Review

Last week

Induction and Recursion

Well-Formed Formulas (wfs)

Propositional logic: Syntax

Recall our inductive definition of the set $W$ of well-formed formulas in

Propositional logic: Well-Formed Formulas
Recall our inductive definition of the set $W$ of well-formed formulas in propositional logic. Given the alphabet $\{ (, ), \land, \lor, \rightarrow, \neg, = \},$

- $\alpha$ and $\beta$ are WFFs.
- The set of expressions consisting of a single propositional symbol.
- The set of expressions consisting of a single propositional symbol.
- The set of all expressions over the alphabet.

Recall our inductive definition of the set $W$ of well-formed formulas in propositional logic. Given the alphabet $\{ (, ), \land, \lor, \rightarrow, \neg, = \},$

- $\alpha$ and $\beta$ are WFFs.
- The set of expressions consisting of a single propositional symbol.
- The set of expressions consisting of a single propositional symbol.
- The set of all expressions over the alphabet.

Recall our inductive definition of the set $W$ of well-formed formulas in propositional logic. Given the alphabet $\{ (, ), \land, \lor, \rightarrow, \neg, = \},$

- $\alpha$ and $\beta$ are WFFs.
- The set of expressions consisting of a single propositional symbol.
- The set of expressions consisting of a single propositional symbol.
- The set of all expressions over the alphabet.

Recall our inductive definition of the set $W$ of well-formed formulas in propositional logic. Given the alphabet $\{ (, ), \land, \lor, \rightarrow, \neg, = \},$

- $\alpha$ and $\beta$ are WFFs.
- The set of expressions consisting of a single propositional symbol.
- The set of expressions consisting of a single propositional symbol.
- The set of all expressions over the alphabet.
An Algorithm for Recognizing WFFs

Lemma

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

1. is a propositional symbol.
2. \( = (\ldots)\) where is a wff.
3. \( = (\ldots)\) where is one of \(\land, \lor, \rightarrow, \leftrightarrow, \neg\), and \(g, h\) are wffs.
4. \( = (\ldots)\) where \(g, h\) are wffs.
5. \( = (\ldots)\) where \(g, h\) are wffs.

How would you prove this?

Induction, of course!

How do we prove termination of this algorithm?

Termination

An Algorithm for Recognizing WFFs

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

0. Begin with an initial construction tree \(T\) containing a single node labeled with \(\bot\).
1. If all leaves of are labeled with propositional symbols, return true.
2. Select a leaf labeled with an expression \(1\) which is not a propositional symbol.
3. If \(1\) does not begin with \(\neg\), return false.
4. If \(1 = (\ldots)\) where \(g\) is a nonempty expression having the same number of left and right parentheses, return false.
5. If there is no such \(g\), return false.
6. If \(1 = (\ldots)\) where \(\downarrow\) is one of \(\land, \lor, \rightarrow, \leftrightarrow, \neg\), and \(g, h\) are wffs, then add two children to the only child of the leaf labeled by \(1\), label them with \(g\) and \(h\), and go to 1.
7. Return false.

Soundness

If the algorithm returns true when given input \(\neg\), then \(\neg\) 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 \(\neg\) is a wff, then the algorithm will return true. Proof using the induction principle for the set of wffs.
An Algorithm for Recognizing WFFs

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 wffs.

Notational Conventions

Larger variety of propositional symbols: \( A, B, C, D, p, q, r \), etc.

Outermost parentheses can be omitted: \( \neg(A \land B) \) instead of \( (\neg(A \land B)) \).

When one symbol is used repeatedly, grouping is to the right: \( A \land B \lor C \).

Negation symbol binds stronger than binary connectives and its scope is as small as possible: \( \neg(A \lor B) \) instead of \( \neg(A \lor B) \).

\( f^g \) binds stronger than \( f!_g \): \( A \lor (B \land C) \) instead of \( A \lor (B \land C) \).

f^g;_g bind stronger than \( f^g \): \( A \lor (B \land C) \).

When one symbol is used repeatedly, grouping is to the right: \( A \land B \lor C \).
Propositional Logic: Semantics

Intuitively, given a well-formed formula (wff) and a value (either T or F) for each propositional symbol, we should be able to determine the value of the entire formula. How do we make this precise?

Let $v$ be a function from $B$ to $\{T, F\}$. We call this function a truth assignment.

Now, we define $v$, a function from $W$ to $\{T, F\}$, as follows (we compute with $T$ and $F$ as if they were 1 and 0, respectively).

For each propositional symbol $A$, $v(A) = T$.

Let $v$ be a function from $\{T, F\}$ to $\{T, F\}$. We call this function a truth assignment.

How do we make this precise?

We should be able to determine the value of a well-formed formula (wff) based on the truth values of its propositional symbols.

Truth Tables

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

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

Truth tables can also be used to calculate all possible values of \( v \) 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. Truth tables can also be used to calculate all possible values of \( v \) for a given wff.

<table>
<thead>
<tr>
<th>( A_1 )</th>
<th>( A_2 )</th>
<th>( A_3 )</th>
<th>( \neg(A_1 \land (A_2 \lor A_3)) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

More intuitively, there are other ways to present the semantics which are less formal but perhaps more intuitive.
Complex truth tables

Truth tables can also be used to calculate all possible values of 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>$\neg A_3$</th>
<th>$A_2 \lor \neg A_3$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

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

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.

$\alpha$ is a wff. Then a truth assignment $v$ satisfies $\alpha$ if $v(\alpha) = T$.

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

Suppose $\Sigma$ is a set of wffs. Then $\Sigma$ tautologically implies $\alpha$, if every truth assignment which satisfies each formula in $\Sigma$ also satisfies $\alpha$.

Particular cases:

- If $\alpha$ is satisfiable, then we say $\alpha$ is a tautology or is valid and write $\alpha = \top$. If $\alpha$ is unsatisfiable, then $\alpha = \bot$ for every wff.
- If $\alpha$ and $\beta$ are tautologically equivalent, then $\alpha = \beta$. For an infinite $\Sigma$, $\alpha = \top$ if and only if $V(\alpha) = \top$. 

12-c

12-b

Definitions

- If $\alpha$ is a wff, then a truth assignment $v$ satisfies $\alpha$ if $v(\alpha) = T$.
- A wff $\alpha$ is satisfiable if there exists some truth assignment $v$ which satisfies $\alpha$.
- Suppose $\Sigma$ is a set of wffs. Then $\Sigma$ tautologically implies $\alpha$, if every truth assignment which satisfies each formula in $\Sigma$ also satisfies $\alpha$.

Particular cases:

- If $\alpha$ is satisfiable, then we say $\alpha$ is a tautology or is valid and write $\alpha = \top$. If $\alpha$ is unsatisfiable, then $\alpha = \bot$ for every wff.
- If $\alpha$ and $\beta$ are tautologically equivalent, then $\alpha = \beta$. For an infinite $\Sigma$, $\alpha = \top$ if and only if $V(\alpha) = \top$. 

12-c

12-b

Definitions
For a finite \( \mathcal{A} \), and only if \( \mathcal{A} \) is valid:

\[
\forall x \ ( \mathcal{A} ) \text{ if and only if } \mathcal{A} \text{ is valid.}
\]

\[\text{Logical equivalence.}\]

Suppose \( \mathcal{A} \) is a set of wffs. Then \( \mathcal{A} \) logically implies \( \mathcal{B} \), if and only if every wff \( \mathcal{A} \) satisfies \( \mathcal{B} \).

A wff is satisfiable if there exists some truth assignment \( v \) which satisfies \( \mathcal{A} \) if \( \mathcal{A} = \{ v \} \).

\[\text{If } \mathcal{A} \text{ is a wff, then a truth assignment } v \text{ satisfies } \mathcal{A} \text{ if } v(\mathcal{A}) = T.\]

\[\text{Particular cases:}\]

Suppose \( \mathcal{A} \) is an assignment which satisfies each formula in \( \mathcal{A} \) and also satisfies \( \mathcal{A} \).

If \( \mathcal{A} \) is a wff, then a truth assignment which satisfies \( \mathcal{A} \) if \( \mathcal{A} = \{ v \} \).

\[\text{If } \mathcal{A} \text{ is a wff, then } v \text{ is a truth assignment of } \mathcal{A} \text{ if } v(\mathcal{A}) = T.\]

\[\text{If } \mathcal{A} \text{ is a wff, then a truth assignment of } \mathcal{A} \text{ if } v(\mathcal{A}) = T.\]

\[\text{If } \mathcal{A} \text{ is a wff, then } v \text{ is a truth assignment of } \mathcal{A} \text{ if } v(\mathcal{A}) = T.\]
verify each of the claims made above?

If true, the formula is satisfiable; otherwise, it is unsatisfiable.

Suppose you had an algorithm that would take a SAT as input and return the following wff which is satisfiable, but not valid.

\[ (A \lor B) \land (\neg A \lor \neg B) \]

Examples:

- \[ (A \lor B) \land (\neg A \lor \neg B) \] is satisfiable, but not valid.
- \[ (A \lor B) \land (\neg A \lor \neg B) \land (A \rightarrow B) \] is satisfiable, but not valid.
- \[ (A \lor B) \land (\neg A \lor \neg B) \land (A \rightarrow \neg B) \] is unsatisfiable.
Verify each of the claims made above.

Suppose you had an algorithm which would take a \( wff \) as input and return \( \text{SAT} \) if it is satisfiable and \( \text{not sat} \) otherwise. How would you use this algorithm to verify each of the claims made above?

\[
\begin{align*}
G \land \neg A & \iff \text{is logically equivalent to} \quad (\neg A \lor \neg G) \iff (\neg G) \lor (\neg A) \\
(\neg A \lor \neg G) & \iff \{ \neg A \lor \neg G \} \\
G & \iff (G \leftrightarrow G) \\
(G \leftrightarrow G) \lor (G \land \neg G) & \iff (G \land \neg G) \lor (G \land \neg G) \\
(G \land \neg G) & \iff \text{is unsatisfiable.} \\
(G \land \neg G) \land (G \land \neg G) & \iff \text{is satisfiable, but not valid.}
\end{align*}
\]

**Examples**

\[
\begin{align*}
G \land \neg A & \iff \text{is logically equivalent to} \quad (\neg A \lor \neg G) \iff (\neg G) \lor (\neg A) \\
(\neg A \lor \neg G) & \iff \{ \neg A \lor \neg G \} \\
G & \iff (G \leftrightarrow G) \\
(G \leftrightarrow G) \lor (G \land \neg G) & \iff (G \land \neg G) \lor (G \land \neg G) \\
(G \land \neg G) & \iff \text{is unsatisfiable.} \\
(G \land \neg G) \land (G \land \neg G) & \iff \text{is satisfiable, but not valid.}
\end{align*}
\]

**Examples**
Satisfiability and validity are dual notions. A is unsatisfiable if and only if \( A \) is valid.

Valid and false are opposite. How would you verify the claims given this algorithm?

Now suppose you had an algorithm \texttt{CHECKVALID} which returns \texttt{true} when \( A \) is valid and \texttt{false} otherwise. How would you use this algorithm to verify each of the claims made above?

**Examples**

Suppose you had an algorithm \texttt{CHECKVALID} which returns \texttt{true} when \( A \) is valid and \texttt{false} otherwise. How would you use this algorithm to verify each of the claims made above?

**Examples**

\[
((\neg A \land \neg A) \leftrightarrow (\neg A \lor \neg A))
\]

is logically equivalent to \( (\neg A \lor \neg A) \).

\[
((\neg A \lor \neg A) \lor (\neg A \lor \neg A)) \equiv \{\neg A \lor \neg A\}
\]

\[
((\neg A \lor (A \lor A)) \lor A) \equiv (A \lor A)
\]

\( A \) is satisfiable, but not valid.

\[
((\neg A \land \neg A) \leftrightarrow (\neg A \lor \neg A))
\]

is logically equivalent to \( (\neg A \lor \neg A) \).

\[
((\neg A \lor \neg A) \lor (\neg A \lor \neg A)) \equiv \{\neg A \lor \neg A\}
\]

\[
((\neg A \lor (A \lor A)) \lor A) \equiv (A \lor A)
\]

\( A \) is satisfiable, but not valid.
Determining Satisfiability using Truth Tables

An Algorithm for Satisfiability

To check whether is satisfiable, form the truth table for . If there is a row in which appears as the value for , then is satisfiable. Otherwise, is unsatisfiable.

An Algorithm for Tautological Implication

To check whether , check the satisfiability of

If it is unsatisfiable, then .

Otherwise .
### Determining Satisfiability using Truth Tables

**Example**

\[
A \land (B \land (C \lor (\neg A \land \neg B))) \lor (\neg A \land C)
\]

<table>
<thead>
<tr>
<th>(A)</th>
<th>(B)</th>
<th>(C)</th>
<th>(S)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(T)</td>
<td>(F)</td>
<td>(T)</td>
<td>(T)</td>
</tr>
<tr>
<td>(T)</td>
<td>(F)</td>
<td>(T)</td>
<td>(T)</td>
</tr>
<tr>
<td>(T)</td>
<td>(F)</td>
<td>(F)</td>
<td>(F)</td>
</tr>
<tr>
<td>(T)</td>
<td>(T)</td>
<td>(T)</td>
<td>(F)</td>
</tr>
<tr>
<td>(T)</td>
<td>(T)</td>
<td>(F)</td>
<td>(T)</td>
</tr>
<tr>
<td>(T)</td>
<td>(F)</td>
<td>(F)</td>
<td>(T)</td>
</tr>
<tr>
<td>(T)</td>
<td>(F)</td>
<td>(F)</td>
<td>(T)</td>
</tr>
<tr>
<td>(T)</td>
<td>(F)</td>
<td>(T)</td>
<td>(T)</td>
</tr>
<tr>
<td>(T)</td>
<td>(T)</td>
<td>(T)</td>
<td>(F)</td>
</tr>
<tr>
<td>(T)</td>
<td>(T)</td>
<td>(F)</td>
<td>(T)</td>
</tr>
<tr>
<td>(T)</td>
<td>(F)</td>
<td>(F)</td>
<td>(T)</td>
</tr>
<tr>
<td>(T)</td>
<td>(T)</td>
<td>(F)</td>
<td>(T)</td>
</tr>
</tbody>
</table>

### Determining Satisfiability using Truth Tables

**Example**

\[
A \land (B \land (C \lor (\neg A \land \neg B))) \lor (\neg A \land C)
\]

<table>
<thead>
<tr>
<th>(A)</th>
<th>(B)</th>
<th>(C)</th>
<th>(S)</th>
</tr>
</thead>
<tbody>
<tr>
<td>(T)</td>
<td>(F)</td>
<td>(T)</td>
<td>(T)</td>
</tr>
<tr>
<td>(T)</td>
<td>(F)</td>
<td>(T)</td>
<td>(T)</td>
</tr>
<tr>
<td>(T)</td>
<td>(F)</td>
<td>(F)</td>
<td>(F)</td>
</tr>
<tr>
<td>(T)</td>
<td>(T)</td>
<td>(T)</td>
<td>(F)</td>
</tr>
<tr>
<td>(T)</td>
<td>(T)</td>
<td>(F)</td>
<td>(T)</td>
</tr>
<tr>
<td>(T)</td>
<td>(F)</td>
<td>(F)</td>
<td>(T)</td>
</tr>
<tr>
<td>(T)</td>
<td>(F)</td>
<td>(F)</td>
<td>(T)</td>
</tr>
<tr>
<td>(T)</td>
<td>(F)</td>
<td>(T)</td>
<td>(T)</td>
</tr>
<tr>
<td>(T)</td>
<td>(T)</td>
<td>(T)</td>
<td>(F)</td>
</tr>
<tr>
<td>(T)</td>
<td>(T)</td>
<td>(F)</td>
<td>(T)</td>
</tr>
<tr>
<td>(T)</td>
<td>(F)</td>
<td>(F)</td>
<td>(T)</td>
</tr>
<tr>
<td>(T)</td>
<td>(T)</td>
<td>(F)</td>
<td>(T)</td>
</tr>
</tbody>
</table>
Determining Satisfiability using Truth Tables

<table>
<thead>
<tr>
<th>Example</th>
<th>$A \wedge ((B \vee \neg A) \wedge (C \vee \neg B))$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>$B$</td>
</tr>
<tr>
<td>$F$</td>
<td>$F$</td>
</tr>
<tr>
<td>$F$</td>
<td>$F$</td>
</tr>
<tr>
<td>$F$</td>
<td>$T$</td>
</tr>
<tr>
<td>$T$</td>
<td>$F$</td>
</tr>
<tr>
<td>$F$</td>
<td>$F$</td>
</tr>
<tr>
<td>$F$</td>
<td>$T$</td>
</tr>
<tr>
<td>$T$</td>
<td>$F$</td>
</tr>
</tbody>
</table>

Determining Satisfiability using Truth Tables

<table>
<thead>
<tr>
<th>Example</th>
<th>$A \wedge ((B \vee \neg A) \wedge (C \vee \neg B))$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>$B$</td>
</tr>
<tr>
<td>$F$</td>
<td>$F$</td>
</tr>
<tr>
<td>$F$</td>
<td>$F$</td>
</tr>
<tr>
<td>$F$</td>
<td>$T$</td>
</tr>
<tr>
<td>$T$</td>
<td>$F$</td>
</tr>
<tr>
<td>$F$</td>
<td>$F$</td>
</tr>
<tr>
<td>$F$</td>
<td>$T$</td>
</tr>
<tr>
<td>$T$</td>
<td>$F$</td>
</tr>
</tbody>
</table>

Determining Satisfiability using Truth Tables

<table>
<thead>
<tr>
<th>Example</th>
<th>$A \wedge ((B \vee \neg A) \wedge (C \vee \neg B))$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>$B$</td>
</tr>
<tr>
<td>$F$</td>
<td>$F$</td>
</tr>
<tr>
<td>$F$</td>
<td>$F$</td>
</tr>
<tr>
<td>$F$</td>
<td>$T$</td>
</tr>
<tr>
<td>$T$</td>
<td>$F$</td>
</tr>
<tr>
<td>$F$</td>
<td>$F$</td>
</tr>
<tr>
<td>$F$</td>
<td>$T$</td>
</tr>
<tr>
<td>$T$</td>
<td>$F$</td>
</tr>
</tbody>
</table>

Determining Satisfiability using Truth Tables

<table>
<thead>
<tr>
<th>Example</th>
<th>$A \wedge ((B \vee \neg A) \wedge (C \vee \neg B))$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$A$</td>
<td>$B$</td>
</tr>
<tr>
<td>$F$</td>
<td>$F$</td>
</tr>
<tr>
<td>$F$</td>
<td>$F$</td>
</tr>
<tr>
<td>$F$</td>
<td>$T$</td>
</tr>
<tr>
<td>$T$</td>
<td>$F$</td>
</tr>
<tr>
<td>$F$</td>
<td>$F$</td>
</tr>
<tr>
<td>$F$</td>
<td>$T$</td>
</tr>
<tr>
<td>$T$</td>
<td>$F$</td>
</tr>
<tr>
<td>$F$</td>
<td>$F$</td>
</tr>
<tr>
<td>$T$</td>
<td>$F$</td>
</tr>
</tbody>
</table>
Determining Satisfiability using Truth Tables

What is the complexity of this algorithm?

$2^n$ where $n$ is the number of propositional symbols.

Can you think of a way to speed up these algorithms?

In an upcoming lecture, we will discuss some of the applications and best-known techniques for the SAT algorithm.
De Morgan's Laws

\[(B \lor \neg A) \leftrightarrow (B \land A) \neg \bullet\]

\[(B \land \neg A) \leftrightarrow (B \lor A) \neg \bullet\]

Negation

\[\neg(\neg A) \leftrightarrow A \lor A \neg \bullet\]

\[\neg(\neg A) \leftrightarrow A \land A \neg \bullet\]

Distributive Laws

\[A \land (B \lor C) \leftrightarrow (A \land B) \lor (A \land C) \neg \bullet\]

\[A \lor (B \land C) \leftrightarrow (A \lor B) \land (A \lor C) \neg \bullet\]

Some Tautologies

\[\neg A \leftrightarrow A \neg \bullet\]

\[A \leftrightarrow A \neg \bullet\]

Associative and Commutative Laws for \(\lor\) and \(\land\)

\[A \lor (B \lor C) \leftrightarrow (A \lor B) \lor C \neg \bullet\]

\[A \land (B \land C) \leftrightarrow (A \land B) \land C \neg \bullet\]
Implication: $(A \rightarrow B) \leftrightarrow (\neg A \lor B)$

Excluded Middle: $A \lor \neg A$

Contradiction: $A \land \neg A$

Contraposition: $(A \rightarrow B) \rightarrow (\neg B \rightarrow \neg A)$

Exportation: $(A \rightarrow (B \rightarrow C)) \rightarrow ((A 

\land \neg B) \lor C)$
Propositional Connectives

We have the connectives: \(\land, \lor, \neg, \Rightarrow, \Leftarrow\). Would we gain anything by having more? Would we lose anything by having fewer?

Example: Ternary Majority Connective

\[ (# (\land, \lor, \neg) ) = (# ) \quad \forall \quad (# (\land, \lor, \neg)) \iff \text{the majority of } v(1), v(2), v(3) \text{ are } T. \]

What does this new connective do for us?

Propositional Connectives

More Tautologies

\[ ((C \Rightarrow B) \Leftarrow \neg C \lor B) \]

Exportation

\[ (\neg (B \Rightarrow C)) \Rightarrow (C \Rightarrow B) \]

Composition

\[ (\neg (B \lor C)) \Rightarrow (B \Rightarrow \neg C) \]

Contradiction

\[ C \Rightarrow \neg \neg C \]

Excluded Middle

\[ (B \land \neg B) \Rightarrow (B \Rightarrow \neg B) \]

Implication
Propositional Connectives

We have various connectives:

\[ \land, \lor, \neg, \rightarrow, \leftrightarrow \]

Would we gain anything by having more? Would we lose anything by having fewer?

Example: Ternary Majority Connective

\[ \#(p, q, r) = \begin{cases} p & \text{if } p, q, r \text{ are } \top \\ q & \text{if } p, q, r \text{ are } \bot \\ \#(p, q, r) & \text{otherwise} \end{cases} \]

What does this new connective do for us?

We have two connectives: \( \land, \lor \)
We lose anything by having more.

Are \( \land \), \( \lor \) Ternary Majority Connective?

The following table shows the truth values of \( \#(p, q, r) \) for all possible combinations of \( p, q, r \):

<table>
<thead>
<tr>
<th>p</th>
<th>q</th>
<th>r</th>
<th>#(p, q, r)</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

Claim: The extended language obtained by allowing this new symbol has the same expressive power as the original language.
Realizing Boolean Functions

In general, suppose that \( \alpha \) is a wff whose propositional symbols are included in \( A_1; \ldots; A_n \). We define a \( n \)-place Boolean function \( B_n \) as

\[
B_n(X_1; \ldots; X_n) = \text{the truth value given to } \alpha \text{ when } A_1; \ldots; A_n \text{ are given the values } X_1; \ldots; X_n.
\]

In other words, \( B_n(X_1; \ldots; X_n) \) is a truth function of \( n \) arguments, and the choice of \( \alpha \) determines the function.

Claim: Every Boolean function can be obtained as a composition of \( I, N, K, A, C, \) and \( E \).

We will explain why this is true shortly.

Examples

\[
(((X \equiv X) \land (X \equiv X)) \land (X \equiv X)) \lor (X \equiv X) \equiv B
\]

From these functions, we can construct others by composition:

\[
\begin{align*}
\forall \alpha \forall \beta \exists \gamma & \equiv \alpha \\
\forall \alpha \forall \beta \exists \gamma & \equiv \alpha \\
\forall \alpha \forall \beta \forall \gamma & \equiv \alpha \\
\forall \alpha \forall \beta \exists \gamma & \equiv \alpha \\
\forall \alpha \forall \beta \forall \gamma & \equiv \alpha \\
\forall \alpha \forall \beta \exists \gamma & \equiv \alpha \\
\forall \alpha \forall \beta \forall \gamma & \equiv \alpha \\
\forall \alpha \forall \beta \exists \gamma & \equiv \alpha \\
\forall \alpha \forall \beta \forall \gamma & \equiv \alpha \\
\forall \alpha \forall \beta \exists \gamma & \equiv \alpha \\
\forall \alpha \forall \beta \forall \gamma & \equiv \alpha \\
\forall \alpha \forall \beta \exists \gamma & \equiv \alpha \\
\forall \alpha \forall \beta \forall \gamma & \equiv \alpha \\
\forall \alpha \forall \beta \exists \gamma & \equiv \alpha \\
\forall \alpha \forall \beta \forall \gamma & \equiv \alpha \\
\forall \alpha \forall \beta \exists \gamma & \equiv \alpha \\
\forall \alpha \forall \beta \forall \gamma & \equiv \alpha \\
\forall \alpha \forall \beta \exists \gamma & \equiv \alpha \\
\forall \alpha \forall \beta \forall \gamma & \equiv \alpha \\
\end{align*}
\]
Theorem

Formulas and the Boolean Functions They Realize

From these functions, we can construct others by composition.

\[
\begin{align*}
\mathcal{V} & = T \\
\mathcal{V} & = \neg T \\
\mathcal{V} & = \neg \mathcal{V} \\
\mathcal{V} & = \neg \mathcal{V} \\
\mathcal{V} & = \top \\
\mathcal{V} & = \bot
\end{align*}
\]

Examples
disjunctive normal form. Thus, a counterexample is that for every wff, there exists a logically equivalent wff in disjunctive normal form.

The negation of a wff is a wff, which is a conjunction of literals, where a literal is either a propositional symbol or its negation. A formula in DNF is also a disjunction of formulas, each of which is a conjunction of literals. It follows that each disjunctive normal form (DNF) formula is in so-called disjunctive normal form.

We say that this set of connectives is complete if every Boolean function can be realized by a wff which uses only the connectives in this set. This shows that every Boolean function can be realized by a wff in fact, every function.

Proof continued

Completeness of Propositional Connectives

Let \( g \in \mathcal{G} \) be such that \( g \) realizes the function \( \varphi \). Let \( \mathcal{R} \) be any relation such that \( \mathcal{R} \) realizes the function \( \varphi \). Then let \( g = \varphi \). Clearly, \( g \) is a wff, such that

\[ \mathcal{R} = \varphi \]

Otherwise, \( \mathcal{R} = \varphi \). Somewhat surprisingly, \( \mathcal{R} = \varphi \). Now let \( \mathcal{R} \) be any relation such that \( \mathcal{R} \) realizes the function \( \varphi \). Let \( \mathcal{R} \) be any relation such that

\[ \mathcal{R} = \varphi \]
Using these identities, the completeness can be easily proved by induction.

In fact, we can do better. It turns out that additional connectives to a complete set may allow a function to be realized more concisely. Additional connectives to a complete set may allow a function to be realized more concisely.

Recall our definition of some basic Boolean functions:

<table>
<thead>
<tr>
<th>Operation</th>
<th>Bool</th>
<th>Operation</th>
<th>Bool</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND</td>
<td>T</td>
<td>OR</td>
<td>T</td>
</tr>
<tr>
<td>NOT</td>
<td>F</td>
<td>AND</td>
<td>T</td>
</tr>
<tr>
<td>OR</td>
<td>T</td>
<td>NOT</td>
<td>F</td>
</tr>
<tr>
<td>NOT</td>
<td>F</td>
<td>OR</td>
<td>T</td>
</tr>
</tbody>
</table>

There are four points at which \( G \) is true, so a DNF formula which realizes \( G \) is

\[
\begin{align*}
\text{Example} & = (\text{F}, \text{F}, \text{F}) \\
\text{Example} & = (\text{F}, \text{F}, \text{T}) \\
\text{Example} & = (\text{F}, \text{T}, \text{F}) \\
\text{Example} & = (\text{F}, \text{T}, \text{T}) \\
\text{Example} & = (\text{T}, \text{F}, \text{F}) \\
\text{Example} & = (\text{T}, \text{F}, \text{T}) \\
\text{Example} & = (\text{T}, \text{T}, \text{F}) \\
\text{Example} & = (\text{T}, \text{T}, \text{T})
\end{align*}
\]
Recall our definition of some basic Boolean functions:

- \( P' = B_{A_1} \)
- \( N = B_{A_1} \)
- \( K = B_{A_1 \land A_2} \)
- \( A = B_{A_1 \lor A_2} \)

Given that \( \{
eg, \land, \lor\} \) is complete, it is not hard to see that any Boolean function can be constructed using only the Boolean functions \( P', N, K, \) and \( A \).

In fact, we can do better. It turns out that \( \{\neg, \land, \lor\} \) are complete as well.

**Why?**

---

**Incompleteness of Connectives**

To prove that some set of connectives is incomplete, we find a property that is true of all wffs built using those connectives, but that is not true for some Boolean function.

Using these identities, the completeness can be easily proved by induction.
Incompleteness of Connectives

\[ J = (\langle I, J \rangle)_{\max} = ((\gamma)_{\max} \land (\alpha)_{\max} - I)_{\max} = (\land \leftarrow \gamma)_{\max} \]

Inductive Case

\[ J = (\langle I, J \rangle)_{\max} = ((\gamma)_{\max} \land (\alpha)_{\max})_{\max} = (\land \land \gamma)_{\max} \]

Base Case

\[ J = (\langle I, J \rangle)_{\max} = ((\gamma)_{\max})_{\max} = (\gamma)_{\max} \]

such that for all \( \gamma \), we prove by induction that let \( \alpha \) be a wff which uses only these connectives and let \( \alpha \) be a truth assignment.

Proof

\[ \{ \neg \land \} \]

Example

Incompleteness of Connectives

\[ J = (\langle I, J \rangle)_{\max} = ((\gamma)_{\max} \land (\alpha)_{\max} - I)_{\max} = (\land \leftarrow \gamma)_{\max} \]

such that for all \( \gamma \), we prove by induction that let \( \alpha \) be a wff which uses only these connectives and let \( \alpha \) be a truth assignment.

Proof

\[ \{ \neg \land \} \]

Example

Incompleteness of Connectives
There are sixteen 0-place Boolean functions. They are cataloged in the following table. Notethat the first six correspond to 0-ary and unary connectives.

**Unary Connectives**

- Negation: \( \neg \)
  - Truth values: \( T \rightarrow \neg T \), \( F \rightarrow \neg F \)
  - Construction: \( \neg \alpha \)

Why?

For each \( n \), there are \( 2^n \) different \( n \)-place Boolean functions.

**Binary Connectives**

- Identity: \( \alpha \)
- And: \( \land \)
  - Truth values: \( T \rightarrow T \land T = T \), \( F \rightarrow T \land F = F \)
  - Construction: \( \alpha \land \beta \)
- Or: \( \lor \)
  - Truth values: \( T \rightarrow T \lor T = T \), \( F \rightarrow T \lor F = T \)
  - Construction: \( \alpha \lor \beta \)
- Implication: \( \rightarrow \)
  - Truth values: \( T \rightarrow T \rightarrow T \), \( F \rightarrow T \rightarrow F = T \)
  - Construction: \( \alpha \rightarrow \beta \)
- Equivalence: \( \leftrightarrow \)
  - Truth values: \( T \rightarrow T \leftrightarrow T = T \), \( F \rightarrow T \leftrightarrow F = F \)
  - Construction: \( \alpha \leftrightarrow \beta \)

Other Propositional Connectives

- Construction: \( \chi_1 \), \( \chi_2 \), \( \ldots \), \( \chi_n \)
- Output values: \( \pm \), \( \mp \), \( 1 \), \( 0 \)
- Truth assignment: \( \{ T \rightarrow \chi_1, \ldots, F \rightarrow \chi_n \} \)

Inductive Case

Let \( \alpha \) be a \( wff \) which uses only these connectives, and let \( \delta \) be a truth assignment.

**Proof**

- Base Case: \( \alpha = \top \) or \( \alpha = \bot \) is not complete.

- Induction Case: Let \( \alpha = ( \Land \beta ) \lor ( \Land \gamma ) \), for all \( \alpha \).
- We prove by induction that \( \alpha \) is not complete.
- The set of \( wff \) built using those connectives, but that is not true for some Boolean function.
- To prove that some set of connectives is incomplete, we find a property that is not satisfied by all \( wff \).
Other Propositional Connectives

For each $n$, there are $2^{2^n}$ different $n$-place Boolean functions $B(X_1; \ldots; X_n)$.

Why?

For each $n$, there are $2^n$ different $n$-ary propositional connectives.

\[ B \] (or Sheffer stroke)

\[ \wedge \] (and)

\[ \vee \] (or)

\[ \neg \] (negation)

\[ = \] (equality)

\[ \neq \] (inequality)

\[ \leftrightarrow \] (if and only if)

\[ \rightarrow \] (implies)

Other Propositional Connectives

Recall that a wff $\alpha$ is satisfiable if there exists a truth assignment $\nu$ such that

\[ \nu(\alpha) = T \]

Compactness

A set of wffs is satisfiable if there exists a truth assignment $\nu$ such that $\nu(\alpha) = T$ for each $\alpha$.

A set is finitely satisfiable if every finite subset of the set is satisfiable.

Compactness Theorem

A set of wffs is satisfiable if and only if it is finitely satisfiable.

Why?

A set is satisfiable if and only if its corresponding set of truth assignments is non-empty.
Compactness

Recall that a wff is satisfiable if there exists a truth assignment $v$ such that $v(\varphi) = T$.

A set of wffs is satisfiable if there exists a truth assignment $v$ such that $v(\varphi_i) = T$ for each $\varphi_i \in \mathcal{S}$.

A set is finitely satisfiable if every finite subset of it is satisfiable.

Compactness Theorem

A set of wffs is satisfiable if and only if it is finitely satisfiable.

Proof

The only if direction is trivial since any subset of a satisfiable set is clearly satisfiable.

To prove the other direction, assume that $\mathcal{S}$ is a set which is finitely satisfiable. We must show that $\mathcal{S}$ is satisfiable.
Compactness

Recall that a wff is satisfiable if there exists a truth assignment \( v \) such that \( v(w) = T \). A set of wffs is satisfiable if there exists a truth assignment \( v \) such that \( \forall \alpha \in S, v(\alpha) = T \). A set is finitely satisfiable if every finite subset of \( S \) is satisfiable.

Compactness Theorem

A set of wffs is satisfiable if it is finitely satisfiable.

Proof

We must show that \( \models S \) is satisfiable.

To prove the other direction, assume that \( \not\models S \). Then, let \( \not\models S \).

The only if direction is trivial since any subset of a satisfiable set is clearly satisfiable.

Let \( \not\models S \) be a finite set of wffs.

If each \( \alpha \in S \), then there exists a truth assignment \( v \) such that \( v(\alpha) = T \).

A set of wffs is satisfiable if there exists a truth assignment \( v \) such that \( \forall \alpha \in S, v(\alpha) = T \).
Compactness

Let $\square \neg \Box \neg A$ be a satisfiable set. We extend it to a maximal $\square \neg \Box \neg$ set as follows.

Let $1, \ldots, n, \ldots$ be an enumeration of all $\text{wffs}$. Why is this possible? The set of all satisfiable sets is countable.

We claim that for any $\text{wff} A$, $A \in S$ if and only if $A$ is satisfiable. The proof is by induction on $A$.

\[ A \in S \iff \text{A is satisfiable}. \]

Compactness

Now we show that $\neg A \in S$ is satisfiable (and thus $\neg A \in S$ is also satisfiable).

Compactness

We claim that for any $\text{wff} A$, $A \in S$ if and only if $A$ is satisfiable. The proof is by induction on $A$.

\[ A \in S \iff \text{A is satisfiable}. \]

Compactness

Now we show that $\neg A \in S$ is satisfiable (and thus $\neg A \in S$ is also satisfiable).

Compactness

Now we show that $\neg A \in S$ is satisfiable (and thus $\neg A \in S$ is also satisfiable).
Corollary

If \( j \neq 0 \) then there is a finite \( A \) such that \( \exists a \in A \). 

Proof

Suppose that \( \exists a \neq 0 \) for every finite \( A \). Then, \( \exists a \neq 0 \) and \( \exists A \) is satisfiable for every finite \( A \).

\[ \exists a \neq 0 \]

By compactness, \( \exists A \) is satisfiable which contradicts the fact that \( j \neq 0 \).