PROPOSITIONAL LOGIC IN CNF

VL Logik: WS 2019/20

(Version 2019.2)

Martina Seidl (martina.seidl@jku.at),
Armin Biere (biere@jku.at)
Institut für Formale Modelle und Verifikation

JOHANNES KEPLER
UNIVERSITY LINZ

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: bulid 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 ($\wedge$), or ($\vee$), ...
    - operators for construction of composite propositions
    - concise meaning
    - argument(s) and return value from Boolean domain
  - parenthesis

example: formula of propositional logic: $(\neg \mathbf{t} \vee \mathbf{s}) \wedge (\mathbf{t} \vee \mathbf{s}) \wedge (\neg \mathbf{t} \vee \neg \mathbf{s})$

atoms: **t**, **s**, connectives: $\neg$, $\vee$, $\wedge$, 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 $(\vee)$ of literals
  - $x \vee y$ (binary clause)
  - $x \vee y \vee \neg z$ (ternary clause)
  - z (unary clause)
  - $\neg\top$ (unary clause)
  - for $(l_1 \vee \ldots \vee l_n)$ 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 ($\wedge$) of clauses.

examples of formulas:

- $\top$
- $\bot$
- x
- $\neg y$

- $x \wedge y \wedge z$
- $(\neg x \vee y \vee \neg z) \wedge z$
- $(x \vee \neg y) \wedge (x \vee \neg y \vee z) \wedge (y \vee \neg z)$
- $((l_{11} \vee \ldots \vee l_{1m_1}) \wedge \ldots \wedge (l_{n1} \vee \ldots \vee l_{nm_n}))$
- for $(C_1 \wedge \ldots \wedge C_n)$ we also write $\bigwedge_{i=1}^{n} C_i$.

Remark: For the moment, we consider formulas of a restricted structure called CNF, e.g., we do not consider formulas like $(x \wedge y) \vee (\neg x \wedge z)$. 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
- ϕ, ψ, γ denote arbitrary formulas
- C, D denote clauses
- clauses are also written as sets
  - $(l_1 \vee \ldots \vee l_n) = \{l_1, \ldots l_n\}$
  - to add a literal l to clause C, we write $C \cup \{l\}$
  - to remove a literal l from clause C, we write $C \backslash \{l\}$
- formulas in CNF are also written as sets of sets
  - $((l_{11} \vee \ldots \vee l_{1m_1}) \wedge \ldots \wedge (l_{n1} \vee \ldots \vee l_{nm_n})) = \{\{l_{11}, \ldots l_{1m_1}\}, \ldots, \{l_{n1}, \ldots l_{nm_n}\}\}$
  - to add a clause C to CNF ϕ, we write $\phi \cup \{C\}$
  - to remove a clause C from CNF ϕ, we write $\phi \backslash \{C\}$

Negation Operator

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

truth table:

x	$\neg x$
0	**1**
1	**0**

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

truth table:

l	k	$l \vee k$
0	**0**	**0**
0	**1**	**1**
1	**0**	**1**
1	**1**	**1**

example:

- $(a \vee \neg a)$ is always true.
- $(\top \vee a)$ is always true.
- $(\bot \vee a)$ is true if a is true.

Properties of Disjunction

- commutative:

$$k \vee l \Leftrightarrow l \vee k$$

- idempotent:

$$l \vee l \Leftrightarrow l$$

- associative:

$$l_1 \vee (l_2 \vee l_3) \Leftrightarrow (l_1 \vee l_2) \vee l_3$$

Clause: Semantics

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

truth table:

l_1	$\dots$	l_n	$l_1 \vee l_2 \vee \dots \vee l_n$
0	$\dots$	**0**	**0**
0	$\dots$	**1**	**1**
	$\dots$		**1**
1	$\dots$	**0**	**1**
1	$\dots$	**1**	**1**

Binary Conjunction Operator

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

truth table:

C	D	$C \wedge D$
0	**0**	**0**
0	**1**	**0**
1	**0**	**0**
1	**1**	**1**

example:

- $(a \wedge \neg a)$ is always false.
- $(\top \wedge a)$ is true if a is true. $(\bot \wedge \phi)$ is always false.
- If $(a \vee b)$ is true and $(\neg c \vee d)$ is true then $(a \vee b) \wedge (\neg c \vee d)$ is true.

Properties of Conjunction

- commutative:

$$C \wedge D \Leftrightarrow D \wedge C$$

- idempotent:

$$C \wedge C \Leftrightarrow C$$

- associative:

$$C_1 \wedge (C_2 \wedge C_3) \Leftrightarrow (C_1 \wedge C_2) \wedge C_3$$

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:

C_1	$\dots$	C_n	$C_1 \land C_2 \land \dots \land C_n$
0	$\dots$	**0**	**0**
0	$\dots$	**1**	**0**
	$\dots$		**0**
1	$\dots$	**0**	**0**
1	$\dots$	**1**	**1**

Rules of Precedence

- $\neg$ binds stronger than $\land$
- $\land$ binds stronger than $\lor$

<u>example</u>

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

$\Rightarrow$ put clauses into parentheses!

Assignment

- a variable can be assigned one of two values from the two-valued domain $\mathbb{B}$, where $\mathbb{B} = \{\mathbf{1}, \mathbf{0}\}$
- the mapping $\nu : \mathcal{P} \to \mathbb{B}$ is called <u>assignment</u>, where $\mathcal{P}$ is the set of variables of a formula
- we sometimes write an assignment ν as set V with $V \subseteq \mathcal{P} \cup \{\neg x | x \in \mathcal{P}\}$ such that
  - $x \in V$ iff $\nu(x) = \mathbf{1}$
  - $\neg x \in V$ iff $\nu(x) = \mathbf{0}$
- for n variables, there are 2^n assignments possible
- an assignment corresponds to one line in the truth table

Assignment: Example

x	y	z	$x \vee y$	$\neg z$	$(x \vee y) \wedge \neg z$
0	**0**	**0**	**0**	**1**	**0**
0	**0**	**1**	**0**	**0**	**0**
0	**1**	**0**	**1**	**1**	**1**
0	**1**	**1**	**1**	**0**	**0**
1	**0**	**0**	**1**	**1**	**1**
1	**0**	**1**	**1**	**0**	**0**
1	**1**	**0**	**1**	**1**	**1**
1	**1**	**1**	**1**	**0**	**0**

- one assignment: $\nu(x) = \mathbf{1}, \nu(y) = \mathbf{0}, \nu(z) = \mathbf{1}$
- 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 $\nu : \mathcal{P} \to \mathbb{B}$, the interpretation $[.]_\nu : \mathcal{L} \to \mathbb{B}$ is defined by:

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

Satisfying/Falsifying Assignments

- an assignment ν is called
  - satisfying a formula ϕ iff $[\phi]_\nu = \mathbf{1}$
  - falsifying a formula ϕ iff $[\phi]_\nu = \mathbf{0}$
- a satisfying assignment for ϕ is a model of ϕ
- a falsifying assignment for ϕ is a counter-model of ϕ

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:[1]
  - variables are strings over letters, digits and – _ . [ ] $ @
  - negation symbol $\neg$ is `!`
  - disjunction symbol $\vee$ is `|`
  - conjunction symbol $\wedge$ is &

<u>example</u>

$(a \vee b \vee \neg c) \wedge (\neg a \vee b) \wedge c$ is represented as

`(a | b | !c) & (!a | b) & c`

[1] For now, we will only use subset of the language supported by Limboole.

Properties of Propositional Formulas (1/2)

- formula ϕ is <u>satisfiable</u> iff
  there exists an assignment ν with $[\phi]_\nu = \mathbf{1}$
  check with `limboole -s`
- formula ϕ is <u>valid</u> iff
  for all assignments ν it holds that $[\phi]_\nu = \mathbf{1}$
  check with `limboole`
- formula ϕ is <u>refutable</u> iff
  there exists an assignment ν with $[\phi]_\nu = \mathbf{0}$
  check with `limboole`
- formula ϕ is <u>unsatisfiable</u> iff
  for all assignments ν it holds that $[\phi]_\nu = \mathbf{0}$
  check with `limboole -s`

Properties of Propositional Formulas (2/2)

- a valid formula is called <u>tautology</u>
- an unsatisfiable formula is called <u>contradiction</u>

example:

- $\top$ is valid.
- $a \vee \neg a$ is a tautology.
- $(a \vee \neg b) \wedge (\neg a \vee b)$ is refutable.
- $\bot$ is unsatisfiable.
- $a \wedge \neg a$ is a contradiction.
- $(a \vee \neg b) \wedge (\neg a \vee b)$ is satisfiable.

SAT: The Boolean Satisfiability Problem

Given a propositional formula ϕ.

Is there an assignment that satisfies ϕ?

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

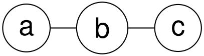
Application: Graph Coloring

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) $E = \{(a, b), (b, c)\}$



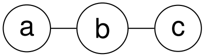
Application: Graph Coloring

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.

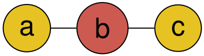
encoding:

1. propositional variables: v_j ... node $v \in V$ has color j $(1 \leq j \leq k)$
2. each node has a color:
$$\bigwedge_{v \in V} (\bigvee_{1 \leq j \leq k} v_j)$$
3. each node has just one color: $(\neg v_i \vee \neg v_j)$ with $v \in V, 1 \leq i < j \leq k$
4. neighbors have different colors: $(\neg v_i \vee \neg w_i)$ 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:

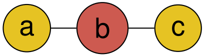


encoding in SAT:

Encoding the k-Coloring Problem: Example

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

possible solution:



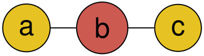
encoding in SAT:

- 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:



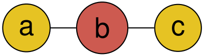
encoding in SAT:

- variables: $a_1, a_2, b_1, b_2, c_1, c_2$
- clauses:
  1. each node has a color: $(a_1 \vee a_2), (b_1 \vee b_2), (c_1 \vee c_2)$
  2. no node has two colors: $(\neg a_1 \vee \neg a_2), (\neg b_1 \vee \neg b_2), (\neg c_1 \vee \neg c_2)$
  3. connected nodes have a different color:
     $(\neg a_1 \vee \neg b_1), (\neg a_2 \vee \neg b_2), (\neg b_1 \vee \neg c_1), (\neg b_2 \vee \neg c_2)$

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: $(a_1 \vee a_2), (b_1 \vee b_2), (c_1 \vee c_2)$
  2. no node has two colors: $(\neg a_1 \vee \neg a_2), (\neg b_1 \vee \neg b_2), (\neg c_1 \vee \neg c_2)$
  3. connected nodes have a different color:
     $(\neg a_1 \vee \neg b_1), (\neg a_2 \vee \neg b_2), (\neg b_1 \vee \neg c_1), (\neg b_2 \vee \neg c_2)$
- full formula:
  $(a_1 \vee a_2) \wedge (b_1 \vee b_2) \wedge (c_1 \vee c_2) \wedge (\neg a_1 \vee \neg a_2) \wedge (\neg b_1 \vee \neg b_2) \wedge (\neg c_1 \vee \neg c_2) \wedge$
  $(\neg a_1 \vee \neg b_1) \wedge (\neg a_2 \vee \neg b_2) \wedge (\neg b_1 \vee \neg c_1) \wedge (\neg b_2 \vee \neg c_2)$