

# UCS1524 – Logic Programming

Fundamentals of Propositional Logic



# Session Meta Data

---

Author	Dr. D. Thenmozhi
Reviewer	
Version Number	1.2
Release Date	16 July 2022

# Session Objectives

---

- Understanding syntax and semantics of propositional logic (PL)
- Learning formal representation of PL, validity, satisfiability and semantic equivalence in PL.

# Session Outcomes

---

- At the end of this session, participants will be able to
  - explain the formal representation of statements in PL and how to check validity of the statements in PL.

# Agenda

---

- Syntax and Semantics of PL
- Validity and satisfiability
- Semantic equivalence

# What is logic?

---

- The philosophical definition is that logic is a description of how **one should think**.
- In the context of AI, logic is "**formal**," which means it resembles math in its clarity and lack of ambiguity.
- According to Merriam-Webster dictionary, logic is a science that deals with the **principles and criteria of validity** of inference and demonstration
- E.g.
  - The arrangement of circuit elements (as in a computer) needed for computation
  - Parsing both natural language and programming language statements

# Formal Logic

---

Formal Logic consists of

- syntax – what is a legal sentence in the logic
- semantics – what is the meaning of a sentence in the logic
- proof theory – formal (syntactic) procedure to construct valid/true sentences

Formal logic provides

- a language to precisely express knowledge, requirements, facts
- a formal way to reason about consequences of given facts rigorously

# Where is Formal Logic used in SE?

---

## Programming Languages

- conditional statements
- meaning (semantics) of programs

## Requirements and Specification

- rigorous definition of what is to be constructed
- e.g., if we used formal logic for assignments, there would be no questions on what is required, what is optional, and no questions

## Computer Hardware

- computers are build out of simple logical gates
- most computer hardware can be specified and understood in propositional logic

## Testing and Verification

- rigorously validate that software satisfies its specifications

## Algorithms and Optimization

- many complex problems can be reduced to logic and solved effectively using automated decision procedures



# Propositional Logic (or Boolean Logic)

---

Explores simple grammatical connections such as *and*, *or*, and *not* between simplest “atomic sentences”

A = “Paris is the capital of France”

B = “mice chase elephants”

The subject of propositional logic is to declare formally the truth of complex structures from the truth of individual atomic components

A and B

A or B

if A then B

# Syntax and Semantics

---

## Syntax

- Rules used to express facts and concepts.
- The way in which linguistic elements (such as words) are put together to form constituents (such as phrases or clauses)
- Determines and restricts how things are written

## Semantics

- The study of meanings
- Determine the truth value of the logic formula.
- Determines how syntax is interpreted to give meaning

# Syntax of Propositional Logic

---

An *atomic formula* has a form  $A_i$ , where  $i = 1, 2, 3 \dots$

*Formulas* are defined inductively as follows:

- All atomic formulas are formulas
- For every formula  $F$ ,  $\neg F$  (called not  $F$ ) is a formula
- For all formulas  $F$  and  $G$ ,  $F \wedge G$  (called and) and  $F \vee G$  (called or) are formulas

## Abbreviations

- use  $A, B, C, \dots$  instead of  $A_1, A_2, \dots$
- use  $F_1 \rightarrow F_2$  instead of  $\neg F_1 \vee F_2$  (implication)
- use  $F_1 \leftrightarrow F_2$  instead of  $(F_1 \rightarrow F_2) \wedge (F_2 \rightarrow F_1)$  (iff)

# Syntax of Propositional Logic (PL)

---

$\text{truth\_symbol} ::= \top(\text{true}) \mid \perp(\text{false})$

$\text{variable} ::= p, q, r, \dots$

$\text{atom} ::= \text{truth\_symbol} \mid \text{variable}$

$\text{literal} ::= \text{atom} \mid \neg \text{atom}$

$\text{formula} ::= \text{literal} \mid$

$\neg \text{formula} \mid$

negation

$\text{formula} \wedge \text{formula} \mid$

conjunction

$\text{formula} \vee \text{formula} \mid$

disjunction

$\text{formula} \rightarrow \text{formula} \mid$

implies

$\text{formula} \leftrightarrow \text{formula}$

equivalence

# Example

---

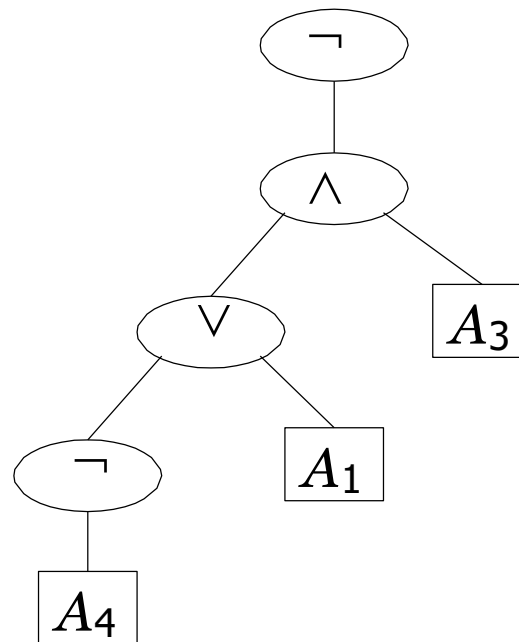
$$F = \neg((A_5 \wedge A_6) \vee \neg A_3)$$

Sub-formulas are

$$\begin{aligned} &F, ((A_5 \wedge A_6) \vee \neg A_3), \\ &A_5 \wedge A_6, \neg A_3, \\ &A_5, A_6, A_3 \end{aligned}$$

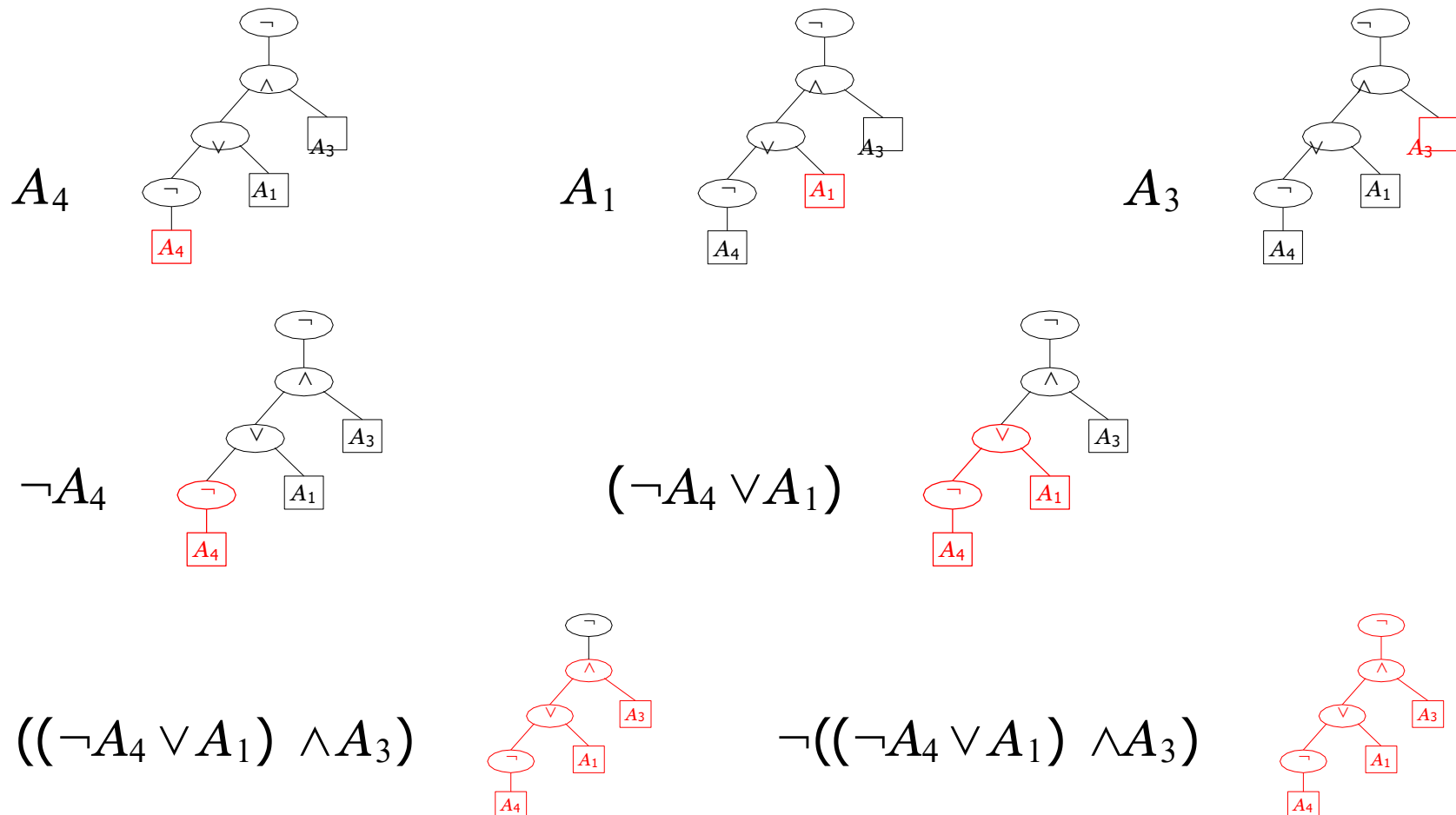
# Syntax tree of a formula

Every formula can be represented by a syntax tree. **Example:**  $F = \neg((\neg A_4 \vee A_1) \wedge A_3)$



# Subformulas

The **subformulas** of a formula are the formulas corresponding to the subtrees of its syntax tree.



# Semantics of propositional logic

---

We start with two truth values:  $\{0, 1\}$

- 0 stands for False, and 1 stands for True

Let  $\mathbf{D}$  be any subset of the *atomic* formulas

An *assignment*  $\mathbf{A}$  is a map  $\mathbf{D} \rightarrow \{0, 1\}$

- $\mathbf{A}$  assigns True or False to every atomic in  $\mathbf{D}$

Let  $\mathbf{E} \supseteq \mathbf{D}$  be set of formulas built from  $\mathbf{D}$  using propositional connectives

*Extended assignment*  $\mathbf{A}'$ :  $\mathbf{E} \rightarrow \{0, 1\}$  extends  $\mathbf{A}$  from atomic formulas to all formulas



# Semantics of propositional logic

For an atomic formula  $A_i$  in  $\mathbf{D}$ :  $\mathbf{A}'(A_i) = \mathbf{A}(A_i)$

$$\begin{aligned} \mathbf{A}'(F \wedge G) &= 1 && \text{if } \mathbf{A}'(F) = 1 \text{ and } \mathbf{A}'(G) = 1 \\ &= 0 && \text{otherwise} \end{aligned}$$

$$\begin{aligned} \mathbf{A}'(F \vee G) &= 1 && \text{if } \mathbf{A}'(F) = 1 \text{ or } \mathbf{A}'(G) = 1 \\ &= 0 && \text{otherwise} \end{aligned}$$

$$\begin{aligned} \mathbf{A}'(\neg F) &= 1 && \text{if } \mathbf{A}'(F) = 0 \\ &= 0 && \text{otherwise} \end{aligned}$$

## Exercise: Define Extended Assignment

---

$$F = \neg(A \wedge B) \vee C$$

$$\mathcal{A}(A) = 1$$

$$\mathcal{A}(B) = 1$$

$$\mathcal{A}(C) = 0$$

Is  $F$  true or false under  $A'$ ?

# Truth Tables for Basic Operators

$\mathcal{A}(F)$	$\mathcal{A}(G)$	$\mathcal{A}'(F \wedge G)$
0	0	0
0	1	0
1	0	0
1	1	1

$\mathcal{A}(F)$	$\mathcal{A}(G)$	$\mathcal{A}'(F \vee G)$
0	0	0
0	1	1
1	0	1
1	1	1

An extended assignment  $\mathcal{A}'$  extends the truth table from atomic propositions to propositional formulas

$\mathcal{A}(F)$	$\mathcal{A}'(\neg F)$
0	1
1	0

## Formula

$$F = \neg(A \wedge B) \vee C$$

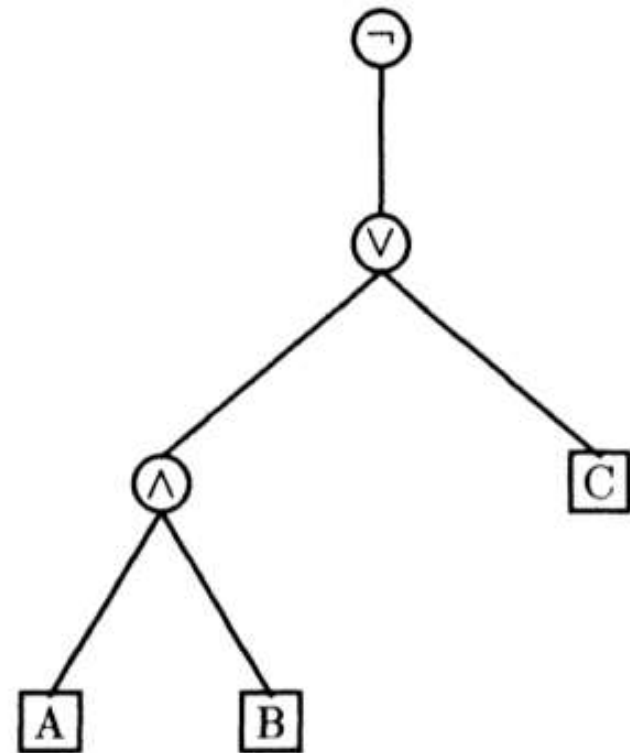
## Assignment

$$\mathcal{A}(A) = 1$$

$$\mathcal{A}(B) = 1$$

$$\mathcal{A}(C) = 0$$

## Abstract Syntax Tree (AST)



# Abbreviations

$A, B, C,$

$P, Q, R,$  or ... for  $A_1, A_2, A_3 \dots$

$(F_1 \rightarrow F_2)$  for  $(\neg F_1 \vee F_2)$

$(F_1 \leftrightarrow F_2)$  for  $((F_1 \wedge F_2) \vee (\neg F_1 \wedge \neg F_2))$

$(\bigvee_{i=1}^n F_i)$  for  $(\dots ((F_1 \vee F_2) \vee F_3) \vee \dots \vee F_n)$

$(\bigwedge_{i=1}^n F_i)$  for  $(\dots ((F_1 \wedge F_2) \wedge F_3) \wedge \dots \wedge F_n)$

$\top$  or true or 1 for  $(A_1 \vee \neg A_1)$

$\perp$  or false or 0 for  $(A_1 \wedge \neg A_1)$

# Truth tables

Tables for the operators  $\rightarrow$ ,  $\leftrightarrow$ :

$A$	$B$	$A$	$\rightarrow$	$B$
0	0	0	1	0
0	1	0	1	1
1	0	1	0	0
1	1	1	1	1

Name: *implication*

Interpretation: If  $A$  holds, then  $B$  holds.

$A$	$B$	$A$	$\leftrightarrow$	$B$
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	1	1	1

Name: *equivalence*

Interpretation:  $A$  holds if and only if  $B$  holds.



# Formalizing natural language

---

A device consists of two parts  $A$  and  $B$ , and a red light. We know that:

- $A$  or  $B$  (or both) are broken.
- If  $A$  is broken, then  $B$  is broken.
- If  $B$  is broken and the red light is on, then  $A$  is not broken.
- The red light is on.

We use the atomic formulas:  $Ro$  (red light on),  $Ab$  ( $A$  is broken),  $Bb$  ( $B$  is broken), and formalize this situation by means of the formula

$$((((Ab \vee Bb) \wedge (Ab \rightarrow Bb)) \wedge ((Bb \wedge Ro) \rightarrow \neg Ab))) \wedge Ro$$

# Formalizing natural language

Full truth table:

$Ro$	$Ab$	$Bb$	$(((((Ab \vee Bb) \wedge (Ab \rightarrow Bb)))) \wedge ((Bb \wedge Ro) \rightarrow \neg Ab)) \wedge Ro)$
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	0



# Propositional Logic: Semantics

---

An assignment  $A$  is *suitable* for a formula  $F$  if  $A$  assigns a truth value to every atomic proposition of  $F$

An assignment  $A$  is a *model* for  $F$ , written  $A \models F$ , iff

- $A$  is suitable for  $F$
- $A'(F) = 1$ , i.e.,  $F$  evaluates to true (or holds) under  $A$

A formula  $F$  is *satisfiable* iff  $F$  has a model, otherwise  $F$  is *unsatisfiable* (or contradictory)

A formula  $F$  is *valid* (or a tautology), written  $\models F$ , iff every suitable assignment for  $F$  is a model for  $F$

# Determining Satisfiability via a Truth Table

A formula  $F$  with  $n$  atomic sub-formulas has  $2^n$  suitable assignments

Build a truth table enumerating all assignments

$F$  is satisfiable iff there is at least one entry with 1 in the output

	$A_1$	$A_2$	$\dots$	$A_{n-1}$	$A_n$	$F$
$\mathcal{A}_1:$	0	0		0	0	$\mathcal{A}_1(F)$
$\mathcal{A}_2:$	0	0		0	1	$\mathcal{A}_2(F)$
$\vdots$			$\ddots$			$\vdots$
$\mathcal{A}_{2^n}:$	1	1		1	1	$\mathcal{A}_{2^n}(F)$

## Problem: Is formula $F$ SAT?

---

$$F = (\neg A \rightarrow (A \rightarrow B))$$

$A$	$B$	$\neg A$	$(A \rightarrow B)$	$F$
0	0	1	1	1
0	1	1	1	1
1	0	0	0	1
1	1	0	1	1

# Validity and Unsatisfiability

---

## Theorem:

A formula  $F$  is valid if and only if  $\neg F$  is unsatisfiable

## Proof:

$F$  is valid  $\Leftrightarrow$  every suitable assignment for  $F$  is a model for  $F$   
 $\Leftrightarrow$  every suitable assignment for  $\neg F$  is not a model for  $\neg F$   
 $\Leftrightarrow \neg F$  does not have a model  
 $\Leftrightarrow \neg F$  is unsatisfiable

# Mirroring principle

---

valid formulas	satisfiable, but not valid formulas		unsatis- fiable formulas
$\neg G$	$F$	$\neg F$	$G$

## Exercise

---

Prove or give a counter example

Valid

(a) If  $(F \Rightarrow G)$  is *valid* and  $F$  is *valid*, then  $G$  is *valid*

(b) If  $(F \Rightarrow G)$  is *sat* and  $F$  is *sat*, then  $G$  is *sat*

Not Valid

(c) If  $(F \Rightarrow G)$  is *valid* and  $F$  is *sat*, then  $G$  is *sat*

Valid

# Semantic Equivalence

---

Two formulas  $F$  and  $G$  are *(semantically) equivalent*, written  $F \equiv G$ , iff for every assignment  $A$  that is suitable for both  $F$  and  $G$ ,  $A'(F) = A'(G)$

For example,  $(F \wedge G)$  is equivalent to  $(G \wedge F)$

Formulas with different atomic propositions can be equivalent

- e.g., all tautologies are equivalent to True
- e.g., all unsatisfiable formulas are equivalent to False

# Substitution Theorem

---

**Theorem:** Let  $F$  and  $G$  be equivalent formulas. Let  $H$  be a formula in which  $F$  occurs as a sub-formula. Let  $H'$  be a formula obtained from  $H$  by replacing every occurrence of  $F$  by  $G$ . Then,  $H$  and  $H'$  are equivalent.

**Proof:**

(Let's talk about proof by induction first...)



# Mathematical Induction (over Natural Numbers)

---

To prove that a property  $P(n)$  holds for all natural numbers  $n$

1. Show that  $P(0)$  is true
2. Assume that  $P(k)$  is true for some natural number  $k$ 
  - This assumption is called an **Inductive Hypothesis (IH)**
3. Show that  $P(k+1)$  is true using IH from the previous step
4. Conclude that  $P(n)$  holds for all natural numbers  $n$ 
  - $P(n)$  is proven (or established, true) by mathematical induction

# Example: Mathematical Induction

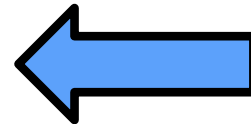
Show by induction that the formula for arithmetic series is correct

$$0 + \dots + n = \frac{n(n+1)}{2}$$

Base Case:  $P(0)$  is  $0 = \frac{0(0+1)}{2}$

IH: Assume  $P(k)$ , show  $P(k+1)$

$$\begin{aligned} & 0 + \dots + k + (k+1) \\ = & \frac{k(k+1)}{2} + (k+1) \\ = & \frac{k(k+1) + 2(k+1)}{2} \\ = & \frac{(k+1)((\frac{k}{2}+1)+1)}{2} \end{aligned}$$



IH is used in this step

# Structural Induction on the formula structure

---

The definition of a syntax of a formula is an *inductive* definition

- first, define atomic formulas; second, define more complex formulas from simple ones, each next definition uses previous definition recursively

The definition of the semantics of a formula is also inductive

- first, determine value of atomic propositions; second, define values of more complex formulas

The same principle works for proving properties of formulas!

- To show that every formula  $F$  satisfies some property  $S$ :
- (base case) show that  $S$  holds for atomic formulae
- (induction step) assume  $S$  holds for an arbitrary fixed formulas  $F$  and  $G$ . Show that  $S$  holds for  $(F \wedge G)$ ,  $(F \vee G)$ , and  $(\neg F)$

# Substitution Theorem

---

**Theorem:** Let  $F$  and  $G$  be equivalent formulas. Let  $H$  be a formula in which  $F$  occurs as a sub-formula. Let  $H'$  be a formula obtained from  $H$  by replacing every occurrence of  $F$  by  $G$ . Then,  $H$  and  $H'$  are equivalent.

**Proof:** by induction on formula structure

(base case) if  $H$  is atomic, then  $F = H$ ,  $H' = G$ , and  $F \equiv G$

(inductive step)

(case 1)  $H = \neg H_1$

(case 2)  $H = H_1 \wedge H_2$

(case 3)  $H = H_1 \vee H_2$

# Useful Equivalences

$$\begin{aligned}(F \wedge F) &\equiv F \\ (F \vee F) &\equiv F\end{aligned}\quad (\text{Idempotency})$$

$$\begin{aligned}(F \wedge G) &\equiv (G \wedge F) \\ (F \vee G) &\equiv (G \vee F)\end{aligned}\quad (\text{Commutativity})$$

$$\begin{aligned}((F \wedge G) \wedge H) &\equiv (F \wedge (G \wedge H)) \\ ((F \vee G) \vee H) &\equiv (F \vee (G \vee H))\end{aligned}\quad (\text{Associativity})$$

$$\begin{aligned}(F \wedge (F \vee G)) &\equiv F \\ (F \vee (F \wedge G)) &\equiv F\end{aligned}\quad (\text{Absorption})$$

$$\begin{aligned}(F \wedge (G \vee H)) &\equiv ((F \wedge G) \vee (F \wedge H)) \\ (F \vee (G \wedge H)) &\equiv ((F \vee G) \wedge (F \vee H))\end{aligned}\quad (\text{Distributivity})$$

$$\neg\neg F \equiv F \quad (\text{Double Negation})$$

# Useful Equivalences

---

$$\begin{aligned}\neg(F \wedge G) &\equiv (\neg F \vee \neg G) \\ \neg(F \vee G) &\equiv (\neg F \wedge \neg G)\end{aligned}\quad (\text{deMorgan's Laws})$$

$$\begin{aligned}(F \vee G) &\equiv F, \text{ if } F \text{ is a tautology} \\ (F \wedge G) &\equiv G, \text{ if } F \text{ is a tautology}\end{aligned}\quad (\text{Tautology Laws})$$

$$\begin{aligned}(F \vee G) &\equiv G, \text{ if } F \text{ is unsatisfiable} \\ (F \wedge G) &\equiv F, \text{ if } F \text{ is unsatisfiable}\end{aligned}\quad (\text{Unsatisfiability Laws})$$

We can prove them using structural induction!

# Equivalence

- Prove that using substitution theorem

$$((A \vee (B \vee C)) \wedge (C \vee \neg A)) = ((B \wedge \neg A) \vee C)$$

$$\begin{aligned} & ((A \vee (B \vee C)) \wedge (C \vee \neg A)) \\ \equiv & (((A \vee B) \vee C) \wedge (C \vee \neg A)) && \text{(Associativity and ST)} \\ \equiv & ((C \vee (A \vee B)) \wedge (C \vee \neg A)) && \text{(Commutativity and ST)} \\ \equiv & (C \vee ((A \vee B) \wedge \neg A)) && \text{(Distributivity)} \\ \equiv & (C \vee (\neg A \wedge (A \vee B))) && \text{(Commutativity und ST)} \\ \equiv & (C \vee ((\neg A \wedge A) \vee (\neg A \wedge B))) && \text{(Distributivity and ST)} \\ \equiv & (C \vee (\neg A \wedge B)) && \text{(Unsatisfiability Law and ST)} \\ \equiv & (C \vee (B \wedge \neg A)) && \text{(Commutativity and ST)} \\ \equiv & ((B \wedge \neg A) \vee C) && \text{(Commutativity)} \end{aligned}$$

## Exercise : Children and Doctors

---

Formalize and show that the two statements are equivalent

- If the child has temperature or has a bad cough and we reach the doctor, then we call him

$$((T \vee C) \wedge R) \Rightarrow D$$

- If the child has temperature, then we call the doctor, provided we reach him, and, if we reach the doctor then we call him, if the child has a bad cough

$$(R \Rightarrow (T \Rightarrow D)) \wedge (C \Rightarrow (R \Rightarrow D))$$



$$((T \vee C) \wedge R) \Rightarrow D$$

$$((T \wedge R) \vee (C \wedge R) \Rightarrow D)$$

$$((T \wedge R) \Rightarrow D) \wedge ((C \wedge R) \Rightarrow D)$$

$$(\neg T \vee \neg R \vee D) \wedge (\neg C \vee \neg R \vee D)$$

$$(\neg R \vee \neg T \vee D) \wedge (\neg C \vee \neg R \vee D)$$

$$(\neg R \vee (T \Rightarrow D) \wedge (\neg C \vee (R \Rightarrow D)$$










$$(R \Rightarrow (T \Rightarrow D)) \wedge (C \Rightarrow (R \Rightarrow D))$$

# Summary

---

- What is propositional logic (PL)?
- Syntax of PL
- Semantics of PL
- Validity and satisfiability
- Semantic equivalence in PL

# Check your understanding

	Valid	Satisfiable	Unsatisfiable
$A$			
$A \vee B$			
$A \vee \neg A$			
$A \wedge \neg A$			
$A \rightarrow \neg A$			
$A \rightarrow B$			
$A \rightarrow (B \rightarrow A)$			
$A \rightarrow (A \rightarrow B)$			
$A \leftrightarrow \neg A$			

# Check your understanding

---

Which of the following statements are true?

	Y/N
If $F$ is valid, then $F$ is satisfiable	
If $F$ is satisfiable, then $\neg F$ is satisfiable	
If $F$ is valid, then $\neg F$ is satisfiable	
If $F$ is unsatisfiable, dann $\neg F$ is valid	