Chapter 2: Propositional Calculus: Formulas, Models, Tableaux

August 22, 2008
2.1 Boolean Operators

- **Boolean type:** T (true), F (false)
- **Boolean operator:** a function on the set \{T,F\}. These operators can be *unary, binary*, etc.
- **Question:** How many *n*-ary Boolean operators are there on \{T,F\}?

\[2^{2^n}\]
• We single out the following five operators:

- (unary)

\( \lor, \land, \rightarrow, \leftrightarrow \) (binary)

• There are other binary operators that are sometimes used e.g. in the theory of Boolean circuits:

\( \oplus \) (XOR, exclusive OR)
\( \uparrow \) (NAND)
\( \downarrow \) (NOR, Sheffer’s stroke)
<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>$p \oplus q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>F</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>$p \uparrow q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>T</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$p$</th>
<th>$q$</th>
<th>$p \downarrow q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>T</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>T</td>
<td>F</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>T</td>
<td>F</td>
</tr>
<tr>
<td>F</td>
<td>F</td>
<td>T</td>
</tr>
</tbody>
</table>
2.2 Propositional Formulas

BNF (Backus-Naur Form) Grammars:

• rules of the form

\[ \text{symbol} ::= \text{symbol}_1 \text{symbol}_2 \ldots \text{symbol}_n, \quad \text{or} \]
\[ \text{symbol} ::= \text{symbol}_1 | \text{symbol}_2 | \ldots | \text{symbol}_n \]

• \textit{symbol} is a non-terminal symbol of the grammar.

• symbols that can never occur on the left-hand side of a grammar rule are called \textbf{terminal} symbols.
• \( \mathcal{P} \) - set of all propositional letters (atoms)

\[
\mathcal{P} = \{ p, q, r, \ldots \}
\]

Definition
A formula in the propositional logic is any string that can be derived from the initial non-terminal \( fml \) using the following BNF rules:
1. \( \text{fml} ::= p, \) for any \( p \in \mathcal{P} \)
2. \( \text{fml} ::= (\neg \text{fml}) \)
3. \( \text{fml} ::= (\text{fml} \lor \text{fml}) \)
4. \( \text{fml} ::= (\text{fml} \land \text{fml}) \)
5. \( \text{fml} ::= (\text{fml} \rightarrow \text{fml}) \)
6. \( \text{fml} ::= (\text{fml} \leftrightarrow \text{fml}) \)

Remark

*If we want to use additional operators such as e.g. \( \oplus, \uparrow, \downarrow, \) etc, the BNF grammar can be easily modified by adding appropriate rules to handle these connectives.*
Example

Derivation of

\[(p \land (r \rightarrow (p \lor (\neg q))))\]

\[
\text{fml} ::= (\text{fml} \land \text{fml})
\]

\[
::= (p \land \text{fml})
\]

\[
::= (p \land (\text{fml} \rightarrow \text{fml}))
\]

\[
::= (p \land (r \rightarrow \text{fml}))
\]

\[
::= (p \land (r \rightarrow (\text{fml} \lor \text{fml})))
\]

\[
::= (p \land (r \rightarrow (p \lor \text{fml})))
\]

\[
::= (p \land (r \rightarrow (p \lor (\neg \text{fml}))))
\]

\[
::= (p \land (r \rightarrow (p \lor (\neg q))))
\]
- **Derivation tree:** tree representing the derivation of the formula using the BNF grammar for propositional logic.

- **Formation tree:** tree representing the structure of the formula; i.e. the tree whose nodes are the connectives occurring in the formula and whose leaves are propositional variables.

**Remark**

*For any formula, the formation tree can be easily obtained from its derivation tree; namely, replace the fml symbol in every node of the derivation tree by the connective used in the rule applied to fml.*
Convention: We can omit writing unnecessary pairs of brackets in a propositional formula, if we introduce the following hierarchy (order of priority) of the Boolean connectives:

1. ¬
2. ∨, ∧
3. →, ↔

Definition
If the propositional formula \( A \) is not an atom (variable), the operator at the root of its formation tree is called the **principal operator** of \( A \).
Theorem

(Structural Induction) To show that some property holds for all propositional formulas $A$, it suffices to show the following:

1. every atom (variable) $p$ has the property.

2. assuming that a formula $A$ has the required property, show that

   $$\neg A$$

   has the property.

3. assuming that the formulas $A$ and $B$ have the required property, show that the formulas

   $$A \lor B, \quad A \land B, \quad A \rightarrow B, \quad A \leftrightarrow B$$

   have the property.
Example
Prove that every formula $A$, formed using BNF form for propositional formulas, is balanced; i.e. $A$ contains the same number of left and right brackets.

Proof.
We will prove this by structural induction.
1. any atom (variable) $p$ is trivially balanced, since it contains no left or right brackets.
2. assume $A$ is a balanced propositional formula, i.e. $A$ contains the same number of left and right brackets. Consider $\neg A$. Since $A$ is balanced, so is $\neg A$.
3. suppose $A$ and $B$ are both balanced formulas. Consider, say, $A \lor B$. Clearly, the number of left brackets in $A \lor B$ is the sum of the left brackets in $A$ and $B$, and similarly for right brackets. Since both $A$ and $B$ are balanced, it is easy to see that this holds for $A \lor B$, too. [Similarly for other three connectives $\land, \rightarrow, \leftrightarrow$.]
2.3 Interpretations

- $\mathcal{P}$ - set of all propositional variables (atoms)

**Definition**

An **assignment** is a function

$$\nu : \mathcal{P} \rightarrow \{T,F\}$$

$\nu$ assigns a **truth value** to any atom in a given formula.

Suppose $\mathcal{F}$ denotes the set of all propositional formulas. We can extend an assignment $\nu$ to a function

$$\nu : \mathcal{F} \rightarrow \{T,F\},$$

which assigns the truth value $\nu(A)$ to any formula $A \in \mathcal{F}$. 
Example

Suppose \( \nu \) is an assignment for which

\[
\nu(p) = F, \quad \nu(q) = T.
\]

If

\[
A = (\neg p \rightarrow q) \leftrightarrow (p \lor q)
\]

what is \( \nu(A) \)?

Solution:

\[
\nu(A) = \nu((\neg p \rightarrow q) \leftrightarrow (p \lor q))
\]

\[
= \nu(\neg p \rightarrow q) \leftrightarrow \nu(p \lor q)
\]

\[
= (\nu(\neg p) \rightarrow \nu(q)) \leftrightarrow (\nu(p) \lor \nu(q))
\]

\[
= (\neg \nu(p) \rightarrow \nu(q)) \leftrightarrow (\nu(p) \lor \nu(q))
\]

\[
= (\neg F \rightarrow \nu(q)) \leftrightarrow (\nu(p) \lor \nu(q))
\]

\[
= (\neg F \rightarrow T) \leftrightarrow (F \lor T)
\]

\[
= (T \rightarrow T) \leftrightarrow (F \lor T)
\]

\[
= T \leftrightarrow T
\]

\[
= T
\]
Theorem
An assignment can be extended to exactly one interpretation. In other words: for a given set of truth values of atoms, the truth value of a formula is uniquely determined.

- In fact: if two assignments agree on all atoms that appear in the formula, the interpretations they induce also agree on that formula.
Suppose
\[ S = \{A_1, A_2, \ldots, A_n\} \]
is a set of formulas and \( \nu \) is an assignment which assigns truth values to all atoms that appear in the set of formulas \( S \). Any interpretation that extends \( \nu \) to all propositional atoms \( \mathcal{P} \) will be called an \textbf{interpretation} for \( S \).

**Example**
The assignment
\[
\nu(p) = F, \quad \nu(q) = T, \quad \nu(r) = T
\]
determines the following interpretation of the set of formulas
\[
S = \{p \lor \neg q, \quad q, \quad p \land r \leftrightarrow (r \rightarrow q)\}
\]
\[
\nu(p \lor \neg q) = F, \quad \nu(q) = T, \quad \nu(p \land r \rightarrow (r \rightarrow q)) = F
\]
2.4 Equivalence and Substitution

Definition
If $A, B \in \mathcal{F}$ are such that

$$v(A) = v(B)$$

for all interpretations $v$, $A$ is (logically) equivalent to $B$.

$$A \equiv B$$

Example

$$\neg p \lor q \equiv p \rightarrow q$$

since both formulas are true in all interpretations except when

$$v(p) = T, \quad v(q) = F$$

and are false for that particular interpretation.
Caution: \equiv \text{ does not} \text{ mean the same thing as } \leftrightarrow:

- \( A \leftrightarrow B \) is a \text{ formula} (syntax)
- \( A \equiv B \) is a \text{ relation} between two \text{ formula} (semantics)

**Theorem**

\( A \equiv B \text{ if and only if } A \leftrightarrow B \text{ is true in every interpretation; i.e. } A \leftrightarrow B \text{ is a tautology.} \)
Definition
A is a subformula of $B$ if it is a formula occurring within $B$; i.e. the formation tree for $A$ is a subtree of the formation tree for $B$.

Example
The subformulas of

$$p \land (r \leftrightarrow p \lor \neg q)$$

are

$$p \land (r \leftrightarrow p \lor \neg q), \quad p, \quad r \leftrightarrow p \lor \neg q, \quad r, \quad p \lor \neg q, \quad \neg q, \quad q$$
Definition
Suppose $A$ is a subformula of $B$, and $A'$ is any formula. Then, we say that $B'$ is a formula that results from substitution of $A'$ for $A$ in $B$, and we write it as

$$B' = B\{A \leftarrow A'\}$$

if we obtain $B'$ from $B$ by replacing all occurrences of $A$ in $B$ with $A'$.

Example
Suppose

$$B = (p \rightarrow q) \leftrightarrow (\neg q \rightarrow \neg p), \quad A = p \rightarrow q, \quad A' = \neg p \lor q$$

Then,

$$B' = B\{A \leftarrow A'\} = (\underline{\neg p} \lor q) \leftrightarrow (\neg q \rightarrow \neg p)$$
**Theorem**

Let $A$ be a subformula of $B$, and let $A'$ be a formula such that $A \equiv A'$. Then

$$B \equiv B\{A \leftarrow A'\}$$

**Proof.**

By induction on the depth of the highest occurrence of the formation tree of $A$ as a subtree of $B$. 

\qed
Logically Equivalent Formulas

\[
\begin{align*}
A & \equiv \neg \neg A \\
A \lor B & \equiv B \lor A \\
(A \lor B) \lor C & \equiv A \lor (B \lor C) \\
A \land (B \lor C) & \equiv (A \land B) \lor (A \land C) \\
A \lor (B \land C) & \equiv (A \lor B) \land (A \lor C) \\
\neg (A \land B) & \equiv \neg A \lor \neg B \\
A \land \text{true} & \equiv A \\
A \land \text{false} & \equiv \neg A
\end{align*}
\]
Example

Simplify

\[ p \lor (\neg p \land q) \]

Solution:

\[
\begin{align*}
p \lor (\neg p \land q) & \equiv (p \lor \neg p) \land (p \lor q) \\
& \equiv T \land (p \lor q) \\
& \equiv p \lor q
\end{align*}
\]
Definition
A set of connectives is **adequate** if it generates all possible Boolean functions.

Example
The usual set of connectives

\[ \{ \neg, \land, \lor, \rightarrow, \leftrightarrow \} \]

is adequate for propositional logic, since every Boolean function can be generated from these five operators. [A nontrivial fact!]
Example

1. The set \( \{\neg, \land, \lor\} \) is adequate.

\[
A \rightarrow B \equiv \neg A \lor B
\]
\[
A \leftrightarrow B \equiv (A \rightarrow B) \land (B \rightarrow A)
\]
\[
\equiv (\neg A \lor B) \land (\neg B \lor A)
\]

2. \( \{\neg, \land\} \) is adequate

We know that \( \{\neg, \land, \lor\} \) is adequate, so it would suffice to show that \( \lor \) can be expressed using \( \neg, \land \) only:

\[
A \lor B \equiv \neg(\neg A \land \neg B)
\]
3. \{\neg, \lor\} is adequate [Exercise.]

4. \{\neg, \rightarrow\} is adequate.

\[
A \lor B \equiv \neg A \rightarrow B \\
A \land B \equiv \neg (A \rightarrow \neg B)
\]
5. \( \{\neg, \leftrightarrow\} \) is not adequate. [This will be proved in the lab.]

**Hint:** Proving the following fact would be useful in order to show inadequacy:
If \( A \) is a formula involving at least two atoms, then the number of truth assignments that make \( A \) true is even and the same is true of the number of truth assignments that make \( A \) false.
6. \(\{\uparrow\}\) is adequate.

It is enough to show that \(\{\uparrow\}\) generates \(\neg\) and \(\wedge\), since we know that these form an adequate set of connectives:

\[
\neg A \equiv A \uparrow A \\
A \wedge B \equiv \neg (A \uparrow B) \\
\equiv (A \uparrow B) \uparrow (A \uparrow B)
\]
2.5 Satisfiability, Validity, and Consequence

Definition
We say that a propositional formula $A$ is **satisfiable** if and only if $v(A) = T$ in some interpretation $v$. Such an interpretation is called a **model** for $A$.

- $A$ is **valid** (or, a **tautology**) if $v(A) = T$, for all interpretations $v$
  $$\models A$$
- $A$ is **unsatisfiable** (or, **contradictory**) if it is false in every interpretation.
- $A$ is not valid (or, **falsifiable**), if we can find some interpretation $v$, such that $v(A) = F$
  $$\not\models A$$
Examples

1. \((p \rightarrow q) \rightarrow (\neg q \rightarrow \neg p)\)
   
   Valid (tautology).

2. \(q \rightarrow (q \rightarrow p)\)
   
   Not valid (take \(v(p) = F, v(q) = T\)), but it is satisfiable (take e.g. \(v(p) = v(q) = T\)).

3. \((p \land \neg p) \lor (q \land \neg q)\)
   
   False (contradiction).
Theorem

(a) A is valid if and only if \( \neg A \) is unsatisfiable.
(b) A is satisfiable if and only if \( \neg A \) is falsifiable.

Definition

Suppose \( \mathcal{V} \) is a set of formulas. An algorithm is a decision procedure for \( \mathcal{V} \) if, given an arbitrary formula \( A \), the algorithm terminates and returns as the answer either

(a) 'yes, \( A \in \mathcal{V} \)'; or
(b) 'no, \( A \notin \mathcal{V} \)'
Main Problem: develop an algorithm which decides whether a propositional formula $A$ is valid or not (So, the set $\mathcal{V}$ in this particular problem is the set of all valid propositional formulas.)

- **Truth-Table Method:** provides a decision algorithm but it is too time-consuming; in general, it requires exponential time for the majority of propositional formulas.
- A different approach: instead of $A$, consider $\neg A$ and try to decide whether $\neg A$ is satisfiable or not.
Definition
Let \( U = \{A_1, A_2, \ldots, A_n\} \) be a set of formulas. We say that \( U \) is **satisfiable** if we can find an interpretation \( \nu \) such that
\[
\nu(A_1) = \nu(A_2) = \ldots = \nu(A_n) = T
\]
Such an interpretation is called a **model** for \( U \). \( U \) is unsatisfiable if no such interpretation exists.
Facts

1. If $U$ is satisfiable, then so is $U - \{A_i\}$ for any $i = 1, 2, \ldots, n$.
2. If $U$ is satisfiable and $B$ is valid, then $U \cup \{B\}$ is also satisfiable.
3. If $U$ is unsatisfiable and $B$ is any formula, $U \cup \{B\}$ is also unsatisfiable.
4. If $U$ is unsatisfiable and some $A_i$ is valid, then $U - \{A_i\}$ is also unsatisfiable.
Definition
Let $U$ be a set of formulas and $A$ a formula. We say that $A$ is a (logical) consequence of $U$, if any interpretation $\nu$ which is a model of $U$ is also a model for $A$.

$$U \models A$$

Example

$$\{p \land r, \neg q \lor (p \land \neg p)\} \models (p \land \neg q) \to r$$

If some interpretation $\nu$ is a model for the set $\{p \land r, \neg q \lor (p \land \neg p)\}$, it must satisfy

$$\nu(p) = \nu(r) = T, \quad \nu(q) = F$$

but in this interpretation, we also have

$$\nu((p \land \neg q) \to r) = T$$
Theorem

1. \( U \models A \) if and only if
\[ \models (A_1 \land A_2 \land \ldots \land A_n) \rightarrow A \]

2. If \( U \models A \), then \( U \cup \{B\} \models A \), for any formula \( B \).

3. If \( U \models A \) and \( B \) is valid, then
\[ U - \{B\} \models A \]
Theories

Definition
A set of formulas \( \mathcal{T} \) is a theory if it closed under logical consequence. This means that, for every formula \( A \), if

\[
\mathcal{T} \models A,
\]

then \( A \in \mathcal{T} \).

- Let \( U \) be a set of formulas. Then, the set of all consequences of \( U \)

\[
\mathcal{T}(U) = \{ A \mid U \models A \}
\]

is called the theory of \( U \). The formulas in \( U \) are called the axioms for the theory \( \mathcal{T}(U) \).
2.6 Semantic Tableaux

- a more efficient method for deciding satisfiability of a propositional formula than using truth-tables.

Definition
A literal is an atom or its negation.
- atom: positive literal
- negation of an atom: negative literal

- \( \{p, \neg p\} \) - complementary pair of literals
- \( \{A, \neg A\} \) - complementary pair of formulas
Example
Consider the formula

\[ A = p \land (\neg q \lor \neg p). \]

When is \( v(A) = T? \)

1 First, we must have

\[ v(p) = T, \quad v(\neg q \lor \neg p) = T \]

2 This splits into two cases; either
   
   (a) \( v(p) = T, \quad v(\neg q) = T; \) or
   (b) \( v(p) = T, \quad v(\neg p) = T. \)

   and the second case is clearly impossible.

So, the truth assignment

\[ v(p) = T, \quad v(q) = F \]

makes \( A \) true, showing that \( A \) is satisfiable.
\[ p \land (\neg q \lor \neg p) \]

\[ p, \neg q \lor \neg p \]

\[ p, \neg q \quad p, \neg p \]

\[ \odot \quad \times \]
General Idea: Given a formula $A$, first transform it into an equivalent formula, which is a disjunction of conjunctions of literals. After this, we can analyze this new form of $A$ to see if we can construct a truth assignment $\nu$, such that $\nu(A) = T$. If there is one, $A$ is satisfiable; if there is no such $\nu$, $A$ is not satisfiable.
Example

Determine if

\[ B = (p \lor q) \land (\neg p \land \neg q) \]

is satisfiable.

\[ \begin{array}{c}
(p \lor q) \land (\neg p \land \neg q) \\
\downarrow \\
p \lor q, \neg p \land \neg q \\
\downarrow \\
p, \neg p \land \neg q & q, \neg p \land \neg q \\
\downarrow & \downarrow \\
p, \neg p, \neg q & q, \neg p, \neg q \\
\times & \times
\end{array} \]

- This is another example of a semantic tableau.
• In order to use this method, we had to rewrite the formula using \( \neg \), \( \lor \), and \( \land \) only.

• The method can be made more general if we can also eliminate the connectives \( \rightarrow \) and \( \leftrightarrow \) within a tableau.

<table>
<thead>
<tr>
<th>( \alpha )</th>
<th>( \alpha_1 )</th>
<th>( \alpha_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \neg \neg A )</td>
<td>( A )</td>
<td></td>
</tr>
<tr>
<td>( A_1 \land A_2 )</td>
<td>( A_1 )</td>
<td>( A_2 )</td>
</tr>
<tr>
<td>( \neg (A_1 \lor A_2) )</td>
<td>( \neg A_1 )</td>
<td>( \neg A_2 )</td>
</tr>
<tr>
<td>( \neg (A_1 \rightarrow A_2) )</td>
<td>( A_1 )</td>
<td>( \neg A_2 )</td>
</tr>
<tr>
<td>( A_1 \leftrightarrow A_2 )</td>
<td>( A_1 \rightarrow A_2 )</td>
<td>( A_2 \rightarrow A_1 )</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( \beta )</th>
<th>( \beta_1 )</th>
<th>( \beta_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \neg (B_1 \land B_2) )</td>
<td>( \neg B_1 )</td>
<td>( \neg B_2 )</td>
</tr>
<tr>
<td>( B_1 \lor B_2 )</td>
<td>( B_1 )</td>
<td>( B_2 )</td>
</tr>
<tr>
<td>( B_1 \rightarrow B_2 )</td>
<td>( \neg B_1 )</td>
<td>( B_2 )</td>
</tr>
<tr>
<td>( \neg (B_1 \leftrightarrow B_2) )</td>
<td>( \neg (B_1 \rightarrow B_2) )</td>
<td>( \neg (B_2 \rightarrow B_1) )</td>
</tr>
</tbody>
</table>
Algorithm (Construction of a Semantic Tableau)

- **INPUT:** formula $A$
- **OUTPUT:** a tableau $T$ for $A$, all of whose leaves are marked as open or closed.

Initially, $T$ is a single node (root) labeled $\{A\}$.

We build the tableau inductively, by choosing an unmarked leaf $l$ which is labeled by a set of formulas $U(l)$, and apply one of the following rules:
1. If \( U(I) \) is just a set of literals, check if it contains a pair of complementary literals. If it does, mark the leaf as closed (\( \times \)); if not, mark it as open (\( \odot \)).

2. If \( U(I) \) is not just a set of literals, choose one formula in \( U(I) \) which is not a literal.
   
   (a) if one of the \( \alpha \)-rules applies, replace \( U(I) \) with \( (U(I) - \{ \alpha \}) \cup \{ \alpha_1, \alpha_2 \} \).

   (b) if one of the \( \beta \)-rules applies, replace \( U(I) \) with two descendent nodes labeled \( (U(I) - \{ \beta \}) \cup \{ \beta_1 \} \) and \( (U(I) - \{ \beta \}) \cup \{ \beta_2 \} \).
Definition
A tableau is said to be completed if its construction terminates; i.e. eventually, all branches end with leaves containing literals only. It is closed if all its leaves are closed; otherwise, we say that the tableau is open.

Theorem
The construction of a semantic tableau for a propositional formula always terminates.

- this construction can be extended to non-atomically closed tableaux: all leaves eventually contain a pair of complementary formulas $A, \neg A$. 
Main Theorem: A completed semantic tableau for a formula $A$ is closed if and only if $A$ is unsatisfiable.

- **Soundness**: If a tableau is closed, then $A$ is unsatisfiable.
- **Completeness**: If $A$ is unsatisfiable, then any tableau for $A$ is closed.
Corollary
A is a satisfiable formula if and only if any tableau for $A$ is open.

Corollary
A is a valid formula (tautology) if and only if a tableau for $\neg A$ is closed.

Corollary
The method of semantic tableaux is a decision procedure for the validity of formulas in propositional logic.

[Stop at Example 2.54 in the textbook.]