Overview

• HW#2: Study! Due in next class (9/19)
• Last lectures
  – Mathematical Induction Proof
  – Strong form of mathematical induction
• Today’s lecture
  – Formal Logic: Propositional Logic

Chapter 2 : Formal Logic
2.1 Propositional Logic

- Crucial and simplest formal language for mathematical reasoning
- Important for program design
- Used for designing electronic circuitry (ENG356)

Propositional Logic is a system based on propositions.

- A proposition is a declarative statement that is either true or false (not both or in-between).
- A proposition has truth value that is either true (T) or false (F).
- Corresponds to 1 and 0 in digital Boolean circuits or set memberships in the membership table.

The Statement/Proposition (Syntax and Semantics)

Example: Elephants are bigger than mice.
- Is this a declarative statement? Yes
- Is this a proposition? Yes
- What is the truth value of the proposition? True

Example: 520 < 111
- Is this a declarative statement? Yes
- Is this a proposition? Yes
- What is the truth value of the proposition? False

Example: y < 111
- Is this a declarative statement? Yes
- Is this a proposition? No
- Its truth value depends on the value of y, but this value is not specified. We call this type of statement a propositional function or open sentence.
Example: 100 is divisible by 10 and 5 < 99.
Is this a declarative statement? Yes
Is this a proposition? Yes
What is the truth value of the proposition? True

Example: Please do not fall asleep.
Is this a declarative statement? No (It is a request)
Is this a proposition? No
What is the truth value of the proposition? N/A

Example: What is your name?
Is this a declarative statement? No (It is a question)
Is this a proposition? No
What is the truth value of the proposition? N/A

**Combining Propositions**: one or more propositions can be combined to form a single **compound proposition** (or **formula**).

---

**Syntax (Grammar) of PL**

- Formally, we denote propositions with letters such as p, q, r, s, P, Q, R (called **propositional symbols** or **propositional variables** of two values, T and F)
- We can also use several **logical operators** aka **logical connectives**.

<table>
<thead>
<tr>
<th>Logical Operator</th>
<th>Symbol(s)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Negation</td>
<td>( \neg )</td>
</tr>
<tr>
<td>Conjunction</td>
<td>( \land )</td>
</tr>
<tr>
<td>Disjunction</td>
<td>( \lor )</td>
</tr>
<tr>
<td>Exclusive-or</td>
<td>( \oplus )</td>
</tr>
<tr>
<td>Implication</td>
<td>( \rightarrow )</td>
</tr>
<tr>
<td>Bicondition</td>
<td>( \leftrightarrow ) or iff,</td>
</tr>
</tbody>
</table>

\( \neg p \), \( p \land q \), \( p \lor q \), \( p \oplus q \), \( p \rightarrow q \), \( p \leftrightarrow q \)
Semantics (Meaning) of PL

- Truth tables can be used to show how each operator mean in terms of the truth values:

1. Negation (NOT) : Unary Operator, Symbol “¬”
   - Negation flips the truth value:
     - True (T) becomes False (F)
     - False (F) becomes True (T)

2. Conjunction (AND) : Binary Operator, Symbol “∧”
   - Both true:
     - P ∧ Q = True (T) if both P and Q are true
       - True (T) ∧ True (T) = True (T)
     - False (F) if either P or Q is false
       - True (T) ∧ False (F) = False (F)
     - False (F) if either P or Q is false
       - False (F) ∧ True (T) = False (F)
     - False (F) if both P and Q are false
       - False (F) ∧ False (F) = False (F)

3. Disjunction (OR) : Binary Operator, Symbol “∨”
   - At least one is true:
     - P ∨ Q = True (T) if either P or Q is true
       - True (T) ∨ True (T) = True (T)
       - True (T) ∨ False (F) = True (T)
       - False (F) ∨ True (T) = True (T)
       - False (F) ∨ False (F) = False (F)

4. Exclusive Or (XOR) : Binary Operator, Symbol “⊕”
   - TV different!
     - P ⊕ Q = True (T) if either P or Q is true, but not both
       - True (T) ⊕ True (T) = False (F)
       - True (T) ⊕ False (F) = True (T)
       - False (F) ⊕ True (T) = True (T)
       - False (F) ⊕ False (F) = False (F)

   - P ⊕ Q = P ∨ Q - P ∧ Q
     - P ∨ Q = True (T) if either P or Q is true
       - True (T) ∨ False (F) = True (T)
       - True (T) ∨ True (T) = True (T)
       - False (F) ∨ True (T) = True (T)
       - False (F) ∨ False (F) = False (F)

     - P ∧ Q = True (T) if both P and Q are true
       - True (T) ∧ True (T) = True (T)
       - True (T) ∧ False (F) = False (F)
       - False (F) ∧ True (T) = False (F)
       - False (F) ∧ False (F) = False (F)

     - P ∨ Q - P ∧ Q = True (T) if either P or Q is true, but not both
       - True (T) ∨ True (T) - True (T) ∧ True (T) = True (T) - True (T) = False (F)
       - True (T) ∨ False (F) - True (T) ∧ False (F) = True (T) - False (F) = True (T)
       - False (F) ∨ True (T) - False (F) ∧ True (T) = False (F) - False (F) = False (F)
       - False (F) ∨ False (F) - False (F) ∧ False (F) = False (F) - False (F) = False (F)
5. Implication (if - then): Binary Operator, Symbol “→”

6. Biconditional (if and only if): Binary Operator, Symbol “↔”

- Propositions and operators can be combined in any way to form a compound proposition.

Example:

- Suppose a compound proposition has \( n \) propositional variables. How many rows are in its truth table? Answer: \( 2^n \)

Each row is called an interpretation which corresponds to one particular combination of truth values for these \( n \) variables, and each variable has two possible values (T and F).
• Example: To take discrete mathematics, you must have taken calculus or an introduction of programming.

\[ (Q \lor R) \rightarrow P \]

\[ P \rightarrow Q \lor R \]

\[ (Q \lor R) \rightarrow P \]

\[ P \rightarrow Q \lor R \]

• Example: When you buy a new car from Acme Motor Company, you get $2000 back in cash or a 2% car loan.

\[ P \rightarrow Q \oplus R \]

Why use XOR here? example of ambiguity of natural languages

• Example: School is closed if more than 2 feet of snow falls or if the wind chill is below -100

\[ Q \lor R \rightarrow P \]

• Only well-formed formula (wff) can form a proposition. It is defined as follows:

  All propositional variables and the constants T & F are wffs.
  If \( \alpha \) and \( \beta \) are wffs, then \( \neg \alpha, \neg \beta, \alpha \land \beta, \alpha \lor \beta, \alpha \rightarrow \beta, \alpha \leftrightarrow \beta \) are wffs.

Note: previously, we called this a compound propositions

• Precedence among operators: \( \neg, \land, \lor, \rightarrow, \leftrightarrow \)

  e.g. ( (P \land Q) \rightarrow R ) is not a wff (i.e. not a proposition).

  [ ( (P \land Q) \lor (R \land S) ) \land \neg R ] is a wff (i.e., is a proposition)
  but is different from [ (P \land Q) \lor (R \land S) \land \neg R ]
• A **tautology** is a wff that is always true. Examples:
  \[ \neg(P \land Q) \iff (\neg P) \lor (\neg Q) \]
  \[ R \lor (\neg R) \]

• A **contradiction** is a wff that is always false. Examples:
  \[ R \land (\neg R) \]
  \[ \neg(\neg(P \land Q) \iff (\neg P) \lor (\neg Q)) \]

• The negation of any tautology is a contradiction, and the negation of any contradiction is a tautology.

• You can use truth table to prove a wff is tautology or contradiction.

• A wff is **valid** if it is true in all interpretations (theorem).

• A wff is **satisfiable** if it is true in some interpretation.

• A wff is **unsatisfiable** if it is true in no interpretation, i.e. valid formulas are tautology and unsatisfiable formulas are contradictions.

• **Definition:** two wff S1 and S2 are said to be **logically equivalent**, if they have the same truth table, or \( S1 \iff S2 \) is a tautology.

• Equivalence can be established by
  – (i) Constructing truth tables (Exhaustive Proof);
  – (ii) Using (tautology) equivalence laws (Direct Proof);

• Some **tautology equivalence laws**: (you can use it without proof)

  1. **commutative** properties:
    \[ (A \lor B) \iff (B \lor A) \quad (A \land B) \iff (B \land A) \]

  2. **associative** properties:
    \[ ((A \lor B) \lor C) \iff (A \lor (B \lor C)) \quad ((A \land B) \land C) \iff (A \land (B \land C)) \]
3. distributive properties:

\[(A \lor (B \land C)) \iff ((A \lor B) \land (A \lor C))\]
\[(A \land (B \lor C)) \iff ((A \land B) \lor (A \land C))\]

4. identity properties:  
\[(A \lor F) \iff A \quad (A \land T) \iff A\]

5. complement properties:  
\[(A \lor \neg A) \iff T \quad (A \land \neg A) \iff F\]

6. De Morgan’s Law:
\[\neg (A \lor B) \iff \neg A \land \neg B\]
\[\neg (A \land B) \iff \neg A \lor \neg B\]

7. Double negative:
\[\neg (\neg A) \iff A\]

8. Rewriting implication:
\[(A \rightarrow B) \iff (\neg A \lor B)\]

9. Contraposition:
\[(A \rightarrow B) \iff (\neg B \rightarrow \neg A)\]

10. Conditional proof:
\[(A \rightarrow (B \rightarrow C)) \iff ((A \land B) \rightarrow C)\]

Note: More tautology equivalences can be found in text book

---

• Example : P = Today is Monday, Q = I’ll go to London. 
Today is Monday or I’ll go to London, but not both. 
P \oplus Q
If and only if today is not Monday then I’ll go to London.
\[\neg P \iff Q\]

• The propositions P \oplus Q and \neg P \iff Q are **logically equivalent**, since they have the same truth table, or put it in another way, (P \oplus Q) \iff(\neg P \iff Q) is always true (tautology / valid formula).
• Example 1: Show that \((P \rightarrow Q)\) and \((\neg P \lor Q)\) are logically equivalent

\[
\begin{array}{c|c|c|c|c}
P & Q & \neg P & P \rightarrow Q & \neg P \lor Q \\
T & T & F & T & T \\
T & F & F & F & T \\
F & T & T & F & T \\
F & F & T & T & T \\
\end{array}
\]

\((P \land Q) \rightarrow (P \lor Q)\)

\[
\leftrightarrow \quad \neg (P \land Q) \lor (P \lor Q) \quad \text{(by Rewriting Impl. law or example 1 above)}
\]

\[
\leftrightarrow \quad (\neg P \lor \neg Q) \lor (P \lor Q) \quad \text{(by DeMorgan law)}
\]

\[
\leftrightarrow \quad (\neg P \lor P) \lor (\neg Q \lor Q) \quad \text{(by Commutative and Associative laws)}
\]

\[
\leftrightarrow \quad T \lor T \quad \text{(by Complement law)}
\]

\[
\leftrightarrow \quad T \quad \text{(trivial)}
\]