Predicate Calculus (con’t)

**Negations of quantified statements:**

The negation of \( \forall \) involves \( \exists \).  The negation of \( \exists \) involves \( \forall \).

<table>
<thead>
<tr>
<th>Informal:</th>
<th>Formal quantified:</th>
<th>Formal negation:</th>
</tr>
</thead>
<tbody>
<tr>
<td>All CS majors are nerds.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Some beautiful people are currently in KH007.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>No fleas have personality.</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**True or false?**

1) All elephants in the Fitness Center are purple.
2) \( \forall \) dogs \( d \), if \( d \) is in KH007 now, \( d \) is a pointer.

**Multiple quantifiers**

A statement can contain more than 1 quantifier.

ex: All classes have someone who’s a curve-breaker.

*How to formally state this?*

ex: Some dorms contain all partiers.

*How to formally state this?*

How would you negate these statements?

\( \forall \) Java program \( p \), \( \exists \) a line in \( p \) that contains the word “main”.

\( \exists \) computer science course \( x \) such that \( \forall \) lecture of \( x \), the lecture is totally incomprehensible.
To negate $\forall x, \exists y$ such that $P(x,y)$

use $\exists x$ such that $\forall y, \sim P(x,y)$

ex: All classes have someone who’s a curve-breaker.

Negation:

How about negating:
“Some classes contain some toadies.”

“All students earn all A’s.”

**Universal conditional statement:**

$\forall x \in D$, if $P(x)$ then $Q(x)$.

ex: For all students s, if s studies, s will succeed.

| Contrapositive |  
| Converse |  
| Inverse |  

Lecture 5p. 2
Necessary and Sufficient Conditions, Only if:
For all students s, studying is a necessary condition for success.

For all students s, studying is a sufficient condition for success.

For all students s, s will succeed only if s studies.

Arguments Containing Quantified Statements

All CS majors are intelligent.  
Mary is a CS major.  
∴ Mary is intelligent.

Demonstration with diagrams:

\[
\forall x, \text{if } P(x) \text{ is true, then } Q(x) \text{ is true.}  
\text{a makes } P(x) \text{ true.}  
∴ \text{a makes } Q(x) \text{ true.}  
\]

\text{Universal Modus Ponens}

Errors (in quantified form)

All males are genetically challenged.  
Pat is genetically challenged.  
∴ Pat is a male.

All flies are friendly.  
Greg is not a fly.  
∴ Greg is not friendly.
More examples:
All yaks are hairy.  
Cleo is not a yak.  
∴ Cleo is not hairy.

All yaks are hairy.  
Cleo is hairy.  
∴ Cleo is a yak.

**Question:** Can computer programs be written to “reason” using predicate calculus?

**Prolog**

**Prolog program** = **database** consisting of **facts** and/or **rules**

From the database the user can extract information by means of **queries**.

**ex:** program, **prey.pl**:

```prolog
eat(bear, fish).
et(bear, fox).
et(deer, grass).
animal(bear).
animal(fish).
animal(fox).
animal(deer).
plant(grass).

prey(X) :-
et(_,X),
animal(X).
```

**queries:**

```prolog
animal(bear).
et(bear, rabbit).
eat(bear, A).
et(A, B), plant(B).
```

Entered and compiled on Linux with
```
> prolog
?- [prey].
```

In the interactive environment, the user enters these at the prompt `| ?-`.

System will answer, “yes”, “no” or supply values for variables.