Propositions

a proposition is a statement that is either true or false

atomic propositions: no further internal structure

example:
- Alice comes to the party.
- It rains.

composite propositions: build from other propositions with Boolean connectives

example:
- Alice comes to the party, Bob as well, but not Cecile.
- If it rains, the street is wet.
Propositional Logic

- **two truth values** (Boolean domain): true/false, verum/falsum, on/off, 1/0

- **language elements**
  - atomic propositions (atoms, variables)
    - no internal structure
    - either true or false
  - logic connectives: not (\(\neg\)), and (\(\land\)), or (\(\lor\)), ...
    - operators for construction of composite propositions
    - concise meaning
    - argument(s) and return value from Boolean domain
  - parenthesis

**example:** formula of propositional logic: \((\neg t \lor s) \land (t \lor s) \land (\neg t \lor \neg s)\)

atoms: \(t, s\), connectives: \(\neg, \lor, \land\), parenthesis for structuring the expression
Background

- historical origins: ancient Greeks
- in philosophy, mathematics, and computer science
- two very basic principles:
  - Law of Excluded Middle:
    a proposition is true or its negation is true
  - Law of Contradiction:
    no expression is both true and false at the same time
- very simple language
  - no objects, no arguments to propositions
  - no functions, no quantifiers
- solving is easy (relative to other logics)
- many applications in industry
Syntax: Structure of Propositional Formulas in Conjunctive Normal Form (CNF)

we build a propositional formula using the following components:

- **literals:**
  - variables (atomic propositions, atoms): $x, y, z, \ldots$
  - negated variables $\neg x, \neg y, \neg z, \ldots$
  - truth constants: $\top$ (verum) and $\bot$ (falsum)
  - negated truth constants: $\neg \top$ and $\neg \bot$

- **clauses:** disjunction ($\lor$) of literals
  - $x \lor y$ (binary clause)
  - $x \lor y \lor \neg z$ (ternary clause)
  - $z$ (unary clause)
  - $\neg \top$ (unary clause)
  - for $(l_1 \lor \ldots \lor l_n)$ we also write $\bigvee_{i=1}^{n} l_i$. 
Syntax: Structure of Propositional Formulas in Conjunctive Normal Form (CNF)

A propositional formula is a conjunction ($\land$) of clauses.

examples of formulas:

- $\top$
- $\bot$
- $x$
- $\neg y$
- $x \land y \land z$
- $(\neg x \lor y \lor \neg z) \land z$
- $(x \lor \neg y) \land (x \lor \neg y \lor z) \land (y \lor \neg z)$
- $(l_{11} \lor \ldots \lor l_{1m_1}) \land \ldots \land (l_{n1} \lor \ldots \lor l_{nm_n})$

for $(C_1 \land \ldots \land C_n)$ we also write $\land_{i=1}^{n} C_i$.

Remark: For the moment, we consider formulas of a restricted structure called CNF, e.g., we do not consider formulas like $(x \land y) \lor (\neg x \land z)$. Any propositional formula can be translated into this structure. We will relax this restriction later.
Conventions

we use the following conventions unless stated otherwise:

- $a, b, c, x, y, z$ denote variables and $l, k$ denote literals
- $\phi, \psi, \gamma$ denote arbitrary formulas
- $C, D$ denote clauses
- clauses are also written as sets
  - $(l_1 \lor \ldots \lor l_n) = \{l_1, \ldots, l_n\}$
  - to add a literal $l$ to clause $C$, we write $C \cup \{l\}$
  - to remove a literal $l$ from clause $C$, we write $C\{l\}$
- formulas in CNF are also written as sets of sets
  - $((l_1 \lor \ldots \lor l_{m_1}) \land \ldots \land (l_{n_1} \lor \ldots \lor l_{m_n})) = \{(l_1, \ldots, l_{m_1}), \ldots, (l_{n_1}, \ldots, l_{m_n})\}$
  - to add a clause $C$ to CNF $\phi$, we write $\phi \cup \{C\}$
  - to remove a clause $C$ from CNF $\phi$, we write $\phi\{C\}$
Negation Operator

- unary connective $\neg$ (operator with exactly one operand)
- alternative notation: $!x$, $\bar{x}$, $-x$, $NOT\ x$
- semantics: flipping the truth value of its operand

<table>
<thead>
<tr>
<th>$x$</th>
<th>$\neg x$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

example:
- If the atom “It rains.” is true then the negation “It does not rain.” is false.
- If the propositional variable $a$ is true then $\neg a$ is false.
- If the propositional variable $a$ is false then $\neg a$ is true.
Binary Disjunction Operator

- binary operator $\lor$ (operator with exactly two operands)
- alternative notation for $l \lor k$: $l \| k, l + k, l \text{ OR } k$
- semantics: true iff at least one operand is true

<table>
<thead>
<tr>
<th>$l$</th>
<th>$k$</th>
<th>$l \lor k$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

truth table:

example:
- $(a \lor \neg a)$ is always true.
- $(\top \lor a)$ is always true.
- $(\bot \lor a)$ is true if $a$ is true.
Properties of Disjunction

- commutative:
  \[ k \lor l \Leftrightarrow l \lor k \]

- idempotent:
  \[ l \lor l \Leftrightarrow l \]

- associative:
  \[ l_1 \lor (l_2 \lor l_3) \Leftrightarrow (l_1 \lor l_2) \lor l_3 \]
Clause: Semantics

- a clause is true iff at least one of the literals is true
- the empty clause is always false

<table>
<thead>
<tr>
<th></th>
<th>( l_1 )</th>
<th>( \ldots )</th>
<th>( l_n )</th>
<th>( l_1 \lor l_2 \lor \ldots \lor l_n )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>\ldots</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>\ldots</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td>\ldots</td>
<td></td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>\ldots</td>
<td>0</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>\ldots</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
</tbody>
</table>
Binary Conjunction Operator

- binary operator $\land$ (operator with exactly two operands)
- alternative notation for $C \land D$: $C \&\& D$, $CD, C \ast D, C \cdot D, C \text{ AND } D$
- semantics: a conjunction is true iff both operands are true

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$C$</td>
<td>$D$</td>
<td>$C \land D$</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

truth table:

example:

- $(a \land \neg a)$ is always false.
- $(\top \land a)$ is true if $a$ is true. $(\bot \land \phi)$ is always false.
- If $(a \lor b)$ is true and $(\neg c \lor d)$ is true then $(a \lor b) \land (\neg c \lor d)$ is true.
Properties of Conjunction

- commutative:
  \[ C \land D \Leftrightarrow D \land C \]

- idempotent:
  \[ C \land C \Leftrightarrow C \]

- associative:
  \[ C_1 \land (C_2 \land C_3) \Leftrightarrow (C_1 \land C_2) \land C_3 \]
CNF Formulas: Semantics

- A formula in CNF is true iff all of its clauses are true.
  - The empty CNF formula is always true.

Truth table:

<table>
<thead>
<tr>
<th>$C_1$</th>
<th>$\ldots$</th>
<th>$C_n$</th>
<th>$C_1 \land C_2 \land \ldots \land C_n$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$\ldots$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>$\ldots$</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>$\ldots$</td>
<td></td>
<td>$\ldots$</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>$\ldots$</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>$\ldots$</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Rules of Precedence

- \neg \text{ binds stronger than } \land
- \land \text{ binds stronger than } \lor

example

- \neg a \lor b \land \neg c \lor d
  - is the same as \((\neg a) \lor (b \land (\neg c)) \lor d\),
  - but not as \(((\neg a) \lor b) \land ((\neg c) \lor d)\)

⇒ put clauses into parentheses!
Assignment

■ a variable can be assigned one of two values from the two-valued domain $\mathbb{B}$, where $\mathbb{B} = \{1, 0\}$

■ the mapping $\nu : \mathcal{P} \rightarrow \mathbb{B}$ is called assignment, where $\mathcal{P}$ is the set of variables of a formula

■ we sometimes write an assignment $\nu$ as set $V$ with $V \subseteq \mathcal{P} \cup \{\neg x|x \in \mathcal{P}\}$ such that
  - $x \in V$ iff $\nu(x) = 1$
  - $\neg x \in V$ iff $\nu(x) = 0$

■ for $n$ variables, there are $2^n$ assignments possible

■ an assignment corresponds to one line in the truth table
Assignment: Example

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>z</th>
<th>x ∨ y</th>
<th>¬z</th>
<th>(x ∨ y) ∧ ¬z</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- one assignment: \( \nu(x) = 1, \nu(y) = 0, \nu(z) = 1 \)
- alternative notation: \( V = \{x, \neg y, z\} \)
- observation: A variable assignment determines the truth value of the formulas containing these variables.
Semantics of Propositional Logic

Let $\mathcal{P}$ be the set of atomic propositions (variables) and $\mathcal{L}$ be the set of all propositional formulas over $\mathcal{P}$ that are syntactically correct (i.e., all possible conjunctions of clauses over $\mathcal{P}$).

Given assignment $\nu : \mathcal{P} \rightarrow \mathbb{B}$, the interpretation $[.]_{\nu} : \mathcal{L} \rightarrow \mathbb{B}$ is defined by:

- $[\top]_{\nu} = 1$, $[\bot]_{\nu} = 0$
- if $x \in \mathcal{P}$ then $[x]_{\nu} = \nu(x)$
- $[\neg x]_{\nu} = 1$ iff $[x]_{\nu} = 0$
- $[C]_{\nu} = 1$ (where $C$ is a clause) iff there is at least one literal $l$ with $l \in C$ and $[l]_{\nu} = 1$
- $[\phi]_{\nu} = 1$ (where $\phi$ is in CNF) iff for all clauses $C \in \phi$ it holds that $[C]_{\nu} = 1$
Satisfying/Falsifying Assignments

- an assignment $\nu$ is called
  - satisfying a formula $\phi$ iff $[\phi]_\nu = 1$
  - falsifying a formula $\phi$ iff $[\phi]_\nu = 0$

- a satisfying assignment for $\phi$ is a model of $\phi$
- a falsifying assignment for $\phi$ is a counter-model of $\phi$

**example:**

For formula $((x \lor y) \land \neg z)$,
- $\{x, y, z\}$ is a counter-model,
- $\{x, y, \neg z\}$ is a model.
SAT-Solver Limboole

- available at http://fmv.jku.at/limboole
- input:
  - variables are strings over letters, digits and \(-\) \_ \.[ ] $ @
  - negation symbol \(\neg\) is !
  - disjunction symbol \(\lor\) is |
  - conjunction symbol \(\land\) is &

**example**

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

is represented as

\[(a \mid b \mid !c) \& (!a \mid b) \& c\]

---

^1For now, we will only use subset of the language supported by Limboole.
Properties of Propositional Formulas (1/2)

- formula $\phi$ is **satisfiable** iff there exists an assignment $\nu$ with $[\phi]_{\nu} = 1$
  
  check with limboole -s

- formula $\phi$ is **valid** iff for all assignments $\nu$ it holds that $[\phi]_{\nu} = 1$
  
  check with limboole

- formula $\phi$ is **refutable** iff there exists an assignment $\nu$ with $[\phi]_{\nu} = 0$
  
  check with limboole

- formula $\phi$ is **unsatisfiable** iff for all assignments $\nu$ it holds that $[\phi]_{\nu} = 0$
  
  check with limboole -s
Properties of Propositional Formulas (2/2)

- a valid formula is called **tautology**
- an unsatisfiable formula is called **contradiction**

**example:**

- $\top$ is valid.
- $a \lor \neg a$ is a tautology.
- $(a \lor \neg b) \land (\neg a \lor b)$ is refutable.
- $\bot$ is unsatisfiable.
- $a \land \neg a$ is a contradiction.
- $(a \lor \neg b) \land (\neg a \lor b)$ is satisfiable.
Given a propositional formula $\phi$. Is there an assignment that satisfies $\phi$?

different formulation: can we find an assignment such that each clause contains at least one true literal?
Application: Graph Coloring

A graph is something like a network consisting of

- vertices (nodes)
- edges (connections between nodes)

Example:

- set of vertices $V = \{a, b, c\}$
- set of edges (pairs of vertices from $V$) $E = \{(a, b), (b, c)\}$

![Graph diagram]
Application: Graph Coloring

A graph is something like a network consisting of

- vertices (nodes)
- edges (connections between nodes)

Example:

- set of vertices \( V = \{a, b, c\} \)
- set of edges (pairs of vertices from \( V\) ) \( E = \{(a, b), (b, c)\} \)

Graph Coloring: Assign colors to vertices such that connected vertices have different colors.
Encoding the k-Coloring Problem

Given graph \((V, E)\) with vertices \(V\) and edges \(E\). Color each node with one of \(k\) colors, such that there is no edge \((v, w) \in E\), with vertices \(v\) and \(w\) colored in the same color.

encoding:

1. **propositional variables**: \(v_j\) ... node \(v \in V\) has color \(j\) (\(1 \leq j \leq k\))

2. each node has a color:

\[
\bigwedge_{v \in V} \left( \bigvee_{1 \leq j \leq k} v_j \right)
\]

3. each node has just one color: \((\neg v_i \lor \neg v_j)\) with \(v \in V\), \(1 \leq i < j \leq k\)

4. neighbors have different colors: \((\neg v_i \lor \neg w_i)\) with \((v, w) \in E\), \(1 \leq i \leq k\)
Encoding the k-Coloring Problem: Example

task: find 2-coloring of graph \(\{a, b, c\}, \{(a, b), (b, c)\}\) with SAT
possible solution:

encoding in SAT:
Encoding the k-Coloring Problem: Example

Task: find 2-coloring of graph \( \{a, b, c\}, \{(a, b), (b, c)\} \) with SAT

Possible solution:

Encoding in SAT:

- Variables: \( a_1, a_2, b_1, b_2, c_1, c_2 \)
Encoding the k-Coloring Problem: Example

task: find 2-coloring of graph \( \{a, b, c\}, \{(a, b), (b, c)\} \) with SAT

possible solution:

encoding in SAT:

- variables: \( a_1, a_2, b_1, b_2, c_1, c_2 \)
- clauses:
  1. each node has a color: \( (a_1 \lor a_2), (b_1 \lor b_2), (c_1 \lor c_2) \)
  2. no node has two colors: \( (\neg a_1 \lor \neg a_2), (\neg b_1 \lor \neg b_2), (\neg c_1 \lor \neg c_2) \)
  3. connected nodes have a different color:
     \( (\neg a_1 \lor \neg b_1), (\neg a_2 \lor \neg b_2), (\neg b_1 \lor \neg c_1), (\neg b_2 \lor \neg c_2) \)
**Encoding the k-Coloring Problem: Example**

**task:** find 2-coloring of graph \( \{a, b, c\}, \{(a, b), (b, c)\}\) with SAT

**possible solution:**

- **encoding in SAT:**
  - **variables:** \(a_1, a_2, b_1, b_2, c_1, c_2\)
  - **clauses:**
    1. each node has a color: \((a_1 \lor a_2), (b_1 \lor b_2), (c_1 \lor c_2)\)
    2. no node has two colors: \((\neg a_1 \lor \neg a_2), (\neg b_1 \lor \neg b_2), (\neg c_1 \lor \neg c_2)\)
    3. connected nodes have a different color:
       - \((\neg a_1 \lor \neg b_1), (\neg a_2 \lor \neg b_2), (\neg b_1 \lor \neg c_1), (\neg b_2 \lor \neg c_2)\)
  - **full formula:**
    \[
    (a_1 \lor a_2) \land (b_1 \lor b_2) \land (c_1 \lor c_2) \land (\neg a_1 \lor \neg a_2) \land (\neg b_1 \lor \neg b_2) \land (\neg c_1 \lor \neg c_2) \land \\
    (\neg a_1 \lor \neg b_1) \land (\neg a_2 \lor \neg b_2) \land (\neg b_1 \lor \neg c_1) \land (\neg b_2 \lor \neg c_2)
    \]