Principles of Knowledge Representation and Reasoning

2. Reminder: Classical Logic

2.1 Propositional Logic

Bernhard Nebel

- Motivation
- Terminology
- Syntax & Semantics
- Normal Forms
- Entailment & Derivability
Why Logic?

- Logic is the best developed system for representing knowledge

- Can be used for analysis, design and specification

- Without knowledge in formal logic, most research papers in KRR cannot be understood
The Right Logic …

- Logic of different orders
The Right Logic ...

- Logic of different orders
- Modal logics
The Right Logic …

- Logic of different orders
- **Modal** logics
  - epistemic
  - temporal
  - multi-
  - …
The Right Logic …

- Logic of different orders
- Modal logics
  - epistemic
  - temporal
  - multi-
  - …
- Conditional logics
The Right Logic …

- Logic of different **orders**
- **Modal** logics
  - epistemic
  - temporal
  - multi-
  - …
- **Conditional** logics
- **Nonmonotonic** logics
The Right Logic …

- Logic of different orders
- **Modal** logics
  - epistemic
  - temporal
  - multi-
  - ...
- **Conditional** logics
- **Nonmonotonic** logics
- **Linear** logics
The Right Logic …

- Logic of different orders
- **Modal** logics
  - epistemic
  - temporal
  - multi-
  - …
- **Conditional** logics
- **Nonmonotonic** logics
- **Linear** logics
- …
The Logical Approach

- Define a **formal language**

  logical & non-logical symbols, syntax rules
The Logical Approach

- Define a **formal language**
  - logical & non-logical symbols, syntax rules
- Provide language with **compositional semantics**
The Logical Approach

- Define a **formal language**

  ➞ logical & non-logical symbols, syntax rules

- Provide language with **compositional semantics**
  - Fix **universe** of discourse
  - Specify how the non-logical symbols can be **interpreted**

  ➞ interpretation
  - Rules how to **combine** interpretation of single symbols
  - **Satisfying interpretation** = **model**
The Logical Approach

- Define a **formal language**
  - logical & non-logical symbols, syntax rules

- Provide language with **compositional semantics**
  - Fix **universe** of discourse
  - Specify how the non-logical symbols can be **interpreted**
  - **interpretation**
    - Rules how to **combine** interpretation of single symbols
    - **Satisfying interpretation = model**
  - From that **logical implication/entailment** follows
The Logical Approach

- Define a **formal language**
  - logical & non-logical symbols, syntax rules

- Provide language with **compositional semantics**
  - Fix **universe** of discourse
  - Specify how the non-logical symbols can be **interpreted**
  - **interpretation**
    - Rules how to **combine** interpretation of single symbols
    - **Satisfying interpretation** = **model**
    - From that **logical implication/entailment** follows

- Specify a **calculus** that allows to **derive** new formulae from old ones – according to the entailment relation
Propositional Logic: Main Ideas

- **Non-logical symbols**: propositional *variables* or *atoms*
  - representing *propositions* which cannot be decomposed
  - which can be *true* or *false*
Propositional Logic: Main Ideas

- **Non-logical symbols**: propositional variables or atoms
  - representing propositions which cannot be decomposed
  - which can be true or false
  - for example:
    - “Snow is white”
    - “It rains”
Propositional Logic: Main Ideas

- **Non-logical symbols**: propositional variables or atoms
  - representing propositions which cannot be decomposed
  - which can be true or false
  - for example:
    - → “Snow is white”
    - → “It rains”

- **Logical Symbols**: propositional connectives such as and ($\land$), or ($\lor$), and not ($\neg$).
Propositional Logic: Main Ideas

- **Non-logical symbols**: propositional variables or atoms
  - representing propositions which cannot be decomposed
  - which can be true or false
  - for example:
    - “Snow is white”
    - “It rains”

- **Logical Symbols**: propositional connectives such as and (∧), or (∨), and not (¬).

- **Formulae**: built out of atoms and connectives

- **Universe of discourse**: truth values
Syntax

Countable alphabet $\Sigma$ of atomic propositions: $a, b, c, \ldots$ ($\Sigma_n$ finite alphabet with $n$ atoms)

Propositional formulae are built according to the following rule:
Syntax

Countable alphabet $\Sigma$ of atomic propositions: $a, b, c, \ldots$ ($\Sigma_n$ finite alphabet with $n$ atoms)

Propositional formulae are built according to the following rule:

$$\varphi \rightarrow a \quad \text{atomic formula}$$
Syntax

Countable alphabet $\Sigma$ of atomic propositions: $a, b, c, \ldots$ ($\Sigma_n$ finite alphabet with $n$ atoms)

Propositional formulae are built according to the following rule:

$$\varphi \rightarrow a \quad \text{atomic formula}$$

$$| \quad \bot \quad \text{falsity}$$
Syntax

Countable alphabet $\Sigma$ of atomic propositions: $a, b, c, \ldots$ ($\Sigma_n$ finite alphabet with $n$ atoms)

Propositional formulae are built according to the following rule:

$$
\varphi \quad \longrightarrow \quad a \quad \text{atomic formula}
$$

$$
\begin{array}{c|c}
| \varphi & \top \quad \text{truth} \\
\end{array}
$$
Syntax

Countable alphabet $\Sigma$ of \textbf{atomic propositions}: $a, b, c, \ldots$ ($\Sigma_n$ finite alphabet with $n$ atoms)

Propositional formulae are built according to the following \textbf{rule}:

\[
\begin{array}{c|c}
\varphi & a & \text{atomic formula} \\
\hline
\bot & \text{falsity} \\
\top & \text{truth} \\
\neg \varphi' & \text{negation}
\end{array}
\]
Syntax

Countable alphabet $\Sigma$ of atomic propositions: $a, b, c, \ldots$ ($\Sigma_n$ finite alphabet with $n$ atoms)

Propositional formulae are built according to the following rule:

$$
\begin{align*}
\varphi & \quad \rightarrow \quad a \quad \text{atomic formula} \\
| & \quad \bot \quad \text{falsity} \\
| & \quad \top \quad \text{truth} \\
| & \quad (\neg \varphi') \quad \text{negation} \\
| & \quad (\varphi' \land \varphi'') \quad \text{conjunction}
\end{align*}
$$
Syntax

Countable alphabet $\Sigma$ of atomic propositions: $a, b, c, \ldots$ ($\Sigma_n$ finite alphabet with $n$ atoms)

Propositional formulae are built according to the following rule:

$$\varphi \rightarrow a \quad \text{atomic formula}$$

$$\mid \bot \quad \text{falsity}$$

$$\mid \top \quad \text{truth}$$

$$\mid (\neg \varphi') \quad \text{negation}$$

$$\mid (\varphi' \land \varphi'') \quad \text{conjunction}$$

$$\mid (\varphi' \lor \varphi'') \quad \text{disjunction}$$
Syntax

Countable alphabet $\Sigma$ of **atomic propositions**: $a, b, c, \ldots$ ($\Sigma_n$ finite alphabet with $n$ atoms)

Propositional formulae are built according to the following **rule**:

$$
\begin{align*}
\varphi & \rightarrow a & \text{atomic formula} \\
& | \bot & \text{falsity} \\
& | \top & \text{truth} \\
& | (\neg \varphi') & \text{negation} \\
& | (\varphi' \land \varphi'') & \text{conjunction} \\
& | (\varphi' \lor \varphi'') & \text{disjunction} \\
& | (\varphi' \rightarrow \varphi'') & \text{implication}
\end{align*}
$$
Syntax

Countable alphabet $\Sigma$ of atomic propositions: $a, b, c, \ldots$ ($\Sigma_n$ finite alphabet with $n$ atoms)

Propositional formulae are built according to the following rule:

- $\varphi \rightarrow a$ atomic formula
- $\bot$ falsity
- $\top$ truth
- $\neg \varphi'$ negation
- $(\varphi' \land \varphi''')$ conjunction
- $(\varphi' \lor \varphi''')$ disjunction
- $(\varphi' \rightarrow \varphi''')$ implication
- $(\varphi' \leftrightarrow \varphi''')$ equivalence
Syntax

Countable alphabet $\Sigma$ of **atomic propositions**: $a, b, c, \ldots$ ($\Sigma_n$ finite alphabet with $n$ atoms)

Propositional formulae are built according to the following **rule**:

- $\varphi \rightarrow a$  \hspace{1cm} **atomic formula**
- $\bot$  \hspace{1cm} **falsity**
- $\top$  \hspace{1cm} **truth**
- $(\neg \varphi')$  \hspace{1cm} **negation**
- $(\varphi' \land \varphi'')$  \hspace{1cm} **conjunction**
- $(\varphi' \lor \varphi'')$  \hspace{1cm} **disjunction**
- $(\varphi' \rightarrow \varphi'')$  \hspace{1cm} **implication**
- $(\varphi' \leftrightarrow \varphi'')$  \hspace{1cm} **equivalence**

Parenthesis can be omitted if no ambiguity arises
Countable alphabet $\Sigma$ of **atomic propositions**: $a, b, c, \ldots$ ($\Sigma_n$ finite alphabet with $n$ atoms)

Propositional formulae are built according to the following **rule**:

$$
\begin{align*}
\varphi & \quad \rightarrow \quad a \quad \text{atomic formula} \\
& \quad \mid \quad \bot \quad \text{falsity} \\
& \quad \mid \quad \top \quad \text{truth} \\
& \quad \mid \quad (\neg \varphi') \quad \text{negation} \\
& \quad \mid \quad (\varphi' \land \varphi'') \quad \text{conjunction} \\
& \quad \mid \quad (\varphi' \lor \varphi'') \quad \text{disjunction} \\
& \quad \mid \quad (\varphi' \rightarrow \varphi'') \quad \text{implication} \\
& \quad \mid \quad (\varphi' \leftrightarrow \varphi'') \quad \text{equivalence}
\end{align*}
$$

Parenthesis can be omitted if no ambiguity arises

**Operator precedence**: $\neg > \land > \lor > \rightarrow = \leftrightarrow$. 
(a ∨ b) is an expression of the language of propositional logic
Language and Meta-Language

- $(a \lor b)$ is an expression of the language of propositional logic

- $\varphi \rightarrow a \ldots |(\varphi' \leftrightarrow \varphi'')$ is a statement about how expressions in the language of propositional logic can be formed. It is stated using meta-language
Language and Meta-Language

- \((a \lor b)\) is an expression of the language of **propositional logic**

- \(\varphi \longrightarrow a \mid \ldots \mid (\varphi' \leftrightarrow \varphi'')\) is a statement about how expressions in the language of propositional logic can be formed. It is stated using **meta-language**

- In order to describe how expressions (in this case formulae) can be formed, we use meta-language.

- When we describe how to interpret formulae, we use meta-language expressions
Semantics: Idea

• Atomic propositions can be true \((1, T)\) or false \((0, F)\).
Atomic propositions can be **true** \((1, T)\) or **false** \((0, F)\).

Provided the truth values of the atoms have been fixed (**truth assignment** or **interpretation**), the truth value of a formula can be computed from the truth values of the atoms and the connectives.
Atomic propositions can be true \((1, T)\) or false \((0, F)\).

Provided the truth values of the atoms have been fixed (truth assignment or interpretation), the truth value of a formula can be computed from the truth values of the atoms and the connectives.

**Example:**

\[
(a \lor b) \land c
\]

is true *iff* \(c\) is true and additionally \(a\) or \(b\) is true.
• Atomic propositions can be true \( (1, T) \) or false \( (0, F) \).

• Provided the truth values of the atoms have been fixed (truth assignment or interpretation), the truth value of a formula can be computed from the truth values of the atoms and the connectives.

• **Example:**

\[
(a \lor b) \land c
\]

is true \textit{iff} \( c \) is true and additionally \( a \) or \( b \) is true.

• Logical implication can then be defined as follows:

\[\leadsto \varphi \text{ is implied by the formulae } \Theta \text{ iff } \varphi \text{ is true for all truth assignments (world states) that make all formulae in } \Theta \text{ true.}\]
An interpretation or truth assignment over $\Sigma$ is a function: $I: \Sigma \rightarrow \{T, F\}$. 
Formal Semantics

An **interpretation** or **truth assignment** over $\Sigma$ is a function: $I: \Sigma \rightarrow \{T, F\}$.

A formula $\psi$ is **true under** $I$ or is **satisfied by** $I$ (symbolically $I \models \psi$):

$$I \models \psi \iff I(\psi) = T$$
Formal Semantics

An **interpretation** or **truth assignment** over $\Sigma$ is a function: $\mathcal{I} : \Sigma \rightarrow \{T, F\}$.

A formula $\psi$ is **true under** $\mathcal{I}$ or is **satisfied by** $\mathcal{I}$ (symbolically $\mathcal{I} \models \psi$):

\[
\mathcal{I} \models a \quad \text{iff} \quad \mathcal{I}(a) = T
\]

\[
\mathcal{I} \models \top
\]

\[
\mathcal{I} \not\models \bot
\]
An **interpretation** or **truth assignment** over $\Sigma$ is a function: $I: \Sigma \rightarrow \{T, F\}$.

A formula $\psi$ is **true under** $I$ or is **satisfied by** $I$ (symbolically $I \models \psi$):

- $I \models a$ iff $I(a) = T$
- $I \models \top$
- $I \not\models \bot$
- $I \models \neg \varphi$ iff $I \not\models \varphi$
Formal Semantics

An interpretation or truth assignment over $\Sigma$ is a function: $I: \Sigma \rightarrow \{T, F\}$.

A formula $\psi$ is true under $I$ or is satisfied by $I$ (symbolically $I \models \psi$):

- $I \models a$ iff $I(a) = T$
- $I \models \top$
- $I \not\models \bot$
- $I \models \neg \varphi$ iff $I \not\models \varphi$
- $I \models \varphi \land \varphi'$ iff $I \models \varphi$ and $I \models \varphi'$
An interpretation or truth assignment over $\Sigma$ is a function: $\mathcal{I}: \Sigma \rightarrow \{T, F\}$.

A formula $\psi$ is true under $\mathcal{I}$ or is satisfied by $\mathcal{I}$ (symbolically $\mathcal{I} \models \psi$):

- $\mathcal{I} \models a$ if and only if $\mathcal{I}(a) = T$
- $\mathcal{I} \models \top$
- $\mathcal{I} \not\models \bot$
- $\mathcal{I} \models \neg \varphi$ if and only if $\mathcal{I} \not\models \varphi$
- $\mathcal{I} \models \varphi \land \varphi'$ if and only if $\mathcal{I} \models \varphi$ and $\mathcal{I} \models \varphi'$
- $\mathcal{I} \models \varphi \lor \varphi'$ if and only if $\mathcal{I} \models \varphi$ or $\mathcal{I} \models \varphi'$
An interpretation or truth assignment over \( \Sigma \) is a function: \( \mathcal{I}: \Sigma \to \{T, F\} \).

A formula \( \psi \) is true under \( \mathcal{I} \) or is satisfied by \( \mathcal{I} \) (symbolically \( \mathcal{I} \models \psi \)):

\[
\begin{align*}
\mathcal{I} \models a & \quad \text{iff} \quad \mathcal{I}(a) = T \\
\mathcal{I} \models T & \\
\mathcal{I} \models \bot & \\
\mathcal{I} \models \neg \varphi & \quad \text{iff} \quad \mathcal{I} \not\models \varphi \\
\mathcal{I} \models \varphi \land \varphi' & \quad \text{iff} \quad \mathcal{I} \models \varphi \text{ and } \mathcal{I} \models \varphi' \\
\mathcal{I} \models \varphi \lor \varphi' & \quad \text{iff} \quad \mathcal{I} \models \varphi \text{ or } \mathcal{I} \models \varphi' \\
\mathcal{I} \models \varphi \rightarrow \varphi' & \quad \text{iff} \quad \text{if } \mathcal{I} \models \varphi, \text{ then } \mathcal{I} \models \varphi'
\end{align*}
\]
Formal Semantics

An **interpretation** or **truth assignment** over $\Sigma$ is a function: $\mathcal{I}: \Sigma \rightarrow \{T, F\}$.

A formula $\psi$ is **true under** $\mathcal{I}$ or is **satisfied by** $\mathcal{I}$ (symbolically $\mathcal{I} \models \psi$):

- $\mathcal{I} \models \psi$ iff $\mathcal{I}(\psi) = T$
- $\mathcal{I} \models \top$
- $\mathcal{I} \not\models \bot$
- $\mathcal{I} \models \neg \varphi$ iff $\mathcal{I} \not\models \varphi$
- $\mathcal{I} \models \varphi \land \varphi'$ iff $\mathcal{I} \models \varphi$ and $\mathcal{I} \models \varphi'$
- $\mathcal{I} \models \varphi \lor \varphi'$ iff $\mathcal{I} \models \varphi$ or $\mathcal{I} \models \varphi'$
- $\mathcal{I} \models \varphi \rightarrow \varphi'$ iff if $\mathcal{I} \models \varphi$, then $\mathcal{I} \models \varphi'$
- $\mathcal{I} \models \varphi \leftrightarrow \varphi'$ iff $\mathcal{I} \models \varphi$, if and only if $\mathcal{I} \models \varphi'$
Example

\[ I : a \mapsto T, \quad b \mapsto F, \quad c \mapsto F, \quad d \mapsto T \]
Example

\[ \mathcal{I} : a \mapsto T, \ b \mapsto F, \ c \mapsto F, \ d \mapsto T \]

\[
( (a \lor b) \iff (c \lor d) ) \land ( \neg (a \land c) \lor (c \land \neg d) )
\]
Example

\[ I : a \mapsto T, \ b \mapsto F, \ c \mapsto F, \ d \mapsto T \]

\[
\left( (a \lor b) \iff (c \lor d) \right) \land \left( \neg (a \land c) \lor (c \land \neg d) \right)
\]

\[ I \models a \lor b \]
Example

\[ \mathcal{I} : a \mapsto T, \ b \mapsto F, \ c \mapsto F, \ d \mapsto T \]

\[
\left( (a \lor b) \leftrightarrow (c \lor d) \right) \land \left( \neg (a \land c) \lor (c \land \neg d) \right)
\]

\[ \mathcal{I} \models a \lor b \quad \mathcal{I} \models c \lor d \]
Example

\[ \mathcal{I} : a \mapsto T, \; b \mapsto F, \; c \mapsto F, \; d \mapsto T \]

\[(a \lor b) \leftrightarrow (c \lor d) \land (\neg (a \land c) \lor (c \land \neg d)) \]

\[ \mathcal{I} \models a \lor b \quad \mathcal{I} \models c \lor d \quad \mathcal{I} \nmodels a \land c \]
Example

\[ I : a \mapsto T, \ b \mapsto F, \ c \mapsto F, \ d \mapsto T \]

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

\[ I \models a \lor b \quad I \models c \lor d \quad I \not\models a \land c \quad I \not\models c \land \neg d \]
Example

\[ \mathcal{I} : a \mapsto T, \ b \mapsto F, \ c \mapsto F, \ d \mapsto T \]

\[ ( (a \lor b) \leftrightarrow (c \lor d) ) \land ( \neg (a \land c) \lor (c \land \neg d) ) \]

\[ \mathcal{I} \models a \lor b \quad \mathcal{I} \models c \lor d \]
\[ \mathcal{I} \not\models a \land c \quad \mathcal{I} \not\models c \land \neg d \]
\[ \mathcal{I} \models \neg (a \land c) \]
Example

\[ I : a \rightarrow T, \quad b \leftrightarrow F, \quad c \rightarrow F, \quad d \rightarrow T \]

\[
\begin{align*}
(a \lor b) & \iff (c \lor d) \\
I \models a \lor b & \quad I \models c \lor d & \quad I \nmid a \land c & \quad I \nmid c \land \neg d & \quad I \models \neg (a \land c) \\
I \models (a \lor b) & \iff (c \lor d)
\end{align*}
\]
Example

\( \mathcal{I} : a \mapsto T, \ b \mapsto F, \ c \mapsto F, \ d \mapsto T \)

\[
\left( (a \lor b) \leftrightarrow (c \lor d) \right) \land \left( \neg (a \land c) \lor (c \land \neg d) \right)
\]

\( \mathcal{I} \models a \lor b \quad \mathcal{I} \models c \lor d \quad \mathcal{I} \nmid a \land c \quad \mathcal{I} \nmid c \land \neg d \)

\( \mathcal{I} \models \neg(a \land c) \)

\( \mathcal{I} \models (a \lor b) \leftrightarrow (c \lor d) \quad \mathcal{I} \models \neg(a \land c) \lor (c \land \neg d) \)
Example

\[ \mathcal{I} : a \mapsto T, \ b \mapsto F, \ c \mapsto F, \ d \mapsto T \]

\[( (a \lor b) \leftrightarrow (c \lor d) ) \land ( \neg (a \land c) \lor (c \land \neg d) ) \]

\[ \mathcal{I} \models a \lor b \quad \mathcal{I} \models c \lor d \quad \mathcal{I} \not\models a \land c \quad \mathcal{I} \not\models c \land \neg d \]

\[ \mathcal{I} \models \neg(a \land c) \quad \mathcal{I} \models (a \lor b) \leftrightarrow (c \lor d) \quad \mathcal{I} \models \neg(a \land c) \lor (c \land \neg d) \]

\[ \mathcal{I} \models ((a \lor b) \leftrightarrow (c \lor d)) \land (\neg(a \land c) \lor (c \land \neg d)) \]
Terminology

An interpretation $\mathcal{I}$ is called a model of $\varphi$ iff

$$\mathcal{I} \models \varphi$$
Terminology

An interpretation $I$ is called a **model** of $\varphi$ iff

\[ I \models \varphi \]

A formula $\varphi$ is called

- **satisfiable** iff there exists $I$ such that: $I \models \varphi$
- **unsatisfiable** otherwise
Terminology

An interpretation $\mathcal{I}$ is called a **model** of $\varphi$ iff

$$\mathcal{I} \models \varphi$$

A formula $\varphi$ is called

- **satisfiable** iff there exists $\mathcal{I}$ such that: $\mathcal{I} \models \varphi$
- **unsatisfiable** otherwise
- **valid** iff for all $\mathcal{I}$: $\mathcal{I} \models \varphi$
- **falsifiable** otherwise
An interpretation $\mathcal{I}$ is called a **model** of $\varphi$ iff

$$\mathcal{I} \models \varphi$$

A formula $\varphi$ is called

- **satisfiable** iff there exists $\mathcal{I}$ such that: $\mathcal{I} \models \varphi$
- **unsatisfiable** otherwise
- **valid** iff for all $\mathcal{I}$: $\mathcal{I} \models \varphi$
- **falsifiable** otherwise

Two formulae $\varphi$ and $\psi$ are **logically equivalent** (symbolically $\varphi \equiv \psi$) iff for all interpretations $\mathcal{I}$:

$$\mathcal{I} \models \varphi \text{ iff } \mathcal{I} \models \psi$$
Examples

Satisfiable, unsatisfiable, falsifiable, valid?

\[(a \lor b \lor \lnot c) \land (\lnot a \lor \lnot b \lor d) \land (\lnot a \lor b \lor \lnot d)\]
Satisfiable, unsatisfiable, falsifiable, valid?

\[(a \lor b \lor \neg c) \land (\neg a \lor \neg b \lor d) \land (\neg a \lor b \lor \neg d)\]

\(\leadsto\) satisfiable: \(a \mapsto T, b \mapsto F, d \mapsto F, \ldots\)
Examples

Satisfiable, unsatisfiable, falsifiable, valid?

\[(a \lor b \lor \neg c) \land (\neg a \lor \neg b \lor d) \land (\neg a \lor b \lor \neg d)\]

\[\Rightarrow\text{ satisfiable: } a \mapsto T, b \mapsto F, d \mapsto F, \ldots\]

\[\Rightarrow\text{ falsifiable: } a \mapsto F, b \mapsto F, c \mapsto T, \ldots\]
Examples

Satisfiable, unsatisfiable, falsifiable, valid?

\[(a \lor b \lor \neg c) \land (\neg a \lor \neg b \lor d) \land (\neg a \lor b \lor \neg d)\]

\(\leadsto\) satisfiable: \(a \mapsto T, b \mapsto F, d \mapsto F, \ldots\)

\(\leadsto\) falsifiable: \(a \mapsto F, b \mapsto F, c \mapsto T, \ldots\)

\(((\neg a \rightarrow \neg b) \rightarrow (b \rightarrow a))\)
Examples

Satisfiable, unsatisfiable, falsifiable, valid?

\[(a \lor b \lor \neg c) \land (\neg a \lor \neg b \lor d) \land (\neg a \lor b \lor \neg d)\]

\(\leadsto\) satisfiable: \(a \mapsto T, b \mapsto F, d \mapsto F, \ldots\)

\(\leadsto\) falsifiable: \(a \mapsto F, b \mapsto F, c \mapsto T, \ldots\)

\[((\neg a \rightarrow \neg b) \rightarrow (b \rightarrow a))\]

\(\leadsto\) satisfiable: \(a \mapsto T, b \mapsto T\)
Examples

Satisfiable, unsatisfiable, falsifiable, valid?

\[(a \lor b \lor \neg c) \land (\neg a \lor \neg b \lor d) \land (\neg a \lor b \lor \neg d)\]

\[\leadsto\text{ satisfiable: } a \mapsto T, b \mapsto F, d \mapsto F, \ldots\]

\[\leadsto\text{ falsifiable: } a \mapsto F, b \mapsto F, c \mapsto T, \ldots\]

\[((\neg a \rightarrow \neg b) \rightarrow (b \rightarrow a))\]

\[\leadsto\text{ satisfiable: } a \mapsto T, b \mapsto T\]

\[\leadsto\text{ valid: Consider all interpretations or argue about falsifying ones}\]
Examples

Satisfiable, unsatisfiable, falsifiable, valid?

\((a \lor b \lor \neg c) \land (\neg a \lor \neg b \lor d) \land (\neg a \lor b \lor \neg d)\)

\(\Rightarrow\) satisfiable: \(a \mapsto T, b \mapsto F, d \mapsto F, \ldots\)

\(\Rightarrow\) falsifiable: \(a \mapsto F, b \mapsto F, c \mapsto T, \ldots\)

\(((\neg a \rightarrow \neg b) \rightarrow (b \rightarrow a))\)

\(\Rightarrow\) satisfiable: \(a \mapsto T, b \mapsto T\)

\(\Rightarrow\) valid: Consider all interpretations or argue about falsifying ones

Equivalence?

\(\neg (a \lor b) \equiv \neg a \land \neg b\)
Examples

Satisfiable, unsatisfiable, falsifiable, valid?

\((a \lor b \lor \neg c) \land (\neg a \lor \neg b \lor d) \land (\neg a \lor b \lor \neg d)\)

\(\rightsquigarrow\) satisfiable: \(a \mapsto T, b \mapsto F, d \mapsto F, \ldots\)

\(\rightsquigarrow\) falsifiable: \(a \mapsto F, b \mapsto F, c \mapsto T, \ldots\)

\(((\neg a \to \neg b) \to (b \to a))\)

\(\rightsquigarrow\) satisfiable: \(a \mapsto T, b \mapsto T\)

\(\rightsquigarrow\) valid: Consider all interpretations or argue about falsifying ones

Equivalence?

\(\neg(a \lor b) \equiv \neg a \land \neg b\)

\(\rightsquigarrow\) Of course, equivalent (De Morgan).
Some Obvious Consequences

Proposition. \( \varphi \) is valid iff \( \neg \varphi \) is unsatisfiable and \( \varphi \) is satisfiable iff \( \neg \varphi \) is falsifiable.
Some Obvious Consequences

**Proposition.** \( \varphi \) is valid iff \( \neg \varphi \) is unsatisfiable and \( \varphi \) is satisfiable iff \( \neg \varphi \) is falsifiable.

**Proposition.** \( \varphi \equiv \psi \) iff \( \varphi \leftrightarrow \psi \) is valid.
Some Obvious Consequences

**Proposition.** $\varphi$ is valid iff $\neg \varphi$ is unsatisfiable and $\varphi$ is satisfiable iff $\neg \varphi$ is falsifiable.

**Proposition.** $\varphi \equiv \psi$ iff $\varphi \leftrightarrow \psi$ is valid.

**Theorem.** If $\varphi \equiv \psi$ and $\chi'$ results from substituting $\varphi$ by $\psi$ in $\chi$, then $\chi' \equiv \chi$. 
Some Equivalences

Simplifications

\[ \varphi \rightarrow \psi \equiv \neg \varphi \lor \psi \]

\[ \varphi \leftrightarrow \psi \equiv (\varphi \rightarrow \psi) \land (\psi \rightarrow \varphi) \]
Some Equivalences

Simplifications

\[ \varphi \rightarrow \psi \equiv \neg \varphi \lor \psi \quad \varphi \leftrightarrow \psi \equiv (\varphi \rightarrow \psi) \land (\psi \rightarrow \varphi) \]

Idempotency

\[ \varphi \lor \varphi \equiv \varphi \quad \varphi \land \varphi \equiv \varphi \]
### Some Equivalences

<table>
<thead>
<tr>
<th>Simplifications</th>
<th>( \varphi \to \psi \equiv \neg \varphi \lor \psi )</th>
<th>( \varphi \leftrightarrow \psi \equiv (\varphi \to \psi) \land (\psi \to \varphi) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idempotency</td>
<td>( \varphi \lor \varphi \equiv \varphi )</td>
<td>( \varphi \land \varphi \equiv \varphi )</td>
</tr>
<tr>
<td>Commutativity</td>
<td>( \varphi \lor \psi \equiv \psi \lor \varphi )</td>
<td>( \varphi \land \psi \equiv \psi \land \varphi )</td>
</tr>
</tbody>
</table>
## Some Equivalences

<table>
<thead>
<tr>
<th>Category</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simplifications</td>
<td>( \varphi \rightarrow \psi \equiv \neg \varphi \lor \psi )</td>
</tr>
<tr>
<td>Idempotency</td>
<td>( \varphi \lor \varphi \equiv \varphi )</td>
</tr>
<tr>
<td>Commutativity</td>
<td>( \varphi \lor \psi \equiv \psi \lor \varphi )</td>
</tr>
<tr>
<td>Associativity</td>
<td>( (\varphi \lor \psi) \lor \chi \equiv \varphi \lor (\psi \lor \chi) )</td>
</tr>
</tbody>
</table>
### Some Equivalences

<table>
<thead>
<tr>
<th>Category</th>
<th>Formula</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simplifications</td>
<td>$\varphi \rightarrow \psi \equiv \neg \varphi \lor \psi$ $\varphi \leftrightarrow \psi \equiv (\varphi \rightarrow \psi) \land (\psi \rightarrow \varphi)$</td>
</tr>
<tr>
<td>Idempotency</td>
<td>$\varphi \lor \varphi \equiv \varphi$ $\varphi \land \varphi \equiv \varphi$</td>
</tr>
<tr>
<td>Commutativity</td>
<td>$\varphi \lor \psi \equiv \psi \lor \varphi$ $\varphi \land \psi \equiv \psi \land \varphi$</td>
</tr>
<tr>
<td>Associativity</td>
<td>$(\varphi \lor \psi) \lor \chi \equiv \varphi \lor (\psi \lor \chi)$ $(\varphi \land \psi) \land \chi \equiv \varphi \land (\psi \land \chi)$</td>
</tr>
<tr>
<td>Absorption</td>
<td>$\varphi \lor (\varphi \land \psi) \equiv \varphi$ $\varphi \land (\varphi \lor \psi) \equiv \varphi$</td>
</tr>
</tbody>
</table>
# Some Equivalences

<table>
<thead>
<tr>
<th>Property</th>
<th>Expression 1</th>
<th>Expression 2</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Simplifications</strong></td>
<td>$\varphi \rightarrow \psi \equiv \neg \varphi \lor \psi$</td>
<td>$\varphi \leftrightarrow \psi \equiv (\varphi \rightarrow \psi) \land (\psi \rightarrow \varphi)$</td>
</tr>
<tr>
<td><strong>Idempotency</strong></td>
<td>$\varphi \lor \varphi \equiv \varphi$</td>
<td>$\varphi \land \varphi \equiv \varphi$</td>
</tr>
<tr>
<td><strong>Commutativity</strong></td>
<td>$\varphi \lor \psi \equiv \psi \lor \varphi$</td>
<td>$\varphi \land \psi \equiv \psi \land \varphi$</td>
</tr>
<tr>
<td><strong>Associativity</strong></td>
<td>$(\varphi \lor \psi) \lor \chi \equiv \varphi \lor (\psi \lor \chi)$</td>
<td>$(\varphi \land \psi) \land \chi \equiv \varphi \land (\psi \land \chi)$</td>
</tr>
<tr>
<td><strong>Absorption</strong></td>
<td>$\varphi \lor (\varphi \land \psi) \equiv \varphi$</td>
<td>$\varphi \land (\varphi \lor \psi) \equiv \varphi$</td>
</tr>
<tr>
<td><strong>Distributivity</strong></td>
<td>$\varphi \land (\psi \lor \chi) \equiv (\varphi \land \psi) \lor (\varphi \land \chi)$</td>
<td>$\varphi \lor (\psi \land \chi) \equiv (\varphi \lor \psi) \land (\varphi \lor \chi)$</td>
</tr>
</tbody>
</table>
### Some Equivalences

<table>
<thead>
<tr>
<th>Property</th>
<th>Equation</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Simplifications</strong></td>
<td>$\varphi \rightarrow \psi \equiv \neg \varphi \vee \psi$</td>
</tr>
<tr>
<td></td>
<td>$\varphi \leftrightarrow \psi \equiv (\varphi \rightarrow \psi) \land (\psi \rightarrow \varphi)$</td>
</tr>
<tr>
<td><strong>Idempotency</strong></td>
<td>$\varphi \lor \varphi \equiv \varphi$</td>
</tr>
<tr>
<td></td>
<td>$\varphi \land \varphi \equiv \varphi$</td>
</tr>
<tr>
<td><strong>Commutativity</strong></td>
<td>$\varphi \lor \psi \equiv \psi \lor \varphi$</td>
</tr>
<tr>
<td></td>
<td>$\varphi \land \psi \equiv \psi \land \varphi$</td>
</tr>
<tr>
<td><strong>Associativity</strong></td>
<td>$(\varphi \lor \psi) \lor \chi \equiv \varphi \lor (\psi \lor \chi)$</td>
</tr>
<tr>
<td></td>
<td>$(\varphi \land \psi) \land \chi \equiv \varphi \land (\psi \land \chi)$</td>
</tr>
<tr>
<td><strong>Absorption</strong></td>
<td>$\varphi \lor (\varphi \land \psi) \equiv \varphi$</td>
</tr>
<tr>
<td></td>
<td>$\varphi \land (\varphi \lor \psi) \equiv \varphi$</td>
</tr>
<tr>
<td><strong>Distributivity</strong></td>
<td>$\varphi \land (\psi \lor \chi) \equiv (\varphi \land \psi) \lor (\varphi \land \chi)$</td>
</tr>
<tr>
<td></td>
<td>$(\varphi \lor \psi) \land (\psi \lor \chi) \equiv (\varphi \lor \psi) \land (\psi \lor \chi)$</td>
</tr>
<tr>
<td><strong>Double Negation</strong></td>
<td>$\neg \neg \varphi \equiv \varphi$</td>
</tr>
</tbody>
</table>
### Some Equivalences

<table>
<thead>
<tr>
<th>Type</th>
<th>Equivalence</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Simplifications</strong></td>
<td>$\varphi \rightarrow \psi \equiv \neg \varphi \lor \psi$</td>
</tr>
<tr>
<td></td>
<td>$\varphi \leftrightarrow \psi \equiv (\varphi \rightarrow \psi) \land (\psi \rightarrow \varphi)$</td>
</tr>
<tr>
<td><strong>Idempotency</strong></td>
<td>$\varphi \lor \varphi \equiv \varphi$</td>
</tr>
<tr>
<td></td>
<td>$\varphi \land \varphi \equiv \varphi$</td>
</tr>
<tr>
<td><strong>Commutativity</strong></td>
<td>$\varphi \lor \psi \equiv \psi \lor \varphi$</td>
</tr>
<tr>
<td></td>
<td>$\varphi \land \psi \equiv \psi \land \varphi$</td>
</tr>
<tr>
<td><strong>Associativity</strong></td>
<td>$(\varphi \lor \psi) \lor \chi \equiv \varphi \lor (\psi \lor \chi)$</td>
</tr>
<tr>
<td></td>
<td>$(\varphi \land \psi) \land \chi \equiv \varphi \land (\psi \land \chi)$</td>
</tr>
<tr>
<td><strong>Absorption</strong></td>
<td>$\varphi \lor (\varphi \land \psi) \equiv \varphi$</td>
</tr>
<tr>
<td></td>
<td>$\varphi \land (\varphi \lor \psi) \equiv \varphi$</td>
</tr>
<tr>
<td><strong>Distributivity</strong></td>
<td>$\varphi \land (\psi \lor \chi) \equiv (\varphi \land \psi) \lor (\varphi \land \chi)$</td>
</tr>
<tr>
<td></td>
<td>$\varphi \lor (\psi \land \chi) \equiv (\varphi \lor \psi) \land (\varphi \lor \chi)$</td>
</tr>
<tr>
<td><strong>Double Negation</strong></td>
<td>$\neg \neg \varphi \equiv \varphi$</td>
</tr>
<tr>
<td><strong>Constant Negation</strong></td>
<td>$\neg \top \equiv \bot$</td>
</tr>
<tr>
<td></td>
<td>$\neg \bot \equiv \top$</td>
</tr>
</tbody>
</table>
# Some Equivalences

<table>
<thead>
<tr>
<th>Simplifications</th>
<th>( \varphi \rightarrow \psi \equiv \neg \varphi \lor \psi )</th>
<th>( \varphi \leftrightarrow \psi \equiv (\varphi \rightarrow \psi) \land (\psi \rightarrow \varphi) )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Idempotency</td>
<td>( \varphi \lor \varphi \equiv \varphi )</td>
<td>( \varphi \land \varphi \equiv \varphi )</td>
</tr>
<tr>
<td>Commutativity</td>
<td>( \varphi \lor \psi \equiv \psi \lor \varphi )</td>
<td>( \varphi \land \psi \equiv \psi \land \varphi )</td>
</tr>
<tr>
<td>Associativity</td>
<td>( (\varphi \lor \psi) \lor \chi \equiv \varphi \lor (\psi \lor \chi) )</td>
<td>( (\varphi \land \psi) \land \chi \equiv \varphi \land (\psi \land \chi) )</td>
</tr>
<tr>
<td>Absorption</td>
<td>( \varphi \lor (\varphi \land \psi) \equiv \varphi )</td>
<td>( \varphi \land (\varphi \lor \psi) \equiv \varphi )</td>
</tr>
<tr>
<td>Distributivity</td>
<td>( \varphi \land (\psi \lor \chi) \equiv (\varphi \land \psi) \lor (\varphi \land \chi) )</td>
<td>( \varphi \lor (\psi \land \chi) \equiv (\varphi \lor \psi) \land (\varphi \lor \chi) )</td>
</tr>
<tr>
<td>Double Negation</td>
<td>( \neg \neg \varphi \equiv \varphi )</td>
<td></td>
</tr>
<tr>
<td>Constant Negation</td>
<td>( \neg \top \equiv \bot )</td>
<td>( \neg \bot \equiv \top )</td>
</tr>
<tr>
<td>De Morgan</td>
<td>( \neg (\varphi \lor \psi) \equiv \neg \varphi \land \neg \psi )</td>
<td>( \neg (\varphi \land \psi) \equiv \neg \varphi \lor \neg \psi )</td>
</tr>
<tr>
<td>Equivalence</td>
<td>Expression</td>
<td></td>
</tr>
<tr>
<td>--------------------------------</td>
<td>-------------------------------------------------</td>
<td></td>
</tr>
<tr>
<td><strong>Simplifications</strong></td>
<td>( \varphi \rightarrow \psi \equiv \neg \varphi \lor \psi )   ( \varphi \leftrightarrow \psi \equiv (\varphi \rightarrow \psi) \land (\psi \rightarrow \varphi) )</td>
<td></td>
</tr>
<tr>
<td><strong>Idempotency</strong></td>
<td>( \varphi \lor \varphi \equiv \varphi )   ( \varphi \land \varphi \equiv \varphi )</td>
<td></td>
</tr>
<tr>
<td><strong>Commutativity</strong></td>
<td>( \varphi \lor \psi \equiv \psi \lor \varphi )   ( \varphi \land \psi \equiv \psi \land \varphi )</td>
<td></td>
</tr>
<tr>
<td><strong>Associativity</strong></td>
<td>( (\varphi \lor \psi) \lor \chi \equiv \varphi \lor (\psi \lor \chi) )   ( (\varphi \land \psi) \land \chi \equiv \varphi \land (\psi \land \chi) )</td>
<td></td>
</tr>
<tr>
<td><strong>Absorption</strong></td>
<td>( \varphi \lor (\varphi \land \psi) \equiv \varphi )   ( \varphi \land (\varphi \lor \psi) \equiv \varphi )</td>
<td></td>
</tr>
<tr>
<td><strong>Distributivity</strong></td>
<td>( \varphi \land (\psi \lor \chi) \equiv (\varphi \land \psi) \lor (\varphi \land \chi) )   ( \varphi \lor (\psi \land \chi) \equiv (\varphi \lor \psi) \land (\varphi \lor \chi) )</td>
<td></td>
</tr>
<tr>
<td><strong>Double Negation</strong></td>
<td>( \neg \neg \varphi \equiv \varphi )</td>
<td></td>
</tr>
<tr>
<td><strong>Constant Negation</strong></td>
<td>( \neg \top \equiv \bot )   ( \neg \bot \equiv \top )</td>
<td></td>
</tr>
<tr>
<td><strong>De Morgan</strong></td>
<td>( \neg (\varphi \lor \psi) \equiv \neg \varphi \land \neg \psi )   ( \neg (\varphi \land \psi) \equiv \neg \varphi \lor \neg \psi )</td>
<td></td>
</tr>
<tr>
<td><strong>Truth</strong></td>
<td>( \varphi \lor \top \equiv \top )   ( \varphi \land \top \equiv \varphi )</td>
<td></td>
</tr>
</tbody>
</table>
# Some Equivalences

<table>
<thead>
<tr>
<th>Category</th>
<th>Equivalence</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simplifications</td>
<td>( \varphi \rightarrow \psi \equiv \neg \varphi \lor \psi )</td>
</tr>
<tr>
<td>Idempotency</td>
<td>( \varphi \lor \varphi \equiv \varphi )</td>
</tr>
<tr>
<td>Commutativity</td>
<td>( \varphi \lor \psi \equiv \psi \lor \varphi )</td>
</tr>
<tr>
<td>Associativity</td>
<td>( (\varphi \lor \psi) \lor \chi \equiv \varphi \lor (\psi \lor \chi) )</td>
</tr>
<tr>
<td>Absorption</td>
<td>( \varphi \lor (\varphi \land \psi) \equiv \varphi )</td>
</tr>
<tr>
<td>Distributivity</td>
<td>( \varphi \land (\psi \lor \chi) \equiv (\varphi \land \psi) \lor (\varphi \land \chi) )</td>
</tr>
<tr>
<td>Double Negation</td>
<td>( \neg \neg \varphi \equiv \varphi )</td>
</tr>
<tr>
<td>Constant Negation</td>
<td>( \neg \top \equiv \bot )</td>
</tr>
<tr>
<td>De Morgan</td>
<td>( \neg (\varphi \lor \psi) \equiv \neg \varphi \land \neg \psi )</td>
</tr>
<tr>
<td>Truth</td>
<td>( \varphi \lor \top \equiv \top )</td>
</tr>
<tr>
<td>Falsity</td>
<td>( \varphi \lor \bot \equiv \varphi )</td>
</tr>
</tbody>
</table>
How Many Different Formulae Are There . . .

. . . for a given finite alphabet $\Sigma_n$?
How Many Different Formulae Are There . . .

. . . for a given finite alphabet \( \Sigma_n \) ?

- Infinitely many: \( a, a \lor a, a \land a, a \lor a \lor a, \ldots \)
How Many Different Formulae Are There . . .

. . . for a given finite alphabet $\Sigma_n$?

- Infinitely many: $a, a \lor a, a \land a, a \lor a \lor a, \ldots$

- How many different logically distinguishable (non-equivalent) formulae?
How Many Different Formulae Are There . . .

. . . for a given finite alphabet $\Sigma_n$?

- Infinitely many: $a, a \lor a, a \land a, a \lor a \lor a, \ldots$

- How many different logically distinguishable (non-equivalent) formulae?
  - For $\Sigma_n$, there exist $2^n$ different interpretations.
  - A formula can be characterized by its set of models
    (if two formulae are logically non-equivalent, then their sets of models differ)
  - There are $2^{(2^n)}$ different sets of interpretations
How Many Different Formulae Are There . . .

. . . for a given finite alphabet $\Sigma_n$?

- Infinitely many: $a, a \lor a, a \land a, a \lor a \lor a, \ldots$

- How many different logically distinguishable (non-equivalent) formulae?
  - For $\Sigma_n$, there exist $2^n$ different interpretations.
  - A formula can be characterized by its set of models
    (if two formulae are logically non-equivalent, then their sets of models differ)
  - There are $2^{(2^n)}$ different sets of interpretations
  - $\leadsto$ There are $2^{(2^n)}$ logical equivalence classes of formulae
Logical Implication

- Extension of the satisfiability relation “|-” to sets $\Theta$ of formulae:

$$ \mathcal{I} \models \Theta \text{ iff } \mathcal{I} \models \varphi \text{ for all } \varphi \in \Theta. $$
Logical Implication

• Extension of the satisfiability relation “$\models$” to sets $\Theta$ of formulae:

$$\mathcal{I} \models \Theta \text{ iff } \mathcal{I} \models \varphi \text{ for all } \varphi \in \Theta.$$ 

• Reminder: $\varphi$ should be considered as logically implied by $\Theta$ (symbolically $\Theta \models \varphi$) iff $\varphi$ is true in all models of $\Theta$:

$$\Theta \models \varphi \text{ iff } \mathcal{I} \models \varphi \text{ for all } \mathcal{I} \text{ such that } \mathcal{I} \models \Theta.$$
Logical Implication

• Extension of the satisfiability relation “|=” to sets $\Theta$ of formulae:

$$\mathcal{I} \models \Theta \text{ iff } \mathcal{I} \models \varphi \text{ for all } \varphi \in \Theta.$$  

• Reminder: $\varphi$ should be considered as logically implied by $\Theta$ (symbolically $\Theta \models \varphi$) iff $\varphi$ is true in all models of $\Theta$:

$$\Theta \models \varphi \text{ iff } \mathcal{I} \models \varphi \text{ for all } \mathcal{I} \text{ such that } \mathcal{I} \models \Theta.$$  

• Some consequences:
  
  ◦ Deduction theorem: $\Theta \cup \{\varphi\} \models \psi$ iff $\Theta \models \varphi \rightarrow \psi$
Logical Implication

• Extension of the satisfiability relation “|=” to sets $\Theta$ of formulae:

$$\mathcal{I} |= \Theta \text{ iff } \mathcal{I} |= \varphi \text{ for all } \varphi \in \Theta.$$  

• Reminder: $\varphi$ should be considered as **logically implied** by $\Theta$ (symbolically $\Theta |= \varphi$) iff $\varphi$ is true in all models of $\Theta$:

$$\Theta |= \varphi \text{ iff } \mathcal{I} |= \varphi \text{ for all } \mathcal{I} \text{ such that } \mathcal{I} |= \Theta$$  

• Some consequences:
  
  o **Deduction theorem**: $\Theta \cup \{\varphi\} |= \psi$ iff $\Theta |= \varphi \rightarrow \psi$
  
  o **Contraposition**: $\Theta \cup \{\varphi\} |= \neg \psi$ iff $\Theta \cup \{\psi\} |= \neg \varphi$
Logical Implication

• Extension of the satisfiability relation “|=” to sets \( \Theta \) of formulae:

\[ \mathcal{I} |\!\!\!\!|= \Theta \text{ iff } \mathcal{I} |\!\!\!\!|= \varphi \text{ for all } \varphi \in \Theta. \]

• Reminder: \( \varphi \) should be considered as logically implied by \( \Theta \) (symbolically \( \Theta |\!\!\!\!|= \varphi \)) iff \( \varphi \) is true in all models of \( \Theta \):

\[ \Theta |\!\!\!\!|= \varphi \text{ iff } \mathcal{I} |\!\!\!\!|= \varphi \text{ for all } \mathcal{I} \text{ such that } \mathcal{I} |\!\!\!\!|= \Theta \]

• Some consequences:
  - Deduction theorem: \( \Theta \cup \{\varphi\} |\!\!\!\!|= \psi \text{ iff } \Theta |\!\!\!\!|= \varphi \rightarrow \psi \)
  - Contraposition: \( \Theta \cup \{\varphi\} |\!\!\!\!|= \neg \psi \text{ iff } \Theta \cup \{\psi\} |\!\!\!\!|= \neg \varphi \)
  - Contradiction: \( \Theta \cup \{\varphi\} \) is unsatisfiable iff \( \Theta |\!\!\!\!|= \neg \varphi \)
Normal Forms

Terminology:

- Possibly negated atomic formulae ($\neg a$), truth ($\top$), and falsity ($\bot$) are called literals.

- A disjunction of literals is called clause.
Normal Forms

**Terminology:**

- Possibly negated atomic formulae ($\neg a$), truth ($\top$), and falsity ($\bot$) are called **literals**
- A disjunction of literals is called **clause**
- If the negated sub-formulae of a formula are all literals, the formula is called **negation normal form (NNF)** formula

**Example:**

$((\neg a \lor \neg b) \land c)$, **but not:** $((\neg a \land b) \land c)$
Normal Forms

**Terminology:**

- Possibly negated atomic formulae ($[\neg]a$), truth ($\top$), and falsity ($\bot$) are called **literals**
- A disjunction of literals is called **clause**
- If the negated sub-formulae of a formula are all literals, the formula is called **negation normal form (NNF) formula**
  
  **Example:** $((\neg a \lor \neg b) \land c)$, but not: $(\neg(a \land b) \land c)$

- A conjunction of clauses is called **conjunctive normal form (CNF) formula**
  
  **Example:** $(a \lor b) \land (\neg a \lor c)$
Normal Forms

Terminology:

- Possibly negated atomic formulae ($\neg a$), truth ($\top$), and falsity ($\bot$) are called **literals**
- A disjunction of literals is called **clause**
- If the negated sub-formulae of a formula are all literals, the formula is called **negation normal form (NNF) formula**
  **Example**: $((\neg a \lor \neg b) \land c)$, **but not**: $(\neg(a \land b) \land c)$
- A conjunction of clauses is called **conjunctive normal form (CNF) formula**
  **Example**: $(a \lor b) \land (\neg a \lor c)$
- The dual form (disjunction of conjunctions of literals) is called **disjunctive normal form (DNF)**
  **Example**: $(a \land b) \lor (\neg a \land c)$
**Negation Normal Form**

**Theorem.** For each propositional formula there exists a logically equivalent formula in NNF.
Negation Normal Form

**Theorem.** For each propositional formula there exists a logically equivalent formula in NNF.

- True for $a$, $\neg a$, $\top$, $\bot$. 
Negation Normal Form

**Theorem.** For each propositional formula there exists a logically equivalent formula in NNF.

- True for $a$, $\neg a$, $\top$, $\bot$.

- Let us assume it is true for all formulae $\varphi$ (up to a certain number of connectives) and call its NNF $\text{nnf}(\varphi)$.
Theorem. For each propositional formula there exists a logically equivalent formula in NNF.

- True for $a, \neg a, \top, \bot$.

- Let us assume it is true for all formulae $\varphi$ (up to a certain number of connectives) and call its NNF $\text{nnf}(\varphi)$
  - $\text{nnf}(\varphi \land \psi) = \text{nnf}(\varphi) \land \text{nnf}(\psi)$
Negation Normal Form

**Theorem.** For each propositional formula there exists a logically equivalent formula in NNF.

- True for $a$, $\neg a$, $\top$, $\bot$.

- Let us assume it is true for all formulae $\varphi$ (up to a certain number of connectives) and call its NNF $\text{nnf}(\varphi)$
  
  - $\text{nnf}(\varphi \land \psi) = \text{nnf}(\varphi) \land \text{nnf}(\psi)$
  - $\text{nnf}(\varphi \lor \psi) = \text{nnf}(\varphi) \lor \text{nnf}(\psi)$
Negation Normal Form

**Theorem.** For each propositional formula there exists a logically equivalent formula in NNF.

- True for $a, \neg a, \top, \bot$.

- Let us assume it is true for all formulae $\varphi$ (up to a certain number of connectives) and call its NNF $\text{nnf}(\varphi)$
  - $\text{nnf}(\varphi \land \psi) = \text{nnf}(\varphi) \land \text{nnf}(\psi)$
  - $\text{nnf}(\varphi \lor \psi) = \text{nnf}(\varphi) \lor \text{nnf}(\psi)$
  - $\text{nnf}(\neg(\varphi \land \psi)) = \text{nnf}(\neg \varphi) \lor \text{nnf}(\neg \psi)$
**Negation Normal Form**

**Theorem.** For each propositional formula there exists a logically equivalent formula in NNF.

- True for $a, \neg a, \top, \bot$.

- Let us assume it is true for all formulae $\varphi$ (up to a certain number of connectives) and call its NNF $\text{nnf}(\varphi)$
  
  - $\text{nnf}(\varphi \land \psi) = \text{nnf}(\varphi) \land \text{nnf}(\psi)$
  - $\text{nnf}(\varphi \lor \psi) = \text{nnf}(\varphi) \lor \text{nnf}(\psi)$
  - $\text{nnf}(\neg(\varphi \land \psi)) = \text{nnf}(\neg \varphi) \lor \text{nnf}(\neg \psi)$
  - $\text{nnf}(\neg(\varphi \lor \psi)) = \text{nnf}(\neg \varphi) \land \text{nnf}(\neg \psi)$
Theorem. For each propositional formula there exists a logically equivalent formula in CNF. Use a similar argument (works also for DNF)
Theorem. For each propositional formula there exists a logically equivalent formula in CNF. Use a similar argument (works also for DNF)

- True for $a, \neg a, \top, \bot$. 
Theorem. For each propositional formula there exists a logically equivalent formula in CNF. Use a similar argument (works also for DNF)

- True for $a$, $\neg a$, $\top$, $\bot$.
- Let us assume it is true for all formulae $\varphi$ (up to a certain number of connectives) and call its CNF $\text{cnf}(\varphi)$
**Theorem.** For each propositional formula there exists a logically equivalent formula in CNF. Use a similar argument (works also for DNF)

- True for $a, \neg a, \top, \bot$.

- Let us assume it is true for all formulae $\varphi$ (up to a certain number of connectives) and call its CNF $\text{cnf}(\varphi)$
  - $\text{cnf}(\neg \varphi) = \text{cnf}(\text{nnf}(\neg \varphi))$
**Theorem.** For each propositional formula there exists a logically equivalent formula in CNF. Use a similar argument (works also for DNF)

- True for \( a, \neg a, \top, \bot \).

- Let us assume it is true for all formulae \( \varphi \) (up to a certain number of connectives) and call its CNF \( \text{cnf}(\varphi) \)
  
  - \( \text{cnf}(\neg \varphi) = \text{cnf}(\text{nnf}(\neg \varphi)) \)
  
  - \( \text{cnf}(\varphi \land \psi) = \text{cnf}(\varphi) \land \text{cnf}(\psi) \)
Theorem. For each propositional formula there exists a logically equivalent formula in CNF. Use a similar argument (works also for DNF)

- True for $a$, $\neg a$, $\top$, $\bot$.

- Let us assume it is true for all formulae $\varphi$ (up to a certain number of connectives) and call its CNF $\text{cnf}(\varphi)$
  - $\text{cnf}(\neg \varphi) = \text{cnf}(\text{nff}(\neg \varphi))$
  - $\text{cnf}(\varphi \land \psi) = \text{cnf}(\varphi) \land \text{cnf}(\psi)$
  - Assume $\text{cnf}(\varphi) = \bigwedge_i \chi_i$ and $\text{cnf}(\psi) = \bigwedge_j \rho_j$ with $\chi_i, \rho_j$ being clauses.
**Theorem.** For each propositional formula there exists a logically equivalent formula in CNF. Use a similar argument (works also for DNF)

- True for $a$, $\neg a$, $\top$, $\bot$.
- Let us assume it is true for all formulae $\varphi$ (up to a certain number of connectives) and call its CNF $\text{cnf}(\varphi)$
  - $\text{cnf}(\neg \varphi) = \text{cnf}(\text{nnf}(\neg \varphi))$
  - $\text{cnf}(\varphi \land \psi) = \text{cnf}(\varphi) \land \text{cnf}(\psi)$
  - Assume $\text{cnf}(\varphi) = \bigwedge_i \chi_i$ and $\text{cnf}(\psi) = \bigwedge_j \rho_j$ with $\chi_i$, $\rho_j$ being clauses.
  - Then, using distributivity:

  $\text{cnf}(\varphi \lor \psi) = \text{cnf}(\bigwedge_i \chi_i \lor \bigwedge_j \rho_j)$
Theorem. For each propositional formula there exists a logically equivalent formula in CNF. Use a similar argument (works also for DNF)

- True for $a$, $\neg a$, $\top$, $\bot$.
- Let us assume it is true for all formulae $\varphi$ (up to a certain number of connectives) and call its CNF $\text{cnf}(\varphi)$
  - $\text{cnf}(\neg \varphi) = \text{cnf}(\text{nnf}(\neg \varphi))$
  - $\text{cnf}(\varphi \land \psi) = \text{cnf}(\varphi) \land \text{cnf}(\psi)$
  - Assume $\text{cnf}(\varphi) = \bigwedge_i \chi_i$ and $\text{cnf}(\psi) = \bigwedge_j \rho_j$ with $\chi_i, \rho_j$ being clauses.
  - Then, using distributivity:
    
    $$
    \text{cnf}(\varphi \lor \psi) = \text{cnf}(\bigwedge_i \chi_i \lor \bigwedge_j \rho_j)
    $$
    
    $$
    = \bigwedge_i \bigwedge_j (\chi_i \lor \rho_j)
    $$
How do we decide whether a formula is satisfiable, unsatisfiable, valid, or falsifiable?

Note: Satisfiability and falsifiability are NP-complete. Validity and unsatisfiability are co-NP-complete.
How to Decide Properties of Formulae

How do we decide whether a formula is satisfiable, unsatisfiable, valid, or falsifiable?

Note: Satisfiability and falsifiability are NP-complete. Validity and unsatisfiability are co-NP-complete.

- A CNF formula is valid iff all clauses contain complementary literals or $\top$. 
How to Decide Properties of Formulae

How do we decide whether a formula is satisfiable, unsatisfiable, valid, or falsifiable?

Note: Satisfiability and falsifiability are **NP-complete**. Validity and unsatisfiability are **co-NP-complete**.

- A CNF formula is valid iff all clauses contain complementary literals or $\top$.

- A DNF formula is satisfiable iff one disjunct does not contain $\bot$ or complementary literals.
How do we decide whether a formula is satisfiable, unsatisfiable, valid, or falsifiable?

Note: Satisfiability and falsifiability are NP-complete. Validity and unsatisfiability are co-NP-complete.

- A CNF formula is valid iff all clauses contain complementary literals or $\top$.
- A DNF formula is satisfiable iff one disjunct does not contain $\bot$ or complementary literals.
- However, transformation to CNF or DNF may take exponential time (and space!).
How to Decide Properties of Formulae

How do we decide whether a formula is satisfiable, unsatisfiable, valid, or falsifiable?

Note: Satisfiability and falsifiability are **NP-complete**. Validity and unsatisfiability are **co-NP-complete**.

- A CNF formula is valid iff all clauses contain complementary literals or $\top$.
- A DNF formula is satisfiable iff one disjunct does not contain $\bot$ or complementary literals.
- However, transformation to CNF or DNF may take exponential time (and space!).
- One can try out all truth assignments.
How do we decide whether a formula is satisfiable, unsatisfiable, valid, or falsifiable?

Note: Satisfiability and falsifiability are \textbf{NP-complete}. Validity and unsatisfiability are \textbf{co-NP-complete}.

- A CNF formula is valid iff all clauses contain complementary literals or \( \top \).
- A DNF formula is satisfiable iff one disjunct does not contain \( \bot \) or complementary literals.
- However, transformation to CNF or DNF may take exponential time (and space!).
- One can try out all truth assignments.
- One can test systematically for satisfying truth assignments (backtracking search) \( \rightsquigarrow \) \textbf{Davis-Putnam procedure (DP)}
Deciding Entailment

- We want to decide $\Theta \models \varphi$. 
Deciding Entailment

- We want to decide $\Theta \models \varphi$.

$\rightarrow$ Use deduction theorem and reduce to validity:

$$\Theta \models \varphi \iff \bigwedge \Theta \rightarrow \varphi \text{ is valid.}$$

$\sim\sim$ Now negate and test for unsatisfiability using DP.
Deciding Entailment

- We want to decide $\Theta \models \varphi$.

\[ \rightarrow \text{Use deduction theorem and reduce to validity:} \]

\[ \Theta \models \varphi \iff \bigwedge \Theta \rightarrow \varphi \text{ is valid.} \]

\[ \leadsto \text{Now negate and test for unsatisfiability using DP.} \]

- Different approach: Try to derive $\varphi$ from $\Theta$ – find a proof of $\varphi$ from $\Theta$
Deciding Entailment

• We want to decide $\Theta \models \varphi$.

→ Use deduction theorem and reduce to validity:

$$\Theta \models \varphi \iff \bigwedge \Theta \rightarrow \varphi \text{ is valid.}$$

⇝ Now negate and test for unsatisfiability using DP.

• Different approach: Try to derive $\varphi$ from $\Theta$ – find a proof of $\varphi$ from $\Theta$

→ Use inference rules to deduce new formulae from $\Theta$. Continue to deduce new formulae until $\varphi$ can be deduced.
Deciding Entailment

• We want to decide $\Theta \models \varphi$.

→ Use deduction theorem and reduce to validity:

$$\Theta \models \varphi \iff \bigwedge \Theta \to \varphi \text{ is valid.}$$

⇝ Now negate and test for unsatisfiability using DP.

• Different approach: Try to derive $\varphi$ from $\Theta$ – find a proof of $\varphi$ from $\Theta$

→ Use inference rules to deduce new formulae from $\Theta$. Continue to deduce new formulae until $\varphi$ can be deduced.

• One particular calculus: resolution
Resolution: Representation

- We assume that all formulae are in CNF
  - Can be generated using the described method
  - Often formulae are already close to CNF
  - There is a “cheap” conversion from arbitrary formulae to CNF that preserves satisfiability – which is enough as we will see.
Resolution: Representation

- We assume that all formulae are in CNF
  - Can be generated using the described method
  - Often formulae are already close to CNF
  - There is a “cheap” conversion from arbitrary formulae to CNF that preserves satisfiability – which is enough as we will see.

- More convenient representation
  - CNF formula represented as set
  - each clause is a set of literals

$$\rightarrow (a \lor \neg b) \land (\neg a \lor c) \iff \{\{a, \neg b\}, \{\neg a, c\}\}$$
Resolution: Representation

• We assume that all formulae are in CNF
  ○ Can be generated using the described method
  ○ Often formulae are already close to CNF
  ○ There is a “cheap” conversion from arbitrary formulae to CNF that preserves satisfiability – which is enough as we will see.

• More convenient representation
  ○ CNF formula represented as set
  ○ each clause is a set of literals
    \[ (a \lor \neg b) \land (\neg a \lor c) \Rightarrow \{\{a, \neg b\}, \{\neg a, c\}\} \]

• Empty clause (symbolically □) and empty set of clauses (symbolically ∅) are different!
Resolution: The Inference Rule

Let \( l \) be a literal and \( \overline{l} \) the negated literal.

The rule:

\[
\frac{C_1 \cup \{l\}, C_2 \cup \{\overline{l}\} \quad C_1 \cup C_2}{C_1 \cup C_2}
\]
Resolution: The Inference Rule

Let $l$ be a literal and $\overline{l}$ the negated literal.

The rule:

\[
\frac{C_1 \cup \{l\}, C_2 \cup \{\overline{l}\}}{C_1 \cup C_2}
\]

$C_1 \cup C_2$ is the resolvent of the parent clauses $C_1 \cup \{l\}$ and $C_2 \cup \{\overline{l}\}$. $l$ and $\overline{l}$ are the resolution literals.
Resolution: The Inference Rule

Let $l$ be a literal and $\overline{l}$ the negated literal.

The rule:

$$\frac{C_1 \cup \{l\}, C_2 \cup \{\overline{l}\}}{C_1 \cup C_2}$$

$C_1 \cup C_2$ is the resolvent of the parent clauses $C_1 \cup \{l\}$ and $C_2 \cup \{\overline{l}\}$. $l$ and $\overline{l}$ are the resolution literals.

Example: $\{a, b, \neg c\}$ resolves with $\{a, d, c\}$ to $\{a, b, d\}$. 
Resolution: The Inference Rule

Let $l$ be a literal and $\overline{l}$ the negated literal.

The rule:

$$\frac{C_1 \cup \{l\}, C_2 \cup \{\overline{l}\}}{C_1 \cup C_2}$$

$C_1 \cup C_2$ is the resolvent of the parent clauses $C_1 \cup \{l\}$ and $C_2 \cup \{\overline{l}\}$. $l$ and $\overline{l}$ are the resolution literals.

Example: $\{a, b, \neg c\}$ resolves with $\{a, d, c\}$ to $\{a, b, d\}$.

Note: The resolvent is not logically equivalent with the set of parent clauses!
Resolution: The Inference Rule

Let \( l \) be a literal and \( \bar{l} \) the negated literal.

The rule:

\[
\begin{align*}
C_1 \cup \{l\}, C_2 \cup \{\bar{l}\} \\
\overline{C_1 \cup C_2}
\end{align*}
\]

\( C_1 \cup C_2 \) is the resolvent of the parent clauses \( C_1 \cup \{l\} \) and \( C_2 \cup \{\bar{l}\} \). \( l \) and \( \bar{l} \) are the resolution literals.

Example: \( \{a, b, \neg c\} \) resolves with \( \{a, d, c\} \) to \( \{a, b, d\} \).

Note: The resolvent is not logically equivalent with the set of parent clauses!

Notation:

\[
R(\Delta) = \Delta \cup \{C | C \text{ is resolvent of two clauses of } \Delta\}
\]
Resolution: Derivations

We say that $D$ can be derived from $\Delta$ using resolution (symbolically $\Delta \vdash D$) if

- there exists a sequence of clauses $C_1, \ldots, C_n$ such that $C_n = D$ and

- $C_i \in R(\Delta \cup \{C_1, \ldots, C_{i-1}\}$, for $i \in \{1, \ldots, n\}$. 


Resolution: Derivations

We say that $D$ can be derived from $\Delta$ using resolution (symbolically $\Delta \vdash D$) if

- there exists a sequence of clauses $C_1, \ldots, C_n$ such that $C_n = D$ and
- $C_i \in R(\Delta \cup \{C_1, \ldots, C_{i-1}\}$, for $i \in \{1, \ldots, n\}$.

$\iff R^*(\Delta) = \{D | \Delta \vdash D\}$
Resolution: Derivations

We say that $D$ can be derived from $\Delta$ using resolution (symbolically $\Delta \vdash D$) if

- there exists a sequence of clauses $C_1, \ldots, C_n$ such that $C_n = D$ and
- $C_i \in R(\Delta \cup \{C_1, \ldots, C_{i-1}\}$, for $i \in \{1, \ldots, n\}$.

$\leadsto R^*(\Delta) = \{D \mid \Delta \vdash D\}$

Theorem (Soundness of resolution). If $\Delta \vdash D$, then $\Delta \models D$. 
Resolution: Derivations

We say that $D$ can be derived from $\Delta$ using resolution (symbolically $\Delta \vdash D$) if

- there exists a sequence of clauses $C_1, \ldots, C_n$ such that $C_n = D$ and
- $C_i \in R(\Delta \cup \{C_1, \ldots, C_{i-1}\}$, for $i \in \{1, \ldots, n\}$.

$\Leftrightarrow R^*(\Delta) = \{D | \Delta \vdash D\}$

**Theorem** (Soundness of resolution). If $\Delta \vdash D$, then $\Delta \models D$.

**Proof idea**: Show $\Delta \models D$ if $D \in R(\Delta)$ and use induction on proof length.
We say that \( D \) can be derived from \( \Delta \) using resolution (symbolically \( \Delta \vdash D \)) if

- there exists a sequence of clauses \( C_1, \ldots, C_n \) such that \( C_n = D \) and
- \( C_i \in R(\Delta \cup \{C_1, \ldots, C_{i-1}\} \), for \( i \in \{1, \ldots, n\} \).

\[ \Rightarrow R^*(\Delta) = \{D|\Delta \vdash D\} \]

**Theorem** (Soundness of resolution). If \( \Delta \vdash D \), then \( \Delta \models D \).

**Proof idea**: Show \( \Delta \models D \) if \( D \in R(\Delta) \) and use induction on proof length.

Let \( C_1 \cup \{l\} \) and \( C_2 \cup \{\bar{l}\} \) be the parent clauses of \( D = C_1 \cup C_2 \).
Resolution: Derivations

We say that $D$ can be derived from $\Delta$ using resolution (symbolically $\Delta \vdash D$) if

- there exists a sequence of clauses $C_1, \ldots, C_n$ such that $C_n = D$ and
- $C_i \in R(\Delta \cup \{C_1, \ldots, C_{i-1}\}$, for $i \in \{1, \ldots, n\}$.

$\Rightarrow \quad R^*(\Delta) = \{D | \Delta \vdash D\}$

Theorem (Soundness of resolution). If $\Delta \vdash D$, then $\Delta \models D$.

Proof idea: Show $\Delta \models D$ if $D \in R(\Delta)$ and use induction on proof length.

Let $C_1 \cup \{l\}$ and $C_2 \cup \{\bar{l}\}$ be the parent clauses of $D = C_1 \cup C_2$.

Assume $\mathcal{I} \models \Delta$, we have to show $\mathcal{I} \models D$. 
Resolution: Derivations

We say that $D$ can be derived from $\Delta$ using resolution (symbolically $\Delta \vdash D$) if

- there exists a sequence of clauses $C_1, \ldots, C_n$ such that $C_n = D$ and
- $C_i \in R(\Delta \cup \{C_1, \ldots, C_{i-1}\}$, for $i \in \{1, \ldots, n\}.$

$\Rightarrow R^*(\Delta) = \{D | \Delta \vdash D\}$

**Theorem** (Soundness of resolution). If $\Delta \vdash D$, then $\Delta \models D$.

**Proof idea**: Show $\Delta \models D$ if $D \in R(\Delta)$ and use induction on proof length.

Let $C_1 \cup \{l\} \text{ and } C_2 \cup \{\bar{l}\}$ be the parent clauses of $D = C_1 \cup C_2$.

Assume $\mathcal{I} \models \Delta$, we have to show $\mathcal{I} \models D$.

Case 1: $\mathcal{I} \models l$, then there must be a literal $m \in C_2 \text{ s.t. } \mathcal{I} \models m.$
Resolution: Derivations

We say that \( D \) can be derived from \( \Delta \) using resolution (symbolically \( \Delta \vdash D \)) if

- there exists a sequence of clauses \( C_1, \ldots, C_n \) such that \( C_n = D \) and
- \( C_i \in R(\Delta \cup \{ C_1, \ldots, C_{i-1} \}) \), for \( i \in \{1, \ldots, n\} \).

\[ R^*(\Delta) = \{ D | \Delta \vdash D \} \]

**Theorem** (Soundness of resolution). If \( \Delta \vdash D \), then \( \Delta \models D \).

**Proof idea**: Show \( \Delta \models D \) if \( D \in R(\Delta) \) and use induction on proof length.

Let \( C_1 \cup \{ l \} \) and \( C_2 \cup \{ \bar{l} \} \) be the parent clauses of \( D = C_1 \cup C_2 \).

Assume \( \mathcal{I} \models \Delta \), we have to show \( \mathcal{I} \models D \).

Case 1: \( \mathcal{I} \models l \), then there must be a literal \( m \in C_2 \) s.t. \( \mathcal{I} \models m \). This implies \( \mathcal{I} \models D \).
Resolution: Derivations

We say that $D$ can be derived from $\Delta$ using resolution (symbolically $\Delta \vdash D$) if

- there exists a sequence of clauses $C_1, \ldots, C_n$ such that $C_n = D$ and
- $C_i \in R(\Delta \cup \{C_1, \ldots, C_{i-1}\}$, for $i \in \{1, \ldots, n\}$.

$\Rightarrow R^*(\Delta) = \{D | \Delta \vdash D\}$

Theorem (Soundness of resolution). If $\Delta \vdash D$, then $\Delta \models D$.

Proof idea: Show $\Delta \models D$ if $D \in R(\Delta)$ and use induction on proof length.

Let $C_1 \cup \{l\}$ and $C_2 \cup \{\bar{l}\}$ be the parent clauses of $D = C_1 \cup C_2$.

Assume $\mathcal{I} \models \Delta$, we have to show $\mathcal{I} \models D$.

Case 1: $\mathcal{I} \models l$, then there must be a literal $m \in C_2$ s.t. $\mathcal{I} \models m$. This implies $\mathcal{I} \models D$.

Case 2: $\mathcal{I} \models \bar{l}$ similarly, there is $m \in C_1$ s.t. $\mathcal{I} \models m$. 
Resolution: Derivations

We say that $D$ can be derived from $\Delta$ using resolution (symbolically $\Delta \vdash D$) if

- there exists a sequence of clauses $C_1, \ldots, C_n$ such that $C_n = D$ and
- $C_i \in R(\Delta \cup \{C_1, \ldots, C_{i-1}\}$, for $i \in \{1, \ldots, n\}$.

$\Rightarrow R^*(\Delta) = \{D | \Delta \vdash D\}$

**Theorem** (Soundness of resolution). If $\Delta \vdash D$, then $\Delta \models D$.

**Proof idea**: Show $\Delta \models D$ if $D \in R(\Delta)$ and use induction on proof length.

Let $C_1 \cup \{l\}$ and $C_2 \cup \{\bar{l}\}$ be the parent clauses of $D = C_1 \cup C_2$.

Assume $\mathcal{I} \models \Delta$, we have to show $\mathcal{I} \models D$.

Case 1: $\mathcal{I} \models l$, then there must be a literal $m \in C_2$ s.t. $\mathcal{I} \models m$. This implies $\mathcal{I} \models D$.

Case 2: $\mathcal{I} \models \bar{l}$ similarly, there is $m \in C_1$ s.t. $\mathcal{I} \models m$.

This means that each model $\mathcal{I}$ of $\Delta$ also satisfies $D$, i.e., $\Delta \models D$. 24
Resolution: Completeness?

Do we have

\[ \Delta \models \varphi \text{ implies } \Delta \vdash \varphi? \]
Resolution: Completeness?

Do we have

$$\Delta \models \varphi \text{ implies } \Delta \vdash \varphi?$$

Of course, could only hold for CNF.
Resolution: Completeness?

Do we have

\[ \Delta \models \varphi \implies \Delta \vdash \varphi? \]

Of course, could only hold for CNF. However:

\[ \left\{ \{a,b\}, \{\neg b, c\} \right\} \models \{a, b, c\} \]

\[ \forall \{a, b, c\} \]
Resolution: Completeness?

Do we have

\[ \Delta \models \varphi \text{ implies } \Delta \vdash \varphi ? \]

Of course, could only hold for CNF. However:

\[
\begin{align*}
\{ \{a, b\}, \{\neg b, c\} \} & \models \{a, b, c\} \\
\forall \{a, b, c\} & \not\models \{a, b, c\}
\end{align*}
\]

However, one can show that resolution is **refutation complete**:

\[ \Delta \text{ is unsatisfiable } \iff \Delta \vdash \Box. \]
Do we have

$$\Delta \models \varphi \text{ implies } \Delta \vdash \varphi?$$

Of course, could only hold for CNF. However:

$$\left\{ \{a, b\}, \{\neg b, c\} \right\} \models \{a, b, c\}$$

$$\nexists \{a, b, c\}$$

However, one can show that resolution is refutation complete:

$$\Delta \text{ is unsatisfiable } \iff \Delta \vdash \Box.$$ 

**Entailment**: Reduce to unsatisfiability testing and decide by resolution.
Resolution Strategies

- Trying out all different resolutions can be very costly,
- and might not be necessary
Resolution Strategies

- Trying out all different resolutions can be very costly,
- and might not be necessary
- Often people devise so-called resolution strategies
- Examples:
  - Input resolution: In each resolution step, one of the parent clauses must be a clause of the input set: $R_I$. 
Resolution Strategies

• Trying out all different resolutions can be very costly,
• and might not be necessary
• Often people devise so-called resolution strategies
• Examples:
  o **Input resolution**: In each resolution step, one of the parent clauses must be a clause of the input set: $R_I$.
  o **Unit resolution**: In each resolution step, one of the parent clauses must be a unit clause $R_U$. 

Resolution Strategies

- Trying out all different resolutions can be very costly,
- and might not be necessary
- Often people devise so-called resolution strategies
- Examples:
  - **Input resolution**: In each resolution step, one of the parent clauses must be a clause of the input set: $R_I$.
  - **Unit resolution**: In each resolution step, one of the parent clauses must be a unit clause $R_U$
  → Not all strategies are (refutation) completeness preserving
Horn Clauses & Resolution

- **Horn clauses**: Clauses with at most one positive literal
  
  **Example**: \((a \lor \neg b \lor \neg c), (\neg b \lor \neg c)\)
Horn Clauses & Resolution

- **Horn clauses**: Clauses with at most one positive literal
  
  **Example**: \((a \lor \neg b \lor \neg c), (\neg b \lor \neg c)\)

- **Claim**: Unit resolution is refutation complete for Horn clauses
Horn Clauses & Resolution

- **Horn clauses**: Clauses with at most one positive literal
  
  **Example**: \((a \lor \neg b \lor \neg c), (\neg b \lor \neg c)\)

- **Claim**: Unit resolution is refutation complete for Horn clauses

- **Proof idea**: Consider \(R^*_U(\Delta)\) of Horn clause set \(\Delta\). We have to show that if \(\Box \not\in R^*_U(\Delta)\), then \(\Delta(\equiv R^*_U(\Delta))\) is satisfiable.
Horn Clauses & Resolution

- **Horn clauses**: Clauses with at most one positive literal
  
  **Example**: \((a \lor \neg b \lor \neg c), (\neg b \lor \neg c)\)

- **Claim**: Unit resolution is refutation complete for Horn clauses

- **Proof idea**: Consider \(R^*_U(\Delta)\) of Horn clause set \(\Delta\). We have to show that if \(\square \notin R^*_U(\Delta)\), then \(\Delta(\equiv R^*_U(\Delta))\) is satisfiable.
  
  - Assign to all unit clauses in \(R^*_U(\Delta)\) true (and simplify).
Horn Clauses & Resolution

- **Horn clauses**: Clauses with at most one positive literal
  
  **Example**: \((a \lor \neg b \lor \neg c), (\neg b \lor \neg c)\)

- **Claim**: Unit resolution is refutation complete for Horn clauses

- **Proof idea**: Consider \(R^*_U(\Delta)\) of Horn clause set \(\Delta\). We have to show that if \(\Box \not\in R^*_U(\Delta)\), then \(\Delta(\equiv R^*_U(\Delta))\) is satisfiable.
  
  - Assign to all unit clauses in \(R^*_U(\Delta)\) true (and simplify).
  
  - The remaining non-tautological clauses have at least one negative literal.
Horn Clauses & Resolution

- **Horn clauses**: Clauses with at most one positive literal
  
  **Example**: \((a \lor \neg b \lor \neg c), (\neg b \lor \neg c)\)

- **Claim**: Unit resolution is refutation complete for Horn clauses

- **Proof idea**: Consider \(R^*_U(\Delta)\) of Horn clause set \(\Delta\). We have to show that if \(\Box \notin R^*_U(\Delta)\), then \(\Delta(\equiv R^*_U(\Delta))\) is satisfiable.
  
  - Assign to all unit clauses in \(R^*_U(\Delta)\) true (and simplify).
  - The remaining non-tautological clauses have at least one negative literal.
  - Assign true to these literals.
Horn Clauses & Resolution

- **Horn clauses**: Clauses with at most one positive literal
  
  **Example**: \((a \lor \neg b \lor \neg c), (\neg b \lor \neg c)\)

- **Claim**: Unit resolution is refutation complete for Horn clauses

- **Proof idea**: Consider \(R^*_U(\Delta)\) of Horn clause set \(\Delta\). We have to show that if \(\Box \not\in R^*_U(\Delta)\), then \(\Delta(\equiv R^*_U(\Delta))\) is satisfiable.
  
  - Assign to all unit clauses in \(R^*_U(\Delta)\) true (and simplify).
  - The remaining non-tautological clauses have at least one negative literal.
  - Assign true to these literals.

  \(\Rightarrow\) results in satisfying truth assignment for \(R^*_U(\Delta)\) (and hence \(\Delta\)),

27