Chap3 – Predicates and Quantifiers

3.1

* Propositions are not powerful enough for us.
* We can say “2 is an even number”
* “3 is an even number”
* “x is an even number” -> not a proposition because we don’t know what number x is.
* Make our logical world more powerful by adding predicates and variables to it.
* A predicate is a function that returns either true or false.
  + Like something you’d put inside of an if statement.
* Predicates usually take arguments.
  + P(x): x is an even number.
  + P(2) true, P(3) false.
  + GreaterThan(x, y): x > y
  + HasBrownHair(x): x has brown hair
* Domain of a predicate is the set of values possible for a variable.
  + Using the term ***set*** here in the sense of a mathematical set, which has a certain definition.
  + For the moment, think of a set as a collection of objects.
  + Domain for P: all numbers? All integers?
  + Domain for HasBrownHair: set of all people.
  + Some specially named domains:
    - Integers (Z) also pos, nonneg, neg (Zahlen = numbers)
    - Rational numbers (Q) (quotient)
    - Real numbers (R)
  + Set symbols (letters)
  + Set membership symbol
* Predicates with literals are propositions:
  + P(2) AND P(4)
  + SleepsThroughFinalExam(Phil) -> FailsClass(Phil)
* Quantifiers: more powerful ways to build propositions
  + Right now we have no way of using predicates to talk about indeterminate things (things that are not specific).
  + Universal quantifier
    - Used when we want to say something is true for an entire set.
      * Statement is true if predicate is true for all members of the set.
      * Statement is false if predicate is false for at least one member of the set.
    - For all integers x, x is even.
    - For all integers x, x is even or odd.
    - For all integers x, x < x + 1.
  + Existential quantifier
    - Used when we want to say something is true for at least one thing in a set.
      * Statement is true if predicate is true for at least one member of the set.
      * Statement is false if predicate is false for all members of the set.
    - Exists x in P such that x is even.
    - Exists x in students in class such that HasBrownHair(x)
  + Statement about nothing in the group?
    - No students in this class have purple hair.
    - Equiv to all students in this class do not have purple hair.
  + Translation practice.
    - Univ keywords = every, each, all, no, none
    - Existential keywords = at least one, some, there is, there are
    - All tricycles are red. \*\*UNIVERSAL CONDITIONAL\*\*
      * Once with domain = tricycles.
      * Once with domain = universal set.
    - For any even integer x, x + 1 is odd.
      * Once with domain = even,
      * Once with domain = integers
    - Some tricycles are red.
    - There is an even prime number.

If super-extra time, talk about barber paradox. In town, there is a barber who has a very specific job. The barber’s job is to shave the face of every man who does not shave his own face. *Does the barber shave their own face?*

Interesting way to escape the paradox – barber is a woman.

* **PRACTICE**
  + P(x) = x > 1/x true or false: P(2), P(1/2), P(-1) [**True, False, False]**
  + Forall x in Z P(x) 🡺 FALSE
  + Forall x in Z>=2 P(x) 🡺 TRUE
  + Exists x in Zneg P(x) =🡺 True
  + Q(x,y) = “if x < y then x^2 < y^2”
    - Q(-2, 1) [True -> False, so overall FALSE]
    - Q(3, 8) [True -> True, so overall TRUE]
  + Write “The square of any odd integer is odd” using domain = Z, Odd(x)

3.2

Review – start with P(x) and Q(x,y) from above.

Is forall x in R P(x) true? (no) Is exists x in R P(x) true? (yes)

Is forall x in Z>=2 P(x) true?

Write the square of any odd integer is odd using domain = Z, Odd(x)

* Negations of quantified statements.
* Show definitions
  + Negation of universal stmt is existential
  + Negation of existential is universal
* Show examples in Symbols/English:
  + Someone in this room likes playing the bagpipes.
    - Exists x L(x) All x ~L(x)
  + All of the Harry Potter books are too long.
    - All x TL(x) Exists x ~TL(x)
  + Every integer is either even or odd.
    - All x in Z E(x) or O(x) Exists x in Z ~E(x) and ~O(x)
  + There is an even integer that is prime.
    - Exists x in Z E(x) and Prime(x) All x in Z ~E(x) or ~Prime(x)
* Negation of universal conditional
  + All CS majors take Discrete math.
    - All x CS(x) -> Discrete(x)
  + All Muppets are red.
    - All x Muppet(x) -> Red(x)
* Get symbols, negation in symbols, negation in words  
  + For all integers x, if x is even, then x + 1 is odd.
    - Negation = exists x in Z even(x) and ~Odd(x+1)
    - Negation = there is an integer such that x is even and x+1 is not odd.
  + Everyone over the age of 18 has a driver’s license.
    - Domain = all people
    - Over18(x) and HasDL(x)
* Universal statement is like an infinite and stmt
  + All of the people in this class are smart.
  + S(1) and S(2) and S(3)
    - Negation is not S(1) or not S(2) or not S(3) …
* Existential statement is like an infinite or.
  + There is someone in this class who wears glasses.
* Kind of like “for” loops.
* Vacuous truth:
  + A universal statement with an empty domain, or
  + A universal conditional statement for which nothing satisfies the hypothesis predicate.
  + All unicorns are pretty.
  + All unicorns are not pretty.
  + All of the hats I am wearing today are polka dotted.
* Mention: contrapositive, inverse, converse, only if, necessary, sufficient.
  + Universal conditional statement, contrapositive is equivalent.
  + Ex: For all ints x, if x > 2, then x^2 > 4.

Ex: If x is a computer program in C++, it is necessary for x to compile for x to run.  
  
Which of the following is a negation for “All dogs are loyal”? More than one answer may be correct.

Orig = all x in D L(x) OR all x in A D(x) -> L(x)

a. All dogs are disloyal. NO   
 All x in D ~L(x) all x in A D(x) -> ~L(x)

b. No dogs are loyal. NO (same as b)

c. Some dogs are disloyal. YES  
 Exists x in D ~L(x) exists x in A D(x) and ~L(x)

d. Some dogs are loyal. NO  
 Exists x in D L(x) exists x in A D(x) and L(x)

e. There is a disloyal animal that is not a dog. NO

Exists x in A ~L(x) and ~D(x)

f. There is a dog that is disloyal. YES Same as (c)

g. No animals that are not dogs are loyal. NO   
For all x in A ~D(x) -> ~L(x)   
Tricky: Can cats be loyal? No. Can fish be loyal? No. Only dogs can be loyal.

h. Some animals that are not dogs are loyal. NO

Exists x in A ~D(x) and L(x)

**3.3 (multiple quantifiers)**

* We can build more powerful statements by using multiple quantifiers
* Let’s define a predicate that uses two arguments:
  + L(x, y) = person x likes kind of music y
  + L(Alice, Pop)
  + L(Bob, Classical)
  + What does this mean? For all x in People L(x, Hip hop)
  + What does this mean? For all x in Music L(Alice, x)
* Do “Everybody likes some kind of music.”
  + “everybody” suggests universal
  + “some kind” suggests existential
  + We use both
  + L(x, y) = person x likes kind of music y
  + P =people, M = types of music
  + For all people p, there is a type of music m s.t. L(p, m)
* Do “There is a course that no student has ever taken.”
  + There is suggests a existential
  + “no” suggests universal
  + We use both
  + T(x, y) = student x has taken course y
  + S =students, C = courses
  + There exists a course c such that for all students s ~T(s, c)
* Order of quantifier matters if the quants are different.
  + Imagine the specific items from the sets that the quantifiers specify are chosen from left to right.
  + Therefore a quantified variable to the right can “see” the values of the quantified variables to the left.
  + Think of a quantifier like a for loop that is looping over all items within the domain of the variable that is quantified.
    - An inner for loop can access variables from the outer for loop, but not the other way around.
  + EX: everybody likes some kind of music.
    - Imagine you and a friend are playing a game. Your friend will name any person he or she wants, and you must name a kind of music that the given person likes. If you can do that, this is a true statement.
    - Therefore, the choice for the music can depend upon the name of the person; it can be different for every person.
  + EX: There is a Justin Bieber (or Taylor Swift) song that everybody dislikes.
    - Now you and your friend play the same game, but you (existential) have to pick your variable first. You have to pick a JB song so that whatever person your friend names, the person will dislike the song you picked.
    - One song has to work for all people.
* If quants are the same, order doesn’t matter.
* Translate these, swapping order of quantifiers (which won’t change meaning)
  + “There is a person who likes a Taylor Swift song.”
  + “All people dislike all of Taylor Swift songs.”
* Translate these:
  + “Everybody likes some TS song.” [same song or diff song?]
  + “There is some TS song that everyone likes.”
  + “There is some TS song that someone likes” = “Someone likes some TS song”
  + “Everybody likes all TS songs.” = “All JB songs are liked by everyone.”
* Smallest positive integer.
  + There is a smallest positive integer.
  + There is a positive integer, such that all other integers are smaller than it.
  + Exists x in POSINT for all y in POSINT (x <= y)
* Negations (translate orig to symbols, negate, then translate to English)
* Cap(s, c): The capital of state s is city c.
  + “All US states have a city that is its capital” [true]
  + for all s in STATES exists c in CITIES Cap(s, c)  
    - (neg) There is some state that does not have a capital [false]
    - (neg) Exists s in STATES forall c in C ~Cap(s, c)
  + “There is a city that is the capital of every state.” [false]
  + Exists c in CITIES forall s in STATES Cap(s, c)  
    - (neg) Every city has a state that the city is not the capital of. [true]
    - (neg) Forall c in CITIES exists s in STATES ~Cap(s, c)
  + “There are some cities that are not the capital of any state.” [true]
  + Exists c in CITIES forall s in STATES ~Cap(s, c)
    - (neg) Every city is the capital of some state. [false]
    - (neg) forall c in CITIES exists s in STATES Cap(s, c)
* IF TIME, go back to universal statement = infinite ands, existential statement = inf “or”s.

**3.4 proofs with quantified statements.**

* **Universal instantiation** (**with constant, but don’t mention**) = if something is true for every member of a set, it is true for any particular member of the set.
  + If a universal statement is true, then also know the statement is true with the variable replaced by any specific member of the set.
  + “All buildings on campus are made of stone.”
    - Therefore Briggs is made of stone.
    - Therefore Barret Lib is made of stone.
  + **DO: Prove Briggs and the RAT are both made of stone.** 
    - P1: forall x in Buildings Stone(x)  
      Prove: Stone(Briggs)  
      Prove: Stone(Rat)
* **Universal modus ponens**:
  + If you know that “if P is true for any member of a set, then Q is true for that same member”
  + And you know p is true for some specific member of the set,
  + Then q is true for that same member.
  + **DO: Given for any integer x, if prime(x) and greaterthan2(x), then odd(x)**
    - P1: forall x in Z Prime(x) and G2(x) -> Odd(x)  
      P2: Prime(5)
    - P3: G2(5)
    - Prove odd(5)
* **Univ. Modus Tollens**
  + DO: given not odd(4), not greater than 2(4) PROVE not prime(4)
    - P1: forall x in Z Prime(x) and G2(x) -> Odd(x)  
      P2: ~Odd(4)  
      P3: G2(4)
    - Prove: ~Prime(4)
* **Existential generalization (with constant)**
  + Given HasComputerLab(Briggs)
  + Show exists x in buildings, s.t. HasCompLab(x)
* **Univ Instantiation with a variable.**
  + Univ instantiation can be used in two ways
    - 1 – with a specific item you choose from the domain
    - 2 – with a variable that represents specific, but arbitrarily chosen member of the domain.
  + P1 – for all x in PplInRoom Smart(x)  
    Conc – Smart(a)
  + Says “a is some specific person in the room. I don’t know exactly who they are, but I know they are in this room. Furthermore, because all people in this room are smart, even though I don’t know exactly who “A” is, I know they’re smart”
* **Universal generalization**
  + Can only be used when thing inside parens is a variable that was universally instantiated.
  + P2 – for all x in PplInRoom Smart(x) -> PassClass(x)  
    Want to get: for all x in Ppl PassClass(x)  
      
    Proof: Smart(a) [univ inst P1]  
    PassClass(a) [univ mp 1, P2]  
    for all x in Ppl PassClass(x) [univ gen 2]
  + Says “I know a was universally instantiated as some specific person, but I don’t know which person. Since I know that whoever “A” is, they passed the class, I should be able to say that for everyone in the class, since it could be any of you.”
* **Existential instantiation (with var only)**
  + Existential instantiation lets you name something in a set that has specific attributes.
  + P1: Exists x in Z such that Prime(x) and Even(x)
  + Conc: Prime(a) and Even(a) [exist inst]
  + Says: Let variable a be the specific integer that is both prime and even. I don’t know exactly what the integer is, but it’s that specific one. [constrast with univ inst, where it could be any integer]
* **Existential generalization (with var)**
  + Existential gen can be used in 2 ways:
    - 1 – with specific constant
    - 2 – with a var that was existentially instantiated
    - Conclusion from prev example that Exists x in Z Prime(x)
* Multiple instantiation rules
  + When you need to instantiate multiple statements within a single proof, use these rules:
  + Universal instantiation can use any variable, even if you’ve already used it before
    - Because a universally instantiated variable doesn’t have any restrictions on it.
    - Ex: P1 – forall x in Class Smart(x)
    - P2 – forall y in Class Over17(x)
    - Instantiate both using “a”
  + **Existential instantiation must use a new variable that you haven’t used for any other instantiation. This is because existential instantiation places restrictions on what that variable means, while univ inst doesn’t.**
    - Ex: P1 – forall x in Class Smart(x)
    - P2 – exists y in Class Glasses(x)
    - Must instantiate P2 first, then P1.
* **SUMMARY**
  + Univ inst: can use with constant, or variable. With variable means variable value is specific, but chosen arbitrarily. Can create a new variable, or re-use an existing one.
  + Univ gen: only works with variable that was instantiated universally
  + Exist inst: can only use with NEW variable, variable defined to have those properties given in statement. Because we are imposing restrictions on this variable, we can’t use one that’s already been instantiated somewhere else, because the existing variable might have other restrictions.
  + Exist gen: can use with constant, or variable. With variable, can only do if variable was existentially instantiated.
* Proof strategies
* Proving universal statement
  + Universally instantiate everything
* Proving existential statement
  + Existentially instantiate first
  + Then universally instantiate as needed.

**PRACTICE**

Forall x P(x) -> Q(x)

Forall y R(y) or P(y)

Exists z ~R(z)

Prove Exists w Q(w)

Forall x S(x) -> ~P(x)

Forall y ~S(y) -> Q(y)

Forall z T(z) or Q(z)

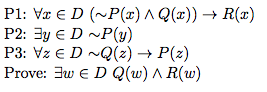
Prove Forall w P(w) -> Q(w) [ show 2 ways: (1) using hyp syll [t or q = ~t -> q] and (2) proof by contra]

Forall x R(x) or (Q(x) -> T(x))

Exists y P(y) or (~R(y) and Q(y))

Forall z ~T(z) -> ~P(z)

Exists w T(w)

from last year’s first test

