Logic and Electricity

Every logical statement corresponds to an electrical circuit. Input lines to the circuit correspond to the simple statements involved.

Electrical power on = 1 = TRUE
Electrical power off = 0 = FALSE

Basic gates:

- **NOT**

- **AND**

- **OR**

A more complicated circuit. What is its corresponding logical statement?

Another:
Number Systems

**Base 10:** digits \{0, 1, 2, \ldots, 9\}

3789.13

**Base 2:** digits \{0, 1\}

1001101.1011

Converting from base 2 to base 10

Converting from base 10 to base 2 (recursive)

**Algorithm:**

*Step 1:* If the number is \( \geq 2 \), divide the number by 2; otherwise just write the number and stop.

*Step 2:* Perform the algorithm again using the quotient from step 1 as the number.

*Step 3:* Write the remainder from step 1.
Predicate Calculus

How do we represent the following with our logic symbols?

- All dogs have fleas.
- Some computer scientists are wacky.
- $x$ is a prime number.

A **predicate** is a statement containing a finite number of **variables**.

Symbolized $P(x, y, ...)$

The possible values for a variable is the **domain** of the variable.

Symbolized $D$.

The subset of $D$ in which are found values of $x$ for which predicate $P(x)$ is **true** is called the **truth set** of $P(x)$.

ex. $P(x)$: $x$ is a natural number and $x$ has no divisors except itself and one.

$$D = \{ x \in \mathbb{Z}^+ \mid x \text{ is a prime number} \}$$

(Recall $\mathbb{Z}$, $\mathbb{R}$, and $\mathbb{Q}$.)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>$P(x) \Rightarrow Q(x)$</td>
<td>means that the truth set of $P(x)$ is a subset of the truth set of $Q(x)$</td>
</tr>
<tr>
<td>$P(x) \iff Q(x)$</td>
<td>means that the truth set of $P(x)$ and the truth set of $Q(x)$ are the same</td>
</tr>
</tbody>
</table>

*What is the relationship between $P(x)$ and $Q(x)$?*

$P(x)$: $x$ is an integer ending in 0.

$Q(x)$: $x$ is an integer divisible by 5.

**How can we change predicates into statements?**

1. Replace the variable with a value.
   
   “$x$ is prime number” becomes “7 is a prime number”.

2. Explicitly state the domain.
   
   “$y$ is an even number” becomes “$y$ is an even number and $y \in \{2, 4, 6, \ldots\}$”
3. Use a quantifier:

<table>
<thead>
<tr>
<th>Quantifier</th>
<th>Symbol</th>
<th>Produces</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>for all</td>
<td>∀</td>
<td>universal statement</td>
<td>∀ x ∈ D, x is even</td>
</tr>
<tr>
<td>there exists</td>
<td>∃</td>
<td>existential statement</td>
<td>∃ x ∈ D s.t. x is prime</td>
</tr>
</tbody>
</table>

Informal English: 
Type of statement: Formal Quantified:

All dogs have fleas.

Some computer scientists are wacky.

Some programs produce incorrect output.

Truth and Falsity of Statements

<table>
<thead>
<tr>
<th>Quantifier</th>
<th>Prove it’s true</th>
<th>Prove it’s false</th>
</tr>
</thead>
<tbody>
<tr>
<td>∀</td>
<td></td>
<td></td>
</tr>
<tr>
<td>∃</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Universal conditional statements

∀ x ∈ D, if p(x), then q(x).

ex: ∀ s ∈ { students }, if s attends Loyola, s will receive a quality education.

If the domain is restricted, the conditional can be removed.

∀ s ∈ {

Formalize in both ways: (universal conditional & universal with restricted domain)

If a person has his/her tongue pierced, that person will have trouble eating spaghetti.
**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 computer scientists are nerds.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Some beautiful people are currently in 270.</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 270 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 companies employ all type-A personalities.

*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 \( \forall x, \exists y \text{ such that } P(x, y) \)
use \( \exists x \text{ such that } \forall y, \sim P(x, y) \)

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

To negate \( \exists x \text{ such that } \forall y P(x, y) \)
use \( \forall x, \exists y \text{ such that } \sim P(x, y) \)

ex: Some companies employ all type-A personalities.
Negation: 

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

“All students earn all A’s.”

**Universal conditional statement:**
\( \forall x \in D, \text{ if } P(x) \text{ then } Q(x). \)
ex: For all students s, if s studies, s will succeed.

<table>
<thead>
<tr>
<th>Contrapositive</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Converse</td>
<td></td>
</tr>
<tr>
<td>Inverse</td>
<td></td>
</tr>
</tbody>
</table>
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 computer scientists are intelligent. Mary is a computer scientist. ∴ Mary is intelligent.

Demonstration with diagrams:

Universal Modus Ponens

All CS professors are bizarre. Margaret is not bizarre. ∴

Universal Modus Tollens

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:

\[
\begin{align*}
text{eat(bear, fish).} \\
text{eat(bear, fox).} \\
text{eat(deer, grass).} \\
text{animal(bear).} \\
text{animal(fish).} \\
text{animal(fox).} \\
text{animal(deer).} \\
text{plant(grass).} \\
\text{prey(X) :-} \\
\quad \text{eat(\_\_X),} \\
\quad \text{animal(X).}
\end{align*}
\]

queries:

\[
\begin{align*}
text{animal(bear).} \\
text{eat(bear, rabbit).} \\
text{eat(bear,A).} \\
text{eat(A, B), plant (B).} \\
\text{prey(A).}
\end{align*}
\]

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