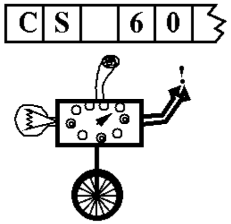


# Predicate Logic

September 7, 2001

## Predicate Logic

- Proposition logic deals with propositions (self-contained things that can be true or false).
- Predicate logic constructs propositions out of predicates that apply to **individuals.**
- A predicate with all argument individuals specified is like a proposition, in that it has a true or false value.

## Quantifiers

- In addition to truth function operators of proposition logic, predicate logic introduces **quantifiers** for expressing variation over individuals:

$(\forall x)\ p(x)$ — for *all* x, p(x) is true

universal quantifier

$(\exists x)\ p(x)$ — there *exists* an x where p(x) is true

existential quantifier

## Interpretation of Formulas with Quantifiers

- To determine the truth of a particular statement, e.g.,

  $(\forall x)\ p(x)$

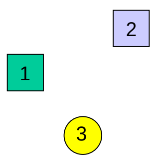
  we need to know:
  - What does the predicate p actually checking for?
  - What are the individuals we care about?
- The answer to these questions is called an *interpretation*.
  - The set of individuals over which we are quantifying, called the **domain** (finite or infinite).
  - An interpretation also gives a specific predicate for each predicate symbol.

# A Specific Interpretation

`triangle(x)` means x is a triangle
`square(x)` means x is a square
`circle(x)` means x is a circle
`blue(x)` means x is blue, etc.

(Domain)

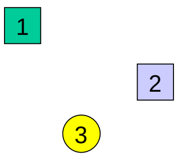
`above(x,y)` means x is above y
`left(x,y)` means x is to the left of y




Which are true in the interpretation shown?

```
blue(3)
square(2)
above(2,1)
left(3,1)
left(1,3) ∧ (red(2) ∨ yellow(3))
```

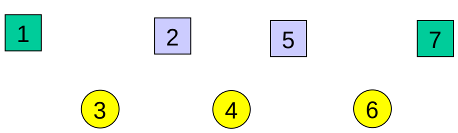
# Using Quantifiers




Which are true in the interpretation shown?

```
(∀x) (square(x) ∨ circle(x))
(∀x) square(x)
(∀x, y) ((square(x) ∧ circle(y)) → above(x,y))
(∃x, y) ((square(y) ∧ circle(x) ∧ left(x,y))
(∀x) (left(x,x))
```

# Exercise: True (General) Propositions?




# Formulas for Natural Numbers

```
(∀x) ((x = 0) v (∃y) (x = S(y))
(∀x) ¬(S(x) = 0)
(∀x, y) S(x) = S(y) → x = y
(p(0) ∧ (∀x)(p(x) → p(S(x)))) => (∀x) p(x)
```

- Here the interpretation is:
  - S represents the *successor* function (`S(x) = x+1`)
  - `0` is an "individual"
  - p is any predicate
- These form a variant on the Peano axioms (Guiseppe Peano, 1889).

## Universally Valid Formulas are true regardless of interpretation

- (Assume that domain is non-empty)

$(\forall x)\ p(x) \rightarrow (\exists x)\ p(x)$

$((\forall x)p(x) \vee (\forall x)q(x)) \rightarrow (\forall x)\ (q(x) \vee p(x))$

$(\exists x)\ (p(x) \wedge q(x)) \rightarrow ((\exists x)\ p(x)) \wedge ((\exists x)\ q(x))$

$(\forall x)\ (p(x) \wedge q(x)) \rightarrow ((\forall x)\ p(x)) \wedge ((\forall x)\ q(x))$

$(\exists x)p(x) \leftrightarrow \neg(\forall x)\neg p(x)$

$(\exists x)\ (\forall y)p(x,y) \rightarrow (\forall y)\ (\exists x)p(x,y)$

- Not valid:

$(\forall x)\ (\exists y)p(x,y) \rightarrow (\exists y)\ (\forall x)p(x,y)$

$(\forall x)\ (p(x) \vee q(x)) \rightarrow ((\forall x)\ p(x)) \vee ((\forall x)\ q(x))$

## Uses of Predicate Logic

- Querying databases
- Program specification
- Program verification
- Advanced properties of logic circuits (time dependence, etc.)

## Querying Databases

- For now, restrict to
  **relational databases**
- View each table in database as a predicate
- Predicate logic expression selects those combinations for which the expression evaluates to true

## Relational Database Example

**lives**

| name | dorm |
|---|---|
| John | East |
| Naima | South |
| Alice | West |
| Toshiko | East |
| Roy | North |
| Albert | South |

**takes**

| name | dept | number |
|---|---|---|
| John | CS | 60 |
| Naima | CS | 60 |
| Alice | CS | 5 |
| Toshiko | CS | 5 |
| Albert | CS | 60 |
| Roy | Math | 55 |
| Naima | Math | 55 |
| Alice | Math | 70 |
| Toshiko | Math | 80 |
| Albert | Math | 55 |

**tutors**

| name | dept | number |
|---|---|---|
| John | CS | 5 |
| Naima | CS | 5 |
| Roy | Math | 3 |
| Alice | Math | 55 |
| Albert | Math | 4 |

Three relations:

***lives***: name x dorm

***takes***: name x dept x number

***tutors***: name x dept x number

## Relational Database Example

Sample Queries:

Who lives in South dorm?

```
x: lives(x, South)
```

Who lives in East dorm and takes CS 5?

```
x: lives(x, East) ^ takes(x, CS, 5)
```

Who takes a CS course?

```
x: (∃y) takes(x, CS, y)
```

## Relational Database Example

Queries:

Who takes a CS course and tutors a Math course?

What tutors live in West dorm?

Who lives in East dorm that is not a tutor?

## Dominant Database Languages

- SQL
  - Structured Query Language
  - The standard query language used in most commercial relational database systems (Oracle, Informix, Sybase, etc.)
  - Invented by Don Chamberlin, HMC ‘66
- Prolog
  - Programming in Logic
  - A complete programming language
  - Used in AI and rapid prototyping