$$\sqrt{1+\sqrt{2+\sqrt{3+\sqrt{4....}}}}$$

$$1-1+1-1+1......... = ?$$

$$\sqrt{1+\sqrt{2+\sqrt{3+\sqrt{4....}}}}$$

Discrete mathematics

The Foundations: Logic and Proofs

$$\exists_{x\in\Re}\exists_{y\in\Re}(x = y)$$

$$\forall_x(\Re \mid x)$$

$$\sum_{x=1}^{\infty}\frac{1}{x} = ?$$

$$\sum_{x=1}^{\infty} x = ?$$

RIZOAN TOUFIQ

ASSISTANT PROFESSOR

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

RAJSHAHI UNIVERSITY OF ENGINEERING & TECHNOLOGY

Predicates and Quantifiers

Section 1.4

Section Summary

- Predicates
  - Variables
- Quantifiers
  - Universal Quantifier
  - Existential Quantifier
    - The Uniqueness Quantifier
- Quantifiers with Restricted Domains
- Precedence of Quantifiers
- Binding Variables
- Logical Equivalences Involving Quantifiers

Section Summary

- Negating Quantifiers
  - De Morgan's Laws for Quantifiers
- Translating English to Logic
- Using Quantifiers in System Specifications
- Examples from Lewis Carroll
- Logic Programming (*optional*)

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).

A proposition, → "x is greater than 3"

x → **variable**
is greater than 3 → **predicate (P)**
P(x)→ "x is greater than 3"

The statement P(x) is also said to be the value of the propositional function P at x.

Examples

- 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) → **T/F/Not a proposition?**

  Solution: F

  R(3,4,7) → **T/F/Not a proposition?**

  Solution: T

  R(x, 3, z) → **T/F/Not a proposition?**

  Solution: Not a Proposition

- 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) → **T/F/Not a proposition?**

  Solution: T

  Q(3,4,7) → **T/F/Not a proposition?**

  Solution: F

  Q(x, 3, z) → **T/F/Not a proposition?**

  Solution: Not a Proposition

Compound Expressions

- Connectives from propositional logic carry over to predicate logic.
- If $P(x)$ denotes "$x > 0$," find these truth values:

  $P(3) \vee P(-1)$ **Solution**: T

  $P(3) \wedge P(-1)$ **Solution**: F

  $P(3) \rightarrow P(-1)$ **Solution**: F

  $P(3) \rightarrow \neg P(-1)$ **Solution**: T
- Expressions with variables are not propositions and therefore do not have truth values. For example,

  $P(3) \wedge P(y)$

  $P(x) \rightarrow P(y)$
- When used with **quantifiers** (to be introduced next), these expressions (propositional functions) become propositions.

Preconditions and postconditions

- **Predicates** are also **used** to establish the correctness of computer programs.
- The statements that describe valid input are known as **preconditions**
- The conditions that the output should satisfy when the program has run are known as **postconditions**

Consider the following program, designed to interchange the values of two variables x and y.

temp := x
x:=y
y := temp

Find predicates that we can use as the **precondition** and the **postcondition** to verify the correctness of this program. Then explain how to use them to verify that for all valid input the program does what is intended.

Preconditions and postconditions

Precondition → P(x,y) predicate → "x = a and y = b,"
Postcondition → Q(x,y) predicate → "x = b and y = a,"

verify

Suppose that the **precondition** P(x,y) holds. "x = a and y = b" is true.

First step:

temp:=x

Holds x = a, temp = a, and y = b

Second Step:

x:=y

Holds x = b, temp = a, and y = b

Third Step:

y:=temp

Holds x = b, temp = a, and y = a

The **postcondition** Q(x, y) holds, that is, the statement "x = b and y = a" is true.

Quantifiers

Charles
Peirce
(1839-1914)

Quantifier represent for → All / Some

- “All men are Mortal.”
- “Some cats do not have fur.”

The two most important quantifiers are:

Universal Quantifier, “For all,” **symbol**: $\forall$
Existential Quantifier, “There exists,” **symbol**: $\exists$

We write as in $\forall x\, P(x)$ and $\exists x\, P(x)$.
$\forall x\, P(x)$ asserts $P(x)$ is true for <u>every</u> x in the *domain*.
$\exists x\, P(x)$ asserts $P(x)$ is true for <u>some</u> x in the *domain*.

Universal Quantifier

- $\forall x \, P(x)$ is **read** as "**For all *x*, P(*x*)**" or "**For every *x*, P(*x*)**"

  Examples:

  If $P(x)$ denotes "$x > 0$" and U is the integers, then $\forall x \, P(x) = ?$

  Solution: false.

  If $P(x)$ denotes "$x > 0$" and U is the positive integers, then $\forall x \, P(x) =?$

  Solution: true.

  If $P(x)$ denotes "x is even" and U is the integers, then $\forall x \, P(x) = ?$

  Solution: false.

Existential Quantifier

$\exists x\, P(x)$ 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:

If $P(x)$ denotes "$x > 0$" and U is the integers, then $\exists x\, P(x) = ?$

Solution: true.

If $P(x)$ denotes "$x < 0$" and U is the positive integers, then $\exists x\, P(x) =?$

Solution: false.

If $P(x)$ denotes "x is even" and U is the integers, then $\exists x\, P(x) = ?$

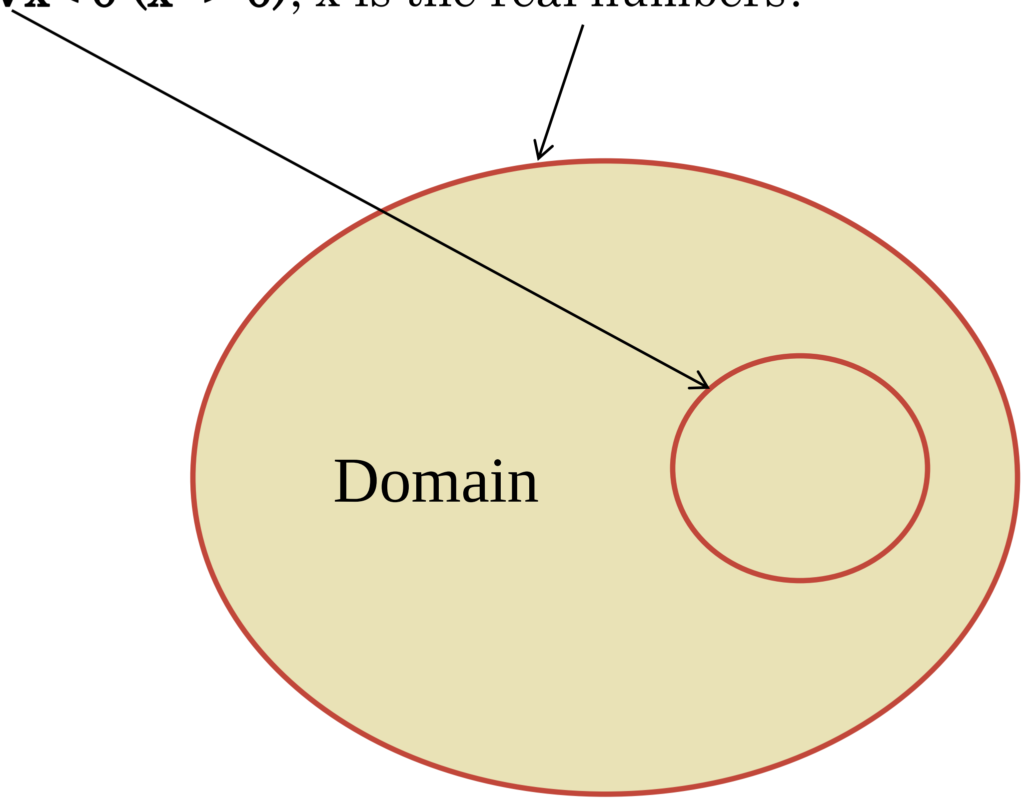
Solution: true.

Uniqueness Quantifier

- $\exists! x\, P(x)$ means that $P(x)$ is true for **one and only one** 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:
  - If $P(x)$ denotes "$x + 1 = 0$" and U is the integers, then $\exists! x\, P(x)$ =?
    Solution: true.
  - But if $P(x)$ denotes "$x > 0$," then $\exists! x\, P(x)$ =?
    Solution: false.

Quantifiers with Restricted Domains

- Used to restrict the domain of a quantifier.
- Examples:
  - **$\forall x < 0\ (x^2 > 0)$**, x is the real numbers?



Quantifiers with Restricted Domains

What do the statements $\forall x < 0\,(x^2 > 0)$, $\forall y \neq 0\,(y^3 \neq 0)$, and $\exists z > 0\,(z^2 = 2)$ mean, where the domain in each case consists of the real numbers?

"The square of a negative real number is positive."
→**Equivalent Statement:** $\forall x(x < 0 \rightarrow x^2 > 0)$

"The cube of every nonzero real number is nonzero."
→**Equivalent Statement :** $\forall y(y \neq 0 \rightarrow y^3 \neq 0)$.

"There is a positive square root of 2."
→**Equivalent Statement :** $\exists z(z > 0 \wedge z^2 = 2)$

Precedence of Quantifiers

The **quantifiers $\forall$ and $\exists$** have **higher** precedence than all logical operators from propositional calculus.

Binding Variables

→When a quantifier is used on the variable x, we say that this occurrence of the variable is **bound.**
→An occurrence of a variable that is not bound by a quantifier or set equal to a particular value is said to be **free**.

$$\exists x(x + y = 1)$$

- The variable x is **bound** by the existential quantification $\exists x$,
- The variable y is **free**

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.

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 $\forall x\ (S(x) \rightarrow J(x))$.

$\forall x\ (S(x) \wedge J(x))$ **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 2: But if U is all people, then translate as $\exists x\, (S(x) \land J(x))$

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

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**: $\forall x(P(x) \wedge Q(x)) \equiv \forall x P(x) \wedge \forall x Q(x)$.

Thinking about Quantifiers as Conjunctions and Disjunctions

- If U consists of the integers 1,2, and 3:

$$\forall x P(x) \equiv P(1) \wedge P(2) \wedge P(3)$$

$$\exists x P(x) \equiv P(1) \vee P(2) \vee P(3)$$

Negating Quantified Expressions

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

  "Every student in your class has taken a course in Java."

  – $J(x)$ is "x has taken a course in Java"

  – 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 Java."

- The negation of $\forall x\, J(x)$ *is* $\exists x\, \neg J(x)$

  Symbolically $\neg\forall x\, J(x) \equiv \exists x\, \neg J(x)$

De Morgan's Laws for Quantifiers

- 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.

Translation from English to Logic

Examples:

1. "Some student in this class has visited Mexico."

   Solution: Let $M(x)$ denote "x has visited Mexico" and $S(x)$ denote "x is a student in this class," and U be all people.

   $$\exists x\ (S(x) \land M(x))$$

2. "Every student in this class has visited Canada or Mexico."

   Solution: Add $C(x)$ denoting "x has visited Canada."

   $$\forall x\ (S(x) \rightarrow (M(x) \lor C(x)))$$

System Specification Example

- Predicate logic is used for specifying properties that systems must satisfy.
- For example, translate into predicate logic:
  - "Every mail message larger than one megabyte will be compressed."
  - "If a user is active, at least one network link will be available."
- Decide on predicates and domains (left implicit here) for the variables:
  - Let $L(m, y)$ be "Mail message m is larger than y megabytes."
  - Let $C(m)$ denote "Mail message m will be compressed."
  - Let $A(u)$ represent "User u is active."
  - Let $S(n, x)$ represent "Network link n is state x.
- Now we have:

$$\forall m(L(m,1) \rightarrow C(m))$$

$$\exists u\, A(u) \rightarrow \exists n\, S(n, available)$$

Lewis Carroll Example

- The first two are called *premises* and the third is called the *conclusion*.
  1. "All lions are fierce."
  2. "Some lions do not drink coffee."
  3. "Some fierce creatures do not drink coffee."

  Let P(x), Q(x), and R(x) be the propositional functions "x is a lion," "x is fierce," and "x drinks coffee," respectively. Assuming that the domain consists of all creatures, express the statements in the argument using quantifiers and P(x), Q(x), and R(x).

Solution:

- $\forall x\,(P(x) \rightarrow Q(x))$
- $\exists x\,(P(x) \land \neg R(x))$
- $\exists x\,(Q(x) \land \neg R(x))$

WHY?

$\cancel{\exists x(P(x) \rightarrow \neg R(x))}$

$\cancel{\exists x(Q(x) \rightarrow \neg R(x)).}$

Logic Programming (optional)

- Prolog (from *Pro*gramming in *Log*ic) is a programming language developed in the 1970s by researchers in artificial intelligence (AI).
- Prolog programs include *Prolog facts* and *Prolog rules*.
- As an example of a set of Prolog facts consider the following:

```
instructor(chan, math273).
instructor(patel, ee222).
instructor(grossman, cs301).
enrolled(kevin, math273).
enrolled(juana, ee222).
enrolled(juana, cs301).
enrolled(kiko, math273).
enrolled(kiko, cs301).
```

- Here the predicates
  - *instructor(p,c)* represents "professor *p* is the instructor of course *c*."
  - *enrolled(s,c)* represents "student *s* is enrolled in course *c*."

Logic Programming (cont)

- In Prolog, names beginning with an uppercase letter are variables.
- If we have apredicate *teaches(p,s)* representing "professor *p* teaches student *s*," we can write the rule:

  ```
  teaches(P,S) :- instructor(P,C), enrolled(S,C).
  ```

- This Prolog rule can be viewed as equivalent to the following statement in logic (using our conventions for logical statements).

  $$\forall p\ \forall c\ \forall s(I(p,c) \wedge E(s,c)) \rightarrow T(p,s))$$

Logic Programming (cont)

- Prolog programs are loaded into a *Prolog interpreter*. The interpreter receives *queries* and returns answers using the Prolog program.
- For example, using our program, the following query may be given:

```
?enrolled(kevin,math273).
```

- Prolog produces the response:

```
yes
```

- Note that the `?` is the prompt given by the Prolog interpreter indicating that it is ready to receive a query.

Logic Programming (cont)

- The query:

```
?enrolled(X,math273).
```

produces the response:

```
X = kevin;
X = kiko;
no
```

- The query:

```
?teaches(X,juana).
```

produces the response:

```
X = patel;
X = grossman;
no
```

•The Prolog interpreter tries to find an instantiation for X. It does so and returns X = kevin.

•Then the user types the ; indicating a request for another answer.

• When Prolog is unable to find another answer it returns no.

Logic Programming (cont)

- The query:

```
?teaches(chan,X).
```

produces the response:

```
X = kevin;
X = kiko;
no
```

- A number of very good Prolog texts are available. *Learn Prolog Now!* is one such text with a free online version at http://www.learnprolognow.org/
- There is much more to Prolog and to the entire field of logic programming.

Query???

$$\sqrt{1+\sqrt{2+\sqrt{3+\sqrt{4....}}}}$$

$$\exists_{x\in\Re}\exists_{y\in\Re}(\,x = y\,) = ?$$

$$\sum_{x=1}^{\infty} x = ?$$

$$\sum_{x=1}^{\infty}\frac{1}{x} = ?$$

$$\forall_{x}(\,\Re \mid x\,) = ?$$

$$\exists_{x\in\Re}\exists_{y\in\Re}(\,x = y\,) = ?$$

$$\sqrt{1+\sqrt{2+\sqrt{3+\sqrt{4....}}}} = ?$$

$$1-1+1-1+1......... = ?$$

$$\sum_{x=1}^{\infty}\frac{1}{x} = ?$$