# Mathematical Foundations of Computing
## Propositional Calculus : 4

N Geetha
AM & CS
PSG College of Technology

# Using Logical Equivalences: Example 1

- Logical equivalences can be used to construct additional logical equivalences
- Example: Show that $(p \wedge q) \rightarrow q$ is a tautology

| | | |
|---|---|---|
| 0. | $(p \wedge q) \rightarrow q$ | |
| 1. | $\equiv \neg(p \wedge q) \vee q$ | Conditional Law |
| 2. | $\equiv (\neg p \vee \neg q) \vee q$ | De Morgan's Law |
| 3. | $\equiv \neg p \vee (\neg q \vee q)$ | Associative Law |
| 4. | $\equiv \neg p \vee 1$ | Negation Law |
| 5. | $\equiv 1$ | Domination Law |

# Using Logical Equivalences: Example 2

- Eg (Exercise 17)*Rosen: Show that $\neg(p \leftrightarrow q) \equiv (p \leftrightarrow \neg q)$
- It helps to start with the 2nd proposition sometimes $(p \leftrightarrow \neg q)$

| | | |
|---|---|---|
| 0. | $(p \leftrightarrow \neg q)$ | |
| 1. | $\equiv (p \rightarrow \neg q) \wedge (\neg q \rightarrow p)$ | Bi-conditional Law |
| 2. | $\equiv (\neg p \vee \neg q) \wedge (q \vee p)$ | Conditional Law |
| 3. | $\equiv \neg(\neg((\neg p \vee \neg q) \wedge (q \vee p)))$ | Double negation Law |
| 4. | $\equiv \neg(\neg(\neg p \vee \neg q) \vee \neg(q \vee p))$ | De Morgan's Law |
| 5. | $\equiv \neg((p \wedge q) \vee (\neg q \wedge \neg p))$ | De Morgan's Law |
| 6. | $\equiv \neg((p \vee \neg q) \wedge (p \vee \neg p) \wedge (q \vee \neg q) \wedge (q \vee \neg p))$ | Distribution Law |
| 7. | $\equiv \neg((p \vee \neg q) \wedge (q \vee \neg p))$ | Identity Law |
| 8. | $\equiv \neg((q \rightarrow p) \wedge (p \rightarrow q))$ | Conditional Law |
| 9. | $\equiv \neg(p \leftrightarrow q)$ | Bi-conditional Law |

# Using Logical Equivalences: Example 3

- Show that $\neg(q \to p) \vee (p \wedge q) \equiv$ q

0. $\neg(q \to p) \vee (p \wedge q)$
1. $\equiv \neg(\neg q \vee p) \vee (p \wedge q)$ Conditional Law
2. $\equiv (\neg(\neg\, q) \wedge \neg p) \vee (p \wedge q)$ De Morgan's
3. $\equiv (\, q) \wedge \neg p) \vee (p \wedge q)$ Involution
4. $\equiv (q \wedge \neg p) \vee (q \wedge p)$ Commutative
5. $\equiv q \wedge (\neg p \vee p)$ Distributive Law
6. $\equiv q \wedge T$ Negation Law
7. $\equiv q$ Identity Law

# Usefulness of Logic

- Logic is more precise than natural language
  - You may have cake or ice cream.
    - Can I have both?
  - If you buy your air ticket in advance, it is cheaper.
    - Are there not cheap last-minute tickets?
- For this reason, logic is used for hardware and software <u>specification or verification</u>
  - Given a set of logic statements,
  - One can decide whether or not they are <u>satisfiable</u> (i.e., consistent), although this is a costly process…

# 1. Bitwise Operations

- Computers represent information as bits (binary digits)
- A bit string is a sequence of bits
- The length of the string is the number of bits in the string
- Logical connectives can be applied to bit strings of equal length
- Example 0110 1010 1101

| | |
|---|---|
| | 0101 0010 1111 |
| Bitwise OR | 0111 1010 1111 |
| Bitwise AND | ... |
| Bitwise XOR | … |

# 2. Logic in TCS

- **What is SAT?** SAT is the problem of determining whether or not a sentence in propositional logic (PL) is satisfiable.
  - **Given**: a PL sentence
  - **Question**: Determine whether or not it is satisfiable
- Characterizing SAT as an NP-complete problem (complexity class) is at the foundation of Theoretical Computer Science.
- What is a PL sentence? What does satisfiable mean?

# Logic in TCS: A Sentence in PL

- A Boolean variable is a variable that can have a value 1 or 0. Thus, Boolean variable is a proposition.
- A term is a Boolean variable
- A literal is a term or its negation
- A clause is a disjunction of literals
- A sentence in PL is a conjunction of clauses
- Example: $(a \vee b \vee \neg c \vee \neg d) \wedge (\neg b \vee c) \wedge (\neg a \vee c \vee d)$
- A sentence in PL is satisfiable iff
  - we can assign a truth value
  - to each Boolean variables
  - such that the sentence evaluates to true (i.e., holds)

# SAT in TCS

- Problem
  - **Given**: A sentence in PL (a complex proposition), which is
    - Boolean variables connected with logical connectives
    - Usually, as a conjunction of clauses (CNF = Conjunctive Normal Form)
  - **Question**:
    - Find an assignment of truth values [0|1] to the variables
    - That makes the sentence true, i.e. the sentence holds

# 3. Logic in Programming: Example 1

- Say you need to define a conditional statement as follows:
  - Increment x if the following condition holds

    $(x > 0 \text{ and } x < 10) \text{ or } x=10$
- You may try: `If (0<x<10 OR x=10) x++;`
- Can’t be written in C++ or Java
- How can you modify this statement by using logical equivalence
- Answer: `If (x>0 AND x<=10) x++;`

# Logic in Programming: Example 2

- Say we have the following loop

```
While
   ((i<size AND A[i]>10) OR
    (i<size AND A[i]<0) OR
    (i<size AND (NOT (A[i]!=0 AND NOT (A[i]>=10)))))
```

- Is this a good code? Keep in mind:
  - Readability
  - Extraneous code is inefficient and poor style
  - Complicated code is more prone to errors and difficult to debug
  - Solution?

# Logic in Programming: Example 2

- the loop

```
While
   ((i<size AND A[i]>10) OR
    (i<size AND A[i]<0) OR
    (i<size AND (NOT (A[i]!=0 AND NOT (A[i]>=10)))))
```

- Now, using logical equivalences, simplify it!
- Using De Morgan's Law and Distributive Law

```
While ((i<size) AND
        ((A[i]>10 OR A[i]<0) OR
         (A[i]==0 OR A[i]>=10)))
```

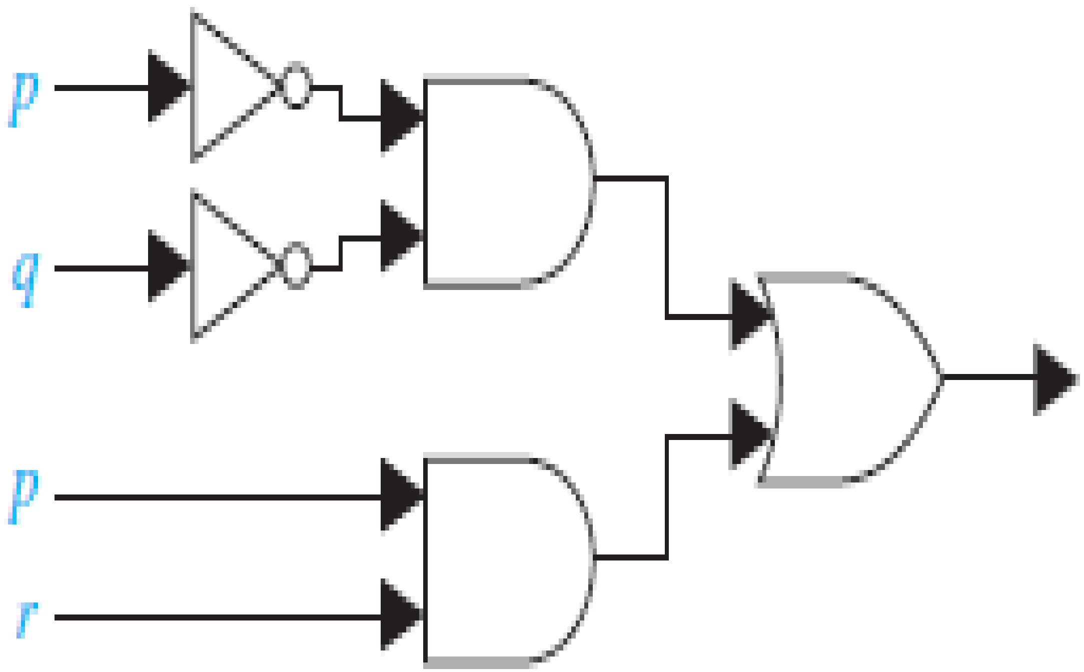
- Notice the ranges of the 4 conditions of `A[i]`

```
While ((i<size) AND (A[i]>=10 OR A[i]<=0))
```

# 4. Sudoku

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | 2 | 9 | | | | 4 | | |
| | | | 5 | | | 1 | | |
| | 4 | | | | | | | |
| | | | | 4 | 2 | | | |
| 6 | | | | | | | 7 | |
| 5 | | | | | | | | |
| 7 | | | 3 | | | | | 5 |
| | 1 | | | 9 | | | | |
| | | | | | | | 6 | |

# 5. Simplification of logic circuits



# Normal Forms

- 1. To determine whether a given compound expression A(p1,p2,p3,…pn) (where p1, p2, .. pn are variables) is
  - a tautology
  - a contradiction
  - atleast satisfiable
- Convert to a normal form and decide
- 2. Whether two given compound propositions A, B are logically equivalent, reduce A and B to some standard forms called normal forms and then decide
- Two types DNF – disjunctive normal form, CNF – conjunctive normal form

# Are $\neg(p \vee (\neg p \wedge q))$ and $(\neg p \wedge \neg q)$ equivalent?

| | |
|---|---|
| $\neg(p \vee (\neg p \wedge q))$ | |
| $\Leftrightarrow \neg p \wedge \neg(\neg p \wedge q)$ | DeMorgan |
| $\Leftrightarrow \neg p \wedge (\neg\neg p \vee \neg q)$ | DeMorgan |
| $\Leftrightarrow \neg p \wedge (p \vee \neg q)$ | Double Negation |
| $\Leftrightarrow (\neg p \wedge p) \vee (\neg p \wedge \neg q)$ | Distribution |
| $\Leftrightarrow (p \wedge \neg p) \vee (\neg p \wedge \neg q)$ | Commutative |
| $\Leftrightarrow F \vee (\neg p \wedge \neg q)$ | And Contradiction |
| $\Leftrightarrow (\neg p \wedge \neg q) \vee F$ | Commutative |
| $\Leftrightarrow (\neg p \wedge \neg q)$ | Identity |

# Are $\neg(p \vee (\neg p \wedge q))$ and $(\neg p \wedge \neg q)$ equivalent?

- Even though both are expressed with only $\wedge$, $\vee$, and $\neg$, it is still hard to tell without doing a proof.
- What we need is a unique representation of a compound proposition that uses $\wedge$, $\vee$, and $\neg$.
- This unique representation is the Normal Form.

# Disjunctive Normal Form

- A product of the variables and their negations is called an **elementary product**
- **Eg. p, ~p, p ^ ~p, ~p ^ q, ~p ^ ~q, p ^ q**

- A sum of the variables and their negations is called an **elementary sum**
- **Eg. p, ~p, q,  p V ~q, ~p V q, ~p V ~q, p V q, q V ~q**

# Disjunctive Normal Form

- A compound proposition which consists of a sum of elementary products and which is equivalent to a given proposition is called disjunctive normal form (DNF) of the given proposition
- **Eg. (~p Vq), (~p ^ q) V (p ^ q)**

# Conjunctive Normal Form

- A compound proposition which consists of a product of elementary sums and which is equivalent to a given proposition is called conjunctive normal form (CNF) of the given proposition
- **Eg. (p ^q), (~p v q) ^ (~p v ~q)**

# Procedure to convert to CNF or DNF

- 1. Eliminate $\rightarrow$, $\leftrightarrow$ by replacing
  - ✓ $p \rightarrow q \equiv \sim p \vee q$
  - ✓ $p \leftrightarrow q \equiv (p \wedge q) \vee (\sim p \wedge \sim q)$
  - ✓ $\equiv (\sim p \vee q) \wedge (\sim q \vee p)$
- 2. If the negation is present before a formula, apply Demorgan's laws.
  - ✓ $\sim(p \vee q) \equiv \sim p \wedge \sim q$
  - ✓ $\sim(p \wedge q) \equiv \sim p \vee \sim q$
- 3. If necessary, the distributive and idempotent laws are applied.
  - ✓ $p \wedge (q \vee r) \equiv (p \wedge q) \vee (p \wedge r)$; $p \vee (q \wedge r) \equiv (p \vee q) \wedge (p \vee r)$
  - ✓ $(p \wedge p) \equiv p$; $(p \vee p) \equiv p$
- 4. If there is an elementary product which is equivalent to F in DNF, it is omitted. If there is an elementary sum which is equivalent to T in CNF, it is omitted. (Identity laws)
  - ✓ $(p \vee F) \equiv p$; $(p \wedge T) \equiv p$

# Find DNF

- Eg1.
- $\neg(\neg(p \leftrightarrow q) \wedge r)$
- $\equiv \neg(\neg[(\mathrm{p} \wedge \mathrm{q}) \vee (\neg p \wedge \neg q)] \wedge r)$ bi*conditional*
- $\equiv [(\mathrm{p} \wedge \mathrm{q}) \vee (\neg p \wedge \neg q)] \vee \neg r)$ *DeMorgan*
- $\equiv (\mathrm{p} \wedge \mathrm{q}) \vee (\neg p \wedge \neg q) \vee \neg r)$ *Associative*

- Eg2: (p v (~p $\rightarrow$ (q v (q $\rightarrow$ ~r))
- Eg3: p ^ ~(q ^ r) v (p $\rightarrow$q)

# Find CNF

- Eg1.
- $\neg((\neg p \rightarrow \neg q) \wedge \neg r)$
- $\equiv \neg((\neg\neg p \vee \neg q) \wedge \neg r)$ *conditional*
- $\equiv \neg((p \vee \neg q) \wedge \neg r)$ *Involution*
- $\equiv \neg(p \vee \neg q) \vee \neg\neg r$ *DeMorgan*
- $\equiv \neg(p \vee \neg q) \vee r$ *Involution*
- $\equiv (\neg p \wedge \neg\neg q) \vee r$ *DeMorgan*
- $\equiv (\neg p \wedge q) \vee r$ *Involution*
- $\equiv (\neg p \vee r) \wedge (q \vee r)$ Distributive
- Eg2: (p ^ ~(q ^ r)) v (p $\rightarrow$ q)
- Eg3: [q v (p ^ q)] ^ ~[ (p v r) ^ q]

# Normal Forms. Contd.

- **MINTERM** :Given a number of variables, the products in which each variable appears exactly once either in T or F form, but not in both. (Also known as a standard product term)
- Eg. P, Q
- Possible minterms: P^Q, P^~Q, ~P^Q, ~P^~Q
- Possible minterms for P,Q,R

- A function can be written as a sum of minterms, which is referred to as a minterm expansion or a standard sum of products.

# Normal Forms

- **MAXTERM** :Given a number of variables, the sum in which each variable appears exactly once either in T or F form, but not in both. (Also known as a standard sum term)
- Eg. P, Q
- Possible maxterms: PVQ, PV~Q, ~PVQ, ~PV~Q
- Possible maxterms for P,Q,R ?

- A function can be written as a product of maxterms, which is referred to as a maxterm expansion or a standard product of sums.
- Maxterms are duals of minterms

# Normal Forms

- **PDNF** :A formula consisting of disjunction of minterms in the variables only and equivalent to the given formula is known as its **Principal disjunctive normal form or Sum of products canonical form of the given formula**
- Eg. (P ^ ~Q) V (~P ^ ~Q)

- **PCNF** :A formula consisting of conjunction of maxterms in the variables only and equivalent to the given formula is known as its **Principal conjunctive normal form or Product of sums canonical form of the given formula**
- Eg. (~PVQ) ^ (PV~Q)

# Normal Forms

- **Procedure to obtain PDNF of a formula**
- 1. Obtain a DNF
- 2. To get the minterms in the disjunction, the missing factors are introduced through the complement law (P V ~P Ξ T) and then applying the distributive law
- 3. Identical minterms are deleted

- Similar procedure for PCNF

# Properties of minterms

- There are $2^n$ minterms for n Boolean variables. These minterms can be generated from the binary numbers from 0 to $2^n -1$
- Any Boolean function can be expressed as a logical sum of minterms.
- The complement of a function contains those minterms not included in the original function.
- A function that contains all $2^n$ minterms is equal to a logical 1.

# Find PCNF

- Eg1.
- $(p \leftrightarrow q)$
- $(p \rightarrow q) \wedge (q \rightarrow p)$ bi*conditional*
- $\equiv (\sim p \vee q) \wedge (\sim q \vee p)$ *conditional*

- Eg2: $(q \rightarrow p) \wedge (\sim p \wedge q)$
- Eg3: $(\sim q \rightarrow r) \wedge (q \leftrightarrow p)$

# Find PDNF

- Eg1.
- $(p \vee \neg q)$
- $\equiv (p \wedge (q \vee \neg q)) \vee (\neg q \wedge (p \vee \neg p))$ *complement*
- $\equiv (p \wedge q) \vee (p \wedge \neg q) \vee (\neg q \wedge p) \vee (\neg q \wedge \neg p)$ dist.
- $\equiv (p \wedge q) \vee (p \wedge \neg q) \vee (\neg q \wedge \neg p)$ idempotent

- Eg2: $(p \wedge \sim(q \wedge r)) \vee (p \to q)$
- Eg3: $[q \vee (p \wedge q)] \wedge \sim[(p \vee r) \wedge q]$

# Find PCNF from PDNF

- If the PDNF of a formula A is known, then PDNF of ~A will consist of the disjunction of the remaining minterms which are not included in the PDNF of A
- To get the PCND of A, we use A ⇔ ~(~A) and apply DeMorgan's laws to the PDNF of ~A repeatedly
- PDNF of A : $(p \vee \neg q)$
- $\equiv (p \wedge q) \vee (p \wedge \neg q) \vee (\neg q \wedge \neg p)$
- PDNF of ¬A is $\neg(p \vee \neg q)$ : Remaining minterm
- $(\neg p \wedge q)$
- PCNF of A : $(p \vee \neg q)$ is $\neg(\neg p \wedge q) \Leftrightarrow (p \vee \neg q)$

# Find PDNF to prove equivalence

- $(p \rightarrow (q \rightarrow p)) \equiv (\neg p \rightarrow (p \rightarrow \neg q))$

- Find PDNF of LHS
- Find PDNF of RHS and compare