x, y) \triangleq x + y = 0$ depends on the values of both x and y.

One way to convert a predicate into a proposition, is by substituting values for predicate variables. For example, taking Q to be the predicate defined in the previous paragraph, $Q(2, -2)$ is the proposition "$2 + (-2) = 0$". Another way propositions in mathematics are obtained from predicates, is through quantification. There are two ways predicates can be quantified.

Universal Quantification. It is written like an upside down A. For a predicate P, the proposition $\forall x\ P(x)$, is read as "for all x, $P(x)$". This proposition is true, if the predicate P is true not matter what value is substituted for x. Based on this interpretation, one can observe a close correspondence between universal quantification and conjunction. A proposition $\forall x \in \mathbb{N}\ P(x)$ is equivalent to saying that $P(0)$ is true, and $P(1)$ is true, and $P(2)$ is true, and so on.

Existential Quantification. The second form of quantification is *existential quantification* which is written like a backwards E — $\exists x\ P(x)$ — and read "exists x $P(x)$". The proposition $\exists x\ P(x)$ is true if there is some value we can assign to x such that th predicate P is true. Thus, $\exists x \in \mathbb{N}\ P(x)$ is effectively saying that either $P(0)$ is true or $P(1)$ is true or $P(2)$ is true and so on.

The correspondence between universal quantification and conjunction, and existential quantification and disjunction, allows one to infer logical equivalences like de Morgan's laws for quantification.

$$\neg(\forall x\ P(x)) \equiv \exists x\ (\neg P(x))$$
$$\neg(\exists x\ P(x)) \equiv \forall x\ (\neg P(x))$$

Domains of Discourse. Recall that a formula $\forall x\ P(x)$ says that the predicate P is true no matter what value is substituted for x. But where are the values drawn from? This given by the *domain of discourse*, which identifies the set over which predicate variables take

values and the meaning of predicates. It is given explicitly in a statement like $\forall x \in \mathbb{Z}(\exists y \in \mathbb{Z}\ (x + y = 0))$. Here variables x and y take values over the integers ($\mathbb{Z}$). It also tells us that $+$ should be interpreted the way addition is usually defined over the integers. In some cases, the domain of discourse is omitted from the statement. It is implicit in such cases, and will be clear from the context. The domain of discourse plays an important role in determining whether a formula is true or not. For example the formula $\forall x \in \mathbb{Z}(\exists y \in \mathbb{Z}\ (x + y = 0))$ is true, because for any integer x, if we take $y = -x$, which is also an integer, then $x + y$ is 0. On the other hand, if we change the domain of discourse to be the set of natural numbers $\mathbb{N}$, then the same formula $\forall x \in \mathbb{N}(\exists y \in \mathbb{N}\ (x + y = 0))$ is no longer true. For example, if $x = 1$, then there is no natural number y you can add to x to get 0!

Order of Quantifiers. It is important to pay attention to the order of quantifiers. Changing the order can completely change the meaning. For example, the formula

$$\forall x \in \mathbb{Z}(\exists y \in \mathbb{Z}\ (x + y = 0))$$

is true, because we can take y to be $-x$. On the other hand,

$$\exists y \in \mathbb{Z}(\forall x \in \mathbb{Z}\ (x + y = 0))$$

not true — there is no number with the property that no matter what we add to it, we get 0!