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:
    - ∧ and
    - ∨ or
    - ¬ not
    - → implies
    - ↔ equivalent to
    - ⊕ xor (different than)
    - ⊥, ⊤ 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
Examples of well-formed formulas:

- $(\neg A)$
- $(\neg(\neg A))$
- $(A \land (B \land C))$
- $(A \rightarrow (B \rightarrow C))$

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 \( \land, \lor, \) and \( \land, \lor \) bind more strictly than \( \to, \leftrightarrow. \)
- Thus, we write:
  - \( \neg \neg A \) for \( (\neg (\neg A)) \),
  - \( \neg A \land B \) for \( ((\neg A) \land B) \)
  - \( A \land B \to C \) for \( ((A \land B) \to C) \), …
Propositional Logic: Semantics

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

<table>
<thead>
<tr>
<th>p</th>
<th>q</th>
<th>p ( \land ) q</th>
<th>p ( \lor ) q</th>
<th>p ( \rightarrow ) q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
Propositional Logic: Semantics

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

<table>
<thead>
<tr>
<th>p</th>
<th>q</th>
<th>¬p</th>
<th>p ↔ q</th>
<th>p ⊕ q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Assignments

■ Definition: A truth-values assignment, \( \alpha \), is an element of \( 2^\text{Prop} \) (i.e., \( \alpha \in 2^\text{Prop} \)).

■ In other words, \( \alpha \) is a subset of the variables that are assigned true.

■ Equivalently, we can see \( \alpha \) as a mapping from variables to truth values:

\[ \alpha : \text{Prop} \mapsto \{0,1\} \]

■ Example: \( \alpha: \{A \mapsto 0, B \mapsto 1,\ldots\} \)
Satisfaction relation ($\models$): intuition

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

- $\alpha \models \phi$ means
  - $\alpha$ satisfies $\phi$ or
  - $\phi$ holds at $\alpha$ or
  - $\alpha$ is a model of $\phi$

- We will first see an example.
- Then we will define these notions formally.
Example

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

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

- Let us now formalize an evaluation process.
The satisfaction relation ($\models$): formalities

- $\models$ is a relation: $\models \subseteq (2^{\text{Prop}} \times \text{Formula})$

  - Examples:
    - $\{a\}, a \lor b$ // the assignment $\alpha = \{a\}$ satisfies $a \lor b$
    - $\{a,b\}, a \land b$ (alternative)

- Alternatively: $\models \subseteq (\{0,1\}^{\text{Prop}} \times \text{Formula})$

  - Examples:
    - $\{0,1\}, a \lor b$ // the assignment $\alpha = \{a \mapsto 0, b \mapsto 1\}$ satisfies $a \lor b$
    - $\{1\}, a \land b$
The satisfaction relation ($\models$): formalities

$\models$ is defined recursively:

- $\alpha \models p$ if $\alpha(p) = \text{true}$
- $\alpha \models \neg \phi$ if $\alpha \not\models \phi$.
- $\alpha \models \phi_1 \land \phi_2$ if $\alpha \models \phi_1$ and $\alpha \models \phi_2$
- $\alpha \models \phi_1 \lor \phi_2$ 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 $\phi \in \text{Formula}$ and $\alpha \in 2^{\text{AP}(\phi)}$, does $\alpha \models \phi$?

**Eval**($\phi$, $\alpha$) {
  
  If $\phi \equiv A$, return $\alpha(A)$.

  If $\phi \equiv (\neg \phi_1)$ return $\neg \text{Eval}(\phi_1, \alpha)$

  If $\phi \equiv (\phi_1 \circ \phi_2)$

  
  return $\text{Eval}(\phi_1, \alpha) \circ \text{Eval}(\phi_2, \alpha)$

  

  }

- Eval uses polynomial time and space.
It doesn’t give us more than what we already know...

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

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

- Hence, $\alpha \models \phi$. 
We can now extend the truth table to formulas:

<table>
<thead>
<tr>
<th>p</th>
<th>q</th>
<th>( (p \rightarrow (q \rightarrow p)) )</th>
<th>( (p \land \neg p) )</th>
<th>( p \lor \neg q )</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
We can now extend the truth table to formulas:

<table>
<thead>
<tr>
<th>$x_1$</th>
<th>$x_2$</th>
<th>$x_3$</th>
<th>$x_1 \rightarrow (x_2 \rightarrow \neg x_3)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
Set of assignments

- Intuition: a formula specifies a set of truth assignments.

- Function \textbf{models}: Formula \( \mapsto 2^{2^{\text{Prop}}}
\) (a formula \( \mapsto \) set of satisfying assignments)

- Recursive definition:
  - \( \text{models}(A) = \{ \alpha \mid \alpha(A) = 1 \}, A \in \text{Prop} \)
  - \( \text{models}(\lnot \phi_1) = 2^{\text{Prop}} - \text{models}(\phi_1) \)
  - \( \text{models}(\phi_1 \land \phi_2) = \text{models}(\phi_1) \cap \text{models}(\phi_2) \)
  - \( \text{models}(\phi_1 \lor \phi_2) = \text{models}(\phi_1) \cup \text{models}(\phi_2) \)
  - \( \text{models}(\phi_1 \rightarrow \phi_2) = (2^{\text{Prop}} - \text{models}(\phi_1)) \cup \text{models}(\phi_2) \)
Example

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

- This is compatible with the recursive definition:

\[
\text{models}(A \lor B) = \\
\text{models}(A) \cup \text{models}(B) = \\
\{{10},{11}\} \cup \{{01},{11}\} = \\
\{{10},{01},{11}\}
\]
Theorem

Let $\phi \in \text{Formula}$ and $\alpha \in 2^{\text{Prop}}$, then the following statements are equivalent:

1. $\alpha \models \phi$
2. $\alpha \in \text{models}(\phi)$
Extension of $\models$ to sets of assignments

- Let $\phi \in \text{Formula}$

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

- Definition.

  $$T \models \phi \text{ if } T \subseteq \text{models}(\phi)$$

- i.e., $\models \subseteq 2^{2^{\text{Prop}}} \times \text{Formula}$
Extension of $\models$ to formulas

- $\models \subseteq 2^{\text{Formula}} \times 2^{\text{Formula}}$

- Definition. Let $\Gamma_1, \Gamma_2$ 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:

- $x_1 \land x_2 \models x_1 \lor x_2$
- $x_1 \land x_2 \models x_2 \lor x_3$
Semantic Classification of formulas

- A formula $\phi$ is called **valid** if $\text{models}(\phi) = 2^{\text{Prop}}$. (also called a **tautology**).

- A formula $\phi$ is called **satisfiable** if $\text{models}(\phi) \neq \emptyset$.

- A formula $\phi$ is called **unsatisfiable** if $\text{models}(\phi) = \emptyset$. (also called a **contradiction**).
Validity, satisfiability... in truth tables

<table>
<thead>
<tr>
<th>p</th>
<th>q</th>
<th>(p → (q → q))</th>
<th>(p ∧ ¬p)</th>
<th>p ∨ ¬q</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Characteristics of valid/sat. formulas...

- **Lemma**
  - A formula $\phi$ is valid iff $\neg\phi$ is unsatisfiable
  - $\phi$ is satisfiable iff $\neg\phi$ is not valid
Look what we can do now...

- **We can write:**

  - $\models \phi$ when $\phi$ is valid
  - $\not\models \phi$ when $\phi$ is not valid
  - $\not\models \neg \phi$ when $\phi$ is satisfiable
  - $\models \neg \phi$ when $\phi$ is unsatisfiable
Examples

- \((x_1 \land x_2) \rightarrow (x_1 \lor x_2)\) is valid
- \((x_1 \lor x_2) \rightarrow x_1\) is satisfiable
- \((x_1 \land x_2) \land \neg x_1\) is unsatisfiable
Time for equivalences

Here are some valid formulas:

- \( \vdash A \land 1 \leftrightarrow A \)
- \( \vdash A \land 0 \leftrightarrow 0 \)
- \( \vdash \neg\neg A \leftrightarrow A \) // The double-negation rule
- \( \vdash A \land (B \lor C) \leftrightarrow (A \land B) \lor (A \land C) \)

Some more (De-Morgan rules):

- \( \vdash \neg(A \land B) \leftrightarrow (\neg A \lor \neg B) \)
- \( \vdash \neg(A \lor B) \leftrightarrow (\neg A \land \neg B) \)
The decision problem of formulas

The decision problem:

Given a propositional formula $\phi$, is $\phi$ 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...
- …
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 \( 1 \leq i \leq n, 1 \leq j \leq k \).

- 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 \( E \),
  \[
  \bigwedge_{t=1}^{k} (x_{it} \rightarrow \neg x_{jt})
  \]
Two classes of algorithms for validity

- Q: Is $\phi$ satisfiable ($\neg\phi$ 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 $\Phi$, is $\Phi$ satisfiable?

```plaintext
Boolean SAT($\Phi$) {
    B := false
    for all $\alpha \in 2^{AP(\Phi)}$
        $B = B \lor \text{Eval}(\Phi, \alpha)$
    end
    return B
}
```

- There must be a better way to do that in practice.
A Brief Introduction to Logic - Outline

- Brief historical notes on logic
- Propositional Logic : Syntax
- Propositional Logic : Semantics
- Satisfiability and validity
- Modeling with Propositional logic
- Normal forms
- Deductive proofs and resolution
Definitions…

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

- Let $\phi = \neg(\neg(A \lor \neg B))$. Then:
  - **Atoms:** $AP(\phi) = \{A, B\}$
  - **Literals:** $lit(\phi) = \{A, \neg B\}$

- Equivalent formulas can have different literals
  - $\phi = \neg(\neg(A \lor \neg B)) = \neg\neg A \land B$
  - **Now** $lit(\phi) = \{\neg A, B\}$
Definitions…

- Definition: a **term** is a conjunction of literals
  - Example: \((A \land \lnot B \land C)\)

- Definition: a **clause** is a disjunction of literals
  - Example: \((A \lor \lnot B \lor C)\)
Negation Normal Form (NNF)

- Definition: A formula is said to be in Negation Normal Form (NNF) if it only contains $\neg$, $\land$ and $\lor$ connectives and only atoms can be negated.

- Examples:
  - $\phi_1 = \neg(A \lor \neg B)$ is not in NNF
  - $\phi_2 = \neg A \land B$ is in NNF
Converting to NNF

- Every formula can be converted to NNF in linear time:
  - Eliminate all connectives other than $\land$, $\lor$, $\neg$
  - Use De Morgan and double-negation rules to push negations to the right
- Example: $\phi = \neg(A \rightarrow \neg B)$
  - Eliminate ‘$\rightarrow$’: $\phi = \neg(\neg A \lor \neg B)$
  - Push negation using De Morgan: $\phi = (\neg \neg A \land \neg \neg B)$
  - Use Double negation rule: $\phi = (A \land B)$
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
\[ \bigvee \left( \bigwedge_{i} \left( \bigwedge_{j} l_{i,j} \right) \right) \]
where \( l_{i,j} \) is the \( j \)-th literal in the \( i \)-th term.

Examples
- \( \phi = (A \land \neg B \land C) \lor (\neg A \land D) \lor (B) \) 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:
    \[ \vdash A \land (B \lor C) \iff ((A \land B) \lor (A \land C)) \]
- Example:
  - \[ \phi = (A \lor B) \land (\neg C \lor D) = ((A \lor B) \land (\neg C)) \lor ((A \lor B) \land D) = (A \land \neg C) \lor (B \land \neg C) \lor (A \land D) \lor (B \land D) \]
- Q: how many clauses would the DNF have had we started from a conjunction of \( n \) clauses?
Satisfiability of DNF

- Is the following DNF formula satisfiable?
  \[(x_1 \land x_2 \land \neg x_1) \lor (x_2 \land x_1) \lor (x_2 \land \neg x_3 \land x_3)\]

- What is the complexity of satisfiability of DNF formulas?
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
    \[ \bigwedge_i \bigvee_j l_{i,j} \]
    where \( l_{i,j} \) is the \( j \)-th literal in the \( i \)-th term.

- Examples
  - \( \phi = (A \vee \neg B \vee C) \land (\neg A \vee D) \land (B) \) 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

\[
\text{CNF}(\phi) \begin{cases} \\
\text{case} \\
\quad \phi \text{ is a literal: return } \phi \\
\quad \phi \text{ is } \psi_1 \land \psi_2: \text{return } \text{CNF}(\psi_1) \land \text{CNF}(\psi_2) \\
\quad \phi \text{ is } \psi_1 \lor \psi_2: \text{return } \text{Dist(CNF}(\psi_1),\text{CNF}(\psi_2)) \\
\end{cases}
\]

\[
\text{Dist}(\psi_1,\psi_2) \begin{cases} \\
\text{case} \\
\quad \psi_1 \text{ is } \phi_{11} \land \phi_{12}: \text{return } \text{Dist}(\phi_{11},\psi_2) \land \text{Dist}(\psi_{12},\psi_2) \\
\quad \psi_2 \text{ is } \phi_{21} \land \phi_{22}: \text{return } \text{Dist}(\psi_1,\phi_{21}) \land \text{Dist}(\psi_1,\phi_{22}) \\
\quad \text{else: return } \psi_1 \lor \psi_2 \\
\end{cases}
\]
Converting to CNF: the exponential way

- Consider the formula
  \[ \phi = (x_1 \land y_1) \lor (x_2 \land y_2) \]
- **CNF(\(\phi\))** =
  \[ (x_1 \lor x_2) \land \\
  (x_1 \lor y_2) \land \\
  (y_1 \lor x_2) \land \\
  (y_1 \lor y_2) \]

- **Now consider:** \(\phi_n = (x_1 \land y_1) \lor (x_2 \land y_2) \lor \cdots \lor (x_n \land y_n)\)
- **Q:** How many clauses **CNF(\(\phi\))** returns ?
- **A:** \(2^n\)
Converting to CNF: Tseitin’s encoding

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

```
  →  a_1
 ▲   a_2
 ▲   ▲
 A   ∧  B   C
```

- 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:

\[(a_1 \leftrightarrow (A \rightarrow a_2)) \land\]
\[(a_2 \leftrightarrow (B \land C)) \land\]
\[(a_1)\]

Each such constraint has a CNF representation with 3 or 4 clauses.
Converting to CNF: Tseitin’s encoding

- Need to satisfy:
  
  \[(a_1 \leftrightarrow (A \rightarrow a_2)) \land (a_2 \leftrightarrow (B \land C)) \land (a_1)\]

- First: \[(a_1 \lor A) \land (a_1 \lor \neg a_2) \land (\neg a_1 \lor \neg A \lor a_2)\]

- Second: \[(\neg a_2 \lor B) \land (\neg a_2 \lor C) \land (a_2 \lor \neg B \lor \neg C)\]
Converting to CNF: Tseitin’s encoding

- Let’s go back to
  \[ \phi_n = (x_1 \land y_1) \lor (x_2 \land y_2) \lor \cdots \lor (x_n \land y_n) \]

- With Tseitin’s encoding we need:
  - n auxiliary variables \( a_1, \ldots, a_n \).
  - Each adds 3 constraints.
  - Top clause: \( (a_1 \lor \cdots \lor a_n) \)

- 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 $\phi$ valid?
  - Equivalently: is $\neg\phi$ 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 $\phi$, is $\phi$ satisfiable?

**Boolean SAT($\phi$) {**

  B:=false

  for all $\alpha \in 2^{AP(\phi)}$
    
    $B = B \lor \text{Eval}(\phi, \alpha)$

  end

  return B

} **

- NP-Complete (the first-ever! – Cook’s theorem)
A Brief Introduction to Logic - Outline

- Brief historical notes on logic
- Propositional Logic: Syntax
- Propositional Logic: Semantics
- Satisfiability and validity
- Modeling with Propositional logic
- Normal forms
- Deductive proofs and resolution
Deduction requires axioms and Inference rules

- **Inference rules:**

  - **Antecedents**
  - **Consequent**

  (rule-name)

- **Examples:**

  \[
  \begin{array}{c}
  A \rightarrow B \quad B \rightarrow C \\
  \hline
  A \rightarrow C
  \end{array}
  \]  
  (trans)

  \[
  \begin{array}{c}
  A \rightarrow B \quad A \\
  \hline
  B
  \end{array}
  \]  
  (M.P.)
Axioms

- Axioms are inference rules with no antecedents, e.g.,
  
  $A \rightarrow (B \rightarrow A)$
  
  (H1)

- We can turn an inference rule into an axiom if we have ‘$\rightarrow$’ in the logic.

- So the difference between them is not sharp.
A proof uses a given set of inference rules and axioms. This is called the *proof system*. Let $\mathcal{H}$ be a proof system.

- $\Gamma \vdash_{\mathcal{H}} \Phi$ means: there is a proof of $\Phi$ 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 $\text{Trans}$ and $\text{M.P.}$ that we saw earlier.

- Does the following relation holds?

$$a \rightarrow b, b \rightarrow c, c \rightarrow d, d \rightarrow e, a \vdash_{\mathcal{H}} e$$
Deductive proof: example

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

1. \( a \rightarrow b \) \hspace{1cm} \text{premise}
2. \( b \rightarrow c \) \hspace{1cm} \text{premise}
3. \( a \rightarrow c \) \hspace{1cm} 1,2,\text{Trans}
4. \( c \rightarrow d \) \hspace{1cm} \text{premise}
5. \( d \rightarrow e \) \hspace{1cm} \text{premise}
6. \( c \rightarrow e \) \hspace{1cm} 4,5, \text{Trans}
7. \( a \rightarrow e \) \hspace{1cm} 3,6, \text{Trans}
8. \( a \) \hspace{1cm} \text{premise}
9. \( e \) \hspace{1cm} 3,8.\text{M.P.}
Proof graph (DAG)

\[ a \rightarrow b \quad b \rightarrow c \]

\[ c \rightarrow d \quad d \rightarrow e \]

\[ \text{(trans)} \]

\[ a \rightarrow c \]

\[ c \rightarrow e \]

\[ \text{(trans)} \]

\[ a \rightarrow e \]

\[ \text{(M.P.)} \]

\[ e \]

\[ a \]

Roots: premises
Proofs

- The problem: $\vdash$ is a relation defined by syntactic transformations of the underlying proof system.

- For a given proof system $\mathcal{H}$,
  - does $\vdash$ conclude “correct” conclusions from premises?
  - 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 gives us this.
Soundness and completeness

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

- **Soundness** of $\mathcal{H}$: if $\not\vdash_{\mathcal{H}} \phi$ then $\models \phi$

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

- How to prove soundness and completeness?
Example: Hilbert axiom system ($\mathcal{H}$)

- Let $\mathcal{H}$ be (M.P) + the following axiom schemas:

  \begin{align*}
  \text{(H1)} & \quad A \rightarrow (B \rightarrow A) \\
  \text{(H2)} & \quad ((A \rightarrow (B \rightarrow C)) \rightarrow ((A \rightarrow B) \rightarrow (A \rightarrow C))) \\
  \text{(H3)} & \quad (\neg B \rightarrow \neg A) \rightarrow (A \rightarrow B)
  \end{align*}

- $\mathcal{H}$ is sound and complete
To prove soundness of $\mathcal{H}$, prove the soundness of its axioms and inference rules (easy with truth-tables). For example:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>$A \rightarrow (B \rightarrow A)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

Completeness – harder, but possible.
The resolution inference system

The resolution inference rule for CNF:

\[
\frac{(l \lor l_1 \lor \ldots \lor l_n) (\neg l \lor l'_1 \lor \ldots \lor l'_n)}{(l_1 \lor \ldots \lor l_n \lor l'_1 \lor \ldots \lor l'_n)} \quad \text{(Resolution)}
\]

Example:

\[
\frac{(a \lor b) (\neg a \lor c)}{(b \lor c)}
\]
Proof by resolution

- Let $\varphi = (1\ 3) \land (-1\ 2\ 5) \land (-1\ 4) \land (-1\ -4)$
- We’ll try to prove $\varphi \rightarrow (3\ 5)$

Diagram:

- (1 3)
- (-1 2 5)
- (2 3 5) (1 -2) (-1 4) (-1 -4)
- (1 3 5)
- (-1)
- (3 5)
Resolution

- 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 $\varphi = (1\ 3) \land (-1\ 2) \land (-1\ 4) \land (-1\ -4) \land (-3)$
Another system: Natural deduction ($\mathcal{N}$)

- **A** $\quad$ **B**
  
  \[ \frac{}{A \land B} \]  
  (Introduction-and) (I - $\land$)

- **A $\land$ B**

  \[ \frac{}{A} \]  
  (Elimination-1-and) (E1 - $\land$)

- **A $\land$ B**

  \[ \frac{}{B} \]  
  (Elimination-2-and) (E2 - $\land$)

Example

- **Theorem:** \( p \land q, r \models_N q \land r \)
- **Note:** the theorem only claims provability relation. Correctness is implied by the soundness of \( N \).
- **Proof:**

1. \( p \land q \) premise
2. \( r \) premise
3. \( q \) E-2-\( \land \), 1
4. \( q \land r \) I-and, 2,3
More rules for $\neg$

- $\neg\neg A$ (E-double negation)
  
  $A$

- $A$ (I-double negation)
  
  $\neg\neg A$
More rules for $\mathcal{N}$

- **$A \quad A \rightarrow B$**  
  \[ B \]  
  (E - implication)

- **$\neg B \quad A \rightarrow B$**  
  \[ \neg A \]  
  (Modus-Tollens (M.T.))

- Similar to another elimination rule:

- If assuming $p$ allows to prove $q$ then \[ p \rightarrow q \]  
  (I – implication)
Example

- Theorem: \( p \rightarrow q \vdash \neg q \rightarrow \neg p \)
- Proof:
  1. \( p \land q \) premise
  2. \( \neg q \) assumption
  3. \( \neg p \) M.T. 1,2
  4. \( \neg q \rightarrow \neg p \) I-implication

- Note the difference between assumptions and premises. The former needs to be discharged.
- The introduction-implication rule lets us discharge assumptions.
Example

**Theorem:** \( \neg q \rightarrow \neg p \vdash p \rightarrow \neg \neg q \)

**Proof:**

1. \( \neg q \rightarrow \neg p \) premise

2. \( p \) assumption

3. \( \neg \neg q \) M.T. 1,2

4. \( p \rightarrow \neg \neg q \) I-implication
Example

Theorem: $\vdash (q \to r) \to ((\neg q \to \neg p) \to (p \to r))$

Proof:

1. $q \to r$ assumption

2. $\neg q \to \neg p$ assumption

3. $p$ assumption

4. $\neg \neg p$ I-double-negation 3

5. $\neg \neg q$ M.T., 2,4

6. $q$ E-double-negation, 5

7. $r$ E-implication 1,6

8. $p \to r$ I-implication 3-7

9. $(\neg q \to \neg p) \to (p \to r)$ I-implication 2-8

10. $(q \to r) \to ((\neg q \to \neg p) \to (p \to r))$ I-implication 1-9
More rules...

- \( A \lor B \)  (I-or1)
- \( B \lor B \)  (I-or2)

- \((p \lor q) (p \rightarrow r) (q \rightarrow r)\)  (E-or)

\[ r \]
Example

- Theorem: \( p \lor q \vdash q \lor p \)
- Proof:
  1. \( p \lor q \) premise
  2. \( p \) assumption
  3. \( q \lor p \) I-or1
  4. \( p \rightarrow q \lor p \) I-implication
  5. \( q \) assumption
  6. \( q \lor p \) I-or2
  7. \( q \rightarrow q \lor p \) I-implication
  8. \( q \lor p \) E-or