G22.3033-002 Logic in Computer Science
Lecture 2
Review

Last week

- Induction and Recursion
- Propositional Logic: Syntax and Semantics
- Well-Formed Formulas (wff’s)
- Unique Readability Theorem for wff’s
- Recognizing Well-Formed Formulas
- Truth Tables
Outline

- Questions?
- Satisfiability and Tautologies
- Propositional Connectives and Boolean Functions
- Compactness
- Computability and Decidability

Material taken from Enderton: 1.2, 1.5, 1.7, as well as *Computability* by N. J. Cutland.
Propositional Logic: Syntax and Semantics

Well-Formed Formulas $W$

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

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

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

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

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

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

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

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

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

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

5-b
Definitions

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

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

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

Particular cases:

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

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

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

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

Particular cases:

- If $\emptyset \models \alpha$, then we say $\alpha$ is a tautology or $\alpha$ is valid and write $\models \alpha$.
- If $\Sigma$ is unsatisfiable, then $\Sigma \models \alpha$ for every wff $\alpha$. 
Definitions

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

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

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

Particular cases:

- If $\emptyset \models \alpha$, then we say $\alpha$ is a tautology or $\alpha$ is valid and write $\models \alpha$.
- If $\Sigma$ is unsatisfiable, then $\Sigma \models \alpha$ for every wff $\alpha$.
- If $\alpha \models \beta$ (shorthand for $\{\alpha\} \models \beta$) and $\beta \models \alpha$, then $\alpha$ and $\beta$ are tautologically equivalent.
Definitions

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

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

Suppose $\Sigma$ is a set of \textit{wff}'s. Then $\Sigma$ \textit{tautologically implies} $\alpha$, $\Sigma \models \alpha$, if every truth assignment which satisfies each formula in $\Sigma$ also satisfies $\alpha$.

Particular cases:

- If $\emptyset \models \alpha$, then we say $\alpha$ is a \textit{tautology} or $\alpha$ is \textit{valid} and write $\models \alpha$.
- If $\Sigma$ is \textit{unsatisfiable}, then $\Sigma \models \alpha$ for every \textit{wff} $\alpha$.
- If $\alpha \models \beta$ (shorthand for $\{\alpha\} \models \beta$) and $\beta \models \alpha$, then $\alpha$ and $\beta$ are \textit{tautologically equivalent}.
- $\Sigma \models \alpha$ if and only if $\bigwedge(\Sigma) \rightarrow \alpha$ is valid.
Examples

- \((A \lor B) \land (\neg A \lor \neg B)\)
Examples

- \((A \lor B) \land (\neg A \lor \neg B)\) is satisfiable, but not valid.
Examples

- \((A \lor B) \land (\neg A \lor \neg B)\) is satisfiable, but not valid.
- \((A \lor B) \land (\neg A \lor \neg B) \land (A \leftrightarrow B)\)
Examples

- \((A \lor B) \land (\neg A \lor \neg B)\) is satisfiable, but not valid.

- \((A \lor B) \land (\neg A \lor \neg B) \land (A \leftrightarrow B)\) is unsatisfiable.
Examples

- $(A \lor B) \land (\neg A \lor \neg B)$ is satisfiable, but not valid.
- $(A \lor B) \land (\neg A \lor \neg B) \land (A \leftrightarrow B)$ is unsatisfiable.
- $\{A, A \rightarrow B\} \models B$
- $\{A, \neg A\} \models (A \land \neg A)$
Examples

- \((A \lor B) \land (\neg A \lor \neg B)\) is satisfiable, but not valid.
- \((A \lor B) \land (\neg A \lor \neg B) \land (A \leftrightarrow B)\) is unsatisfiable.
- \(\{A, A \rightarrow B\} \models B\)
- \(\{A, \neg A\} \models (A \land \neg A)\)
- \(\neg(A \land B)\) is tautologically equivalent to \(\neg A \lor \neg B\)
Examples

- \((A \lor B) \land (\neg A \lor \neg B)\) is satisfiable, but not valid.
- \((A \lor B) \land (\neg A \lor \neg B) \land (A \leftrightarrow B)\) is unsatisfiable.
- \(\{A, A \rightarrow B\} \models B\)
- \(\{A, \neg A\} \models (A \land \neg A)\)
- \(\neg(A \land B)\) is tautologically equivalent to \(\neg A \lor \neg B\)

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

- \((A \lor B) \land (\neg A \lor \neg B)\) is satisfiable, but not valid.
- \((A \lor B) \land (\neg A \lor \neg B) \land (A \leftrightarrow B)\) is unsatisfiable.
- \(\{A, A \rightarrow B\} \models B\) \((A \land (A \rightarrow B) \land (\neg B))\)
- \(\{A, \neg A\} \models (A \land \neg A)\)
- \(\neg(A \land B)\) is tautologically equivalent to \(\neg A \lor \neg B\)

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

• \((A \lor B) \land (\neg A \lor \neg B)\) is satisfiable, but not valid.

• \((A \lor B) \land (\neg A \lor \neg B) \land (A \leftrightarrow B)\) is unsatisfiable.

• \(\{A, A \rightarrow B\} \models B \quad (A \land (A \rightarrow B) \land (\neg B))\)

• \(\{A, \neg A\} \models (A \land \neg A) \quad (A \land (\neg A) \land \neg(A \land \neg A))\)

• \(\neg (A \land B)\) is tautologically equivalent to \(\neg A \lor \neg B\)

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

- \((A \lor B) \land (\neg A \lor \neg B)\) is satisfiable, but not valid.

- \((A \lor B) \land (\neg A \lor \neg B) \land (A \leftrightarrow B)\) is unsatisfiable.

- \(\{A, A \rightarrow B\} \models B\quad (A \land (A \rightarrow B) \land (\neg B))\)

- \(\{A, \neg A\} \models (A \land \neg A)\quad (A \land (\neg A) \land \neg(A \land \neg A))\)

- \(\neg(A \land B)\) is tautologically equivalent to \(\neg A \lor \neg B\)
  
  \(\neg(\neg(A \land B) \leftrightarrow (\neg A \lor \neg B))\)

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

- \((A \lor B) \land (\neg A \lor \neg B)\) is satisfiable, but not valid.
- \((A \lor B) \land (\neg A \lor \neg B) \land (A \leftrightarrow B)\) is unsatisfiable.
- \(\{A, A \rightarrow B\} \models B\) \((A \land (A \rightarrow B) \land (\neg B))\)
- \(\{A, \neg A\} \models (A \land \neg A)\) \((A \land (\neg A) \land \neg(A \land \neg A))\)
- \(\neg(A \land B)\) is tautologically equivalent to \(\neg A \lor \neg B\)
  \(\neg(\neg(A \land B) \leftrightarrow (\neg A \lor \neg B))\)

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?
Examples

- \((A \lor B) \land (\neg A \lor \neg B)\) is satisfiable, but not valid.
- \((A \lor B) \land (\neg A \lor \neg B) \land (A \leftrightarrow B)\) is unsatisfiable.
- \(\{A, A \rightarrow B\} \models B\)  
  \((A \land (A \rightarrow B) \land (\neg B))\)
- \(\{A, \neg A\} \models (A \land \neg A)\)  
  \((A \land (\neg A) \land \neg(A \land \neg A))\)
- \(\neg(A \land B)\) is tautologically equivalent to \(\neg A \lor \neg B\)  
  \(\neg(\neg(A \land B) \leftrightarrow (\neg A \lor \neg B))\)

Now suppose you had an algorithm \textit{CHECKVALID} which returns \texttt{true} when \(\alpha\) is valid and \texttt{false} otherwise. How would you verify the claims given this algorithm?

Satisfiability and validity are dual notions: \(\alpha\) is unsatisfiable if and only if \(\neg\alpha\) is valid.
Determining Satisfiability using Truth Tables

An Algorithm for Satisfiability

To check whether $\alpha$ is satisfiable, form the truth table for $\alpha$. If there is a row in which 1 appears as the value for $\alpha$, then $\alpha$ is satisfiable. Otherwise, $\alpha$ is unsatisfiable.
Determining Satisfiability using Truth Tables

An Algorithm for Satisfiability

To check whether $\alpha$ is satisfiable, form the truth table for $\alpha$. If there is a row in which 1 appears as the value for $\alpha$, then $\alpha$ is satisfiable. Otherwise, $\alpha$ is unsatisfiable.

An Algorithm for Tautological Implication

To check whether $\{\alpha_1, \ldots, \alpha_k\} \models \beta$, check the satisfiability of $(\alpha_1 \land \ldots \land \alpha_k) \land (\neg \beta)$. If it is unsatisfiable, then $\{\alpha_1, \ldots, \alpha_k\} \models \beta$, otherwise $\{\alpha_1, \ldots, \alpha_k\} \not\models \beta$. 
Determining Satisfiability using Truth Tables

Example

\[ A \land ((B \lor \neg A) \land (C \lor \neg B)) \]
Determining Satisfiability using Truth Tables

Example

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

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

Example

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

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>A \land ((B \lor \neg A) \land (C \lor \neg B))</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---------------------------------</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Determining Satisfiability using Truth Tables

Example

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

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>A  \land ((B \lor \neg A) \land (C \lor \neg B))</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0   1  1  1  1  1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0   1  1  1  1  1</td>
</tr>
</tbody>
</table>
Determining Satisfiability using Truth Tables

Example

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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>( A \land ( ( B \lor \neg A ) \land ( C \lor \neg B ) ) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0</td>
</tr>
</tbody>
</table>
Determining Satisfiability using Truth Tables

Example

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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>( A \land ((B \lor \neg A) \land (C \lor \neg B)) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0 1 1 1 1 1 1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0 1 1 1 1 1 1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0 1 1 0 0 0 0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0 1 1 1 1 0 0</td>
</tr>
</tbody>
</table>
Determining Satisfiability using Truth Tables

Example

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

<table>
<thead>
<tr>
<th>( A )</th>
<th>( B )</th>
<th>( C )</th>
<th>( A \land ((B \lor \neg A) \land (C \lor \neg B)) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0 1 1 1 1 1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0 1 1 1 1 1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0 1 1 0 0 0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0 1 1 1 1 0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0 0 0 0 0 1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0 0 0 0 1 1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1 0 0 0 0 1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1 1 1 1 1 1</td>
</tr>
</tbody>
</table>
Determining Satisfiability using Truth Tables

Example

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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>( A \land ((B \lor \neg A) \land (C \lor \neg B)) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0 \quad 1 \quad 1 \quad 1 \quad 1 \quad 0 \quad 1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 1</td>
</tr>
</tbody>
</table>
Determining Satisfiability using Truth Tables

**Example**

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

<p>| | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>B</td>
<td>C</td>
<td>A</td>
<td>(B</td>
<td>\lor</td>
<td>\neg A</td>
</tr>
<tr>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Determining Satisfiability using Truth Tables

Example

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

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th>A \land ((B \lor \neg A) \land (C \lor \neg B))</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1 1 1 1 1 1</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1 1 1 1 1 1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1 1 0 0 0 0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1 1 1 1 0 0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0 0 0 0 1 1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0 0 0 0 1 1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1 0 0 0 0 0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1 0 1 1 1 0</td>
</tr>
</tbody>
</table>
Determining Satisfiability using Truth Tables

What is the complexity of this algorithm?
Determining Satisfiability using Truth Tables

What is the complexity of this algorithm?

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

Next lecture, we will discuss some of the applications and best-known techniques for the SAT algorithm.
Some tautologies

Associative and Commutative laws for $\land, \lor, \leftrightarrow$
Some tautologies

**Associative and Commutative laws for** $\land$, $\lor$, $\leftrightarrow$

**Distributive Laws**

- $(A \land (B \lor C')) \leftrightarrow ((A \land B) \lor (A \land C'))$.
- $(A \lor (B \land C')) \leftrightarrow ((A \lor B) \land (A \lor C'))$. 
Some tautologies

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

Distributive Laws

- $(A \land (B \lor C)) \leftrightarrow ((A \land B) \lor (A \land C))$.
- $(A \lor (B \land C)) \leftrightarrow ((A \lor B) \land (A \lor C))$.

Negation

- $\neg \neg A \leftrightarrow A$
- $\neg (A \rightarrow B) \leftrightarrow (A \land \neg B)$
- $\neg (A \leftrightarrow B) \leftrightarrow ((A \land \neg B) \lor (\neg A \land B))$
Some tautologies

**Associative and Commutative laws for** $\land$, $\lor$, $\leftrightarrow$

**Distributive Laws**

- $(A \land (B \lor C)) \leftrightarrow ((A \land B) \lor (A \land C))$.
- $(A \lor (B \land C)) \leftrightarrow ((A \lor B) \land (A \lor C))$.

**Negation**

- $\neg \neg A \leftrightarrow A$
- $\neg (A \rightarrow B) \leftrightarrow (A \land \neg B)$
- $\neg (A \leftrightarrow B) \leftrightarrow ((A \land \neg B) \lor (\neg A \land B))$

**De Morgan’s Laws**

- $\neg (A \land B) \leftrightarrow (\neg A \lor \neg B)$
- $\neg (A \lor B) \leftrightarrow (\neg A \land \neg B)$
More Tautologies

Implication

- \((A \rightarrow B) \iff (\neg A \vee B)\)
More Tautologies

Implication

• \((A \rightarrow B) \leftrightarrow (\neg A \vee B)\)

Excluded Middle

• \(A \vee \neg A\)
More Tautologies

Implication

- \((A \rightarrow B) \iff (\neg A \lor B)\)

Excluded Middle

- \(A \lor \neg A\)

Contradiction

- \(\neg (A \land \neg A)\)
More Tautologies

**Implication**

- \((A \rightarrow B) \leftrightarrow (\neg A \lor B)\)

**Excluded Middle**

- \(A \lor \neg A\)

**Contradiction**

- \(\neg (A \land \neg A)\)

**Contraposition**

- \((A \rightarrow B) \leftrightarrow (\neg B \rightarrow \neg A)\)
More Tautologies

**Implication**

- \((A \rightarrow B) \leftrightarrow (\neg A \lor B)\)

**Excluded Middle**

- \(A \lor \neg A\)

**Contradiction**

- \(\neg (A \land \neg A)\)

**Contraposition**

- \((A \rightarrow B) \leftrightarrow (\neg B \rightarrow \neg A)\)

**Exportation**

- \(((A \land B) \rightarrow C') \leftrightarrow (A \rightarrow (B \rightarrow C'))\)
Propositional Connectives

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

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

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 five connectives: $\neg$, $\land$, $\lor$, $\rightarrow$, $\leftrightarrow$. Would we gain anything by having more? Would we lose anything by having fewer?

**Example: Ternary Majority Connective $\#$**

$$\mathcal{E}_{\#}(\alpha, \beta, \gamma) = (\#\alpha\beta\gamma)$$

$$\overline{\mathcal{E}}((\#\alpha\beta\gamma)) = 1$$ iff the majority of $\overline{\mathcal{E}}(\alpha)$, $\overline{\mathcal{E}}(\beta)$, and $\overline{\mathcal{E}}(\gamma)$ are 1.
Propositional Connectives

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

**Example: Ternary Majority Connective \#**

\[
E_\#(\alpha, \beta, \gamma) = (\#\alpha\beta\gamma)
\]

\[
\overline{v}(\#\alpha\beta\gamma) = 1 \text{ iff the majority of } \overline{v}(\alpha), \overline{v}(\beta), \text{ and } \overline{v}(\gamma) \text{ are } 1.
\]

What does this new connective do for us?
Propositional Connectives

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 $\#$

$E_\#(\alpha, \beta, \gamma) = (\#\alpha\beta\gamma)$

$\overline{v}(\#\alpha\beta\gamma) = 1$ iff the majority of $\overline{v}(\alpha)$, $\overline{v}(\beta)$, and $\overline{v}(\gamma)$ are 1.

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.
Propositional Connectives

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

**Example: Ternary Majority Connective \#**

\[
\mathcal{E}_\#(\alpha, \beta, \gamma) = (\# \alpha \beta \gamma)
\]

\[
\overline{v}(\# \alpha \beta \gamma) = 1 \text{ iff the majority of } \overline{v}(\alpha), \overline{v}(\beta), \text{ and } \overline{v}(\gamma) \text{ are } 1.
\]

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?
Boolean Functions

For $k \geq 0$, a $k$-place Boolean function is a function from $\{0, 1\}^k$ to $\{0, 1\}$. A Boolean function then is anything which is a $k$-place Boolean function for some $k$.

Each wff $\alpha$ determines a corresponding Boolean function $B_\alpha$. For example, if $\alpha = A_1 \land A_2$, then $B_\alpha$ 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>$B_\alpha(X_1, X_2)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Realizing Boolean Functions

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 $B^n_\alpha$, the Boolean function realized by $\alpha$ as

$$B^n_\alpha(X_1, \ldots, X_n) = \text{the truth value given to } \alpha \text{ when } A_1, \ldots, A_n \text{ are given the values } X_1, \ldots, X_n.$$
Realizing Boolean Functions

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 $B^n_\alpha$, the Boolean function realized by $\alpha$ as

$$B^n_\alpha(X_1, \ldots, X_n) = \text{the truth value given to } \alpha \text{ when } A_1, \ldots, A_n \text{ are given the values } X_1, \ldots, X_n.$$ 

In other words,

$$B^n_\alpha(X_1, \ldots, X_n) = \overline{\nu}(\alpha) \text{ where } \nu(A_i) = X_i.$$
Realizing Boolean Functions

In general, suppose that $\alpha$ is a well-formed formula (wff) whose propositional symbols are included in $A_1, \ldots, A_n$. We define an $n$-place Boolean function $B^n_\alpha$, the Boolean function realized by $\alpha$ as

$$B^n_\alpha(X_1, \ldots, X_n) = \text{the truth value given to } \alpha \text{ when } A_1, \ldots, A_n \text{ are given the values } X_1, \ldots, X_n.$$ 

In other words,

$$B^n_\alpha(X_1, \ldots, X_n) = \overline{\nu(\alpha)} \text{ where } \nu(A_i) = X_i.$$ 

Note that the function $B^n_\alpha$ is determined by both the formula $\alpha$ and the choice of $n$. In particular, $\alpha$ does not need to include all the symbols in $A_1, \ldots, A_n$. 
Examples

- $I_i^n = B_{A_i}^n$
- $N = B_{\neg A_1}^1$
- $K = B_{A_1 \land A_2}^2$
- $A = B_{A_1 \lor A_2}^2$
- $C = B_{A_1 \rightarrow A_2}^2$
- $E = B_{A_1 \leftrightarrow A_2}^2$
Examples

- $I^n_i = B^n_{A_i}$
- $N = B^1_{\neg A_1}$
- $K = B^2_{A_1 \land A_2}$
- $A = B^2_{A_1 \lor A_2}$
- $C = B^2_{A_1 \to A_2}$
- $E = B^2_{A_1 \leftrightarrow A_2}$

From these functions, we can construct others by composition.

$$B^2_{\neg A_1 \lor \neg A_2}(X_1, X_2) = A(N(I^2_1(X_1, X_2)), N(I^2_2(X_1, X_2)))$$
Examples

- \( I_i^m = B_{A_i}^m \)
- \( N = B_{\neg A_1}^1 \)
- \( K = B_{A_1 \land A_2}^2 \)
- \( A = B_{A_1 \lor A_2}^2 \)
- \( C = B_{A_1 \rightarrow A_2}^2 \)
- \( E = B_{A_1 \leftrightarrow A_2}^2 \)

From these functions, we can construct others by composition.

\[
B_{\neg A_1 \lor \neg A_2}^2(X_1, X_2) = A(N(I_1^2(X_1, X_2)), N(I_2^2(X_1, X_2)))
\]

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.
Formulas and the Boolean Functions they Realize

Theorem

Let $\alpha$ and $\beta$ be wff's whose sentence symbols are among $A_1, \ldots, A_n$.

(a) $\alpha \models \beta$ iff $B^n_\alpha(\vec{X}) \leq B^n_\beta(\vec{X})$ for all $\vec{X} \in \{0, 1\}^n$.

(b) $\alpha$ is tautologically equivalent to $\beta$ iff $B^n_\alpha = B^n_\beta$.

(c) $\models \beta$ iff the range of $B^n_\beta = \{1\}$.
Formulas and the Boolean Functions they Realize

Theorem

Let $\alpha$ and $\beta$ be wff's whose sentence symbols are among $A_1, \ldots, A_n$.

(a) $\alpha \models \beta$ iff $B^n_\alpha(\vec{X}) \leq B^n_\beta(\vec{X})$ for all $\vec{X} \in \{0, 1\}^n$.

(b) $\alpha$ is tautologically equivalent to $\beta$ iff $B^n_\alpha = B^n_\beta$.

(c) $\models \beta$ iff the range of $B^n_\beta = \{1\}$.

Proof

(a) $\alpha \models \beta$ iff every truth assignment satisfying $\alpha$ also satisfies $\beta$
iff for every truth assignment $\nu$, $\nu(\alpha) = 1$ implies $\nu(\beta) = 1$
iff for all $n$-tuples $\vec{X}$, $B^n_\alpha(\vec{X}) = 1$ implies $B^n_\beta(\vec{X}) = 1$
iff for all $n$-tuples $\vec{X}$, $B^n_\alpha(\vec{X}) \leq B^n_\beta(\vec{X}) = 1$

(b) Follows from (a) and $X = Y$ iff $X \leq Y$ and $Y \leq X$.

(c) Follows from (a) and definition of tautology.
Formulas and the Boolean Functions they Realize

Theorem

Let \( \alpha \) and \( \beta \) be wff’s whose sentence symbols are among \( A_1, \ldots, A_n \).

(a) \( \alpha \models \beta \) iff \( B^n_\alpha(\vec{X}) \leq B^n_\beta(\vec{X}) \) for all \( \vec{X} \in \{0, 1\}^n \).

(b) \( \alpha \) is tautologically equivalent to \( \beta \) iff \( B^n_\alpha = B^n_\beta \).

(c) \( \models \beta \) iff the range of \( B^n_\beta = \{1\} \).

Proof

(a)
\[
\alpha \models \beta \quad \text{iff} \quad \text{every truth assignment satisfying } \alpha \text{ also satisfies } \beta \\
\quad \text{iff} \quad \text{for every truth assignment } \nu, \overline{\nu}(\alpha) = 1 \text{ implies } \overline{\nu}(\beta) = 1 \\
\quad \text{iff} \quad \text{for all } n\text{-tuples } \vec{X}, B^n_\alpha(\vec{X}) = 1 \text{ implies } B^n_\beta(\vec{X}) = 1 \\
\quad \text{iff} \quad \text{for all } n\text{-tuples } \vec{X}, B^n_\alpha(\vec{X}) \leq B^n_\beta(\vec{X}) = 1
\]

(b) Follows from (a) and \( X = Y \) iff \( X \leq Y \) and \( Y \leq X \).

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

By shifting our focus from formulas to Boolean functions, tautologically equivalent wff’s are identified.
Completeness of Propositional Connectives

**Theorem**

Let $G$ be an $n$-place Boolean function, $n \geq 1$. There exists a wff $\alpha$ such that $G = B^n_\alpha$, i.e., such that $\alpha$ realizes the function $G$. 
Completeness of Propositional Connectives

**Theorem**

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

**Proof**

If the range of $G$ is just $\{0\}$, then let $\alpha = A_1 \land \neg A_1$. Clearly, $B^n_{\alpha} = G$.

Otherwise, $G = 1$ somewhere. Suppose there are $k$ points where $G = 1$:

\[
\begin{align*}
G(X_{11}, X_{12}, \ldots, X_{1n}) &= 1 \\
G(X_{21}, X_{22}, \ldots, X_{2n}) &= 1 \\
&\vdots \\
G(X_{k1}, X_{k2}, \ldots, X_{kn}) &= 1
\end{align*}
\]

Let $\beta_{ij} = \begin{cases} A_j & \text{if } X_{ij} = 1 \\ \neg A_j & \text{if } X_{ij} = 0 \end{cases}$

$\gamma_i = \beta_{i1} \land \ldots \land \beta_{in}$

$\alpha = \gamma_1 \lor \gamma_2 \lor \ldots \lor \gamma_k$

Then $\alpha$ realizes $G$
Completeness of Propositional Connectives

Proof, continued

We know that $B^n_{\alpha}(\vec{X}) = \overline{v(\alpha)}$ where $v(A_i) = X_i$.

Since $\alpha = \gamma_1 \lor \gamma_2 \lor \ldots \lor \gamma_k$, it follows that $B^n_{\alpha}(\vec{X}) = \max(B^n_{\gamma_i}(\vec{X}))$.

But by construction, $B^n_{\gamma_i}(\vec{X}) = 1$ iff $\vec{X} =< X_{i_1}, \ldots, X_{i_n} >$.

Thus $B^n_{\alpha}(\vec{X}) = 1$ iff $\vec{X}$ is one of the points where $G$ is 1.

\[\square\]
Completeness of Propositional Connectives

Proof, continued

We know that $B^n_\alpha(\vec{X}) = \bar{v}(\alpha)$ where $v(A_i) = X_i$.

Since $\alpha = \gamma_1 \lor \gamma_2 \lor \ldots \lor \gamma_k$, it follows that $B^n_\alpha(\vec{X}) = \max(B^n_{\gamma_i}(\vec{X}))$.

But by construction, $B^n_{\gamma_i}(\vec{X}) = 1$ iff $\vec{X} =< X_i^1, \ldots, X_i^n >$.

Thus $B^n_\alpha(\vec{X}) = 1$ iff $\vec{X}$ is one of the points where $G$ is 1.

This shows that every Boolean function can be realized by a wff. In fact, every Boolean function can be realized by a wff which uses only the connectives $\{\neg, \land, \lor\}$. We say that this set of connectives is complete.

The realizing 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 which is a conjunction of literals, where a literal is either a propositional symbol or its negation.

Thus, a corollary is that for every wff, there exists a tautologically equivalent wff in disjunctive normal form.
Completeness of Propositional Connectives

Example

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

\[
\begin{align*}
G(0, 0, 0) &= 0 \\
G(0, 0, 1) &= 1 \\
G(0, 1, 0) &= 1 \\
G(0, 1, 1) &= 0 \\
G(1, 0, 0) &= 1 \\
G(1, 0, 1) &= 0 \\
G(1, 1, 0) &= 0 \\
G(1, 1, 1) &= 1
\end{align*}
\]
Completeness of Propositional Connectives

Example

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

\[
\begin{align*}
G(0, 0, 0) &= 0 \\
G(0, 0, 1) &= 1 \\
G(0, 1, 0) &= 1 \\
G(0, 1, 1) &= 0 \\
G(1, 0, 0) &= 1 \\
G(1, 0, 1) &= 0 \\
G(1, 1, 0) &= 0 \\
G(1, 1, 1) &= 1
\end{align*}
\]

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

\[
(\neg A_1 \land \neg A_2 \land A_3) \lor (\neg A_1 \land A_2 \land \neg A_3) \lor (A_1 \land \neg A_2 \land \neg A_3) \lor (A_1 \land A_2 \land A_3).
\]
Completeness of Propositional Connectives

Example

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

\[
\begin{align*}
G(0, 0, 0) &= 0 \\
G(0, 0, 1) &= 1 \\
G(0, 1, 0) &= 1 \\
G(0, 1, 1) &= 0 \\
G(1, 0, 0) &= 1 \\
G(1, 0, 1) &= 0 \\
G(1, 1, 0) &= 0 \\
G(1, 1, 1) &= 1
\end{align*}
\]

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

\[
(\neg A_1 \land \neg A_2 \land A_3) \lor (\neg A_1 \land A_2 \land \neg A_3) \lor (A_1 \land \neg A_2 \land \neg A_3) \lor (A_1 \land A_2 \land A_3).
\]

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

Recall our definition of some basic Boolean functions:

- \( I^m_i = B^n_{A_i} \)
- \( N = B^1_{\neg A_1} \)
- \( K = B^2_{A_1 \land A_2} \)
- \( A = B^2_{A_1 \lor A_2} \)

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

Recall our definition of some basic Boolean functions:

- $I_i^m = B^n_{A_i}$
- $N = B^{1}_{\neg A_1}$
- $K = B^{2}_{A_1 \land A_2}$
- $A = B^{2}_{A_1 \lor A_2}$

Given that $\{\neg, \land, \lor\}$ 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 $\{\neg, \land\}$ and $\{\neg, \lor\}$ are complete as well.
Completeness of Propositional Connectives

Recall our definition of some basic Boolean functions:

- \( I_i^m = B_{A_i}^n \)
- \( N = B_{\neg A_1}^1 \)
- \( K = B_{A_1 \land A_2}^2 \)
- \( A = B_{A_1 \lor A_2}^2 \)

Given that \( \{\neg, \land, \lor\} \) 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 \( \{\neg, \land\} \) and \( \{\neg, \lor\} \) are complete as well.

Why?
Completeness of Propositional Connectives

Recall our definition of some basic Boolean functions:

- \( I_i^m = B_{A_i}^n \)
- \( N = B_{\lnot A_1}^1 \)
- \( K = B_{A_1 \land A_2}^2 \)
- \( A = B_{A_1 \lor A_2}^2 \)

Given that \( \{ \lnot, \land, \lor \} \) 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 \( \{ \lnot, \land \} \) and \( \{ \lnot, \lor \} \) are complete as well.

Why?

\[
\alpha \lor \beta \iff \lnot(\lnot \alpha \land \lnot \beta)
\]
\[
\alpha \land \beta \iff \lnot(\lnot \alpha \lor \lnot \beta)
\]

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

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

**Example**

\( \{ \land, \rightarrow \} \) is not complete.
Incompleteness of Connectives

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

Example

\{\land, \rightarrow\} is not complete.

Proof

Let \(\alpha\) be a wff which uses only these connectives, and let \(\nu\) be a truth assignment such that \(\nu(A_i) = 1\) for all \(A_i\). We prove by induction that \(\overline{\nu}(\alpha) = 1\).
Incompleteness of Connectives

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

Example

$\{\land, \rightarrow\}$ is not complete.

Proof

Let $\alpha$ be a wff which uses only these connectives, and let $\nu$ be a truth assignment such that $\nu(A_i) = 1$ for all $A_i$. We prove by induction that $\overline{\nu}(\alpha) = 1$.

Base Case

$\overline{\nu}(A_i) = \nu(A_i) = 1$. 
Incompleteness of Connectives

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

Example

\{\land, \rightarrow\} is not complete.

Proof

Let \(\alpha\) be a wff which uses only these connectives, and let \(\nu\) be a truth assignment such that \(\nu(A_i) = 1\) for all \(A_i\). We prove by induction that \(\overline{\nu}(\alpha) = 1\).

Base Case

\(\overline{\nu}(A_i) = \nu(A_i) = 1\).

Inductive Case

\[\begin{align*}
\overline{\nu}(\beta \land \gamma) &= \max(\overline{\nu}(\beta), \overline{\nu}(\gamma)) = \max(1, 1) = 1 \\
\overline{\nu}(\beta \rightarrow \gamma) &= \max(1 - \overline{\nu}(\alpha), \overline{\nu}(\beta)) = \max(0, 1) = 1
\end{align*}\]
Incompleteness of Connectives

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

Example

\{\land, \rightarrow\} is not complete.

Proof

Let \(\alpha\) be a wff which uses only these connectives, and let \(\nu\) be a truth assignment such that \(\nu(A_i) = 1\) for all \(A_i\). We prove by induction that \(\bar{\nu}(\alpha) = 1\).

Base Case

\(\bar{\nu}(A_i) = \nu(A_i) = 1\).

Inductive Case

\[
\begin{align*}
\bar{\nu}(\beta \land \gamma) &= \max(\bar{\nu}(\beta), \bar{\nu}(\gamma)) = \max(1, 1) = 1 \\
\bar{\nu}(\beta \rightarrow \gamma) &= \max(1 - \bar{\nu}(\alpha), \bar{\nu}(\beta)) = \max(0, 1) = 1
\end{align*}
\]

Thus, \(\bar{\nu}(\alpha) = 1\) for all wff’s \(\alpha\) built from \{\land, \rightarrow\}. But \(\bar{\nu}(\neg A_1) = 0\), so there is no such formula tautologically equivalent to \(\neg A_1\).  

\(\square\)
Other Propositional Connectives

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

Why?
Other Propositional Connectives

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

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.
Other Propositional Connectives

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

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.

0-ary connectives

There are two 0-place Boolean functions: the constants 0 and 1. We can construct corresponding 0-ary connectives \( \perp \) and \( \top \) with the meaning that \( \overline{v}(\perp) = 0 \) and \( \overline{v}(\top) = 1 \) regardless of the truth assignment \( v \).
Other Propositional Connectives

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

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.

0-ary connectives

There are two 0-place Boolean functions: the constants 0 and 1. We can construct corresponding 0-ary connectives $\bot$ and $\top$ with the meaning that $\overline{\nu}(\bot) = 0$ and $\overline{\nu}(\top) = 1$ regardless of the truth assignment $\nu$.

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: $\neg$. 
Other Propositional Connectives

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

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.

0-ary connectives

There are two 0-place Boolean functions: the constants 0 and 1. We can construct corresponding 0-ary connectives $\bot$ and $\top$ with the meaning that $\overline{v}(\bot) = 0$ and $\overline{v}(\top) = 1$ 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: $\neg$.

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 connectives.
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Equivalent</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>⊥</td>
<td>constant 0</td>
<td>projection of first argument</td>
</tr>
<tr>
<td>⊤</td>
<td>constant 1</td>
<td>projection of second argument</td>
</tr>
<tr>
<td>A</td>
<td></td>
<td>negation of first argument</td>
</tr>
<tr>
<td>B</td>
<td></td>
<td>negation of second argument</td>
</tr>
<tr>
<td>∧</td>
<td>A ∧ B</td>
<td>and</td>
</tr>
<tr>
<td>∨</td>
<td>A ∨ B</td>
<td>or</td>
</tr>
<tr>
<td>→</td>
<td>A → B</td>
<td>conditional</td>
</tr>
<tr>
<td>⇔</td>
<td>A ⇔ B</td>
<td>bi-conditional</td>
</tr>
<tr>
<td>←</td>
<td>B → A</td>
<td>reverse conditional</td>
</tr>
<tr>
<td>⊕</td>
<td>(A ∧ ¬B) ∨ (¬A ∧ B)</td>
<td>exclusive or</td>
</tr>
<tr>
<td>↓</td>
<td>¬(A ∨ B)</td>
<td>nor (or Nicod stroke)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>¬(A ∧ B)</td>
</tr>
<tr>
<td>&lt;</td>
<td>¬A ∧ B</td>
<td>less than</td>
</tr>
<tr>
<td>&gt;</td>
<td>A ∧ ¬B</td>
<td>greater than</td>
</tr>
</tbody>
</table>
Compactness

Recall that a \textit{wff} $\alpha$ is satisfiable if there exists a truth assignment $\nu$ such that $\nu(\alpha) = 1$. 

Compactness Theorem

A set of wff's 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 $\beta$ is a set which is finitely satisfiable. We must show that $\alpha$ is satisfiable.
Compactness

Recall that a wff $\alpha$ is satisfiable if there exists a truth assignment $\nu$ such that $\nu(\alpha) = 1$.

A set $\Sigma$ of wff’s is satisfiable if there exists a truth assignment $\nu$ such that $\nu(\alpha) = 1$ for each $\alpha \in \Sigma$.
Compactness

Recall that a wff $\alpha$ is satisfiable if there exists a truth assignment $\nu$ such that $\nu(\alpha) = 1$.

A set $\Sigma$ of wff’s is satisfiable if there exists a truth assignment $\nu$ such that $\nu(\alpha) = 1$ for each $\alpha \in \Sigma$.

A set $\Sigma$ is \textit{finitely satisfiable} iff every finite subset of $\Sigma$ is satisfiable.
Compactness

Recall that a wff $\alpha$ is satisfiable if there exists a truth assignment $\nu$ such that $\overline{\nu}(\alpha) = 1$.

A set $\Sigma$ of wff’s is satisfiable if there exists a truth assignment $\nu$ such that $\overline{\nu}(\alpha) = 1$ for each $\alpha \in \Sigma$.

A set $\Sigma$ is *finitely satisfiable* iff every finite subset of $\Sigma$ is satisfiable.

**Compactness Theorem**

A set of wff’s is satisfiable iff it is finitely satisfiable.
Compactness

Recall that a \textit{wff} $\alpha$ is satisfiable if there exists a truth assignment $\nu$ such that $\overline{\nu}(\alpha) = 1$.

A set $\Sigma$ of \textit{wff}'s is satisfiable if there exists a truth assignment $\nu$ such that $\overline{\nu}(\alpha) = 1$ for each $\alpha \in \Sigma$.

A set $\Sigma$ is \textit{finitely satisfiable} iff every finite subset of $\Sigma$ is satisfiable.

\textbf{Compactness Theorem}

A set of \textit{wff}'s is satisfiable iff it is finitely satisfiable.

\textbf{Proof}

The \textit{only if} direction is trivial since any subset of a satisfiable set is clearly satisfiable.
Compactness

Recall that a wff $\alpha$ is satisfiable if there exists a truth assignment $\nu$ such that $\nu(\alpha) = 1$.

A set $\Sigma$ of wff’s is satisfiable if there exists a truth assignment $\nu$ such that $\nu(\alpha) = 1$ for each $\alpha \in \Sigma$.

A set $\Sigma$ is \textit{finitely satisfiable} iff every finite subset of $\Sigma$ is satisfiable.

Compactness Theorem

A set of wff’s is satisfiable iff it is finitely satisfiable.

Proof

The \textit{only if} direction is trivial since any subset of a satisfiable set is clearly satisfiable.

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

Let $\Sigma$ be finitely satisfiable. We extend $\Sigma$ to form a *maximal* finitely satisfiable set $\Delta$ as follows.

Let $\alpha_1, \ldots, \alpha_n, \ldots$ be a fixed enumeration of all wff’s.

Why is this possible?
Compactness

Let $\Sigma$ be finitely satisfiable. We extend $\Sigma$ to form a maximal finitely satisfiable set $\Delta$ as follows.

Let $\alpha_1, \ldots, \alpha_n, \ldots$ be a fixed enumeration of all wff's.

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

Let $\Sigma$ be finitely satisfiable. We extend $\Sigma$ to form a *maximal* finitely satisfiable set $\Delta$ as follows.

Let $\alpha_1, \ldots, \alpha_n, \ldots$ be a fixed enumeration of all wff’s.

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

Then, let

\[
\Delta_0 = \Sigma,
\]

\[
\Delta_{n+1} = \begin{cases} 
\Delta_n \cup \{\alpha_{n+1}\} & \text{if this is finitely satisfiable}, \\
\Delta_n \cup \{\neg \alpha_{n+1}\} & \text{otherwise}.
\end{cases}
\]
Compactness

Let $\Sigma$ be finitely satisfiable. We extend $\Sigma$ to form a *maximal* finitely satisfiable set $\Delta$ as follows.

Let $\alpha_1, \ldots, \alpha_n, \ldots$ be a fixed enumeration of all wff’s.

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

Then, let

$$\Delta_0 = \Sigma,$$

$$\Delta_{n+1} = \begin{cases} 
\Delta_n \cup \{\alpha_{n+1}\} & \text{if this is finitely satisfiable,} \\
\Delta_n \cup \{\neg\alpha_{n+1}\} & \text{otherwise.}
\end{cases}$$

It is not hard to show that each $\Delta_n$ is finitely satisfiable (homework problem).

Let $\Delta = \bigcup_n \Delta_n$. It is then clear that

1. $\Sigma \subseteq \Delta$
2. $\alpha \in \Delta$ or $\neg\alpha \in \Delta$ for any wff $\alpha$, and
3. $\Delta$ is finitely satisfiable.
Compactness

Now we show that $\Delta$ is satisfiable (and thus $\Sigma \subseteq \Delta$ is also satisfiable).

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

$$v(A_i) = 1 \text{ iff } A_i \in \Delta.$$ 

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

Now we show that $\Delta$ is satisfiable (and thus $\Sigma \subseteq \Delta$ is also satisfiable).

Define a truth assignment $\nu$ as follows. For each propositional symbol $A_i$,

$$\nu(A_i) = 1 \text{ iff } A_i \in \Delta.$$  

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

**Base Case**

Follows directly from the definition of $\nu$. 

Compactness

Now we show that $\Delta$ is satisfiable (and thus $\Sigma \subseteq \Delta$ is also satisfiable).

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

$$v(A_i) = 1 \text{ iff } A_i \in \Delta.$$  

We claim that for any wff $\alpha$, $v$ satisfies $\alpha$ iff $\alpha \in \Delta$. 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 $\alpha = \beta \land \gamma$. Then

$$v(\alpha) = 1 \text{ iff both } v(\beta) = 1 \text{ and } v(\gamma) = 1 \text{ iff both } \beta \in \Delta \text{ and } \gamma \in \Delta.$$  

Now, if both $\beta$ and $\gamma$ are in $\Delta$, then since $\{\beta, \gamma, \neg \alpha\}$ is not satisfiable, we must have $\alpha \in \Delta$.

Similarly, if one of $\beta$ or $\gamma$ is not in $\Delta$, then its negation must be in $\Delta$, so $\alpha \notin \Delta$. \(\square\)
Compactness

Corollary

If \( \Sigma \models \alpha \) then there is a finite \( \Sigma_0 \subseteq \Sigma \) such that \( \Sigma_0 \models \alpha \).

Proof

Suppose that \( \Sigma_0 \not\models \alpha \) for every finite \( \Sigma_0 \subseteq \Sigma \).

Then, \( \Sigma_0 \cup \{\neg \alpha\} \) is satisfiable for every finite \( \Sigma_0 \subseteq \Sigma \).

So, by compactness, \( \Sigma \cup \{\neg \alpha\} \) is satisfiable which contradicts the fact that \( \Sigma \models \alpha \).

\[ \square \]
Computability

The important notion of *computability* relies on a formal model of computation.

Many formal models have been proposed:

1. General recursive functions defined by means of an equation calculus (Gödel-Herbrand-Kleene)
2. \(\lambda\)-definable functions (Church)
3. \(\mu\)-recursive functions and partial recursive functions (Gödel-Kleene)
4. Functions computable by finite machines known as Turing machines (Turing)
5. Functions defined from canonical deduction systems (Post)
6. Functions given by certain algorithms over a finite alphabet (Markov)
7. Universal Register Machine-computable functions (Shepherdson-Sturgis)

**Fundamental Result**

All of these (and many other) models of computation are equivalent. That is, they give rise to the same class of functions.
Computability and Decidability

All of these models are equivalent to what can be achieved by a computer with any standard programming language, given arbitrary (but finite) time and memory.

Church’s Thesis

A notion known as Church’s thesis states that all models of computation are either equivalent to or less powerful than those just described.

We will accept Church’s thesis and thus define a function to be *computable* if we can describe precisely (using any model of computation) how to compute it. Such a description will be called an *effective procedure*. 
Computability and Decidability

All of these models are equivalent to what can be achieved by a computer with any standard programming language, given arbitrary (but finite) time and memory.

Church’s Thesis

A notion known as Church’s thesis states that all models of computation are either equivalent to or less powerful than those just described.

We will accept Church’s thesis and thus define a function to be computable if we can describe precisely (using any model of computation) how to compute it. Such a description will be called an effective procedure.

Decidability

Given a universal set $U$, a set $S \subseteq U$ is decidable if there exists a computable function $f : U \rightarrow \{0, 1\}$ such that $f(x) = 1$ iff $x \in S$. 

Computability and Decidability

All of these models are equivalent to what can be achieved by a computer with any standard programming language, given arbitrary (but finite) time and memory.

Church’s Thesis

A notion known as Church’s thesis states that all models of computation are either equivalent to or less powerful than those just described.

We will accept Church’s thesis and thus define a function to be computable if we can describe precisely (using any model of computation) how to compute it. Such a description will be called an effective procedure.

Decidability

Given a universal set $U$, a set $S \subseteq U$ is decidable if there exists a computable function $f : U \rightarrow \{0, 1\}$ such that $f(x) = 1$ iff $x \in S$.

Decidability of $W$

Earlier, we presented an algorithm which, given any expression $\alpha$ determines whether the expression is well-formed. Thus, the set $W$ of well-formed formulas is decidable.
Decidability

Some decidable sets

- For a given finite set of wff's $\Sigma$, the set of all \textit{tautological consequences} of $\Sigma$ (i.e. $\{\alpha \mid \Sigma \models \alpha\}$) is decidable.
Decidability

Some decidable sets

- For a given finite set of wff’s $\Sigma$, the set of all tautological consequences of $\Sigma$ (i.e. $\{\alpha \mid \Sigma \models \alpha\}$) is decidable.

  The truth table algorithm given earlier decides $\Sigma \models \alpha$. 
Decidability

Some decidable sets

- For a given finite set of wff’s $\Sigma$, the set of all tautological consequences of $\Sigma$ (i.e. $\{\alpha \mid \Sigma \models \alpha\}$) is decidable.
  
The truth table algorithm given earlier decides $\Sigma \models \alpha$.
- The set of tautologies is decidable.
Decidability

Some decidable sets

- For a given finite set of wff's $\Sigma$, the set of all tautological consequences of $\Sigma$ (i.e. $\{\alpha \mid \Sigma \models \alpha\}$) is decidable.
  The truth table algorithm given earlier decides $\Sigma \models \alpha$.

- The set of tautologies is decidable.
  The set of tautologies is just the set of tautological consequences of the empty set.
Decidability

Some decidable sets

- For a given finite set of wff’s $\Sigma$, the set of all tautological consequences of $\Sigma$ (i.e. $\{\alpha | \Sigma \models \alpha\}$) is decidable.
  
  The truth table algorithm given earlier decides $\Sigma \models \alpha$.

- The set of tautologies is decidable.
  
  The set of tautologies is just the set of tautological consequences of the empty set.

Existence of undecidable sets

A simple argument shows the existence of undecidable sets of expressions: an algorithm is completely determined by its finite description. Thus, there are only countably many effective procedures. But there are uncountably many sets of expressions.

Why?
Decidability

Some decidable sets

- For a given finite set of wff's $\Sigma$, the set of all tautological consequences of $\Sigma$ (i.e. $\{\alpha \mid \Sigma \models \alpha\}$) is decidable.
  
  The truth table algorithm given earlier decides $\Sigma \models \alpha$.

- The set of tautologies is decidable.
  
  The set of tautologies is just the set of tautological consequences of the empty set.

Existence of undecidable sets

A simple argument shows the existence of undecidable sets of expressions: an algorithm is completely determined by its finite description. Thus, there are only countably many effective procedures. But there are uncountably many sets of expressions.

Why?

The set of expressions is countably infinite. Therefore, its power set is uncountable.
Semi-Decidability

Suppose we wish to determine whether $\Sigma \models \alpha$ where $\Sigma$ is infinite. In general, this is not decidable.

But we can obtain a weaker result:

A set $A$ is semi-decidable (or effectively enumerable) if there is an effective procedure which lists, in some order, every member of $A$.

Note that if $A$ is infinite, then the procedure will never finish, but every member of $A$ must appear in the list after some finite amount of time.
Semi-Decidability

Suppose we wish to determine whether $\Sigma \models \alpha$ where $\Sigma$ is infinite. In general, this is not decidable.

But we can obtain a weaker result:

A set $A$ is *semi-decidable* (or *effectively enumerable*) if there is an effective procedure which lists, in some order, every member of $A$.

Note that if $A$ is infinite, then the procedure will never finish, but every member of $A$ must appear in the list after some finite amount of time.

**Theorem**

A set $A$ of expressions is effectively enumerable iff there is an effective procedure which, given any expression $\alpha$, produces the answer “yes” iff $\alpha \in A$. 

Semi-Decidability

Suppose we wish to determine whether $\Sigma \models \alpha$ where $\Sigma$ is infinite. In general, this is not decidable.

But we can obtain a weaker result:

A set $A$ is **semi-decidable** (or **effectively enumerable**) if there is an effective procedure which lists, in some order, every member of $A$.

Note that if $A$ is infinite, then the procedure will never finish, but every member of $A$ must appear in the list after some finite amount of time.

**Theorem**

A set $A$ of expressions is effectively enumerable iff there is an effective procedure which, given any expression $\alpha$, produces the answer “yes” iff $\alpha \in A$.

**Proof**

If $A$ is effectively enumerable, then we simply enumerate its members and check each one to see if it is equivalent to $\alpha$. If it is, we return “yes” and stop. Otherwise, we keep going. Thus, if $\alpha \in A$, the procedure produces “yes”. If $\alpha \notin A$, the procedure runs forever.
Proof, continued

On the other hand, suppose that we have an effective procedure $P$ which produces “yes” iff $\alpha \in A$. To produce an enumeration of $A$, we proceed as follows. First enumerate all expressions:

\[ \epsilon_1, \epsilon_2, \epsilon_3, \ldots \]

Then proceed as follows.

- Break the procedure $P$ into a finite number of “steps”.
- Run $P$ on $\epsilon_1$ for 1 step.
- Run $P$ on $\epsilon_1$ for 2 steps, and then run $P$ on $\epsilon_2$ for 2 steps.
- \ldots
- Run $P$ on each of $\epsilon_1, \ldots, \epsilon_n$ for $n$ steps each
- \ldots

If at any time, the procedure $P$ produces “yes”, then we list the expression which produced “yes” and continue.

This procedure will eventually enumerate all members of $A$.  \qed
Semi-Decidability

Theorem

A set is decidable iff both it and its complement (with respect to a given universal set) are effectively enumerable.
Semi-Decidability

Theorem

A set is decidable iff both it and its complement (with respect to a given universal set) are effectively enumerable.

Proof

Alternate between running the procedure for the set and the procedure for its complement. One of them will eventually produce “yes”.
Semi-Decidability

Theorem

A set is decidable iff both it and its complement (with respect to a given universal set) are effectively enumerable.

Proof

Alternate between running the procedure for the set and the procedure for its complement. One of them will eventually produce “yes”.

Properties of decidable and semi-decidable sets

Decidable sets are closed under union, intersection, and complement.

Semi-decidable sets are closed under union and intersection.
Semi-Decidability

Theorem

If $\Sigma$ is an effectively enumerable set of wff's, then the set of tautological consequences of $\Sigma$ is effectively enumerable.
Semi-Decidability

Theorem

If $\Sigma$ is an effectively enumerable set of wff’s, then the set of tautological consequences of $\Sigma$ is effectively enumerable.

Proof

Consider an enumeration of the elements of $\Sigma$:

$$\sigma_1, \sigma_2, \sigma_3, \ldots$$

By the compactness theorem, $\Sigma \models \alpha$ iff $\{\sigma_1, \ldots, \sigma_n\} \models \alpha$ for some $n$.

Hence, it is sufficient to successively test:

$$\emptyset \models \alpha$$

$$\{\sigma_1\} \models \alpha$$

$$\{\sigma_1, \sigma_2\} \models \alpha$$

$$\ldots$$

If any of these conditions is met (each of which is decidable), the answer is “yes”.
Semi-Decidability

Theorem

If $\Sigma$ is an effectively enumerable set of wff’s, then the set of tautological consequences of $\Sigma$ is effectively enumerable.

Proof (continued)

This demonstrates that there is an effective procedure that, given any wff $\alpha$, will output “yes” iff $\alpha$ is a tautological consequence of $\Sigma$.

Thus, $\Sigma$ is effectively enumerable. $\square$