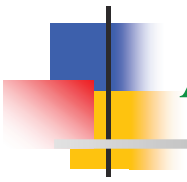


A brief introduction to Logic – part I

Source: www.decision-procedures.org

Modified by Aditya Kanade

(E0 223 – Indian Institute of Science)

Logic in Computer Science

- Logic has a profound impact on computer-science. Some examples:
  - Propositional logic – the foundation of computers and circuitry
  - Databases – query languages
  - Programming languages (e.g. prolog)
  - Type systems
  - Design Validation and verification
  - AI (e.g. inference systems)
  - ...

Logic in Computer Science

- Propositional Logic
- First Order Logic
- Higher Order Logic
- Temporal Logic
- ...
- ...

Modeling with propositional logic

Assignment of frequencies

- n radio stations
- For each 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 satisfy these constraints?
- Q: Which graph problem does this remind you of ?

A Brief Introduction to Logic - Outline

- Propositional Logic :Syntax
- Propositional Logic :Semantics
- Satisfiability and validity
- Normal forms
- Deductive proofs and resolution

Propositional logic

- A proposition – a sentence that can be either true or false.
- Propositions:
  - x is greater than y
  - Noam wrote this letter

Propositional logic: Syntax

- The symbols of the language:
  - Propositional symbols (Prop): A, B, C,…
  - Connectives:
    - $\wedge$ and
    - $\vee$ or
    - $\neg$ not
    - $\rightarrow$ implies
    - $\leftrightarrow$ equivalent to
    - $\oplus$ xor (different than)
    - $\bot$, $\top$ False, True
  - Parenthesis:(,).
- Q1: how many different binary symbols can we define ?
- Q2: what is the minimal number of such symbols?

Formulas

- Grammar of well-formed propositional formulas
  - Formula := prop | (¬Formula) | (Formula o Formula).
  - ... where prop ∈ Prop and o is one of the binary relations

Formulas

- Examples of well-formed formulas:
  - $(\neg A)$
  - $(\neg(\neg A))$
  - $(A \wedge (B \wedge C))$
  - $(A \rightarrow (B \rightarrow C))$

- Correct expressions of Propositional Logic are full of unnecessary parenthesis.

Formulas

- We omit parenthesis whenever we may restore them through operator precedence:
- $\neg$ binds more strictly than $\wedge$, $\vee$, and $\wedge$, $\vee$ bind more strictly than $\rightarrow$, $\leftrightarrow$.
- Thus, we write:

$\neg\neg A$	for	$(\neg(\neg A))$,
$\neg A \wedge B$	for	$((\neg A) \wedge B)$
$A \wedge B \rightarrow C$	for	$((A \wedge B) \rightarrow C)$, …

Propositional Logic: Semantics

- Truth tables define the semantics (=meaning) of the operators
- Convention: 0 = false, 1 = true

p	q	$p \wedge q$	$p \vee q$	$p \rightarrow q$
0	0	0	0	1
0	1	0	1	1
1	0	0	1	0
1	1	1	1	1

Propositional Logic: Semantics

- Truth tables define the semantics (=meaning) of the operators

p	q	$\neg p$	$p \leftrightarrow q$	$p \oplus q$
0	0	1	1	0
0	1	1	0	1
1	0	0	0	1
1	1	0	1	0

Assignments

- Definition: A truth-values assignment, α, is an element of 2^{Prop} (i.e., $\alpha \in 2^{Prop}$).
- In other words, α is a subset of the variables that are assigned true.
- Equivalently, we can see α as a mapping from variables to truth values:
  $\alpha : Prop \mapsto \{0,1\}$
  - Example: α: $\{A \mapsto 0, B \mapsto 1,...\}$

Satisfaction relation ($\models$): intuition

- An assignment can either satisfy or not satisfy a given formula.

- $\alpha \models \phi$ means
  - α satisfies ϕ or
  - ϕ holds at α or
  - α is a model of ϕ
- We will first see an example.
- Then we will define these notions formally.

Example

- Let $\phi = (A \vee (B \rightarrow C))$
- Let $\alpha = \{A \mapsto 0, B \mapsto 0, C \mapsto 1\}$
- Q: Does α satisfy ϕ?
  - (in symbols: does it hold that $\alpha \models \phi$?)

- A: $(0 \vee (0 \rightarrow 1)) = (0 \vee 1) = 1$
  - Hence, $\alpha \models \phi$.

- Let us now formalize an evaluation process.

The satisfaction relation ($\models$): formalities

- $\models$ is a relation: $\models \subseteq (2^{Prop} \times Formula)$
  - Examples:
    - ({a}, a $\vee$ b) // the assignment α = {a} satisfies a $\vee$ b
    - ({a,b}, a $\wedge$ b)

- Alternatively: $\models \subseteq (\{0,1\}^{Prop} \times Formula)$
  - Examples:
    - (01, a $\vee$ b) // the assignment $\alpha = \{a \mapsto 0, b \mapsto 1\}$ satisfies a $\vee$ b
    - (11, a $\wedge$ b)

The satisfaction relation ($\models$): formalities

- $\models$ is defined recursively:
  - $\alpha \models p$ if $\alpha(p) = \text{true}$
  - $\alpha \models \neg\phi$ if $\alpha \not\models \phi$.
  - $\alpha \models \phi_1 \wedge \phi_2$ if $\alpha \models \phi_1$ and $\alpha \models \phi_2$
  - $\alpha \models \phi_1 \vee \phi_2$ if $\alpha \models \phi_1$ or $\alpha \models \phi_2$
  - $\alpha \models \phi_1 \rightarrow \phi_2$ if $\alpha \models \phi_1$ implies $\alpha \models \phi_2$
  - $\alpha \models \phi_1 \leftrightarrow \phi_2$ if $\alpha \models \phi_1$ iff $\alpha \models \phi_2$

From definition to an evaluation algorithm

- Truth Evaluation Problem
  - Given $\phi \in$ Formula and $\alpha \in 2^{AP(\phi)}$, does $\alpha \models \phi$?

```
Eval(φ, α){
   If φ ≡ A, return α(A).
   If φ ≡ (¬φ1) return ¬Eval(φ1, α))
   If φ ≡ (φ1 ○ φ2)
     return Eval(φ1, α) ○ Eval(φ2, α)
   }
```

- Eval uses polynomial time and space.

It doesn't give us more than what we already know...

- Recall our example
  - Let $\phi = (A \vee (B \rightarrow C))$
  - Let $\alpha = \{A \mapsto 0, B \mapsto 0, C \mapsto 1\}$

- $\text{Eval}(\phi,\alpha) = \text{Eval}(A, \alpha) \vee \text{Eval}(B \rightarrow C, \alpha) =$
  $0 \vee \text{Eval}(B, \alpha) \rightarrow \text{Eval}(C, \alpha) =$
  $0 \vee (0 \rightarrow 1) = 0 \vee 1 = 1$

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

We can now extend the truth table to formulas

p	q	$(p \rightarrow (q \rightarrow p))$	$(p \wedge \neg p)$	$p \vee \neg q$
0	0	1	0	1
0	1	1	0	0
1	0	1	0	1
1	1	1	0	1

We can now extend the truth table to formulas

x_1	x_2	x_3	$x_1 \rightarrow (x_2 \rightarrow \neg x_3)$
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Set of assignments

- Intuition: a formula specifies a set of truth assignments.
- Function models: Formula $\mapsto 2^{2^{\mathbf{Prop}}}$
  (a formula $\mapsto$ set of satisfying assignments)
- Recursive definition:
  - $\text{models}(A) = \{\alpha \mid \alpha(A) = 1\}$, $A \in \text{Prop}$
  - $\text{models}(\neg\phi_1) = 2^{\text{Prop}} - \text{models}(\phi_1)$
  - $\text{models}(\phi_1 \wedge \phi_2) = \text{models}(\phi_1) \cap \text{models}(\phi_2)$
  - $\text{models}(\phi_1 \vee \phi_2) = \text{models}(\phi_1) \cup \text{models}(\phi_2)$
  - $\text{models}(\phi_1 \rightarrow \phi_2) = (2^{\text{Prop}} - \text{models}(\phi_1)) \cup \text{models}(\phi_2)$

Example

- models $(A \vee B) = \{\{10\},\{01\},\{11\}\}$

- This is compatible with the recursive definition:

$$\text{models}(A \vee B) =$$
$$\text{models}(A) \cup \text{models}(B) =$$
$$\{\{10\},\{11\}\} \cup \{\{01\},\{11\}\} =$$
$$\{\{10\},\{01\},\{11\}\}$$

Theorem

- Let $\phi \in$ Formula and $\alpha \in 2^{\text{Prop}}$, then the following statements are equivalent:
  1. $\alpha \models \phi$
  2. $\alpha \in$ models(ϕ)

Extension of $\models$ to sets of assignments

- Let $\phi \in$ Formula
- Let T be a set of assignments, i.e., $T \subseteq 2^{2^{Prop}}$
- Definition.

  $T \models \phi$ if $T \subseteq$ models(ϕ)

  - i.e., $\models \subseteq 2^{2^{Prop}} \times$ Formula

Extension of $\models$ to formulas

- $\models \subseteq 2^{\text{Formula}} \times 2^{\text{Formula}}$
- Definition. Let Γ_1, Γ_2 be prop. formulas.

$\Gamma_1 \models \Gamma_2$
iff $\text{models}(\Gamma_1) \subseteq \text{models}(\Gamma_2)$
iff for all $\alpha \in 2^{\text{Prop}}$
if $\alpha \models \Gamma_1$ then $\alpha \models \Gamma_2$

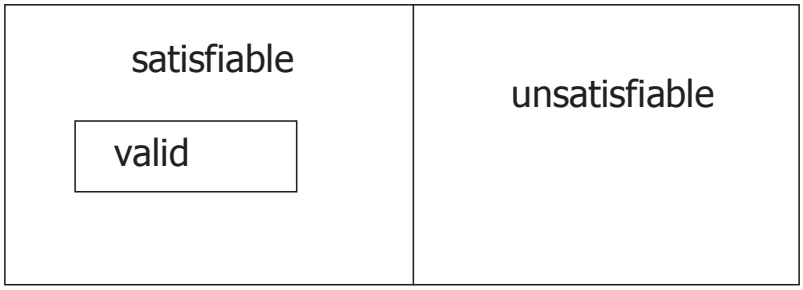
Examples:

$x_1 \wedge x_2 \models x_1 \vee x_2$

$x_1 \wedge x_2 \models x_2 \vee x_3$

Semantic Classification of formulas

- A formula ϕ is called valid if $models(\phi) = 2^{Prop}$. (also called a tautology).
- A formula ϕ is called satisfiable if $models(\phi) \neq \emptyset$.
- A formula ϕ is called unsatisfiable if $models(\phi) = \emptyset$. (also called a contradiction).

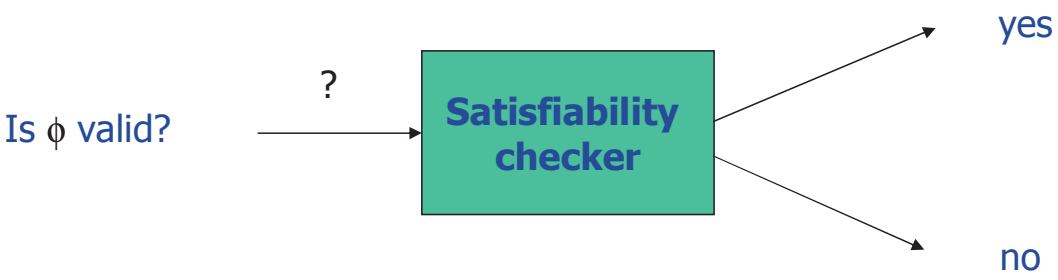


Validity, satisfiability... in truth tables

p	q	$(p \rightarrow (q \rightarrow q))$	$(p \wedge \neg p)$	$p \vee \neg q$
0	0	1	0	1
0	1	1	0	0
1	0	1	0	1
1	1	1	0	1

Characteristics of valid/sat. formulas...

- Lemma
  - A formula ϕ is valid iff $\neg\phi$ is unsatisfiable
  - ϕ is satisfiable iff $\neg\phi$ is not valid



Look what we can do now...

- We can write:
  - $\models \phi$ when ϕ is valid
  - $\not\models \phi$ when ϕ is not valid
  - $\not\models \neg\phi$ when ϕ is satisfiable
  - $\models \neg\phi$ when ϕ is unsatisfiable

Examples

- $(x_1 \wedge x_2) \rightarrow (x_1 \vee x_2)$ is valid
- $(x_1 \vee x_2) \rightarrow x_1$ is satisfiable
- $(x_1 \wedge x_2) \wedge \neg x_1$ is unsatisfiable

Time for equivalences

- Here are some valid formulas:
  - $\models A \wedge 1 \leftrightarrow A$
  - $\models A \wedge 0 \leftrightarrow 0$
  - $\models \neg\neg A \leftrightarrow A$ // The 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 decision problem of formulas

- The decision problem:

> Given a propositional formula ϕ, is ϕ satisfiable ?

- An algorithm that always terminates with a correct answer to this problem is called a **decision procedure** for propositional logic.

Before we solve this problem...

- Q: Suppose we can solve the satisfiability problem... how can this help us?

- A: There are numerous problems in the industry that are solved via the satisfiability problem of propositional logic
  - Logistics...
  - Planning...
  - Electronic Design Automation industry...
  - Cryptography...
  - …

Modeling with propositional logic

Assignment of frequencies

- n radio stations
- For each 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 satisfy these constraints?
- Q: Which graph problem does this remind you of ?

Modeling with propositional logic

- $x_{i,j}$ – station i is assigned frequency j, for 1 <= i <= n, 1 <= j <= k.
  - Every station is assigned at least one frequency:

$$\bigwedge_{i=1}^{n} \bigvee_{j=1}^{k} x_{ij}$$

  - Every station is assigned not more than one frequency:

$$\bigwedge_{i=1}^{n} \bigwedge_{j=1}^{k-1} (x_{ij} \rightarrow \bigwedge_{j<t\leq k} \neg x_{it})$$

  - Close stations are not assigned the same frequency. For each (i,j) in E,

$$\bigwedge_{t=1}^{k} (x_{it} \rightarrow \neg x_{jt})$$