CMPSCI 250: Introduction to Computation

Lecture #6: Predicates and Relations
David Mix Barrington
16 September 2013

Predicates and Relations

- Statements That Include Variables
- Signatures and Templates
- Predicates Viewed as Boolean Functions
- Sets of Pairs and Tuples
- Storing Relations in a Computer
- Viewing Functions as Relations

Statements that Include Variables

- In propositional logic we view the atomic statements as either true or false -- we do not consider semantic relationships between statements.
- There is a similarity between "Cardie is a terrier" and "Biscuit is a terrier" that we may want to capture -- if we are told that "all terriers are dogs" then both of the statements will have consequences.

Predicates

- Rather than just give a name to each proposition, we might write T(c) and T(b), using the predicate T(x) to mean "x is a terrier". Then our new statement would be that "T(x) → D(x)" is always true (where D(x) means "x is a dog").
- Formally a **predicate** is a statement that *would* become a proposition if the value of some variable is supplied.
- The variables needed to define the meaning of the predicate are called **free variables**.

Signatures and Templates

- Methods in Java have **signatures** that indicate what arguments they take, and in what order. If we define `int foo (int x, String w)`, we know that the method `foo` takes two arguments, the first an `int` and the second a `String`.
- If we then make a method call `foo (7, "Cardie")`, we know that the interpreter will run the code of `foo`, replacing occurrences of `x` by 7 and occurrences of `w` by `"Cardie"`.

Signatures and Templates

- Predicates also have signatures that indicate how many free variables there are, what types they are, and what order they come in.
- A predicate also has a **template**, which is a statement about the free variables. (Often the types of the free variables are made clear in the template.) When we evaluate the predicate for particular objects, we substitute them for the corresponding free variables in the template and thus get a proposition that is true or false. We can think of the predicate as a *function with boolean output*.

Predicates as Boolean Functions

- In mathematics, we define general functions in the same way. In a calculus course, we might say "let f(x) be $3x^2 + 6x + 2$" and then later talk about f(x + Δx) by which we mean $3(x + \Delta x)^2 + 6(x + \Delta x) + 2$.
- The signature of this function is "x", a real variable, and the template is the expression $3x^2 + 6x + 2$.

Return Values of Predicates

- A mathematical or Java function may have any **return type,** but predicates always return booleans. Thus if the template involves numbers, there will also be a boolean operator like = or ≤.
- In fact these operators are *themselves* predicates, although we use **infix notation** rather then the usual functional notation. (For the latter we might write x ≤ y as LE(x, y).)

More Predicates

- When we write "x < y", we are giving an expression that will become a boolean once the values of x and y are supplied. Operators like < are **overloaded** for different argument types.
- Set builder notation involves a predicate, e.g., {x: x is a terrier}. The set is the collection of values (from the correct type) that make the predicate true.

Ordered Pairs

- If A and B are any data types, we can define **ordered pairs** of the form (a, b), where a is from A and b is from B. The pair is ordered because (a, b) is different from (b, a). (In fact (b, a) is not an ordered pair of that type, unless A = B.)
- Two ordered pairs (a, b) and (c, d) are *equal* if and only if a = c and b = d.

Direct Products

- We can similarly make ordered triples, ordered 4-tuples, or ordered k-tuples. A point in three-dimensional Euclidean space is represented by an ordered triple where each element is a real number, such as (2, π, -4.6).
- The set of all ordered pairs with first element from A and second element from B is called A × B, the **direct product** of A and B. Similarly we can have direct products of more than two sets -- Euclidean 3-space is the product $\mathbb{R} \times \mathbb{R} \times \mathbb{R}$.

Clicker Question #1

- Let D be the set {Biscuit, Cardie, Duncan} and let B be the set {Golden, Terrier}. Which of these sets is the direct product D × B?
- (a) {Biscuit, Cardie, Duncan, Golden, Terrier}
- (b) {(Biscuit, Golden), (Biscuit, Terrier), (Cardie, Golden), (Cardie, Terrier), (Duncan, Golden), (Duncan, Terrier)}
- (c) {(Cardie, Golden), (Duncan, Terrier)}
- (d) {(Biscuit, Golden), (Cardie, Golden), (Duncan, Golden), (Cardie, Terrier), (Duncan, Terrier)}

Answer #1

- Let D be the set {Biscuit, Cardie, Duncan} and let B be the set {Golden, Terrier}. Which of these sets is the direct product D × B?
- (a) {Biscuit, Cardie, Duncan, Golden, Terrier}
- (b) *{(Biscuit, Golden), (Biscuit, Terrier), (Cardie, Golden), (Cardie, Terrier), (Duncan, Golden), (Duncan, Terrier)}*
- (c) {(Cardie, Golden), (Duncan, Terrier)}
- (d) {(Biscuit, Golden), (Cardie, Golden), (Duncan, Golden), (Cardie, Terrier), (Duncan, Terrier)}

Pairs in Predicates

- Let's let D be a set of dogs and C a set of colors. Let the predicate P (d, c), with signature (D, C), have the template "Dog d has color c".
- Now consider the direct product D × C. An element (d, c) of D × C is exactly what we need to supply to determine whether P is true or false. (We could think of P as having one free variable, whose type is D × C.)

Relations

- The **relation** corresponding to P is the set {(d, c): P(d, c) is true} -- the set of pairs that make P true. Any subset of D × C is a relation "from D to C".
- Every predicate has a corresponding relation, and every relation has a corresponding predicate. For example, if $X \subseteq D \times C$, we can define a predicate $P_X(d, c)$ with template "$(d, c) \in X$".

Clicker Question #2

- Let D be a set of dogs and T be a set of toys. Let the predicate L(d, t) have template "dog d likes toy t". What set is the relation corresponding to this predicate?
- (a) the set of all dogs that like at least one toy
- (b) the set of all dog-toy pairs such that the dog in the pair likes the toy in the pair
- (c) the set of all toys that the dog d likes
- (d) the direct product $D \times T$

Answer #2

- Let D be a set of dogs and T be a set of toys. Let the predicate L(d, t) have template "dog d likes toy t". What set is the relation corresponding to this predicate?
- (a) the set of all dogs that like at least one toy
- (b) *the set of all dog-toy pairs such that the dog in the pair likes the toy in the pair*
- (c) the set of all toys that the dog d likes
- (d) the direct product $D \times T$

Arity of Relations

- A relation is **unary** if it is just a subset of a single set, **binary** if it is a set of ordered pairs, **ternary** if it is a set of ordered triples, and in general **k-ary** if it is a set of ordered k-tuples. Similarly we have unary, binary,..., k-ary predicates depending on their number of free variables.
- The **arity** of a relation or predicates is its number of free variables.

Storing Relations on Computers

- There are three basic ways to store a predicate/relation on a computer.
- We can have an **array of boolean values** where for every possible choice of the predicate's free variables, there is a boolean saying whether that choice is in the relation.
- We can have a **boolean method** that takes the values as arguments and computes a boolean telling whether the k-tuple of arguments is in the relation.

Storing Relations on Computers

- We can have a **list of the tuples** in the relation, so that we test the predicate by seeing whether the particular tuple is in the list.
- There are tradeoffs among these three methods. For example, the boolean array is generally the fastest but takes the most storage.

Clicker Question #3

- Let S be the set of 50 U.S. states, and let F be a set of 100 types of fireworks. Let the relation L(f, s) have the template "firework f is legal in state s". If I want to store this relation in a boolean array, how many entries do I need?
- (a) 100
- (b) 150
- (c) 5000
- (d) 2^{5000}

Answer #3

- Let S be the set of 50 U.S. states, and let F be a set of 100 types of fireworks. Let the relation L(f, s) have the template "firework f is legal in state s". If I want to store this relation in a boolean array, how many entries do I need?
- (a) 100
- (b) 150
- (c) *5000*
- (d) 2^{5000}

Viewing Functions as Relations

- In computing, we usually think of a function from A to B as an entity that takes **input** of some particular type A and produces **output** of some particular type B.
- The formal mathematical definition of "function", however, is different.
- In calculus, they may have tried to impress upon you that a function from $\mathbb{R}$ to $\mathbb{R}$ is a **set of ordered pairs** of real numbers.

Viewing Relations as Functions

- A curve in Cartesian coordinates may ($y = x^2$) or may not ($x = y^2$) represent a function, depending on whether it gives a unique output value for every input value. If it is not a function it is "just a relation".
- This is precisely the language we are using here. The set of points on the graph is a set of ordered pairs in $\mathbb{R} \times \mathbb{R}$, a binary relation.

Functions in Mathematics

- In mathematics a relation from A to B is called a **function from A to B** if for every element a of A, there is exactly one element b such that (a, b) is in the relation.
- Remember that "exactly one" means both "at least one" and "not more than one". Given an input value a, there must be some value b such that (a, b) is in the relation, but there *may not* be two such values.