PROPOSITIONAL FORMULAS: SATISFIABILITY EQUIVALENCES

VL Logic
Part I: Propositional Logic
Recap: Semantic Equivalence

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

- $\iff$ is a **meta-symbol**, i.e., it is not part of the language.
- $\phi \iff \psi$ 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:**

- $a \lor \neg a \not\iff b \rightarrow \neg b$
- $a \lor \neg a \iff b \lor \neg b$
- $(a \lor b) \land \neg(a \lor b) \iff \bot$
- $a \iff (b \iff c) \iff (a \iff b) \iff c$
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 literal $x$ occurs in an NNF formula but $\neg x$ does not occur in this formula, then $x$ can be substituted by $\top$. The resulting formula is satisfiability-equivalent.

Example:

- $x \iff_{SAT} \top$, but $x \not\iff \top$
- $(a \land b) \lor (\neg c \land a) \iff_{SAT} b \lor \neg c$, but $(a \land b) \lor (\neg c \land a) \not\iff b \lor \neg c$
PROPOSITIONAL FORMULAS: FUNCTIONAL COMPLETENESS

VL Logic
Part I: Propositional Logic
### Examples of Semantic Equivalences (1/2)

<table>
<thead>
<tr>
<th>Commutativity</th>
<th>Associativity</th>
<th>Absorption</th>
<th>Distributivity</th>
<th>Laws of De Morgan</th>
<th>Synt. Equivalence</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \phi \land \psi \iff \psi \land \phi )</td>
<td>( \phi \lor \psi \iff \psi \lor \phi )</td>
<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>( \neg (\phi \land \psi) \iff \neg \phi \lor \neg \psi )</td>
<td>( \phi \iff \psi \iff (\phi \rightarrow \psi) \land (\psi \rightarrow \phi) )</td>
</tr>
<tr>
<td>( \phi \lor (\phi \land \psi) \iff \phi \lor (\phi \land \psi) )</td>
<td>( \phi \land (\phi \land \psi) \iff \phi )</td>
<td>( \phi \lor (\phi \land \psi) \iff \phi )</td>
<td>( \phi \land (\psi \land \gamma) \iff (\phi \land \psi) \land (\phi \land \gamma) )</td>
<td>( \neg (\phi \lor \psi) \iff \neg \phi \land \neg \psi )</td>
<td>( \phi \iff \psi \iff (\phi \lor \psi) \lor (\neg \phi \land \neg \psi) )</td>
</tr>
</tbody>
</table>
## Examples of Semantic Equivalences (2/2)

<table>
<thead>
<tr>
<th>( \phi \lor \psi \leftrightarrow \neg \phi \rightarrow \psi )</th>
<th>( \phi \rightarrow \psi \leftrightarrow \neg \psi \rightarrow \neg \phi )</th>
<th>implications</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \phi \land \neg \phi \leftrightarrow \bot )</td>
<td>( \phi \lor \neg \phi \leftrightarrow \top )</td>
<td>complement</td>
</tr>
<tr>
<td>( \neg \neg \phi \leftrightarrow \phi )</td>
<td></td>
<td>double negation</td>
</tr>
<tr>
<td>( \phi \land \top \leftrightarrow \phi )</td>
<td>( \phi \lor \bot \leftrightarrow \phi )</td>
<td>neutrality</td>
</tr>
<tr>
<td>( \phi \lor \top \leftrightarrow \top )</td>
<td>( \phi \land \bot \leftrightarrow \bot )</td>
<td></td>
</tr>
<tr>
<td>( \neg \top \leftrightarrow \bot )</td>
<td>( \neg \bot \leftrightarrow \top )</td>
<td></td>
</tr>
</tbody>
</table>
In propositional logic there are
- 2 functions of arity 0 (\( \top, \bot \))
- 4 functions of arity 1 (e.g., not)
- 16 functions of arity 2 (e.g., and, or, ...)
- \(2^n\) functions of arity \(n\).

A function of arity \(n\) has \(2^n\) different combinations of arguments (lines in the truth table).

A functions maps its arguments either to 1 or 0.

A set of functions is called **functional complete** for propositional logic iff it is possible to express all other functions of propositional logic with functions from this set.

\{\neg, \land\}, \{\neg, \lor\}, \{\text{nand}\} are functional complete.
PROPOSITIONAL FORMULAS: NORMAL FORMS

VL Logic
Part I: Propositional Logic
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.

A formula in conjunctive normal form is

- in negation normal form
- $\top$ if it contains no clauses
- easy to check whether it can be refuted

remark: CNF is the input of most SAT-solvers (DIMACS format)
Disjunctive Normal Form (DNF)

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 cubes
- easy to check whether it can be satisfied
Examples for CNF and DNF

Examples CNF

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

Examples DNF

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

- **Problem**: Given the truth table of a Boolean function $\phi$. How is the function represented in propositional logic?

**Solution (in CNF):**

1. Represent each assignment $\nu$ where $\phi$ has value 0 as clause:
   - If variable $x$ is 1 in $\nu$, add $\neg x$ to clause.
   - If variable $x$ is 0 in $\nu$, add $x$ to clause.

2. Connect all clauses by conjunction.

```
<table>
<thead>
<tr>
<th>a</th>
<th>b</th>
<th>c</th>
<th>$\phi$</th>
<th>clauses</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>$a \lor b \lor c$</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>$a \lor \neg b \lor \neg c$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>$\neg a \lor b \lor \neg c$</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>$\neg a \lor \neg b \lor c$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>$\neg a \lor \neg b \lor c$</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>$\neg a \lor b \lor \neg c$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>$\neg a \lor \neg b \lor c$</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>$\neg a \lor \neg b \lor c$</td>
</tr>
</tbody>
</table>

$\phi = (a \lor b \lor c) \land (a \lor \neg b \lor \neg c) \land (\neg a \lor b \lor \neg c) \land (\neg a \lor \neg b \lor c)$
Representing Functions as DNFs

- **Problem**: Given the truth table of a Boolean function $\phi$. How is the function represented in propositional logic?

**Solution (in DNF):**

1. Represent each assignment $\nu$ where $\phi$ has value 1 as cube:
   - If variable $x$ is 1 in $\nu$, add $x$ to cube.
   - If variable $x$ is 0 in $\nu$, add $\neg x$ to cube.

2. Connect all cubes by disjunction.

<table>
<thead>
<tr>
<th>$a$</th>
<th>$b$</th>
<th>$c$</th>
<th>$\phi$</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>1</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>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

$$\phi = (\neg a \land \neg b \land c) \lor (\neg a \land b \land \neg c) \lor (a \land \neg b \land \neg c) \lor (a \land b \land c)$$
PROPOSITIONAL FORMULAS: NORMAL FORM TRANSFORMATION

VL Logic
Part I: Propositional Logic
Transformation to Conjunctive Normal Form 1

Approach 1: Transformation by “multiplication”

1. Remove $\leftrightarrow, \rightarrow, \oplus$ as follows:
   - $\phi \leftrightarrow \psi \iff (\phi \Rightarrow \psi) \land (\psi \Rightarrow \phi)$
   - $\phi \rightarrow \psi \iff \neg \phi \lor \psi$
   - $\phi \oplus \psi \iff (\phi \lor \psi) \land (\neg \phi \lor \neg \psi)$

2. Transform formula to negation normal form (NNF) by
   - application of laws of De Morgan
   - elimination of double negation

3. Transform formula to CNF by laws of distributivity
Example: Transformation to CNF 1

Transformation of $\neg(a \leftrightarrow b) \rightarrow (\neg(c \land d) \land e)$ to an equivalent formula in CNF by approach 1

1. a) remove equivalences: $\iff \neg((a \rightarrow b) \land (b \rightarrow a)) \rightarrow (\neg(c \land d) \land e)$
   
   b) remove implications: $\iff \neg\neg((\neg a \lor b) \land (\neg b \lor a)) \lor (\neg(c \land d) \land e)$

2. NNF: $\iff ((\neg a \lor b) \land (\neg b \lor a)) \lor ((\neg c \lor \neg d) \land e)$

3. $\iff ((\neg a \lor b) \lor ((\neg c \lor \neg d) \land e)) \land ((\neg b \lor a) \lor ((\neg c \lor \neg d) \land e))$

   $\iff (\neg a \lor b \lor \neg c \lor \neg d) \land (\neg a \lor b \lor e) \land (\neg b \lor a \lor \neg c \lor \neg d) \land (\neg b \lor a \lor e)$
Approach 2: Transformation by introducing labels for subformulas

Given formula $\phi$. The following approach transforms $\phi$ to an equi-satisfiable formula in CNF.

1. Introduce new label $\ell_\psi$ for each subformula $\psi$ that is not a literal
2. Collect all definitions $\ell_\psi \leftrightarrow \psi'$ in a big conjunction $\Phi'$
   ($\psi'$ is obtained from $\psi$ by replacing its immediate subformulas by the respective labels)
3. Transform $\Phi'$ to CNF $\Phi$ by approach 1 (no exponential blowup!)

$(\Phi \land \ell_\phi)$ and $\phi$ are equi-satisfiable
Example: Transformation to CNF 2

Transform $\phi := \neg(a \leftrightarrow b) \rightarrow (\neg(c \land d) \land e)$ to an equi-satisfiable formula in CNF

<table>
<thead>
<tr>
<th>definitions $\Phi'$</th>
<th>clauses $\Phi$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$v_1 \leftrightarrow (a \leftrightarrow b)$</td>
<td>$(\bar{v}_1 \lor \bar{a} \lor b), (\bar{v}_1 \lor a \lor \bar{b}), (v_1 \lor \bar{a} \lor \bar{b}), (v_1 \lor a \lor b)$</td>
</tr>
<tr>
<td>$v_2 \leftrightarrow \neg v_1$</td>
<td>$(v_1 \lor v_2), (\bar{v}_1 \lor \bar{v}_2)$</td>
</tr>
<tr>
<td>$v_3 \leftrightarrow (c \land d)$</td>
<td>$(\bar{v}_3 \lor c), (\bar{v}_3 \lor d), (v_3 \lor \bar{c} \lor \bar{d})$</td>
</tr>
<tr>
<td>$v_4 \leftrightarrow \neg v_3$</td>
<td>$(v_3 \lor v_4), (\bar{v}_3 \lor \bar{v}_4)$</td>
</tr>
<tr>
<td>$v_5 \leftrightarrow (v_4 \land e)$</td>
<td>$(\bar{v}_5 \lor v_4), (\bar{v}_5 \lor e), (v_5 \lor \bar{v}_4 \lor \bar{e})$</td>
</tr>
<tr>
<td>$v_6 \leftrightarrow (v_2 \rightarrow v_5)$</td>
<td>$(\bar{v}_6 \lor \bar{v}_2 \lor v_5), (v_2 \lor v_6), (\bar{v}_5 \lor v_6)$</td>
</tr>
</tbody>
</table>

$(\Phi \land v_6)$ and $\phi$ are equi-satisfiable.
Some Remarks on Normal Forms

- Approach 1 is exponential in the worst case (e.g., transform
  \((a_1 \land b_1) \lor (a_2 \land b_2) \lor \cdots \lor (a_n \land b_n)\) to CNF).

- Approach 2 is polynomial
  - Basic idea: introduce labels for subformulas.
  - Also works for formulas with sharing (circuits).
  - Also known as “Tseitin Encoding”.

- CNF is usually not easier to solve, but easier to handle:
  - compact data structures: a CNF is simply a list of lists of literals.

- CNF very popular in practice: standard input format DIMACS

- To solve satisfiability of CNF, there are many optimization techniques and dedicated algorithms.
PROPOSITIONAL FORMULAS: RESOLUTION

VL Logic
Part I: Propositional Logic
Resolution

- the resolution calculus consists of the single resolution rule

\[
\frac{x \lor C \quad \neg x \lor D}{C \lor D}
\]

- \(C\) and \(D\) are (possibly empty) clauses
- the clause \(C \lor D\) is called resolvent
- variable \(x\) is called pivot
- usually antecedent clauses \(x \lor C\) and \(\neg x \lor D\) are assumed not to be tautological

- resolution is sound and complete.

- the resolution calculus works only on formulas in CNF
- if the empty clause can be derived then the formula is unsatisfiable
- if no new clause can be generated by application of the resolution rule then the formula is satisfiable

Example

one application of resolution

\[
\frac{x \lor y \lor \neg z \quad \neg x \lor y' \lor \neg z}{y \lor \neg z \lor y'}
\]

derivation of empty clause:

\[
\frac{y \quad \neg y}{\bot}
\]

derivation of tautology:

\[
\frac{x \lor a \quad \neg x \lor \neg a}{a \lor \neg a}
\]
Resolution Example

We prove unsatisfiability of

\{(¬x_1 \lor ¬x_5), (x_4 \lor x_5), (x_2 \lor ¬x_4), (x_3 \lor ¬x_4), (¬x_2 \lor ¬x_3), (x_1 \lor x_4 \lor ¬x_6), (x_6)\}

as follows: