

**CSE
31F**

**Foundations of
Computing I**

Pre-Lecture Problem

Translate the following into predicate logic:

“There is a student who is friends with every other student except her enemies.”

H!

Negations of Quantifiers

Domain: Fruit

Predicate Definitions

$PF(x) ::= \text{"x is a purple fruit"}$

$$\forall x PF(x)$$


Imagine our domain is {plum, banana, apple}.

Can you write the statement without any quantifiers?

$$\neg (\neg PF(\text{plum}) \wedge PF(\text{banana}) \wedge PF(\text{apple}))$$

What is the negation of that statement?

$$\neg \neg PF(\text{plum}) \vee \neg PF(\text{banana}) \vee \neg PF(\text{apple})$$


$$\exists x \neg PF(x)$$

Negations of Quantifiers

Predicate Definitions

$PF(x) ::= \text{“}x \text{ is a purple fruit”}$

$$\forall x PF(x)$$

Imagine our domain is {plum, banana, apple}.

Can you write the statement without any quantifiers?

$$PF(\text{plum}) \wedge PF(\text{banana}) \wedge PF(\text{apple})$$

What is the negation of that statement?

$$\begin{aligned} & \neg(PF(\text{plum}) \wedge PF(\text{banana}) \wedge PF(\text{apple})) \\ \equiv & \neg PF(\text{plum}) \vee \neg PF(\text{banana}) \vee \neg PF(\text{apple}) \end{aligned}$$

“One of the fruits is not purple”

$$\exists x \neg P(x)$$

De Morgan's Laws for Quantifiers

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

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

De Morgan's Laws for Quantifiers

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

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

“There is no largest integer”

$$\begin{aligned} \forall x \left(\neg (\forall y (x \geq y)) \right) &\equiv \forall x \left(\exists y (\neg (x \geq y)) \right) \\ &\equiv \forall x \left(\exists y (x < y) \right) \end{aligned}$$

“For every integer there is a larger integer”

Negations of Quantifiers

- **not every positive integer is prime**
- **some positive integer is not prime**
- **prime numbers do not exist**
- **every positive integer is not prime**

Negations of Quantifiers

- **not every positive integer is prime**

$$\neg \forall x \text{Prime}(x) \equiv \exists x \neg \text{Prime}(x)$$

- **some positive integer is not prime**

$$\exists x \neg \text{Prime}(x)$$

- **prime numbers do not exist**

$$\neg \exists x \text{Prime}(x)$$

- **every positive integer is not prime**

$$\forall x \neg \text{Prime}(x)$$

Bound and Free Variables

Consider the following program:

```
hello(x) {  
    return x + y;  
}
```

In this program, we say “x” is **bound** and “y” is **free**.

Bound and Free Variables

Consider the following program:

```
hello(x) {  
    return x + y;  
}
```

x is defined here

x is used here

y is never defined ☹️

In this program, we say “x” is **bound** and “y” is **free**.

Scope of Quantifiers

It's the same idea with quantifiers.

$$\text{Has_Bigger}(x) \equiv \exists y \text{ Greater}(y, x)$$

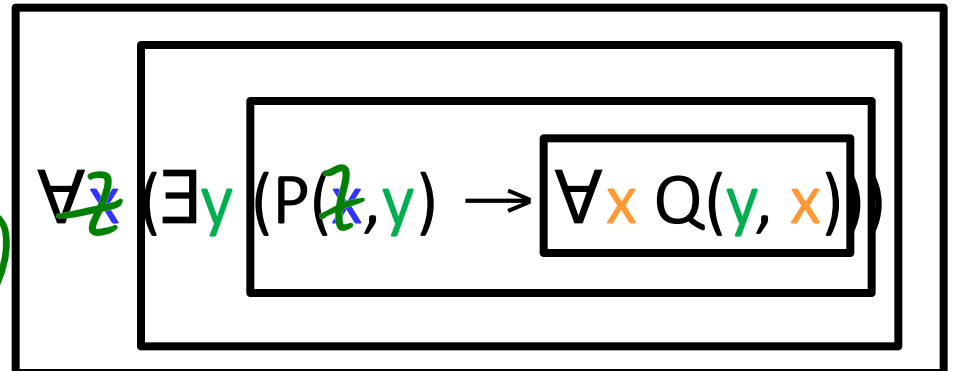
$$\forall x \exists y \text{ Greater}(y, x)$$

We figure out what a formula means “inside-out”.

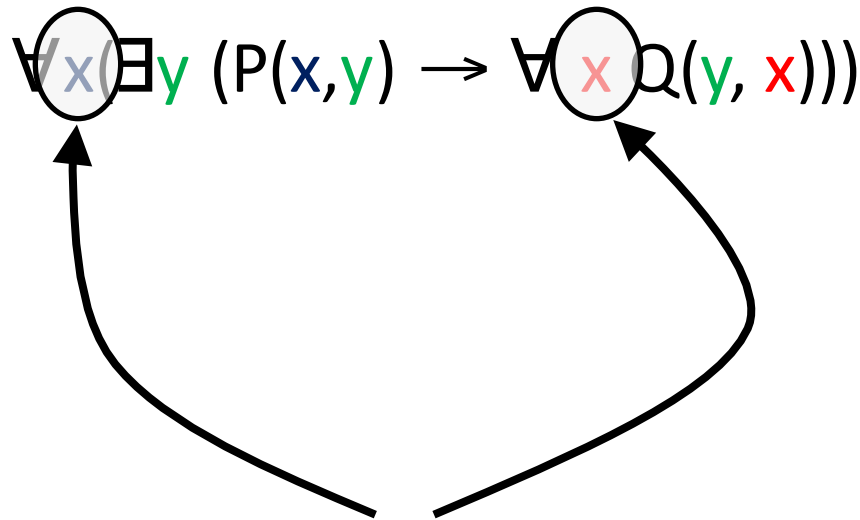
So, variables bind to the inner-most quantifier that tries to “capture” them.

$\exists y \forall x$ —————
 $\forall \exists y \text{ Greater}(y, x)$
 ↑
 This quantifier does nothing!

$(\exists y \text{ Greater}(y, x))$



BTW: Quantifier “Style”



This isn't “wrong”, it's just horrible style.

Don't confuse your reader by using the same variable multiple times...there are a lot of letters...

Scope of Quantifiers

$$\exists x (P(x) \wedge Q(x))$$

$x = y$
vs.

$$\exists x P(x) \wedge \exists y Q(y)$$

This one asserts P and Q of the *same* x.

This one asserts P and Q of potentially different x's.

Variable Renaming

NotLargest1(x) $\equiv \exists y$ Greater(y, x) vs. NotLargest2(x) $\equiv \exists z$ Greater(z, x)

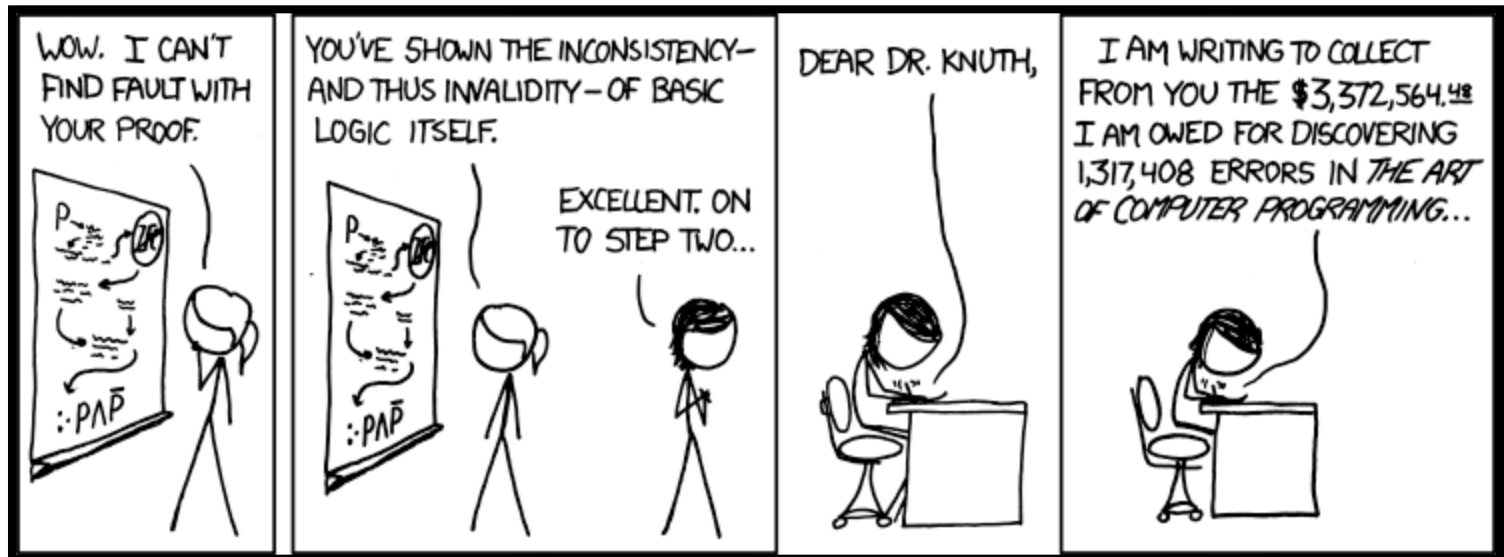
```
notLargest1(x) {  
    boolean result = false;  
    for (y : DOMAIN) {  
        result = result || y > x  
    }  
    return result;  
}
```

```
notLargest2(x) {  
    boolean result = false;  
    for (z : DOMAIN) {  
        result = result || z > x  
    }  
    return result;  
}
```

These are the same program!
Variable names are irrelevant!

CSE 311: Foundations of Computing

Lecture 6: Predicate Logic, Logical Inference



MMM Candy!

Let B be the “starred” bag of M&Ms.

Translate “There is a green M&M in B.” into predicate logic.

MMM Candy!

Let B be the "starred" bag of M&Ms.

Translate "There is a green M&M in B." into predicate logic.

Domain of Discourse
Colors & Bags

$\{red, B\}$

Predicate Definitions

Bag(x) ::= "x is a bag of M&Ms"

Color(x) ::= "x is a color"

Has(b, c) ::= "b has a c M&M."

Has(B, Green)

Green(x) ::= "x is green."

$\exists c (Green(c) \wedge Color(c) \wedge Has(B, c))$

$\rightarrow \forall c (Green(c) \wedge Color(c) \rightarrow Has(B, c))$ no please!!

MMM Candy!

Let **B** be the “starred” bag of M&Ms.

Translate “There is a green M&M in **B**.” into predicate logic.

Domain of Discourse

Colors & Bags

Predicate Definitions

Bag(x) ::= “ x is a bag of M&Ms”
--

Color(x) ::= “ x is a color”

Has(b, c) ::= “ b has a c M&M.”

Has(**Green**, **B**)

Notice that both “**Green**” and “**B**” are constants here! You could, instead, define a predicate **Green**(x):

$$\exists c(\text{Color}(c) \wedge \text{Green}(c) \wedge \text{Has}(c, \mathbf{B}))$$

MMM Candy!

Domain of Discourse

Colors & Bags

Predicate Definitions

Bag(x) ::= "x is a bag of M&Ms"

Color(x) ::= "x is a color"

Has(b, c) ::= "b has a c M&M."

Translate "Every bag of M&Ms has an M&M of some color."

$$\forall x (\text{Bag}(x) \rightarrow \exists c (\text{Color}(c) \wedge \text{Has}(x, c)))$$

Translate "There is a color that all bags of M&Ms have."

$$\exists c (\text{Color}(c) \wedge \forall b (\text{Bag}(b) \rightarrow \text{Has}(b, c)))$$

MMM Candy!

Domain of Discourse

Colors & Bags

Predicate Definitions

Bag(x) ::= "x is a bag of M&Ms"

Color(x) ::= "x is a color"

Has(b, c) ::= "b has a c M&M."

Translate "Every bag of M&Ms has an M&M of some color."

$$\forall b \left(\text{Bag}(b) \rightarrow \exists c (\text{Color}(c) \wedge \text{Has}(b, c)) \right)$$

Translate "There is a color that all bags of M&Ms have."

$$\exists c \left(\text{Color}(c) \wedge \forall b (\text{Bag}(b) \rightarrow \text{Has}(b, c)) \right)$$

We're not done!

Domain of Discourse

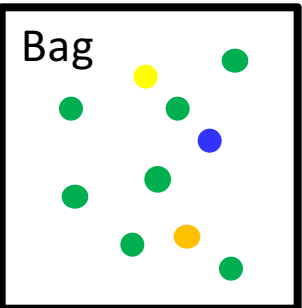
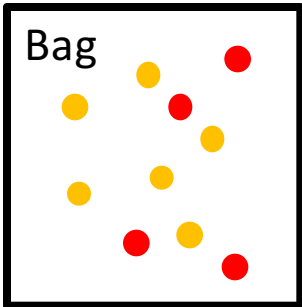
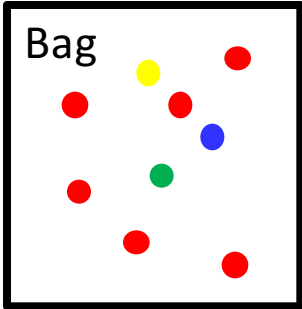
Colors & Bags

Predicate Definitions

Bag(x) ::= "x is a bag of M&Ms"

Color(x) ::= "x is a color"

Has(b, c) ::= "b has a c M&M."



"There is a color that all bags of M&Ms have."

$$\exists c \left(\text{Color}(c) \wedge \forall b (\text{Bag}(b) \rightarrow \text{Has}(b, c)) \right)$$

VS.

"Every bag of M&Ms has an M&M of some color."

$$\forall b \left(\text{Bag}(b) \rightarrow \exists c (\text{Color}(c) \wedge \text{Has}(b, c)) \right)$$

We're not done!

Domain of Discourse

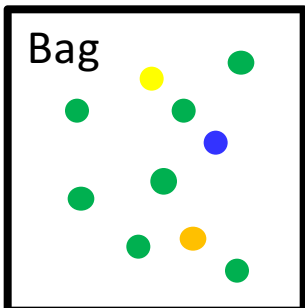
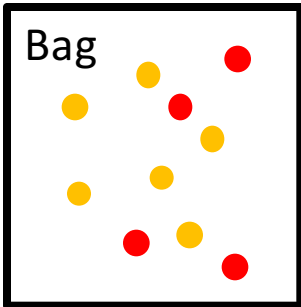
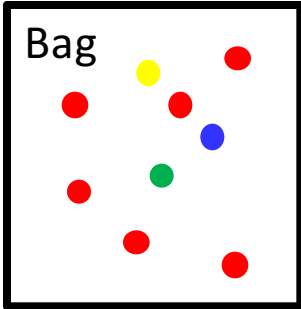
Colors & Bags

Predicate Definitions

$\text{Bag}(x) ::= \text{"x is a bag of M\&Ms"}$

$\text{Color}(x) ::= \text{"x is a color"}$

$\text{Has}(b, c) ::= \text{"b has a c M\&M."}$



“There is a color that all bags of M&Ms have.”

$$\exists c \left(\text{Color}(c) \wedge \forall b (\text{Bag}(b) \rightarrow \text{Has}(b, c)) \right)$$

For the bags on the left, this is not true. We need a **single color** that all the bags share.

VS.

“Every bag of M&Ms has an M&M of some color.”

$$\forall b \left(\text{Bag}(b) \rightarrow \exists c (\text{Color}(c) \wedge \text{Has}(b, c)) \right)$$

For the bags on the left, this is **true**. The first bag has red; the second has orange, and the third has orange.

Still not done!

Domain of Discourse

{1, 2, 3, 4}

Predicate Definitions

GreaterEq(x, y) ::= "x ≥ y"

"There is a number greater than or equal to all numbers."

$$\exists x \left(\forall y (\text{GreaterEq}(x, y)) \right)$$

VS.

"Every number has a number greater than or equal to it."

$$\forall y \left(\exists x (\text{GreaterEq}(x, y)) \right)$$

Still not done!

Domain of Discourse

Integers
OR
{1, 2, 3, 4}

Predicate Definitions

GreaterEq(x, y) ::= "x ≥ y"

“There is a number greater than or equal to all numbers.”

$$\exists x \left(\forall y (\text{GreaterEq}(x, y)) \right)$$

“Every number has a number greater than or equal to it.”

$$\forall y \left(\exists x (\text{GreaterEq}(x, y)) \right)$$

	y			
	1	2	3	4
1	T	F	F	F
2	T	T	F	F
3	T	T	T	F
4	T	T	T	T

The purple statement requires an entire row to be true.

The red statement requires one entry in each column to be true.

Nested Quantifiers

- **Bound variable names don't matter**

$$\forall x \exists y P(x, y) \equiv \forall a \exists b P(a, b)$$

- **Positions of quantifiers can sometimes change**

$$\forall x (Q(x) \wedge \exists y P(x, y)) \equiv \forall x \exists y (Q(x) \wedge P(x, y))$$

- **But: order is important...**

Quantification with Two Variables

expression	when true	when false
$\forall x \forall y P(x, y)$	Every pair is true.	At least one pair is false.
$\exists x \exists y P(x, y)$	At least one pair is true.	All pairs are false.
$\forall x \exists y P(x, y)$	We can find a specific y for each x . $(x_1, y_1), (x_2, y_2), (x_3, y_3)$	Some x doesn't have a corresponding y .
$\exists y \forall x P(x, y)$	We can find ONE y that works no matter what x is. $(x_1, y), (x_2, y), (x_3, y)$	For any candidate y , there is an x that it doesn't work for.

Logical Inference

- **So far we've considered:**
 - How to understand and *express* things using propositional and predicate logic
 - How to *compute* using Boolean (propositional) logic
 - How to show that different ways of expressing or computing them are *equivalent* to each other
- **Logic also has methods that let us *infer* implied properties from ones that we know**
 - Equivalence is a small part of this

Why Proofs?

Consider $f(n) = 991n^2 + 1$

Is $f(n)$ a perfect square for any $n > 0$?

Applications of Logical Inference

- **Software Engineering**
 - Express desired properties of program as set of logical constraints
 - Use inference rules to show that program implies that those constraints are satisfied
- **Artificial Intelligence**
 - Automated reasoning
- **Algorithm design and analysis**
 - e.g., Correctness, Loop invariants.
- **Logic Programming, e.g. Prolog**
 - Express desired outcome as set of constraints
 - Automatically apply logic inference to derive solution

Proofs

- Start with hypotheses and facts (**Axioms**)
- Use “rules” to generate more facts from existing facts (**Inference Rules**)
- Result is proved when it is included in the set of “proven facts”

Inference Rules

If I have a proof of A and
I have a proof of B....

Requirements: A B

Conclusions: \therefore C D

Then, I have
also proven C

Then, I have
also proven D

Example (Modus Ponens):

A A \rightarrow B
 \therefore B

If I have a proof of A and a proof of A \rightarrow B, then I have a proof of B.

An inference rule: *Modus Ponens*

- If p and $p \rightarrow q$ are both true then q must be true

- Write this rule as

Modus Ponens	
A	$A \rightarrow B$
<hr/>	
\therefore	B

- Given:
 - If it's Saturday, then you have a 311 lecture today.
 - It's Saturday.
- Therefore, by modus ponens:
 - You have a 311 lecture today.

My First Proof!

Show that r follows from p , $p \rightarrow q$, and $q \rightarrow r$

1. p Given
2. $p \rightarrow q$ Given
3. $q \rightarrow r$ Given
- 4.
- 5.

My First Proof!

Show that r follows from p , $p \rightarrow q$, and $q \rightarrow r$

- | | | |
|----|-------------------|----------|
| 1. | p | Given |
| 2. | $p \rightarrow q$ | Given |
| 3. | $q \rightarrow r$ | Given |
| 4. | q | MP: 1, 2 |
| 5. | r | MP: 3, 4 |

Proofs can use equivalences too

Show that $\neg p$ follows from $p \rightarrow q$ and $\neg q$

- | | | |
|----|-----------------------------|-------------------|
| 1. | $p \rightarrow q$ | Given |
| 2. | $\neg q$ | Given |
| 3. | $\neg q \rightarrow \neg p$ | Contrapositive: 1 |
| 4. | $\neg p$ | MP: 2, 3 |

Important: Applications of Inference Rules

- You can use equivalences to make substitutions of any sub-formula.
- Inference rules only can be applied to whole formulas (not correct otherwise).

e.g. 1. $p \rightarrow q$ given
~~2. $(p \vee r) \rightarrow q$ intro \vee from 1.~~

Does not follow! e.g. $p=F, q=F, r=T$