

Theory of Computer Science

B2. Propositional Logic II

Gabriele Röger

University of Basel

February 27, 2019

Theory of Computer Science

February 27, 2019 — B2. Propositional Logic II

B2.1 Simplified Notation

B2.2 Normal Forms

B2.3 Logical Consequences

B2.4 Summary

B2.1 Simplified Notation

Parentheses

Associativity:

$$((\varphi \wedge \psi) \wedge \chi) \equiv (\varphi \wedge (\psi \wedge \chi))$$

$$((\varphi \vee \psi) \vee \chi) \equiv (\varphi \vee (\psi \vee \chi))$$

- ▶ Placement of parentheses for a conjunction of conjunctions does not influence whether an interpretation is a model.
- ▶ ditto for disjunctions of disjunctions
- can omit parentheses and treat this as if parentheses placed arbitrarily
- ▶ Example: $(A_1 \wedge A_2 \wedge A_3 \wedge A_4)$ instead of $((A_1 \wedge (A_2 \wedge A_3)) \wedge A_4)$
- ▶ Example: $(\neg A \vee (B \wedge C) \vee D)$ instead of $((\neg A \vee (B \wedge C)) \vee D)$

Parentheses

Does this mean we can always omit all parentheses and assume an arbitrary placement? → **No!**

$$((\varphi \wedge \psi) \vee \chi) \not\equiv (\varphi \wedge (\psi \vee \chi))$$

What should $\varphi \wedge \psi \vee \chi$ mean?

Placement of Parentheses by Convention

Often parentheses can be dropped in specific cases and an **implicit** placement is assumed:

- ▶ \neg binds more strongly than \wedge
- ▶ \wedge binds more strongly than \vee
- ▶ \vee binds more strongly than \rightarrow or \leftrightarrow

→ cf. PEMDAS/“Punkt vor Strich”

Example

$A \vee \neg C \wedge B \rightarrow A \vee \neg D$ stands for $((A \vee (\neg C \wedge B)) \rightarrow (A \vee \neg D))$

- ▶ often harder to read
- ▶ error-prone
- not used in this course

Short Notations for Conjunctions and Disjunctions

Short notation for addition:

$$\sum_{i=1}^n x_i = x_1 + x_2 + \dots + x_n$$

$$\sum_{x \in \{x_1, \dots, x_n\}} x = x_1 + x_2 + \dots + x_n$$

Analogously:

$$\left(\bigwedge_{i=1}^n \varphi_i\right) = (\varphi_1 \wedge \varphi_2 \wedge \dots \wedge \varphi_n)$$

$$\left(\bigvee_{i=1}^n \varphi_i\right) = (\varphi_1 \vee \varphi_2 \vee \dots \vee \varphi_n)$$

$$\left(\bigwedge_{\varphi \in X} \varphi\right) = (\varphi_1 \wedge \varphi_2 \wedge \dots \wedge \varphi_n)$$

$$\left(\bigvee_{\varphi \in X} \varphi\right) = (\varphi_1 \vee \varphi_2 \vee \dots \vee \varphi_n)$$

for $X = \{\varphi_1, \dots, \varphi_n\}$

Short Notation: Corner Cases

Is $\mathcal{I} \models \psi$ true for

$$\psi = \left(\bigwedge_{\varphi \in X} \varphi\right) \text{ and } \psi = \left(\bigvee_{\varphi \in X} \varphi\right)$$

if $X = \emptyset$ or $X = \{\chi\}$?

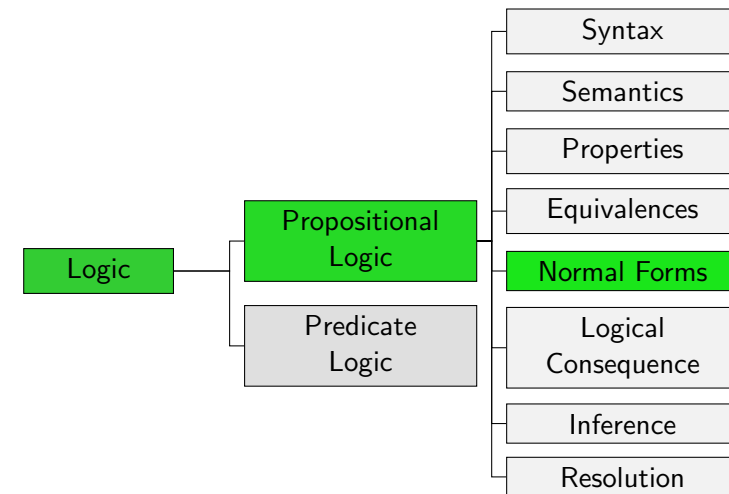
convention:

- ▶ $\left(\bigwedge_{\varphi \in \emptyset} \varphi\right)$ is tautology.
- ▶ $\left(\bigvee_{\varphi \in \emptyset} \varphi\right)$ is unsatisfiable.
- ▶ $\left(\bigwedge_{\varphi \in \{\chi\}} \varphi\right) = \left(\bigvee_{\varphi \in \{\chi\}} \varphi\right) = \chi$

↪ Why?

B2.2 Normal Forms

Logic: Overview



Why Normal Forms?

- ▶ A **normal form** is a representation with **certain syntactic restrictions**.
- ▶ condition for reasonable normal form: **every formula** must have a logically **equivalent formula in normal form**
- ▶ **advantages**:
 - ▶ can restrict proofs to formulas in normal form
 - ▶ can define algorithms only for formulas in normal form

German: Normalform

Literals, Clauses and Monomials

- ▶ A **literal** is an atomic proposition or the negation of an atomic proposition (e. g., A and $\neg A$).
- ▶ A **clause** is a disjunction of literals (e. g., $(Q \vee \neg P \vee \neg S \vee R)$).
- ▶ A **monomial** is a conjunction of literals (e. g., $(Q \wedge \neg P \wedge \neg S \wedge R)$).

The terms **clause** and **monomial** are also used for the corner case with **only one literal**.

German: Literal, Klausel, Monom

Terminology: Examples

Examples

- ▶ $(\neg Q \wedge R)$ is a monomial
- ▶ $(P \vee \neg Q)$ is a clause
- ▶ $((P \vee \neg Q) \wedge P)$ is neither literal nor clause nor monomial
- ▶ $\neg P$ is a literal, a clause and a monomial
- ▶ $(P \rightarrow Q)$ is neither literal nor clause nor monomial
(but $(\neg P \vee Q)$ is a clause!)
- ▶ $(P \vee P)$ is a clause, but not a literal or monomial
- ▶ $\neg\neg P$ is neither literal nor clause nor monomial

Conjunctive Normal Form

Definition (Conjunctive Normal Form)

A formula is in **conjunctive normal form (CNF)** if it is a conjunction of clauses, i. e., if it has the form

$$\left(\bigwedge_{i=1}^n \left(\bigvee_{j=1}^{m_i} L_{ij} \right) \right)$$

with $n, m_i > 0$ (for $1 \leq i \leq n$), where the L_{ij} are literals.

German: konjunktive Normalform (KNF)

Example

$((\neg P \vee Q) \wedge R \wedge (P \vee \neg S))$ is in CNF.

Disjunctive Normal Form

Definition (Disjunctive Normal Form)

A formula is in **disjunctive normal form (DNF)** if it is a disjunction of monomials, i. e., if it has the form

$$\left(\bigvee_{i=1}^n \left(\bigwedge_{j=1}^{m_i} L_{ij} \right) \right)$$

with $n, m_i > 0$ (for $1 \leq i \leq n$), where the L_{ij} are literals.

German: disjunktive Normalform (DNF)

Example

$((\neg P \wedge Q) \vee R \vee (P \wedge \neg S))$ is in DNF.

CNF and DNF: Examples

Examples

- ▶ $((P \vee \neg Q) \wedge P)$ is in CNF
- ▶ $((R \vee Q) \wedge P \wedge (R \vee S))$ is in CNF
- ▶ $(P \vee (\neg Q \wedge R))$ is in DNF
- ▶ $((P \vee \neg Q) \rightarrow P)$ is neither in CNF nor in DNF
- ▶ P is in CNF and in DNF

Construction of CNF (and DNF)

Algorithm to Construct CNF

- 1 Replace abbreviations \rightarrow and \leftrightarrow by their definitions (\rightarrow -elimination and \leftrightarrow -elimination).
 \rightsquigarrow formula structure: only \vee, \wedge, \neg
- 2 Move negations inside using De Morgan and double negation.
 \rightsquigarrow formula structure: only \vee, \wedge , literals
- 3 Distribute \vee over \wedge with distributivity (strictly speaking also with commutativity).
 \rightsquigarrow formula structure: CNF
- 4 optionally: Simplify the formula at the end or at intermediate steps (e. g., with idempotence).

Note: For DNF, distribute \wedge over \vee instead.

Constructing CNF: Example

Construction of Conjunctive Normal Form

Given: $\varphi = (((P \wedge \neg Q) \vee R) \rightarrow (P \vee \neg(S \vee T)))$

$$\begin{aligned} \varphi &\equiv (\neg((P \wedge \neg Q) \vee R) \vee P \vee \neg(S \vee T)) && \text{[Step 1]} \\ &\equiv ((\neg(P \wedge \neg Q) \wedge \neg R) \vee P \vee \neg(S \vee T)) && \text{[Step 2]} \\ &\equiv (((\neg P \vee \neg\neg Q) \wedge \neg R) \vee P \vee \neg(S \vee T)) && \text{[Step 2]} \\ &\equiv (((\neg P \vee Q) \wedge \neg R) \vee P \vee \neg(S \vee T)) && \text{[Step 2]} \\ &\equiv (((\neg P \vee Q) \wedge \neg R) \vee P \vee (\neg S \wedge \neg T)) && \text{[Step 2]} \\ &\equiv ((\neg P \vee Q \vee P \vee (\neg S \wedge \neg T)) \wedge \\ &\quad (\neg R \vee P \vee (\neg S \wedge \neg T))) && \text{[Step 3]} \\ &\equiv (\neg R \vee P \vee (\neg S \wedge \neg T)) && \text{[Step 4]} \\ &\equiv ((\neg R \vee P \vee \neg S) \wedge (\neg R \vee P \vee \neg T)) && \text{[Step 3]} \end{aligned}$$

Construct DNF: Example

Construction of Disjunctive Normal Form

Given: $\varphi = (((P \wedge \neg Q) \vee R) \rightarrow (P \vee \neg(S \vee T)))$

$$\begin{aligned} \varphi &\equiv (\neg((P \wedge \neg Q) \vee R) \vee P \vee \neg(S \vee T)) && \text{[Step 1]} \\ &\equiv ((\neg(P \wedge \neg Q) \wedge \neg R) \vee P \vee \neg(S \vee T)) && \text{[Step 2]} \\ &\equiv (((\neg P \vee \neg\neg Q) \wedge \neg R) \vee P \vee \neg(S \vee T)) && \text{[Step 2]} \\ &\equiv (((\neg P \vee Q) \wedge \neg R) \vee P \vee \neg(S \vee T)) && \text{[Step 2]} \\ &\equiv (((\neg P \vee Q) \wedge \neg R) \vee P \vee (\neg S \wedge \neg T)) && \text{[Step 2]} \\ &\equiv ((\neg P \wedge \neg R) \vee (Q \wedge \neg R) \vee P \vee (\neg S \wedge \neg T)) && \text{[Step 3]} \end{aligned}$$

Existence of an Equivalent Formula in Normal Form

Theorem

For every formula φ there is a logically equivalent formula in CNF and a logically equivalent formula in DNF.

- ▶ “There is a” always means “there is at least one”. Otherwise we would write “there is exactly one”.
- ▶ Intuition: algorithm to construct normal form works with any given formula and only uses equivalence rewriting.
- ▶ actual proof would use induction over structure of formula

Size of Normal Forms

- ▶ In the worst case, a logically equivalent formula in CNF or DNF can be exponentially larger than the original formula.
- ▶ **Example:** for $(x_1 \vee y_1) \wedge \dots \wedge (x_n \vee y_n)$ there is no smaller logically equivalent formula in DNF than:

$$\bigvee_{S \in \mathcal{P}(\{1, \dots, n\})} \left(\bigwedge_{i \in S} x_i \wedge \bigwedge_{i \in \{1, \dots, n\} \setminus S} y_i \right)$$

- ▶ As a consequence, the construction of the CNF/DNF formula can take exponential time.

More Theorems

Theorem

A formula in CNF is a tautology iff every clause is a tautology.

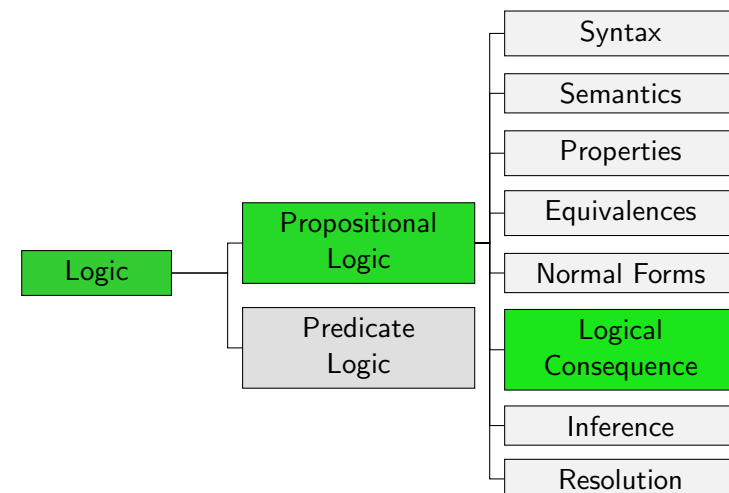
Theorem

A formula in DNF is satisfiable iff at least one of its monomials is satisfiable.

\rightsquigarrow both proved easily with semantics of propositional logic

B2.3 Logical Consequences

Logic: Overview



Knowledge Bases: Example



If not DrinkBeer, then EatFish.
 If EatFish and DrinkBeer,
 then not EatIceCream.
 If EatIceCream or not DrinkBeer,
 then not EatFish.

$$\text{KB} = \{(\neg\text{DrinkBeer} \rightarrow \text{EatFish}), \\ ((\text{EatFish} \wedge \text{DrinkBeer}) \rightarrow \neg\text{EatIceCream}), \\ ((\text{EatIceCream} \vee \neg\text{DrinkBeer}) \rightarrow \neg\text{EatFish})\}$$

Exercise from U. Schöning: Logik für Informatiker
 Picture courtesy of graur razvan ionut / FreeDigitalPhotos.net

Models for Sets of Formulas

Definition (Model for Knowledge Base)

Let KB be a **knowledge base** over A ,
 i. e., a set of propositional formulas over A .

A truth assignment \mathcal{I} for A is a **model for KB** (written: $\mathcal{I} \models \text{KB}$)
 if \mathcal{I} is a **model for every formula** $\varphi \in \text{KB}$.

German: Wissensbasis, Modell

Properties of Sets of Formulas

A knowledge base KB is

- ▶ **satisfiable** if KB has at least one model
- ▶ **unsatisfiable** if KB is not satisfiable
- ▶ **valid** (or a **tautology**) if every interpretation is a model for KB
- ▶ **falsifiable** if KB is no tautology

German: erfüllbar, unerfüllbar, gültig, gültig/eine Tautologie,
 falsifizierbar

Example I

Which of the properties does $\text{KB} = \{(A \wedge \neg B), \neg(B \vee A)\}$ have?

KB is **unsatisfiable**:

For every model \mathcal{I} with $\mathcal{I} \models (A \wedge \neg B)$ we have $\mathcal{I}(A) = 1$.
 This means $\mathcal{I} \models (B \vee A)$ and thus $\mathcal{I} \not\models \neg(B \vee A)$.

This directly implies that KB is **falsifiable**, **not satisfiable**
 and **no tautology**.

Example II

Which of the properties does

$$\text{KB} = \{(\neg\text{DrinkBeer} \rightarrow \text{EatFish}), \\ ((\text{EatFish} \wedge \text{DrinkBeer}) \rightarrow \neg\text{EatIceCream}), \\ ((\text{EatIceCream} \vee \neg\text{DrinkBeer}) \rightarrow \neg\text{EatFish})\}$$
 have?

- ▶ **satisfiable**, e. g. with $\mathcal{I} = \{\text{EatFish} \mapsto 1, \text{DrinkBeer} \mapsto 1, \text{EatIceCream} \mapsto 0\}$
- ▶ thus **not unsatisfiable**
- ▶ **falsifiable**, e. g. with $\mathcal{I} = \{\text{EatFish} \mapsto 0, \text{DrinkBeer} \mapsto 0, \text{EatIceCream} \mapsto 1\}$
- ▶ thus **not valid**

Logical Consequences: Motivation

What's the secret of your long life?



I am on a strict diet: If I don't drink beer to a meal, then I always eat fish. Whenever I have fish and beer with the same meal, I abstain from ice cream. When I eat ice cream or don't drink beer, then I never touch fish.

Claim: the woman drinks beer to every meal.

How can we prove this?

Exercise from U. Schöning: Logik für Informatiker
Picture courtesy of graur razvan ionut/FreeDigitalPhotos.net

Logical Consequences

Definition (Logical Consequence)

Let KB be a set of formulas and φ a formula.

We say that KB **logically implies** φ (written as $\text{KB} \models \varphi$) if **all models** of KB are also models of φ .

also: KB **logically entails** φ , φ **logically follows** from KB , φ is a **logical consequence** of KB

German: KB impliziert φ logisch, φ folgt logisch aus KB , φ ist logische Konsequenz von KB

Attention: the symbol \models is "overloaded": $\text{KB} \models \varphi$ vs. $\mathcal{I} \models \varphi$.

What if KB is unsatisfiable or the empty set?

Logical Consequences: Example

Let $\varphi = \text{DrinkBeer}$ and

$$\text{KB} = \{(\neg\text{DrinkBeer} \rightarrow \text{EatFish}), \\ ((\text{EatFish} \wedge \text{DrinkBeer}) \rightarrow \neg\text{EatIceCream}), \\ ((\text{EatIceCream} \vee \neg\text{DrinkBeer}) \rightarrow \neg\text{EatFish})\}.$$

Show: $\text{KB} \models \varphi$

Proof sketch.

Proof by contradiction: assume $\mathcal{I} \models \text{KB}$, but $\mathcal{I} \not\models \text{DrinkBeer}$.

Then it follows that $\mathcal{I} \models \neg\text{DrinkBeer}$.

Because \mathcal{I} is a model of KB , we also have

$\mathcal{I} \models (\neg\text{DrinkBeer} \rightarrow \text{EatFish})$ and thus $\mathcal{I} \models \text{EatFish}$. (Why?)

With an analogous argumentation starting from $\mathcal{I} \models ((\text{EatIceCream} \vee \neg\text{DrinkBeer}) \rightarrow \neg\text{EatFish})$

we get $\mathcal{I} \models \neg\text{EatFish}$ and thus $\mathcal{I} \not\models \text{EatFish}$. \rightsquigarrow **Contradiction!**

Important Theorems about Logical Consequences

Theorem (Deduction Theorem)

$KB \cup \{\varphi\} \models \psi$ iff $KB \models (\varphi \rightarrow \psi)$

German: Deduktionssatz

Theorem (Contraposition Theorem)

$KB \cup \{\varphi\} \models \neg\psi$ iff $KB \cup \{\psi\} \models \neg\varphi$

German: Kontrapositionssatz

Theorem (Contradiction Theorem)

$KB \cup \{\varphi\}$ is *unsatisfiable* iff $KB \models \neg\varphi$

German: Widerlegungssatz

(without proof)

B2.4 Summary

Summary

- ▶ **CNF**: formula is a conjunction of clauses
- ▶ **DNF**: formula is a disjunction of monomials
- ▶ every formula has **equivalent formulas in DNF and in CNF**
- ▶ **knowledge base**: set of formulas describing given information; satisfiable, valid etc. used like for individual formulas
- ▶ **logical consequence** $KB \models \varphi$ means that φ is true whenever (= in all models where) KB is true