St. Francis Xavier University
Department of Computer Science

CSCI 544: Computational Logic
Lecture 2: Propositional Logic I—Syntax and Semantics
Winter 2023

1 The Syntax of Propositional Logic

Propositional logic is the logical system that is concerned primarily with propositions, formulas, and the logical connectives that join them. In a sense, it's the simplest logical system with which we can still do interesting things, and so it will form the basis of our studies in the first part of this course.

Before we get into the fine details of propositional logic, we must first define the symbols and language we'll be using. We touched upon these notions in the introductory lecture, but for completeness we'll review everything here.

The atomic unit that we'll be using is a *proposition*: some kind of statement with a truth value. Propositions will typically be denoted by lowercase letters such as p, q, r, and so on. Occasionally, we will denote a set of propositions by the symbol $\mathscr{P}$.

Since we can only do so much with propositions alone, we require some way of joining propositions to make even more interesting and complex statements. To this end, we use *logical connectives*. For our purposes, logical connectives can be either *unary* or *binary*, meaning they can be applied to either one or two propositions, respectively.

There are a total of $2^2 = 4$ unary logical connectives, which we summarize in the following *truth table*:

p	op_1	op_2	op_3	op_4
	1	p	$\neg$	0
T	T	T	F	F
F	T	F	T	F

Of these four logical connectives, the only one with any utility is the *negation* connective, $\neg$. If we apply the negation operator to a proposition p, then $\neg p$ takes on the opposite truth value to p.

Let's now turn to binary logical connectives, of which there are $2^{2^2} = 16$. Each binary logical connective is summarized in the following truth table:

p	q	op_1	op_2	op_3	op_4	op_5	op_6	op_7	op_8
		1	$\vee$	$\Leftarrow$	p	$\Rightarrow$	q	$\Leftrightarrow$	$\wedge$
T	T	T	T	T	T	T	T	T	T
T	F	T	T	T	T	F	F	F	F
F	T	T	T	F	F	T	T	F	F
F	F	T	F	T	F	T	F	T	F

p	q	op_9	op_{10}	op_{11}	op_{12}	op_{13}	op_{14}	op_{15}	op_{16}
		NAND	XOR	$\neg q$	$\not\Leftarrow$	$\neg p$	$\not\Rightarrow$	NOR	0
T	T	F	F	F	F	F	F	F	F
T	F	T	T	T	T	F	F	F	F
F	T	T	T	F	F	T	T	F	F
F	F	T	F	T	F	T	F	T	F

As before, there are only a few binary logical connectives with enough utility that we will use and reuse them throughout this course. Those logical connectives are *disjunction* ($\vee$), *conjunction* ($\wedge$), *implication* ($\Rightarrow$), and *equivalence* ($\Leftrightarrow$).

A few binary logical connectives are of secondary importance in formal logic, but are very common in related areas such as electrical engineering and computer engineering. These logical connectives include NAND, NOR, and XOR, which are most commonly found and used in logic gates within electric circuits. The main reason why we don't emphasize these logical connectives here is because we can define each connective in terms of our five "main" connectives:

- p NAND q can be written as $p \Rightarrow \neg q$;
- p NOR q can be written as $\neg p \wedge \neg q$; and
- p XOR q can be written as either $p \Leftrightarrow \neg q$ or $\neg p \Leftrightarrow q$.

Taking propositions together with logical connectives, we can now define the star of propositional logic: *formulas.* Our definition will make use of recursion by defining formulas first in terms of atomic propositions (i.e., a base case) and then in terms of other formulas.

Definition 1 (Formula)**.** A statement in propositional logic is a formula if it can be constructed according to the following rules:

1. Every atomic proposition is a formula;
2. If p is a formula, then $\neg p$ is a formula;
3. If p and q are formulas, then each of $p \wedge q$, $p \vee q$, $p \Rightarrow q$, and $p \Leftrightarrow q$ is a formula; and
4. No other statement is a formula.

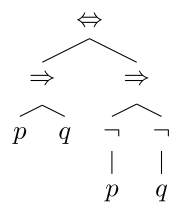
Like propositions, we will occasionally denote a set of formulas by the symbol $\mathscr{F}$.

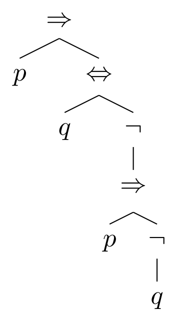
If you've taken a course in theoretical computer science, you may recall that words (or strings) can be represented either as a linear sequence of symbols or as a parse tree. Indeed, the same can be said about formulas: we can write formulas either linearly or using a tree structure.

Example 2. Consider the following formula:

$$p \Rightarrow q \Leftrightarrow \neg p \Rightarrow \neg q.$$

This formula can be obtained by performing an inorder traversal on either of the following trees:





As you might have noticed (or recalled, from a discussion of parse trees in a past theory course), our example has a problem. Our single formula has two different tree representations! Fortunately, we can resolve this ambiguity in a couple of ways: either by using parentheses or by introducing a logical connective *precedence hierarchy.*

The simplest way to remove ambiguity—although not the most aesthetically pleasing way—is to insert parentheses into our formula where necessary. Following an inorder traversal of a tree, we would place parentheses around the subformula produced by each subtree. Thus, the left tree would produce the formula

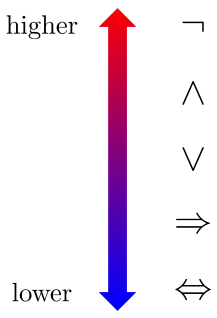
$$(p \Rightarrow q) \Leftrightarrow ((\neg p) \Rightarrow (\neg q))$$

while the right tree would produce the formula

$$p \Rightarrow (q \Leftrightarrow \neg(p \Rightarrow (\neg q))).$$

Thus, the ambiguity is removed: the first formula reads "(p implies q) if and only if (not-p implies not-q)", while the second formula reads "p implies (q if and only if not-(p implies not-q))". These two formulas can be shown to be distinct by writing out their truth tables.

Alternatively, we can remove ambiguity from our formulas by introducing a precedence hierarchy. Just like the grade school BEDMAS hierarchy for arithmetic operations, a precedence hierarchy for logical connectives specifies which connectives are to be applied to their operands in which order. In propositional logic, the precedence hierarchy we will follow is



Thus, according to this precedence hierarchy, our formula $p \Rightarrow q \Leftrightarrow \neg p \Rightarrow \neg q$ would be parsed as

$$(p \Rightarrow q) \Leftrightarrow ((\neg p) \Rightarrow (\neg q)),$$

which is also the formula corresponding to the left tree in the example.

1.1 Interlude: Structural Induction

One of the common proof techniques we will use in this course is *structural induction*, a form of induction that is particularly applicable to mathematical structures like trees. In a proof by structural induction, we replace the set of natural numbers used in a standard induction proof with some mathematical structure that can be recursively defined; that is, a structure made up of smaller substructures that share the same properties as the structure itself. (For example, trees are made up of subtrees, and a subtree is itself a tree so it shares the same properties.)

Since we defined formulas recursively, and since we can represent formulas as trees, structural induction will lend itself quite well to proving properties of formulas.

Proposition 3 (Principle of structural induction—formulas)**.** *Let $\mathscr{F}$ be a set of formulas. To show that some property P holds for all formulas $A \in \mathscr{F}$, we must prove each of the following:*

1. *P holds for all atomic propositions p;*
2. *Assuming P holds for a formula A, P also holds for the formula $\neg A$; and*
3. *Assuming P holds for formulas A_1 and A_2, P also holds for the formula $A_1 \, op \, A_2$, where op is a binary logical connective.*

2 The Semantics of Propositional Logic

Having established the syntax of propositional logic, we can move on to the more in-depth matter of discussing semantics. Put simply, semantics focuses on assigning truth values to propositions and evaluating truth values of formulas. In order to do either of these things, we must build ourselves a bit of logical machinery.

In order to assign truth values to propositions, we must first define something called an *interpretation*, which is really nothing more than a function from propositions to truth values.

Definition 4 (Interpretation for a formula)**.** Let $A \in \mathscr{F}$ be a formula, and let $\mathscr{P}_A$ be the set of atomic propositions appearing in A. An interpretation for A is a total function

$$\mathscr{I}_A \colon \mathscr{P}_A \to \{\mathrm{T}, \mathrm{F}\}.$$

Using an interpretation, we can assign truth values to propositions in the following way.

Definition 5 (Truth value of a formula)**.** Let $\mathscr{I}$ be an interpretation for a formula $A \in \mathscr{F}$ (where A may be composed of smaller formulas A_1 and A_2). The truth value of A under the interpretation $\mathscr{I}$, denoted $v_{\mathscr{I}}(A)$, is defined inductively according to the following rules:

1.	$v_{\mathscr{I}}(A) = \mathscr{I}(A)$	if A is an atomic proposition;
2.	$v_{\mathscr{I}}(\neg A) = \mathrm{T}$	if $v_{\mathscr{I}}(A) = \mathrm{F}$;
3.	$v_{\mathscr{I}}(\neg A) = \mathrm{F}$	if $v_{\mathscr{I}}(A) = \mathrm{T}$;
4.	$v_{\mathscr{I}}(A_1 \wedge A_2) = \mathrm{T}$	if $v_{\mathscr{I}}(A_1) = \mathrm{T}$ and $v_{\mathscr{I}}(A_2) = \mathrm{T}$;
5.	$v_{\mathscr{I}}(A_1 \wedge A_2) = \mathrm{F}$	otherwise;
6.	$v_{\mathscr{I}}(A_1 \vee A_2) = \mathrm{F}$	if $v_{\mathscr{I}}(A_1) = \mathrm{F}$ and $v_{\mathscr{I}}(A_2) = \mathrm{F}$;
7.	$v_{\mathscr{I}}(A_1 \vee A_2) = \mathrm{T}$	otherwise;
8.	$v_{\mathscr{I}}(A_1 \Rightarrow A_2) = \mathrm{F}$	if $v_{\mathscr{I}}(A_1) = \mathrm{T}$ and $v_{\mathscr{I}}(A_2) = \mathrm{F}$;
9.	$v_{\mathscr{I}}(A_1 \Rightarrow A_2) = \mathrm{T}$	otherwise;
10.	$v_{\mathscr{I}}(A_1 \Leftrightarrow A_2) = \mathrm{T}$	if $v_{\mathscr{I}}(A_1) = v_{\mathscr{I}}(A_2)$; and
11.	$v_{\mathscr{I}}(A_1 \Leftrightarrow A_2) = \mathrm{F}$	otherwise.

The truth values in each case essentially formalize the same truth values that appeared in our earlier truth tables. Indeed, we can even formally define the notion of a truth table using interpretations and truth values.

Definition 6 (Truth table)**.** Let $A \in \mathscr{F}$ be a formula, let $\mathscr{I}$ be an interpretation for A, and suppose there are n atomic propositions in the set $\mathscr{P}_A$. A truth table is a table with 2^n rows and $n+1$ columns where the first n columns specify the interpretation $\mathscr{I} \colon \mathscr{P}_A \to \{\mathrm{T}, \mathrm{F}\}$ and the last column specifies $v_{\mathscr{I}}(A)$.

Of course, we can generalize the notion of a truth table to include additional columns containing the truth values of subformulas.

Example 7. Recall our earlier example formula, $(p \Rightarrow q) \Leftrightarrow ((\neg p) \Rightarrow (\neg q))$. Suppose that we have an interpretation $\mathscr{I}$ where $\mathscr{I}(p) = \mathrm{F}$ and $\mathscr{I}(q) = \mathrm{T}$. Following the rules in Definition 5, we can evaluate the truth value of the overall formula inductively.

- $v_{\mathscr{I}}(p) = \mathrm{F}$ by rule 1, since $\mathscr{I}(p) = \mathrm{F}$;
- $v_{\mathscr{I}}(q) = \mathrm{T}$ by rule 1, since $\mathscr{I}(q) = \mathrm{T}$;
- $v_{\mathscr{I}}(p \Rightarrow q) = \mathrm{T}$ by rule 9;
- $v_{\mathscr{I}}(\neg p) = \mathrm{T}$ by rule 2;
- $v_{\mathscr{I}}(\neg q) = \mathrm{F}$ by rule 3;
- $v_{\mathscr{I}}((\neg p) \Rightarrow (\neg q)) = \mathrm{F}$ by rule 8; and
- $v_{\mathscr{I}}((p \Rightarrow q) \Leftrightarrow ((\neg p) \Rightarrow (\neg q))) = \mathrm{F}$ by rule 11.

Example 8. Using again our familiar formula $(p \Rightarrow q) \Leftrightarrow ((\neg p) \Rightarrow (\neg q))$, let's construct a truth table for the formula. Note that our truth table contains intermediate columns for each subformula, though the first n columns still specify each atomic proposition and the last column still specifies the formula itself.

The row corresponding to our interpretation in the previous example is highlighted for reference.

p	q	$p \Rightarrow q$	$\neg p$	$\neg q$	$(\neg p) \Rightarrow (\neg q)$	$(p \Rightarrow q) \Leftrightarrow ((\neg p) \Rightarrow (\neg q))$
T	T	T	F	F	T	T
T	F	F	F	T	T	F
F	T	T	T	F	F	F
F	F	T	T	T	T	T

Lastly, just as we defined an interpretation for a single formula, so too can we define an interpretation for a set of formulas.

Definition 9 (Interpretation for a set of formulas)**.** Let $S = \{A_1, A_2, \dots\}$ be a set of formulas where $S \subseteq \mathscr{F}$, and let $\mathscr{P}_S = \bigcup_{i \in \mathbb{N}} \mathscr{P}_A$ be the set of atomic propositions that appear in any of the formulas of S. An interpretation for S is a total function

$$\mathscr{I}_S \colon \mathscr{P}_S \to \{\mathrm{T}, \mathrm{F}\}.$$

The truth value of each formula A_i is defined in exactly the same way as in Definition 5.

Example 10. Suppose we have a set of three formulas $S = \{p \vee \neg q, q, p \wedge r \Leftrightarrow (r \Rightarrow q)\}$ and, additionally, we are working under an interpretation $\mathscr{I}$ where $v_{\mathscr{I}}(p) = \mathrm{F}$, $v_{\mathscr{I}}(q) = \mathrm{T}$, and $v_{\mathscr{I}}(r) = \mathrm{T}$.

Under this interpretation, we can conclude that $v_{\mathscr{I}}(p \vee \neg q) = \mathrm{F}$, $v_{\mathscr{I}}(q) = \mathrm{T}$, and $v_{\mathscr{I}}(p \wedge r \Leftrightarrow (r \Rightarrow q)) = \mathrm{F}$.

2.1 Logical Equivalence

Recall that one of the main binary logical connectives we defined earlier was that of equivalence ($\Leftrightarrow$). According to its definition, $p \Leftrightarrow q$ is true whenever p and q share the same truth value in whatever interpretation we're using.

However, there exists a stronger notion of equivalence, which we will call *logical equivalence*. With logical equivalence, we can show that two formulas are equivalent regardless of the interpretation we're using.

Definition 11 (Logical equivalence)**.** Let $A_1, A_2 \in \mathscr{F}$ be formulas. If $v_{\mathscr{I}}(A_1) = v_{\mathscr{I}}(A_2)$ for all interpretations $\mathscr{I}$, then we say that A_1 and A_2 are logically equivalent, and we denote this by $A_1 \equiv A_2$.

Example 12. Consider the formulas $A_1 = p \Rightarrow q$ and $A_2 = \neg p \vee q$. Are these two formulas logically equivalent? To check, let's write a truth table for both formulas:

p	q	$p \Rightarrow q$	$\neg p$	$\neg p \vee q$
T	T	T	F	T
T	F	F	F	F
F	T	T	T	T
F	F	T	T	T

As we can see, the column corresponding to A_1 exactly matches the column corresponding to A_2 for all possible interpretations. Thus, the two formulas are logically equivalent.

Note that logical equivalence and the equivalence logical connective are distinct: logical equivalence is a property of two formulas (i.e., semantic), while the equivalence logical connective is a symbol that appears in formulas (i.e., syntactic). However, since the truth values of two logically equivalent formulas match for every possible interpretation, and since the equivalence connective is true whenever the truth values of its operands match, it seems prudent for us to prove a clear relationship between logical equivalence and the equivalence connective.

Theorem 13. *For any two formulas $A_1, A_2 \in \mathscr{F}$, $A_1 \equiv A_2$ if and only if $A_1 \Leftrightarrow A_2$ is true in every possible interpretation.*

Proof. (⇒): Suppose $A_1 \equiv A_2$, and let $\mathscr{I}$ be an arbitrary interpretation. Then, by the definition of logical equivalence, $v_{\mathscr{I}}(A_1) = v_{\mathscr{I}}(A_2)$. By rule 10 of Definition 5, we then have that $v_{\mathscr{I}}(A_1 \Leftrightarrow A_2) = \text{T}$.

(⇐): Suppose $A_1 \Leftrightarrow A_2$ is true in every possible interpretation $\mathscr{I}$. Then we have that $v_{\mathscr{I}}(A_1) = v_{\mathscr{I}}(A_2)$ for all $\mathscr{I}$, which is the definition of logical equivalence. □

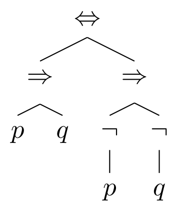
2.2 Substitution

At times, we may be given a formula that is rather complex or difficult to parse, and it may be our job to determine the truth value of that formula. Fortunately, using the notion of logical equivalence, we can make *substitutions* in a formula in order to simplify it drastically.

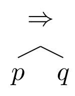
Often, we aren't so lucky that we can substitute the entire formula for something simpler all in one step. Thus, we often need to decompose a formula into *subformulas* and make substitutions for certain subformulas. Using our tree representation of a formula, we can define the notion of a subformula quite naturally.

Definition 14 (Subformula)**.** Given a formula A, a formula S is a subformula of A if, in the tree representation, S is a subtree of A.

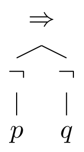
Example 15. Recall our earlier formula, $(p \Rightarrow q) \Leftrightarrow ((\neg p) \Rightarrow (\neg q))$, and its associated tree:



In this formula, we find a number of subformulas; for example,



and



corresponding to the subformulas $p \Rightarrow q$ and $\neg p \Rightarrow \neg q$, respectively.

If we know of a logically-equivalent way to express a particular subformula, then we can perform a substitution for that subformula.

Definition 16 (Substitution of a subformula)**.** Given a formula A having a subformula B, let B' be any formula logically equivalent to B. The substitution of B' for B in A, denoted $A\{B \leftarrow B'\}$, is the formula obtained by replacing all occurrences of the subformula B in A with B'.

It is rather important to note (and prove) that performing a substitution of one logically-equivalent subformula for another does not affect the overall truth value of the formula.

Theorem 17. *Let A be a formula having a subformula B, and let B' be a formula such that $B' \equiv B$. Then $A \equiv A\{B \leftarrow B'\}$.*

Proof. Consider an arbitrary interpretation $\mathscr{I}$. We know that $v_{\mathscr{I}}(B) = v_{\mathscr{I}}(B')$, and we want to show that $v_{\mathscr{I}}(A) = v_{\mathscr{I}}(A\{B \leftarrow B'\})$.

We will prove the statement by induction on the depth d of the highest occurrence of the subtree corresponding to B in the tree representation of A.

- If $d = 0$, then the subtree corresponding to B is the same as the tree corresponding to A itself. Thus, we have that $v_{\mathscr{I}}(B) = v_{\mathscr{I}}(A) = v_{\mathscr{I}}(A\{B \leftarrow B'\}) = v_{\mathscr{I}}(B')$ as required.

- If $d > 0$, then we have that either $A = \neg A_1$ or $A = A_1 \operatorname{op} A_2$ for some subformulas A_1 and A_2 and logical connective op. Without loss of generality, suppose that $B = A_1$. Then the depth of B in the tree representation of A is less than d, so by the inductive hypothesis, we have that $v_{\mathscr{I}}(A_1) = v_{\mathscr{I}}(A_1\{B \leftarrow B'\})$. Then, by the definition of v on logical connectives, we have that $v_{\mathscr{I}}(A) = v_{\mathscr{I}}(A\{B \leftarrow B'\})$ as required. $\square$

Example 18. Recall once again the formula $A = (p \Rightarrow q) \Leftrightarrow ((\neg p) \Rightarrow (\neg q))$, and let $B = p \Rightarrow q$.

Observe that $B' = \neg p \vee q$ is logically equivalent to the subformula B. (You can verify this by drawing the truth tables of both formulas, for instance.) Then, performing a substitution of B' for B in A, we get the formula

$$A\{B \leftarrow B'\} = (\neg p \vee q) \Leftrightarrow ((\neg p) \Rightarrow (\neg q)).$$

There are a number of common substitutions that we can make in order to simplify a given formula, and each of these substitutions fall into one of a handful of categories. In the following tables, we cover all of these common substitutions.[1]

Conjunction	**Negation**
$A \wedge A \equiv A$	$\neg\neg A \equiv A$
$A \wedge \neg A \equiv \mathrm{F}$	**Implication**
Disjunction	$A \Rightarrow A \equiv \mathrm{T}$
$A \vee A \equiv A$	**Equivalence**
$A \vee \neg A \equiv \mathrm{T}$	$A \Leftrightarrow A \equiv \mathrm{T}$

Table 1: Identities

Conjunction	**Implication**
$A \wedge \mathrm{T} \equiv A$	$A \Rightarrow \mathrm{T} \equiv A$
$A \wedge \mathrm{F} \equiv \mathrm{F}$	$A \Rightarrow \mathrm{F} \equiv \neg A$
Disjunction	$\mathrm{T} \Rightarrow A \equiv A$
$A \vee \mathrm{T} \equiv \mathrm{T}$	$\mathrm{F} \Rightarrow A \equiv \mathrm{T}$
$A \vee \mathrm{F} \equiv A$	**Equivalence**
	$A \Leftrightarrow \mathrm{T} \equiv A$
	$A \Leftrightarrow \mathrm{F} \equiv \neg A$

Table 2: Absorption of constants

$$\begin{gathered} A \wedge B \equiv B \wedge A \\ A \vee B \equiv B \vee A \\ A \Leftrightarrow B \equiv B \Leftrightarrow A \end{gathered}$$

Table 3: Commutativity

$$\begin{gathered} A \wedge (B \wedge C) \equiv (A \wedge B) \wedge C \\ A \vee (B \vee C) \equiv (A \vee B) \vee C) \\ A \Leftrightarrow (B \Leftrightarrow C) \equiv (A \Leftrightarrow B) \Leftrightarrow C \end{gathered}$$

Table 4: Associativity

$$\begin{gathered} A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C) \\ A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C) \end{gathered}$$

Table 5: Distributivity

$$\begin{gathered} A \wedge B \equiv \neg(\neg A \vee \neg B) \\ A \vee B \equiv \neg(\neg A \wedge \neg B) \end{gathered}$$

Table 6: De Morgan's Laws

$$\begin{gathered} A \Rightarrow B \equiv \neg(A \wedge \neg B) \\ A \Rightarrow B \equiv \neg A \vee B \\ A \Leftrightarrow B \equiv (A \Rightarrow B) \wedge (B \Rightarrow A) \end{gathered}$$

Table 7: Removal of $\Rightarrow$ and $\Leftrightarrow$

2.3 Satisfiability and Validity

To continue our discussion of semantics, we will go beyond simple truth values and now consider some properties of formulas and their truth values. The semantical notions we are about to define will guide much of our study of formal logic for the remainder of the course.

One of the fundamental properties we often want to check for a given formula is whether there exists some interpretation that assigns to the formula a truth value of true. In this way, we can verify that a formula is not a contradiction; that is, it's possible in at least one case for the formula to be true. If this is the case, we say that the formula is *satisfiable*.

Definition 19 (Satisfiability)**.** Let A be a formula. Then A is satisfiable if and only if $v_{\mathscr{I}}(A) = \mathrm{T}$ for some interpretation $\mathscr{I}$.

[1] Note that you don't need to memorize these substitutions! Think of these tables as a reference sheet.

Furthermore, if some interpretation $\mathscr{I}$ is a satisfying interpretation for A, then we say that $\mathscr{I}$ is a *model* for A. Naturally, if there does not exist any interpretation where A is true—that is, if $v_{\mathscr{I}}(A) = \text{F}$ for all interpretations $\mathscr{I}$—then we say that A is *unsatisfiable*.

Remark. You may already be familiar with the notion of satisfiability from a past course on theory of computing. Indeed, we're talking about the exact same problem here: given a formula, we want to determine whether there exists some assignment of truth values to each variable of the formula such that the overall formula evaluates to true.

Just as we did with truth values, we can generalize the notion of satisfiability from a single formula to a set of formulas.

Definition 20 (Satisfiability for a set of formulas)**.** Let $U = \{A_1, A_2, \dots\}$ be a set of formulas. Then U is simultaneously satisfiable, or just satisfiable, if and only if there exists an interpretation $\mathscr{I}$ where $v_{\mathscr{I}}(A_i) = \text{T}$ for all i.

The definitions of a model and of unsatisfiability can similarly be generalized to sets of formulas.

Going one step further, if we're able to show that a formula is satisfiable regardless of the interpretation we use, then we say the formula is *valid*. Validity is a particularly useful property to establish, as it removes the need for us to focus on which interpretation to use.

Definition 21 (Validity)**.** Let A be a formula. Then A is valid if and only if $v_{\mathscr{I}}(A) = \text{T}$ for all interpretations $\mathscr{I}$, and we denote the validity of A by $\models A$.

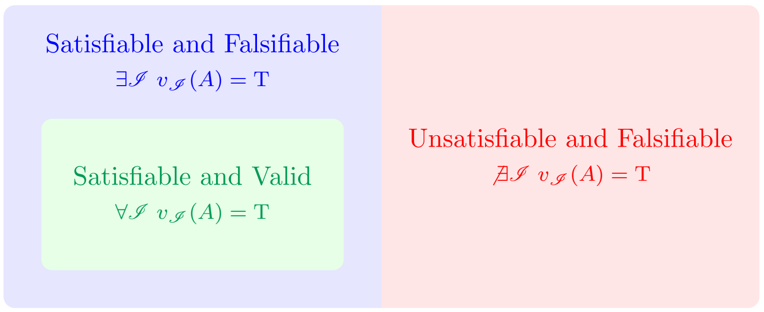
A valid formula is sometimes referred to as a *tautology*, since it is true regardless of the interpretation we use. If there exists some interpretation where A is false—that is, if $v_{\mathscr{I}}(A) = \text{F}$ for some interpretation $\mathscr{I}$—then we say that A is *falsifiable* and we denote this by the notation $\not\models A$.

Remark. Note that writing $\not\models A$ is *not* the same as writing $\models \neg A$.

Example 22. Let's consider a selection of different formulas and evaluate both their satisfiability and validity.

- Let $A_1 = (p \Rightarrow q) \Rightarrow (\neg q \Rightarrow \neg p)$. This formula is both satisfiable and valid. In the three possible cases where $(p \Rightarrow q)$ evaluates to true, $(\neg q \Rightarrow \neg p)$ will also evaluate to true, and the overall expression will therefore also be true. In the one case where $(p \Rightarrow q)$ evaluates to false, $(\neg q \Rightarrow \neg p)$ will also evaluate to false, and the overall expression will be true.
- Let $A_2 = q \Rightarrow (q \Rightarrow p)$. This formula is satisfiable, for instance, if we use the interpretation $v_{\mathscr{I}}(p) = v_{\mathscr{I}}(q) = \text{T}$. However, the formula is not valid, since it evaluates to false using the interpretation $v_{\mathscr{I}}(p) = \text{F}, v_{\mathscr{I}}(q) = \text{T}$.
- Let $A_3 = (p \wedge \neg p) \vee (q \wedge \neg q)$. This formula is neither satisfiable nor valid, since both subformulas $(p \wedge \neg p)$ and $(q \wedge \neg q)$ will evaluate to false regardless of the interpretation we use.

It may be difficult at first to separate these four notions in your head at once. Thus, we summarize the relations between satisfiability, unsatisfiability, validity, and falsifiability in the following diagram:


Satisfiable and Falsifiable
$\exists \mathscr{I}\ v_{\mathscr{I}}(A) = \text{T}$
Satisfiable and Valid
$\forall \mathscr{I}\ v_{\mathscr{I}}(A) = \text{T}$
Unsatisfiable and Falsifiable
$\nexists \mathscr{I}\ v_{\mathscr{I}}(A) = \text{T}$

We can additionally relate the four notions to one another via the following two lemmas.

Lemma 23. *Let A be a formula. Then A is satisfiable if and only if $\neg A$ is falsifiable.*

Proof. Suppose A is satisfiable for some interpretation $\mathscr{I}$. Then we have that $v_{\mathscr{I}}(A) = \mathrm{T}$. By Definition 5, we then have that $v_{\mathscr{I}}(\neg A) = \mathrm{F}$ for the same interpretation $\mathscr{I}$. Thus, $\neg A$ is falsifiable. The converse direction is analogous. □

Lemma 24. *Let A be a formula. Then A is valid if and only if $\neg A$ is unsatisfiable.*

Proof. Consider an arbitrary interpretation $\mathscr{I}$. Since A is valid, we have that $v_{\mathscr{I}}(A) = \mathrm{T}$. By Definition 5, we then have that $v_{\mathscr{I}}(\neg A) = \mathrm{F}$. Since $\mathscr{I}$ is arbitrary, A is true in all interpretations if and only if $\neg A$ is false in all interpretations, and so it is unsatisfiable. □

2.4 Logical Consequence

A few sections ago, we defined the notion of logical equivalence and drew a connection between the logical operator of equivalence ($\Leftrightarrow$) and the property of logical equivalence ($\equiv$). Indeed, there exists another notion that corresponds to the logical operator of implication ($\Rightarrow$), and in this section we will introduce that notion.

In some cases, we're able to start with a set of formulas and deduce another formula. Deduction, in this sense, merely means that any interpretation that satisfies our initial set of formulas—that is, any model for our formula set—will also satisfy the new formula, so we can "transfer" that model from our formula set to the new formula and maintain the truth of the formula. If we're able to find a model that works for our formula set and for our new formula, then we say that the new formula is a *logical consequence* of our formula set.

Definition 25 (Logical consequence)**.** Let S be a set of formulas and let A be a formula. We say that A is a logical consequence of S if and only if every model of S is also a model of A, and we denote this logical consequence by $S \vDash A$.

If A is not a logical consequence of S, then we sometimes write $S \nvDash A$.

Remark. Again, note that writing $S \nvDash A$ is *not* the same as writing $S \vDash \neg A$.

Example 26. Let $S = \{p \wedge r, \neg q \vee (p \wedge \neg p)\}$ and let $A = (p \wedge \neg q) \Rightarrow r$.

If some interpretation $\mathscr{I}$ is a model for S—in other words, if $\mathscr{I}$ satisfies S—then it must be the case that $v_{\mathscr{I}}(p) = \mathrm{T}$, $v_{\mathscr{I}}(q) = \mathrm{F}$, and $v_{\mathscr{I}}(r) = \mathrm{T}$. However, using that same interpretation, we can show that $v_{\mathscr{I}}((p \wedge \neg q) \Rightarrow r) = \mathrm{T}$. Thus, $S \vDash A$; that is, A is a logical consequence of S.

Like before, the logical operator of implication ($\Rightarrow$) and the notion of logical consequence ($\vDash$) are distinct; the former is syntactic, while the latter is semantic. However, also like before, we can relate the two notions via the following result.

Theorem 27. *Let $S = \{A_1, A_2, \ldots, A_n\}$ be a set of formulas and let A be a formula. Then $S \vDash A$ if and only if $\vDash (A_1 \wedge A_2 \wedge \cdots \wedge A_n) \Rightarrow A$.*

Proof. Note that any model for S can be used to show that A is true. On the other hand, by the definitions of conjunction and implication, any interpretation that renders S false would result in the formula $(A_1 \wedge A_2 \wedge \cdots \wedge A_n) \Rightarrow A$ being true. □

If we take a set of formulas T where, for every formula A, $T \vDash A$ implies $A \in T$, then we say that the set T is closed under logical consequence. Any set of formulas T that is closed under logical consequence is called a *theory*. Theories are quite important in formal logic, as they give us a way to define a set of formulas that we assume to be true (also known as *axioms*) and see what conclusions we can draw from these formulas.