

# Satisfiability Checking

## Propositional Logic

Prof. Dr. Erika Ábrahám

RWTH Aachen University  
Informatik 2  
LuFG Theory of Hybrid Systems

WS 14/15

The slides are partly taken from:

[www.decision-procedures.org/slides/](http://www.decision-procedures.org/slides/)

- Syntax of propositional logic
- Semantics of propositional logic
- Satisfiability and validity
- Modeling with propositional logic
- Normal forms
- Enumeration and deduction

- Syntax of propositional logic
- Semantics of propositional logic
- Satisfiability and validity
- Modeling with propositional logic
- Normal forms
- Enumeration and deduction

# Syntax of propositional logic

**Abstract syntax** of well-formed propositional formulae:

$$\varphi := a \mid (\neg\varphi) \mid (\varphi \wedge \varphi)$$

where Prop is a set of (atomic) **propositions** (Boolean variables) and  $a \in \text{Prop}$ .

We write PropFormulae for the set of all propositional logic formulae.

**Syntactic sugar:**

$$\begin{aligned} \perp &:= (a \wedge \neg a) \\ \top &:= (a \vee \neg a) \\ (\varphi_1 \vee \varphi_2) &:= \neg((\neg\varphi_1) \wedge (\neg\varphi_2)) \\ (\varphi_1 \rightarrow \varphi_2) &:= ((\neg\varphi_1) \vee \varphi_2) \\ (\varphi_1 \leftrightarrow \varphi_2) &:= ((\varphi_1 \rightarrow \varphi_2) \wedge (\varphi_2 \rightarrow \varphi_1)) \\ (\varphi_1 \oplus \varphi_2) &:= (\varphi_1 \leftrightarrow (\neg\varphi_2)) \end{aligned}$$

- Examples of **well-formed** formulae:
  - $(\neg a)$
  - $(\neg(\neg a))$
  - $(a \wedge (b \wedge c))$
  - $(a \rightarrow (b \rightarrow c))$
- We omit parentheses whenever we may restore them through operator precedence:

binds stronger



$\neg \quad \wedge \quad \vee \quad \rightarrow \quad \leftrightarrow$

- Syntax of propositional logic
- **Semantics of propositional logic**
- Satisfiability and validity
- Modeling with propositional logic
- Normal forms
- Enumeration and deduction

**Structures** for predicate logic:

- The **domain** is  $\mathbb{B} = \{0, 1\}$ .
- The **interpretation** assigns Boolean values to the variables:

$$\alpha : \text{Prop} \rightarrow \{0, 1\}$$

We call these special interpretations **assignments** and use  $V$  to denote the set of all assignments.

**Example:**  $\text{Prop} = \{a, b\}, \alpha(a) = 0, \alpha(b) = 1$

Equivalently, we can see an assignment  $\alpha$  as a set of variables ( $\alpha \in 2^{\text{Prop}}$ ), defining the variables from the set to be true and the others false.

**Example:**  $\text{Prop} = \{a, b\}, \alpha = \{b\}$

An assignment can also be seen as being of type  $\alpha \in \{0, 1\}^{\text{Prop}}$ , if we have an order on the propositions.

**Example:**  $\text{Prop} = \{a, b\}, \alpha = 01$



# Only the projected assignment matters...

- Let  $\alpha_1, \alpha_2 \in V$  and  $\varphi \in \text{PropFormulae}$ .
- Let  $\text{AP}(\varphi)$  be the atomic propositions in  $\varphi$ .
- Clearly  $\text{AP}(\varphi) \subseteq \text{Prop}$ .
- **Lemma:** if  $\alpha_1|_{\text{AP}(\varphi)} = \alpha_2|_{\text{AP}(\varphi)}$ , then



Projection

$(\alpha_1 \text{ satisfies } \varphi) \text{ iff } (\alpha_2 \text{ satisfies } \varphi)$

- We will assume, for simplicity, that  $\text{Prop} = \text{AP}(\varphi)$ .

# Semantics I: Truth tables

- **Truth tables** define the semantics (=meaning) of the operators. They can be used to define the semantics of formulae inductively over their structure.
- Convention: **0**= false, **1**= true

$p$	$q$	$\neg p$	$p \wedge q$	$p \vee q$	$p \rightarrow q$	$p \leftrightarrow q$	$p \oplus q$
0	0	1	0	0	1	1	0
0	1	1	0	1	1	0	1
1	0	0	0	1	0	0	1
1	1	0	1	1	1	1	0

Each possible assignment is covered by a line of the truth table.

$\alpha$  **satisfies**  $\varphi$  iff in the line for  $\alpha$  and the column for  $\varphi$  the entry is 1.

**Q:** How many binary operators can we define that have different semantics?

**A:** 16

# Semantics I: Example

- Let  $\varphi$  be defined as  $(a \vee (b \rightarrow c))$ .
- Let  $\alpha : \{a, b, c\} \rightarrow \{0, 1\}$  be an assignment with  $\alpha(a) = 0$ ,  $\alpha(b) = 0$ , and  $\alpha(c) = 1$ .
- **Q:** Does  $\alpha$  satisfy  $\varphi$ ?
- **A1:** Compute with truth table:

$a$	$b$	$c$	$b \rightarrow c$	$a \vee (b \rightarrow c)$
0	0	0	1	1
0	0	1	1	1
0	1	0	0	0
0	1	1	1	1
1	0	0	1	1
1	0	1	1	1
1	1	0	0	1
1	1	1	1	1

## Semantics II: Satisfaction relation

**Satisfaction relation:**  $\models \subseteq V \times \text{PropFormulae}$

Instead of  $(\alpha, \varphi) \in \models$  we write  $\alpha \models \varphi$  and say that

- $\alpha$  satisfies  $\varphi$  or
- $\varphi$  holds for  $\alpha$  or
- $\alpha$  is a model of  $\varphi$ .

$\models$  is defined recursively:

$\alpha \models p$	<i>iff</i>	$\alpha(p) = \text{true}$
$\alpha \models \neg\varphi$	<i>iff</i>	$\alpha \not\models \varphi$
$\alpha \models \varphi_1 \wedge \varphi_2$	<i>iff</i>	$\alpha \models \varphi_1$ <i>and</i> $\alpha \models \varphi_2$
$\alpha \models \varphi_1 \vee \varphi_2$	<i>iff</i>	$\alpha \models \varphi_1$ <i>or</i> $\alpha \models \varphi_2$
$\alpha \models \varphi_1 \rightarrow \varphi_2$	<i>iff</i>	$\alpha \models \varphi_1$ <i>implies</i> $\alpha \models \varphi_2$
$\alpha \models \varphi_1 \leftrightarrow \varphi_2$	<i>iff</i>	$\alpha \models \varphi_1$ <i>iff</i> $\alpha \models \varphi_2$

**Note:** More elegant but semantically equivalent to truth tables.

## Semantics II: Example

- Let  $\varphi$  be defined as  $(a \vee (b \rightarrow c))$ .
- Let  $\alpha : \{a, b, c\} \rightarrow \{0, 1\}$  be an assignment with  $\alpha(a) = 0$ ,  $\alpha(b) = 0$ , and  $\alpha(c) = 1$ .
- Q: Does  $\alpha$  satisfy  $\varphi$ ?

A2: Compute with the satisfaction relation:

$$\alpha \models (a \vee (b \rightarrow c))$$

$$\text{iff } \alpha \models a \text{ or } \alpha \models (b \rightarrow c)$$

$$\text{iff } \alpha \models a \text{ or } (\alpha \models b \text{ implies } \alpha \models c)$$

$$\text{iff } 0 \text{ or } (0 \text{ implies } 1)$$

$$\text{iff } 0 \text{ or } 1$$

$$\text{iff } 1$$

## Semantics III: The algorithmic view

- Using the satisfaction relation we can define an **algorithm** for the problem to decide whether an assignment  $\alpha : AP \rightarrow \{0, 1\}$  is a model of a propositional logic formula  $\varphi \in \text{PropFormulae}$ :

```
Eval( $\alpha$ ,  $\varphi$ ) {  
  if  $\varphi \equiv a$  return  $\alpha(a)$ ;  
  if  $\varphi \equiv (\neg\varphi_1)$  return not Eval( $\alpha$ ,  $\varphi_1$ );  
  if  $\varphi \equiv (\varphi_1 \text{ op } \varphi_2)$   
    return Eval( $\alpha$ ,  $\varphi_1$ ) [[op]] Eval( $\alpha$ ,  $\varphi_2$ );  
}
```

- Equivalent to the  $\models$  relation, but from the algorithmic view.
- **Q:** Complexity? **A:** **Polynomial** (time and space).

- Recall our example

- $\varphi = (a \vee (b \rightarrow c))$

- $\alpha : \{a, b, c\} \rightarrow \{0, 1\}$  with  $\alpha(a) = 0$ ,  $\alpha(b) = 0$ , and  $\alpha(c) = 1$ .

- $Eval(\alpha, \varphi) = Eval(\alpha, a) \text{ or } Eval(\alpha, b \rightarrow c) =$   
 $0 \text{ or } (Eval(\alpha, b) \text{ implies } Eval(\alpha, c)) =$   
 $0 \text{ or } (0 \text{ implies } 1) =$   
 $0 \text{ or } 1 =$   
 $1$

- Hence,  $\alpha \models \varphi$ .

# Satisfying assignments

- Intuition: each formula specifies a **set of assignments** satisfying it.
- Remember:  $V$  denotes the set of all assignments.
- Function  $sat : \text{PropFormulae} \rightarrow 2^V$   
(a formula  $\rightarrow$  set of its satisfying assignments)
- Recursive definition:

$$sat(a) = \{\alpha \mid \alpha(a) = 1\}, \quad a \in \text{Prop}$$

$$sat(\neg\varphi_1) = V \setminus sat(\varphi_1)$$

$$sat(\varphi_1 \wedge \varphi_2) = sat(\varphi_1) \cap sat(\varphi_2)$$

$$sat(\varphi_1 \vee \varphi_2) = sat(\varphi_1) \cup sat(\varphi_2)$$

$$sat(\varphi_1 \rightarrow \varphi_2) = (V \setminus sat(\varphi_1)) \cup sat(\varphi_2)$$

- For  $\varphi \in \text{PropFormulae}$  and  $\alpha \in V$  it holds that

$$\alpha \models \varphi \quad \text{iff} \quad \alpha \in sat(\varphi)$$



# Satisfying assignments: Example

$$\text{sat}(a \vee (b \rightarrow c)) =$$

$$\text{sat}(a) \cup \text{sat}(b \rightarrow c) =$$

$$\text{sat}(a) \cup ((V \setminus \text{sat}(b)) \cup \text{sat}(c)) =$$

$$\{\alpha \in V \mid \alpha(a) = 1\} \cup$$

$$\{\alpha \in V \mid \alpha(b) = 0\} \cup$$

$$\{\alpha \in V \mid \alpha(c) = 1\} =$$

$$\{\alpha \in V \mid \alpha(a) = 1 \text{ or } \alpha(b) = 0 \text{ or } \alpha(c) = 1\}$$

- We define  $\models \subseteq 2^V \times \text{PropFormulae}$  by

$$T \models \varphi \text{ iff } T \subseteq \text{sat}(\varphi)$$

for formulae  $\varphi \in \text{PropFormulae}$  and assignment sets  $T \subseteq 2^V$ .

Examples:  $\{\alpha \in V \mid \alpha(a) = \alpha(c) = 1\} \models a \vee (b \rightarrow c)$   
 $\{\alpha \in V \mid \alpha(x_1) = 1\} \models x_1 \vee x_2$

- We define  $\models \subseteq 2^{\text{PropFormulae}} \times 2^{\text{PropFormulae}}$  by

$$\varphi_1 \models \varphi_2 \text{ iff } \text{sat}(\varphi_1) \subseteq \text{sat}(\varphi_2)$$

for formulae  $\varphi_1, \varphi_2 \in \text{PropFormulae}$ .

Examples:  $a \wedge c \models a \vee (b \rightarrow c)$   
 $x_1 \models x_1 \vee x_2$

# Short summary for propositional logic

- **Syntax** of propositional formulae  $\varphi \in \text{PropFormulae}$ :

$$\varphi := \text{prop} \mid (\neg\varphi) \mid (\varphi \wedge \varphi)$$

- **Semantics:**

- **Assignments**  $\alpha \in V$ :

$$\alpha : \text{Prop} \rightarrow \{0, 1\}$$

$$\alpha \in 2^{\text{Prop}}$$

$$\alpha \in \{0, 1\}^{\text{Prop}}$$

- **Satisfaction relation:**

$$\models \subseteq V \times \text{PropFormulae} \quad , \quad (\text{e.g., } \alpha \models \varphi)$$

$$\models \subseteq 2^V \times \text{PropFormulae} \quad , \quad (\text{e.g., } \{\alpha_1, \dots, \alpha_n\} \models \varphi)$$

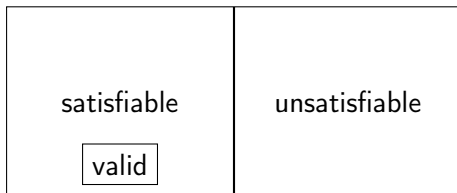
$$\models \subseteq \text{PropFormulae} \times \text{PropFormulae} \quad , \quad (\text{e.g., } \varphi_1 \models \varphi_2)$$

$$\text{sat} : \text{PropFormulae} \rightarrow 2^V \quad , \quad (\text{e.g., } \text{sat}(\varphi))$$

- Syntax of propositional logic
- Semantics of propositional logic
- Satisfiability and validity
- Modeling with propositional logic
- Normal forms
- Enumeration and deduction

# Semantic classification of formulae

- A formula  $\varphi$  is called **valid** if  $\text{sat}(\varphi) = V$ .  
(Also called a **tautology**).
- A formula  $\varphi$  is called **satisfiable** if  $\text{sat}(\varphi) \neq \emptyset$ .
- A formula  $\varphi$  is called **unsatisfiable** if  $\text{sat}(\varphi) = \emptyset$ .  
(Also called a **contradiction**).



- We can write:
  - $\models \varphi$  when  $\varphi$  is **valid**
  - $\not\models \varphi$  when  $\varphi$  is **not valid**
  - $\not\models \neg\varphi$  when  $\varphi$  is **satisfiable**
  - $\models \neg\varphi$  when  $\varphi$  is **unsatisfiable**

- $(x_1 \wedge x_2) \rightarrow (x_1 \vee x_2)$

is **valid**

- $(x_1 \vee x_2) \rightarrow x_1$

is **satisfiable**

- $(x_1 \wedge x_2) \wedge \neg x_1$

is **unsatisfiable**

- Here are some valid formulae:
  - $\models a \wedge 1 \leftrightarrow a$
  - $\models a \wedge 0 \leftrightarrow 0$
  - $\models \neg\neg a \leftrightarrow a$  (double-negation rule)
  - $\models a \wedge (b \vee c) \leftrightarrow (a \wedge b) \vee (a \wedge c)$
  
- Some more (De Morgan rules):
  - $\models \neg(a \wedge b) \leftrightarrow (\neg a \vee \neg b)$
  - $\models \neg(a \vee b) \leftrightarrow (\neg a \wedge \neg b)$



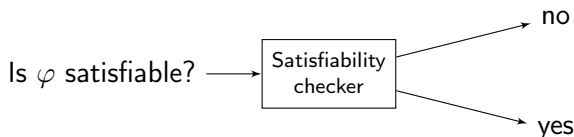
# The satisfiability problem for propositional logic

- The **satisfiability problem** for propositional logic is as follows:

*Given an input propositional formula  $\varphi$ , decide whether  $\varphi$  is satisfiable.*

- This problem is decidable but **NP-complete**.
- An algorithm that always terminates for each propositional logic formula with the correct answer is called a **decision procedure** for propositional logic.

**Goal:** Design and implement such a decision procedure:



**Note:** A formula  $\varphi$  is valid iff  $\neg\varphi$  is unsatisfiable.

- Syntax of propositional logic
- Semantics of propositional logic
- Satisfiability and validity
- Modeling with propositional logic
- Normal forms
- Enumeration and deduction

# Before we solve this problem...

- Suppose we can solve the satisfiability problem... how can this help us?
- There are numerous problems in the industry that are solved via the satisfiability problem of propositional logic
  - Logistics
  - Planning
  - Electronic Design Automation industry
  - Cryptography
  - ...

## Example 1: Placement of wedding guests

- Three chairs in a row: 1, 2, 3
- We need to place Aunt, Sister and Father.
- Constraints:
  - Aunt doesn't want to sit near Father
  - Aunt doesn't want to sit in the left chair
  - Sister doesn't want to sit to the right of Father
  
- Q: Can we satisfy these constraints?

## Example 1 (continued)

- **Notation:** Aunt = 1, Sister = 2, Father = 3

Left chair = 1, Middle chair = 2, Right chair = 3

Introduce a propositional variable for each pair (person, chair):

$x_{p,c}$  = "person  $p$  is sited in chair  $c$ " for  $1 \leq p, c \leq 3$

- **Constraints:**

Aunt doesn't want to sit near Father:

$$((x_{1,1} \vee x_{1,3}) \rightarrow \neg x_{3,2}) \wedge (x_{1,2} \rightarrow (\neg x_{3,1} \wedge \neg x_{3,3}))$$

Aunt doesn't want to sit in the left chair:

$$\neg x_{1,1}$$

Sister doesn't want to sit to the right of Father:

$$(x_{3,1} \rightarrow \neg x_{2,2}) \wedge (x_{3,2} \rightarrow \neg x_{2,3})$$

## Example 1 (continued)

Each person is placed:

$$(x_{1,1} \vee x_{1,2} \vee x_{1,3}) \wedge (x_{2,1} \vee x_{2,2} \vee x_{2,3}) \wedge (x_{3,1} \vee x_{3,2} \vee x_{3,3})$$

$$\bigwedge_{p=1}^3 \bigvee_{c=1}^3 x_{p,c}$$

No person is placed in more than one chair:

$$\bigwedge_{p=1}^3 \bigwedge_{c1=1}^3 \bigwedge_{c2=c1+1}^3 (\neg x_{p,c1} \vee \neg x_{p,c2})$$

At most one person per chair:

$$\bigwedge_{p1=1}^3 \bigwedge_{p2=p1+1}^3 \bigwedge_{c=1}^3 (\neg x_{p1,c} \vee \neg x_{p2,c})$$

## Example 2: Assignment of frequencies

- $n$  radio stations
- For each station assign one of  $k$  transmission frequencies,  $k < n$ .
- $E$  – set of pairs of stations, that are too close to have the same frequency.
- **Q:** Can we assign to each station a frequency, such that no station pairs from  $E$  have the same frequency?

## Example 2 (continued)

- **Notation:**

$x_{s,f}$  = “station  $s$  is assigned frequency  $f$ ” for  $1 \leq s \leq n$ ,  $1 \leq f \leq k$

- **Constraints:**

Every station is assigned at least one frequency:

$$\bigwedge_{s=1}^n \left( \bigvee_{f=1}^k x_{s,f} \right)$$

Every station is assigned at most one frequency:

$$\bigwedge_{s=1}^n \bigwedge_{f_1=1}^{k-1} \bigwedge_{f_2=f_1+1}^k (\neg x_{s,f_1} \vee \neg x_{s,f_2})$$

Close stations are not assigned the same frequency:

For each  $(s_1, s_2) \in E$ ,

$$\bigwedge_{f=1}^k (\neg x_{s_1,f} \vee \neg x_{s_2,f})$$



## Example 3: Seminar topic assignment

- $n$  participants
- $n$  topics
- Set of preferences  $E \subseteq \{1, \dots, n\} \times \{1, \dots, n\}$   
 $(p, t) \in E$  means: participant  $p$  would take topic  $t$
- **Q:** Can we assign to each participant a topic which he/she is willing to take?

## Example 3 (continued)

- **Notation:**  $x_{p,t}$  = “participant  $p$  is assigned topic  $t$ ”
- **Constraints:**

Each participant is assigned at least one topic:

$$\bigwedge_{p=1}^n \left( \bigvee_{t=1}^n x_{p,t} \right)$$

Each participant is assigned at most one topic:

$$\bigwedge_{p=1}^n \bigwedge_{t_1=1}^{n-1} \bigwedge_{t_2=t_1+1}^n (\neg x_{p,t_1} \vee \neg x_{p,t_2})$$

Each participant is willing to take his/her assigned topic:

$$\bigwedge_{p=1}^n \bigwedge_{(p,t) \notin E} \neg x_{p,t}$$

## Example 3 (continued)

Each topic is assigned to at most one participant:

$$\bigwedge_{t=1}^n \bigwedge_{p1=1}^n \bigwedge_{p2=p1+1}^n (\neg x_{p1,t} \vee \neg x_{p2,t})$$

- Syntax of propositional logic
- Semantics of propositional logic
- Satisfiability and validity
- Modeling with propositional logic
- Normal forms
- Enumeration and deduction

- Definition: A **literal** is either a variable or a negation of a variable.

- Example:  $\varphi = \neg(a \vee \neg b)$

Variables:  $AP(\varphi) = \{a, b\}$

Literals:  $lit(\varphi) = \{a, \neg b\}$

- Note: Equivalent formulae can have different literals.

Example:  $\varphi' = \neg a \wedge b$

Literals:  $lit(\varphi') = \{\neg a, b\}$

- Definition: a **term** is a conjunction of literals
  - Example:  $(a \wedge \neg b \wedge c)$
- Definition: a **clause** is a disjunction of literals
  - Example:  $(a \vee \neg b \vee c)$

# Negation Normal Form (NNF)

- Definition: A formula is in **Negation Normal Form (NNF)** iff
  - (1) it contains only  $\neg$ ,  $\wedge$  and  $\vee$  as connectives and
  - (2) only variables are negated.
- **Examples:**
- $\varphi_1 = \neg(a \vee \neg b)$  is **not** in NNF
- $\varphi_2 = \neg a \wedge b$  is **in** NNF

- Every formula can be converted to NNF in linear time:
  - Eliminate all connectives other than  $\wedge$ ,  $\vee$ ,  $\neg$
  - Use De Morgan and double-negation rules to push negations to operands
- **Example:**  $\varphi = \neg(a \rightarrow \neg b)$ 
  - Eliminate ' $\rightarrow$ ':  $\varphi = \neg(\neg a \vee \neg b)$
  - Push negation using De Morgan:  $\varphi = (\neg\neg a \wedge \neg\neg b)$
  - Use double-negation rule:  $\varphi = (a \wedge b)$



# Disjunctive Normal Form (DNF)

- Definition: A formula is said to be in **Disjunctive Normal Form (DNF)** iff it is a disjunction of terms.
- In other words, it is a formula of the form

$$\bigvee_i \left( \bigwedge_j l_{i,j} \right)$$

where  $l_{i,j}$  is the  $j$ -th literal in the  $i$ -th term.

- Example:

$$\varphi = (a \wedge \neg b \wedge c) \vee (\neg a \wedge d) \vee (b) \text{ is in DNF}$$

- DNF is a special case of NNF.

# Converting to DNF

- Every formula can be converted to DNF in **exponential** time and space:

- 1 Convert to NNF

- 2 Distribute disjunctions following the rule:

$$\models \varphi_1 \wedge (\varphi_2 \vee \varphi_3) \leftrightarrow (\varphi_1 \wedge \varphi_2) \vee (\varphi_1 \wedge \varphi_3)$$

- **Example:**

$$\begin{aligned}\varphi &= (a \vee b) \wedge (\neg c \vee d) \\ &= ((a \vee b) \wedge (\neg c)) \vee ((a \vee b) \wedge d) \\ &= (a \wedge \neg c) \vee (b \wedge \neg c) \vee (a \wedge d) \vee (b \wedge d)\end{aligned}$$

- Now consider  $\varphi_n = (a_1 \vee b_1) \wedge (a_2 \vee b_2) \wedge \dots \wedge (a_n \vee b_n)$ .

- **Q:** How many clauses will the DNF have?

**A:**  $2^n$

- Q: Is the following DNF formula satisfiable?

$$(a_1 \wedge a_2 \wedge \neg a_1) \vee (a_2 \wedge a_1) \vee (a_2 \wedge \neg a_3 \wedge a_3)$$

A: Yes, because the term  $a_2 \wedge a_1$  is satisfiable.

- Q: What is the complexity of the satisfiability check of DNF formulae?

A: Linear (time and space).

- Q: Can there be any polynomial transformation into DNF?

- A: No, it would violate the NP-completeness of the problem.

# Conjunctive Normal Form (CNF)

- Definition: A formula is said to be in **Conjunctive Normal Form (CNF)** iff it is a conjunction of clauses.
- In other words, it is a formula of the form

$$\bigwedge_i \left( \bigvee_j l_{i,j} \right)$$

where  $l_{i,j}$  is the  $j$ -th literal in the  $i$ -th clause.

- Example:

$$\varphi = (a \vee \neg b \vee c) \wedge (\neg a \vee d) \wedge (b) \text{ is in CNF}$$

- Also CNF is a special case of NNF.

- Every formula can be converted to CNF in **exponential** time and space:

- 1 Convert to NNF

- 2 Distribute disjunctions following the rule:

$$\models \varphi_1 \vee (\varphi_2 \wedge \varphi_3) \leftrightarrow (\varphi_1 \vee \varphi_2) \wedge (\varphi_1 \vee \varphi_3)$$

- Consider the formula  $\varphi = (a_1 \wedge b_1) \vee (a_2 \wedge b_2)$ .

Transformation:  $(a_1 \vee a_2) \wedge (a_1 \vee b_2) \wedge (b_1 \vee a_2) \wedge (b_1 \vee b_2)$

- Now consider  $\varphi_n = (a_1 \wedge b_1) \vee (a_2 \wedge b_2) \vee \dots \vee (a_n \wedge b_n)$ .

Q: How many clauses does the resulting CNF have?

A:  $2^n$

# Converting to CNF: Tseitin's encoding

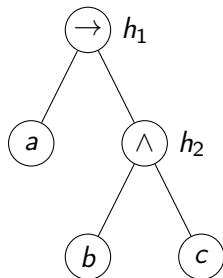
- Every formula can be converted to CNF in **linear** time and space if new variables are added.
- The original and the converted formulae are **not equivalent** but **equi-satisfiable**.

- Consider the formula

$$\varphi = (a \rightarrow (b \wedge c))$$

- Associate a new auxiliary variable with each gate.
- Add constraints that define these new variables.
- Finally, enforce the root node.

Parse tree:

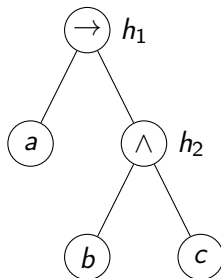


- Need to satisfy:

$$(h_1 \leftrightarrow (a \rightarrow h_2)) \wedge$$

$$(h_2 \leftrightarrow (b \wedge c)) \wedge$$

$$(h_1)$$



- Each gate encoding has a CNF representation with 3 or 4 clauses.

- Need to satisfy:

$$(h_1 \leftrightarrow (a \rightarrow h_2)) \wedge (h_2 \leftrightarrow (b \wedge c)) \wedge (h_1)$$

- First:  $(h_1 \vee a) \wedge (h_1 \vee \neg h_2) \wedge (\neg h_1 \vee \neg a \vee h_2)$
- Second:  $(\neg h_2 \vee b) \wedge (\neg h_2 \vee c) \wedge (h_2 \vee \neg b \vee \neg c)$



- Let's go back to

$$\varphi_n = (x_1 \wedge y_1) \vee (x_2 \wedge y_2) \vee \cdots \vee (x_n \wedge y_n)$$

- With Tseitin's encoding we need:

- $n$  auxiliary variables  $h_1, \dots, h_n$ .
- Each adds 3 constraints.
- Top clause:  $(h_1 \vee \cdots \vee h_n)$

- Hence, we have

- $3n + 1$  clauses, instead of  $2^n$ .
- $3n$  variables rather than  $2n$ .

- Syntax of propositional logic
- Semantics of propositional logic
- Satisfiability and validity
- Modeling with propositional logic
- Normal forms
- Enumeration and deduction

# Two classes of algorithms for validity

- **Q:** Is  $\varphi$  satisfiable? (Is  $\neg\varphi$  valid?)
- Complexity: **NP-Complete** (Cook's theorem)
- Two classes of algorithms for finding out:
  - Enumeration of possible solutions (Truth tables etc.)
  - Deduction
  
- More generally (beyond propositional logic):
  - **Enumeration** is possible only in some logics.
  - **Deduction** cannot necessarily be fully automated.

# The satisfiability problem

- Given a formula  $\varphi$ , is  $\varphi$  satisfiable?

## Enumeration the first:

```
Boolean SAT( $\varphi$ ) {  
    result := false ;  
    for all  $\alpha \in V$   
        result = result  $\vee$  Eval( $\alpha$ ,  $\varphi$ ) ;  
    return result ;  
}
```

## Enumeration the second:

Use substitution to eliminate all variables one by one:

$$\varphi \quad \text{iff} \quad \varphi[0/a] \vee \varphi[1/a]$$

- Q: What is the difference?

# Deduction requires axioms and inference rules

## ■ Inference rules:

$$\frac{\textit{Antecedents}}{\textit{Consequents}} \quad (\textit{rule name})$$

Meaning: If all antecedents hold then at least one of the consequents can be derived.

## ■ Examples:

$$\frac{a \rightarrow b \quad b \rightarrow c}{a \rightarrow c} \quad (\textit{Trans})$$

$$\frac{a \rightarrow b \quad a}{b} \quad (\textit{M.P.})$$

- **Axioms** are inference rules with no antecedents, e.g.,

$$\frac{}{a \rightarrow (b \rightarrow a)} \quad (H1)$$

- A **proof system** consists of a set of axioms and inference rules.

- Let  $\mathcal{H}$  be a proof system.
- $\Gamma \vdash_{\mathcal{H}} \varphi$  means: There is a proof of  $\varphi$  in system  $\mathcal{H}$  whose premises are included in  $\Gamma$
- $\vdash_{\mathcal{H}}$  is called the **provability relation**.

# Example

- Let  $\mathcal{H}$  be the proof system comprised of the rules **Trans** and **M.P.** that we saw earlier:

$$\frac{a \rightarrow b \quad b \rightarrow c}{a \rightarrow c} \quad (\text{Trans})$$

$$\frac{a \rightarrow b \quad a}{b} \quad (\text{M.P.})$$

- Does the following relation hold?

$$a \rightarrow b, b \rightarrow c, c \rightarrow d, d \rightarrow e, a \vdash_{\mathcal{H}} e$$



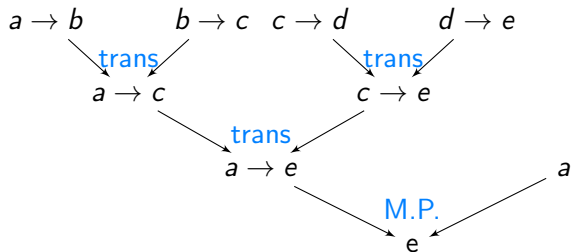
# Deductive proof: Example

$$\frac{a \rightarrow b \quad b \rightarrow c}{a \rightarrow c} \quad (\text{Trans}) \quad \frac{a \rightarrow b \quad a}{b} \quad (\text{M.P.})$$

$a \rightarrow b, b \rightarrow c, c \rightarrow d, d \rightarrow e, a \vdash_{\mathcal{H}} e$

1.  $a \rightarrow b$  *premise*
2.  $b \rightarrow c$  *premise*
3.  $a \rightarrow c$  1, 2, *Trans*
4.  $c \rightarrow d$  *premise*
5.  $d \rightarrow e$  *premise*
6.  $c \rightarrow e$  4, 5, *Trans*
7.  $a \rightarrow e$  3, 6, *Trans*
8.  $a$  *premise*
9.  $e$  7, 8, *M.P.*

# Proof graph



- For a given proof system  $\mathcal{H}$ ,
  - **Soundness:** Does  $\vdash$  conclude “correct” conclusions from premises?
  - **Completeness:** Can we conclude all true statements with  $\mathcal{H}$ ?
- **Correct with respect to what?**

With respect to the semantic definition of the logic. In the case of propositional logic truth tables give us this.

- Let  $\mathcal{H}$  be a proof system

*Soundness of  $\mathcal{H}$*  :    if  $\vdash_{\mathcal{H}} \varphi$     then  $\models \varphi$

*Completeness of  $\mathcal{H}$*  :    if  $\models \varphi$     then  $\vdash_{\mathcal{H}} \varphi$

- How to prove soundness and completeness?

## Example: Hilbert axiom system (H)

- Let H be (M.P.) together with the following axiom schemes:

$$\overline{a \rightarrow (b \rightarrow a)} \quad (H1)$$

$$\overline{((a \rightarrow (b \rightarrow c)) \rightarrow ((a \rightarrow b) \rightarrow (a \rightarrow c)))} \quad (H2)$$

$$\overline{(\neg b \rightarrow \neg a) \rightarrow (a \rightarrow b)} \quad (H3)$$

- H is **sound and complete** for propositional logic.

- To prove soundness of H, prove the soundness of its axioms and inference rules (easy with truth-tables).

For example:

$a$	$b$	$a \rightarrow (b \rightarrow a)$
0	0	1
0	1	1
1	0	1
1	1	1

- Completeness: harder, but possible.

# The resolution inference system

- The **resolution** inference rule for CNF:

$$\frac{(I \vee l_1 \vee l_2 \vee \dots \vee l_n) \quad (\neg I \vee l'_1 \vee \dots \vee l'_m)}{(l_1 \vee \dots \vee l_n \vee l'_1 \vee \dots \vee l'_m)} \text{ Resolution}$$

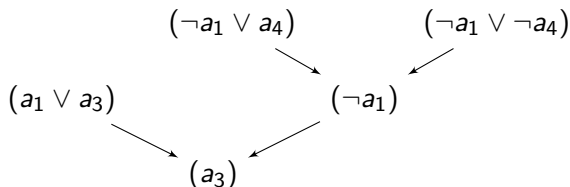
- **Example:**

$$\frac{(a \vee b) \quad (\neg a \vee c)}{(b \vee c)}$$

- We first see some example proofs, before proving soundness and completeness.

# Proof by resolution

- Let  $\varphi = (a_1 \vee a_3) \wedge (\neg a_1 \vee a_2 \vee a_5) \wedge (\neg a_1 \vee a_4) \wedge (\neg a_1 \vee \neg a_4)$
- We want to prove  $\varphi \rightarrow (a_3)$

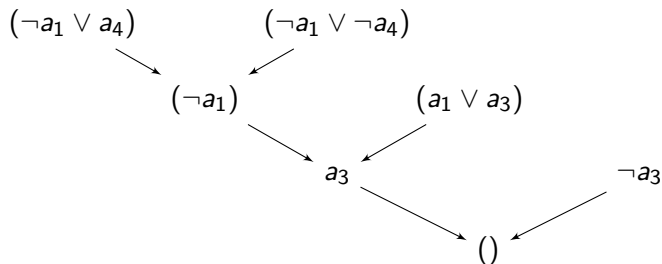




- Resolution is a sound and complete inference system for CNF.
- If the input formula is unsatisfiable, there exists a proof of the empty clause.

# Example

Let  $\varphi = (a_1 \vee a_3) \wedge (\neg a_1 \vee a_2) \wedge (\neg a_1 \vee a_4) \wedge (\neg a_1 \vee \neg a_4) \wedge (\neg a_3)$ .



# Soundness and completeness of resolution

- **Soundness** is straightforward. Just prove by truth table that

$$\models ((\varphi_1 \vee a) \wedge (\varphi_2 \vee \neg a)) \rightarrow (\varphi_1 \vee \varphi_2).$$

- **Completeness** is a bit more involved.

Basic idea: Use resolution for **variable elimination**.

$$\begin{aligned} & (a \vee \varphi_1) \wedge \dots \wedge (a \vee \varphi_n) \wedge \\ & (\neg a \vee \psi_1) \wedge \dots \wedge (\neg a \vee \psi_m) \wedge \\ & \quad R \\ & \Leftrightarrow \\ & (\varphi_1 \vee \psi_1) \wedge \dots \wedge (\varphi_1 \vee \psi_m) \wedge \\ & \quad \dots \\ & (\varphi_n \vee \psi_1) \wedge \dots \wedge (\varphi_n \vee \psi_m) \wedge \\ & \quad R \end{aligned}$$

where  $\varphi_i$  ( $i = 1, \dots, n$ ),  $\psi_j$  ( $j = 1, \dots, m$ ), and  $R$  contains neither  $a$  nor  $\neg a$ .