# CompSci 102
# Discrete Math for Computer Science

January 19, 2012

Prof. Rodger

Most Slides are modified from Rosen

# Announcements

- Read for next time Chap. 1.4-1.6
- Recitation 1 is tomorrow
- Homework will be posted by Friday

- Today more logic

# Classwork problem from last time

Each inhabitant of a remote village always tells the truth or always lies. A villager will only give "yes" or "no" response to a question a tourist asks.

Suppose you are a tourist visiting this area and come to a fork in the road. One branch leads to the ruins you want to visit; the other leads deep into the jungle.

A villager is standing at the fork in the road. What one question can you ask the villager to determine which branch to take?

# Precedence of Logical operators

| Operator | Precedence |
|----------|------------|
| ¬ | 1 |
| ∧ | 2 |
| ∨ | 3 |
| → | 4 |
| ↔ | 5 |

Example: $p \lor \neg q \land r \to s \lor q$

# Translating English Sentences

- Steps to convert an English sentence to a statement in propositional logic
  - Identify atomic propositions and represent using propositional variables.
  - Determine appropriate logical connectives
- "If I go to Harry's or to the country, I will not go shopping."
  - $p$: I go to Harry's
  - q: I go to the country.
  - $r$:  I will go shopping.

# Example

**Problem:** Translate the following sentence into propositional logic:

"You can access the Internet from campus only if you are a computer science major or you are not a freshman."

# System Specifications

- System and Software engineers take requirements in English and express them in a precise specification language based on logic.

  **Example**: Express in propositional logic:

  "The automated reply cannot be sent when the file system is full"

  **Solution**: One possible solution: Let $p$ denote "The automated reply can be sent" and $q$ denote "The file system is full."

# Consistent System Specifications

**Definition**: A list of propositions is *consistent* if it is possible to assign truth values to the proposition variables so that each proposition is true.

**Exercise**: Are these specifications consistent?

- "The diagnostic message is  stored in the buffer or it is retransmitted."
- "The diagnostic message is not stored in the buffer."
- "If the diagnostic message is stored in the buffer, then it is retransmitted."

- What if "The diagnostic message is not retransmitted is added."

# Logic Puzzles

Raymond Smullyan (Born 1919)

- An island has two kinds of inhabitants, *knights*, who always tell the truth, and *knaves*, who always lie.
- You go to the island and meet A and B.
  - A says "The two of us are both knights"
  - B says "A is a Knave."

  **Example**: What are the types of A and B?

# Tautologies, Contradictions, and Contingencies

- A  tautology is a proposition which is always true.
  - Example: $p \vee \neg p$
- A  *contradiction* is a proposition which is always false.
  - Example: $p \wedge \neg p$
- A  *contingency* is a proposition which is neither a tautology nor a contradiction, such as  $p$

| $P$ | $\neg p$ | $p \vee \neg p$ | $p \wedge \neg p$ |
|---|---|---|---|
| T | F | T | F |
| F | T | T | F |

# Logically Equivalent

- Two compound propositions p and q are logically equivalent if  $p \leftrightarrow q$  is a tautology.
- We write this as $p \Leftrightarrow q$  or as $p \equiv q$ where $p$ and $q$ are compound propositions.
- Two compound propositions $p$ and $q$ are equivalent if and only if the columns in a truth table giving their truth values agree.
- This truth table show $\neg p \vee q$ is equivalent to $p \rightarrow q$.

| $p$ | $q$ | $\neg p$ | $\neg p \vee q$ | $p \rightarrow q$ |
|---|---|---|---|---|
| T | T | F | T | T |
| T | F | F | F | F |
| F | T | T | T | T |
| F | F | T | T | T |

# De Morgan's Laws

$$\neg(p \wedge q) \equiv \neg p \vee \neg q$$

$$\neg(p \vee q) \equiv \neg p \wedge \neg q$$

Augustus De Morgan  1806-1871

This truth table shows that De Morgan's Second Law holds.

| $p$ | $q$ | $\neg p$ | $\neg q$ | $(p \vee q)$ | $\neg(p \vee q)$ | $\neg p \wedge \neg q$ |
|---|---|---|---|---|---|---|
| T | T | F | F | T | F | F |
| T | F | F | T | T | F | F |
| F | T | T | F | T | F | F |
| F | F | T | T | F | T | T |

# Key Logical Equivalences

- Identity Laws:
$$p \wedge T \equiv p \qquad p \vee F \equiv p$$

- Domination Laws:
$$p \vee T \equiv T \qquad p \wedge F \equiv F$$

- Idempotent laws:
$$p \vee p \equiv p \qquad p \wedge p \equiv p$$

- Double Negation Law:
$$\neg(\neg p) \equiv p$$

- Negation Laws:
$$p \vee \neg p \equiv T \qquad p \wedge \neg p \equiv F$$

# Key Logical Equivalences (*cont*)

- Commutative Laws: $p \vee q \equiv q \vee p$ , $p \wedge q \equiv q \wedge p$

- Associative Laws:
$$(p \wedge q) \wedge r \equiv p \wedge (q \wedge r)$$
$$(p \vee q) \vee r \equiv p \vee (q \vee r)$$

- Distributive Laws:
$$(p \vee (q \wedge r) \equiv (p \vee q)) \wedge (p \vee r)$$
$$(p \wedge (q \vee r)) \equiv (p \wedge q) \vee (p \wedge r)$$

- Absorption Laws: $p \vee (p \wedge q) \equiv p \quad p \wedge (p \vee q) \equiv p$

# More Logical Equivalences

**TABLE 7** Logical Equivalences Involving Conditional Statements.

$$p \rightarrow q \equiv \neg p \vee q$$
$$p \rightarrow q \equiv \neg q \rightarrow \neg p$$
$$p \vee q \equiv \neg p \rightarrow q$$
$$p \wedge q \equiv \neg(p \rightarrow \neg q)$$
$$\neg(p \rightarrow q) \equiv p \wedge \neg q$$
$$(p \rightarrow q) \wedge (p \rightarrow r) \equiv p \rightarrow (q \wedge r)$$
$$(p \rightarrow r) \wedge (q \rightarrow r) \equiv (p \vee q) \rightarrow r$$
$$(p \rightarrow q) \vee (p \rightarrow r) \equiv p \rightarrow (q \vee r)$$
$$(p \rightarrow r) \vee (q \rightarrow r) \equiv (p \wedge q) \rightarrow r$$

**TABLE 8** Logical Equivalences Involving Biconditional Statements.

$$p \leftrightarrow q \equiv (p \rightarrow q) \wedge (q \rightarrow p)$$
$$p \leftrightarrow q \equiv \neg p \leftrightarrow \neg q$$
$$p \leftrightarrow q \equiv (p \wedge q) \vee (\neg p \wedge \neg q)$$
$$\neg(p \leftrightarrow q) \equiv p \leftrightarrow \neg q$$

# Constructing New Logical Equivalences

- We can show that two expressions are logically equivalent by developing a series of logically equivalent statements.
- To prove that $A \equiv B$ we produce a series of equivalences beginning with A and ending with B.

$$A \equiv A_1$$
$$\vdots$$
$$A_n \equiv B$$

- Keep in mind that whenever a proposition (represented by a propositional variable) occurs in the equivalences listed earlier, it may be replaced by an arbitrarily complex compound proposition.

# Equivalence Proofs

**Example**: Show that $\neg(p \vee (\neg p \wedge q))$

is logically equivalent to $\neg p \wedge \neg q$

# Equivalence Proofs

**Example**: Show that $(p \wedge q) \rightarrow (p \vee q)$

is a tautology.

**Solution**:

# Propositional Satisfiability

- A compound proposition is *satisfiable* if there is an assignment of truth values to its variables that make it true. When no such assignments exist, the compound proposition is *unsatisfiable*.

- A compound proposition is unsatisfiable if and only if its negation is a tautology.

# Questions on Propositional Satisfiability

**Example**: Determine the satisfiability of the following compound propositions:

$$(p \vee \neg q) \wedge (q \vee \neg r) \wedge (r \vee \neg p)$$

$$(p \vee q \vee r) \wedge (\neg p \vee \neg q \vee \neg r)$$

$$(p \vee \neg q) \wedge (q \vee \neg r) \wedge (r \vee \neg p) \wedge (p \vee q \vee r) \wedge (\neg p \vee \neg q \vee \neg r)$$

# Satisfiability problem

- First CS problem to be shown NP-Complete
  - Problems that take too much time to solve….
  - Cook 1971
  - Math professor at UC Berkeley – now U Toronto
- Start of the area: Complexity theory
- Many problems now shown NP-Complete

# Notation

$\bigvee_{j=1}^{n} p_j$ is used for $p_1 \vee p_2 \vee \ldots \vee p_n$

$\bigwedge_{j=1}^{n} p_j$ is used for $p_1 \wedge p_2 \wedge \ldots \wedge p_n$

Needed for the next example.

# Sudoku

- A **Sudoku puzzle** is represented by a 9×9 grid made up of nine 3×3 subgrids, known as **blocks**. Some of the 81 cells of the puzzle are assigned one of the numbers 1,2, …, 9.
- The puzzle is solved by assigning numbers to each blank cell so that every row, column and block contains each of the nine possible numbers.
- Example



# Encoding as a Satisfiability Problem

- Let $p(i,j,n)$ denote the proposition that is true when the number $n$ is in the cell in the $i$th row and the $j$th column.
- There are $9 \times 9 \times 9 = 729$ such propositions.
- In the sample puzzle $p(5,1,6)$ is true, but $p(5,j,6)$ is false for $j = 2,3,\ldots 9$

# Encoding (cont)

- For each cell with a given value, assert $p(d,j,n)$, when the cell in row $i$ and column $j$ has the given value.

- Assert that every row contains every number.

$$\bigwedge_{i=1}^{9} \bigwedge_{n=1}^{9} \bigvee_{j=1}^{9} p(i,j,n)$$

- Assert that every column contains every number.

$$\bigwedge_{j=1}^{9} \bigwedge_{n=1}^{9} \bigvee_{i=1}^{9} p(i,j,n)$$

# Encoding (cont)

- Assert that each of the 3 x 3 blocks contain every number.

$$\bigwedge_{r=0}^{2} \bigwedge_{s=0}^{2} \bigwedge_{n=1}^{9} \bigwedge_{i=1}^{3} \bigvee_{j=1}^{3} p(3r+i, 3s+j, n)$$

- Assert that no cell contains more than one number. Take the conjunction over all values of $n$, $n'$, $i$, and j, where each variable ranges from 1 to 9 and $n \neq n'$,

  of $\qquad p(i,j,n) \rightarrow \neg p(i,j,n')$

# Solving Satisfiability Problems

- To solve a  Sudoku puzzle, we need to find an assignment of truth values to the 729 variables of the form  $p(i,j,n)$ that makes the conjunction of the assertions true. Those variables that are assigned T yield a solution to the puzzle.
- A truth table can always be used to determine the satisfiability of a compound proposition. But this is too complex even for modern computers for large problems.
- There has been much work on developing efficient methods for solving satisfiability problems as many practical problems can be translated into satisfiability problems.

# Propositional Logic Not Enough

- If we have:
  - "All men are mortal."
  - "Socrates is a man."
- Does it follow that "Socrates is mortal?"
- Can't  be represented in propositional logic. Need a language that talks about objects, their properties, and their relations.
- Later we'll see how to draw inferences.

# Introducing Predicate Logic

- Predicate logic uses the following new features:
  - Variables: $x, y, z$
  - Predicates: $P(x), M(x)$
  - Quantifiers (*to be covered in a few slides*):
- *Propositional functions* are a generalization of propositions.
  - They contain variables and a predicate, e.g., $P(x)$
  - Variables can be replaced by elements from their *domain*.

# Propositional Functions

- Propositional functions become propositions (and have truth values) when their variables are each replaced by a value from the *domain* (or *bound* by a quantifier, as we will see later).
- The statement *P(x)* is said to be the value of the propositional function *P* at *x*.
- For example, let *P(x)* denote "$x > 0$" and the domain be the integers. Then:
  
  P(-3)   is false.
  
  P(0)   is false.
  
  P(3)   is true.
- Often the domain is denoted by *U*. So in this example *U* is the integers.

# Examples of Propositional Functions

- Let "$x + y = z$" be denoted by $R(x, y, z)$ and *U* (for all three variables) be the integers. Find these truth values:
  
  R(2,-1,5)
  
  R(3,4,7)
  
  $R(x, 3, z)$
- Now let "$x - y = z$" be denoted by $Q(x, y, z)$, with U as the integers. Find these truth values:
  
  Q(2,-1,3)
  
  Q(3,4,7)
  
  $Q(x, 3, z)$

# Compound Expressions

- Connectives from propositional logic carry over to predicate logic.
- If *P(x)* denotes "$x > 0$," find these truth values:
  
  P(3) ∨ P(-1)
  
  P(3) ∧ P(-1)
  
  P(3) → P(-1)
  
  P(3) → P(-1)
- Expressions with variables are not propositions and therefore do not have truth values. For example,
  
  P(3) ∧ P(*y*)
  
  P(*x*) → P(*y*)
- When used with quantifiers (to be introduced next), these expressions (propositional functions) become propositions.

# Quantifiers

Charles Peirce (1839-1914)

- We need *quantifiers* to express the meaning of English words including *all* and *some*:
  - "All men are Mortal."
  - "Some cats do not have fur."
- The two most important quantifiers are:
  - *Universal Quantifier*, "For all," symbol: ∀
  - *Existential Quantifier*, "There exists," symbol: ∃
- We write as in ∀x P(x) and ∃x P(x).
- ∀x P(x) asserts P(x) is true for <u>every</u> x in the *domain*.
- ∃x P(x) asserts P(x) is true for <u>some</u> x in the *domain*.
- The quantifiers are said to bind the variable x in these expressions.

# Universal Quantifier

∀x P(x) is read as "For all x, P(x)" or "For every x, P(x)"

**Examples**:

1) If *P(x)* denotes "$x > 0$" and *U* is the integers, then ∀x P(x) is false.
2) If *P(x)* denotes "$x > 0$" and *U* is the positive integers, then ∀x P(x) is true.
3) If *P(x)* denotes "x is even" and *U* is the integers, then ∀ x P(x) is false.

# Existential Quantifier

- ∃x P(x) is read as "For some x, P(x)", or as "There is an x such that P(x)," or "For at least one x, P(x)."

**Examples**:

1. If *P(x)* denotes "$x > 0$" and *U* is the integers, then ∃x P(x) is true. It is also true if U is the positive integers.
2. If *P(x)* denotes "$x < 0$" and *U* is the positive integers, then ∃x P(x) is false.
3. If *P(x)* denotes "x is even" and *U* is the integers, then ∃x P(x) is true.

# Uniqueness Quantifier

- ∃!x P(x) means that P(x) is true for <u>one and only one</u> x in the universe of discourse.
- This is commonly expressed in English in the following equivalent ways:
  - "There is a unique x such that P(x)."
  - "There is one and only one x such that P(x)"
- Examples:
  1. If *P(x)* denotes "$x + 1 = 0$" and U is the integers, then ∃!x P(x) is true.
  2. But if *P(x)* denotes "$x > 0$," then ∃!x P(x) is false.
- The uniqueness quantifier is not really needed as the restriction that there is a unique x such that P(x) can be expressed as:

$$\exists x \, (P(x) \wedge \forall y \, (P(y) \to y = x))$$

# Thinking about Quantifiers

- When the domain of discourse is finite, we can think of quantification as looping through the elements of the domain.
- To evaluate $\forall x\, P(x)$ loop through all $x$ in the domain.
  - If at every step P($x$) is true, then $\forall x\, P(x)$ is true.
  - If at a step P($x$) is false, then $\forall x\, P(x)$ is false and the loop terminates.
- To evaluate $\exists x\, P(x)$ loop through all $x$ in the domain.
  - If at some step, P($x$) is true, then $\exists x\, P(x)$ is true and the loop terminates.
  - If the loop ends without finding an $x$ for which P($x$) is true, then $\exists x\, P(x)$ is false.
- Even if the domains are infinite, we can still think of the quantifiers this fashion, but the loops will not terminate in some cases.

# Properties of Quantifiers

- The truth value of $\exists x\, P(x)$ and $\forall x\, P(x)$ depend on both the propositional function $P(x)$ and on the domain $U$.
- **Examples**:
  1. If $U$ is the positive integers and $P(x)$ is the statement "$x < 2$", then $\exists x\, P(x)$ is true, but $\forall x\, P(x)$ is false.
  2. If $U$ is the negative integers and $P(x)$ is the statement "$x < 2$", then both $\exists x\, P(x)$ and $\forall x\, P(x)$ are true.
  3. If $U$ consists of 3, 4, and 5, and $P(x)$ is the statement "$x > 2$", then both $\exists x\, P(x)$ and $\forall x\, P(x)$ are true. But if $P(x)$ is the statement "$x < 2$", then both $\exists x\, P(x)$ and $\forall x\, P(x)$ are false.

# Precedence of Quantifiers

- The quantifiers $\forall$ and $\exists$ have higher precedence than all the logical operators.
- For example, $\forall x\, P(x) \lor Q(x)$ means $(\forall x\, P(x)) \lor Q(x)$
- $\forall x\, (P(x) \lor Q(x))$ means something different.
- Unfortunately, often people write $\forall x\, P(x) \lor Q(x)$ when they mean $\forall x\, (P(x) \lor Q(x))$.

# Translating from English to Logic

**Example 1**: Translate the following sentence into predicate logic: "Every student in this class has taken a course in Java."

**Solution**:

First decide on the domain $U$.

**Solution 1**: If $U$ is all students in this class, define a propositional function J(x) denoting "x has taken a course in Java" and translate as $\forall x\, J(x)$.

**Solution 2**: But if $U$ is all people, also define a propositional function S(x) denoting "x is a student in this class" and translate as $\forall x\, (S(x) \to J(x))$.

$\forall x\, (S(x) \land J(x))$ is not correct. What does it mean?

# Translating from English to Logic

**Example 2**: Translate the following sentence into predicate logic: "Some student in this class has taken a course in Java."

**Solution**:

First decide on the domain $U$.

   **Solution 1**: If $U$ is all students in this class, translate as
   $$\exists x\, J(x)$$

   **Solution 1**: But if $U$ is all people, then translate as
   $\exists x\, (S(x) \wedge J(x))$
   $\exists x\, (S(x) \rightarrow J(x))$ is not correct. What does it mean?

# Returning to the Socrates Example

- Introduce the propositional functions *Man(x)* denoting "*x* is a man" and *Mortal(x)* denoting "*x* is mortal." Specify the domain as all people.
- The two premises are: $\forall x\, Man(x) \rightarrow Mortal(x)$
$$Man(Socrates)$$
- The conclusion is: $Mortal(Socrates)$

# Equivalences in Predicate Logic

- Statements involving predicates and quantifiers are *logically equivalent* if and only if they have the same truth value
  - for every predicate substituted into these statements and
  - for every domain of discourse used for the variables in the expressions.
- The notation $S \equiv T$ indicates that $S$ and $T$ are logically equivalent.
- **Example**: $\forall x\, \neg\neg S(x) \equiv \forall x\, S(x)$

# Thinking about Quantifiers as Conjunctions and Disjunctions

- If the domain is finite, a universally quantified proposition is equivalent to a conjunction of propositions without quantifiers and an existentially quantified proposition is equivalent to a disjunction of propositions without quantifiers.
- If $U$ consists of the integers 1,2, and 3:
$$\forall x P(x) \equiv P(1) \wedge P(2) \wedge P(3)$$
$$\exists x P(x) \equiv P(1) \vee P(2) \vee P(3)$$

- Even if the domains are infinite, you can still think of the quantifiers in this fashion, but the equivalent expressions without quantifiers will be infinitely long.

# Negating Quantified Expressions

- Consider $\forall x\, J(x)$

  "Every student in your class has taken a course in Java."
  Here $J(x)$ is "x has taken a course in calculus" and
  the domain is students in your class.

- Negating the original statement gives "It is not the case that every student in your class has taken Java." This implies that "There is a student in your class who has not taken calculus."

  Symbolically $\neg \forall x\, J(x)$ and $\exists x\, \neg J(x)$ are equivalent

# Negating Quantified Expressions (*continued*)

- Now Consider $\exists x\, J(x)$

  "There is a student in this class who has taken a course in Java."
  Where $J(x)$ is "x has taken a course in Java."

- Negating the original statement gives "It is not the case that there is a student in this class who has taken Java." This implies that "Every student in this class has not taken Java"

  Symbolically $\neg \exists x\, J(x)$ and $\forall x\, \neg J(x)$ are equivalent

# De Morgan's Laws for Quantifiers

- The rules for negating quantifiers are:

| TABLE 2 De Morgan's Laws for Quantifiers. | | | |
|---|---|---|---|
| Negation | Equivalent Statement | When Is Negation True? | When False? |
| $\neg \exists x\, P(x)$ | $\forall x\, \neg P(x)$ | For every $x$, $P(x)$ is false. | There is an $x$ for which $P(x)$ is true. |
| $\neg \forall x\, P(x)$ | $\exists x\, \neg P(x)$ | There is an $x$ for which $P(x)$ is false. | $P(x)$ is true for every $x$. |

- The reasoning in the table shows that:

$$\neg \forall x\, P(x) \equiv \exists x\, \neg P(x)$$

$$\neg \exists x\, P(x) \equiv \forall x\, \neg P(x)$$

- These are important. You will use these.