Lecture 2
Fall 2008
G22.2390-001 Logic in Computer Science
Last week:

- Propositional Logic: Syntax
- Well-Formed Formulas (wffs)
- Induction and Recursion
Outline

Recognizing Well-Formed Formulas

Propositional Logic: Semantics

Truth Tables

Satisfiability and Tautologies

Propositional Connectives and Boolean Functions

Compactness

Sources:


Enderton, Sections: 1.2, 1.3, 1.5, 1.7.
Recall our inductive definition of the set \( W \) of well-formed formulas in propositional logic. Given the alphabet \( \{',',(,)',,\wedge,\vee,\neg,\to,\leftrightarrow,\exists,\forall,1,2,\cdots\} \), let

- \( = P \)
- \( R \)
- \( \cup \)

\( B \)

\( F \)

\( E \) : \( ( \) \)

\( E \) ^ \( , \)

\( E \) _ \( , \)

\( E \) ! \( , \)

\( E \) $ \( , \)
Recall our inductive definition of the set \( \mathcal{W} \) of well-formed formulas in propositional logic. Given the alphabet \( \{ 1, 2, \ldots \} \), \( \bigwedge \), \( \bigvee \), \( \rightarrow \), \( \neg \), \( ( \), \( ) \), the set of all expressions over the alphabet:

\[
\mathcal{W} = \mathcal{B} \cup \mathcal{F}
\]

where:

- \( \mathcal{B} \) is the set of expressions consisting of a single propositional symbol.
- \( \mathcal{F} \) is the set of formula-building operations:
  - \( \bigvee \): \( ( \ ) = \bigvee ( ) \)
  - \( \bigwedge \): \( ( \ ) = \bigwedge ( ) \)
  - \( \neg \): \( ( \ ) = \neg ( ) \)
  - \( \bigvee \): \( ( \ ) = \bigvee ( ) \)

Propositional Logic: Well-Formed Formulas
Propositional Logic: Well-Formed Formulas

Recall our inductive definition of the set $W$ of well-formed formulas in propositional logic. Given the alphabet $\{ A_1, A_2, \ldots \}$

- the set of all expressions over the alphabet.
- $\mathcal{A}$
- $\mathcal{B}$
- $\bigwedge$
- $\bigvee$
- $\leftrightarrow$
- $\forall$
- $\exists$

Recall our inductive definition of the set $W$ of well-formed formulas in propositional logic.
Propositional Logic: Well-Formed Formulas

Recall our inductive definition of the set \( W \) of well-formed formulas in propositional logic. Given the alphabet \( \{ (, ), ;, ;, ^, _, !, $ \} \), the set of all expressions over the alphabet is

\[
\begin{align*}
(\varphi \leftrightarrow \psi) &= (\varphi, \psi) \rightarrow 3 \\
(\varphi \rightarrow \psi) &= (\varphi, \psi) \leftrightarrow 3 \\
(\varphi \land \psi) &= (\varphi, \psi) \land 3 \\
(\varphi \lor \psi) &= (\varphi, \psi) \lor 3 \\
(\lnot \psi) &= (\psi) \neg 3
\end{align*}
\]

Recall our inductive definition of the set \( W \) of well-formed formulas in propositional logic. Given the alphabet \( \{ (, ), ;, ;, ^, _, !, $ \} \), the set of all expressions over the alphabet is

\[
\begin{align*}
\{ &\cdots \rightarrow, \land, \lor, \lnot, \psi, \varphi \} \}
\end{align*}
\]
An Algorithm for Recognizing WFFs

Lemma

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

- $\alpha$ is a propositional symbol.
- $\exists \gamma, \exists \neg \gamma$ where $\alpha \equiv (\land \neg \gamma)$
- $\exists \gamma$ where $\exists \gamma \equiv (\land \neg \gamma)
- \exists \gamma$ where $\exists \gamma \equiv (\land \neg \gamma)
- \exists \gamma$ where $\exists \gamma \equiv (\land \neg \gamma)$
- $\exists \gamma$ where $\exists \gamma \equiv (\land \neg \gamma)$
- $\exists \gamma$ where $\exists \gamma \equiv (\land \neg \gamma)$
- $\exists \gamma$ where $\exists \gamma \equiv (\land \neg \gamma)$

How would you prove this?
Induction, of course!
Lemma

Let \( \alpha \) be a \( \text{wff} \). Then exactly one of the following is true:

- \( \alpha \) is a propositional symbol.
- \( \alpha \) is a \( \text{wff} \).
- \( \alpha \) is of the form \( \langle \rightarrow, \land, \lor \rangle \) where the first \( \land \) or \( \lor \) is the first \( \text{wff} \).
- \( \alpha \) is of the form \( \langle \rightarrow, \rightarrow \rangle \) where the first \( \rightarrow \) is the first \( \text{wff} \).
- \( \alpha \) is of the form \( \langle \neg \neg \rangle \) where the first \( \neg \) is the first \( \text{wff} \).

How would you prove this?

Induction, of course!
An Algorithm for Recognizing WFFs

Lemma

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

- $\alpha$ is a propositional symbol.
- The first parentheses-balanced initial segment of the result of dropping the first ( from $\alpha$, and $\alpha'$ is a wff.
- $\alpha'$ where $\alpha'$ is one of $\{\leftrightarrow, \rightarrow, \land, \lor\}$.
- $\alpha' = \alpha$.
- $\alpha$ is a wff.
- $\alpha'$ is a wff.

How would you prove this?

Induction, of course!

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

Lemma

An Algorithm for Recognizing WFFs
An Algorithm for Recognizing WFFs

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

0. Begin with an initial construction tree containing a single node labeled with α.

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

2. Select a leaf labeled with an expression α1 which is not a propositional symbol.

3. If α1 does not begin with ( return false.

4. If α1 = ( then add a child to the leaf labeled by α1, label it with g, and goto 1.

5. Scan α1 until first reaching a nonempty expression having the same number of left and right parentheses. If there is no such g, return false.

6. If α1 = (g then add two children to the leaf labeled by α1, label them with g and g′, and goto 1.

7. If α1, label them with g and g′, and goto 1.

8. Return false.
How do we prove termination of this algorithm?

Termination

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

Soundness

If the algorithm returns true when given input $\phi$, then $\phi$ 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 $\phi$ 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 \( \text{wff} \), then \( \text{wff} \) is.

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

Completeness

If \( \text{wff} \), 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. The proof is by induction on the tree $T$ generated by the algorithm from the leaves up to the root.

**Soundness**

If the algorithm returns true when given input $a$, then $a$ is a WFF.

**Completeness**

If $a$ 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 the 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

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

If \( \alpha \) is a WFF, then the algorithm will return \text{true}.

Completeness

If \( \alpha \) is a WFF, then \( \alpha \) is a WFF.

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: \( \text{AvB} \), \( \text{AvB} \), \( \text{AvB} \), \( \text{AvB} \), etc.

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

\( \left( \left( A \lor B \right) \lor C \right) \)

\( \left( \left( A \lor B \right) \lor C \right) \)

Negation symbol binds stronger than binary connectives and its scope is as small as possible: \( \forall \rightarrow (B \lor A) \) means: \( B \lor A \land \forall \rightarrow \)

Symbolic binary connectives can be omitted: \( \text{AvB} \lor \text{AvB} \)

Outermost parentheses can be omitted: \( \text{AvB} \land \text{AvB} \)

Note that conventions are only unambiguous for \textit{wffs}, not for arbitrary expressions.

\( (A \lor B) \lor C \)

\( \left( (A \lor B) \lor C \right) \)

\( \left( (A \lor B) \lor C \right) \)

\( \left( (A \lor B) \lor C \right) \)

\( \left( (A \lor B) \lor C \right) \)
Propositional Logic: Semantics

Intuitively, given a wff and a value (either \( T \) or \( F \)) for each propositional symbol in the wff, we should be able to determine the value of \( \phi \).

The recursion theorem and the unique readability theorem guarantee that is \( \omega \) well-defined.

\[
| (\phi ')_\omega \rightarrow (\phi)_\omega | = \omega
\]

\[
(\phi ')_\omega \land (\phi)_\omega \rightarrow \omega = (\phi ')_\omega \rightarrow (\phi)_\omega \cdot
\]

\[
(\phi ')_\omega \lor (\phi)_\omega \rightarrow \omega = (\phi ')_\omega \rightarrow (\phi)_\omega \cdot
\]

\[
(\forall \phi \omega) = (\forall \phi)_\omega \cdot
\]

For each propositional symbol \( \phi \), and as if they were 0 and 1 respectively.

Now, we define a function \( \omega \) as follows (we compute with \( \omega \) and \( \omega ' \) as follows (we compute with \( \omega \) and \( \omega ' \)).

Let \( \omega \) be a function from \( \{ \land, \lor, \rightarrow, \neg \} \) to \( \omega \).

We call this function a truth assignment.

How do we make this precise?

Intuitively, given a wff \( \omega \) and a value (either \( \omega \) or \( \omega ' \)) for each propositional symbol \( \phi \),
There are other ways to present the semantics which are less formal but perhaps more intuitive.

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

<table>
<thead>
<tr>
<th>$\mathcal{G} \leftrightarrow \alpha$</th>
<th>$\mathcal{G}$</th>
<th>$\alpha$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\mathcal{G} \leftarrow \alpha$</td>
<td>$\mathcal{G}$</td>
<td>$\alpha$</td>
</tr>
<tr>
<td>$\mathcal{G} \land \alpha$</td>
<td>$\mathcal{G}$</td>
<td>$\alpha$</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$\neg \alpha$</th>
<th>$\neg \alpha$</th>
</tr>
</thead>
</table>

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.

<table>
<thead>
<tr>
<th>( \mathcal{G} )</th>
<th>( \alpha )</th>
<th>( \mathcal{G} )</th>
<th>( \alpha )</th>
<th>( \mathcal{G} )</th>
<th>( \alpha )</th>
<th>( \mathcal{G} )</th>
<th>( \alpha )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \bot )</td>
<td>( \top  )</td>
<td>( \top  )</td>
<td>( \top  )</td>
<td>( \bot )</td>
<td>( \bot  )</td>
<td>( \bot )</td>
<td>( \bot  )</td>
</tr>
<tr>
<td>( \top  )</td>
<td>( \bot  )</td>
<td>( \bot  )</td>
<td>( \bot  )</td>
<td>( \top )</td>
<td>( \top  )</td>
<td>( \top )</td>
<td>( \top  )</td>
</tr>
<tr>
<td>( \bot )</td>
<td>( \bot  )</td>
<td>( \bot  )</td>
<td>( \bot )</td>
<td>( \bot )</td>
<td>( \bot  )</td>
<td>( \bot )</td>
<td>( \bot  )</td>
</tr>
<tr>
<td>( \bot )</td>
<td>( \bot  )</td>
<td>( \bot  )</td>
<td>( \bot )</td>
<td>( \bot )</td>
<td>( \bot  )</td>
<td>( \bot )</td>
<td>( \bot  )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( \mathcal{G} )</th>
<th>( \alpha )</th>
<th>( \mathcal{G} )</th>
<th>( \alpha )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \bot )</td>
<td>( \bot  )</td>
<td>( \bot  )</td>
<td>( \bot  )</td>
</tr>
<tr>
<td>( \top  )</td>
<td>( \top  )</td>
<td>( \top  )</td>
<td>( \top  )</td>
</tr>
<tr>
<td>( \bot )</td>
<td>( \bot  )</td>
<td>( \bot  )</td>
<td>( \bot  )</td>
</tr>
</tbody>
</table>

Truth Tables
Complex truth tables can also be used to calculate all possible values of \( \psi \) for a given \( \text{wff} \).

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

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 \( \text{wff} \).
Complex 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. There is a row for each possible truth assignment to the propositional connectives. There is a row for each possible truth assignment to the propositional connectives.

<table>
<thead>
<tr>
<th>( \neg )</th>
<th>( \land )</th>
<th>( \lor )</th>
<th>( \rightarrow )</th>
<th>( \land )</th>
<th>( \rightarrow )</th>
<th>( \land )</th>
<th>( \rightarrow )</th>
<th>( \land )</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Truth tables can also be used to calculate all possible values of a given wff.

Complex truth tables
Complex truth tables

Truth tables can also be used to calculate all possible values of \( \phi \) 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. 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>
</tr>
</thead>
<tbody>
<tr>
<td>( T )</td>
<td>( T )</td>
<td>( T )</td>
</tr>
<tr>
<td>( F )</td>
<td>( T )</td>
<td>( T )</td>
</tr>
<tr>
<td>( T )</td>
<td>( T )</td>
<td>( T )</td>
</tr>
<tr>
<td>( F )</td>
<td>( T )</td>
<td>( T )</td>
</tr>
<tr>
<td>( T )</td>
<td>( T )</td>
<td>( T )</td>
</tr>
<tr>
<td>( F )</td>
<td>( T )</td>
<td>( T )</td>
</tr>
<tr>
<td>( T )</td>
<td>( T )</td>
<td>( T )</td>
</tr>
<tr>
<td>( F )</td>
<td>( T )</td>
<td>( T )</td>
</tr>
<tr>
<td>( T )</td>
<td>( T )</td>
<td>( T )</td>
</tr>
<tr>
<td>( F )</td>
<td>( T )</td>
<td>( T )</td>
</tr>
</tbody>
</table>

\(((\neg A_3 \lor A_2) \land A_1)\)
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. Thus, a truth table can also be used to calculate all possible values of a given wff.
If \( \alpha \) is a wff, then a truth assignment \( \nu \) satisfies \( \alpha \) if \( \nu(\alpha) = T \).

Definitions

A wff is satisfiable if there exists a truth assignment \( \nu \) which satisfies it.

Suppose \( \Gamma \) is a set of wffs. Then \( \Gamma \) tautologically implies \( \phi \), if every truth assignment which satisfies each formulæ in \( \Gamma \) also satisfies \( \phi \).

Particular cases:

- If \( \Gamma \) is satisfiable, then we say \( \phi \) is a tautology or is valid and write \( \Gamma \models \phi \).
- If \( \phi \) is unsatisfiable, then \( \phi \) is true for every wff.
- If \( \phi \equiv (\psi \land \psi) = \phi \) and \( \phi \equiv \psi \), then \( \phi \) and \( \psi \) are tautologically equivalent.
- For a finite \( \Gamma \), \( \Gamma \models \phi \) if and only if \( V(\phi) \neq \emptyset \).
Definitions

If \( \alpha \) is a wff, then a \emph{truth assignment} \( \nu \) \emph{satisfies} \( \alpha \).

A wff \( \alpha \) \emph{is satisfiable} if there exists some truth assignment \( \nu \) which satisfies \( \alpha \).

If \( \alpha \) is a wff, then a \emph{truth assignment} \( \nu \) \emph{satisfies} \( \alpha \) if \( \nu (\alpha) = T \).

\( \text{Particular cases:} \)

- If \( \nu (\alpha) = T \) for every wff, then \( \nu \) is unsatisfiable.
- If \( \nu (\alpha) = F \) (shorthand for \( \nu (g \alpha) = F \)), then \( \alpha \) is unsatisfiable.
Definitions

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

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

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

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

Particular cases:

- If $\mathcal{A} = \emptyset$, then we say $\alpha$ is a tautology or is valid and write $\mathcal{A} \models \alpha$.
- If $\mathcal{A}$ is unsatisfiable, then $\mathcal{A} \models \nu$ for every wff $\nu$.
- If $\mathcal{A} = \{f, g\}$ (shorthand for $f \land g$) and $\mathcal{A} \models \nu$, then $\mathcal{A} \models \nu$ and $\mathcal{A}$ are tautologically equivalent.
Definitions

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

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

**Particular cases:**

- If $\emptyset \models \alpha$, then we say $\alpha$ is a **tautology** or $\alpha$ is **valid** and write $\models \alpha$.

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

If $\alpha = \beta$, then $\alpha$ and $\beta$ are **tautologically equivalent**.

For an finite $\alpha$, $\models \alpha$ if and only if $\models \forall (\alpha)$.
Definitions

If $\alpha$ is unsatisfiable, then for every wff $\varphi$.

- If $\varphi$ is unsatisfiable, then
- $\models \alpha \Rightarrow \neg \models \alpha$

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

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

Particular cases:

- If $\models \alpha$, then we say $\alpha$ is a tautology or $\alpha$ is valid and write $\models \alpha$.
- If $\models \emptyset$, then we say $\emptyset$ is a contradiction and write $\models \emptyset$.

If $\models \varphi \Rightarrow \models \alpha$, then $\varphi$ and $\alpha$ are tautologically equivalent.

For an infinite $\mathcal{B}$, $\models \alpha$ if and only if $\mathcal{B} \models \alpha$.

$\models \alpha \iff \mathcal{B} \models \alpha$.

12-d
Definitions

If \( \varphi \) is a wff, then a truth assignment \( \nu \) satisfies \( \varphi \) if and only if \( \nu(\varphi) = T \).

A wff \( \varphi \) is satisfiable if there exists some truth assignment \( \nu \) which satisfies \( \varphi \).

If \( \varphi \) is a wff, then a truth assignment \( \nu \) satisfies \( \varphi \) if \( \nu(\varphi) \neq F \).

Suppose \( \varphi \) is a set of wffs. Then \( \varphi \) tautologically implies \( \varphi \), if every truth assignment which satisfies each formula in \( \varphi \) also satisfies \( \varphi \).

Particular cases:

1. If \( \varphi \) \( \equiv \) \( \psi \) (shorthand for \( \varphi \implies \psi \) and \( \psi \implies \varphi \)), then \( \varphi \) and \( \psi \) are tautologically equivalent.
2. If \( \varphi \) is unsatisfiable, then \( \varphi \) \( \equiv \) \( \bot \) for every wff \( \alpha \).
3. If \( \alpha \) \( \equiv \) \( \bot \), then we say \( \alpha \) is a tautology or \( \alpha \) is valid and write \( \models \alpha \).

\[ T = \{ \alpha \mid \varphi \in \Phi \implies \alpha \} \]
Definitions

If $\alpha$ is a wff, then a truth assignment $v$ satisfies $\alpha$ if

$$v(\alpha) = \top.$$ 

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

$$v(\alpha) = \top.$$ 

A set $\Delta$ of wffs is a tautology if

$$\Delta \models \alpha,$$

for every $\alpha$. A set $\Delta$ of wffs is unsatisfiable if

$$\Delta \models \bot$$

For a finite $\alpha$, $\alpha \models \chi \iff \chi$ is valid.

Particular cases:

- $\alpha$ is tautologically equivalent to $\beta$ if

$$\models \alpha \iff \beta.$$ 

Suppose $\Delta$ is a set of wffs. Then $\alpha$ is a tautology if

$$\Delta \models \alpha.$$
(A \rightarrow \neg B) \lor (B \land \forall) \bullet
examples

(\(A \land B\)) \^ (\(A \land B\)) \[false\], but not valid.

(\(A \land B\)) \^ (\(A \land B\)) \[false\], (\(A \land B\)) \[false\].

Suppose you had an algorithm SAT which would take a wff as input and return true if issatisfiable, but not valid.
Examples

\((A \iff V) \lor (B \leftarrow \land A \leftarrow) \lor (B \land V)\) •

\((B \leftarrow \land A \leftarrow) \lor (B \land V)\) •

is satisfiable, but not valid.
Examples

\[(A \rightarrow B) \lor (B \rightarrow A) \lor (B \land V) \]

is unsatisfiable.

\[(B \rightarrow V) \lor (B \land A') \lor (B \land V) \]

is satisfiable, but not valid.

Suppose you had an algorithm SAT which would take a wff as input and return true if issatisfiable and false otherwise. How would you use this algorithm to verify each of the claims made above?
Examples

\((\forall \neg \lor \forall) = \{ \forall \neg \forall \}\)

\(B = \{ B \leftarrow \forall \forall \}\)

is unsatisfiable.

\((B \leftrightarrow \forall) \lor (B \neg \land \forall \neg) \lor (B \land \forall)\)

is satisfiable, but not valid.

\(B \land \forall \land \forall \lor (B \land \forall)\)
Examples

\( B \leftarrow \land \forall \leftarrow \) is tautologically equivalent to \( (B \lor \forall) \leftarrow \)

\( (\forall \leftarrow \lor \forall) = \{ \forall \leftarrow \forall \} \)

\( B = \{ B \leftrightarrow \forall, \forall \} \)

\( (B \leftrightarrow \forall) \lor (B \leftarrow \land \forall) \lor (B \land \forall) \) is unsatisfiable.

\( (B \leftarrow \land \forall) \lor (B \land \forall) \) is satisfiable, but not valid.
Verify each of the claims made above.

\( (A \_ B) \land (B \_ A) \) is tautologically equivalent to \( (A \lor B) \lor (B \lor A) \).
\( (A \lor B) = \{ A \lor A \} \).
\( B = \{ B \leftarrow A, A \} \).
\( (B \leftrightarrow A) = (B \lor \neg A \lor \neg B \lor A) \).
\( (B \land A) \lor (B \land \neg A) \lor (B \lor A) \).

Examples
Suppose you had an algorithm \( \text{SAT} \) which would take a \( \text{wff} \) \( \alpha \) as input and return \( \text{true} \) if \( \alpha \) is satisfiable and \( \text{false} \) otherwise. How would you use this algorithm to verify each of the claims made above?

\[
B \land \forall \lnot \forall \equiv (B \lor \forall) \equiv \\
(\forall \lnot \lor \forall) = \{ \forall \lnot, \forall \} \equiv \\
((B \lnot) \lor (B \leftarrow \forall) \lor \forall) \land \\
(B \leftarrow \forall) \lor (B \lnot \land \forall) \lor (B \land \forall) \\
\]

1. \( B \land \forall \lnot \forall \) is tautologically equivalent to \( (B \lor \forall) \equiv \).
2. \( (\forall \lnot \lor \forall) = \{ \forall \lnot, \forall \} \equiv \).
3. \( ((B \lnot) \lor (B \leftarrow \forall) \lor \forall) \land \\
(B \leftarrow \forall) \lor (B \lnot \land \forall) \lor (B \land \forall) \equiv \).

Examples
verify each of the claims made above?

true if $\alpha$ is satisfiable and false otherwise. How would you use this algorithm to
Suppose you had an algorithm $\text{SAT}$ which would take a wff $\alpha$ as input and return

$B \leftarrow \land \forall \leftarrow (B \lor \forall) \leftarrow$ •

$((\forall \leftarrow \lor \forall) \land (\forall \leftarrow) \lor A) \land (\forall \leftarrow \lor A) = \vdash \{ \forall \leftarrow A \}$ •

$((B \leftarrow) \lor (B \leftarrow A) \lor A) \quad B = \vdash \{ B \leftarrow A, \forall \}$ •

is unsatisfiable.

is satisfiable but not valid.

Examples
Examples

Suppose you had an algorithm \texttt{SAT} which would take a wff \( \alpha \) as input and return \texttt{true} if \( \alpha \) is satisfiable and \texttt{false} otherwise. How would you use this algorithm to verify each of the claims made above?

\[
((B \leftarrow \land \lor) \leftrightarrow (\overline{B} \lor \overline{A} \lor)) \leftarrow \quad \text{\( B \leftarrow \land \lor \) is tautologically equivalent to} \quad \text{\( (B \lor \overline{A} \lor) \leftarrow \)} \bullet
\]

\[
((\overline{V} \lor V) \lor (V \lor) \lor V) \quad (\overline{V} \lor V) = \{ \overline{V}, V \} \bullet
\]

\[
((\overline{B} \lor (B \leftarrow V) \lor V) \quad B = \{ B \leftarrow V, V \} \bullet
\]

is unsatisfiable.

\[
(B \leftarrow V) \lor (B \leftarrow \land \lor) \lor (B \land V) \bullet
\]

is satisfiable, but not valid.

\[
(B \leftarrow \land \lor) \lor (B \land V) \bullet
\]
Examples

Now suppose you had an algorithm CHECKVALID which returns true when $\alpha$ is valid and false otherwise. How would you verify the claims given this algorithm?

$$((\neg \alpha \land \alpha) \leftrightarrow (\alpha \lor \alpha) \land \alpha)$$

$\beta \land \alpha \land \alpha$ is tautologically equivalent to $(\alpha \lor \alpha) \land \alpha$.

$$(\alpha \lor \alpha) \land (\alpha \land \alpha) \lor \alpha) \equiv \{ \alpha \land \alpha \}$$

$$(\beta \lor (\beta \land \alpha) \lor \alpha) \quad \beta = \{ \beta \land \alpha \}$$

$\beta \land \alpha$ is unsatisfiable.

$\alpha \land \alpha \lor (\beta \land \alpha) \land (\alpha \land \alpha) \land \alpha$ is satisfiable, but not valid.
Examples

Satisfiability and validity are dual notions: is unsatisfiable if and only if is valid.

Now suppose you had an algorithm CHECKVALID which returns true when is valid and false otherwise. How would you verify the claims given this algorithm?

\[
((B \leftarrow \land \forall \neg) \leftrightarrow (B \lor \forall \neg) \rightarrow) \rightarrow \land \forall \neg \land \forall \neg \leftrightarrow (B \lor \forall \neg) \rightarrow \bullet
\]

\[
((\forall \neg \lor \forall \neg) \lor (\forall \neg \lor \forall \neg) \rightarrow (\forall \neg \lor \forall) = \{ \forall \neg, \forall \} \bullet
\]

\[
((B \leftarrow \lor (B \leftarrow \forall \neg) \lor \forall \neg) \rightarrow \forall \rightarrow \{ B \leftarrow \forall \neg \forall \} \bullet
\]

is unsatisfiable.

is satisfiable but not valid.
Determining Satisfiability using Truth Tables

An Algorithm for Satisiability

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

An Algorithm for Tautological Implication

To check whether \( \phi_1, \phi_2, \ldots, \phi_k \models \psi \), check the satisfiability of \((\phi_1 \land \cdots \land \phi_k) \lor \neg \psi \). If it is unsatisfiable, then \( \phi_1, \phi_2, \ldots, \phi_k \models \psi \), otherwise \( \phi_1, \phi_2, \ldots, \phi_k \not\models \psi \).
Determining Satisfiability using Truth Tables

An Algorithm for Satisifiability

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

An Algorithm for Tautological Implication

To check whether \( \phi \) is satisfiable, form the truth table for \( \phi \). If there is a row in which \( T \) appears as the value for \( \phi \), then \( \phi \) is satisfiable. Otherwise, \( \phi \) is unsatisfiable.
Determining Satisfiability using Truth Tables

Example

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

Determining Satisfiability using Truth Tables

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>C</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>2</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>3</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>4</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>5</td>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>6</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>7</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

((B ⊃ C) ∨ (A ⊃ B)) ∨ A

(((B ⊃ C) ∨ (A ⊃ B)) ∨ A)
### Example

Determining Satisfiability using Truth Tables
Determining Satisfiability using Truth Tables

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>((\neg B \land C) \lor (\forall A \land B))\</th>
<th>\forall A</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

Example
### Example

Determining Satisfiability using Truth Tables

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>((\neg B \land C) \lor (\forall A \land B))\</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
</tbody>
</table>

\((\forall A \land B) \lor (\neg B \land C)\) \lor A
### Example

Determining Satisfiability using Truth Tables

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

\[ ((B \land C) \land (A \land B)) \land (A \land (C \land B)) \land A \]
### Example

Determining Satisfiability using Truth Tables

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
<td>F</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>F</td>
</tr>
<tr>
<td>F</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>T</td>
<td>F</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>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>F</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</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>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>

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

Determining Satisfiability using Truth Tables

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<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>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

\[
((B \rightarrow C) \land (A \rightarrow C)) \land \lnot A
\]

\[
((B \rightarrow C) \land (A \rightarrow C)) \land \lnot A
\]

\[
((B \rightarrow C) \land (A \rightarrow C)) \land \lnot A
\]
Determining Satisfiability using Truth Tables

Example

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

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>((\neg B \land C) \lor (\forall \neg \land B)) \lor A)</th>
</tr>
</thead>
<tbody>
<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>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>T</td>
<td>T</td>
<td>T</td>
</tr>
</tbody>
</table>
Determining Satisfiability using Truth Tables

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

Example
Determining Satisfiability using Truth Tables

What is the complexity of this algorithm?

$n^2$ 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.
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.
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?
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.

Determining Satisfiability using Truth Tables
Some tautologies

Associative and Commutative laws for $\land$, $\lor$, $\equiv$

Distributive Laws

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

Negation

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

DeMorgan's Laws

\[
(A \land B) \equiv \neg (A \lor \neg B) \\
(A \lor B) \equiv \neg (A \land \neg B)
\]
**Distributive Laws**

\[
((a \land c) \lor (b \land c)) \leftrightarrow ((a \lor b) \land c) \quad \bullet
\]

\[
((a \lor c) \land (b \lor c)) \leftrightarrow ((a \land b) \lor c) \quad \bullet
\]

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

**Some Tautologies**
Some tautologies

**Associative and Commutative Laws**

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

**Negation**

\[
((C \land A) \lor (B \land A)) \leftrightarrow ((C \lor A) \land A) \\
((C \lor A) \land (B \land A)) \leftrightarrow ((C \land A) \lor A)
\]
Some tautologies

De Morgan’s Laws

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

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

Negation

\[\text{Negation}\]

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

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

\[A \leftrightarrow A \land\]

Distributive Laws

\[\leftrightarrow \land, \lor\]

Associative and Commutative Laws for $\land, \lor$
More Tautologies

**Implication**

\[(B \land \neg\neg) \iff (B \leftarrow \neg)\]

**Contradiction**

\[\neg(B \land \neg\neg) \iff \neg(B \leftarrow \neg)\]

**Contraposition**

\[(B \land \neg\neg) \iff \neg(B \leftarrow \neg)\]

**Exportation**

\[(B \land \neg\neg) \iff (B \leftarrow \neg)\]
More Tautologies

\( \forall \rightarrow \land \forall \) •

Excluded Middle

\((B \land \forall \rightarrow) \leftrightarrow (B \leftarrow \forall)\) •

Implication

19.4
(∀\neg \lor \forall) \bullet

Contradiction

∀\neg \land ∀ \bullet

Excluded Middle

(\forall \land \forall) \leftrightarrow (\forall \leftrightarrow \forall) \bullet

Implication

More Tautologies
(A \iff B) \iff (B \iff A) •

Contraposition

(A \iff \neg A) •

Contradiction

A \iff A •

Excluded Middle

(B \iff A) \iff (B \iff A) •

Implication
More Tautologies

Implication

\[(\mathcal{C} \leftrightarrow B) \leftrightarrow A \leftrightarrow (\mathcal{C} \leftrightarrow (B \lor A))\]

Exportation

\[(\mathcal{A} \leftrightarrow B) \leftrightarrow (\mathcal{B} \leftrightarrow A)\]

Contraposition

\[(\mathcal{A} \leftrightarrow \mathcal{B}) \leftrightarrow (\mathcal{B} \leftrightarrow A)\]

Contradiction

\[(\mathcal{A} \leftrightarrow \neg \mathcal{A})\]

Contradiction

\[\neg \mathcal{A} \lor \mathcal{A}\]

Excluded Middle

\[\neg \mathcal{B} \lor B\]

Implication
We have five connectives: \(\neg, \wedge, \vee, \rightarrow, \leftrightarrow\). Would we gain anything by having more? Would we lose anything by having fewer?

Example: Ternary Majority Connective

\[ E \# (\# \# \#) = E \lor (E \# E) \text{ iff the majority of } E \lor E \lor E \text{ are } T. \]

What does this new connective do for us?

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

How do we show this formally?
Propositional Connectives

We have connectives:

\( \land \), \( \lor \), \( \rightarrow \), \( \leftrightarrow \), \( \neg \), \( \exists \), \( \forall \). Would we gain anything by having more? Would we lose anything by having fewer?

Example: Ternary Majority Connective

\[
\mathbf{L} = \left( \left( \left( \land \neg x \neg y \right) \land \neg z \right) \land \neg x \right)
\]

\[
\left( \left( \left( \land \neg z \neg y \right) \land \neg x \right) \land \neg z \right)
\]

Example: Ternary Majority Connective

More? Would we lose anything by having fewer? We have three connectives: \( \land \), \( \lor \), \( \rightarrow \), \( \leftrightarrow \), \( \neg \). Would we gain anything by having three connectives? Propositional Connectives

Propositional Connectives
Propositional Connectives

We have ve connectives: 

- \(\lor\)
- \(\land\)
- \(\neg\)
- \(!\)
- \($\)

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

Example: Ternary Majority Connective

\[ \text{If the majority of } a, b, c \text{ are } \top \text{ then } \top = ((\lor b c \not\top) a) \]

\[ (\lor b c \not\top) = (\lor b, c, \not\top) \]

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

How do we show this formally?

What does this new connective do for us?

More? Would we lose anything by having fewer? Would we gain anything by having more?
We have connectives: 

\[ \land, \lor, \lnot, \neg, \#, \cdots \]

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

Example: Ternary Majority Connective

\[ \# \]

What does this new connective do for us?

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

What does this new connective do for us?

\[ \mathbf{J} = (\lor \lor \lor \#) \]

Example: Ternary Majority Connective

\[ \# \]

more? Would we lose anything by having fewer?

We have five connectives: \( \land, \lor, \lnot, \neg, \# \). Would we gain anything by having

Propositional Connectives
Propositional Connectives

We have the connectives:

\( \land, \lor, \neg, \iff \).

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

Example: Ternary Majority Connective

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

What does this new connective do for us?

\[ \text{iff the majority of } a, b, c \text{ are } T \]

\[ \text{Example: Ternary Majority Connective} \]

How do we show this formally?

More? Would we lose anything by having fewer?

We have five connectives: \( \land, \lor, \neg, \iff \). Would we gain anything by having

Propositional Connectives
Boolean Functions

For $k \geq 0$, a $k$-place Boolean function is anything which is a $k$-place Boolean function from $\mathcal{F}_k$ to $\mathcal{F}_k$. For example, if

\[ B(x_1, x_2) = x_1 \lor x_2, \]

then $B$ is a 2-place Boolean function whose value is given by the following table:

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$B(x_1, x_2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$T$</td>
<td>$T$</td>
<td>$T$</td>
</tr>
<tr>
<td>$T$</td>
<td>$F$</td>
<td>$T$</td>
</tr>
<tr>
<td>$F$</td>
<td>$F$</td>
<td>$F$</td>
</tr>
</tbody>
</table>

Each wff $\alpha$ determines a corresponding Boolean function $B^{\alpha}$. For example, if $\alpha = \forall x \forall y \forall z \exists x \phi$, then $B^{\alpha}$ is a function from $\mathcal{F}_\alpha$ to $\mathcal{F}_\phi$. For $\alpha \geq 0$, a $k$-place Boolean function is a function from $\mathcal{F}_\alpha$ to $\mathcal{F}_k$. A.

\[ \emptyset \subseteq \mathcal{F}_\alpha \subseteq \mathcal{F}_k \]
In general, suppose that $\alpha$ is a WFF whose propositional symbols are included in $A_1, \ldots, A_n$. We define an $n$-place Boolean function $B_n$, the Boolean function realized by $\alpha$ as $B_n(X_1, \ldots, X_n) = (\alpha)(X_1, \ldots, X_n)$, where $(\alpha)(X_i) = X_i$. Note that the function $B_n$ is determined by both the formula and the choice of $n$. In particular, $\alpha$ does not need to include all the symbols in $A_1, \ldots, A_n$.
Realizing Boolean Functions
In general, suppose that \( \varphi \) is a \( \varphi \)-formula whose propositional symbols are included in \( A_1, \ldots, A_n \). We define an \( n \)-place Boolean function \( B_n \), the Boolean function realized by \( \varphi \) as
\[
\mu X = (\forall X)\varphi = (\forall X_1, \ldots, X_n)_{\mu B_n}
\]
where \( \varphi \). In other words, given the values \( \mu X \ldots X \) of the truth value of \( \varphi \) when \( A_1, \ldots, A_n \) are \( \mu \), \( \ldots, \mu \), the Boolean function \( B_n \) is realized by \( \varphi \) as
\[
\mu X_1, \ldots, X_n = (\forall X_1, \ldots, X_n)_{\mu B_n}
\]
In general, suppose that \( \varphi \) is a \( \varphi \)-formula whose propositional symbols are included in
Realizing Boolean Functions

In general, suppose that \( \alpha \) is a wff whose propositional symbols are included in \( \{A_1, \ldots, A_n\} \). In particular, \( \alpha \) does not need to include all the symbols in \( \{A_1, \ldots, A_n\} \). We define an \( n \)-place Boolean function \( B_n \), the Boolean function realized by \( \alpha \) is determined by both the formula \( \alpha \) and the choice of \( n \).

Note that the function \( B_n \) is determined by both the formula \( \alpha \) and the choice of \( n \).

\[
\begin{align*}
\bar{\alpha} = (\forall \bar{A}) \alpha, \\
\forall \bar{A} X = (\forall \bar{A} \bar{X}) \alpha = (\forall \bar{X}) \bar{A} = (\forall \bar{X}) (\forall \bar{X} \bar{A} \bar{X})^{\alpha}
\end{align*}
\]

In other words,

\[
\begin{align*}
\bar{A} X = (\forall \bar{A} \bar{X}) \alpha, \\
\forall \bar{X} \bar{A} = (\forall \bar{X}) (\forall \bar{X} \bar{A} \bar{X})^{\alpha}
\end{align*}
\]

Written the values given \( \bar{A} \), \( X \) are \( \bar{A} \); \( \bar{X} \) when \( \bar{A} \) when \( \bar{A} \) is given value given to \( \bar{A} \) as \( \bar{A} \).

Recall that \( \alpha \) is not included in \( \{A_1, \ldots, A_n\} \).
Examples

\[ I \quad = \quad B \]

\[ N \quad = \quad B_1 \]

\[ K \quad = \quad B_2 \]

\[ A_1 \]

\[ C \quad = \quad B_2 \]

\[ E \quad = \quad B_2 \]

From these functions, we can construct others by composition.

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.
From these functions, we can construct others by composition.

\[ (((X \cdot I X) \cdot I) N \cdot (((X \cdot I X) \cdot I) N) V = (X \cdot I X) \cdot V \cdot I \cdot V \cdot I \cdot B \]

\[
\begin{align*}
\text{Claim: Every Boolean function can be obtained as a composition of } & \text{I, N, K, A, C, and E.} \\
\text{W} & \text{e will explain why this is true shortly.}
\end{align*}
\]

\[
\begin{align*}
\text{From these functions, we can construct others by composition.} \\
\text{I}_V & \text{I} \cdot V_B = E \\
\text{I}_V & \text{I} \cdot V_B = C \\
\text{I}_V & \text{I} \cdot V_B = A \\
\text{I}_V & \text{I} \cdot V_B = K \\
\text{I}_V & \text{I} \cdot V_B = N \\
\text{I}_V & \text{I} \cdot V_B = I
\end{align*}
\]
Examples

We will explain why this is true shortly.

From these functions, we can construct others by composition.

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

\[
(((\tilde{V} \land I) \land I) \land V) = (\tilde{V} \land I) \land \tilde{V} \land I \land V
\]

\[
E = (\tilde{V} \land I) \land \tilde{V} \land I
\]

\[
E = (\tilde{V} \land I) \land \tilde{V} \land I
\]
Let $\alpha$ and $\beta$ be wffs whose sentence symbols are among $A_1, \ldots, A_n$. Then:

**Theorem**

Formulas and the Boolean Functions they Realize
Theorem

Let \( \varphi \) and \( \psi \) be wffs whose sentence symbols are among \( A_1, \ldots, A_n \).

(a) \( \varphi \models \psi \) iff \( \psi \models \varphi \) iff for all \( n \)-tuples \( \bar{x} \), \( \varphi(\bar{x}) = T \) implies \( \psi(\bar{x}) = T \) iff \( \{ \psi \} = \{ \varphi \} \).

(b) \( \varphi \models \psi \) iff \( \psi \models \varphi \) iff for every truth assignment satisfying \( \varphi \) also satisfies \( \psi \) iff \( \varphi \) is tautologically equivalent to \( \psi \).

(c) Follows from (a) and definition of tautology.

(d) Follows from (a) and

\[
\varphi \models \psi \text{ iff for all } n \text{-tuples } \bar{x}, \psi(\bar{x}) = T \implies \varphi(\bar{x}) = T.
\]

Proof

Let \( \varphi \) and \( \psi \) be wffs whose sentence symbols are among \( A_1, \ldots, A_n \).
By shifting our focus from formulas to Boolean functions, tautologically equivalent

\[ X \supseteq \lambda \quad \text{iff} \quad \lambda \supseteq X \quad \lambda = X \]

\[ \text{If for all } \lambda \text{-tuples, } \lambda = \text{implies} \lambda = \text{implies} \]
\[ \lambda = (\epsilon) \alpha \text{ implies } \lambda = (\alpha) \alpha \text{ implies } \]
\[ \alpha \text{ satisfies a truth assignment satisfying } \alpha \text{ also satisfies } \]
\[ \alpha \text{ and } \alpha \text{ are wffs.} \]

\[ \{ \lambda \} = (\lambda) \alpha \text{ if } \alpha \text{ is the range of } \lambda \]
\[ (\lambda) \alpha \text{ is logically equivalent to } \lambda \text{ if } \alpha \]
\[ (\epsilon) \alpha \text{ is the range of } \lambda \text{ if for all } \lambda \text{ we have } \lambda \]
\[ \alpha \text{ among } \lambda \text{ symbols are among } \lambda \text{ and } \lambda \text{ are wffs.} \]

\[ \lambda \text{ and the Boolean Functions They Realize} \]

Theorem
Theorem

Let \( G \) be an \( n \)-place Boolean function, \( n \geq 1 \). There exists a \( \text{wff} \) \( \psi \) such that

\[
G = \psi \circ B^n
\]

i.e., such that \( \psi \) realizes the function \( G \).
Completeness of Propositional Connectives

**Theorem**

Let $G$ be an $n$-place Boolean function, $n \geq 1$. There exists a wff such that $G = B^n$, i.e., such that $\text{realizes } G$.

**Proof**

Let $\mathcal{C}$ be an $n$-place Boolean function, $n \geq 1$. There exists a wff such that $\mathcal{C} = B^n$.

$$\forall x_1 \wedge \cdots \wedge x_n \wedge \exists x_1 = \alpha$$

$$\forall x_1 \vee \cdots \vee \exists x_1 = \exists x_1$$

$$\mathcal{F} = \{x_1 \exists \forall \}$$

$$\mathcal{L} = \{x_1 \exists \forall \}$$

$$\exists x_1 \wedge \cdots \wedge x_n \wedge \exists x_1 = \alpha$$

$$\exists x_1 \vee \cdots \vee \exists x_1 = \exists x_1$$

$$\mathcal{L} = \{x_1 \exists \forall \}$$

$$\mathcal{L} = \{x_1 \exists \forall \}$$

Then $\mathcal{C}$ realizes $\mathcal{C}$.
We know that \( x = (\forall \alpha) \beta \) where \( \beta \) is some set of propositional connectives.

Thus, one of the points where \( \mathbf{L} \) is \( \mathbf{C} \) is

\[
\mathbf{L} = (x)_{\forall \beta}^{u \beta} \quad \text{iff} \quad x = (\forall \alpha) \beta.
\]

But by construction, \( (x)_{\forall \beta}^{u \beta} \) is \( \mathbf{L} \) if and only if

\[
\langle \forall \alpha x, \ldots, \forall x \rangle = x \quad \text{iff} \quad x = (\forall \alpha) \beta.
\]

Since \( x = (\forall \alpha) \beta \) and \( \beta \) is a disjunction of literals, it follows that

\[
\forall x = (\forall \alpha) \beta \quad \text{where} \quad (\forall \alpha) \beta = (x)_{\forall \beta}^{u \beta}.
\]

Proof, continued

Completeness of Propositional Connectives
Completeness of Propositional Connectives

Proof, continued

We know that
\[
B_n (\neg X) = \max (B_n i (\neg X))
\]

But by construction,
\[
B_n i (\neg X) = \text{true} \iff \neg X = \langle x_i^1, \ldots, x_i^n \rangle
\]

Thus, a corollary is that for every wff, there exists a tautologically equivalent wff in its negation.

which is a conjunction of literals, where a literal is either a propositional symbol or

A formula is in DNF if it is a disjunction of formulas, each of

The resulting formula is not unique. The formula built is in so-called disjunctive

normal form (DNF). A formula is in DNF if it is a disjunction of formulas, each of

Thus shows that every Boolean function can be realized by a wff which uses only the connectives

This shows that every Boolean function can be realized by a wff. In fact, every

Thus, a corollary is that for every wff, there exists a tautologically equivalent wff in its negation.

which is a conjunction of literals, where a literal is either a propositional symbol or

A formula is in DNF if it is a disjunction of formulas, each of

The resulting formula is not unique. The formula built is in so-called disjunctive

normal form (DNF). A formula is in DNF if it is a disjunction of formulas, each of

Thus, a corollary is that for every wff, there exists a tautologically equivalent wff in its negation.
Let $G$ be a 3-place Boolean function defined as follows:

\[
\begin{align*}
\mathsf{T} & = (\mathsf{T}, \mathsf{T}, \mathsf{T}, \mathsf{T}) \land \neg G \\
\mathsf{F} & = (\mathsf{T}, \mathsf{T}, \mathsf{T}, \mathsf{T}) \land \neg G \\
\mathsf{F} & = (\mathsf{T}, \mathsf{T}, \mathsf{T}, \mathsf{T}) \land \neg G \\
\mathsf{T} & = (\mathsf{T}, \mathsf{T}, \mathsf{T}, \mathsf{T}) \land \neg G \\
\mathsf{F} & = (\mathsf{T}, \mathsf{T}, \mathsf{T}, \mathsf{T}) \land \neg G \\
\mathsf{T} & = (\mathsf{T}, \mathsf{T}, \mathsf{T}, \mathsf{T}) \land \neg G \\
\mathsf{F} & = (\mathsf{T}, \mathsf{T}, \mathsf{T}, \mathsf{T}) \land \neg G \\
\mathsf{T} & = (\mathsf{T}, \mathsf{T}, \mathsf{T}, \mathsf{T}) \land \neg G \\
\mathsf{F} & = (\mathsf{T}, \mathsf{T}, \mathsf{T}, \mathsf{T}) \land \neg G
\end{align*}
\]

Note that another formul\(\mathsf{a}\) which realizes $G$ is $\mathsf{F} \land \mathsf{F} \land \mathsf{F}$. Thus, adding additional connectives to a complete set may allow a function to be realized more concisely.
Completeness of Propositional Connectives

Example

Let $G$ be a 3-place Boolean function defined as follows:

$G(F; F; F) = F$

$G(F; F; T) = T$

$G(F; T; F) = T$

$G(F; T; T) = F$

$G(T; F; F) = T$

$G(T; F; T) = F$

$G(T; T; F) = F$

$G(T; T; T) = T$

There are four points at which $G$ is true, so a DNF formula which realizes $G$ is:

$$
\neg (\neg A_1 \lor \neg A_2 \lor \neg A_3) \lor (A_1 \land \neg A_2 \lor \neg A_3) \lor (A_1 \land A_2 \land \neg A_3) \lor (A_1 \land \neg A_2 \land A_3) 
$$

Note that another formula which realizes $G$ is $A_1 \land A_2 \land A_3$. Thus, adding additional connectives to a complete set may allow a function to be realized more concisely.
Completeness of Propositional Connectives

Example

Let $G$ be a 3-place Boolean function defined as follows:

$G(F; F; F) = F$

$G(F; F; T) = T$

$G(F; T; F) = T$

$G(F; T; T) = F$

$G(T; F; F) = T$

$G(T; F; T) = F$

$G(T; T; F) = F$

$G(T; T; T) = T$

There are four points at which $G$ is true, so a DNF formula which realizes $G$ is

$\neg = (\neg, \neg, \neg) G$

$\neg = (\neg, \neg, \neg) G$

$\neg = (\neg, \neg, \neg) G$

$\neg = (\neg, \neg, \neg) G$

$\neg = (\neg, \neg, \neg) G$

$\neg = (\neg, \neg, \neg) G$

$\neg = (\neg, \neg, \neg) G$

$\neg = (\neg, \neg, \neg) G$

$\neg = (\neg, \neg, \neg) G$

$\neg = (\neg, \neg, \neg) G$

Let $G$ be a 3-place Boolean function defined as follows:

Example

Completeness of Propositional Connectives
Completeness of Propositional Connectives

Recall our definition of some basic Boolean functions:

\[ \land, \lor, \neg \]

Given that \( f: \lor \), \( f: \land \), \( f: \neg \), \( f: \lor \land \), and \( f: \land \lor \) is complete, it is not hard to see that any Boolean function can be constructed using only the Boolean functions \( \land, \lor, \neg \), and \( \lor \land \).

In fact, we can do better. It turns out that \( f: \lor \land \) and \( f: \land \lor \) are complete as well. Why?

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

Recall our definition of some basic Boolean functions:
Recall our definition of some basic Boolean functions:

<table>
<thead>
<tr>
<th>Boolean Function</th>
<th>Truth Table</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\lor$</td>
<td>$T$</td>
</tr>
<tr>
<td>$\land$</td>
<td>$F$</td>
</tr>
<tr>
<td>$\neg$</td>
<td>$F$</td>
</tr>
<tr>
<td>$\lor$</td>
<td>$T$</td>
</tr>
<tr>
<td>$\land$</td>
<td>$T$</td>
</tr>
<tr>
<td>$\neg$</td>
<td>$T$</td>
</tr>
</tbody>
</table>

Given that $\lor$, $\land$, and $\neg$ are complete, it is not hard to see that any Boolean function can be constructed using only the Boolean functions $\lor$, $\land$, and $\neg$.

In fact, we can do better. It turns out that $\land$, $\land$, $\lor$, and $\lor$ are complete as well.

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

Recall our definition of some basic Boolean functions:

\[ n = B \]
\[ N = B \]
\[ K = B \]
\[ A = B \]

Given that \( f; \land; \lor; \neg \) is complete, it is not hard to see that any Boolean function can be constructed using only the Boolean functions \( I, N, K, \) and \( A \).

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

Well, given that \( \land, \lor, \neg \) is complete, it is not hard to see that any Boolean function can be constructed using only the Boolean functions \( I, N, K, \) and \( A \).

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

Recall our definition of some basic Boolean functions:

\[ \forall \land \forall B = V \]
\[ \forall \lor \forall B = K \]
\[ \forall \neg B = N \]
\[ \forall A B = A \]

Completeness of Propositional Connectives
Completeness of Propositional Connectives

Recall our definition of some basic Boolean functions:

\[ I_n = B \]
\[ N = 1 \]
\[ K = 2 \]
\[ A_1 \land A_2 = B \]
\[ A_1 \lor A_2 = B \]

Given that \( f : \land \), \( \lor \), \( \land \), \( \lor \) are complete, then (\( \land \), \( \lor \)) are complete as well.

Why?

In fact, we can do better. It turns out that any Boolean function can be constructed using only the Boolean functions \( I \), \( N \), \( K \), \( \land \), \( \lor \).

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

\[(g \land (\land a) \land) \leftrightarrow g \lor a\]
\[(g \land (\lor a) \land) \leftrightarrow g \land a\]

Recall our definition of some basic Boolean functions:
To prove that some set of connectives is incomplete, we find a property that is true of all \textit{wffs} built using those connectives, but that is not true for some Boolean function.

Example: \(f \land \neg g\) is not complete.

Proof:

Let \(v\) be a \textit{wff} which uses only these connectives, and let \(v\) be a truth assignment such that \(v(A_i) = T\) for all \(A_i\).

We prove by induction that \(v(\cdot) = T\).

**Base Case**
\[ v(A_i) = v(A_i) = T. \]

**Inductive Case**
\[ v(\cdot \land \cdot) = \max(v(\cdot), v(\cdot)) = \max(T, T) = T. \]
\[ v(\cdot \land \cdot) = \max(v(\cdot), v(\cdot)) = \max(F, T) = T. \]

Thus, \(v(\cdot) = T\) for all \textit{wffs} built from \(f \land \neg g\). But \(v(\cdot \land \cdot) = F\), so there is no such formula autlogically equivalent to \(A_1\).
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.

Example

Let be a wff which uses only these connectives, and let be a truth assignment such that for all . We prove by induction that .

Base Case

\[ v(\mathcal{A}_1) = v(\mathcal{A}_1) = T . \]

Inductive Case

\[ v(\mathcal{\land}) = \max(v(\mathcal{\land}), v(\mathcal{\land})) = \max(T, T) = T . \]

\[ v(\mathcal{\lnot}) = \max(T, v(\mathcal{\lnot})) = \max(F, T) = T . \]

Thus, .

But , so there is no such formula autologically equivalent to .

Incompleteness of Connectives
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. Let \( \phi \) be a wff which uses only these connectives, and let \( \nu \) be a truth assignment such that \( \nu(A_i) = T \) for all \( A_i \). We prove by induction that \( \nu(\phi) = T \).

**Base Case**
\( \nu(A_i) = T \).

**Inductive Case**
\[ \nu(\phi) = \max(\nu(\phi_1), \nu(\phi_2)) = \max(T, T) = T \]
\[ \nu(\neg \phi) = \max(\neg \nu(\phi_1), \neg \nu(\phi_2)) = \max(F, T) = T \]

Thus, \( \nu(\phi) = T \) for all wffs built from \( \phi \). But \( \nu(A_1) = F \), so there is no such formula autologically equivalent to \( A_1 \).
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. Example function.

\[ \mathbf{T} = (\forall \forall) \mathbf{a} = (\forall \forall) \mathbf{a} \]

Base Case

\[ \mathbf{T} = (\forall \forall) \mathbf{a} = (\forall \forall) \mathbf{a} \]

Such that \forall \forall for all wff. Let \mathbf{a} be a wff which uses only these connectives, and let \mathbf{v} be a truth assignment. Proof.

Example

\[ \{ \mathbf{v}, \vee \} \]

is not complete.

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.
Incompleteness of Connectives

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

\[ \mathcal{L} = (\mathcal{L}, \mathcal{A}) \]

\[ \mathcal{L} = (\mathcal{L}, \mathcal{A})_{\text{max}} = (g\mathcal{A}, (\mathcal{A}) \mathcal{A} - \mathcal{L})_{\text{max}} = (\land \leftarrow g\mathcal{A}) \]

\[ \mathcal{L} = (\mathcal{L}, \mathcal{A})_{\text{max}} = ((\land \mathcal{A}, (g\mathcal{A}) \mathcal{A} = (\land \lor g\mathcal{A}) \]

Inductive Case

Base Case

\[ \mathcal{L} = (\exists \land \mathcal{A} = (\exists \land \mathcal{A} \]

Proof

Example

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

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

Example \( f, \neg \) is not complete.

Proof

Let \( \alpha \) be a \textit{wff} which uses only those connectives, and let \( \nu \) be a truth assignment such that \( \nu(A) = T \) for all \( A \).

We prove by induction that \( \nu = (\forall) \alpha \) for all \textit{wffs} built using those connectives.

Base case

\( \nu = (\exists) \alpha \) is not complete.

Inductive case

\( \nu = (\neg) \alpha \) is not complete.

Thus, \( \nu = (\forall) \alpha \) for all \textit{wffs} built using those connectives, but that is not true for some Boolean function.
For each $n$, there are $2^2^n$ different $n$-place Boolean functions $B(X_1, \ldots, X_n)$. Why?

$0$-ary connectives

There are two $0$-place Boolean functions: the constants $F$ and $T$. We can construct corresponding $0$-ary connectives $\bot$ and $\top$ with the meaning that $v(\bot) = F$ and $v(\top) = T$ regardless of the truth assignment $v$.

Unary connectives

There are four $1$-place functions, but these include the two constant functions mentioned above and the identity function. Thus the only additional connective of interest is negation: $\lnot$.

Binary connectives

There are sixteen $2$-place Boolean functions. They are cataloged in the following table. Note that the first six correspond to $0$-ary and unary connectives.
For each $n$, there are $2^{2n}$ different $n$-ary Boolean functions. Why?

There are $2^n$ different input points and 2 possible output values for each input point. $2^{2^n}$ is also the number of possible $n$-ary propositional connectives.

There are two 0-ary Boolean functions: the constants $F$ and $T$. We can construct corresponding 0-ary connectives $?$ and $>$ with the meaning that $v(?) = F$ and $v(>) = T$ regardless of the truth assignment $v$.

Unary connectives

There are four 1-ary functions, but these include the two constant functions mentioned above and the identity function. Thus, the only additional connective of interest is negation: $\neg$.

Binary connectives

There are sixteen 2-ary Boolean functions. They are catalogued in the following table. Note that the first six correspond to 0-ary and unary connectives.

Other Propositional Connectives
Other Propositional Connectives

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

0-ary connectives

There are two 0-place Boolean functions: the constants \( F \) and \( T \). We can construct corresponding 0-ary connectives: the constants \( F \) and \( T \).

Unary connectives

There are four 1-place functions, but these include the two constant functions mentioned above and the identity function. Thus, the only additional connective of interest is negation: \( \neg \).

Binary connectives

There are sixteen 2-place Boolean functions. They are cataloged in the following table. Note that the first six correspond to 0-ary and unary propositional connectives.

Why?

Regardless of the truth assignment \( a \), \( \bot = (\bot) a \) and \( 0 = (\top) a \). There are two 0-ary connectives: the constants \( F \) and \( T \).

Why?

For each \( u \), there are \( 2^{2^u} \) different \( u \)-place Boolean functions.
Other Propositional Connectives

For each $u$, there are $2^u$ different $u$-place Boolean functions.

Unary Connectives

There are two 0-place Boolean functions: the constants $\bot$ and $\top$.

Construction: For each truth assignment $\omega$, $\bot = (\bot)\omega$ and $\top = (\top)\omega$.

0-ary connectives

- $\bot$ and $\top$ correspond to the 0-ary connectives $\bot$ and $\top$.

1-ary Connectives

There are four 1-place functions. We can construct corresponding 1-ary connectives.

Binary Connectives

There are sixteen 2-place Boolean functions. They are cataloged in the following table.

Why?

$X_1, \ldots, X_u$
Other Propositional Connectives

For each \( n \), there are \( 2^2^n \) different \( n \)-place Boolean functions. They are cataloged in the following table.

Why? There are \( 2^2^n \) different input points and \( 2^2 \) possible output values for each input point.

\( 2^2^n \) is also the number of possible \( n \)-ary propositional connectives.

Why? For each \( u \), there are \( 2^2^n \) different \( u \)-place Boolean functions.
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Equivalent</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$&gt;$</td>
<td>$B \lor A$</td>
<td>greater than</td>
</tr>
<tr>
<td>$&lt;$</td>
<td>$A \lor B$</td>
<td>less than</td>
</tr>
<tr>
<td>$\neg$</td>
<td>$B \lor \neg A$</td>
<td>negation of first argument</td>
</tr>
<tr>
<td>$\neg$</td>
<td>$A \lor B$</td>
<td>negation of second argument</td>
</tr>
<tr>
<td>$\land$</td>
<td>$(B \lor A) \land (A \lor B)$</td>
<td>and</td>
</tr>
<tr>
<td>$\lor$</td>
<td>$(B \land A) \lor (A \land B)$</td>
<td>or</td>
</tr>
<tr>
<td>$\rightarrow$</td>
<td>$B \rightarrow A$</td>
<td>conditional</td>
</tr>
<tr>
<td>$\leftarrow$</td>
<td>$A \rightarrow B$</td>
<td>reverse conditional</td>
</tr>
<tr>
<td>$\leftrightarrow$</td>
<td>$B \leftrightarrow A$</td>
<td>bi-conditional</td>
</tr>
<tr>
<td>$\wedge$</td>
<td>$B \land A$</td>
<td>and</td>
</tr>
<tr>
<td>$\lor$</td>
<td>$B \lor A$</td>
<td>or</td>
</tr>
<tr>
<td>$\oplus$</td>
<td>$(B \lor A) \land (A \lor B)$</td>
<td>exclusive or</td>
</tr>
<tr>
<td>$\nabla$</td>
<td>$(B \land A) \lor (A \land B)$</td>
<td>nor (or Nicod stroke)</td>
</tr>
<tr>
<td>$:$</td>
<td>$(B \lor A) \land (A \lor B)$</td>
<td>nor (or Sheffer stroke)</td>
</tr>
<tr>
<td>$&lt;$</td>
<td>$B \lor A$</td>
<td>less than</td>
</tr>
<tr>
<td>$&gt;$</td>
<td>$A \lor B$</td>
<td>greater than</td>
</tr>
<tr>
<td>$\bot$</td>
<td>$\neg A$</td>
<td>constant $\top$</td>
</tr>
<tr>
<td>$\top$</td>
<td>$\neg B$</td>
<td>constant $\bot$</td>
</tr>
</tbody>
</table>
Recall that a wff α is satisfiable if there exists a truth assignment v such that

\[ \mathcal{J} = \{α\} \models v \]

Compactness

A set of wffs is satisfiable 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 \( \mathcal{J} \) is a set which is finitely satisfiable. We must show that \( \mathcal{J} \) is satisfiable.
Compactness

Recall that a wff $\varphi$ 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 $\mathfrak{L} = (\forall)v$ for each $\alpha \in \mathfrak{L}$.

A set of wffs is finitely satisfiable if every finite subset is satisfiable. The Compactness Theorem states that 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 $\mathfrak{L}$ is a set which is finitely satisfiable. We must show that $\mathfrak{L}$ is satisfiable. Recall that a wff $\varphi$ is satisfiable if there exists a truth assignment $v$ such that $\mathfrak{L} = (\forall)v$.  

\[ \mathfrak{L} = (\forall)v \]

such that

Recall that a wff $\varphi$ is satisfiable if there exists a truth assignment $v$ such that $\mathfrak{L} = (\forall)v$. 

\[ \mathfrak{L} = (\forall)v \]

such that

Compactness
Compactness

Recall that a wff is satisfiable if there exists a truth assignment \( v \) such that

\[ v(\phi) = T \]

A set of wffs is satisfiable if there exists a truth assignment \( v \) such that

\[ v(\chi) = T \]

for each \( \chi \in \chi \).

A set of wffs is satisfiable if there exists a truth assignment \( v \) such that

\[ v(\forall \alpha) = \top \]

Recall that a wff \( \alpha \) is satisfiable if there exists a truth assignment \( v \) such that

\[ v(\alpha) = \top \]

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 \( \Psi \) is a set which is finitely satisfiable. We must show that \( \Psi \) 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.

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

For each $\alpha \in \mathcal{S}$, let $\mathcal{L} = (\forall \alpha)$. Then $\mathcal{L}$ is satisfiable if there exists a truth assignment $v$ such that $v(\alpha) = T$ for each $\alpha \in \mathcal{S}$.

A set of wffs is satisfiable if and only if every finite subset of it is satisfiable.

Note that a truth assignment assigns a truth value to each variable in a formula.

A truth assignment for a wff $\alpha$ assigns a truth value $v(\alpha)$ to $\alpha$.

If $\alpha$ is a wff, then $v(\alpha)$ is either $T$ or $F$.

If $\mathcal{S}$ is satisfiable, then there exists a truth assignment $v$ such that $v(\alpha) = T$ for each $\alpha \in \mathcal{S}$.

Therefore, $\mathcal{S}$ is satisfiable.
A set of wffs is satisfiable if it is finitely satisfiable.

Compactness Theorem

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

Recall that a wff \( \phi \) is satisfiable if there exists a truth assignment \( \nu \) such that
\[
\mathcal{J} \models \phi\nu
\]
A set of wffs is satisfiable if it is finitely satisfiable.

Compactness Theorem

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

\[ \forall \phi \in \Sigma \quad \exists \sigma \text{ such that } \phi \in \sigma \]

Recall that a wff \( \phi \) is satisfiable if there exists a truth assignment \( \sigma \) such that...

Proof

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

To prove the other direction, assume that \( \Sigma \) is satisfiable.

We must show that \( \Sigma \) is satisfiable.
Let \( \phi_1, \phi_2, \ldots, \phi_n \) be a fixed enumeration of all \( \text{wffs} \).

Why is this possible?

Let \( \phi_1, \ldots, \phi_n \) be a fixed enumeration of all \( \text{wffs} \).

Let \( \phi_1 \) be finitely satisfiable. We extend \( \phi \) to form a \text{maximal finitely satisfiable set}.

Compactness
Compactness

Let \( \text{definitely satisfiable}. \) We extend to form a maximal definitly satisfiable set as follows.

Let \( 1 \); \( \vdots \); \( n \); \( \vdots \) be a fixed enumeration of all \( \text{wffs} \).

Why is this possible? The set of all sequences of a countable set is countable.

\( \exists \) is a countable, \( \vdots \); \( \exists \) an \( \exists \) be a fixed enumeration of all \( \text{wffs} \).

It is not hard to show that each \( \exists \) is definitly satisfiable.

Let \( \exists \) be definitly satisfiable. We extend to form a maximal definitly satisfiable set.

\[ \begin{align*}
1. & \\
2. & \text{or} \quad 2 \quad \text{or} \quad 2 \\
3. & \\
\end{align*} \]
Compactness

Let \( \text{a}\) be a fixed enumeration of all \(\text{wffs}\). Then, let \(\nabla\)'s be a maximal \(\text{finitely satisfiable}\) set. We extend \(\nabla\) to form a \text{maximal finitely satisfiable set}.

Why is this possible? The set of all sequences of a countable set is countable.

\[
\begin{align*}
\left \{ I^{+\infty} \cap \bigcup_{n} u \vdash \right \} \cap \bigcup_{n} u \vdash &= I^{+\infty} \\
\ldots &= 0 \vdash
\end{align*}
\]

Let \(\nabla\) be \text{finitely satisfiable}. We extend \(\nabla\) as follows.

\(\nabla\) as follows.

\(\nabla\) as follows.
Let $\exists$ be finitely satisfiable. We extend to a maximal finitely satisfiable set as follows.

Let $1, 2, \ldots, n, \ldots$ be an enumeration of all wffs. Then, let

\[
\exists = \bigcap_{n \in \mathbb{N}} \left( \bigcup_{\varphi \in \exists^{n}} \varphi \right)
\]

Why is this possible? The set of all sequences of a countable set is countable.

Let $\alpha, \beta, \ldots, \gamma$ be a fixed enumeration of all wffs. Let $\exists$ be finitely satisfiable. We extend to form a maximal finitely satisfiable set

Compa...
Now we show that $\exists \subseteq \forall$ is also satisfiable (and thus satisfiable).

Define a truth assignment $v$ as follows. For each propositional symbol $A$,

$$\exists \subseteq \forall \iff \exists (\forall A)^{v}$$

We claim that for any wff $\alpha$, $v$ satisfies $\alpha$ iff $\alpha \in v$. The proof is by induction on well-formed formulas.

Base Case: Follows directly from the definition of $v$.

Induction Case: We will just consider one case. Suppose $= \subseteq \wedge$. Then $v(\wedge) = T$ iff both $v(\wedge) = T$ and $v(\wedge) = T$ iff both $2$ and $2$.

Now, if both $2$ and $2$ are in $\wedge$, then since $; \wedge ; g$ is not satisfiable, we must have $2$.

Similarly, if one of $2$ or $2$ is not in $\wedge$, then its negation must be in $\wedge$, so $\notin 2$.
now we show that $\forall$ is satisfiable (and thus also satisfiable).

**Compactness**
We claim that for any \( \varphi \), \( \nu \) satisfies \( \varphi \) iff \( \exists! \nu \in \mathfrak{V} \nu \in \mathfrak{V} \). The proof is by induction on

\[ \nu \in \mathfrak{V} \Leftrightarrow (\exists! \nu) \varphi. \]

Define a truth assignment \( \nu \) as follows. For each propositional symbol \( \varphi \),

\[ \nu \in \mathfrak{V} \Leftrightarrow (\exists! \nu) \varphi. \]
Corollary

Compactness

If \( j = n \) then there is a finite \( n \) such that \( \varnothing \models \alpha \).

Proof

So, by compactness, \( \{ \alpha \} \cap \varnothing \) is satisfiable which contradicts the fact that

\[ \varnothing \models \alpha \Rightarrow \{ \alpha \} \cap \varnothing \models \alpha. \]

Then, \( \varnothing \models \{ \alpha \} \cap \varnothing \models \alpha \).

Suppose that \( \varnothing \models \{ \alpha \} \cap \varnothing \models \alpha \) for every finite \( \alpha \).

If \( \varnothing \models \alpha \) then there is a finite \( n \) such that \( \varnothing \models \alpha \).

\[ \square \]