CSE 321 Discrete Structures

Winter 2008

Lecture 4

Predicate Calculus

Announcements

- Reading for this week
  - Today: 1.3, 1.4
  - Wednesday/Friday: 1.5, 1.6

Highlights from Lecture 3

- Introduction of predicates
  - Functions with range {T, F}
- Quantifiers
  - $\forall x\, P(x)$: $P(x)$ is true for every x in the domain
  - $\exists x\, P(x)$: There is an x in the domain for which $P(x)$ is true

Statements with quantifiers

- $\forall x \exists y$ Greater (y, x)

  For every number there is some number that is greater than it
- $\exists$ y $\forall$ x Greater (y, x)
- $\forall x \exists y$ (Greater(y, x) $\wedge$ Prime(y))
- $\forall x$ (Prime(x) $\rightarrow$ (Equal(x, 2) $\vee$ Odd(x))
- $\exists x \exists y$(Equal(x, y + 2) $\wedge$ Prime(x) $\wedge$ Prime(y))

Domain:
Positive Integers

Greater(a, b) ≡ "a > b"

Statements with quantifiers

Domain:
Positive Integers

Even(x)
Odd(x)
Prime(x)
Greater(x,y)
Equal(x,y)

- "There is an odd prime"
- "If x is greater than two, x is not an even prime"
- $\forall x \forall y \forall z$ ((Equal(z, x+y) $\wedge$ Odd(x) $\wedge$ Odd(y))$\rightarrow$ Even(z))
- "There exists an odd integer that is the sum of two primes"

Goldbach's Conjecture

- Every even integer greater than two can be expressed as the sum of two primes

Even(x)
Odd(x)
Prime(x)
Greater(x,y)
Equal(x,y)

Domain:
Positive Integers

Systems vulnerability
Reasoning about machine status

- Specify systems state and policy with logic
  - Formal domain
    - reasoning about security
    - automatic implementation of policies
- Domains
  - Machines in the organization
  - Operating Systems
  - Versions
  - Vulnerabilities
  - Security warnings
- Predicates
  - RunsOS(M, O)
  - Vulnerable(M)
  - OSVersion(M, Ve)
  - LaterVersion(Ve, Ve)
  - Unpatched(M)

System vulnerability statements

- Unpatched machines are vulnerable
- There is an unpatched Linux machine
- All Windows machines have versions later than SP1

Prolog

- Logic programming language
- Facts and Rules

```
RunsOS(SlipperPC,  Windows)
RunsOS(SlipperTablet, Windows)
RunsOS(CarmelLaptop, Linux)

OSVersion(SlipperPC, SP2)
OSVersion(SlipperTablet, SP1)
OSVersion(CarmelLaptop, Ver3)

LaterVersion(SP2, SP1)
LaterVersion(Ver3, Ver2)
LaterVersion(Ver2, Ver1)
```

```
Later(x, y) :-
   Later(x, z), Later(z, y)

NotLater(x, y) :- Later(y, x)
NotLater(x, y) :-
   SameVersion(x, y)

MachineVulnerable(m) :-
   OSVersion(m, v),
   VersionVulnerable(v)
VersionVulnerable(v) :-
   CriticalVulnerability(x),
   Version(x, n),
   NotLater(v, n)
```

Nested Quantifiers

- Iteration over multiple variables
- Nested loops
- Details
  - Use distinct variables
    - $\forall$ x($\exists$ y(P(x,y) $\rightarrow$ $\forall$ x Q(y, x)))
  - Variable name doesn't matter
    - $\forall$ x $\exists$ y P(x, y) $\equiv$ $\forall$ a $\exists$ b P(a, b)
  - Positions of quantifiers can change (but order is important)
    - $\forall$ x (Q(x) $\wedge$ $\exists$ x P(x, y)) $\equiv$ $\forall$ x $\exists$ y (Q(x) $\wedge$ P(x, y))

Quantification with two variables

Expression	When true	When false
$\forall$ x $\forall$ y P(x,y)		
$\exists$ x $\exists$ y P(x,y)		
$\forall$ x $\exists$ y P(x, y)		
$\exists$ y $\forall$ x P(x, y)		