

Propositional Logic

Computation and propositional tableau

Mario Alviano

University of Calabria, Italy

A.Y. 2018/2019

- 1 Computation
 - Satisfiability
 - The need for syntactic methods
 - Theoretical foundation
- 2 Normal forms (first part)
 - Why normal forms?
 - Negation Normal Form
- 3 Propositional Tableau
- 4 Exercises

Too many (computational) problems!

- Validity, equivalence, entailment, satisfiability, unsatisfiability, . . .
- How many algorithms do we need?
- One for each of these computational problems?

Let us reduce all of them to (un)satisfiability!

Observation

A set Γ of wffs is unsatisfiable iff $\Gamma \models \perp$

Observation

ϕ is valid iff

- $\models \phi$ iff

- $\models \top \rightarrow \phi$ iff

(apply Contraposition Theorem)

- $\neg\phi \models \neg\top$ iff

- $\neg\phi \models \perp$

How to check whether ϕ is valid?

- Check whether $\neg\phi$ is unsatisfiable

- If yes then ϕ is valid

- If no then ϕ is not valid

Observation

$\phi \equiv \psi$ holds iff

- $\models \phi \leftrightarrow \psi$ iff
- $\models \top \rightarrow (\phi \leftrightarrow \psi)$ iff (apply Contraposition Theorem)
- $\neg(\phi \leftrightarrow \psi) \models \neg\top$ iff
- $\neg(\phi \leftrightarrow \psi) \models \perp$

How to check whether $\phi \equiv \psi$ holds?

- Check whether $\neg(\phi \leftrightarrow \psi)$ is unsatisfiable
 - If yes then $\phi \equiv \psi$ holds
 - If no then $\phi \equiv \psi$ does not hold

Observation

$\Gamma \models \phi$ holds iff

- $\Gamma \models \top \rightarrow \phi$ iff (apply Contraposition Theorem)
- $\Gamma, \neg\phi \models \neg\top$ iff
- $\Gamma, \neg\phi \models \perp$

How to check whether $\Gamma \models \phi$ holds?

- Check whether $\Gamma \cup \{\neg\phi\}$ is unsatisfiable
 - If yes then $\Gamma \models \phi$ holds
 - If no then $\Gamma \models \phi$ does not hold

Satisfiability by truth tables

- Validity, equivalence, entailment, ...
- Can be reduced to satisfiability testing (SAT)

Algorithm: SAT by truth table

Input : a set Γ of wffs

Output: true if Γ is SAT; false otherwise

```
1 begin
2   foreach interpretation I do
3     if evaluate( $\Gamma, I$ ) then
4       return true;
5   return false
```

-
- Simple method
 - Usually quite inefficient
 - Works on the semantic level (by trying interpretations)

Satisfiability: Complexity

Theorem (Cook 1971)

Satisfiability of wffs is NP-complete

- First NP-completeness proof ever!

Membership

- 1 Guess an interpretation I
(from 2^n possibilities, for n variables)
- 2 Verify in polynomial time that $I \models \phi$

Hardness

- Simulate nondeterministic Turing machine with polynomial time bound

- Cook's Theorem implies that very efficient methods are unlikely to exist
- But one can try to be as efficient as possible!

- Sequent calculus
- Tableaux
- Resolution
- Davis-Putnam-Logemann-Loveland (DPLL) algorithm
- Conflict-driven clause learning (CDCL) algorithm

They require input in some normal form

Observation

For all unsatisfiable sets Γ of wffs:

$$\Gamma \equiv \perp$$

- Can we find a transformation which takes Γ to \perp iff Γ is unsatisfiable?
- Can we use a normal form?

Goal

- 1 Start with any set Γ of wffs
- 2 Transform Γ to Γ' in normal form (possibly in linear time)
- 3 Transform Γ' to \perp (if possible, i.e., if Γ is unsatisfiable)

Formula substitution

- In the previous slides we replaced formulas with equivalent formulas
- Can we do the same for subformulas?
- Can we replace all occurrences of subformula with a different (hopefully simpler) subformula?

Definition

For wffs γ, ϕ and variable A , let $\gamma[\phi/A]$ denote the formula in which ϕ replaces all occurrences of A

Example

Consider $\gamma = \neg A \vee (A \wedge B)$

- $\phi = B \rightarrow C$ $\gamma[\phi/A] = \neg(B \rightarrow C) \vee ((B \rightarrow C) \wedge B)$
- $\psi = \neg B \vee C$ $\gamma[\psi/A] = \neg(\neg B \vee C) \vee ((\neg B \vee C) \wedge B)$

Substitution Theorem

Example

Consider $\gamma = \neg A \vee (A \wedge B)$

■ $\phi = B \rightarrow C$ $\gamma[\phi/A] = \neg(B \rightarrow C) \vee ((B \rightarrow C) \wedge B)$

■ $\psi = \neg B \vee C$ $\gamma[\psi/A] = \neg(\neg B \vee C) \vee ((\neg B \vee C) \wedge B)$

■ Since $B \rightarrow C \equiv \neg B \vee C$ we have

$$\neg(B \rightarrow C) \vee ((B \rightarrow C) \wedge B) \equiv \neg(\neg B \vee C) \vee ((\neg B \vee C) \wedge B)$$

Theorem

For wffs ϕ, ψ and variable A , if $\phi \equiv \psi$ then for each wff γ we have $\gamma[\phi/A] \equiv \gamma[\psi/A]$

Proof.

By induction on the formula structure
(shown on the blackboard)

Simplify formulas

- Too many connectives! $\top, \perp, \neg, \wedge, \vee, \rightarrow, \leftrightarrow$
 - Can we find a **subset of connectives** C , such that any wff is equivalent to a formula with connectives of C ?
- Too much heterogeneous structure!
 - Can we find a **formula structure**, such that any wff is equivalent to a formula of this structure?

Eliminating connectives

Consider only \neg, \wedge, \vee !

- $\top \equiv \neg A \vee A$ (we need at least one variable in V for this!)
- $\perp \equiv \neg A \wedge A$ (we need at least one variable in V for this!)
- $\phi \rightarrow \psi \equiv \neg\phi \vee \psi$
- $\phi \leftrightarrow \psi \equiv (\phi \rightarrow \psi) \wedge (\psi \rightarrow \phi) \equiv (\neg\phi \vee \psi) \wedge (\neg\psi \vee \phi)$
- Use repeated formula substitution to eliminate $\top, \perp, \rightarrow, \leftrightarrow$

Every formula is equivalent to one containing only connectives \neg, \wedge, \vee

1 Negation Normal Form (NNF)

- Negation only in front of propositional variables

2 Conjunctive Normal Form (CNF)

- $C_1 \wedge \dots \wedge C_n$
- Each **conjunct** or **clause** C_i is of the form $L_1 \vee \dots \vee L_{m_i}$
- Each **literal** L_j is a variable A or the negation of a variable $\neg A$

3 Disjunctive Normal Form (DNF)

- $D_1 \vee \dots \vee D_n$
- Each **disjunct** D_i is of the form $L_1 \wedge \dots \wedge L_{m_i}$
- Each **literal** L_j is a variable A or the negation of a variable $\neg A$

NNF transformation

- 1 Eliminate \top , \perp , \rightarrow , \leftrightarrow
- 2 Apply double negation equivalences:
 - $\neg\neg\phi \equiv \phi$
- 3 Apply De Morgan equivalences:
 - $\neg(\phi \vee \psi) \equiv \neg\psi \wedge \neg\phi$
 - $\neg(\phi \wedge \psi) \equiv \neg\psi \vee \neg\phi$

Order of 2-3 does not matter!

$$\neg((A \rightarrow B) \wedge (B \leftrightarrow C))$$

$$1 \quad \neg((A \rightarrow B) \wedge (B \leftrightarrow C)) \equiv \neg((\neg A \vee B) \wedge (B \leftrightarrow C))$$

$$1 \quad \neg((\neg A \vee B) \wedge (B \leftrightarrow C)) \equiv \neg((\neg A \vee B) \wedge ((\neg B \vee C) \wedge (\neg C \vee B)))$$

$$3 \quad \neg((\neg A \vee B) \wedge ((\neg B \vee C) \wedge (\neg C \vee B))) \equiv \\ \neg(\neg A \vee B) \vee \neg((\neg B \vee C) \wedge (\neg C \vee B))$$

$$3 \quad \neg(\neg A \vee B) \vee \neg((\neg B \vee C) \wedge (\neg C \vee B)) \equiv \\ (\neg\neg A \wedge \neg B) \vee \neg((\neg B \vee C) \wedge (\neg C \vee B))$$

$$2 \quad (\neg\neg A \wedge \neg B) \vee \neg((\neg B \vee C) \wedge (\neg C \vee B)) \equiv \\ (A \wedge \neg B) \vee \neg((\neg B \vee C) \wedge (\neg C \vee B))$$

$$3 \quad (A \wedge \neg B) \vee \neg((\neg B \vee C) \wedge (\neg C \vee B)) \equiv \\ (A \wedge \neg B) \vee (\neg(\neg B \vee C) \vee \neg(\neg C \vee B))$$

$$\begin{aligned} \text{3} \quad & (A \wedge \neg B) \vee (\neg(\neg B \vee C) \vee \neg(\neg C \vee B)) \equiv \\ & (A \wedge \neg B) \vee ((\neg\neg B \wedge \neg C) \vee \neg(\neg C \vee B)) \end{aligned}$$

$$\begin{aligned} \text{2} \quad & (A \wedge \neg B) \vee ((\neg\neg B \wedge \neg C) \vee \neg(\neg C \vee B)) \equiv \\ & (A \wedge \neg B) \vee ((B \wedge \neg C) \vee \neg(\neg C \vee B)) \end{aligned}$$

$$\begin{aligned} \text{3} \quad & (A \wedge \neg B) \vee ((B \wedge \neg C) \vee \neg(\neg C \vee B)) \equiv \\ & (A \wedge \neg B) \vee ((B \wedge \neg C) \vee (\neg\neg C \wedge \neg B)) \end{aligned}$$

$$\begin{aligned} \text{2} \quad & (A \wedge \neg B) \vee ((B \wedge \neg C) \vee (\neg\neg C \wedge \neg B)) \equiv \\ & (A \wedge \neg B) \vee ((B \wedge \neg C) \vee (C \wedge \neg B)) \end{aligned}$$

Observation

With the exception of **1**, we are reducing the size of the formula

- A decision procedure for unsatisfiability
- The idea is to break down logical connectives into their constituent parts
- We consider the tableau for formulas in Negation Normal Form (NNF)

Negation Normal Form

- 1 Eliminate \top , \perp , \rightarrow , \leftrightarrow
- 2 Apply double negation equivalences:
 - $\neg\neg\phi \equiv \phi$
- 3 Apply De Morgan equivalences:
 - $\neg(\phi \vee \psi) \equiv \neg\psi \wedge \neg\phi$
 - $\neg(\phi \wedge \psi) \equiv \neg\psi \vee \neg\phi$

Let Γ be a set of wffs in NNF.

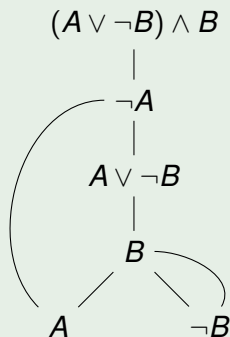
- 1 Start with a chain made of the formulas in Γ
- 2 If a branch contains a formula $\phi \wedge \psi$ then add to its leaf the chain $\phi \text{ — } \psi$
- 3 If a branch contains a formula $\phi \vee \psi$ then add to its leaf two children ϕ and ψ
- 4 If a branch contains a formula and its negation then close the branch
- 5 If all branches are closed then Γ is unsatisfiable

Notation

$$\frac{\phi \wedge \psi}{\begin{array}{c} \phi \\ \psi \end{array}} (\wedge)$$

$$\frac{\phi \vee \psi}{\phi \mid \psi} (\vee)$$

Example: $\Gamma := \{(A \vee \neg B) \wedge B, \neg A\}$



Γ is unsatisfiable!

We can simplify the notation used for a closure by putting a marker on the leaf of the closed path!

On the blackboard

$A \wedge C, \neg A \vee B$

Theorem

Propositional tableau is **sound** and **complete**, i.e.,
 if the tableau for Γ is closed then $\Gamma \models \perp$, and
 if $\Gamma \models \perp$ then the tableau for Γ is closed.

(From *Logic for Computer Science: Foundations of Automatic Theorem Proving*)

Give proof trees (by means of tableau) for the following tautologies:

1 $A \rightarrow (B \rightarrow A)$

2 $(A \rightarrow B) \rightarrow ((A \rightarrow (B \rightarrow C)) \rightarrow (A \rightarrow C))$

3 $A \rightarrow (B \rightarrow A \wedge B)$

4 $A \rightarrow A \vee B$

5 $B \rightarrow A \vee B$

6 $(A \rightarrow B) \rightarrow ((A \rightarrow \neg B) \rightarrow \neg A)$

7 $A \wedge B \rightarrow A$

8 $A \wedge B \rightarrow B$

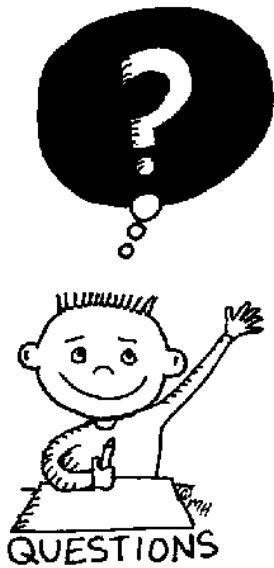
9 $(A \rightarrow C) \rightarrow ((B \rightarrow C) \rightarrow (A \vee B \rightarrow C))$

10 $\neg\neg A \rightarrow A$

(From *Logic for Computer Science: Foundations of Automatic Theorem Proving*)

Give proof trees (by means of tableau) for the following equivalences:

- 1 $(A \vee B) \vee C \equiv A \vee (B \vee C)$ (associativity)
- 2 $(A \wedge B) \wedge C \equiv A \wedge (B \wedge C)$ (associativity)
- 3 $A \vee B \equiv B \vee A$ (commutativity)
- 4 $A \wedge B \equiv B \wedge A$ (commutativity)
- 5 $A \vee (B \wedge C) \equiv (A \vee B) \wedge (A \vee C)$ (distributivity)
- 6 $A \wedge (B \vee C) \equiv (A \wedge B) \vee (A \wedge C)$ (distributivity)
- 7 $\neg(A \vee B) \equiv \neg A \wedge \neg B$ (De Morgan)
- 8 $\neg(A \wedge B) \equiv \neg A \vee \neg B$ (De Morgan)
- 9 $A \vee A \equiv A$ (idempotency)
- 10 $A \wedge A \equiv A$ (idempotency)
- 11 $\neg\neg A \equiv A$ (double negation)
- 12 $(A \vee B) \wedge (\neg A \vee C) \equiv (A \vee B) \wedge (\neg A \vee C) \wedge (B \vee C)$ (resolution)



END OF THE
LECTURE