A brief introduction to Logic

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

A Brief Introduction to Logic - Outline

- Propositional Logic :Syntax
- Propositional Logic :Semantics
- Satisfiability and validity
- Normal forms

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:(,).

Formulas

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

Assignments

- Definition: A truth-values assignment, α, is an element of 2^{Prop} (i.e., $\alpha \in 2^{\mathrm{Prop}}$).
- 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:
  $\alpha : \mathrm{Prop} \mapsto \{0,1\}$
  - Example: α: $\{A \mapsto 0, B \mapsto 1, \ldots\}$

Satisfaction relation ($\models$): intuition

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

- $\alpha \models \varphi$ 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 $\alpha \vDash \phi$?)
- A: $(0 \vee (0 \rightarrow 1)) = (0 \vee 1) = 1$
  - Hence, $\alpha \vDash \phi$.
- Let us now formalize an evaluation process.

The satisfaction relation ($\models$): formalities

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

- Alternatively: $\models \subseteq (\{0,1\}^{Prop} \times \text{Formula})$
  - 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 $\alpha(p) = true$
  - $\alpha \models \neg\varphi$ if $\alpha \not\models \varphi$.
  - $\alpha \models \varphi_1 \wedge \varphi_2$ if $\alpha \models \varphi_1$ and $\alpha \models \varphi_2$
  - $\alpha \models \varphi_1 \vee \varphi_2$ if $\alpha \models \varphi_1$ or $\alpha \models \varphi_2$
  - $\alpha \models \varphi_1 \rightarrow \varphi_2$ if $\alpha \models \varphi_1$ implies $\alpha \models \varphi_2$
  - $\alpha \models \varphi_1 \leftrightarrow \varphi_2$ if $\alpha \models \varphi_1$ iff $\alpha \models \varphi_2$

From definition to an evaluation algorithm

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

```
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.

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:
  - $\text{models}(A) = \{\alpha \mid \alpha(A) = 1\}$, $A \in \text{Prop}$
  - $\text{models}(\neg\varphi_1) = 2^{\text{Prop}} - \text{models}(\varphi_1)$
  - $\text{models}(\varphi_1 \wedge \varphi_2) = \text{models}(\varphi_1) \cap \text{models}(\varphi_2)$
  - $\text{models}(\varphi_1 \vee \varphi_2) = \text{models}(\varphi_1) \cup \text{models}(\varphi_2)$
  - $\text{models}(\varphi_1 \rightarrow \varphi_2) = (2^{\text{Prop}} - \text{models}(\varphi_1)) \cup \text{models}(\varphi_2)$

Theorem

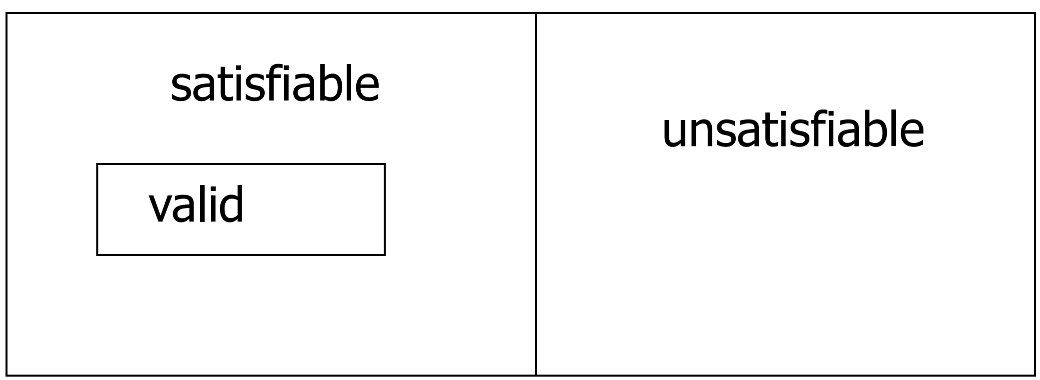
- Let $\varphi \in$ Formula and $\alpha \in 2^{\text{Prop}}$, then the following statements are equivalent:
  1. $\alpha \models \varphi$
  2. $\alpha \in \text{models}(\varphi)$

Semantic Classification of formulas

- A formula φ is called valid if models(φ) = 2^{Prop}.
(also called a tautology).

- A formula φ is called satisfiable if models(φ) ≠ ∅.

- A formula φ is called unsatisfiable if models(φ) = ∅.
(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

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

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.

Two classes of algorithms for validity

- Q: Is φ satisfiable (/$\neg\varphi$ is valid) ?
- Complexity: NP-Complete (the first-ever! – Cook's theorem)

- Two classes of algorithms for finding out:
  1. Enumeration of possible solutions (Truth tables etc).
  2. Deduction

- More generally (beyond propositional logic):
  - Enumeration is possible only in some logics.
  - Deduction cannot necessarily be fully automated.

The satisfiability problem: enumeration

- Given a formula φ, is φ satisfiable?

```
Boolean SAT(φ) {
    B:=false
    for all α ∈ 2^AP(φ)
        B = B ∨ Eval(φ,α)
    end
    return B
}
```

- There must be a better way to do that in practice.

A Brief Introduction to Logic - Outline

- Propositional Logic :Syntax
- Propositional Logic :Semantics
- Satisfiability and validity
- Normal forms

Definitions…

- Definition: A literal is either an atom or a negation of an atom.
- Let $\phi = \neg(A \vee \neg B)$. Then:
  - Atoms: $AP(\phi) = \{A,B\}$
  - Literals: $lit(\phi) = \{A, \neg B\}$

- Equivalent formulas can have different literals
  - $\phi = \neg(A \vee \neg B) = \neg A \wedge B$
  - Now $lit(\phi) = \{\neg A, B\}$

Definitions…

- 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 said to be in Negation Normal Form (NNF) if it only contains $\neg$, $\wedge$ and $\vee$ connectives and only atoms can be negated.

- Examples:
  - $\phi_1 = \neg(A \vee \neg B)$ is not in NNF
  - $\phi_2 = \neg A \wedge B$ is in NNF

Converting to 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 the right
- Example: $\phi = \neg(A \rightarrow \neg B)$
  - Eliminate '$\rightarrow$': $\phi = \neg(\neg A \vee \neg B)$
  - Push negation using De Morgan: $\phi = (\neg\neg A \wedge \neg\neg B)$
  - Use Double negation rule: $\phi = (A \wedge B)$

Disjunctive Normal Form (DNF)

- Definition: A formula is said to be in Disjunctive Normal Form (DNF) if 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.

- Examples
  - $\phi = (A \wedge \neg B \wedge C) \vee (\neg A \wedge D) \vee (B)$ 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:
  - Convert to NNF
  - Distribute disjunctions following the rule:
    $\models A \land (B \lor C) \leftrightarrow ((A \land B) \lor (A \land C))$
- Example:
  - $\phi = (A \lor B) \land (\neg C \lor D) =$
    $((A \lor B) \land (\neg C)) \lor ((A \lor B) \land D) =$
    $(A \land \neg C) \lor (B \land \neg C) \lor (A \land D) \lor (B \land D)$

Conjunctive Normal Form (CNF)

- Definition: A formula is said to be in Conjunctive Normal Form (CNF) if 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 term.

- Examples
  - $\phi = (A \vee \neg B \vee C) \wedge (\neg A \vee D) \wedge (B)$ is in CNF

- CNF is a special case of NNF

Converting to CNF

- Every formula can be converted to CNF:
  - in exponential time and space with the same set of atoms
  - in linear time and space if new variables are added.
    - In this case the original and converted formulas are "equi-satisfiable".
    - This technique is called Tseitin's encoding.

Converting to CNF: the exponential way

CNF(ϕ) {

case

ϕ is a literal: return ϕ

ϕ is $\psi_1 \wedge \psi_2$: return CNF(ψ_1) $\wedge$ CNF(ψ_2)

ϕ is $\psi_1 \vee \psi_2$: return Dist(CNF(ψ_1),CNF(ψ_2))

}

Dist(ψ_1,ψ_2) {

case

ψ_1 is $\phi_{11} \wedge \phi_{12}$: return Dist($\phi_{11}$,$\psi_2$) $\wedge$ Dist(ψ_{12},ψ_2)
ψ_2 is $\phi_{21} \wedge \phi_{22}$: return Dist($\psi_1$,$\phi_{21}$) $\wedge$ Dist(ψ_1,ϕ_{22})
else: return $\psi_1 \vee \psi_2$

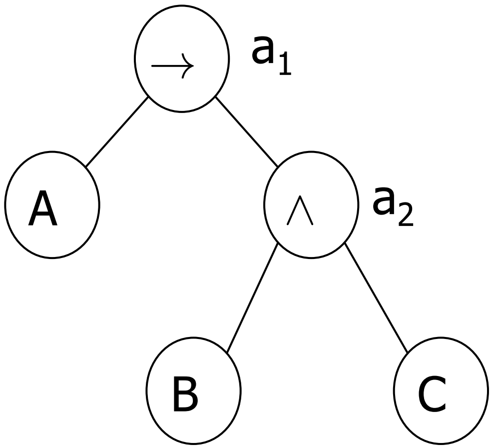
Converting to CNF: the exponential way

- Consider the formula
  $\phi = (x_1 \wedge y_1) \vee (x_2 \wedge y_2)$
- CNF(ϕ)=
  $(x_1 \vee x_2) \wedge$
  $(x_1 \vee y_2) \wedge$
  $(y_1 \vee x_2) \wedge$
  $(y_1 \vee y_2)$

- Now consider: $\phi_n = (x_1 \wedge y_1) \vee (x_2 \wedge y_2) \vee \cdots \vee (x_n \wedge y_n)$
- Q: How many clauses CNF(ϕ) returns ?
- A: 2^n

Converting to CNF: Tseitin's encoding

- Consider the formula $\phi = (A \rightarrow (B \wedge C))$
- The parse tree:



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

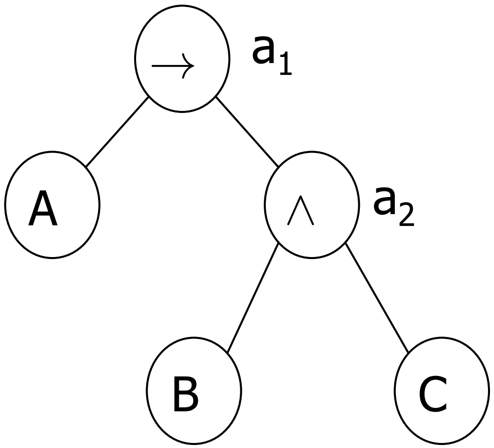
Converting to CNF: Tseitin's encoding

- Need to satisfy:

  $(a_1 \leftrightarrow (A \rightarrow a_2)) \wedge$

  $(a_2 \leftrightarrow (B \wedge C)) \wedge$

  (a_1)



- Each such constraint has a CNF representation with 3 or 4 clauses.

Converting to CNF: Tseitin's encoding

- Need to satisfy:

  $(a_1 \leftrightarrow (A \rightarrow a_2)) \wedge$

  $(a_2 \leftrightarrow (B \wedge C)) \wedge$

  (a_1)

- First: $(a_1 \vee A) \wedge (a_1 \vee \neg a_2) \wedge (\neg a_1 \vee \neg A \vee a_2)$
- Second: $(\neg a_2 \vee B) \wedge (\neg a_2 \vee C) \wedge (a_2 \vee \neg B \vee \neg C)$

Converting to CNF: Tseitin's encoding

- Let's go back to
  $\phi_n = (x_1 \wedge y_1) \vee (x_2 \wedge y_2) \vee \cdots \vee (x_n \wedge y_n)$
- With Tseitin's encoding we need:
  - n auxiliary variables $a_1,\ldots,a_n$.
  - Each adds 3 constraints.
  - Top clause: $(a_1 \vee \cdots \vee a_n)$
- Hence, we have
  - $3n + 1$ clauses, instead of 2^n.
  - $3n$ variables rather than $2n$.

What now?

- Time to solve the decision problem for propositional logic.
  - The only algorithm we saw so far was building truth tables.

Two classes of algorithms for validity

- Q: Is φ valid ?
  - Equivalently: is $\neg\varphi$ satisfiable?
- Two classes of algorithm for finding out:
  1. Enumeration of possible solutions (Truth tables etc).
  2. Deduction

- In general (beyond propositional logic):
  - Enumeration is possible only in some theories.
  - Deduction typically cannot be fully automated.

The satisfiability Problem: enumeration

- Given a formula φ, is φ satisfiable?

```
Boolean SAT(φ) {
   B:=false
   for all α ∈ 2^AP(φ)
      B = B ∨ Eval(φ,α)
   end
   return B
 }
```

- NP-Complete (the first-ever! – Cook's theorem)

A Brief Introduction to Logic - Outline

- Propositional Logic :Syntax
- Propositional Logic :Semantics
- Satisfiability and validity
- Normal forms