Discrete Mathematical Structures
CS 2233 Lecture Four

Prof. William Winsborough
January 22, 2009

Business

- **Recall: Attendance in recitation is not optional!**
- **Recall:** Homework 2 due Thursday January 29
  – Section 1.2: 2, 6, 10
  – Section 1.3: 10d, 10e, 14, 24c, 24d, 32a, 32b, 44
- Questions???

Universe of Discourse and Quantifiers

- The *universe of discourse* or *domain* is the set of all possible values for variables
- We can refer to values in the universe either by using constant symbols (like "Fred") or by using quantifiers
- There are two quantifiers in standard predicate calculus: *for all* (∀) and *there exists* (∃)
- There are called the universal quantifier and the existential quantifier, respectively

Universal Quantifiers

- The universal quantification of p(x) is the following proposition:
  – "p(x) is true for all values of x in the universe of discourse"
  – Written $\forall x\ p(x)$ or $\forall x.p(x)$
- Similarly, if $\phi(x)$ is a formula in x, $\forall x.\phi(x)$ means the formula holds for all elements of the universe
  – What does $\forall x.(r(x) \rightarrow q(x))$ mean?
  – How is it different from $(\forall x.r(x)) \rightarrow (\forall x.q(x))$?

Existential Quantifiers

- The existential quantification of p(x) is the proposition
  – "there exists an element x in the universe of discourse such that p(x) is true"
  – Written $\exists x\ p(x)$ or $\exists x.p(x)$
  – What does $\exists x.(r(x) \wedge \neg q(x))$ mean?
  – How is it different from $(\exists x.r(x)) \wedge (\exists x.\neg q(x))$
- If the universe of discourse is {0,1,2}, then $\exists x.p(x)$ has the same truth value as $p(0) \vee p(1) \vee p(2)$

Scope of Quantifiers

- $(\exists x.x>3) \wedge (\exists x.x<1)$
- The scope of a quantifier is the subformula following the dot
- Copy syntax trees drawn on board into your notes for each of the following:
  – $\forall x.(r(x) \rightarrow q(x))$
  – $(\forall x.r(x)) \rightarrow (\forall x.q(x))$
  – $\exists x.(r(x) \wedge \neg q(x))$
  – $(\exists x.r(x)) \wedge (\exists x.\neg q(x))$

Universal Quantifiers

- Note that ∀x.p(x) ≡ ∀y.p(y)
- If the universe of discourse is {0, 1, 2}, then ∀x.p(x) has the same truth value as p(0) ∧ p(1) ∧ p(2)
- Can you always rewrite ∀x.p(x) this way?
  - What if the universe of discourse is infinite?
  - Logical statements are finite objects

Negations of Quantified Formulas

- De Morgan's Laws for Quantifiers

  ¬∃x.p(x) ≡ ∀x.¬p(x)

  ¬∀x.p(x) ≡ ∃x.¬p(x)

Predicate Formulas as Specifications

- For the domain of integers, is ∀x.(x>3) true or false?
  - How about ∃x.(x>3) ?
  - Note: ">" is the predicate symbol here
  - X>3 is another way of writing >(x,3)
- What does ∀x ∀y.(x+y = y+x) mean?
  - Is it true?
  - What is the predicate here?
  - In logic, "+" is called a *function symbol* (term not introduced in the text)

Formal Notion of "Interpretation"

- An *Interpretation* of a formula consists of:
  - A domain of discourse
    - *E.g.,* bit values (Boolean), natural numbers, integers, rationals, reals, vectors of same
  - A meaning for each predicate symbol
    - *I.e.,* a propositional function
    - *E.g.,* "<" refers to "less than" over bit vectors
  - A meaning for each constant symbol
    - *E.g.,* "1" refers to the natural number 1, "**T**" denotes the Boolean value *true*, "∅" denotes the empty set, "λ" denotes the empty string
  - A meaning for each function symbol
    - *E.g.,* "+" refers to addition of natural numbers, "•" denotes dot product over vectors of real numbers

Predicate Formulas in Software Specifications

- One of the primary uses of predicate-logic formulas in computer science is specification of software
  - Software behavior can thought of in terms of an intended interpretation
    - E.g., a library for high-precision arithmetic
  - Formulas serve to specify requirements of software behavior

Exercise: In Which Numeric Domains Does each of the Following Hold?

- Domains
  - Z – The integers
  - N – The natural numbers (non-negative integers)
  - R^+ – The positive reals
  - $R^+ \cup \{0\}$ – The non-negative reals
- Formulas (Assume "<" and "≤" have usual interpretation)
  - ∀x.∃y.y<x
  - ∃x.∀y.x ≤ y
  - ∀x.∀z.(x<z →∃y.(x<y)∧(y<z))