Mathematical Logic

Kerstin Eder

University of Bristol
Department of Computer Science

Kerstin.Eder@bristol.ac.uk

Overview

- In our last lecture we considered the connectives $\wedge$, $\vee$, $\oplus$, $\neg$, $\Rightarrow$ and $\Leftrightarrow$.

Overview

- In our last lecture we considered the connectives $\wedge$, $\vee$, $\oplus$, $\neg$, $\Rightarrow$ and $\Leftrightarrow$.
- We formally defined the Language of Propositional Logic.

Overview

- In our last lecture we considered the connectives $\wedge$, $\vee$, $\oplus$, $\neg$, $\Rightarrow$ and $\Leftrightarrow$.
- We formally defined the Language of Propositional Logic.
- Today we look in detail at truth tables for compound propositions.

Overview

- In our last lecture we considered the connectives $\wedge$, $\vee$, $\oplus$, $\neg$, $\Rightarrow$ and $\Leftrightarrow$.
- We formally defined the Language of Propositional Logic.
- Today we look in detail at truth tables for compound propositions.
- We explain the notions of validity and satisfiability.

Overview

- In our last lecture we considered the connectives $\wedge$, $\vee$, $\oplus$, $\neg$, $\Rightarrow$ and $\Leftrightarrow$.
- We formally defined the Language of Propositional Logic.
- Today we look in detail at truth tables for compound propositions.
- We explain the notions of validity and satisfiability.
- We investigate logic equivalence.

Overview

- In our last lecture we considered the connectives $\wedge$, $\vee$, $\oplus$, $\neg$, $\Rightarrow$ and $\Leftrightarrow$.
- We formally defined the Language of Propositional Logic.
- Today we look in detail at truth tables for compound propositions.
- We explain the notions of validity and satisfiability.
- We investigate logic equivalence.
- We learn that there are sets of functionally complete connectives.

Overview

- In our last lecture we considered the connectives $\wedge$, $\vee$, $\oplus$, $\neg$, $\Rightarrow$ and $\Leftrightarrow$.
- We formally defined the Language of Propositional Logic.
- Today we look in detail at truth tables for compound propositions.
- We explain the notions of validity and satisfiability.
- We investigate logic equivalence.
- We learn that there are sets of functionally complete connectives.
- (We give the semantics of propositional logic.)

Language of Propositional Logic: Well-Formed Formulae

The alphabet consist of the following sets:

Language of Propositional Logic: Well-Formed Formulae

The alphabet consist of the following sets:

- A set of propositional variables $PROP = \{p, q, r, s, ...\}$.

Language of Propositional Logic: Well-Formed Formulae

The alphabet consist of the following sets:

- A set of propositional variables $PROP = \{p, q, r, s, ...\}$.
- A set of propositional connectives $\{\textbf{true}, \textbf{false}, \neg, \wedge, \vee, \oplus, \Rightarrow, \Leftrightarrow\}$.

Language of Propositional Logic: Well-Formed Formulae

The alphabet consist of the following sets:

- A set of propositional variables $PROP = \{p, q, r, s, ...\}$.
- A set of propositional connectives $\{\textbf{true}, \textbf{false}, \neg, \wedge, \vee, \oplus, \Rightarrow, \Leftrightarrow\}$.
- A set of punctuation symbols $\{(,)\}$.

Language of Propositional Logic: Well-Formed Formulae

The alphabet consist of the following sets:

- A set of propositional variables $PROP = \{p, q, r, s, ...\}$.
- A set of propositional connectives $\{\mathbf{true}, \mathbf{false}, \neg, \wedge, \vee, \oplus, \Rightarrow, \Leftrightarrow\}$.
- A set of punctuation symbols $\{(,)\}$.

The grammar defines what constitutes a well-formed formula (wff):

Language of Propositional Logic: Well-Formed Formulae

The alphabet consist of the following sets:

- A set of propositional variables $PROP = \{p, q, r, s, ...\}$.
- A set of propositional connectives $\{\textbf{true}, \textbf{false}, \neg, \wedge, \vee, \oplus, \Rightarrow, \Leftrightarrow\}$.
- A set of punctuation symbols $\{(,)\}$.

The grammar defines what constitutes a well-formed formula (wff):

- Each of the elements in $PROP$ is a wff.

Language of Propositional Logic: Well-Formed Formulae

The alphabet consist of the following sets:

- A set of propositional variables $PROP = \{p, q, r, s, ...\}$.
- A set of propositional connectives $\{\textbf{true}, \textbf{false}, \neg, \wedge, \vee, \oplus, \Rightarrow, \Leftrightarrow\}$.
- A set of punctuation symbols $\{(,)\}$.

The grammar defines what constitutes a well-formed formula (wff):

- Each of the elements in $PROP$ is a wff.
- Each of **true** and **false** is a wff.

Language of Propositional Logic: Well-Formed Formulae

The alphabet consist of the following sets:

- A set of propositional variables $PROP = \{p, q, r, s, ...\}$.
- A set of propositional connectives $\{\mathbf{true}, \mathbf{false}, \neg, \wedge, \vee, \oplus, \Rightarrow, \Leftrightarrow\}$.
- A set of punctuation symbols $\{(,)\}$.

The grammar defines what constitutes a well-formed formula (wff):

- Each of the elements in $PROP$ is a wff.
- Each of **true** and **false** is a wff.
- If p is a wff, then $(\neg p)$ is a wff.

Language of Propositional Logic: Well-Formed Formulae

The alphabet consist of the following sets:

- A set of propositional variables $PROP = \{p, q, r, s, ...\}$.
- A set of propositional connectives $\{\mathbf{true}, \mathbf{false}, \neg, \wedge, \vee, \oplus, \Rightarrow, \Leftrightarrow\}$.
- A set of punctuation symbols $\{(,)\}$.

The grammar defines what constitutes a well-formed formula (wff):

- Each of the elements in $PROP$ is a wff.
- Each of **true** and **false** is a wff.
- If p is a wff, then $(\neg p)$ is a wff.
- If p and q are wffs, then so are $(p \wedge q)$, $(p \vee q)$, $(p \oplus q)$, $(p \Rightarrow q)$, $(p \Leftrightarrow q)$.

Language of Propositional Logic: Well-Formed Formulae

The alphabet consist of the following sets:

- A set of propositional variables $PROP = \{p, q, r, s, ...\}$.
- A set of propositional connectives $\{\mathbf{true}, \mathbf{false}, \neg, \wedge, \vee, \oplus, \Rightarrow, \Leftrightarrow\}$.
- A set of punctuation symbols $\{(,)\}$.

The grammar defines what constitutes a well-formed formula (wff):

- Each of the elements in $PROP$ is a wff.
- Each of **true** and **false** is a wff.
- If p is a wff, then $(\neg p)$ is a wff.
- If p and q are wffs, then so are $(p \wedge q)$, $(p \vee q)$, $(p \oplus q)$, $(p \Rightarrow q)$, $(p \Leftrightarrow q)$.

This is an *inductive* definition.

Language of Propositional Logic: Well-Formed Formulae

The alphabet consist of the following sets:

- A set of propositional variables $PROP = \{p, q, r, s, \ldots\}$.
- A set of propositional connectives $\{\mathbf{true}, \mathbf{false}, \neg, \wedge, \vee, \oplus, \Rightarrow, \Leftrightarrow\}$.
- A set of punctuation symbols $\{(,)\}$.

The grammar defines what constitutes a well-formed formula (wff):

- Each of the elements in $PROP$ is a wff.
- Each of **true** and **false** is a wff.
- If p is a wff, then $(\neg p)$ is a wff.
- If p and q are wffs, then so are $(p \wedge q)$, $(p \vee q)$, $(p \oplus q)$, $(p \Rightarrow q)$, $(p \Leftrightarrow q)$.

This is an *inductive* definition.

The set of wffs is called the Propositional Language $L(PROP)$.

Checking for syntactical correctness

syntaxcheck is a function that takes an expression and returns **true** if the expression is syntactically correct, and **false** otherwise.

$$syntaxcheck(\text{"}((p \vee (\neg q)) \vee s) \Leftrightarrow (\neg(s \wedge (\neg r)))\text{"})$$

Checking for syntactical correctness

syntaxcheck is a function that takes an expression and returns **true** if the expression is syntactically correct, and **false** otherwise.

$$\begin{aligned}&\mathit{syntaxcheck}(\text{"}((p \vee (\neg q)) \vee s) \Leftrightarrow (\neg(s \wedge (\neg r)))\text{"})\\ &= \mathit{syntaxcheck}(\text{"}(p \vee (\neg q)) \vee s\text{"})\end{aligned}$$

Checking for syntactical correctness

syntaxcheck is a function that takes an expression and returns **true** if the expression is syntactically correct, and **false** otherwise.

$syntaxcheck(\text{"}((p \vee (\neg q)) \vee s) \Leftrightarrow (\neg(s \wedge (\neg r)))\text{"})$
$= syntaxcheck(\text{"}(p \vee (\neg q)) \vee s\text{"})$ **and**

Checking for syntactical correctness

syntaxcheck is a function that takes an expression and returns **true** if the expression is syntactically correct, and **false** otherwise.

$$syntaxcheck(\text{"}((p \vee (\neg q)) \vee s) \Leftrightarrow (\neg(s \wedge (\neg r)))\text{"})$$
$$= syntaxcheck(\text{"}(p \vee (\neg q)) \vee s\text{"}) \textbf{ and } syntaxcheck(\text{"}\neg(s \wedge (\neg r))\text{"}).$$

Checking for syntactical correctness

syntaxcheck is a function that takes an expression and returns **true** if the expression is syntactically correct, and **false** otherwise.

$$
\begin{aligned}
& \mathit{syntaxcheck}(\text{"}((p \vee (\neg q)) \vee s) \Leftrightarrow (\neg(s \wedge (\neg r)))\text{"}) \\
& = \mathit{syntaxcheck}(\text{"}(p \vee (\neg q)) \vee s\text{"}) \textbf{ and } \mathit{syntaxcheck}(\text{"}\neg(s \wedge (\neg r))\text{"}). \\
& \quad \mathit{syntaxcheck}(\text{"}(p \vee (\neg q)) \vee s\text{"})
\end{aligned}
$$

Checking for syntactical correctness

syntaxcheck is a function that takes an expression and returns **true** if the expression is syntactically correct, and **false** otherwise.

$$\begin{aligned} & \mathit{syntaxcheck}(\text{"}((p \vee (\neg q)) \vee s) \Leftrightarrow (\neg(s \wedge (\neg r)))\text{"}) \\ & = \mathit{syntaxcheck}(\text{"}(p \vee (\neg q)) \vee s\text{"}) \textbf{ and } \mathit{syntaxcheck}(\text{"}\neg(s \wedge (\neg r))\text{"}). \\ & \quad \mathit{syntaxcheck}(\text{"}(p \vee (\neg q)) \vee s\text{"}) \\ & \quad = \mathit{syntaxcheck}(\text{"}p \vee (\neg q)\text{"}) \end{aligned}$$

Checking for syntactical correctness

syntaxcheck is a function that takes an expression and returns **true** if the expression is syntactically correct, and **false** otherwise.

$$\begin{aligned}
&syntaxcheck(\text{"}((p \vee (\neg q)) \vee s) \Leftrightarrow (\neg(s \wedge (\neg r)))\text{"})\\
&= syntaxcheck(\text{"}(p \vee (\neg q)) \vee s\text{"}) \textbf{ and } syntaxcheck(\text{"}\neg(s \wedge (\neg r))\text{"}).\\
&\quad syntaxcheck(\text{"}(p \vee (\neg q)) \vee s\text{"})\\
&\quad = syntaxcheck(\text{"}p \vee (\neg q)\text{"}) \textbf{ and}
\end{aligned}$$

Checking for syntactical correctness

syntaxcheck is a function that takes an expression and returns **true** if the expression is syntactically correct, and **false** otherwise.

$$\begin{aligned}
&\mathit{syntaxcheck}(\text{"}((p \vee (\neg q)) \vee s) \Leftrightarrow (\neg(s \wedge (\neg r)))\text{"}) \\
&= \mathit{syntaxcheck}(\text{"}(p \vee (\neg q)) \vee s\text{"}) \textbf{ and } \mathit{syntaxcheck}(\text{"}\neg(s \wedge (\neg r))\text{"}). \\
&\quad \mathit{syntaxcheck}(\text{"}(p \vee (\neg q)) \vee s\text{"}) \\
&\quad = \mathit{syntaxcheck}(\text{"}p \vee (\neg q)\text{"}) \textbf{ and } \mathit{syntaxcheck}(\text{"}s\text{"}).
\end{aligned}$$

Checking for syntactical correctness

syntaxcheck is a function that takes an expression and returns **true** if the expression is syntactically correct, and **false** otherwise.

$$\begin{array}{l} syntaxcheck(\text{"}((p \vee (\neg q)) \vee s) \Leftrightarrow (\neg(s \wedge (\neg r)))\text{"}) \\ = syntaxcheck(\text{"}(p \vee (\neg q)) \vee s\text{"}) \textbf{ and } syntaxcheck(\text{"}\neg(s \wedge (\neg r))\text{"}). \\ \quad syntaxcheck(\text{"}(p \vee (\neg q)) \vee s\text{"}) \\ \quad = syntaxcheck(\text{"}p \vee (\neg q)\text{"}) \textbf{ and } syntaxcheck(\text{"}s\text{"}). \\ \qquad syntaxcheck(\text{"}p \vee (\neg q)\text{"}) \end{array}$$

Checking for syntactical correctness

syntaxcheck is a function that takes an expression and returns **true** if the expression is syntactically correct, and **false** otherwise.

$$
\begin{aligned}
&\mathit{syntaxcheck}(\text{"}((p \vee (\neg q)) \vee s) \Leftrightarrow (\neg(s \wedge (\neg r)))\text{"})\\
&= \mathit{syntaxcheck}(\text{"}(p \vee (\neg q)) \vee s\text{"}) \textbf{ and } \mathit{syntaxcheck}(\text{"}\neg(s \wedge (\neg r))\text{"}).\\
&\quad \mathit{syntaxcheck}(\text{"}(p \vee (\neg q)) \vee s\text{"})\\
&\quad = \mathit{syntaxcheck}(\text{"}p \vee (\neg q)\text{"}) \textbf{ and } \mathit{syntaxcheck}(\text{"}s\text{"}).\\
&\quad\quad \mathit{syntaxcheck}(\text{"}p \vee (\neg q)\text{"})\\
&\quad\quad = \mathit{syntaxcheck}(\text{"}p\text{"})
\end{aligned}
$$

Checking for syntactical correctness

syntaxcheck is a function that takes an expression and returns **true** if the expression is syntactically correct, and **false** otherwise.

$$
\begin{aligned}
& \mathit{syntaxcheck}(\text{"}((p \lor (\neg q)) \lor s) \Leftrightarrow (\neg(s \land (\neg r)))\text{"}) \\
& = \mathit{syntaxcheck}(\text{"}(p \lor (\neg q)) \lor s\text{"}) \textbf{ and } \mathit{syntaxcheck}(\text{"}\neg(s \land (\neg r))\text{"}). \\
& \quad \mathit{syntaxcheck}(\text{"}(p \lor (\neg q)) \lor s\text{"}) \\
& \quad = \mathit{syntaxcheck}(\text{"}p \lor (\neg q)\text{"}) \textbf{ and } \mathit{syntaxcheck}(\text{"}s\text{"}). \\
& \qquad \mathit{syntaxcheck}(\text{"}p \lor (\neg q)\text{"}) \\
& \qquad = \mathit{syntaxcheck}(\text{"}p\text{"}) \textbf{ and }
\end{aligned}
$$

Checking for syntactical correctness

syntaxcheck is a function that takes an expression and returns **true** if the expression is syntactically correct, and **false** otherwise.

$$
\begin{array}{l}
\mathit{syntaxcheck}(\text{"}((p \vee (\neg q)) \vee s) \Leftrightarrow (\neg(s \wedge (\neg r)))\text{"}) \\
= \mathit{syntaxcheck}(\text{"}(p \vee (\neg q)) \vee s\text{"}) \textbf{ and } \mathit{syntaxcheck}(\text{"}\neg(s \wedge (\neg r))\text{"}). \\
\quad \mathit{syntaxcheck}(\text{"}(p \vee (\neg q)) \vee s\text{"}) \\
\quad = \mathit{syntaxcheck}(\text{"}p \vee (\neg q)\text{"}) \textbf{ and } \mathit{syntaxcheck}(\text{"}s\text{"}). \\
\qquad \mathit{syntaxcheck}(\text{"}p \vee (\neg q)\text{"}) \\
\qquad = \mathit{syntaxcheck}(\text{"}p\text{"}) \textbf{ and } \mathit{syntaxcheck}(\text{"}\neg q\text{"}).
\end{array}
$$

Checking for syntactical correctness

syntaxcheck is a function that takes an expression and returns **true** if the expression is syntactically correct, and **false** otherwise.

$$
\begin{aligned}
&\mathit{syntaxcheck}(\text{"}((p \vee (\neg q)) \vee s) \Leftrightarrow (\neg(s \wedge (\neg r)))\text{"})\\
&= \mathit{syntaxcheck}(\text{"}(p \vee (\neg q)) \vee s\text{"}) \textbf{ and } \mathit{syntaxcheck}(\text{"}\neg(s \wedge (\neg r))\text{"}).\\
&\quad \mathit{syntaxcheck}(\text{"}(p \vee (\neg q)) \vee s\text{"})\\
&\quad = \mathit{syntaxcheck}(\text{"}p \vee (\neg q)\text{"}) \textbf{ and } \mathit{syntaxcheck}(\text{"}s\text{"}).\\
&\qquad \mathit{syntaxcheck}(\text{"}p \vee (\neg q)\text{"})\\
&\qquad = \mathit{syntaxcheck}(\text{"}p\text{"}) \textbf{ and } \mathit{syntaxcheck}(\text{"}\neg q\text{"}).\\
&\qquad\quad \mathit{syntaxcheck}(\text{"}p\text{"}) \;=\; \textbf{true}, \mathit{because}\ p \in \mathit{PROP}.
\end{aligned}
$$

Checking for syntactical correctness

syntaxcheck is a function that takes an expression and returns **true** if the expression is syntactically correct, and **false** otherwise.

$$
\begin{aligned}
&syntaxcheck(\text{"}((p \lor (\neg q)) \lor s) \Leftrightarrow (\neg(s \land (\neg r)))\text{"})\\
&= syntaxcheck(\text{"}(p \lor (\neg q)) \lor s\text{"}) \textbf{ and } syntaxcheck(\text{"}\neg(s \land (\neg r))\text{"}).\\
&\quad syntaxcheck(\text{"}(p \lor (\neg q)) \lor s\text{"})\\
&\quad = syntaxcheck(\text{"}p \lor (\neg q)\text{"}) \textbf{ and } syntaxcheck(\text{"}s\text{"}).\\
&\quad\quad syntaxcheck(\text{"}p \lor (\neg q)\text{"})\\
&\quad\quad = syntaxcheck(\text{"}p\text{"}) \textbf{ and } syntaxcheck(\text{"}\neg q\text{"}).\\
&\quad\quad\quad syntaxcheck(\text{"}p\text{"}) \ = \ \textbf{true}, \textit{because } p \in PROP.\\
&\quad\quad\quad syntaxcheck(\text{"}\neg q\text{"}) = syntaxcheck(\text{"}q\text{"}).
\end{aligned}
$$

Checking for syntactical correctness

syntaxcheck is a function that takes an expression and returns **true** if the expression is syntactically correct, and **false** otherwise.

$$
\begin{array}{l}
syntaxcheck(\text{"}((p \vee (\neg q)) \vee s) \Leftrightarrow (\neg(s \wedge (\neg r)))\text{"}) \\
= syntaxcheck(\text{"}(p \vee (\neg q)) \vee s\text{"}) \textbf{ and } syntaxcheck(\text{"}\neg(s \wedge (\neg r))\text{"}). \\
\quad syntaxcheck(\text{"}(p \vee (\neg q)) \vee s\text{"}) \\
\quad = syntaxcheck(\text{"}p \vee (\neg q)\text{"}) \textbf{ and } syntaxcheck(\text{"}s\text{"}). \\
\qquad syntaxcheck(\text{"}p \vee (\neg q)\text{"}) \\
\qquad = syntaxcheck(\text{"}p\text{"}) \textbf{ and } syntaxcheck(\text{"}\neg q\text{"}). \\
\qquad\quad syntaxcheck(\text{"}p\text{"}) \; = \; \textbf{true}, \textit{ because } p \in PROP. \\
\qquad\quad syntaxcheck(\text{"}\neg q\text{"}) = syntaxcheck(\text{"}q\text{"}). \\
\qquad\qquad syntaxcheck(\text{"}q\text{"}) \; = \; \textbf{true}, \textit{ because } q \in PROP.
\end{array}
$$

Checking for syntactical correctness

syntaxcheck is a function that takes an expression and returns **true** if the expression is syntactically correct, and **false** otherwise.

$$
\begin{array}{l}
\mathit{syntaxcheck}(\text{"}((p \vee (\neg q)) \vee s) \Leftrightarrow (\neg(s \wedge (\neg r)))\text{"}) \\
= \mathit{syntaxcheck}(\text{"}(p \vee (\neg q)) \vee s\text{"}) \ \textbf{and} \ \mathit{syntaxcheck}(\text{"}\neg(s \wedge (\neg r))\text{"}). \\
\quad \mathit{syntaxcheck}(\text{"}(p \vee (\neg q)) \vee s\text{"}) \\
\quad = \mathit{syntaxcheck}(\text{"}p \vee (\neg q)\text{"}) \ \textbf{and} \ \mathit{syntaxcheck}(\text{"}s\text{"}). \\
\qquad \mathit{syntaxcheck}(\text{"}p \vee (\neg q)\text{"}) \\
\qquad = \mathit{syntaxcheck}(\text{"}p\text{"}) \ \textbf{and} \ \mathit{syntaxcheck}(\text{"}\neg q\text{"}). \\
\qquad\quad \mathit{syntaxcheck}(\text{"}p\text{"}) \ = \ \textbf{true}, \mathit{because} \ p \in \mathit{PROP}. \\
\qquad\quad \mathit{syntaxcheck}(\text{"}\neg q\text{"}) = \mathit{syntaxcheck}(\text{"}q\text{"}). \\
\qquad\qquad \mathit{syntaxcheck}(\text{"}q\text{"}) \ = \ \textbf{true}, \mathit{because} \ q \in \mathit{PROP}. \\
\qquad\quad \mathit{syntaxcheck}(\text{"}\neg q\text{"}) = \textbf{true}.
\end{array}
$$

Checking for syntactical correctness

syntaxcheck is a function that takes an expression and returns **true** if the expression is syntactically correct, and **false** otherwise.

$$
\begin{aligned}
&\mathit{syntaxcheck}(\text{"}((p \vee (\neg q)) \vee s) \Leftrightarrow (\neg(s \wedge (\neg r)))\text{"}) \\
&= \mathit{syntaxcheck}(\text{"}(p \vee (\neg q)) \vee s\text{"}) \textbf{ and } \mathit{syntaxcheck}(\text{"}\neg(s \wedge (\neg r))\text{"}). \\
&\quad \mathit{syntaxcheck}(\text{"}(p \vee (\neg q)) \vee s\text{"}) \\
&\quad = \mathit{syntaxcheck}(\text{"}p \vee (\neg q)\text{"}) \textbf{ and } \mathit{syntaxcheck}(\text{"}s\text{"}). \\
&\qquad \mathit{syntaxcheck}(\text{"}p \vee (\neg q)\text{"}) \\
&\qquad = \mathit{syntaxcheck}(\text{"}p\text{"}) \textbf{ and } \mathit{syntaxcheck}(\text{"}\neg q\text{"}). \\
&\qquad\quad \mathit{syntaxcheck}(\text{"}p\text{"}) \;=\; \textbf{true}, \mathit{because}\ p \in \mathit{PROP}. \\
&\qquad\quad \mathit{syntaxcheck}(\text{"}\neg q\text{"}) = \mathit{syntaxcheck}(\text{"}q\text{"}). \\
&\qquad\qquad \mathit{syntaxcheck}(\text{"}q\text{"}) \;=\; \textbf{true}, \mathit{because}\ q \in \mathit{PROP}. \\
&\qquad\quad \mathit{syntaxcheck}(\text{"}\neg q\text{"}) = \textbf{true}. \\
&\qquad \mathit{syntaxcheck}(\text{"}p \vee (\neg q)\text{"}) = \textbf{true and true} = \textbf{true}.
\end{aligned}
$$

Checking for syntactical correctness

syntaxcheck is a function that takes an expression and returns **true** if the expression is syntactically correct, and **false** otherwise.

$$
\begin{array}{l}
syntaxcheck(\text{"}((p \vee (\neg q)) \vee s) \Leftrightarrow (\neg(s \wedge (\neg r)))\text{"}) \\
= syntaxcheck(\text{"}(p \vee (\neg q)) \vee s\text{"}) \textbf{ and } syntaxcheck(\text{"}\neg(s \wedge (\neg r))\text{"}). \\
\quad syntaxcheck(\text{"}(p \vee (\neg q)) \vee s\text{"}) \\
\quad = syntaxcheck(\text{"}p \vee (\neg q)\text{"}) \textbf{ and } syntaxcheck(\text{"}s\text{"}). \\
\qquad syntaxcheck(\text{"}p \vee (\neg q)\text{"}) \\
\qquad = syntaxcheck(\text{"}p\text{"}) \textbf{ and } syntaxcheck(\text{"}\neg q\text{"}). \\
\qquad\quad syntaxcheck(\text{"}p\text{"}) \;=\; \textbf{true}, \textit{because } p \in PROP. \\
\qquad\quad syntaxcheck(\text{"}\neg q\text{"}) = syntaxcheck(\text{"}q\text{"}). \\
\qquad\qquad syntaxcheck(\text{"}q\text{"}) \;=\; \textbf{true}, \textit{because } q \in PROP. \\
\qquad\quad syntaxcheck(\text{"}\neg q\text{"}) = \textbf{true}. \\
\qquad syntaxcheck(\text{"}p \vee (\neg q)\text{"}) = \textbf{true and true} = \textbf{true}. \\
\quad syntaxcheck(\text{"}(p \vee (\neg q)) \vee s\text{"}) = \textbf{true and } syntaxcheck(\text{"}s\text{"}) = syntaxcheck(\text{"}s\text{"}).
\end{array}
$$

Checking for syntactical correctness

syntaxcheck is a function that takes an expression and returns **true** if the expression is syntactically correct, and **false** otherwise.

$$
\begin{array}{l}
\mathit{syntaxcheck}(\text{"}((p \lor (\neg q)) \lor s) \Leftrightarrow (\neg(s \land (\neg r)))\text{"}) \\
= \mathit{syntaxcheck}(\text{"}(p \lor (\neg q)) \lor s\text{"}) \textbf{ and } \mathit{syntaxcheck}(\text{"}\neg(s \land (\neg r))\text{"}). \\
\quad \mathit{syntaxcheck}(\text{"}(p \lor (\neg q)) \lor s\text{"}) \\
\quad = \mathit{syntaxcheck}(\text{"}p \lor (\neg q)\text{"}) \textbf{ and } \mathit{syntaxcheck}(\text{"}s\text{"}). \\
\quad\quad \mathit{syntaxcheck}(\text{"}p \lor (\neg q)\text{"}) \\
\quad\quad = \mathit{syntaxcheck}(\text{"}p\text{"}) \textbf{ and } \mathit{syntaxcheck}(\text{"}\neg q\text{"}). \\
\quad\quad\quad \mathit{syntaxcheck}(\text{"}p\text{"}) \ = \ \textbf{true}, \mathit{because}\ p \in \mathit{PROP}. \\
\quad\quad\quad \mathit{syntaxcheck}(\text{"}\neg q\text{"}) = \mathit{syntaxcheck}(\text{"}q\text{"}). \\
\quad\quad\quad\quad \mathit{syntaxcheck}(\text{"}q\text{"}) \ = \ \textbf{true}, \mathit{because}\ q \in \mathit{PROP}. \\
\quad\quad\quad \mathit{syntaxcheck}(\text{"}\neg q\text{"}) = \textbf{true}. \\
\quad\quad \mathit{syntaxcheck}(\text{"}p \lor (\neg q)\text{"}) = \textbf{true and true} = \textbf{true}. \\
\quad \mathit{syntaxcheck}(\text{"}(p \lor (\neg q)) \lor s\text{"}) = \textbf{true and } \mathit{syntaxcheck}(\text{"}s\text{"}) = \mathit{syntaxcheck}(\text{"}s\text{"}). \\
\quad\quad \mathit{syntaxcheck}(\text{"}s\text{"}) \ = \ \textbf{true}, \mathit{because}\ s \in \mathit{PROP}.
\end{array}
$$

Checking for syntactical correctness

syntaxcheck is a function that takes an expression and returns **true** if the expression is syntactically correct, and **false** otherwise.

$$
\begin{array}{l}
syntaxcheck(\text{"}((p \vee (\neg q)) \vee s) \Leftrightarrow (\neg(s \wedge (\neg r)))\text{"}) \\
= syntaxcheck(\text{"}(p \vee (\neg q)) \vee s\text{"}) \textbf{ and } syntaxcheck(\text{"}\neg(s \wedge (\neg r))\text{"}). \\
\quad syntaxcheck(\text{"}(p \vee (\neg q)) \vee s\text{"}) \\
\quad = syntaxcheck(\text{"}p \vee (\neg q)\text{"}) \textbf{ and } syntaxcheck(\text{"}s\text{"}). \\
\quad\quad syntaxcheck(\text{"}p \vee (\neg q)\text{"}) \\
\quad\quad = syntaxcheck(\text{"}p\text{"}) \textbf{ and } syntaxcheck(\text{"}\neg q\text{"}). \\
\quad\quad\quad syntaxcheck(\text{"}p\text{"}) \;=\; \textbf{true}, \textit{because } p \in PROP. \\
\quad\quad\quad syntaxcheck(\text{"}\neg q\text{"}) = syntaxcheck(\text{"}q\text{"}). \\
\quad\quad\quad\quad syntaxcheck(\text{"}q\text{"}) \;=\; \textbf{true}, \textit{because } q \in PROP. \\
\quad\quad\quad syntaxcheck(\text{"}\neg q\text{"}) = \textbf{true}. \\
\quad\quad syntaxcheck(\text{"}p \vee (\neg q)\text{"}) = \textbf{true and true} = \textbf{true}. \\
\quad syntaxcheck(\text{"}(p \vee (\neg q)) \vee s\text{"}) = \textbf{true and } syntaxcheck(\text{"}s\text{"}) = syntaxcheck(\text{"}s\text{"}). \\
\quad\quad syntaxcheck(\text{"}s\text{"}) \;=\; \textbf{true}, \textit{because } s \in PROP. \\
\quad syntaxcheck(\text{"}(p \vee (\neg q)) \vee s\text{"}) = \textbf{true and true} = \textbf{true}.
\end{array}
$$

Checking for syntactical correctness

syntaxcheck is a function that takes an expression and returns **true** if the expression is syntactically correct, and **false** otherwise.

$$\begin{array}{l}
\textit{syntaxcheck}(\text{"}((p \vee (\neg q)) \vee s) \Leftrightarrow (\neg(s \wedge (\neg r)))\text{"}) \\
= \textit{syntaxcheck}(\text{"}(p \vee (\neg q)) \vee s\text{"}) \textbf{ and } \textit{syntaxcheck}(\text{"}\neg(s \wedge (\neg r))\text{"}). \\
\quad \textit{syntaxcheck}(\text{"}(p \vee (\neg q)) \vee s\text{"}) \\
\quad = \textit{syntaxcheck}(\text{"}p \vee (\neg q)\text{"}) \textbf{ and } \textit{syntaxcheck}(\text{"}s\text{"}). \\
\qquad \textit{syntaxcheck}(\text{"}p \vee (\neg q)\text{"}) \\
\qquad = \textit{syntaxcheck}(\text{"}p\text{"}) \textbf{ and } \textit{syntaxcheck}(\text{"}\neg q\text{"}). \\
\qquad\quad \textit{syntaxcheck}(\text{"}p\text{"}) = \textbf{true}, \textit{because } p \in PROP. \\
\qquad\quad \textit{syntaxcheck}(\text{"}\neg q\text{"}) = \textit{syntaxcheck}(\text{"}q\text{"}). \\
\qquad\qquad \textit{syntaxcheck}(\text{"}q\text{"}) = \textbf{true}, \textit{because } q \in PROP. \\
\qquad\quad \textit{syntaxcheck}(\text{"}\neg q\text{"}) = \textbf{true}. \\
\qquad \textit{syntaxcheck}(\text{"}p \vee (\neg q)\text{"}) = \textbf{true and true} = \textbf{true}. \\
\quad \textit{syntaxcheck}(\text{"}(p \vee (\neg q)) \vee s\text{"}) = \textbf{true and } \textit{syntaxcheck}(\text{"}s\text{"}) = \textit{syntaxcheck}(\text{"}s\text{"}). \\
\qquad \textit{syntaxcheck}(\text{"}s\text{"}) = \textbf{true}, \textit{because } s \in PROP. \\
\quad \textit{syntaxcheck}(\text{"}(p \vee (\neg q)) \vee s\text{"}) = \textbf{true and true} = \textbf{true}. \\
= \textbf{true and } \textit{syntaxcheck}(\text{"}\neg(s \wedge (\neg r))\text{"}).
\end{array}$$

Checking for syntactical correctness

syntaxcheck is a function that takes an expression and returns **true** if the expression is syntactically correct, and **false** otherwise.

$$
\begin{aligned}
& syntaxcheck(\text{"}((p \vee (\neg q)) \vee s) \Leftrightarrow (\neg(s \wedge (\neg r)))\text{"}) \\
& = syntaxcheck(\text{"}(p \vee (\neg q)) \vee s\text{"}) \textbf{ and } syntaxcheck(\text{"}\neg(s \wedge (\neg r))\text{"}). \\
& \quad syntaxcheck(\text{"}(p \vee (\neg q)) \vee s\text{"}) \\
& \quad = syntaxcheck(\text{"}p \vee (\neg q)\text{"}) \textbf{ and } syntaxcheck(\text{"}s\text{"}). \\
& \quad\quad syntaxcheck(\text{"}p \vee (\neg q)\text{"}) \\
& \quad\quad = syntaxcheck(\text{"}p\text{"}) \textbf{ and } syntaxcheck(\text{"}\neg q\text{"}). \\
& \quad\quad\quad syntaxcheck(\text{"}p\text{"}) \ = \ \textbf{true}, \textit{ because } p \in PROP. \\
& \quad\quad\quad syntaxcheck(\text{"}\neg q\text{"}) = syntaxcheck(\text{"}q\text{"}). \\
& \quad\quad\quad\quad syntaxcheck(\text{"}q\text{"}) \ = \ \textbf{true}, \textit{ because } q \in PROP. \\
& \quad\quad\quad syntaxcheck(\text{"}\neg q\text{"}) = \textbf{true}. \\
& \quad\quad syntaxcheck(\text{"}p \vee (\neg q)\text{"}) = \textbf{true and true} = \textbf{true}. \\
& \quad syntaxcheck(\text{"}(p \vee (\neg q)) \vee s\text{"}) = \textbf{true and } syntaxcheck(\text{"}s\text{"}) = syntaxcheck(\text{"}s\text{"}). \\
& \quad\quad syntaxcheck(\text{"}s\text{"}) \ = \ \textbf{true}, \textit{ because } s \in PROP. \\
& \quad syntaxcheck(\text{"}(p \vee (\neg q)) \vee s\text{"}) = \textbf{true and true} = \textbf{true}. \\
& = \textbf{true and } syntaxcheck(\text{"}\neg(s \wedge (\neg r))\text{"}). \\
& = syntaxcheck(\text{"}\neg(s \wedge (\neg r))\text{"}). \qquad \text{(Left as SSE.)}
\end{aligned}
$$

Checking for syntactical correctness

syntaxcheck is a function that takes an expression and returns **true** if the expression is syntactically correct, and **false** otherwise.

$syntaxcheck(\text{"}((p \vee (\neg q)) \vee s) \Leftrightarrow (\neg(s \wedge (\neg r)))\text{"})$
$= syntaxcheck(\text{"}(p \vee (\neg q)) \vee s\text{"})$ **and** $syntaxcheck(\text{"}\neg(s \wedge (\neg r))\text{"})$.
$syntaxcheck(\text{"}(p \vee (\neg q)) \vee s\text{"})$
$= syntaxcheck(\text{"}p \vee (\neg q)\text{"})$ **and** $syntaxcheck(\text{"}s\text{"})$.
$syntaxcheck(\text{"}p \vee (\neg q)\text{"})$
$= syntaxcheck(\text{"}p\text{"})$ **and** $syntaxcheck(\text{"}\neg q\text{"})$.
$syntaxcheck(\text{"}p\text{"}) =$ **true**, *because* $p \in PROP$.
$syntaxcheck(\text{"}\neg q\text{"}) = syntaxcheck(\text{"}q\text{"})$.
$syntaxcheck(\text{"}q\text{"}) =$ **true**, *because* $q \in PROP$.
$syntaxcheck(\text{"}\neg q\text{"}) =$ **true**.
$syntaxcheck(\text{"}p \vee (\neg q)\text{"}) =$ **true and true** $=$ **true**.
$syntaxcheck(\text{"}(p \vee (\neg q)) \vee s\text{"}) =$ **true and** $syntaxcheck(\text{"}s\text{"}) = syntaxcheck(\text{"}s\text{"})$.
$syntaxcheck(\text{"}s\text{"}) =$ **true**, *because* $s \in PROP$.
$syntaxcheck(\text{"}(p \vee (\neg q)) \vee s\text{"}) =$ **true and true** $=$ **true**.
$=$ **true and** $syntaxcheck(\text{"}\neg(s \wedge (\neg r))\text{"})$.
$= syntaxcheck(\text{"}\neg(s \wedge (\neg r))\text{"})$. (Left as SSE.)

Challenge: Write a program that implements the above to determine the syntactic correctness of arbitrary expressions. (Include a check for operator validity!)

Determining the Truth Table for a Compound Proposition

p	(my breakfast is) toast.
q	(my breakfast is) cereal.
r	(my breakfast is) juice.

The statement "my breakfast is either toast or cereal, and juice" may be written in symbolic form as

Determining the Truth Table for a Compound Proposition

p	(my breakfast is) toast.
q	(my breakfast is) cereal.
r	(my breakfast is) juice.

The statement "my breakfast is either toast or cereal, and juice" may be written in symbolic form as $(p \lor q) \land r$.

Determining the Truth Table for a Compound Proposition

p	(my breakfast is) toast.
q	(my breakfast is) cereal.
r	(my breakfast is) juice.

The statement "my breakfast is either toast or cereal, and juice" may be written in symbolic form as $(p \vee q) \wedge r$.

p	q	r	$p \vee q$	$(p \vee q) \wedge r$
T	T	T		
T	T	F		
T	F	T		
T	F	F		
F	T	T		
F	T	F		
F	F	T		
F	F	F		

Determining the Truth Table for a Compound Proposition

p (my breakfast is) toast.
q (my breakfast is) cereal.
r (my breakfast is) juice.

The statement "my breakfast is either toast or cereal, and juice" may be written in symbolic form as $(p \vee q) \wedge r$.

p	q	r	$p \vee q$	$(p \vee q) \wedge r$
T	T	T		
T	T	F		
T	F	T		
T	F	F		
F	T	T		
F	T	F		
F	F	T		
F	F	F		

The compound proposition is true if

- I eat toast, cereal, and juice; or
- I eat toast and juice; or
- I eat cereal and juice.

Determining the Truth Table for a Compound Proposition

p (my breakfast is) toast.
q (my breakfast is) cereal.
r (my breakfast is) juice.

The statement "my breakfast is either toast or cereal, and juice" may be written in symbolic form as $(p \vee q) \wedge r$.

p	q	r	$p \vee q$	$(p \vee q) \wedge r$
T	T	T		
T	T	F		
T	F	T		
T	F	F		
F	T	T		
F	T	F		
F	F	T		
F	F	F		

The compound proposition is true if

- I eat toast, cereal, and juice; or
- I eat toast and juice; or
- I eat cereal and juice.

The truth table gives results for **all possible value assignments** of p, q and r.

Truth Tables for Compound Propositions

- Truth tables may be used to show value assignments (also called **interpretations**) of compound propositions.

Truth Tables for Compound Propositions

- Truth tables may be used to show value assignments (also called **interpretations**) of compound propositions.
- To draw up a truth table, construct a column for each proposition involved.

Truth Tables for Compound Propositions

- Truth tables may be used to show value assignments (also called **interpretations**) of compound propositions.
- To draw up a truth table, construct a column for each proposition involved.
- You need 2^n rows for n propositions for all possible ways of setting the propositions to T and F.

Truth Tables for Compound Propositions

- Truth tables may be used to show value assignments (also called **interpretations**) of compound propositions.
- To draw up a truth table, construct a column for each proposition involved.
- You need 2^n rows for n propositions for all possible ways of setting the propositions to T and F.
- If we have 3 propositions, p, q, r, then we need $2^3 = 8$ rows.

Truth Tables for Compound Propositions

- Truth tables may be used to show value assignments (also called **interpretations**) of compound propositions.
- To draw up a truth table, construct a column for each proposition involved.
- You need 2^n rows for n propositions for all possible ways of setting the propositions to T and F.
- If we have 3 propositions, p, q, r, then we need $2^3 = 8$ rows.
- Next, construct a column for each connective, the most *deeply nested* first.

Truth Tables for Compound Propositions

- Truth tables may be used to show value assignments (also called **interpretations**) of compound propositions.
- To draw up a truth table, construct a column for each proposition involved.
- You need 2^n rows for n propositions for all possible ways of setting the propositions to T and F.
- If we have 3 propositions, p, q, r, then we need $2^3 = 8$ rows.
- Next, construct a column for each connective, the most *deeply nested* first.
- Evaluate each column based on the semantics of the connectives using values for propositions or previous columns.

Value Assignments (also called *interpretations*)

Given a particular formula, can you tell if it is true or not?

Value Assignments (also called *interpretations*)

Given a particular formula, can you tell if it is true or not?

- You need to know the truth values of the component propositions in order to be able to tell whether a formula is true.

Value Assignments (also called *interpretations*)

Given a particular formula, can you tell if it is true or not?

- You need to know the truth values of the component propositions in order to be able to tell whether a formula is true.
- Let $I_f : PROP \rightarrow \{T, F\}$ be an interpretation which assigns a truth value to each atomic proposition.

Value Assignments (also called *interpretations*)

Given a particular formula, can you tell if it is true or not?

- You need to know the truth values of the component propositions in order to be able to tell whether a formula is true.
- Let $I_f : PROP \to \{T, F\}$ be an interpretation which assigns a truth value to each atomic proposition.
- Suppose we have an interpretation function I_f such that: $I_f(p) = F, I_f(q) = T, I_f(r) = F$

Value Assignments (also called *interpretations*)

Given a particular formula, can you tell if it is true or not?

- You need to know the truth values of the component propositions in order to be able to tell whether a formula is true.
- Let $I_f : PROP \rightarrow \{T, F\}$ be an interpretation which assigns a truth value to each atomic proposition.
- Suppose we have an interpretation function I_f such that:
  $I_f(p) = F, I_f(q) = T, I_f(r) = F$
- Then the truth value of $(p \vee q) \Rightarrow r$ is evaluated by

$$
\begin{array}{rll}
 & (I_f(p) \vee I_f(q)) \Rightarrow I_f(r) & \mid I_f \\
= & (F \vee T) \Rightarrow F & \mid \vee \\
= & T \Rightarrow F & \mid \Rightarrow \\
= & F &
\end{array}
$$

Value Assignments (also called *interpretations*)

Given a particular formula, can you tell if it is true or not?

- You need to know the truth values of the component propositions in order to be able to tell whether a formula is true.
- Let $I_f : PROP \rightarrow \{T, F\}$ be an interpretation which assigns a truth value to each atomic proposition.
- Suppose we have an interpretation function I_f such that: $I_f(p) = F, I_f(q) = T, I_f(r) = F$
- Then the truth value of $(p \vee q) \Rightarrow r$ is evaluated by

$$
\begin{array}{cll}
 & (I_f(p) \vee I_f(q)) \Rightarrow I_f(r) & \mid I_f \\
= & (F \vee T) \Rightarrow F & \mid \vee \\
= & T \Rightarrow F & \mid \Rightarrow \\
= & F &
\end{array}
$$

- Which line in the truth table for $(p \vee q) \Rightarrow r$ corresponds to this particular valuation?

Validity and Satisfiability

- A formula is said to be *valid* (or a *tautology*) iff it is true under *every* value assignment.

p	$\neg p$	$p \vee \neg p$
T	F	
F	T	

Validity and Satisfiability

- A formula is said to be *valid* (or a *tautology*) iff it is true under *every* value assignment.

p	$\neg p$	$p \vee \neg p$
T	F	T
F	T	T

This is also called the "law of excluded middle", i.e. for any proposition, either that proposition is true, or its negation is true.

Validity and Satisfiability

- A formula is said to be *satisfiable* (or *consistent*) iff it is true under *at least one* value assignment.

p	q	$p \vee q$
T	T	
T	F	
F	T	
F	F	

Validity and Satisfiability

- A formula is said to be *satisfiable* (or *consistent*) iff it is true under *at least one* value assignment.

p	q	$p \lor q$
T	T	T
T	F	T
F	T	T
F	F	F

Validity and Satisfiability

- A formula is said to be *valid* (or a *tautology*) iff it is true under *every* value assignment.

Validity and Satisfiability

- A formula is said to be *valid* (or a *tautology*) iff it is true under *every* value assignment.
- A formula is said to be *satisfiable* (or *consistent*) iff it is true under *at least one* value assignment.

Validity and Satisfiability

- A formula is said to be *valid* (or a *tautology*) iff it is true under *every* value assignment.
- A formula is said to be *satisfiable* (or *consistent*) iff it is true under *at least one* value assignment.
- A formula is said to be *unsatisfiable* (or *contradictory*) iff it is not made true under *any* value assignment.

p	$\neg p$	$p \wedge \neg p$
T	F	
F	T	

Validity and Satisfiability

- A formula is said to be *valid* (or a *tautology*) iff it is true under *every* value assignment.
- A formula is said to be *satisfiable* (or *consistent*) iff it is true under *at least one* value assignment.
- A formula is said to be *unsatisfiable* (or *contradictory*) iff it is not made true under *any* value assignment.

p	$\neg p$	$p \wedge \neg p$
T	F	F
F	T	F

Validity and Satisfiability

- A formula is said to be *valid* (or a *tautology*) iff it is true under *every* value assignment.
- A formula is said to be *satisfiable* (or *consistent*) iff it is true under *at least one* value assignment.
- A formula is said to be *unsatisfiable* (or *contradictory*) iff it is not made true under *any* value assignment.

p	$\neg p$	$p \wedge \neg p$
T	F	F
F	T	F

- **Note, a formula φ is a contradiction if and only if $\neg\varphi$ is a tautology.**

Validity and Satisfiability

- A formula is said to be *valid* (or a *tautology*) iff it is true under *every* value assignment.
- A formula is said to be *satisfiable* (or *consistent*) iff it is true under *at least one* value assignment.
- A formula is said to be *unsatisfiable* (or *contradictory*) iff it is not made true under *any* value assignment.

p	$\neg p$	$p \wedge \neg p$
T	F	F
F	T	F

- **Note, a formula φ is a contradiction if and only if $\neg\varphi$ is a tautology.**
- A formula that is neither a tautology nor a contradiction is called a *contingency.*

Truth Tables and Tautology

Using a truth table determine whether $(p \Rightarrow q) \vee (q \Rightarrow p)$ is a tautology, a contradiction or satisfiable (i.e. a contingency).

p	q	$(p \Rightarrow q)$	$(q \Rightarrow p)$	$(p \Rightarrow q) \vee (q \Rightarrow p)$
T	T			
T	F			
F	T			
F	F			

Determine the truth table for $\neg((p \Rightarrow q) \vee (q \Rightarrow p))$:

Using a truth table determine whether $((p \wedge q) \Rightarrow r)$ is a tautology, a contradiction or satisfiable (i.e. a contingency).

Logical Equivalence

When two compound propositions have the same truth values for all value assignments then they are *logically equivalent*.

Logical Equivalence

When two compound propositions have the same truth values for all value assignments then they are *logically equivalent*.

Definition

The propositions p and q are called *logically equivalent* iff $p \Leftrightarrow q$ is a tautology. Logical equivalence of p and q is denoted by $p \equiv q$.

Logical Equivalence

When two compound propositions have the same truth values for all value assignments then they are *logically equivalent*.

Definition

The propositions p and q are called *logically equivalent* iff $p \Leftrightarrow q$ is a tautology. Logical equivalence of p and q is denoted by $p \equiv q$.

To determine whether two propositions are logically equivalent we can compare their truth tables.

Logical Equivalence

When two compound propositions have the same truth values for all value assignments then they are *logically equivalent*.

Definition

The propositions p and q are called *logically equivalent* iff $p \Leftrightarrow q$ is a tautology. Logical equivalence of p and q is denoted by $p \equiv q$.

To determine whether two propositions are logically equivalent we can compare their truth tables. For p and q to be equivalent, the colums that contain their truth values must match.

Logical Equivalence

When two compound propositions have the same truth values for all value assignments then they are *logically equivalent*.

Definition

The propositions p and q are called *logically equivalent* iff $p \Leftrightarrow q$ is a tautology. Logical equivalence of p and q is denoted by $p \equiv q$.

To determine whether two propositions are logically equivalent we can compare their truth tables. For p and q to be equivalent, the colums that contain their truth values must match.

Determine whether $\neg(p \vee q)$ and $\neg p \wedge \neg q$ are logically equivalent.

p	q	$p \vee q$	$\neg(p \vee q)$	$\neg p$	$\neg q$	$\neg p \wedge \neg q$
T	T					
T	F					
F	T					
F	F					

Logical Equivalences

Equivalence	Name

Logical Equivalences

Equivalence	Name
$p \wedge \mathbb{T} \equiv p$	Identity Laws

Logical Equivalences

Equivalence	Name
$p \wedge \mathbb{T} \equiv p$ $p \vee \mathbb{F} \equiv p$	Identity Laws

Logical Equivalences

Equivalence	Name
$p \wedge \mathbb{T} \equiv p$ $p \vee \mathbb{F} \equiv p$	Identity Laws
$p \vee \mathbb{T} \equiv \mathbb{T}$	Domination Laws

Logical Equivalences

Equivalence	Name
$p \wedge \mathbb{T} \equiv p$ $p \vee \mathbb{F} \equiv p$	Identity Laws
$p \vee \mathbb{T} \equiv \mathbb{T}$ $p \wedge \mathbb{F} \equiv \mathbb{F}$	Domination Laws

Logical Equivalences

Equivalence	Name
$p \wedge \mathbb{T} \equiv p$ $p \vee \mathbb{F} \equiv p$	Identity Laws
$p \vee \mathbb{T} \equiv \mathbb{T}$ $p \wedge \mathbb{F} \equiv \mathbb{F}$	Domination Laws
$p \vee p \equiv p$	Idempotent Laws

Logical Equivalences

Equivalence	Name
$p \wedge \mathbb{T} \equiv p$ $p \vee \mathbb{F} \equiv p$	Identity Laws
$p \vee \mathbb{T} \equiv \mathbb{T}$ $p \wedge \mathbb{F} \equiv \mathbb{F}$	Domination Laws
$p \vee p \equiv p$ $p \wedge p \equiv p$	Idempotent Laws

Logical Equivalences

Equivalence	Name
$p \wedge \mathbb{T} \equiv p$ $p \vee \mathbb{F} \equiv p$	Identity Laws
$p \vee \mathbb{T} \equiv \mathbb{T}$ $p \wedge \mathbb{F} \equiv \mathbb{F}$	Domination Laws
$p \vee p \equiv p$ $p \wedge p \equiv p$	Idempotent Laws
$\neg(\neg p) \equiv p$	Double Negation Law

Logical Equivalences

Equivalence	Name
$p \wedge \mathbb{T} \equiv p$ $p \vee \mathbb{F} \equiv p$	Identity Laws
$p \vee \mathbb{T} \equiv \mathbb{T}$ $p \wedge \mathbb{F} \equiv \mathbb{F}$	Domination Laws
$p \vee p \equiv p$ $p \wedge p \equiv p$	Idempotent Laws
$\neg(\neg p) \equiv p$	Double Negation Law
$p \vee q \equiv q \vee p$	Commutative Laws

Logical Equivalences

Equivalence	Name
$p \wedge \mathbb{T} \equiv p$ $p \vee \mathbb{F} \equiv p$	Identity Laws
$p \vee \mathbb{T} \equiv \mathbb{T}$ $p \wedge \mathbb{F} \equiv \mathbb{F}$	Domination Laws
$p \vee p \equiv p$ $p \wedge p \equiv p$	Idempotent Laws
$\neg(\neg p) \equiv p$	Double Negation Law
$p \vee q \equiv q \vee p$ $p \wedge q \equiv q \wedge p$	Commutative Laws

Logical Equivalences

Equivalence	Name
$p \wedge \mathbb{T} \equiv p$ $p \vee \mathbb{F} \equiv p$	Identity Laws
$p \vee \mathbb{T} \equiv \mathbb{T}$ $p \wedge \mathbb{F} \equiv \mathbb{F}$	Domination Laws
$p \vee p \equiv p$ $p \wedge p \equiv p$	Idempotent Laws
$\neg(\neg p) \equiv p$	Double Negation Law
$p \vee q \equiv q \vee p$ $p \wedge q \equiv q \wedge p$	Commutative Laws
$p \vee (q \vee r) \equiv (p \vee q) \vee r$	Associative Laws

Logical Equivalences

Equivalence	Name
$p \wedge \mathbb{T} \equiv p$ $p \vee \mathbb{F} \equiv p$	Identity Laws
$p \vee \mathbb{T} \equiv \mathbb{T}$ $p \wedge \mathbb{F} \equiv \mathbb{F}$	Domination Laws
$p \vee p \equiv p$ $p \wedge p \equiv p$	Idempotent Laws
$\neg(\neg p) \equiv p$	Double Negation Law
$p \vee q \equiv q \vee p$ $p \wedge q \equiv q \wedge p$	Commutative Laws
$p \vee (q \vee r) \equiv (p \vee q) \vee r$ $p \wedge (q \wedge r) \equiv (p \wedge q) \wedge r$	Associative Laws

Logical Equivalences

Equivalence	Name
$p \wedge \mathbb{T} \equiv p$ $p \vee \mathbb{F} \equiv p$	Identity Laws
$p \vee \mathbb{T} \equiv \mathbb{T}$ $p \wedge \mathbb{F} \equiv \mathbb{F}$	Domination Laws
$p \vee p \equiv p$ $p \wedge p \equiv p$	Idempotent Laws
$\neg(\neg p) \equiv p$	Double Negation Law
$p \vee q \equiv q \vee p$ $p \wedge q \equiv q \wedge p$	Commutative Laws
$p \vee (q \vee r) \equiv (p \vee q) \vee r$ $p \wedge (q \wedge r) \equiv (p \wedge q) \wedge r$	Associative Laws
$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$	Distributive Laws

Logical Equivalences

Equivalence	Name
$p \wedge \mathbb{T} \equiv p$ $p \vee \mathbb{F} \equiv p$	Identity Laws
$p \vee \mathbb{T} \equiv \mathbb{T}$ $p \wedge \mathbb{F} \equiv \mathbb{F}$	Domination Laws
$p \vee p \equiv p$ $p \wedge p \equiv p$	Idempotent Laws
$\neg(\neg p) \equiv p$	Double Negation Law
$p \vee q \equiv q \vee p$ $p \wedge q \equiv q \wedge p$	Commutative Laws
$p \vee (q \vee r) \equiv (p \vee q) \vee r$ $p \wedge (q \wedge r) \equiv (p \wedge q) \wedge r$	Associative Laws
$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$ $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$	Distributive Laws

Logical Equivalences

Equivalence	Name
$p \wedge \mathbb{T} \equiv p$ $p \vee \mathbb{F} \equiv p$	Identity Laws
$p \vee \mathbb{T} \equiv \mathbb{T}$ $p \wedge \mathbb{F} \equiv \mathbb{F}$	Domination Laws
$p \vee p \equiv p$ $p \wedge p \equiv p$	Idempotent Laws
$\neg(\neg p) \equiv p$	Double Negation Law
$p \vee q \equiv q \vee p$ $p \wedge q \equiv q \wedge p$	Commutative Laws
$p \vee (q \vee r) \equiv (p \vee q) \vee r$ $p \wedge (q \wedge r) \equiv (p \wedge q) \wedge r$	Associative Laws
$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$ $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$	Distributive Laws
$\neg(p \wedge q) \equiv \neg p \vee \neg q$	De Morgan's Laws

Logical Equivalences

Equivalence	Name
$p \wedge \mathbb{T} \equiv p$ $p \vee \mathbb{F} \equiv p$	Identity Laws
$p \vee \mathbb{T} \equiv \mathbb{T}$ $p \wedge \mathbb{F} \equiv \mathbb{F}$	Domination Laws
$p \vee p \equiv p$ $p \wedge p \equiv p$	Idempotent Laws
$\neg(\neg p) \equiv p$	Double Negation Law
$p \vee q \equiv q \vee p$ $p \wedge q \equiv q \wedge p$	Commutative Laws
$p \vee (q \vee r) \equiv (p \vee q) \vee r$ $p \wedge (q \wedge r) \equiv (p \wedge q) \wedge r$	Associative Laws
$p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$ $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$	Distributive Laws
$\neg(p \wedge q) \equiv \neg p \vee \neg q$ $\neg(p \vee q) \equiv \neg p \wedge \neg q$	De Morgan's Laws

Practical Use of Truth Tables

We could use a truth table to determine whether two propositions are logically equivalent.

number of variables	number of rows of truth table	
1	2	$= 2^1$
2	4	$= 2^2$
3	8	$= 2^3$
4	16	$= 2^4$
5	32	$= 2^5$
6	64	$= 2^6$
10	1024	$= 2^{10}$
20	1048576	$= 2^{20}$
100	1267650600228229401496703205376	$= 2^{100}$

For propositions with large numbers of variables this can't be done by a computer in reasonbable time!

Symbolic Manipulation

- Instead of using truth tables to determine the equivalence of propositions, we can use the logical equivalences.

Symbolic Manipulation

- Instead of using truth tables to determine the equivalence of propositions, we can use the logical equivalences.
- A proposition in a compound proposition can be replaced by an equivalent proposition without changing the truth value of the compount proposition.

Symbolic Manipulation

- Instead of using truth tables to determine the equivalence of propositions, we can use the logical equivalences.
- A proposition in a compound proposition can be replaced by an equivalent proposition without changing the truth value of the compount proposition.
- Determine whether $\neg(p \vee (\neg p \wedge q))$ is logically equivalent to $\neg p \wedge \neg q$.

$$\neg(p \vee (\neg p \wedge q)) \quad \equiv$$

Symbolic Manipulation

- Instead of using truth tables to determine the equivalence of propositions, we can use the logical equivalences.
- A proposition in a compound proposition can be replaced by an equivalent proposition without changing the truth value of the compount proposition.
- Determine whether $\neg(p \vee (\neg p \wedge q))$ is logically equivalent to $\neg p \wedge \neg q$.

$$\neg(p \vee (\neg p \wedge q)) \quad \equiv \quad \neg p \wedge \neg(\neg p \wedge q) \qquad |Second\ De\ Morgan's$$

Symbolic Manipulation

- Instead of using truth tables to determine the equivalence of propositions, we can use the logical equivalences.
- A proposition in a compound proposition can be replaced by an equivalent proposition without changing the truth value of the compount proposition.
- Determine whether $\neg(p \vee (\neg p \wedge q))$ is logically equivalent to $\neg p \wedge \neg q$.

$$
\begin{aligned}
\neg(p \vee (\neg p \wedge q)) &\equiv \neg p \wedge \neg(\neg p \wedge q) && |Second\ De\ Morgan's \\
&\equiv \neg p \wedge (\neg\neg p \vee \neg q) && |First\ De\ Morgan's
\end{aligned}
$$

Symbolic Manipulation

- Instead of using truth tables to determine the equivalence of propositions, we can use the logical equivalences.
- A proposition in a compound proposition can be replaced by an equivalent proposition without changing the truth value of the compount proposition.
- Determine whether $\neg(p \vee (\neg p \wedge q))$ is logically equivalent to $\neg p \wedge \neg q$.

$$
\begin{array}{lcll}
\neg(p \vee (\neg p \wedge q)) & \equiv & \neg p \wedge \neg(\neg p \wedge q) & |\textit{Second De Morgan's} \\
 & \equiv & \neg p \wedge (\neg\neg p \vee \neg q) & |\textit{First De Morgan's} \\
 & \equiv & \neg p \wedge (p \vee \neg q) & |\textit{Double Negation}
\end{array}
$$

Symbolic Manipulation

- Instead of using truth tables to determine the equivalence of propositions, we can use the logical equivalences.
- A proposition in a compound proposition can be replaced by an equivalent proposition without changing the truth value of the compount proposition.
- Determine whether $\neg(p \vee (\neg p \wedge q))$ is logically equivalent to $\neg p \wedge \neg q$.

$$
\begin{array}{lcll}
\neg(p \vee (\neg p \wedge q)) & \equiv & \neg p \wedge \neg(\neg p \wedge q) & |\mathit{Second\ De\ Morgan's} \\
 & \equiv & \neg p \wedge (\neg\neg p \vee \neg q) & |\mathit{First\ De\ Morgan's} \\
 & \equiv & \neg p \wedge (p \vee \neg q) & |\mathit{Double\ Negation} \\
 & \equiv & (\neg p \wedge p) \vee (\neg p \wedge \neg q) & |\mathit{Distributive\ Law}
\end{array}
$$

Symbolic Manipulation

- Instead of using truth tables to determine the equivalence of propositions, we can use the logical equivalences.
- A proposition in a compound proposition can be replaced by an equivalent proposition without changing the truth value of the compount proposition.
- Determine whether $\neg(p \vee (\neg p \wedge q))$ is logically equivalent to $\neg p \wedge \neg q$.

$$
\begin{array}{lcll}
\neg(p \vee (\neg p \wedge q)) & \equiv & \neg p \wedge \neg(\neg p \wedge q) & |Second\ De\ Morgan's \\
 & \equiv & \neg p \wedge (\neg\neg p \vee \neg q) & |First\ De\ Morgan's \\
 & \equiv & \neg p \wedge (p \vee \neg q) & |Double\ Negation \\
 & \equiv & (\neg p \wedge p) \vee (\neg p \wedge \neg q) & |Distributive\ Law \\
 & \equiv & F \vee (\neg p \wedge \neg q) & |Contradiction\ p \wedge \neg p
\end{array}
$$

Symbolic Manipulation

- Instead of using truth tables to determine the equivalence of propositions, we can use the logical equivalences.
- A proposition in a compound proposition can be replaced by an equivalent proposition without changing the truth value of the compount proposition.
- Determine whether $\neg(p \vee (\neg p \wedge q))$ is logically equivalent to $\neg p \wedge \neg q$.

$$
\begin{array}{lcll}
\neg(p \vee (\neg p \wedge q)) & \equiv & \neg p \wedge \neg(\neg p \wedge q) & |Second\ De\ Morgan's \\
 & \equiv & \neg p \wedge (\neg\neg p \vee \neg q) & |First\ De\ Morgan's \\
 & \equiv & \neg p \wedge (p \vee \neg q) & |Double\ Negation \\
 & \equiv & (\neg p \wedge p) \vee (\neg p \wedge \neg q) & |Distributive\ Law \\
 & \equiv & F \vee (\neg p \wedge \neg q) & |Contradiction\ p \wedge \neg p \\
 & \equiv & (\neg p \wedge \neg q) \vee F & |Commutativity
\end{array}
$$

Symbolic Manipulation

- Instead of using truth tables to determine the equivalence of propositions, we can use the logical equivalences.
- A proposition in a compound proposition can be replaced by an equivalent proposition without changing the truth value of the compount proposition.
- Determine whether $\neg(p \vee (\neg p \wedge q))$ is logically equivalent to $\neg p \wedge \neg q$.

$$
\begin{array}{rcll}
\neg(p \vee (\neg p \wedge q)) & \equiv & \neg p \wedge \neg(\neg p \wedge q) & |Second\ De\ Morgan's \\
& \equiv & \neg p \wedge (\neg\neg p \vee \neg q) & |First\ De\ Morgan's \\
& \equiv & \neg p \wedge (p \vee \neg q) & |Double\ Negation \\
& \equiv & (\neg p \wedge p) \vee (\neg p \wedge \neg q) & |Distributive\ Law \\
& \equiv & F \vee (\neg p \wedge \neg q) & |Contradiction\ p \wedge \neg p \\
& \equiv & (\neg p \wedge \neg q) \vee F & |Commutativity \\
& \equiv & \neg p \wedge \neg q & |Identity\ Law
\end{array}
$$

Symbolic Manipulation

- Instead of using truth tables to determine the equivalence of propositions, we can use the logical equivalences.
- A proposition in a compound proposition can be replaced by an equivalent proposition without changing the truth value of the compount proposition.
- Determine whether $\neg(p \vee (\neg p \wedge q))$ is logically equivalent to $\neg p \wedge \neg q$.

$$
\begin{array}{rcll}
\neg(p \vee (\neg p \wedge q)) & \equiv & \neg p \wedge \neg(\neg p \wedge q) & |Second\ De\ Morgan's \\
& \equiv & \neg p \wedge (\neg\neg p \vee \neg q) & |First\ De\ Morgan's \\
& \equiv & \neg p \wedge (p \vee \neg q) & |Double\ Negation \\
& \equiv & (\neg p \wedge p) \vee (\neg p \wedge \neg q) & |Distributive\ Law \\
& \equiv & F \vee (\neg p \wedge \neg q) & |Contradiction\ p \wedge \neg p \\
& \equiv & (\neg p \wedge \neg q) \vee F & |Commutativity \\
& \equiv & \neg p \wedge \neg q & |Identity\ Law
\end{array}
$$

Therefore, $\neg(p \vee (\neg p \wedge q))$ is logically equivalent to $\neg p \wedge \neg q$.

Disjunctive Normal Form

Write $\neg(p \vee (q \wedge (\neg p \Rightarrow \neg r)))$ in *Disjunctive Normal Form (DNF)*.

Disjunctive Normal Form

Write $\neg(p \vee (q \wedge (\neg p \Rightarrow \neg r)))$ in *Disjunctive Normal Form (DNF)*.

p	q	r	$\neg p \Rightarrow \neg r$	$q \wedge (\neg p \Rightarrow \neg r)$	$p \vee (q \wedge (\neg p \Rightarrow \neg r))$	$\neg(p \vee (q \wedge (\neg p \Rightarrow \neg r)))$
F	F	F	T	F	F	T
F	F	T	F	F	F	T
F	T	F	T	T	T	F
F	T	T	F	F	F	T
T	F	F	T	F	T	F
T	F	T	T	F	T	F
T	T	F	T	T	T	F
T	T	T	T	T	T	F

Disjunctive Normal Form

Write $\neg(p \vee (q \wedge (\neg p \Rightarrow \neg r)))$ in *Disjunctive Normal Form (DNF)*.

p	q	r	$\neg p \Rightarrow \neg r$	$q \wedge (\neg p \Rightarrow \neg r)$	$p \vee (q \wedge (\neg p \Rightarrow \neg r))$	$\neg(p \vee (q \wedge (\neg p \Rightarrow \neg r)))$
F	F	F	T	F	F	T
F	F	T	F	F	F	T
F	T	F	T	T	T	F
F	T	T	F	F	F	T
T	F	F	T	F	T	F
T	F	T	T	F	T	F
T	T	F	T	T	T	F
T	T	T	T	T	T	F

Disjunctive Normal Form

Write $\neg(p \vee (q \wedge (\neg p \Rightarrow \neg r)))$ in *Disjunctive Normal Form (DNF)*.

p	q	r	$\neg p \Rightarrow \neg r$	$q \wedge (\neg p \Rightarrow \neg r)$	$p \vee (q \wedge (\neg p \Rightarrow \neg r))$	$\neg(p \vee (q \wedge (\neg p \Rightarrow \neg r)))$
F	F	F	T	F	F	T
F	F	T	F	F	F	T
F	T	F	T	T	T	F
F	T	T	F	F	F	T
T	F	F	T	F	T	F
T	F	T	T	F	T	F
T	T	F	T	T	T	F
T	T	T	T	T	T	F

$(\neg p \wedge \neg q \wedge \neg r)$

Disjunctive Normal Form

Write $\neg(p \vee (q \wedge (\neg p \Rightarrow \neg r)))$ in *Disjunctive Normal Form (DNF)*.

p	q	r	$\neg p \Rightarrow \neg r$	$q \wedge (\neg p \Rightarrow \neg r)$	$p \vee (q \wedge (\neg p \Rightarrow \neg r))$	$\neg(p \vee (q \wedge (\neg p \Rightarrow \neg r)))$
F	F	F	T	F	F	T
F	F	T	F	F	F	T
F	T	F	T	T	T	F
F	T	T	F	F	F	T
T	F	F	T	F	T	F
T	F	T	T	F	T	F
T	T	F	T	T	T	F
T	T	T	T	T	T	F

$(\neg p \wedge \neg q \wedge \neg r)$

Disjunctive Normal Form

Write $\neg(p \vee (q \wedge (\neg p \Rightarrow \neg r)))$ in *Disjunctive Normal Form (DNF)*.

p	q	r	$\neg p \Rightarrow \neg r$	$q \wedge (\neg p \Rightarrow \neg r)$	$p \vee (q \wedge (\neg p \Rightarrow \neg r))$	$\neg(p \vee (q \wedge (\neg p \Rightarrow \neg r)))$
F	F	F	T	F	F	T
F	F	T	F	F	F	T
F	T	F	T	T	T	F
F	T	T	F	F	F	T
T	F	F	T	F	T	F
T	F	T	T	F	T	F
T	T	F	T	T	T	F
T	T	T	T	T	T	F

$(\neg p \wedge \neg q \wedge \neg r) \vee (\neg p \wedge \neg q \wedge r)$

Disjunctive Normal Form

Write $\neg(p \vee (q \wedge (\neg p \Rightarrow \neg r)))$ in *Disjunctive Normal Form (DNF)*.

p	q	r	$\neg p \Rightarrow \neg r$	$q \wedge (\neg p \Rightarrow \neg r)$	$p \vee (q \wedge (\neg p \Rightarrow \neg r))$	$\neg(p \vee (q \wedge (\neg p \Rightarrow \neg r)))$
F	F	F	T	F	F	T
F	F	T	F	F	F	T
F	T	F	T	T	T	F
F	T	T	F	F	F	T
T	F	F	T	F	T	F
T	F	T	T	F	T	F
T	T	F	T	T	T	F
T	T	T	T	T	T	F

$(\neg p \wedge \neg q \wedge \neg r) \vee (\neg p \wedge \neg q \wedge r)$

Disjunctive Normal Form

Write $\neg(p \vee (q \wedge (\neg p \Rightarrow \neg r)))$ in *Disjunctive Normal Form (DNF)*.

p	q	r	$\neg p \Rightarrow \neg r$	$q \wedge (\neg p \Rightarrow \neg r)$	$p \vee (q \wedge (\neg p \Rightarrow \neg r))$	$\neg(p \vee (q \wedge (\neg p \Rightarrow \neg r)))$
F	F	F	T	F	F	T
F	F	T	F	F	F	T
F	T	F	T	T	T	F
F	T	T	F	F	F	T
T	F	F	T	F	T	F
T	F	T	T	F	T	F
T	T	F	T	T	T	F
T	T	T	T	T	T	F

$(\neg p \wedge \neg q \wedge \neg r) \vee (\neg p \wedge \neg q \wedge r) \vee (\neg p \wedge q \wedge r)$

Disjunctive Normal Form

Write $\neg(p \vee (q \wedge (\neg p \Rightarrow \neg r)))$ in *Disjunctive Normal Form (DNF)*.

p	q	r	$\neg p \Rightarrow \neg r$	$q \wedge (\neg p \Rightarrow \neg r)$	$p \vee (q \wedge (\neg p \Rightarrow \neg r))$	$\neg(p \vee (q \wedge (\neg p \Rightarrow \neg r)))$
F	F	F	T	F	F	T
F	F	T	F	F	F	T
F	T	F	T	T	T	F
F	T	T	F	F	F	T
T	F	F	T	F	T	F
T	F	T	T	F	T	F
T	T	F	T	T	T	F
T	T	T	T	T	T	F

$(\neg p \wedge \neg q \wedge \neg r) \vee (\neg p \wedge \neg q \wedge r) \vee (\neg p \wedge q \wedge r)$

- Every wff can be expressed in DNF.

Disjunctive Normal Form

Write $\neg(p \vee (q \wedge (\neg p \Rightarrow \neg r)))$ in *Disjunctive Normal Form (DNF)*.

p	q	r	$\neg p \Rightarrow \neg r$	$q \wedge (\neg p \Rightarrow \neg r)$	$p \vee (q \wedge (\neg p \Rightarrow \neg r))$	$\neg(p \vee (q \wedge (\neg p \Rightarrow \neg r)))$
F	F	F	T	F	F	T
F	F	T	F	F	F	T
F	T	F	T	T	T	F
F	T	T	F	F	F	T
T	F	F	T	F	T	F
T	F	T	T	F	T	F
T	T	F	T	T	T	F
T	T	T	T	T	T	F

$(\neg p \wedge \neg q \wedge \neg r) \vee (\neg p \wedge \neg q \wedge r) \vee (\neg p \wedge q \wedge r)$

- Every wff can be expressed in DNF.
- DNF only uses the connectives conjuction, disjunction and negation.

Disjunctive Normal Form

Write $\neg(p \vee (q \wedge (\neg p \Rightarrow \neg r)))$ in *Disjunctive Normal Form (DNF)*.

p	q	r	$\neg p \Rightarrow \neg r$	$q \wedge (\neg p \Rightarrow \neg r)$	$p \vee (q \wedge (\neg p \Rightarrow \neg r))$	$\neg(p \vee (q \wedge (\neg p \Rightarrow \neg r)))$
F	F	F	T	F	F	T
F	F	T	F	F	F	T
F	T	F	T	T	T	F
F	T	T	F	F	F	T
T	F	F	T	F	T	F
T	F	T	T	F	T	F
T	T	F	T	T	T	F
T	T	T	T	T	T	F

$(\neg p \wedge \neg q \wedge \neg r) \vee (\neg p \wedge \neg q \wedge r) \vee (\neg p \wedge q \wedge r)$

- Every wff can be expressed in DNF.
- DNF only uses the connectives conjuction, disjunction and negation.
- The set $\{\wedge, \vee, \neg\}$ is said to be *functionally complete*.

Functional Completeness

- The set $\{\wedge, \vee, \neg\}$ is said to be *functionally complete*.

Functional Completeness

- The set $\{\wedge, \vee, \neg\}$ is said to be *functionally complete*.
- **Is this the smallest set of functionally complete connectives?**

Functional Completeness

- The set $\{\wedge, \vee, \neg\}$ is said to be *functionally complete*.
- **Is this the smallest set of functionally complete connectives?**
- Can one of the three connectives be expressed in terms of the other two?

Functional Completeness

- The set $\{\wedge, \vee, \neg\}$ is said to be *functionally complete*.
- **Is this the smallest set of functionally complete connectives?**
- Can one of the three connectives be expressed in terms of the other two?
- De Morgan's Laws:

$$\begin{aligned} \neg(p \wedge q) &\equiv \neg p \vee \neg q \\ \neg(p \vee q) &\equiv \neg p \wedge \neg q \end{aligned}$$

Functional Completeness

- The set $\{\wedge, \vee, \neg\}$ is said to be *functionally complete*.
- **Is this the smallest set of functionally complete connectives?**
- Can one of the three connectives be expressed in terms of the other two?
- De Morgan's Laws:

$$\begin{aligned} \neg(p \wedge q) &\equiv \neg p \vee \neg q \\ \neg(p \vee q) &\equiv \neg p \wedge \neg q \end{aligned}$$

- $p \vee q = \neg(\neg(p \vee q)) = \neg(\neg p \wedge \neg q)$

Functional Completeness

- The set $\{\wedge, \vee, \neg\}$ is said to be *functionally complete*.
- **Is this the smallest set of functionally complete connectives?**
- Can one of the three connectives be expressed in terms of the other two?
- De Morgan's Laws:

$$\begin{aligned} \neg(p \wedge q) &\equiv \neg p \vee \neg q \\ \neg(p \vee q) &\equiv \neg p \wedge \neg q \end{aligned}$$

- $p \vee q = \neg(\neg(p \vee q)) = \neg(\neg p \wedge \neg q)$
- Therefore, $\{\wedge, \neg\}$ is a set of functionally complete connectives.

Functional Completeness

- The set $\{\wedge, \vee, \neg\}$ is said to be *functionally complete*.
- **Is this the smallest set of functionally complete connectives?**
- Can one of the three connectives be expressed in terms of the other two?
- De Morgan's Laws:

$$\begin{aligned} \neg(p \wedge q) &\equiv \neg p \vee \neg q \\ \neg(p \vee q) &\equiv \neg p \wedge \neg q \end{aligned}$$

- $p \vee q = \neg(\neg(p \vee q)) = \neg(\neg p \wedge \neg q)$
- Therefore, $\{\wedge, \neg\}$ is a set of functionally complete connectives.
- $p \wedge q = \neg(\neg(p \wedge q)) = \neg(\neg p \vee \neg q))$

Functional Completeness

- The set $\{\wedge, \vee, \neg\}$ is said to be *functionally complete*.
- **Is this the smallest set of functionally complete connectives?**
- Can one of the three connectives be expressed in terms of the other two?
- De Morgan's Laws:

$$\begin{aligned} \neg(p \wedge q) &\equiv \neg p \vee \neg q \\ \neg(p \vee q) &\equiv \neg p \wedge \neg q \end{aligned}$$

- $p \vee q = \neg(\neg(p \vee q)) = \neg(\neg p \wedge \neg q)$
- Therefore, $\{\wedge, \neg\}$ is a set of functionally complete connectives.
- $p \wedge q = \neg(\neg(p \wedge q)) = \neg(\neg p \vee \neg q))$
- The set $\{\vee, \neg\}$ is also functionally complete.

Functional Completeness - NAND

- **Is there an even smaller set, a set containing just a single connective, that is functionally complete?**

Functional Completeness - NAND

- **Is there an even smaller set, a set containing just a single connective, that is functionally complete?**
- *NAND*, denoted $\uparrow$, is a connective with the following truth table:

p	q	$\neg(p \wedge q)$	$p \uparrow q$
T	T	F	F
T	F	T	T
F	T	T	T
F	F	T	T

Functional Completeness - NAND

- **Is there an even smaller set, a set containing just a single connective, that is functionally complete?**
- *NAND*, denoted $\uparrow$, is a connective with the following truth table:

p	q	$\neg(p \wedge q)$	$p \uparrow q$
T	T	F	F
T	F	T	T
F	T	T	T
F	F	T	T

- $\{\uparrow\}$ **is functionally complete.**

Functional Completeness - NAND

- **Is there an even smaller set, a set containing just a single connective, that is functionally complete?**
- *NAND*, denoted $\uparrow$, is a connective with the following truth table:

p	q	$\neg(p \wedge q)$	$p \uparrow q$
T	T	F	F
T	F	T	T
F	T	T	T
F	F	T	T

- $\{\uparrow\}$ **is functionally complete.**
- We already know that $\{\wedge, \neg\}$ is functionally complete.

Functional Completeness - NAND

- **Is there an even smaller set, a set containing just a single connective, that is functionally complete?**
- *NAND*, denoted $\uparrow$, is a connective with the following truth table:

p	q	$\neg(p \wedge q)$	$p \uparrow q$
T	T	F	F
T	F	T	T
F	T	T	T
F	F	T	T

- **$\{\uparrow\}$ is functionally complete.**
- We already know that $\{\wedge, \neg\}$ is functionally complete.
- We only need to demonstrate that both $\wedge$ and $\neg$ can be expressed using just $\uparrow$.

Functional Completeness - NAND

- **Is there an even smaller set, a set containing just a single connective, that is functionally complete?**
- *NAND*, denoted $\uparrow$, is a connective with the following truth table:

p	q	$\neg(p \wedge q)$	$p \uparrow q$
T	T	F	F
T	F	T	T
F	T	T	T
F	F	T	T

- **$\{\uparrow\}$ is functionally complete.**
- We already know that $\{\wedge, \neg\}$ is functionally complete.
- We only need to demonstrate that both $\wedge$ and $\neg$ can be expressed using just $\uparrow$.
- This can be done as follows:

$$\begin{aligned} \neg p &\equiv p \uparrow p \\ p \wedge q &\equiv (p \uparrow q) \uparrow (p \uparrow q) \end{aligned}$$

(How can you check these equivalences?)

Functional Completeness - NOR

- The connective *NOR*, denoted $\downarrow$, has the following truth table:

p	q	$\neg(p \vee q)$	$p \downarrow q$
T	T	F	F
T	F	F	F
F	T	F	F
F	F	T	T

Functional Completeness - NOR

- The connective *NOR*, denoted $\downarrow$, has the following truth table:

p	q	$\neg(p \vee q)$	$p \downarrow q$
T	T	F	F
T	F	F	F
F	T	F	F
F	F	T	T

- $\{\downarrow\}$ **is also functionally complete.**

Functional Completeness - NOR

- The connective *NOR*, denoted $\downarrow$, has the following truth table:

p	q	$\neg(p \lor q)$	$p \downarrow q$
T	T	F	F
T	F	F	F
F	T	F	F
F	F	T	T

- $\{\downarrow\}$ **is also functionally complete.**
- SSE: Demonstrate that $\{\downarrow\}$ is functionally complete.

Summary

- Evaluation of compound propositions

Summary

- Evaluation of compound propositions
- Notions of validity and satisfiability

Summary

- Evaluation of compound propositions
- Notions of validity and satisfiability
- Logic equivalence

Summary

- Evaluation of compound propositions
- Notions of validity and satisfiability
- Logic equivalence
- Symbolic manipuation using logical equivalences

Summary

- Evaluation of compound propositions
- Notions of validity and satisfiability
- Logic equivalence
- Symbolic manipuation using logical equivalences
- Sets of functionally complete connectives, special role of NAND and NOR

Summary

- Evaluation of compound propositions
- Notions of validity and satisfiability
- Logic equivalence
- Symbolic manipuation using logical equivalences
- Sets of functionally complete connectives, special role of NAND and NOR
- (Semantics of propositional logic)

Formal Semantics of Propositional Logic (SSE)

- More formally, an **interpretation** is an assignment of either T or F to each statement of the language.

Formal Semantics of Propositional Logic (SSE)

- More formally, an **interpretation** is an assignment of either T or F to each statement of the language.
- Associated with each interpretation is a valuation function $I_f : PROP \rightarrow \{T, F\}$ which assigns a truth value to each atomic proposition.

Formal Semantics of Propositional Logic (SSE)

- More formally, an **interpretation** is an assignment of either T or F to each statement of the language.
- Associated with each interpretation is a valuation function $I_f : PROP \rightarrow \{T, F\}$ which assigns a truth value to each atomic proposition.
- Given an interpretation I, we can say for any formula whether it is true or false.

Formal Semantics of Propositional Logic (SSE)

- More formally, an **interpretation** is an assignment of either T or F to each statement of the language.
- Associated with each interpretation is a valuation function $I_f : PROP \rightarrow \{T, F\}$ which assigns a truth value to each atomic proposition.
- Given an interpretation I, we can say for any formula whether it is true or false.

  I models **true**

Formal Semantics of Propositional Logic (SSE)

- More formally, an **interpretation** is an assignment of either T or F to each statement of the language.
- Associated with each interpretation is a valuation function $I_f : PROP \rightarrow \{T, F\}$ which assigns a truth value to each atomic proposition.
- Given an interpretation I, we can say for any formula whether it is true or false.

  I models **true**

  I does not model **false**

Formal Semantics of Propositional Logic (SSE)

- More formally, an **interpretation** is an assignment of either T or F to each statement of the language.
- Associated with each interpretation is a valuation function $I_f : PROP \rightarrow \{T, F\}$ which assigns a truth value to each atomic proposition.
- Given an interpretation I, we can say for any formula whether it is true or false.

  I models **true**

  I does not model **false**

  I models p if, and only if $I_f(p) = T$

Formal Semantics of Propositional Logic (SSE)

- More formally, an **interpretation** is an assignment of either T or F to each statement of the language.
- Associated with each interpretation is a valuation function $I_f : PROP \rightarrow \{T, F\}$ which assigns a truth value to each atomic proposition.
- Given an interpretation I, we can say for any formula whether it is true or false.

  I models **true**

  I does not model **false**

  I models p if, and only if $I_f(p) = T$

  I models $\neg A$ if, and only if I does not model A

Formal Semantics of Propositional Logic (SSE)

- More formally, an **interpretation** is an assignment of either T or F to each statement of the language.
- Associated with each interpretation is a valuation function $I_f : PROP \rightarrow \{T, F\}$ which assigns a truth value to each atomic proposition.
- Given an interpretation I, we can say for any formula whether it is true or false.

  I models **true**

  I does not model **false**

  I models p if, and only if $I_f(p) = T$

  I models $\neg A$ if, and only if I does not model A

  I models $A \wedge B$ if, and only if I models A and I models B

Formal Semantics of Propositional Logic (SSE)

- More formally, an **interpretation** is an assignment of either T or F to each statement of the language.
- Associated with each interpretation is a valuation function $I_f : PROP \rightarrow \{T, F\}$ which assigns a truth value to each atomic proposition.
- Given an interpretation I, we can say for any formula whether it is true or false.

  I models **true**
  I does not model **false**
  I models p if, and only if $I_f(p) = T$
  I models $\neg A$ if, and only if I does not model A
  I models $A \wedge B$ if, and only if I models A and I models B
  I models $A \vee B$ if, and only if I models A or I models B

Formal Semantics of Propositional Logic (SSE)

- More formally, an **interpretation** is an assignment of either T or F to each statement of the language.
- Associated with each interpretation is a valuation function $I_f : PROP \rightarrow \{T, F\}$ which assigns a truth value to each atomic proposition.
- Given an interpretation I, we can say for any formula whether it is true or false.

$$\begin{array}{l} I \text{ models } \textbf{true} \\ I \text{ does not model } \textbf{false} \\ I \text{ models } p \text{ if, and only if } I_f(p) = T \\ I \text{ models } \neg A \text{ if, and only if } I \text{ does not model } A \\ I \text{ models } A \wedge B \text{ if, and only if } I \text{ models } A \text{ and } I \text{ models } B \\ I \text{ models } A \vee B \text{ if, and only if } I \text{ models } A \text{ or } I \text{ models } B \\ I \text{ models } A \oplus B \text{ if, and only if} \ldots \end{array} \quad \text{(SSE)}$$

Formal Semantics of Propositional Logic (SSE)

- More formally, an **interpretation** is an assignment of either T or F to each statement of the language.
- Associated with each interpretation is a valuation function $I_f : PROP \rightarrow \{T, F\}$ which assigns a truth value to each atomic proposition.
- Given an interpretation I, we can say for any formula whether it is true or false.

$$
\begin{array}{l}
I \text{ models } \textbf{true} \\
I \text{ does not model } \textbf{false} \\
I \text{ models } p \text{ if, and only if } I_f(p) = T \\
I \text{ models } \neg A \text{ if, and only if } I \text{ does not model } A \\
I \text{ models } A \wedge B \text{ if, and only if } I \text{ models } A \text{ and } I \text{ models } B \\
I \text{ models } A \vee B \text{ if, and only if } I \text{ models } A \text{ or } I \text{ models } B \\
I \text{ models } A \oplus B \text{ if, and only if} \ldots \\
I \text{ models } A \Rightarrow B \text{ if, and only if if } I \text{ models } A \text{ then } I \text{ models } B
\end{array}
\tag{SSE}
$$

Formal Semantics of Propositional Logic (SSE)

- More formally, an **interpretation** is an assignment of either T or F to each statement of the language.
- Associated with each interpretation is a valuation function $I_f : PROP \rightarrow \{T, F\}$ which assigns a truth value to each atomic proposition.
- Given an interpretation I, we can say for any formula whether it is true or false.

  I models **true**

  I does not model **false**

  I models p if, and only if $I_f(p) = T$

  I models $\neg A$ if, and only if I does not model A

  I models $A \wedge B$ if, and only if I models A and I models B

  I models $A \vee B$ if, and only if I models A or I models B

  I models $A \oplus B$ if, and only if ... (SSE)

  I models $A \Rightarrow B$ if, and only if if I models A then I models B

  I models $A \Leftrightarrow B$ if, and only if ... (SSE)

Formal Semantics of Propositional Logic (SSE)

- More formally, an **interpretation** is an assignment of either T or F to each statement of the language.
- Associated with each interpretation is a valuation function $I_f : PROP \rightarrow \{T, F\}$ which assigns a truth value to each atomic proposition.
- Given an interpretation I, we can say for any formula whether it is true or false.

  I models **true**

  I does not model **false**

  I models p if, and only if $I_f(p) = T$

  I models $\neg A$ if, and only if I does not model A

  I models $A \wedge B$ if, and only if I models A and I models B

  I models $A \vee B$ if, and only if I models A or I models B

  I models $A \oplus B$ if, and only if ... (SSE)

  I models $A \Rightarrow B$ if, and only if if I models A then I models B

  I models $A \Leftrightarrow B$ if, and only if ... (SSE)

- "models" is formally denoted using the *semantic turnstile* symbol $\models$.

Formal Semantics of Propositional Logic (SSE)

- More formally, an **interpretation** is an assignment of either T or F to each statement of the language.
- Associated with each interpretation is a valuation function $I_f : PROP \rightarrow \{T, F\}$ which assigns a truth value to each atomic proposition.
- Given an interpretation I, we can say for any formula whether it is true or false.

$I \models$ **true**

$I \not\models$ **false**

$I \models p$ if, and only if $I_f(p) = T$

$I \models \neg A$ if, and only if $I \not\models A$

$I \models A \wedge B$ if, and only if $I \models A$ and $I \models B$

$I \models A \vee B$ if, and only if $I \models A$ or $I \models B$

$I \models A \oplus B$ if, and only if ... (SSE)

$I \models A \Rightarrow B$ if, and only if if $I \models A$ then $I \models B$

$I \models A \Leftrightarrow B$ if, and only if ... (SSE)

Formal Semantics of Propositional Logic (SSE)

- More formally, an **interpretation** is an assignment of either T or F to each statement of the language.
- Associated with each interpretation is a valuation function $I_f : PROP \rightarrow \{T, F\}$ which assigns a truth value to each atomic proposition.
- Given an interpretation I, we can say for any formula whether it is true or false.

$$I \models \textbf{true}$$
$$I \not\models \textbf{false}$$
$$I \models p \text{ if, and only if } I_f(p) = T$$
$$I \models \neg A \text{ if, and only if } I \not\models A$$
$$I \models A \wedge B \text{ if, and only if } I \models A \text{ and } I \models B$$
$$I \models A \vee B \text{ if, and only if } I \models A \text{ or } I \models B$$
$$I \models A \oplus B \text{ if, and only if} \ldots \qquad \text{(SSE)}$$
$$I \models A \Rightarrow B \text{ if, and only if if } \ I \models A \ \text{ then } \ I \models B$$
$$I \models A \Leftrightarrow B \text{ if, and only if} \ldots \qquad \text{(SSE)}$$

- $I \models A$ can be read as *I satisfies A* or *I models* A.