Propositional Logic

Dr Paolo Guagliardo



Fall 2020 (v20.1.0)

Logic in general

Logics are formal languages for

- representing what we know about the world
- reasoning about this knowledge (draw conclusions from it)

Two components:

Syntax defines the sentences in the language Semantics defines the **meaning** of the sentences

Used in many areas of Computer Science:

- ► Artificial Intelligence
- Semantic Web
- Software & Hardware verification
- Databases
- ... many many others

Many families of logics

There is not **one** logic, but many families of logics:

- ► Modal logics (epistemic, temporal, spatial, ...)
- Description logics
- ► Non-monotonic logics
- ► Intuitionistic logic
- ... many many others

We will study two classical logics:

- propositional logic
- ► first-order logic

Propositional logic: Building blocks

Propositions

Atomic statements that cannot be further decomposed

For example:

- "It is raining"
- "The cat is on the table"
- "The sky is blue"

Usually denoted with uppercase letters: P, Q, ...

Also called propositional variables

Logical connectives

- ► Conjunction: ∧ (and)
- ► Negation: ¬ (not)

Propositional logic: Syntax

Let **Prop** be a countable set of propositional variables

The language \mathcal{L} of propositional logic over **Prop** is defined inductively as follows:

- 1. Every $P \in \mathbf{Prop}$ is in \mathcal{L} (atomic formulae or atoms)
- 2. If ϕ and ψ are in \mathcal{L} , then $\phi \wedge \psi$ is also in \mathcal{L}
- 3. If ϕ is in \mathcal{L} , then also $\neg \phi$ is in \mathcal{L}
- 4. Nothing else is in \mathcal{L}

In other words, \mathcal{L} is generated by the following grammar:

$$\phi := P \mid \phi \land \phi \mid \neg \phi \qquad \text{(with } P \in \mathsf{Prop)}$$

Propositional logic: Semantics

Intuition

- ► Atomic statements can be either true (t) or false (f)
- ► The truth value of a formula is determined by the truth values of its atoms

Formally

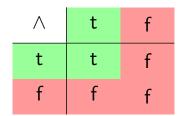
A truth-value assignment is a function $\alpha \colon \mathbf{Prop} \to \{\mathbf{t}, \mathbf{f}\}$

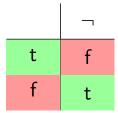
Then, α satisfies a formula ϕ (written $\alpha \models \phi$) inductively as follows:

$$\begin{array}{lll} \alpha \models P & \iff & \alpha(P) = \mathbf{t} \\ \alpha \models \neg \phi & \iff & \alpha \not\models \phi \\ \alpha \models \phi \land \psi & \iff & \alpha \models \phi \text{ and } \alpha \models \psi \end{array}$$

Truth tables

Reflect the semantics of the connectives





Given α and ϕ , allow us to determine whether $\alpha \models \phi$:

- 1. Replace each propositional variable P in ϕ by $\alpha(P)$
- 2. Propagate t and f according to the truth tables

Example

Given the formula

$$\phi = \neg(\neg A \land B) \land \neg(C \land \neg D)$$

and the assignment

$$\alpha = \{ A \mapsto \mathbf{t}, B \mapsto \mathbf{f}, C \mapsto \mathbf{f}, D \mapsto \mathbf{t} \}$$

does α satisfy ϕ ?

BLACKBOARD-TIME!

Given α and ϕ , checking whether $\alpha \models \phi$ can be done in **polynomial time** in the size of ϕ

Satisfiability and Validity

```
A formula \phi is satisfiable if there is some \alpha that satisfies \phi unsatisfiable if \phi is not satisfiable falsifiable if there is some \alpha that does not satisfy \phi valid if every \alpha satisfies \phi (in this case \phi is called a tautology)
```

Consequences

- $ightharpoonup \phi$ is a tautology iff $\neg \phi$ is unsatisfiable
- $\blacktriangleright \phi$ is unsatisfiable iff $\neg \phi$ is a tautology

Equivalence

Two formulas are logically equivalent (written $\phi \equiv \psi$) if for all α

$$\alpha \models \phi \iff \alpha \models \phi$$

(i.e., there is no assignment that satisfies one formula but not the other)

Intuition: Equivalent formulae have the same meaning even though they may differ syntactically (they say the same thing in different ways)

Propositional logic: Extended language

Syntax

New connectives:

- ▶ Disjunction ∨ (or)
- ► Implication → (if-then)
- ▶ Double implication ↔ (if and only if)

New grammar:

$$\phi := P \ | \ \neg \phi \ | \ \phi \land \phi \ | \ \phi \lor \phi \ | \ \phi \to \phi \ | \ \phi \leftrightarrow \phi$$

Semantics

$$\begin{array}{lll} \alpha \models \phi \lor \psi & \text{iff} & \alpha \models \phi \text{ or } \alpha \models \psi \\ \alpha \models \phi \to \psi & \text{iff} & \text{if } \alpha \models \phi \text{, then } \alpha \models \psi \\ \alpha \models \phi \leftrightarrow \psi & \text{iff} & (\alpha \models \phi \text{ if and only if } \alpha \models \psi) \end{array}$$

Expressive power

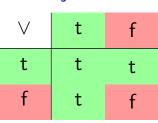
Every formula in the extended language can be **equivalently expressed** using only \wedge and \neg

The new connectives do not add expressive power to the language

- they are just syntactic sugar
- but useful to write formulae more succintly

Truth tables of the new connectives

Disjuction t



Implication

\rightarrow	t	f
t	t	f
f	t	t

Double Implication

\leftrightarrow	t	f
t	t	f
f	f	t

All of the above can be derived from the truth tables for \wedge and \neg

Equivalences

Commutativity	$\phi \vee \psi$	=	$\psi \lor \phi$
	$\phi \wedge \psi$	=	$\psi \wedge \phi$
Associativity	$(\phi \vee \psi) \vee \chi$	=	$\phi \vee (\psi \vee \chi)$
	$(\phi \wedge \psi) \wedge \chi$	=	$\phi \wedge (\psi \wedge \chi)$
Distributivity	$\phi \wedge (\psi \vee \chi)$	=	$(\phi \wedge \psi) \vee (\phi \wedge \chi)$
	$\phi \vee (\psi \wedge \chi)$	=	$(\phi \lor \psi) \land (\phi \lor \chi)$
Idempotence	$\phi \lor \phi$	=	ϕ
	$\phi \wedge \phi$	=	ϕ
Absorption	$\phi \lor (\phi \land \psi)$	=	ϕ
	$\phi \wedge (\phi \vee \psi)$	\equiv	ϕ

Equivalences (continued)

Implication

Double Negation
$$\neg\neg\phi \ \equiv \ \phi$$
 De Morgan
$$\neg(\phi\vee\psi) \ \equiv \ \neg\phi\wedge\neg\psi$$

$$\neg(\phi\wedge\psi) \ \equiv \ \neg\phi\vee\neg\psi$$
 Implication
$$\phi\to\psi \ \equiv \ \neg\phi\vee\psi$$

Entailment

We extend the satisfaction relationship \models to sets Σ of formulae:

$$\alpha \models \Sigma \quad \Longleftrightarrow \quad \alpha \models \phi \text{ for all } \phi \in \Sigma \quad \Longleftrightarrow \quad \alpha \models \bigwedge_{\phi \in \Sigma} \phi$$

Then, we say that Σ entails a formula ϕ if for all α

$$\alpha \models \phi$$
 whenever $\alpha \models \Sigma$

(i.e., every assignment that satisfies Σ also satisfies ϕ)

Properties of Entailment

Deduction Theorem

$$\Sigma \cup \{\phi\} \models \psi \quad \text{iff} \quad \Sigma \models \phi \to \psi$$

Contraposition Theorem

$$\Sigma \cup \{\phi\} \models \neg \psi \quad \text{iff} \quad \Sigma \cup \{\psi\} \models \neg \phi$$

Contradiction Theorem

$$\Sigma \cup \{\phi\} \text{ is unsatisfiable } \text{ iff } \Sigma \models \neg \phi$$

Decision problems

A decision problem

- takes (one or more) inputs
- answers a Yes / No question

Example

SAT: Given a propositional formula ϕ , is ϕ satisfiable? (i.e., can we find an assignment α such that $\alpha \models \phi$?)

A decision procedure is an algorithm that

- always terminates
- solves a decision problem

A decision problem is decidable if there is decision procedure for it

Solving SAT

Satisfiability in propositional logic is a decidable problem

Naive algorithm

- 1. Enumerate all possible assignments (there are 2^n where n is the number of atoms in the formula)
- 2. For each assignment check whether the formula is satisfied:
 - 2.1 if it is, stop and answer YES
 - 2.2 otherwise, continue to next assignment
- 3. If there are no more assignment, stop and answer NO

SAT is an **NP-complete** problem:

- ▶ a (candidate) solution can be verified in polynomial time
- no known efficient (polynomial) way to locate a solution (in the worst case, it requires exponential time)

Reduction to satisfiability

Validity, equivalence, and entailment can all be reduced to checking satisfiability:

- $ightharpoonup \phi$ is valid iff $\neg \phi$ is not satisfiable
- $ightharpoonup \phi \models \psi \text{ iff } \phi \rightarrow \psi \text{ is valid (deduction theorem)}$

A decision procedure for satisfiability is all we need

Validity, equivalence and entailment are decidable problems (in particular, coNP-complete) in propositional logic

Some additional terminology

Atom atomic formula

Literal atom (positive literal)
or its negation (negative literal)

Clause disjunction of literals

Term conjunction of literals

Normal forms

Formulae expressed in a standard syntactic form

Conjunctive Normal Form (CNF)

Conjunction of clauses: $\bigwedge_{i=1}^{n} (\bigvee_{j=1}^{m} L_{i,j})$

Example: $(A \lor \neg B) \land (B \lor \neg C \lor \neg D)$

Disjunctive Normal Form (DNF)

Disjunction of terms: $\bigvee_{i=1}^{n} (\bigwedge_{j=1}^{m} L_{i,j})$

Example: $(A \wedge B) \vee (A \wedge \neg C) \vee (A \wedge \neg D) \vee (\neg B \wedge \neg C) \vee (\neg B \wedge \neg D)$

Negation Normal Form (NNF)

Only \land , \lor , \neg and negation can appear only in front of atoms (in particular, every formula in CNF or DNF is also in NNF)

Example: $(A \land (B \lor \neg C \lor \neg D)) \lor (\neg B \land (B \lor \neg C \lor \neg D))$

Converting to NNF, CNF and DNF

For every formula there exist **equivalent** formulae in CNF, DNF and NNF

To convert into NNF

- 1. Replace $\{\rightarrow,\leftrightarrow\}$ by $\{\land,\lor,\lnot\}$
- 2. Apply De Morgan's laws

To convert into CNF or DNF

- 1. Convert into NNF
- 2. Apply the distributivity laws

Why normal forms?

Some approaches to inference use syntactic operations on formulae, often expressed in standardized forms

CNF tells us something as to whether a formula is valid:

- ► If all clauses contain complementary literals, then the formula is a tautology
- Otherwise, the formula is falsifiable

DNF tells us something as to whether a formula is satisfiable:

- ► If all terms contain complementary literals, then the formula is unsatisfiable
- ▶ Otherwise, the formula is satisfiable

DNF and Satisfiability

Whether a formula in DNF is satisfiable can be decided in **linear time** in the size of the formula

Wait! Didn't we say that SAT is NP-complete? Yes

So what's the catch with DNF?

The transformation into DNF is expensive (in time/space)

► The size of the formula may blow-up exponentially!