Propositions

A proposition is an atomic statement that is either *true* or *false*.

**Example:**
- Alice comes to the party.
- It rains.

With connectives, propositions can be combined.

**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 (~), and (\&), 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: ~, \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
The set $\mathcal{L}$ of well-formed propositional formulas is the smallest set such that

1. $\top, \bot \in \mathcal{L}$;
2. $\mathcal{P} \subseteq \mathcal{L}$ where $\mathcal{P}$ is the set of atomic propositions (atoms, variables);
3. if $\phi \in \mathcal{L}$ then $(\neg \phi) \in \mathcal{L}$;
4. if $\phi, \psi \in \mathcal{L}$ then $(\phi \circ \psi) \in \mathcal{L}$ with $\circ \in \{\lor, \land, \leftrightarrow, \rightarrow\}$.

$\mathcal{L}$ is the language of propositional logic. The elements of $\mathcal{L}$ are propositional formulas.
Syntax of Propositional Logic (2/2)

In Backus-Naur form (BNF) propositional formulas are described as follows:

\[ \phi ::= \top | \bot | p | (\neg \phi) | (\phi \lor \phi) | (\phi \land \phi) | (\phi \leftrightarrow \phi) | (\phi \rightarrow \phi) \]

Example:

- \( \top \)
- \( (\neg a) \)
- \( (\neg (\neg a)) \)
- \( (\neg (a \lor b)) \)
- \( a \)
- \( (\neg \bot) \)
- \( (a_1 \lor a_2) \)
- \( (\neg (a \leftrightarrow b)) \)
- \( (((\neg a) \lor a') \leftrightarrow (b \rightarrow c)) \)
- \( (((a_1 \lor a_2) \lor (a_3 \land \bot)) \rightarrow b) \)
Rules of Precedence

To reduce the number of parenthesis, we use the following conventions (in case of doubt, uses parenthesis!):

- ¬ is stronger than ∧
- ∧ is stronger than ∨
- ∨ is stronger than →
- → is stronger than ⇔
- Binary operators of same strength are assumed to be left parenthesized (also called “left associative”)

Example:

- ¬a ∧ b ∨ c → d ⇔ f is the same as (((((¬a) ∧ b) ∨ c) → d) ⇔ f).
- a' ∨ a'' ∨ a'' ∨ b' ∨ b'' is the same as (((a' ∨ a'') ∨ (a'' ∧ b')) ∨ b'').
- a' ∧ a'' ∧ a'' ∨ b' ∧ b'' is the same as (((a' ∧ a'') ∧ a''') ∨ (b' ∧ b''')).
Formula Tree

- formulas have a tree structure
  - inner nodes: connectives
  - leaves: truth constants, variables

- default: inner nodes have one child node (negation) or two nodes as children (other connectives).

- tree structure reflects the use of parenthesis

- simplification:
  disjunction and conjunction may be considered as $n$-ary operators,
i.e., if a node $N$ and its child node $C$ are of the same kind of connective (conjunction / disjunction), then the children of $C$ can become direct children of $N$ and the $C$ is removed.
The formula

\[(a \lor (b \lor \neg c)) \leftrightarrow (\top \land ((a \rightarrow \neg b) \lor (\bot \lor a \lor b)))\]

defines the following formula tree:
The formula

\[(a \lor (b \lor \neg c)) \leftrightarrow (\top \land ((a \to \neg b) \lor (\bot \lor a \lor b)))\]

has the simplified formula tree
Subformulas

An immediate subformula is defined as follows:

- truth constants and atoms have no immediate subformula.
- only immediate subformula of $\neg \phi$ is $\phi$.
- formula $\phi \circ \psi$ ($\circ \in \{\land, \lor, \leftrightarrow, \rightarrow\}$) has immediate subformulas $\phi$ and $\psi$.

Informal: a subformula is a formula that is part of a formula

The set of subformulas of a formula $\phi$ is the smallest set $S$ with

1. $\phi \in S$
2. if $\psi \in S$ then all immediate subformulas of $\psi$ are in $S$

The subformulas of $(a \lor b) \rightarrow (c \land \neg \neg d)$ are

$$\{a, b, c, d, \neg d, \neg \neg d, a \lor b, c \land \neg \neg d, (a \lor b) \rightarrow (c \land \neg \neg d)\}$$
Limboole

- SAT-solver
- available at http://fmv.jku.at/limboole/
- input format in BNF:

\[
\langle expr \rangle ::= \langle iff \rangle
\]
\[
\langle iff \rangle ::= \langle implies \rangle \mid \langle implies \rangle \text{"<->"} \langle implies \rangle
\]
\[
\langle implies \rangle ::= \langle or \rangle \mid \langle or \rangle \text{"->"} \langle or \rangle \mid \langle or \rangle \text{"<-"} \langle or \rangle
\]
\[
\langle or \rangle ::= \langle and \rangle \mid \langle and \rangle \text{"|"} \langle and \rangle
\]
\[
\langle and \rangle ::= \langle not \rangle \mid \langle not \rangle \text{"&"} \langle not \rangle
\]
\[
\langle not \rangle ::= \langle basic \rangle \mid \text{"!"} \langle not \rangle
\]
\[
\langle basic \rangle ::= \langle var \rangle \mid \text{"("} \langle expr \rangle \text{"\)"}
\]

where 'var' is a string over letters, digits, and – _ . [ ] $ @

In Limboole the formula \((a \lor b) \rightarrow (c \land \neg\neg d)\) is represented as

\[\((\langle a \mid b \rangle \rightarrow \langle c \land \neg\neg d \rangle)\)\]
Special Formula Structures

- **literal:** variable or a negated variable (also (negated) truth constants)
  - examples of literals: \( x, \neg x, y, \neg y \)
  - If \( l \) is a literal with \( l = x \) or \( l = \neg x \) then \( \text{var}(l) = x \).
  - For literals we use letter \( l, k \) (possibly indexed or primed).
  - In principle, we identify \( \neg \neg l \) with \( l \).

- **clause:** disjunction of literals
  - unary clause (clause of size one): \( l \) where \( l \) is a literal
  - empty clause (clause of size zero): \( \bot \)
  - examples of clauses: \( (x \lor y), (\neg x \lor x' \lor \neg x''), x, \neg y \)

- **cube:** conjunction of literals
  - unary cube (cubes of size one): \( l \) where \( l \) is a literal
  - empty cubes (cubes of size zero): \( \top \)
  - examples of cubes: \( (x \land y), (\neg x \land x' \land \neg x''), x, \neg y \)
Negation Normal Form (1/2)

Negation Normal Form (NNF) is defined as follows:

- Literals and truth constants are in NNF;
- \( \phi \circ \psi \ (\circ \in \{\lor, \land\}) \) is in NNF iff \( \phi \) and \( \psi \) are in NNF;
- no other formulas are in NNF.

In other words: A formula in NNF contains only conjunctions, disjunctions, and negations and negations only occur in front of variables and constants.
Negation Normal Form (2/2)

If a formula is in negation normal form then

■ in the formula tree, nodes with negation symbols only occur directly before leaves.

■ there are no subformulas of the form $\neg \phi$ where $\phi$ is something else than a variable or a constant.

■ it does not contain NAND, NOR, XOR, equivalence, and implication connectives.

**Example:** The formula $((x \lor \neg x_1) \land (x \lor (\neg z \lor \neg x_1)))$ is in NNF but $\neg((x \lor \neg x_1) \land (x \lor (\neg z \lor \neg x_1)))$ is not in NNF.
Conjunctive Normal Form (CNF)

A propositional formula is in *conjunctive normal form* (CNF) iff it is a conjunction of clauses.

Remark: CNF is the input of most SAT-solvers (DIMACS format)
A propositional formula is in *disjunctive normal form (DNF)* if it is a disjunction of cubes.

A formula in disjunctive normal form is

- in negation normal form
- $\bot$ if it contains no clauses
- easy to check whether it can be satisfied
Examples for CNF and DNF

Examples CNF

- $\top$
- $\bot$
- $a$
- $\neg a$
- $l_1 \land l_2 \land l_3$
- $l_1 \lor l_2 \lor l_3$
- $(a_1 \lor \neg a_2) \land (a_1 \lor b_2 \lor a_2) \land a_2$
- $((l_{11} \lor \ldots \lor l_{1m_1}) \land \ldots \land (l_{n1} \lor \ldots \lor l_{nm_n}))$

Examples DNF

- $\top$
- $\bot$
- $a$
- $\neg a$
- $l_1 \land l_2 \land l_3$
- $l_1 \lor l_2 \lor l_3$
- $(a_1 \land \neg a_2) \lor (a_1 \land b_2 \land a_2) \lor a_2$
- $((l_{11} \land \ldots \land l_{1m_1}) \lor \ldots \lor (l_{n1} \land \ldots \land l_{nm_n}))$
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 or cubes (clear from context)
- 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 $C \setminus \{l\}$
- formulas in CNF are also written as sets of sets
  - $((l_{11} \lor \ldots \lor l_{1m_1}) \land \ldots \land (l_{n1} \lor \ldots \lor l_{nm_n})) = \{\{l_{11}, \ldots l_{1m_1}\}, \ldots, \{l_{n1}, \ldots l_{nm_n}\}\}$
  - to add a clause $C$ to CNF $\phi$, we write $\phi \cup \{C\}$
  - to remove a clause $C$ from CNF $\phi$, we write $\phi \setminus \{C\}$
Negation

- unary connective ¬ (operator with exactly one argument)
- negating the truth value of its argument
- alternative notation: !φ, ̅φ, −φ, NOTφ

<table>
<thead>
<tr>
<th>truth table:</th>
<th>φ</th>
<th>¬φ</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<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 atom a is true then ¬a is false.
- If formula ((a ∨ x) ∧ y) is true then formula ¬((a ∨ x) ∧ y) is false.
- If formula ((b → y) ∧ z) is true then formula ¬((b → y) ∧ z) is false.
Conjunction

- A conjunction is true iff both arguments are true.
- Alternative notation for $\phi \land \psi$: $\&$, $\psi$, $\phi \ast \psi$, $\phi \cdot \psi$, $\phi \text{AND} \psi$
- For $(\phi_1 \land \ldots \land \phi_n)$ we also write $\land_{i=1}^{n} \phi_i$.

Truth table:

<table>
<thead>
<tr>
<th>$\phi$</th>
<th>$\psi$</th>
<th>$\phi \land \psi$</th>
</tr>
</thead>
<tbody>
<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>

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.
Disjunction

- a disjunction is true iff at least one of the arguments is true
- alternative notation for $\phi \lor \psi$: $\phi | \psi$, $\phi + \psi$, $\phi OR \psi$
- For $(\phi_1 \lor \ldots \lor \phi_n)$ we also write $\bigvee_{i=1}^{n} \phi_i$.

<table>
<thead>
<tr>
<th>$\phi$</th>
<th>$\psi$</th>
<th>$\phi \lor \psi$</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>

Example:

- $(a \lor \neg a)$ is always true.
- $(\top \lor a)$ is always true. $(\bot \lor a)$ is true if $a$ is true.
- If $(a \rightarrow b)$ is true and $(\neg c \rightarrow d)$ then $(a \rightarrow b) \lor (\neg c \rightarrow d)$ is true.
Implication

- An implication is true iff the first argument is false or both arguments are true (Ex falsum quodlibet.)
- Alternative notation: $\phi \supset \psi$, $\phi \text{ IMPL } \psi$

**Truth table:**

<table>
<thead>
<tr>
<th>$\phi$</th>
<th>$\psi$</th>
<th>$\phi \rightarrow \psi$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</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>

**Example:**

- If atom "It rains." is true and atom "The street is wet." is true then the statement "If it rains, the street is wet." is true.
- $(\bot \rightarrow a)$ and $(a \rightarrow a)$ are always true. $\top \rightarrow \phi$ is true if $\phi$ is true.
Equivalence

- true iff both subformulas have the same value
- alternative notation: \( \phi = \psi, \phi \equiv \psi, \phi \sim \psi \)

Truth table:

<table>
<thead>
<tr>
<th>( \phi )</th>
<th>( \psi )</th>
<th>( \phi \leftrightarrow \psi )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</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>

Example:
- The formula \( a \leftrightarrow a \) is always true.
- The formula \( a \leftrightarrow b \) is true iff \( a \) is true and \( b \) is true or \( a \) is false and \( b \) is false.
- \( \top \leftrightarrow \bot \) is never true.
## The Logic Connectives at a Glance

<table>
<thead>
<tr>
<th>$\phi$</th>
<th>$\psi$</th>
<th>$\top$</th>
<th>$\bot$</th>
<th>$\neg \phi$</th>
<th>$\phi \land \psi$</th>
<th>$\phi \lor \psi$</th>
<th>$\phi \rightarrow \psi$</th>
<th>$\phi \leftrightarrow \psi$</th>
<th>$\phi \oplus \psi$</th>
<th>$\phi \uparrow \psi$</th>
<th>$\phi \downarrow \psi$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

### Example:

<table>
<thead>
<tr>
<th>$\phi$</th>
<th>$\psi$</th>
<th>$\neg (\neg \phi \land \neg \psi)$</th>
<th>$\neg \phi \lor \psi$</th>
<th>$(\phi \rightarrow \psi) \land (\psi \rightarrow \phi)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

**Observation:** Connectives can be expressed by other connectives.
Other Connectives

- there are 16 different functions for binary connectives
- so far, we had $\land$, $\lor$, $\leftrightarrow$, $\rightarrow$
- further connectives:
  - $\phi \not\leftrightarrow \psi$ (also $\oplus$, xor, antivalence)
  - $\phi \uparrow \psi$ (nand, Sheffer Stroke Function)
  - $\phi \downarrow \psi$ (nor, Pierce Function)

<table>
<thead>
<tr>
<th>$\phi$</th>
<th>$\psi$</th>
<th>$\phi \not\leftrightarrow \psi$</th>
<th>$\phi \uparrow \psi$</th>
<th>$\phi \downarrow \psi$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

- nor and nand can express every other boolean function (i.e., they are functional complete)
- often used for building digital circuits (like processors)
Propositional Formulas and Digital Circuits

- **and gate**
  - Input: A, B
  - Output: A AND B

- **nand gate**
  - Input: A, B
  - Output: NOT (A AND B)

- **or gate**
  - Input: A, B
  - Output: A OR B

- **nor gate**
  - Input: A, B
  - Output: NOT (A OR B)

- **xor gate**
  - Input: A, B
  - Output: A XOR B

- **not gate**
  - Input: A
  - Output: NOT A
Example of a Digital Circuit: Half Adder

<table>
<thead>
<tr>
<th>$x$</th>
<th>$y$</th>
<th>$c$</th>
<th>$s$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

From the truth table, we see that

\[ c \iff x \land y \]

and

\[ s \iff x \oplus y. \]
## Different Notations

<table>
<thead>
<tr>
<th>operator</th>
<th>logic</th>
<th>circuits</th>
<th>C/C++/Java/C#</th>
<th>VHDL</th>
<th>Limboole</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>⊤</td>
<td>1</td>
<td>true</td>
<td>1</td>
<td>–</td>
</tr>
<tr>
<td>0</td>
<td>⊥</td>
<td>0</td>
<td>false</td>
<td>0</td>
<td>–</td>
</tr>
<tr>
<td>negation</td>
<td>¬φ</td>
<td>φ̅</td>
<td>¬φ</td>
<td>!φ</td>
<td>!φ</td>
</tr>
<tr>
<td>conjunction</td>
<td>φ ∧ ψ</td>
<td>φψ</td>
<td>φ · ψ</td>
<td>φ &amp; &amp; ψ</td>
<td>φ and ψ</td>
</tr>
<tr>
<td>disjunction</td>
<td>φ ∨ ψ</td>
<td>φ + ψ</td>
<td>φ</td>
<td></td>
<td>ψ</td>
</tr>
<tr>
<td>exclusive or</td>
<td>φ ⊕ ψ</td>
<td>φ ⊕ ψ</td>
<td>φ != ψ</td>
<td>φ xor ψ</td>
<td>–</td>
</tr>
<tr>
<td>implication</td>
<td>φ → ψ</td>
<td>φ ⊃ ψ</td>
<td>–</td>
<td>–</td>
<td>φ -&gt; ψ</td>
</tr>
<tr>
<td>equivalence</td>
<td>φ ↔ ψ</td>
<td>φ = ψ</td>
<td>φ == ψ</td>
<td>φ xnor ψ</td>
<td>φ &lt;-&gt; ψ</td>
</tr>
</tbody>
</table>

### Example:
- \[(a ∨ (b ∨ ¬c)) ↔ (⊤ ∧ ((a → ¬b) ∨ (c ∨ a ∨ b)))\]
- \[(a + (b + ¬c)) = c ((a ⊃ ¬b) + (0 + a + b))\]
- \[(a || (b || !c)) == (c & & (((a || ! b) || (false || a || b)))))\]
## All 16 Binary Functions

<table>
<thead>
<tr>
<th>$\phi$</th>
<th>$\psi$</th>
<th>constant 0</th>
<th>nor</th>
<th>xor</th>
<th>nand</th>
<th>and</th>
<th>equivalence</th>
<th>implication</th>
<th>or</th>
<th>constant 1</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0 1 0 1 0 1 0 1 0 1 0 1 0 1 0 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0 0 1 1 0 0 1 1 0 0 1 1 0 0 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0 0 0 0 1 1 1 1 0 0 0 0 1 1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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 atomic propositions.
- 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>$(x \lor y) \land \neg z$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</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

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 $[x]_\nu = \nu(x)$
- $[\neg \phi]_\nu = 1$ iff $[\phi]_\nu = 0$
- $[\phi \lor \psi]_\nu = 1$ iff $[\phi]_\nu = 1$ or $[\psi]_\nu = 1$
An assignment is called
\- \textit{satisfying} a formula $\phi$ iff $[\phi]_\nu = 1$.
\- \textit{falsifying} a formula $\phi$ iff $[\phi]_\nu = 0$.

A satisfying assignment for $\phi$ is a \textit{model} of $\phi$.

A falsifying assignment for $\phi$ is a \textit{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.
Properties of Propositional Formulas (1/3)

- formula $\phi$ is **satisfiable** iff
  there exists interpretation $[.]_\nu$ with $[\phi]_\nu = 1$
  check with limboole -s

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

- formula $\phi$ is **refutable** iff
  exists interpretation $[.]_\nu$ with $[\phi]_\nu = 0$
  check with limboole

- formula $\phi$ is **unsatisfiable** iff
  $[\phi]_\nu = 0$ for all interpretations $[.]_\nu$
  check with limboole -s
Properties of Propositional Formulas (2/3)

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

Example:

- $\top$ is valid.
- $\bot$ is unsatisfiable.
- $(a \lor \neg b) \land (\neg a \lor b)$ is refutable.
- $a \rightarrow b$ is satisfiable.
- $a \leftrightarrow \neg a$ is a contradiction.
- $(a \lor \neg b) \land (\neg a \lor b)$ is satisfiable.
Properties of Propositional Formulas (3/3)

■ A satisfiable formula is
  □ possibly valid
  □ possibly refutable
  □ not unsatisfiable.

■ A valid formula is
  □ satisfiable
  □ not refutable
  □ not unsatisfiable.

■ A refutable formula is
  □ possibly satisfiable
  □ possibly unsatisfiable
  □ not valid.

■ An unsatisfiable formula is
  □ refutable
  □ not valid
  □ not satisfiable.

Example:

■ satisfiable, but not valid: $a \leftrightarrow b$

■ satisfiable and refutable: $(a \lor b) \land (\neg a \lor c)$

■ valid, not refutable $\top \lor (a \land \neg a)$; not valid, refutable $\bot \lor b$
Further Connections between Formulas

- A formula $\phi$ is valid iff $\neg \phi$ is unsatisfiable.

- A formula $\phi$ is satisfiable iff $\neg \phi$ is not valid.

- The formulas $\phi$ and $\psi$ are equivalent iff $\phi \leftrightarrow \psi$ is valid.

- The formulas $\phi$ and $\psi$ are equivalent iff $\neg(\phi \leftrightarrow \psi)$ is unsatisfiable.

- A formula $\phi$ is satisfiable iff $\phi \not\leftrightarrow \bot$. 

$\Box$
Simple Algorithm for Satisfiability Checking

1 Algorithm: evaluate
   Data: formula φ
   Result: 1 iff φ is satisfiable

2 if φ contains a variable x then
   pick v ∈ {T, ⊥}
   /* replace x by truth constant v, evaluate resulting formula */
   if evaluate(φ[x|v]) then return 1;
   else return evaluate(φ[x|¬v]) ;

7 else
   switch φ do
   case T return 1;
   case ⊥ return 0;
   case ¬ψ return ! evaluate(ψ) /* true iff ψ is false */ ;
   case ψ' ∧ ψ''
   return evaluate(ψ') && evaluate(ψ'') /* true iff both ψ' and ψ'' are true */
   case ψ' ∨ ψ''
   return evaluate(ψ') || evaluate(ψ'') /* true iff ψ' or ψ'' is true */
Semantic Equivalence

Two formulas $\phi$ and $\psi$ are **semantic equivalent** (written as $\phi \iff \psi$) iff for all interpretations $[. ]_\nu$ it holds that $[\phi]_\nu = [\psi]_\nu$.

Note:

- $\iff$ is a *meta-symbol*, i.e., it is not part of the language.
- *natural language*: if and only if (iff)
- $\phi \iff \psi$ iff $\phi \leftrightarrow \psi$ is valid, i.e., we can express semantics by means of syntactics.
- If $\phi$ and $\psi$ are not equivalent, we write $\phi \not\iff \psi$.

**Example:**

- $a \lor \neg a \not\iff b \rightarrow \neg b$
- $(a \lor b) \land \neg(a \lor b) \iff \bot$
- $a \lor \neg a \iff b \lor \neg b$
- $a \iff (b \iff c)) \iff ((a \iff b) \iff c$
Examples of Semantic Equivalences (1/2)

<table>
<thead>
<tr>
<th>Example</th>
<th>Example</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\phi \land \psi \iff \psi \land \phi$</td>
<td>$\phi \lor \psi \iff \psi \lor \phi$</td>
<td>commutativity</td>
</tr>
<tr>
<td>$\phi \land (\psi \land \gamma) \iff (\phi \land \psi) \land \gamma$</td>
<td>$\phi \lor (\psi \lor \gamma) \iff (\phi \lor \psi) \lor \gamma$</td>
<td>associativity</td>
</tr>
<tr>
<td>$\phi \land (\phi \lor \psi) \iff \phi$</td>
<td>$\phi \lor (\phi \land \psi) \iff \phi$</td>
<td>absorption</td>
</tr>
<tr>
<td>$\phi \land (\psi \lor \gamma) \iff (\phi \land \psi) \lor (\phi \land \gamma)$</td>
<td>$\phi \lor (\psi \land \gamma) \iff (\phi \lor \psi) \land (\phi \lor \gamma)$</td>
<td>distributivity</td>
</tr>
<tr>
<td>$\neg (\phi \land \psi) \iff \neg \phi \lor \neg \psi$</td>
<td>$\neg (\phi \lor \psi) \iff \neg \phi \land \neg \psi$</td>
<td>laws of De Morgan</td>
</tr>
<tr>
<td>$\phi \iff \psi \iff (\phi \rightarrow \psi) \land (\psi \rightarrow \phi)$</td>
<td>$\phi \iff \psi \iff (\phi \land \psi) \lor (\neg \phi \land \neg \psi)$</td>
<td>synt. equivalence</td>
</tr>
</tbody>
</table>
Examples of Semantic Equivalences (2/2)

<table>
<thead>
<tr>
<th>$\phi \lor \psi \iff \neg \phi \rightarrow \psi$</th>
<th>$\phi \rightarrow \psi \iff \neg \psi \rightarrow \neg \phi$</th>
<th>implications</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\phi \land \neg \phi \iff \bot$</td>
<td>$\phi \lor \neg \phi \iff \top$</td>
<td>complement</td>
</tr>
<tr>
<td>$\neg \neg \phi \iff \phi$</td>
<td></td>
<td>double negation</td>
</tr>
<tr>
<td>$\phi \land \top \iff \phi$</td>
<td>$\phi \lor \bot \iff \phi$</td>
<td>neutrality</td>
</tr>
<tr>
<td>$\phi \lor \top \iff \top$</td>
<td>$\phi \land \bot \iff \bot$</td>
<td></td>
</tr>
<tr>
<td>$\neg \top \iff \bot$</td>
<td>$\neg \bot \iff \top$</td>
<td></td>
</tr>
</tbody>
</table>
Logic Entailment

Let $\phi_1, \ldots, \phi_n, \psi$ be propositional formulas. Then $\phi_1, \ldots, \phi_n$ entail $\psi$ (written as $\phi_1, \ldots, \phi_n \models \psi$) iff $[\phi_1]_\nu = 1, \ldots, [\phi_n]_\nu = 1$ implies that $[\psi]_\nu = 1$.

Informal meaning: True premises derive a true conclusion.

- $\models$ is a meta-symbol, i.e., it is not part of the language.
- $\phi_1, \ldots, \phi_n \models \psi$ iff $(\phi_1 \land \ldots \land \phi_n) \rightarrow \psi$ is valid, i.e., we can express semantics by means of syntactics.
- If $\phi_1, \ldots, \phi_n$ do not entail $\psi$, we write $\phi_1, \ldots, \phi_n \not\models \psi$.

Example:

- $a \models a \lor b$
- $\models a \lor \neg a$
- $a, a \rightarrow b \models b$
- $a, b \models a \land b$
- $\not\models a \land \neg a$
- $\bot \models a \land \neg a$
Satisfiability Equivalence

Two formulas $\phi$ and $\psi$ are *satisfiability-equivalent* (written as $\phi \Leftrightarrow_{SAT} \psi$) iff both formulas are satisfiable or both are contradictory.

- Satisfiability-equivalent formulas are not necessarily satisfied by the same assignments.
- Satisfiability equivalence is a weaker property than semantic equivalence.
- Often sufficient for simplification rules: If the complicated formula is satisfiable then also the simplified formula is satisfiable.
Example: Satisfiability Equivalence

**positive pure literal elimination rule:**

If a variable \( x \) occurs in a formula but \( \neg x \) does not occur in the formula, then \( x \) can be substituted by \( \top \). The resulting formula is satisfiability-equivalent.

**Example:**

- \( x \Leftrightarrow_{SAT} \top \), but \( x \not\Leftrightarrow \top \)
- \( (a \land b) \lor (\neg c \land a) \Leftrightarrow_{SAT} b \lor \neg c \), but \( (a \land b) \lor (\neg c \land a) \not\Leftrightarrow b \lor \neg c \)