G22.3033-002 Logic in Computer Science

Instructor:

Clark W. Barrett
barrett@cs.nyu.edu
Outline

- Goals and Organization
- A Motivating Example
- Propositional Logic: Syntax
- Induction
- Propositional Logic: Well-Formed Formulas
- Recursion
- Propositional Logic: Semantics

Material is drawn from Sections 1.1–1.4 of Enderton.
Course Goals

- **Prerequisites:**
  - Familiarity with discrete mathematics (sets, functions, induction, graphs)
  - Working acquaintance with the language of logic

- **Contents of the Course**
  - A precise and formal treatment of two fundamental logics: propositional and first-order
  - A brief overview of some other important logics including second-order and modal
  - An introduction to the mathematical tools required to reason about and within these logics
  - A periodic glimpse of *Interesting Applications* of formal logic in Computer Science
Course Information

Webpage:

http://cs.nyu.edu/courses/fall03/G22.3033-002/index.htm

Book:


Assignments:

There will be a weekly assignment which will generally be due the following week (unless otherwise indicated).

Exams:

There will be a take-home midterm exam and a final exam.

Grading:

Weekly Assignments: 40%, Midterm: 30%, Final: 30%
A Motivating Example

Recall that a graph consists of a set $V$ of vertices and a set $E$ of edges, where each edge is an unordered pair of vertices.

A complete graph on $n$ vertices is a graph with $|V| = n$ such that $E$ contains all possible pairs of vertices.

The Ramsey number is the smallest integer such that if a complete graph on $n$ vertices is colored with colors, then for some $k$, there must exist a complete subgraph of $k$ vertices, all of whose edges have the same color.
A Motivating Example

Recall that a graph consists of a set $V$ of vertices and a set $E$ of edges, where each edge is an unordered pair of vertices.

A complete graph on $n$ vertices is a graph with $|V| = n$ such that $E$ contains all possible pairs of vertices.

How many edges?
A Motivating Example

Recall that a graph consists of a set $V$ of vertices and a set $E$ of edges, where each edge is an unordered pair of vertices.

A complete graph on $n$ vertices is a graph with $\left| V \right| = n$ such that $E$ contains all possible pairs of vertices.

How many edges?

The Ramses number $r(x_1, \ldots, x_n)$ is the smallest integer $p$ such that if a complete graph $G$ on $p$ vertices is colored with $n$ colors, then for some $i$, $1 \leq i \leq n$, there must exist a complete subgraph of $G$ with $x_i$ vertices, all of whose edges have the same color.
A Motivating Example

Recall that a *graph* consists of a set $V$ of vertices and a set $E$ of edges, where each edge is an unordered pair of vertices.

A *complete graph* on $n$ vertices is a graph with $|V| = n$ such that $E$ contains all possible pairs of vertices.

How many edges?

The *Ramses number* $r(x_1, \ldots, x_n)$ is the smallest integer $p$ such that if a complete graph $G$ on $p$ vertices is colored with $n$ colors, then for some $i$, $1 \leq i \leq n$, there must exist a complete subgraph of $G$ with $x_i$ vertices, all of whose edges have the same color.

What is $r(3, 3)$?
A Motivating Example

Recall that a graph consists of a set $V$ of vertices and a set $E$ of edges, where each edge is an unordered pair of vertices.

A complete graph on $n$ vertices is a graph with $|V| = n$ such that $E$ contains all possible pairs of vertices.

How many edges?

The Ramses number $r(x_1, \ldots, x_n)$ is the smallest integer $p$ such that if a complete graph $G$ on $p$ vertices is colored with $n$ colors, then for some $i$, $1 \leq i \leq n$, there must exist a complete subgraph of $G$ with $x_i$ vertices, all of whose edges have the same color.

What is $r(3, 3)$?

What is $r(3, 3, 3)$?
Logic

A formal logic is defined by its \textit{syntax} and \textit{semantics}.

**Syntax**

- An \textit{alphabet} is a set of symbols.
- A finite sequence of these symbols is called an \textit{expression}.
- A set of rules defines the \textit{well-formed} expressions.

**Semantics**

- Gives meaning to well-formed expressions
- Formal notions of induction and recursion are required to provide a rigorous semantics.
Propositional (Sentential) Logic

Propositional logic is simple but extremely important in Computer Science

1. It is the basis for day-to-day reasoning (in programming, LSATs, etc.)

2. It is the theory behind digital circuits.

3. Many problems can be translated into propositional logic.

4. It is an important part of more complex logics (such as first-order logic, also called predicate logic, which we’ll discuss later.)
# Propositional Logic: Syntax

## Alphabet

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(</td>
<td>Left parenthesis</td>
</tr>
<tr>
<td>)</td>
<td>Right parenthesis</td>
</tr>
<tr>
<td>¬</td>
<td>Negation symbol</td>
</tr>
<tr>
<td>∧</td>
<td>Conjunction symbol</td>
</tr>
<tr>
<td>∨</td>
<td>Disjunction symbol</td>
</tr>
<tr>
<td>→</td>
<td>Conditional symbol</td>
</tr>
<tr>
<td>↔</td>
<td>Bi-conditional symbol</td>
</tr>
<tr>
<td>$A_1$</td>
<td>First propositional symbol</td>
</tr>
<tr>
<td>$A_2$</td>
<td>Second propositional symbol</td>
</tr>
<tr>
<td>$\ldots$</td>
<td></td>
</tr>
<tr>
<td>$A_n$</td>
<td>$n$th propositional symbol</td>
</tr>
<tr>
<td>$\ldots$</td>
<td></td>
</tr>
</tbody>
</table>

We are assuming a *countable* alphabet, but most of our conclusions hold equally well for an *uncountable* alphabet.
Propositional Logic: Syntax

Alphabet

- *Propositional connective* symbols: $\neg$, $\land$, $\lor$, $\rightarrow$, $\leftrightarrow$.

- *Logical* symbols: $\neg$, $\land$, $\lor$, $\rightarrow$, $\leftrightarrow$, $(, )$.

- *Parameters* or *nonlogical symbols*: $A_1$, $A_2$, $A_3$, $\ldots$

The meaning of logical symbols is always the same. The meaning of nonlogical symbols depends on the context.
Propositional Logic: Syntax

An expression is a sequence of symbols. A sequence is denoted explicitly by a comma separated list enclosed in angle brackets: \(<a_1, \ldots, a_m>\).

Examples

\(<(, A_1, \land, A_3, )>\)
\(<(, (, \neg, A_1, ), \rightarrow, A_2, )>\)
\(<(, ), \leftrightarrow, ), A_5>\)
Propositional Logic: Syntax

An expression is a sequence of symbols. A sequence is denoted explicitly by a comma separated list enclosed in angle brackets: \(<a_1, \ldots, a_m>\).

Examples

\(<(, A_1, \land, A_3, )>\) \hspace{1cm} (A_1 \land A_3)
\(<(, (, \neg, A_1, ), \rightarrow, A_2, )>\) \hspace{1cm} ((\neg A_1) \rightarrow A_2)
\(<), ), \leftrightarrow, ), A_5>\) \hspace{1cm} )) \leftrightarrow)A_5

For convenience, we will write these sequences as a simple string of symbols, with the understanding that the formal structure represented is a sequence containing exactly the symbols in the string.

The formal meaning becomes important when trying to prove things about expressions.
Propositional Logic: Syntax

An expression is a sequence of symbols. A sequence is denoted explicitly by a comma separated list enclosed in angle brackets: $<a_1, \ldots, a_n>$. 

Examples

$<(<, A_1, \wedge, A_3, >)>$ \hspace{1cm} $(A_1 \wedge A_3)$

$<(<, (<, \neg, A_1, >), \rightarrow, A_2, >) >$ \hspace{1cm} $((\neg A_1) \rightarrow A_2)$

$<(<), >, \leftrightarrow, >, A_5>$ \hspace{1cm} $>) \leftrightarrow A_5$

For convenience, we will write these sequences as a simple string of symbols, with the understanding that the formal structure represented is a sequence containing exactly the symbols in the string.

The formal meaning becomes important when trying to prove things about expressions.

We want to restrict the kinds of expressions that will be allowed.
Propositional Logic: Syntax

We define the set $W$ of *well-formed formulas* (wff’s) as follows.

(a) Every expression consisting of a single propositional symbol is in $W$.

(b) If $\alpha$ and $\beta$ are in $W$, so are $(\neg \alpha), (\alpha \land \beta), (\alpha \lor \beta), (\alpha \rightarrow \beta)$, and $(\alpha \leftrightarrow \beta)$.

(c) No expression is in $W$ unless forced by (a) or (b)

This definition is *inductive*: the set being defined is used as part of the definition.
Propositional Logic: Syntax

We define the set $W$ of *well-formed formulas* (wff’s) as follows.

(a) Every expression consisting of a single propositional symbol is in $W$.

(b) If $\alpha$ and $\beta$ are in $W$, so are $(-\alpha)$, $(\alpha \land \beta)$, $(\alpha \lor \beta)$, $(\alpha \rightarrow \beta)$, and $(\alpha \leftrightarrow \beta)$.

(c) No expression is in $W$ unless forced by (a) or (b).

This definition is *inductive*: the set being defined is used as part of the definition.

How would you use this definition to prove that $)) \leftrightarrow A_5$ is not a wff?
Propositional Logic: Syntax

We define the set $W$ of well-formed formulas (wff’s) as follows.

(a) Every expression consisting of a single propositional symbol is in $W$.

(b) If $\alpha$ and $\beta$ are in $W$, so are $(\neg \alpha)$, $(\alpha \land \beta)$, $(\alpha \lor \beta)$, $(\alpha \rightarrow \beta)$, and $(\alpha \leftrightarrow \beta)$.

(c) No expression is in $W$ unless forced by (a) or (b).

This definition is inductive: the set being defined is used as part of the definition.

How would you use this definition to prove that $() \leftrightarrow A_5$ is not a wff?

Item (c) is too vague for our purposes. There are two ways to make it more precise: top-down and bottom-up. Both require a formal notion of induction.
Induction

Suppose we have a property $P$ which is defined in terms of a natural number $n$. We wish to show that $P$ holds for all natural numbers.

**Base case**

Show that $P$ holds for 0.

**Inductive case**

Show that if $P$ holds for $n$, then $P$ holds for $n + 1$. 
Example

$P(n)$ is the property $\sum_{i=0}^{n} i = \frac{n(n + 1)}{2}$. 
Example

\[ P(n) \] is the property \[ \sum_{i=0}^{n} i = \frac{n(n + 1)}{2}. \]

Base case: \[ \sum_{i=0}^{0} i = 0 = \frac{0(0 + 1)}{2}. \]
Example

$P(n)$ is the property $\sum_{i=0}^{n} i = \frac{n(n+1)}{2}$.

Base case: $\sum_{i=0}^{0} i = 0 = \frac{0(0+1)}{2}$.

Inductive case: Assume $P(k): \sum_{i=1}^{k} i = \frac{k(k+1)}{2}$. 
Example

\[ P(n) \] is the property \[ \sum_{i=0}^{n} i = \frac{n(n + 1)}{2}. \]

**Base case:** \[ \sum_{i=0}^{0} i = 0 = \frac{0(0 + 1)}{2}. \]

**Inductive case:** Assume \( P(k) : \sum_{i=1}^{k} i = \frac{k(k + 1)}{2}. \)

Then \[ \sum_{1}^{k+1} i = \sum_{1}^{k} i + (k + 1) \]
Example

$P(n)$ is the property $\sum_{i=0}^{n} i = \frac{n(n + 1)}{2}$.

Base case: $\sum_{i=0}^{0} i = 0 = \frac{0(0 + 1)}{2}$.

Inductive case: Assume $P(k): \sum_{i=1}^{k} i = \frac{k(k + 1)}{2}$.

Then $\sum_{1}^{k+1} i = \sum_{1}^{k} i + (k + 1)$

$= \frac{k(k + 1)}{2} + (k + 1)$
Example

\( P(n) \) is the property \( \sum_{i=0}^{n} i = \frac{n(n + 1)}{2} \).

Base case: \( \sum_{i=0}^{0} i = 0 = \frac{0(0 + 1)}{2} \).

Inductive case: Assume \( P(k): \sum_{i=1}^{k} i = \frac{k(k + 1)}{2} \).

Then \( \sum_{1}^{k+1} i = \sum_{1}^{k} i + (k + 1) \)

\[ = \frac{k(k + 1)}{2} + (k + 1) \]

\[ = \frac{k(k + 1) + 2(k + 1)}{2} \]
Example

$P(n)$ is the property $\sum_{i=0}^{n} i = \frac{n(n + 1)}{2}$.

Base case: $\sum_{i=0}^{0} i = 0 = \frac{0(0 + 1)}{2}$.

Inductive case: Assume $P(k): \sum_{i=1}^{k} i = \frac{k(k + 1)}{2}$.

Then $\sum_{1}^{k+1} i = \sum_{1}^{k} i + (k + 1) = \frac{k(k + 1)}{2} + (k + 1) = \frac{k(k + 1) + 2(k + 1)}{2} = \frac{(k + 1)(k + 2)}{2}$.
Example

\( P(n) \) is the property \[ \sum_{i=0}^{n} i = \frac{n(n + 1)}{2}. \]

**Base case:** \[ \sum_{i=0}^{0} i = 0 = \frac{0(0 + 1)}{2}. \]

**Inductive case:** Assume \( P(k) \): \[ \sum_{i=1}^{k} i = \frac{k(k + 1)}{2}. \]

Then \[ \sum_{i=1}^{k+1} i = \sum_{i=1}^{k} i + (k + 1) \]
\[ = \frac{k(k + 1)}{2} + (k + 1) \]
\[ = \frac{k(k + 1) + 2(k + 1)}{2} \]
\[ = \frac{(k + 1)(k + 2)}{2} \]

Since \( P(0) \) holds and \( P(k) \rightarrow P(k + 1) \), it follows that \( P(n) \) holds for all natural numbers \( n \).
Induction

Mathematical induction is a special case of a more general principle.

In general, whenever a set can be defined inductively, induction can be used to prove things about elements in the set.

What is an inductive definition?
**Induction**

Let $U$ be some *universal* set, and suppose we wish to define some subset $C$ of $U$ inductively. This can be done as follows.

- $B$ is an initial subset of $U$.
- $F$ is a family of functions on $U$.

Informally, $B$ is the base case for our inductive definition. These are the elements we are starting with. The set $F$ describes how to obtain new elements from old elements. The set $C$ is the set of all elements that are either in $B$ or can be obtained from $B$ using the functions in $F$.

**Example**

The natural numbers $\mathcal{N}$ can be defined as follows:

Let $U$ be the set of all real numbers, $B=\{0\}$ and $F=\{\text{succ}\}$, where $\text{succ}$ is the successor function defined as $\text{succ}(x)=x+1$. 
Induction

General Inductive Definition

- $U$ is a universal set
- $B$ is an initial subset of $U$.
- $F$ is a family of functions on $U$.

How do we use this to obtain the desired set $C$?

We can define $C^*$, the \textit{top-down} version of $C$ as follows:

- A set $S$ is \textit{closed} under $F$ iff for each $f \in F$, if $x_1, \ldots, x_n \in S$ and $f(x_1, \ldots, x_n) = y$ for some $y \in U$, then $y \in S$.
- A set $S$ is \textit{inductive} if $B \subseteq S$ and $S$ is closed under $F$.
- The set $C^*$ is defined as the intersection of all inductive subsets of $U$.

This is \textit{top-down} because we take something too big (inductive sets) and use their intersection to construct the desired set.
Induction

Example

Recall our inductive definition of the natural numbers:

- \( U = \mathcal{R} \), where \( \mathcal{R} \) is the set of real numbers.
- \( B = \{0\} \)
- \( F = \{\text{succ}\} \), where \( \text{succ}(x) = x + 1 \).

\( \mathcal{R} \) is closed under \( \text{succ} \) and is also inductive because \( 0 \in \mathcal{R} \).
Induction

Example

Recall our inductive definition of the natural numbers:

- $U = \mathcal{R}$, where $\mathcal{R}$ is the set of real numbers.
- $B = \{0\}$
- $F = \{\text{succ}\}$, where $\text{succ}(x) = x + 1$.

$\mathcal{R}$ is closed under $\text{succ}$ and is also inductive because $0 \in \mathcal{R}$.

What about

- The set of all (positive and negative) integers?
Induction

Example

Recall our inductive definition of the natural numbers:

- $U = \mathcal{R}$, where $\mathcal{R}$ is the set of real numbers.
- $B = \{0\}$
- $F = \{\text{succ}\}$, where $\text{succ}(x) = x + 1$.

$\mathcal{R}$ is closed under $\text{succ}$ and is also inductive because $0 \in \mathcal{R}$.

What about

- The set of all (positive and negative) integers?
- The set $\{1, 2, 3, \ldots\}$?
Induction

Example

Recall our inductive definition of the natural numbers:

- \( U = \mathcal{R} \), where \( \mathcal{R} \) is the set of real numbers.
- \( B = \{0\} \)
- \( F = \{\text{succ}\} \), where \( \text{succ}(x) = x + 1 \).

\( \mathcal{R} \) is closed under \( \text{succ} \) and is also inductive because \( 0 \in \mathcal{R} \).

What about

- The set of all (positive and negative) integers?
- The set \( \{1, 2, 3, \ldots\} \)?
- The set \( \{0.5, 1.5, 2.5, \ldots\} \)?
**Induction**

**Example**

Recall our inductive definition of the natural numbers:

- \( U = R \), where \( R \) is the set of real numbers.
- \( B = \{0\} \)
- \( F = \{\text{succ}\} \), where \( \text{succ}(x) = x + 1 \).

\( R \) is closed under \( \text{succ} \) and is also inductive because \( 0 \in R \).

What about

- The set of all (positive and negative) integers?
- The set \( \{1, 2, 3, \ldots\} \)?
- The set \( \{0.5, 1.5, 2.5, \ldots\} \)?

It is not hard to see that \( N \) is the smallest set which is closed and inductive.
Induction

General Inductive Definition

- \( U \) is a universal set
- \( B \) is an initial subset of \( U \).
- \( F \) is a family of functions on \( U \).

We can define \( C_* \), the \textit{bottom-up} version of \( C \) as follows:

- A \textit{construction sequence} is a finite sequence \(< x_0, \ldots, x_n >\) of elements of \( U \) such that for each \( i < n \), one of the following holds:
  - \( x_i \in B \)
  - \( f(x_{j_0}, \ldots, x_{j_m}) = x_i \) where \( 0 \leq j_k < i \) (for \( k = 0 \ldots m \)) for some \( f \in F \)

- Let \( C_n \) be the set of elements \( x \) such that some construction sequence of length \( n \) ends with \( x \). Note that \( C_1 = B \).

- The set \( C_* \) is defined as the set of all elements \( x \) such that some construction sequence ends with \( x \) (i.e. \( C_* = \bigcup C_n \)).

This is \textit{bottom-up} because we show how to construct each element and then put them together to get \( C_* \).
Induction

Example

Recall our inductive definition of the natural numbers:

- $U = \mathcal{R}$, where $\mathcal{R}$ is the set of real numbers.
- $B = \{0\}$
- $F = \{\text{succ}\}$, where $\text{succ}(x) = x + 1$.

The construction sequences of this definition are

- $\langle 0 \rangle$
- $\langle 0, 1 \rangle$
- $\langle 0, 1, 2 \rangle$
- $\ldots$

The set of all the last items in the construction sequences gives $\mathcal{N}$. 
Induction

As you may have suspected, given any inductive definition, it is always the case that $C^* = C_*$.

Proof

$C^* \subseteq C_*$: We show that $C_*$ is inductive. Clearly $B \subseteq C_*$ since $C_1 = B$. Suppose $x_1, \ldots, x_n \in C_*$ and $f(x_1, \ldots, x_n) = y$ for some $f \in F$. Then we can concatenate the construction sequences for each $x_i$ and append $y$ to get a valid construction sequence for $y$. Thus, $C_*$ is closed under $F$, and thus $C_*$ is inductive. Since $C^*$ is the intersection of all inductive sets, it follows that $C^* \subseteq C_*$.

$C_* \subseteq C^*$: We show that if $< x_0, \ldots, x_n >$ is any construction sequence, then $x_n \in C^*$. We use ordinary induction on $n$. For the base case, when $n = 0$, we have that $x_0 \in B$, so it follows that $x_0 \in C^*$. For the induction case, consider a sequence $< x_0, \ldots, x_{n+1} >$. We know that $f(x_{j_0}, \ldots, x_{j_m}) = x_{n+1}$ where $0 \leq j_k < n + 1$ (for each $k = 0 \ldots m$) for some $f \in F$, but by the induction hypothesis, each $x_{j_i} \in C^*$ for $i < m$, so, since $C^*$ is closed under $F$ it follows that $x_{n+1} \in C^*$.
Induction

Since $C_* = C^*$, we can call the set simply $C$. We also refer to it as the set generated from $B$ by $F$.

Now, given any inductive definition of a set, we can prove things about that set using the following principle.

**Induction Principle**

If $C$ is the set generated from $B$ by $F$ and $S$ is a set which includes $B$ and is closed under $F$ (i.e. $S$ is inductive), then $C \subseteq S$.

**Proof**

Since $S$ is inductive, and $C = C^*$ is the intersection of all inductive sets, it follows that $C \subseteq S$.

We can now show how mathematical induction is a special case of the induction principle.
Induction

Example

Consider again the natural numbers defined inductively as follows:

- $U = \mathcal{R}$, where $\mathcal{R}$ is the set of real numbers.
- $B = \{0\}$
- $F = \{\text{succ}\}$, where $\text{succ}(x) = x + 1$.

Let $\mathcal{N}$ be the set generated by $B$ from $F$.

Let $S$ be the set of all real numbers $n$ for which $\sum_{i=0}^{n} i = \frac{n(n + 1)}{2}$.

Our earlier proof shows that $0 \in S$ and if $k \in S$ then $k + 1 \in S$. In other words, $S$ is inductive.

Therefore, by the induction principle, $\mathcal{N} \subseteq S$.

Thus, $\sum_{i=0}^{n} i = \frac{n(n + 1)}{2}$ for all natural numbers $n \in \mathcal{N}$.
Propositional Logic: Well-Formed Formulas

We can now use a formal inductive definition to define the set $W$ of well-formed formulas in propositional logic.

- $U =$
- $B =$
- $F =$
Propositional Logic: Well-Formed Formulas

We can now use a formal inductive definition to define the set $W$ of well-formed formulas in propositional logic.

- $U =$ the set of all expressions.
- $B =$
- $F =$
Propositional Logic: Well-Formed Formulas

We can now use a formal inductive definition to define the set $W$ of well-formed formulas in propositional logic.

- $U = \text{the set of all expressions.}$
- $B = \text{the set of expressions consisting of a single propositional symbol.}$
- $F =$
Propositional Logic: Well-Formed Formulas

We can now use a formal inductive definition to define the set $W$ of well-formed formulas in propositional logic.

- $U =$ the set of all expressions.
- $B =$ the set of expressions consisting of a single propositional symbol.
- $F =$ the set of formula-building operations:
  - $\mathcal{E}_\neg(\alpha) = (\neg \alpha)$
  - $\mathcal{E}_\land(\alpha, \beta) = (\alpha \land \beta)$
  - $\mathcal{E}_\lor(\alpha, \beta) = (\alpha \lor \beta)$
  - $\mathcal{E}_\to(\alpha, \beta) = (\alpha \to \beta)$
  - $\mathcal{E}_\leftrightarrow(\alpha, \beta) = (\alpha \leftrightarrow \beta)$
Propositional Logic: Well-Formed Formulas

Given our inductive definition of well-formed formulas, we can use the induction principle to prove things about the set $W$ of well-formed formulas.

**Example**

Prove that any $\text{wff}$ has the same number of left parentheses and right parentheses.

**Proof**

Let $l(\alpha)$ be the number of left parentheses and $r(\alpha)$ the number of right parentheses in an expression $\alpha$. Let $S$ be the set of all expressions $\alpha$ such that $l(\alpha) = r(\alpha)$. We wish to show that $W \subseteq S$. This follows from the induction principle if we can show that $S$ is inductive.

**Base Case:**

We must show that $B \subseteq S$. Recall that $B$ is the set of expressions consisting of a single propositional symbol. It is clear that for such expressions, $l(\alpha) = r(\alpha) = 0$. 
Propositional Logic: Well-Formed Formulas

Inductive Case:

We must show that $S$ is closed under each formula-building operator in $F$. 

Propositional Logic: Well-Formed Formulas

Inductive Case:

We must show that $S$ is closed under each formula-building operator in $F$.

- $\mathcal{E}_\neg$
  Suppose $\alpha \in S$. We know that $\mathcal{E}_\neg(\alpha) = (\neg\alpha)$. It follows that $l(\mathcal{E}_\neg(\alpha)) = 1 + l(\alpha)$ and $r(\mathcal{E}_\neg(\alpha)) = 1 + r(\alpha)$.
  But because $\alpha \in S$, we know that $l(\alpha) = r(\alpha)$, so it follows that $l(\mathcal{E}_\neg(\alpha)) = r(\mathcal{E}_\neg(\alpha))$, and thus $\mathcal{E}_\neg(\alpha) \in S$. 
Propositional Logic: Well-Formed Formulas

Inductive Case:

We must show that $S$ is closed under each formula-building operator in $F$.

- $\mathcal{E}_-$
  Suppose $\alpha \in S$. We know that $\mathcal{E}_-(\alpha) = (\neg \alpha)$. It follows that $l(\mathcal{E}_-(\alpha)) = 1 + l(\alpha)$ and $r(\mathcal{E}_-(\alpha)) = 1 + r(\alpha)$.
  But because $\alpha \in S$, we know that $l(\alpha) = r(\alpha)$, so it follows that $l(\mathcal{E}_-(\alpha)) = r(\mathcal{E}_-(\alpha))$, and thus $\mathcal{E}_-(\alpha) \in S$.

- $\mathcal{E}_\land$
  Suppose $\alpha, \beta \in S$. We know that $\mathcal{E}_\land(\alpha, \beta) = (\alpha \land \beta)$. Thus $l(\mathcal{E}_\land(\alpha, \beta)) = 1 + l(\alpha) + l(\beta)$ and $r(\mathcal{E}_\land(\alpha, \beta)) = 1 + r(\alpha) + r(\beta)$.
  As before, it follows from the inductive hypothesis that $\mathcal{E}_\land(\alpha, \beta) \in S$. 
Propositional Logic: Well-Formed Formulas

Inductive Case:

We must show that $S$ is closed under each formula-building operator in $F$.

- $\text{E}_\neg$
  Suppose $\alpha \in S$. We know that $\text{E}_\neg(\alpha) = (\neg \alpha)$. It follows that
  \[ l(\text{E}_\neg(\alpha)) = 1 + l(\alpha) \text{ and } r(\text{E}_\neg(\alpha)) = 1 + r(\alpha). \]
  But because $\alpha \in S$, we know that $l(\alpha) = r(\alpha)$, so it follows that
  \[ l(\text{E}_\neg(\alpha)) = r(\text{E}_\neg(\alpha)), \text{ and thus } \text{E}_\neg(\alpha) \in S. \]

- $\text{E}_\land$
  Suppose $\alpha, \beta \in S$. We know that $\text{E}_\land(\alpha, \beta) = (\alpha \land \beta)$. Thus
  \[ l(\text{E}_\land(\alpha, \beta)) = 1 + l(\alpha) + l(\beta) \text{ and } r(\text{E}_\land(\alpha, \beta)) = 1 + r(\alpha) + r(\beta). \]
  As before, it follows from the inductive hypothesis that $\text{E}_\land(\alpha, \beta) \in S$.

- The arguments for $\text{E}_\lor$, $\text{E}_\rightarrow$, and $\text{E}_\leftrightarrow$ are exactly analogous to the one for $\text{E}_\land$. 

\[ \square \]
Propositional Logic: Well-Formed Formulas

Inductive Case:

We must show that $S$ is closed under each formula-building operator in $F$.

- $\mathcal{E}_-$
  Suppose $\alpha \in S$. We know that $\mathcal{E}_-(\alpha) = (\neg \alpha)$. It follows that $l(\mathcal{E}_-(\alpha)) = 1 + l(\alpha)$ and $r(\mathcal{E}_-(\alpha)) = 1 + r(\alpha)$.
  But because $\alpha \in S$, we know that $l(\alpha) = r(\alpha)$, so it follows that $l(\mathcal{E}_-(\alpha)) = r(\mathcal{E}_-(\alpha))$, and thus $\mathcal{E}_-(\alpha) \in S$.

- $\mathcal{E}_\land$
  Suppose $\alpha, \beta \in S$. We know that $\mathcal{E}_\land(\alpha, \beta) = (\alpha \land \beta)$. Thus $l(\mathcal{E}_\land(\alpha, \beta)) = 1 + l(\alpha) + l(\beta)$ and $r(\mathcal{E}_\land(\alpha, \beta)) = 1 + r(\alpha) + r(\beta)$.
  As before, it follows from the inductive hypothesis that $\mathcal{E}_\land(\alpha, \beta) \in S$.

- The arguments for $\mathcal{E}_\lor$, $\mathcal{E}_\rightarrow$, and $\mathcal{E}_\leftrightarrow$ are exactly analogous to the one for $\mathcal{E}_\land$.

Since $S$ includes $\mathcal{B}$ and is closed under the operations in $F$, it is inductive. It follows by the induction principle that $W \subseteq S$. 

$\square$
Propositional Logic: Well-Formed Formulas

Now we can return to the question of how to prove that an expression is not a \textit{wff}.

How do we know that \( ))) \leftrightarrow )) A_5 \) is not a \textit{wff}?
Propositional Logic: Well-Formed Formulas

Now we can return to the question of how to prove that an expression is not a \textit{wff}.

How do we know that \( \)) \leftrightarrow \) \( A_5 \) is not a \textit{wff}?

It does not have the same number of left and right parentheses.

It follows from the theorem we just proved that \( \)) \leftrightarrow \) \( A_5 \) is not a \textit{wff}. 
Recursion

Suppose we wish to define a function whose domain is an inductively defined set. The natural way to do this is using recursion.

Assume an inductive definition with universal set $U$, base set $B \subseteq U$, and a family of functions $F$ which take one or more arguments from $U$ and return an element of $U$. Let $C$ be the set defined by this definition.

Now, we wish to define a function $h$ whose domain is $C$. We can do this as follows.

- For each $x \in B$, explicitly define $h(x)$.
- For each function $f(x_0, \ldots, x_n) \in F$, give a rule for computing $h(f(x_0, \ldots, x_n))$ given $h(x_0), \ldots, h(x_n)$. 
Recursion

Examples

Recall our inductive definition of $\mathcal{N} : U = \mathcal{R}, B = \{0\}, F = \{\text{succ}\}$. Suppose we wish to define the factorial function $\text{fact}$. We can do this as follows:

- $\text{fact}(0) = 1$
- $\text{fact}(\text{succ}(n)) = (n + 1) \times \text{fact}(n)$
Recursion

Examples

Recall our inductive definition of $\mathcal{N}: U = \mathcal{R}, B = \{0\}, F = \{\text{succ}\}$. Suppose we wish to define the factorial function $\text{fact}$. We can do this as follows:

- $\text{fact}(0) = 1$
- $\text{fact}(\text{succ}(n)) = (n + 1) \times \text{fact}(n)$

In general, however, a recursive definition like this one does not guarantee the existence of such a function. Consider the following alternative inductive definition of $\mathcal{N}: U = \mathcal{R}, B = \{0\}, F = \{\text{succ}, \text{mult}\}$, where $\text{mult}(x, y) = x \times y$. Now we define a function $h$ as follows:

- $h(0) = 0$
- $h(\text{succ}(n)) = h(n) + 2$
- $h(\text{mult}(m, n)) = h(m) \times h(n)$

What is $h(1)$?
Recursion

Examples

Recall our inductive definition of $\mathcal{N}: U = \mathcal{R}, B = \{0\}, F = \{\text{succ}\}$. Suppose we wish to define the factorial function $\text{fact}$. We can do this as follows:

- $\text{fact}(0) = 1$
- $\text{fact}(\text{succ}(n)) = (n + 1) \times \text{fact}(n)$

In general, however, a recursive definition like this one does not guarantee the existence of such a function. Consider the following alternative inductive definition of $\mathcal{N}: U = \mathcal{R}, B = \{0\}, F = \{\text{succ}, \text{mult}\}$, where $\text{mult}(x, y) = x \times y$. Now we define a function $h$ as follows:

- $h(0) = 0$
- $h(\text{succ}(n)) = h(n) + 2$
- $h(\text{mult}(m, n)) = h(m) \times h(n)$

What is $h(1)$?

How can we be sure that a recursive definition is well-defined?
Recursion

To ensure well-defined recursive definitions, an inductive definition of a set $C$ must satisfy some additional constraints.

- The restriction of each function $f \in F$ to $C$ must be one-to-one.
- The range of the same restriction of each function in $F$ must be disjoint from the range of all other restricted functions in $F$ and from $B$.

If these conditions are met, we say that $C$ is freely generated from $B$ by $F$.

Recursion Theorem

Suppose $C$ is freely generated from $B$ by $F$. Suppose also that $V$ is a set and $h$ is a function from $B$ to $V$. Suppose further that for each function $f : U^n \rightarrow U$ in $F$, there is a corresponding function $\overline{f} : V^n \rightarrow V$. Then there exists a unique function $\overline{h} : C \rightarrow V$ such that

- for $x \in B$, $\overline{h}(x) = h(x)$, and
- for each $f : U^n \rightarrow U$ in $F$ and $x_1, \ldots, x_n \in C$, $\overline{h}(f(x_1, \ldots, x_n)) = \overline{f}(\overline{h}(x_1), \ldots, \overline{h}(x_n))$. 
Recursion

Given $C$ freely generated from $B$ by $F$, show that $h : B \to V$ and $\overline{f} : V^n \to V$ for each $f : U^n \to U$ in $F$ determine a unique function $\overline{h} : C \to V$.

Proof sketch

A function $g : D \to E$ is called acceptable if $D \subseteq C$, $E \subseteq V$, and

- for $x \in B \cap D$, $g(x) = h(x)$, and
- for each $f : U^n \to U$ in $F$ and $x_1, \ldots, x_n \in C$, if $f(x_1, \ldots, x_n) \in D$, then $x_1, \ldots, x_n \in D$ and $g(f(x_1, \ldots, x_n)) = \overline{f}(g(x_1), \ldots, g(x_n))$.

Let $K$ be the collection of all acceptable functions, and let $\overline{h}$ be the union of $K$. Then $\overline{h}$ meets our requirements. Specifically,

- $\overline{h}$ is a function.
- $\overline{h}$ is an acceptable function.
- The domain of $\overline{h}$ is all of $C$.
- $\overline{h}$ is unique.

Details omitted (requires repeated use of induction principle).
Propositional Logic: Semantics

Intuitively, given a wff $\alpha$ and a value (either true or false) for each propositional symbol in $\alpha$, we should be able to determine the value of $\alpha$.

How do we make this precise?

Let $v$ be a function from $B$ to $\{0, 1\}$, where 0 represents false and 1 represents true. Recall that in the inductive definition of wff’s, $B$ contains the propositional symbols.

Now, we define $\overline{v}$, a function from $W$ to $\{0, 1\}$ as follows

- For each propositional symbol $A_i$, $\overline{v}(A_i) = v(A_i)$.
- $\overline{v}(\mathcal{E}_{\neg}(\alpha)) = 1 - \overline{v}(\alpha)$
- $\overline{v}(\mathcal{E}_{\land}(\alpha, \beta)) = \min(\overline{v}(\alpha), \overline{v}(\beta))$
- $\overline{v}(\mathcal{E}_{\lor}(\alpha, \beta)) = \max(\overline{v}(\alpha), \overline{v}(\beta))$
- $\overline{v}(\mathcal{E}_{\rightarrow}(\alpha, \beta)) = \max(1 - \overline{v}(\alpha), \overline{v}(\beta))$
- $\overline{v}(\mathcal{E}_{\leftrightarrow}(\alpha, \beta)) = 1 - |\overline{v}(\alpha) - \overline{v}(\beta)|$

The recursion theorem guarantees that $\overline{v}$ is well-defined.
Propositional Logic: Semantics

Intuitively, given a \textit{wff} \( \alpha \) and a value (either \textit{true} or \textit{false}) for each propositional symbol in \( \alpha \), we should be able to determine the value of \( \alpha \).

How do we make this precise?

Let \( \nu \) be a function from \( B \) to \( \{0, 1\} \), where 0 represents \textit{false} and 1 represents \textit{true}. Recall that in the inductive definition of \textit{wff}’s, \( B \) contains the propositional symbols.

Now, we define \( \overline{\nu} \), a function from \( W \) to \( \{0, 1\} \) as follows

- For each propositional symbol \( A_i \), \( \overline{\nu}(A_i) = \nu(A_i) \).
- \( \overline{\nu}(E_\neg(\alpha)) = 1 - \overline{\nu}(\alpha) \)
- \( \overline{\nu}(E_\land(\alpha, \beta)) = \min(\overline{\nu}(\alpha), \overline{\nu}(\beta)) \)
- \( \overline{\nu}(E_\lor(\alpha, \beta)) = \max(\overline{\nu}(\alpha), \overline{\nu}(\beta)) \)
- \( \overline{\nu}(E_\rightarrow(\alpha, \beta)) = \max(1 - \overline{\nu}(\alpha), \overline{\nu}(\beta)) \)
- \( \overline{\nu}(E_\leftrightarrow(\alpha, \beta)) = 1 - |\overline{\nu}(\alpha) - \overline{\nu}(\beta)| \)

The recursion theorem guarantees that \( \overline{\nu} \) is well-defined... under what conditions?
Propositional Logic: Semantics

Unique Readability Theorem

Given our inductive definition of the set $W$ of wff’s, $W$ is freely generated from $B$ by $F$. Specifically, the restriction of each operation in $F$ to $W$ is one-to-one and has a range disjoint from the range of the other restricted operations in $F$ and from $B$.

First we need the following lemma.

Lemma

Any proper initial segment of a wff contains an excess of left parentheses, and is therefore not a wff.

Proof

Let $S$ be the set of wff’s which have this property. We will show that $S$ is inductive. First note that the elements of $B$ all consist of a single symbol so it is not possible to construct a proper initial segment of any of them. Thus, $B \subseteq S$ vacuously.
Propositional Logic: Semantics

Proof, continued

To show that $S$ is closed under $E_\wedge$, suppose that $\alpha, \beta \in S$ and consider a proper initial segment of $E_\wedge(\alpha, \beta)$. There are 6 possibilities:

- $(\alpha_0)$, where $\alpha_0$ is a proper initial segment of $\alpha$.
- $(\alpha)$
- $(\alpha \wedge$
- $(\alpha \wedge \beta_0)$, where $\beta_0$ is a proper initial segment of $\beta_0$.
- $(\alpha \wedge \beta$

By using the inductive hypothesis and the fact (proved earlier) that all wff’s have the same number of left and right parentheses, each of these cases can be seen to have more left parentheses than right. Thus, $E_\wedge(\alpha, \beta) \in S$.

The cases for $E_\neg$, $E_\lor$, $E_\rightarrow$, and $E_\leftrightarrow$ are similar.
Propositional Logic: Semantics

Unique Readability Theorem

Given our inductive definition of the set $W$ of wff’s, $W$ is freely generated from $B$ by $F$. Specifically, the restriction of each operation in $F$ to $W$ is one-to-one and has a range disjoint from the range of the other restricted operations in $F$ and from $B$.

Proof

To show that the operation $E_\wedge$ restricted to $W$ is one-to-one, suppose that $(\alpha \wedge \beta) = (\gamma \wedge \delta)$, where $\alpha$, $\beta$, $\gamma$, and $\delta$ are wff’s.

Since both start with a left parenthesis, it follows that $\alpha \wedge \beta) = \gamma \wedge \delta)$. Since $\alpha$ and $\gamma$ are wff’s, the previous lemma implies that neither one can be a prefix of the other, and thus $\alpha = \gamma$. The same argument then shows that $\beta = \delta$.

Similar arguments can be applied to show that the other operations are one-to-one and that their ranges are all disjoint.
An Algorithm for Recognizing WFF’s

Lemma

Let $\alpha$ be a $wff$. Then exactly one of the following is true.

- $\alpha$ is a propositional symbol.
- $\alpha = (\neg \beta)$ where $\beta$ is a $wff$.
- $\alpha = (\beta \circ \gamma)$ where $\circ$ is one of $\{\land, \lor, \to, \leftrightarrow\}$, $\beta$ is the first parentheses-balanced initial segment of the result of dropping the first ( from $\alpha$, and $\beta$ and $\gamma$ are $wff$’s.

Proof is left as an exercise.
An Algorithm for Recognizing WFF’s

Lemma

Let $\alpha$ be a wff. Then exactly one of the following is true.

- $\alpha$ is a propositional symbol.
- $\alpha = (\neg \beta)$ where $\beta$ is a wff.
- $\alpha = (\beta \odot \gamma)$ where $\odot$ is one of $\{\land, \lor, \rightarrow, \leftrightarrow\}$, $\beta$ is the first parentheses-balanced initial segment of the result of dropping the first ( from $\alpha$, and $\beta$ and $\gamma$ are wff’s.

Proof is left as an exercise.
An Algorithm for Recognizing WFF’s

Input: expression $\alpha$ Output: true or false (indicating whether $\alpha$ is a wff).

0. Begin with an initial construction tree $T$ containing a single node labeled with $\alpha$.

1. If all leaves of $T$ are labeled with propositional symbols, return true.

2. Select a leaf labeled with an expression $\alpha_1$ which is not a propositional symbol.

3. If $\alpha_1$ does not begin with $($ return false.

4. Examine $\alpha_1$ to find $\beta$, the first balanced proper initial segment of $\alpha_1$. If there is no such $\beta$, return false.

5. If $\alpha_1 = (\neg \beta)$, then add a child to the leaf labeled by $\alpha_1$, label it with $\beta$, and goto 1.

6. If $\alpha_1 = (\beta \circ \gamma)$ where $\circ$ is one of \{ $\land, \lor, \rightarrow, \leftrightarrow$ \} and $\beta$ is balanced, then add two children to the leaf labeled by $\alpha_1$, label them with $\beta$ and $\gamma$, and goto 1.

7. Return false.
An Algorithm for Recognizing WFF’s

Termination

How do we prove termination of this algorithm?
An Algorithm for Recognizing WFF’s

Termination

How do we prove termination of this algorithm?

We can show that the sum of the lengths of all the expressions labeling leaves decreases on each iteration of the loop.
An Algorithm for Recognizing WFF’s

Termination

How do we prove termination of this algorithm?

We can show that the sum of the lengths of all the expressions labeling leaves decreases on each iteration of the loop.

Soundness

If the algorithm returns $true$ when given input $\alpha$, then $\alpha$ is a wff.

The proof is by induction on the tree $T$ generated by the algorithm from the leaves up to the root.
An Algorithm for Recognizing WFF’s

Termination

How do we prove termination of this algorithm?

We can show that the sum of the lengths of all the expressions labeling leaves decreases on each iteration of the loop.

Soundness

If the algorithm returns true when given input $\alpha$, then $\alpha$ is a wff.

The proof is by induction on the tree $T$ generated by the algorithm from the leaves up to the root.

Completeness

If $\alpha$ is a wff, then the algorithm will return true.

Proof using the induction principle for the set of wff’s.
Notational Conventions

- Larger variety of propositional symbols: $A, B, C, D, p, q, r$, etc.
- Outermost parentheses can be omitted: $A \land B$ instead of $(A \land B)$.
- Negation symbol binds stronger than binary connectives and its scope is as small as possible: $\neg A \land B$ means $(\neg A) \land B$.
- $\{\land, \lor\}$ bind stronger than $\{\rightarrow, \leftrightarrow\}$: $A \land B \rightarrow \neg C \lor D$ is $(A \land B) \rightarrow ((\neg C) \lor D))$
- When one symbol is used repeatedly, grouping is to the right: $A \land B \land C$ is $(A \land (B \land C))$

Note that conventions are only unambiguous for $wff$'s, not for arbitrary expressions.
Propositional Logic: Semantics

Well-Formed Formulas $W$

- $U = \text{all expressions}$
- $B = \{A_1, A_2, A_3, \ldots\}$
- $F = \{\mathcal{E}_-, \mathcal{E}_\wedge, \mathcal{E}_\vee, \mathcal{E}_\rightarrow, \mathcal{E}_\leftrightarrow\}$

Given a truth assignment function $\nu : B \rightarrow \{0, 1\}$ for the propositional symbols, we can construct a valuation function $\overline{\nu}$ for formulas in $W$ as follows

- For $\alpha \in B$, $\overline{\nu}(\alpha) = \nu(\alpha)$.
- $\overline{\nu}(\mathcal{E}_-(\alpha)) = 1 - \overline{\nu}(\alpha)$
- $\overline{\nu}(\mathcal{E}_\wedge(\alpha, \beta)) = \min(\overline{\nu}(\alpha), \overline{\nu}(\beta))$
- $\overline{\nu}(\mathcal{E}_\vee(\alpha, \beta)) = \max(\overline{\nu}(\alpha), \overline{\nu}(\beta))$
- $\overline{\nu}(\mathcal{E}_\rightarrow(\alpha, \beta)) = \max(1 - \overline{\nu}(\alpha), \overline{\nu}(\beta))$
- $\overline{\nu}(\mathcal{E}_\leftrightarrow(\alpha, \beta)) = 1 - |\overline{\nu}(\alpha) - \overline{\nu}(\beta)|$

The recursion theorem and the unique readability theorem guarantee that $\overline{\nu}$ is well-defined.
Truth Tables

There are other ways to present the semantics which are less formal but perhaps more intuitive.

\[
\begin{array}{c|c|c|c}
\alpha & \neg\alpha & \alpha \land \beta \\
\hline
1 & 1 & 1 \\
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{c|c|c|c}
\alpha & \beta & \alpha \lor \beta \\
\hline
1 & 1 & 1 \\
1 & 0 & 1 \\
0 & 1 & 1 \\
0 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{c|c|c|c}
\alpha & \beta & \alpha \rightarrow \beta \\
\hline
1 & 1 & 1 \\
1 & 0 & 1 \\
0 & 1 & 0 \\
0 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{c|c|c|c}
\alpha & \beta & \alpha \leftrightarrow \beta \\
\hline
1 & 1 & 1 \\
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 0 \\
\end{array}
\]
There are other ways to present the semantics which are less formal but perhaps more intuitive.

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>$\neg \alpha$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>$\beta$</th>
<th>$\alpha \land \beta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>$\beta$</th>
<th>$\alpha \lor \beta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>$\beta$</th>
<th>$\alpha \rightarrow \beta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\alpha$</th>
<th>$\beta$</th>
<th>$\alpha \leftrightarrow \beta$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Truth Tables

There are other ways to present the semantics which are less formal but perhaps more intuitive.

\[
\begin{array}{cc}
\alpha & \neg \alpha \\
1 & 0 \\
0 & 1
\end{array}
\quad
\begin{array}{ccc}
\alpha & \beta & \alpha \land \beta \\
1 & 1 & 1 \\
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 0
\end{array}
\]

\[
\begin{array}{ccc}
\alpha & \beta & \alpha \lor \beta \\
1 & 1 & 1 \\
1 & 0 & 1 \\
0 & 1 & 1 \\
0 & 0 & 0
\end{array}
\quad
\begin{array}{ccc}
\alpha & \beta & \alpha \rightarrow \beta \\
1 & 1 & 1 \\
1 & 0 & 1 \\
0 & 1 & 0 \\
0 & 0 & 0
\end{array}
\quad
\begin{array}{ccc}
\alpha & \beta & \alpha \leftrightarrow \beta \\
1 & 1 & 1 \\
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 0
\end{array}
\]
Truth Tables

There are other ways to present the semantics which are less formal but perhaps more intuitive.

\[
\begin{array}{c|c}
\alpha & \neg \alpha \\
1 & 0 \\
0 & 1 \\
\end{array}
\]

\[
\begin{array}{c|c|c}
\alpha & \beta & \alpha \land \beta \\
1 & 1 & 1 \\
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{c|c|c}
\alpha & \beta & \alpha \lor \beta \\
1 & 1 & 1 \\
1 & 0 & 1 \\
0 & 1 & 1 \\
0 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{c|c|c}
\alpha & \beta & \alpha \rightarrow \beta \\
1 & 1 & 1 \\
1 & 0 & 0 \\
0 & 1 & 1 \\
0 & 0 & 1 \\
\end{array}
\]

\[
\begin{array}{c|c|c}
\alpha & \beta & \alpha \leftrightarrow \beta \\
1 & 1 & 1 \\
1 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 1 \\
\end{array}
\]
Complex truth tables

Truth tables can also be used to calculate all possible values of $\overline{v}$ for a given wff: We associate a column with each propositional symbol and a column with each propositional connective. There is a row for each possible truth assignment to the propositional connectives.

<table>
<thead>
<tr>
<th>$A_1$</th>
<th>$A_2$</th>
<th>$A_3$</th>
<th>$(A_1 \lor (A_2 \land \overline{A_3}))$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Complex truth tables

Truth tables can also be used to calculate all possible values of $\bar{\nu}$ for a given wff: We associate a column with each propositional symbol and a column with each propositional connective. There is a row for each possible truth assignment to the propositional connectives.

<table>
<thead>
<tr>
<th>$A_1$</th>
<th>$A_2$</th>
<th>$A_3$</th>
<th>$(A_1 \lor (A_2 \land \neg A_3))$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Complex truth tables

Truth tables can also be used to calculate all possible values of $\bar{v}$ for a given $wff$: We associate a column with each propositional symbol and a column with each propositional connective. There is a row for each possible truth assignment to the propositional connectives.

<table>
<thead>
<tr>
<th>$A_1$</th>
<th>$A_2$</th>
<th>$A_3$</th>
<th>$(A_1 \lor (A_2 \land \neg A_3))$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1 1 0 0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1 1 1 1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1 0 0 0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1 0 0 1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0 1 0 0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0 1 1 1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0 0 0 0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0 0 0 1</td>
</tr>
</tbody>
</table>
Complex truth tables

Truth tables can also be used to calculate all possible values of \( \overline{\neg} \overline{\neg} \) for a given wff: We associate a column with each propositional symbol and a column with each propositional connective. There is a row for each possible truth assignment to the propositional connectives.

<table>
<thead>
<tr>
<th>A_1</th>
<th>A_2</th>
<th>A_3</th>
<th>(A_1 \lor (A_2 \land \neg A_3))</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1 1 1 1 0 0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1 1 1 1 1 1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1 1 1 0 0 0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1 1 0 0 1 1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0 0 1 0 0 1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0 1 1 1 1 1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0 0 0 0 0 0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0 0 0 0 0 1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0 0 0 0 0 1</td>
</tr>
</tbody>
</table>
Definitions

If $\alpha$ is a wff, then a truth assignment $\nu$ satisfies $\alpha$ if $\nu(\alpha) = 1$. 
Definitions

If $\alpha$ is a wff, then a truth assignment $\nu$ satisfies $\alpha$ if $\nu(\alpha) = 1$.

A wff $\alpha$ is satisfiable if there exists some truth assignment $\nu$ which satisfies $\alpha$. 