PROPOSITIONAL LOGIC: THE FULL LANGUAGE

VL Logic, Lecture 2, WS 20/21

Armin Biere, Martina Seidl

Institute for Formal Models and Verification
Johannes Kepler University Linz

JOHANNES KEPLER
UNIVERSITY LINZ

PROPOSITIONAL FORMULAS: SYNTAX

VL Logic

Part I: Propositional Logic

JOHANNES KEPLER
UNIVERSITY LINZ

Example: Party Planning

We want to plan a party.

Unfortunately, the selection of the guests is not straight forward.

We have to consider the following rules.

1. If two people are married, we have to invite them both or none of them. Alice is married to Bob and Cecile is married to David.
2. If we invite Alice then we also have to invite Cecile.
   Cecile does not care if we invite Alice but not her.
3. David and Eva can't stand each other, so it is not possible to invite both. One of them should be invited.
4. We want to invite Bob and Fred.

Question: Can we find a guest list?

Party Planning with Propositional Logic

- **propositional variables**:
  inviteAlice, inviteBob, inviteCecile, inviteDavid, inviteEva, inviteFred
- **constraints**:
  1. invite married: inviteAlice $\leftrightarrow$ inviteBob, inviteCecile $\leftrightarrow$ inviteDavid
  2. if Alice then Cecile: inviteAlice $\rightarrow$ inviteCecile
  3. either David or Eva: $\neg$ (inviteEva $\leftrightarrow$ inviteDavid)
  4. invite Bob and Fred: inviteBob $\wedge$ inviteFred
- **encoding in propositional logic**:

  (inviteAlice $\leftrightarrow$ inviteBob) $\wedge$ (inviteCecile $\leftrightarrow$ inviteDavid) $\wedge$
  (inviteAlice $\rightarrow$ inviteCecile) $\wedge$ $\neg$ (inviteEva $\leftrightarrow$ inviteDavid) $\wedge$
  inviteBob $\wedge$ inviteFred

Syntax of Propositional Logic

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 $\circ \in \{\vee, \wedge, \leftrightarrow, \rightarrow\}$.

$\mathcal{L}$ is the language of propositional logic. The elements of $\mathcal{L}$ are **propositional formulas**.

Rules of Precedence

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

- $\neg$ is stronger than $\wedge$
- $\wedge$ is stronger than $\vee$
- $\vee$ is stronger than $\rightarrow$
- $\rightarrow$ is stronger than $\leftrightarrow$
- Binary operators of same strength are assumed to be left parenthesized (also called “left associative”)

Example:

- $\neg a \wedge b \vee c \rightarrow d \leftrightarrow f$ is the same as $(((((\neg a) \wedge b) \vee c) \rightarrow d) \leftrightarrow f)$.
- $a' \vee a'' \vee a'' \wedge b' \vee b''$ is the same as $(((a' \vee a'') \vee (a'' \wedge b')) \vee b'')$.
- $a' \wedge a'' \wedge a'' \vee b' \wedge b''$ is the same as $(((a' \wedge a'') \wedge a''') \vee (b' \wedge b''))$.

Excursus: Rooted Tree

A rooted tree is a special kind of graph of the following shape:

- A single vertex v is a tree.
  The vertex v is the **root** of this tree.
- Let $t_1, \ldots, t_n$ be n trees, such that
  - t_1 has root v_1
  - $\ldots$
  - t_n has root v_n.

  Further, let v be a vertex not occuring in $t_1, \ldots, t_n$. We obtain a **tree with root** v if we add edges from v to $v_1, \ldots, v_n$.
  The vertices $v_1, \ldots, v_n$ are called **children** of v.
- Nothing else is a tree.

A node without children is called **leaf**.

Formula Tree

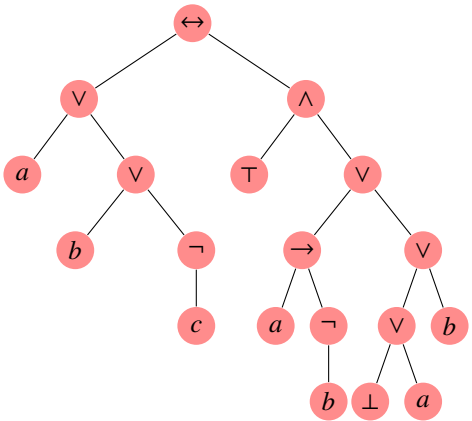
- formulas have a tree structure
  - **inner nodes**: connectives
  - **leaves**: truth constants, variables
- **default**: inner nodes have one child node (negation) or two nodes as children (other connectives).
- tree structure reflects the use of parenthesis
- **simplification**:
  disjunction and 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.

Formula Tree: Example (1/2)

The formula

$$(a \vee (b \vee \neg c)) \leftrightarrow (\top \wedge ((a \rightarrow \neg b) \vee (\bot \vee a \vee b)))$$

has the formula tree

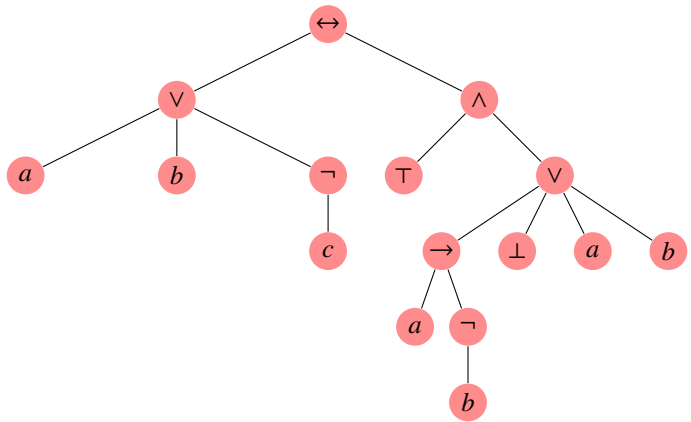


Formula Tree: Example (2/2)

The formula

$$(a \vee (b \vee \neg c)) \leftrightarrow (\top \wedge ((a \rightarrow \neg b) \vee (\bot \vee a \vee b)))$$

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 ϕ.
- formula $\phi \circ \psi$ ($\circ \in \{\wedge, \vee, \leftrightarrow, \rightarrow\}$) has immediate subformulas ϕ and ψ.

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

The **set of subformulas** of a formula ϕ is the smallest set S with

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

The subformulas of $(a \vee b) \rightarrow (c \wedge \neg\neg d)$ are

$$\{a, b, c, d, \neg d, \neg\neg d, a \vee b, c \wedge \neg\neg d, (a \vee b) \rightarrow (c \wedge \neg\neg d)\}$$

Limboole

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

$$
\begin{aligned}
\langle expr \rangle &::= \langle iff \rangle \\
\langle iff \rangle &::= \langle implies \rangle \mid \langle implies \rangle \text{ “<->” } \langle implies \rangle \\
\langle implies \rangle &::= \langle or \rangle \mid \langle or \rangle \text{ “->” } \langle or \rangle \mid \langle or \rangle \text{ “<-” } \langle or \rangle \\
\langle or \rangle &::= \langle and \rangle \mid \langle and \rangle \text{ “|” } \langle and \rangle \\
\langle and \rangle &::= \langle not \rangle \mid \langle not \rangle \text{ “&” } \langle not \rangle \\
\langle not \rangle &::= \langle basic \rangle \mid \text{“!” } \langle not \rangle \\
\langle basic \rangle &::= \langle var \rangle \mid \text{“(” } \langle expr \rangle \text{ “)”}
\end{aligned}
$$

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

In Limboole the formula $(a \vee b) \rightarrow (c \wedge \neg\neg d)$ is represented as

```
((a | b) -> (c & !!d))
```

PROPOSITIONAL FORMULAS: SEMANTICS

VL Logic

Part I: Propositional Logic

JOHANNES KEPLER
UNIVERSITY LINZ

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 **assignment**, where $\mathcal{P}$ is the set of atomic propositions
- 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

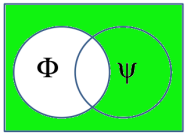
Negation

- unary connective ¬ (operator with exactly one argument)
- negating the truth value of its argument
- alternative notation: $!\phi$, $\overline{\phi}$, $-\phi$, $NOT\,\phi$

truth table:

ϕ	$\neg\phi$
0	**1**
1	**0**

set view:



Example:

- If the atom “It rains.” is true then the negation “It does not rain.” is false.
- If atom a is true then $\neg a$ is false.
- If formula $((a \vee x) \wedge y)$ is true then formula $\neg((a \vee x) \wedge y)$ is false.
- If formula $((b \rightarrow y) \wedge z)$ is true then formula $\neg((b \rightarrow y) \wedge z)$ is false.

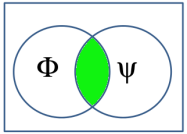
Conjunction

- a conjunction is true iff both arguments are true
- alternative notation for $\phi \wedge \psi$: $\phi\&\psi$, $\phi\psi$, $\phi * \psi$, $\phi \cdot \psi$, $\phi AND \psi$
- For $(\phi_1 \wedge \ldots \wedge \phi_n)$ we also write $\bigwedge_{i=1}^{n} \phi_i$.

truth table:

ϕ	ψ	$\phi \wedge \psi$
0	**0**	**0**
0	**1**	**0**
1	**0**	**0**
1	**1**	**1**

set view:



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.

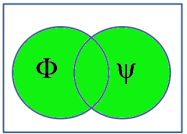
Disjunction

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

truth table:

ϕ	ψ	$\phi \vee \psi$
0	**0**	**0**
0	**1**	**1**
1	**0**	**1**
1	**1**	**1**

set view:



Example:

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

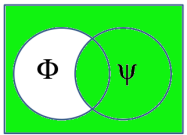
Implication

- an implication is true iff the first argument is false or both arguments are true (Ex falsum quodlibet.)
- alternative notation: $\phi \supset \psi, \phi$ IMPL ψ

truth table:

ϕ	ψ	$\phi \rightarrow \psi$
0	**0**	**1**
0	**1**	**1**
1	**0**	**0**
1	**1**	**1**

set view:



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 ϕ is true.

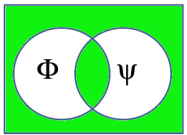
Equivalence

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

truth table:

ϕ	ψ	$\phi \leftrightarrow \psi$
0	**0**	**1**
0	**1**	**0**
1	**0**	**0**
1	**1**	**1**

set view:



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

ϕ	ψ	$\top$	$\bot$	$\neg\phi$	$\phi \wedge \psi$	$\phi \vee \psi$	$\phi \to \psi$	$\phi \leftrightarrow \psi$	$\phi \oplus \psi$	$\phi \uparrow \psi$	$\phi \downarrow \psi$
0	**0**	**1**	**0**	**1**	**0**	**0**	**1**	**1**	**0**	**1**	**1**
0	**1**	**1**	**0**	**1**	**0**	**1**	**1**	**0**	**1**	**1**	**0**
1	**0**	**1**	**0**	**0**	**0**	**1**	**0**	**0**	**1**	**1**	**0**
1	**1**	**1**	**0**	**0**	**1**	**1**	**1**	**1**	**0**	**0**	**0**

Example:

ϕ	ψ	$\neg(\neg\phi \wedge \neg\psi)$	$\neg\phi \vee \psi$	$(\phi \to \psi) \wedge (\psi \to \phi)$
0	**0**	**0**	**1**	**1**
0	**1**	**1**	**1**	**0**
1	**0**	**1**	**0**	**0**
1	**1**	**1**	**1**	**1**

Observation: connectives can be expressed by other connectives.

Other Connectives

- there are 16 different functions for binary connectives
- so far, we had $\wedge, \vee, \leftrightarrow, \rightarrow$
- further connectives:
  - $\phi \nleftrightarrow \psi$ (also $\oplus$, **xor**, antivalence)
  - $\phi \uparrow \psi$ (**nand**, Sheffer Stroke Function)
  - $\phi \downarrow \psi$ (**nor**, Pierce Function)

ϕ	ψ	$\phi \nleftrightarrow \psi$	$\phi \uparrow \psi$	$\phi \downarrow \psi$
0	**0**	**0**	**1**	**1**
0	**1**	**1**	**1**	**0**
1	**0**	**1**	**1**	**0**
1	**1**	**0**	**0**	**0**

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

Semantics of Propositional Logic

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 $[x]_\nu = \nu(x)$
- $[\neg\phi]_\nu = \mathbf{1}$ iff $[\phi]_\nu = \mathbf{0}$
- $[\phi \vee \psi]_\nu = \mathbf{1}$ iff $[\phi]_\nu = \mathbf{1}$ or $[\psi]_\nu = \mathbf{1}$

What about the other connectives?

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
3      pick v ∈ {⊤, ⊥}
4      /* replace x by truth constant v, evaluate resulting formula */
5      if evaluate(φ[x|v]) then return 1;
6      else return evaluate(φ[x|v̄]) ;
7  else
8      switch φ do
9          case ⊤ do return 1;
10         case ⊥ do return 0;
11         case ¬ψ do return ! evaluate(ψ)            /* true iff ψ is false */ ;
12         case ψ′ ∧ ψ′′ do
13             return evaluate(ψ′) && evaluate(ψ′′)    /* true iff both ψ′ and ψ′′ are true */
14         case ψ′ ∨ ψ′′ do
15             return evaluate(ψ′) || evaluate(ψ′′)    /* true iff ψ′ or ψ′′ is true */
```

Satisfying/Falsifying Assigments

- An assignment 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 $((x \wedge y) \vee \neg z)$,

- $\{\neg x, y, z\}$ is a counter-model,
- $\{x, y, z\}$ is a model.
- $\{x, y, \neg z\}$ is another model.

Properties of Propositional Formulas (1/3)

- formula ϕ is **satisfiable** iff
  there exists interpretation $[.]_\nu$ with $[\phi]_\nu = \mathbf{1}$
  check with `limboole -s`
- formula ϕ is **valid** iff
  for all interpretations $[.]_\nu$ it holds that $[\phi]_\nu = \mathbf{1}$
  check with `limboole`
- formula ϕ is **refutable** iff
  exists interpretation $[.]_\nu$ with $[\phi]_\nu = \mathbf{0}$
  check with `limboole`
- formula ϕ is **unsatisfiable** iff
  $[\phi]_\nu = \mathbf{0}$ 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.
- $(a \vee \neg b) \wedge (\neg a \vee b)$ is refutable.
- $a \rightarrow b$ is satisfiable.
- $a \leftrightarrow \neg a$ is a contradiction.
- $(a \vee \neg b) \wedge (\neg a \vee b)$ 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: $(a \vee b) \wedge (\neg a \vee c)$
- valid, not refutable $\top \vee (a \wedge \neg a)$; not valid, refutable $(\bot \vee b)$

Further Connections between Formulas

- A formula ϕ is valid iff $\neg\phi$ is unsatisfiable.
- A formula ϕ is satisfiable iff $\neg\phi$ is not valid.
- The formulas ϕ and ψ are equivalent iff $\phi \leftrightarrow \psi$ is valid.
- The formulas ϕ and ψ are equivalent iff $\neg(\phi \leftrightarrow \psi)$ is unsatisfiable.

PROPOSITIONAL FORMULAS: IMPORTANT EQUIVALENCES

VL Logic

Part I: Propositional Logic

JOHANNES KEPLER
UNIVERSITY LINZ

Semantic Equivalence

Two formulas ϕ and ψ are **semanticly equivalent** (written as $\phi \Leftrightarrow \psi$) iff forall interpretations $[.]_\nu$ it holds that $[\phi]_\nu = [\psi]_\nu$.

- $\Leftrightarrow$ is a **meta-symbol**, i.e., it is not part of the language.
- $\phi \Leftrightarrow \psi$ iff $\phi \leftrightarrow \psi$ is valid, i.e., we can express semantics by means of syntactics.
- If ϕ and ψ are not equivalent, we write $\phi \not\Leftrightarrow \psi$.

Example:

- $a \vee \neg a \not\Leftrightarrow b \rightarrow \neg b$
- $a \vee \neg a \Leftrightarrow b \vee \neg b$
- $(a \vee b) \wedge \neg(a \vee b) \Leftrightarrow \bot$
- $a \leftrightarrow (b \leftrightarrow c) \Leftrightarrow (a \leftrightarrow b) \leftrightarrow c$

Examples of Semantic Equivalences (1/2)

$\phi \wedge \psi \Leftrightarrow \psi \wedge \phi$	$\phi \vee \psi \Leftrightarrow \psi \vee \phi$	commutativity
$\phi \wedge (\psi \wedge \gamma) \Leftrightarrow (\phi \wedge \psi) \wedge \gamma$	$\phi \vee (\psi \vee \gamma) \Leftrightarrow (\phi \vee \psi) \vee \gamma$	associativity
$\phi \wedge (\phi \vee \psi) \Leftrightarrow \phi$	$\phi \vee (\phi \wedge \psi) \Leftrightarrow \phi$	absorption
$\phi \wedge (\psi \vee \gamma) \Leftrightarrow (\phi \wedge \psi) \vee (\phi \wedge \gamma)$	$\phi \vee (\psi \wedge \gamma) \Leftrightarrow (\phi \vee \psi) \wedge (\phi \vee \gamma)$	distributivity
$\neg(\phi \wedge \psi) \Leftrightarrow \neg\phi \vee \neg\psi$	$\neg(\phi \vee \psi) \Leftrightarrow \neg\phi \wedge \neg\psi$	laws of De Morgan
$\phi \leftrightarrow \psi \Leftrightarrow (\phi \rightarrow \psi) \wedge (\psi \rightarrow \phi)$	$\phi \leftrightarrow \psi \Leftrightarrow (\phi \wedge \psi) \vee (\neg\phi \wedge \neg\psi)$	synt. equivalence

Examples of Semantic Equivalences (2/2)

$\phi \vee \psi \Leftrightarrow \neg\phi \rightarrow \psi$	$\phi \rightarrow \psi \Leftrightarrow \neg\psi \rightarrow \neg\phi$	implications
$\phi \wedge \neg\phi \Leftrightarrow \bot$	$\phi \vee \neg\phi \Leftrightarrow \top$	complement
$\neg\neg\phi \Leftrightarrow \phi$		double negation
$\phi \wedge \top \Leftrightarrow \phi$	$\phi \vee \bot \Leftrightarrow \phi$	neutrality
$\phi \vee \top \Leftrightarrow \top$	$\phi \wedge \bot \Leftrightarrow \bot$	
$\neg\top \Leftrightarrow \bot$	$\neg\bot \Leftrightarrow \top$	