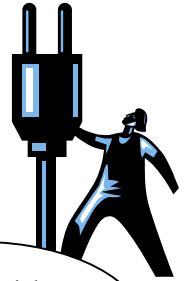


Logic and Electricity

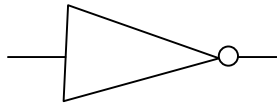


Every logical statement corresponds to an electrical circuit. Input lines to the circuit correspond to the simple statements involved.

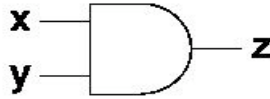
So every circuit has a corresponding

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

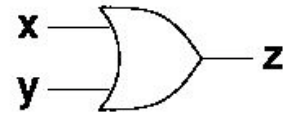
Basic gates:



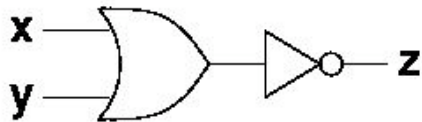
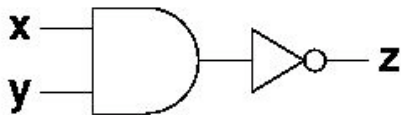
NOT



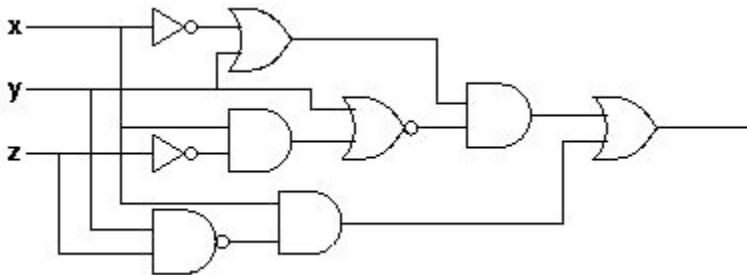
AND



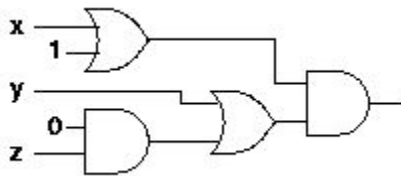
OR



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



Another:



Number Systems

Base 10: digits $\{0, 1, 2, \dots, 9\}$

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 ≥ 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 $P(x, y, \dots)$

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.

$$D = \{ x \in \mathbb{Z}^+ \mid x \text{ is a prime number} \} \quad (\text{Recall } \mathbf{Z}, \mathbf{R}, \text{ and } \mathbf{Q}.)$$

$P(x) \Rightarrow Q(x)$	means that the truth set of $P(x)$ is a subset of the truth set of $Q(x)$
$P(x) \Leftrightarrow Q(x)$	means that the truth set of $P(x)$ and the truth set of $Q(x)$ are the same

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 $y \in \{2, 4, 6, \dots\}$ ”

3. Use a quantifier:

Quantifier	Symbol	Produces	Example
for all	\forall	universal statement	$\forall x \in D, x$ is even
there exists	\exists	existential statement	$\exists x \in D$ s.t. x is prime

Informal English:

All dogs have fleas.

Type of statement:

Some computer scientists are wacky.

Formal Quantified:

Some programs produce incorrect output.

Truth and Falsity of Statements

Quantifier	Prove it's true	Prove it's false
\forall		
\exists		

Universal conditional statements

$\forall x \in D$, if $p(x)$, then $q(x)$.

ex: $\forall s \in \{ \text{students} \}$, if s attends Loyola, s will receive a quality education.

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

$\forall s \in \{$

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 .

<i>Informal:</i>	<i>Formal quantified:</i>	<i>Formal negation:</i>
All computer scientists are nerds.		
Some beautiful people are currently in 270.		
No fleas have personality.		

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

$\forall x, \exists y$ 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:

To negate

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

use

$\forall x, \exists y$ 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:

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

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

Contrapositive		
Converse		
Inverse		

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.

\therefore Mary is intelligent.

Demonstration with diagrams:

$\forall x$, if $P(x)$ is true, then $Q(x)$ is true.

a makes $P(x)$ true.

\therefore a makes $Q(x)$ true.

Universal Modus Ponens

All CS professors are bizarre.

Margaret is not bizarre.

\therefore

$\forall x$, if $P(x)$ is true, then $Q(x)$ is true.

a does not make $Q(x)$ true.

\therefore

Universal Modus Tollens

Errors (in quantified form)

All males are genetically challenged.

Pat is genetically challenged.

\therefore Pat is a male.

All flies are friendly.

Greg is not a fly.

\therefore 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**:

eat(bear, fish).
eat(bear, fox).
eat(deer, grass).
animal(bear).
animal(fish).
animal(fox).
animal(deer).
plant(grass).
prey(X) :-
 eat(_,X),
 animal(X).

queries:
animal(bear).

eat(bear, rabbit).

eat(bear,A).

eat(A, B), plant (B).

prey(A).

In the interactive environment, the user enters these at the prompt
| ?-
System will answer, “yes”, “no” or supply values for variables

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