# Propositional Logic and Semantics

English is naturally *ambiguous*. For example, consider the following *employee (non)recommendations* and their *ambiguity* in the English language:

- *"I can assure you that no person would be better for the job."*
- *"All in all, I cannot say enough good things about this candidate or recommend him too highly."*

**Goal:** We want to be able to write *formal boolean expressions* such that there is no *ambiguity*.

For example, $p \to q \to r$ means $(p \to q) \to r$ or $p \to (q \to r)$?

## Propositional Formulas

- Formal expressions involving *conjunctions* and *propositional variables*.
- We denote this *set* by $\mathcal{F}_{\mathcal{PV}}$ or simply $\mathcal{F}$, and define $\mathcal{F}$ *inductively*.

# Slight Diversion - Defining Sets Inductively

## Defining Sets Inductively

What does the following *definition* construct?

Let $X$ be the smallest set such that:

**Basis**: $0 \in X$

**Inductive Step:** if $x \in X$ then $x + 1 \in X$.

**Q:** How could we define the integers, $\mathbb{Z}$?

Let $\mathbb{Z}$ be the smallest set containing:

**Basis:**

**Inductive Step:**

**Q:** How abou the *rationals*, $\mathbb{Q}$?

**Basis:**

**Inductive Step:**

1.

2.

3.

**Q:** How abou the *language of arithmetic*, $\mathcal{L}A$?

Let $\mathcal{LA}$ be the smallest set such that:

**Basis**: $\mathbb{Q} \in \mathcal{LA}$

**Inductive Step:** Suppose that $x, y \in \mathcal{LA}$ then

1.

2.

3.

4.

## Why define sets by induction?

Consider the following conjecture:

Let $e$ be an element of $\mathcal{LA}$.

Let $vr(e)$ represent the number of characters in $e$.

Let $op(e)$ represent the number of *operations*, ie., characters from $\{+, -, *, \div\}$ in $e$.

CLAIM 1: *Let P(e) be "vr(e) = op(e) + 1". Then* $\forall e \in \mathcal{LA}$, $P(e)$*.*

We can *prove* this using a special version of induction called *structural induction.*

CLAIM 1: *Let P(e) be "vr(e) = op(e) + 1". Then* $\forall e \in \mathcal{LA}, P(e)$.

We can *prove* this using a special version of induction called *structural induction.*

*Proof.* STRUCTURAL INDUCTION on $e$:

1. **Basis:** Suppose $e \in \mathbb{Q}$, then

2. **Induction Step:** Assume that $P(e_1)$ and $P(e_2)$ are *true* for arbitrary expressions in $\mathcal{LA}$. Let $e = e_1 \oplus e_2$ where $\oplus \in \{+, -, *, \div\}$.

   Then,

$\square$

$\mathcal{F}_{\mathcal{PV}}$ is the smallest set such that:

**Base Case:**

- **true** and **false** belong to $\mathcal{F}_{\mathcal{PV}}$, and if $p \in PV$ then $p \in \mathcal{F}_{\mathcal{PV}}$.

**Induction Step:** If $p$ and $q \in \mathcal{F}_{\mathcal{PV}}$, then so are

- **NEGATION**: $\neg p$
- **CONJUNCTION**: $(p \wedge q)$
- **DISJUNCTION**: $(p \vee q)$
- **CONDITIONAL**: $(p \rightarrow q)$
- **BICONDITIONAL**: $(p \leftrightarrow q)$

A formula in $\mathcal{F}_{\mathcal{PV}}$ is *uniquely* defined, i.e., there is no *ambiguity*. (see the **Unique Readibility Theorem** in the notes.)

**Q:** What happens when a *propositional formula* is quite *complex*? such as,

$$(((p \wedge y) \vee (q \rightarrow (r \wedge t))) \wedge \neg(s \wedge (u \vee (v \vee (x \vee z)))))$$

This has lead to *conventions* that define an *informal* notation that uses less brackets.

## Bracketing Conventions

1. drop the *outer most parenthesis* e.g,

2. give $\wedge$ and $\vee$ precedence over $\rightarrow$ and $\leftrightarrow$ (like $\times, +$ *vs.* $<, =$ in *arithmetic*) eg.,

3. give $\wedge$ precedence over $\vee$ (similar to $\times$ *vs.* $+$ in *arithmetic*) eg.,

4. *group* from the *right* when the *same connective* appears *consecutively*, eg.,

**Q:** Using these *conventions*, how can

$$(((p \wedge y) \vee (q \rightarrow (r \wedge t))) \wedge \neg(s \wedge (u \vee (v \vee (x \vee z)))))$$

be *simplified*?

## The Meaning of $\tau$

**Q:** What is the difference between a *propositional formula* and a *propositional statement*?

Therefore we need a method to determine the *truth value* of a *statement* from the *truth values* assigned to the *propositional variables*.

- Let $\tau$ be a *truth assignment*, ie., a function.

$$\tau : PV \rightarrow \{\mathbf{true}, \mathbf{false}\}.$$

- If $p \in PV$ and $\tau$ assigns **true** to $p$, then we write

$$\tau(p) = \mathbf{true}.$$

- How does $\tau$ affect a *propositional statement*?

- We need a *function* that behaves like $\tau$, but *operates* on *propositional statements*.

- Let $\tau^* : \mathcal{F}_{\mathcal{PV}} \rightarrow \{\mathbf{true}, \mathbf{false}\}$. What does this mean?

We formally define $\tau^*$ using *structural induction*:

Let $\mathbf{Q}, \mathbf{P} \in \mathcal{F}_{\mathcal{PV}}$.

**Base Case:** $P \in PV$. What is $\tau^*(P)$?

**Inductive Step**

Now we assume that $\mathbf{P}, \mathbf{Q} \in \mathcal{F}_{\mathcal{PV}}$ and that $\tau^*(\mathbf{P})$ and $\tau^*(\mathbf{Q})$ return a value from $\{\mathbf{true}, \mathbf{false}\}$. Then:

$$\tau^*(\neg Q) = \begin{cases} \mathbf{true}, & if \\ \mathbf{false}, & otherwise \end{cases}$$

$$\tau^*(Q \wedge P) = \begin{cases} \mathbf{true}, & if \\ \mathbf{false}, & otherwise \end{cases}$$

$$\tau^*(Q \vee P) = \begin{cases} \mathbf{false}, & if \\ \mathbf{true}, & otherwise \end{cases}$$

## Semantics

- **Satisfies** If $\tau^*(Q)$ = $\mathbf{true}$, then we say that $\tau^*$ *satisfies* $Q$.
- **Falsifies** If $\tau^*(Q) = \mathbf{false}$, then we say that $\tau^*$ *falsifies* $Q$.
- We can determine which *truth assignments* of the propositional variables *satisfy* a particular *propositional statement* using a *truth table*.

# Truth Tables

We will use $\{0,1\}$ to represent $\{\mathbf{true}, \mathbf{false}\}$.

| $p_1$ | $p_2$ | $\neg p_1$ | $\neg p_2$ | $p_1 \wedge p_2$ | $p_1 \vee p_2$ | $p_1 \to p_2$ | $p_1 \leftrightarrow p_2$ |
|---|---|---|---|---|---|---|---|
| 0 | 0 | | | | | | |
| 0 | 1 | | | | | | |
| 1 | 0 | | | | | | |
| 1 | 1 | | | | | | |

**Q:** What does $p_1 \to p_2$ *really* mean?

Exercise: using Venn diagrams, remind yourself about the meaning of $\to$ by showing when $p_1 \to p_2$ is true and when it is false.

**Example:** Can we determine which *truth assignments* $\tau$ *satisfy* $(x \vee y) \to (\neg x \wedge z)$?

| $x$ | $y$ | $z$ | $x \vee y$ | $\neg x \wedge z$ | $(x \vee y) \to (\neg x \wedge z)$ | $\tau$ |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | | | | |
| 0 | 0 | 1 | | | | |
| 0 | 1 | 0 | | | | |
| 0 | 1 | 1 | | | | |
| 1 | 0 | 0 | | | | |
| 1 | 0 | 1 | | | | |
| 1 | 1 | 0 | | | | |
| 1 | 1 | 1 | | | | |

So, $(x \vee y) \rightarrow (\neg x \wedge z)$ is *true* whenever

$$(\neg x \wedge \neg y \wedge \neg z) \text{ or } (\neg x \wedge \neg y \wedge z) \text{ or } (\neg x \wedge y \wedge z)$$

are true.

Therefore,

$$(x \vee y) \rightarrow (\neg x \wedge z) \Leftrightarrow (\neg x \wedge \neg y \wedge \neg z) \vee (\neg x \wedge \neg y \wedge z) \vee (\neg x \wedge y \wedge z)$$

A *formula* that is a *conjuction* or a bunch of $\wedge$s of *propositional variables* or their *negation* is called a

**DNF:**

A formula is in *Disjunctive Normal Form* if it is the *disjunction* ($\vee$) of *minterms*.

**Example:**

$$(\neg x \wedge \neg y \wedge \neg z) \vee (\neg x \wedge \neg y \wedge z) \vee (\neg x \wedge y \wedge z)$$

is the *DNF* of

$$(x \vee y) \rightarrow (\neg x \wedge z)$$

**Q:** What does the *DNF* construction tell us about *all boolean functions*?

# Boolean Functions and Circuit Diagrams

**Q:** How are *DNF* formulas useful?

Suppose we have a *boolean function* $f(x, y, z)$, *equivalent* to the *DNF* formula:

$$(\neg x \wedge \neg y \wedge \neg z) \vee (\neg x \wedge y \wedge z) \vee (x \wedge \neg y \wedge z) \vee (x \wedge y \wedge \neg z).$$

Then, we can convert $f(x, y, z)$ into a *parse tree*:

How can we create a *circuit diagram* from the *parse tree*?

# Conjuctive Normal Form

Let's look at the truth table again:

| $x$ | $y$ | $z$ | $(x \vee y) \rightarrow (\neg x \wedge z)$ | $\tau$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | |
| 0 | 0 | 1 | 1 | |
| 0 | 1 | 0 | 0 | |
| 0 | 1 | 1 | 1 | |
| 1 | 0 | 0 | 0 | |
| 1 | 0 | 1 | 0 | |
| 1 | 1 | 0 | 0 | |
| 1 | 1 | 1 | 0 | |

We used the *truth assignments* that make $(x \vee y) \rightarrow (\neg x \wedge z)$ true, to construct an equivalent formula in *DNF*.

**Q:** Can we use the *truth values* that make $(x \vee y) \rightarrow (\neg x \wedge z)$ *false* to also construct an *equivalent formula*?

Notice that $(x \vee y) \rightarrow (\neg x \wedge z)$ is *true* when:

Therefore, $(x \vee y) \rightarrow (\neg x \wedge z) \Leftrightarrow$

So *truth tables* are great, right??

**Q:** How many *rows* would we need in a truth table if we have $k$ *propositional variables*?

**Goal:** A *proof system* or *method to determine* whether a *propositional statement* is always true regardless of the truth assignments.

**More Definitions**

**Tautology** We say that a *propositional formula* $P$ is a *tautology* if ________ truth assignment *satisfies* $P$.

**Satisfiable** We say that a *propositional formula* $P$ is *satisfiable* if _________ truth assignment *satisfies* $P$.

**Unsatisfiable** We say that a *propositional formula* $P$ is *unsatisfiable* if ______ truth assignment *satisfies* $P$.

**Examples**

**Tautology**:

**Satisfiable**:

**Unsatisfiable**:

**Logically Implies:** $P$ *logically implies* $Q$ ***iff*** $P \to Q$ *is a tautology.*

**Q:** When is $P \to Q$ a *tautology*?

We can denote "$P$ *logically implies* $Q$" by $P \models Q$ or $P \Rightarrow Q$.

**Q:** What is the difference between $P \Rightarrow Q$ and $P \to Q$?

**Logically Equivalent:** $P$ *and* $Q$ *are logically equivalent* ***iff*** $P \Rightarrow Q$ *and* $Q \Rightarrow P$.

We denote this $P \Leftrightarrow Q$

**Q:** How are "$P \Leftrightarrow Q$" and "$P \leftrightarrow Q$" related?

## Some Logical Equivalences

| | |
|---|---|
| Law of Double negation: | $\Leftrightarrow$ |
| De Morgan's Laws: | $\Leftrightarrow$ |
| | $\Leftrightarrow$ |
| Commutative Laws: | $\Leftrightarrow$ |
| | $\Leftrightarrow$ |
| Associative Laws: | $\Leftrightarrow$ |
| | $\Leftrightarrow$ |
| Distributive Laws: | $\Leftrightarrow$ |
| | $\Leftrightarrow$ |
| Identity Laws: | $\Leftrightarrow$ |
| | $\Leftrightarrow$ |
| $\rightarrow$ Law: | $\Leftrightarrow$ |
| $\leftrightarrow$ Law: | $\Leftrightarrow$ |

# Propositional Logic Review

**Idea** Want a *formal* way to make inferences from *boolean statements*.

**DEFINITIONS:**

- **Syntax** The *symbols* that we use to *represent expressions*

  e.g., a programing language: a piece of code *compiles* if it has *proper syntax*.

- **Semantics** The *meaning* of what the *symbols represent.*

  e.g., a programming language: a piece of code meets its *specifications* if the *semantics* are correct.

- **Proposition** a *statement* that is a *sentence* that can be *evaluated* to **true** or **false**.

- **Propositional Variable** a *variable* that stands for or represents a *primitive proposition*, i.e., the *simplest propositions* we are considering.

  We denote the *set* of *propositional variables* as $PV$

- **Connectives** The *symbols*, $\{\vee, \wedge, \rightarrow, \leftrightarrow, \neg\}$, that we use to join *propositions* together to make *new propositions*.

# Proving Two Formulas are Logically Equivalent

## Example 1

$$(x \to y) \land (x \to z) \Leftrightarrow x \to (y \land z)$$

**Proof.**

$$\begin{aligned}(x \to y) \land (x \to z) &\Leftrightarrow \\ &\Leftrightarrow \\ &\Leftrightarrow \quad x \to (y \land z) \quad \to \text{ law}\end{aligned}$$

## Example 2

$$(Q \to P) \land (\neg Q \to P) \Leftrightarrow P$$

**Proof:**

$$\begin{aligned}(Q \to P) \land (\neg Q \to P) &\Leftrightarrow \\ &\Leftrightarrow \\ &\Leftrightarrow \\ &\Leftrightarrow \\ &\Leftrightarrow \\ &\Leftrightarrow\end{aligned}$$

**Q:** What did we just prove?

## Proving Two Formulas are *NOT* Equivalent

**Q:** How do we show that two formula are **not** *equivalent*?

**Example 3**

$$(y \to x) \wedge (z \to x) \overset{??}{\Leftrightarrow} (y \wedge z) \to x$$

Simplify a bit first:

$$\begin{aligned} (y \to x) \wedge (z \to x) &\Leftrightarrow \\ &\Leftrightarrow \\ &\Leftrightarrow \\ &\Leftrightarrow \end{aligned}$$

**Q:** Is there a *truth assignment* that satisfies *only* one of $(y \wedge z) \to x$ and $(y \vee z) \to x$?

# Back to Stuctural Induction...

We have already seen that $p \to q \Leftrightarrow \neg p \vee q$.

**Q.** Can all *propositional formulas* be rewritten using just $\vee$ and $\neg$?

## An Example

Recall the definition of $\mathcal{F}$. $\mathcal{F}$ be the smallest set such that:

**Basis**: The set of *propositional variables* belong to $\mathcal{F}$, e.g., $P, Q, R, \ldots \in \mathcal{F}$

**Induction Step:** If $P, Q$ belong to $\mathcal{F}$ then

1. $(P \vee Q) \in \mathcal{F}$
2. $(P \wedge Q) \in \mathcal{F}$
3. $(P \to Q) \in \mathcal{F}$
4. $\neg P \in \mathcal{F}$

**CLAIM:** Let $\mathcal{F}$ be as defined above. If $R \in \mathcal{F}$ then $R$ can be *constructed* using only 4. and 1. above. I.e.,

> "*$\forall R \in \mathcal{F}$, there exists a logically equivalent formula in $\mathcal{F}$ constructed using only the operators $\neg$ and $\vee$.*"

**Claim:** "*$\forall R \in \mathcal{F}$, there exists a logically equivalent formula in $\mathcal{F}$ constructed using only the operators $\neg$ and $\vee$.*"

*Proof.* Structural induction on $R \in \mathcal{F}$.

**Basis:**

**Inductive Step:** If $R$ is not a propositional variable, then $R$ is constructed from one of the 4 rules.

**Case 1.** $R$ is $(P \vee Q)$

**Case 2.** $R$ is $(P \wedge Q)$.

What is $(P \wedge Q)$ logically equivalent to in terms of $\vee$ and $\neg$?

**Case 3.** $R$ is $(P \rightarrow Q)$.

What is $(P \rightarrow Q)$ logically equivalent to in terms of $\vee$ and $\neg$?

**Case 4.** $R$ is $\neg P$.

Therefore, by structural induction the claim holds. □

**Q.** What does this tell you about $\wedge, \vee, \rightarrow$ and $\neg$?

**A.**

## Proving an item does *NOT* belong to a set

Consider the following *set* $\mathcal{H}$ defined by *induction*:

$\mathcal{H}$ is the smallest set such that:

**Basis:** The set of *propositional variables* belongs to $\mathcal{H}$

**Induction Step:** if $P \in \mathcal{H}$ and $Q \in \mathcal{H}$ then

1. $P \vee Q \in \mathcal{H}$
2. $P \wedge Q \in \mathcal{H}$

**Q:** Can all *propositional formulas* belong to $\mathcal{H}$?

**Q:** How do we prove that an item *does not belong* to an *inductively defined set*?

**Q:** Suppose that all *propositional variables* are assigned a *value* of *true*. What does this tell you about *every* item in $\mathcal{H}$?

**Q:** How does this help us?

Consider again $\neg P$. If $P \Leftrightarrow T$, what is the *truth value* of $\neg P$?

Let's prove our claim:

**CLAIM 3**: *$\forall h \in \mathcal{H}$, if every propositional variable in $h$ has value true, then $h$ is true.*

*Proof.* By structural induction on $R \in \mathcal{H}$.

**Basis:**

**Inductive Step:** Assume that $P, Q \in \mathcal{H}$ satisify the claim.

**Case 1:** $R \leftrightarrow P \vee Q$:

**Case 2:**$R \leftrightarrow P \wedge Q$:

□

**CLAIM 4**: $\neg P \notin \mathcal{H}$.

*Proof.*

$\square$

**Q:** What does CLAIM 4 tell us about $\wedge$ and $\vee$ with respect to the *set* of all *propositional formulas*?