Lecture 2: Sets, Functions, & Lambda Calculus

CS 181
Spring 2016
Kim Bruce

Some slide content taken from Unger and Michaelis

Mathematical Logic

- Propositional & predicate logic
  - Can you translate from English to symbolic forms?
    - John will work only if he is paid well.
    - Everyone loves Mary
    - John found a black cat.
- If little exposure to logic, see
  - "Logic in Action", free text on links page
    - Especially chapters 2.1-2.6 & 4.1-4.2 (but you'll find the other sections in chapters 1 to 4 interesting as well)

Bureaucracy

- On-line syllabus, lecture notes, homework
  - Turn in via Sakai
- Academic honesty

The Joy of Sets

- Mathematicians use sets to model pretty much everything.
  - Set is unordered collection of elements with no duplicates and where ordering is irrelevant.
  - Specify as a list {1,3,5,7}
    or as "set comprehension" $\{n \in N \mid n \text{ is odd} \& 1 \leq n \leq 7\}$
  - Set operations: $\cup$, $\cap$, -, and complement, written $\bar{A}$.
    - Usually specify against fixed universe, U. Then $\bar{A} = U - A$
  - $A \subseteq B$ iff every element $x \in A$ is also an element of B.
  - Important theorem: $A = B$ iff $A \subseteq B$ and $B \subseteq A$.

Set Products & Relations

- If A, B are sets,
  then $A \times B = \{(a,b) \mid a \in A, b \in B\}$
- Similarly for $A \times B \times C$, etc.
  - Write $A \times A \times \ldots \times A$ as A^n for n copies of A
- Unary relation on a set is subset of universe:
  - $I(\text{Dog}) = \{d \in U \mid d \text{ is a dog}\}$ *(where I means interpretation)*
- Binary relation is a subset of the set of pairs
  - $I(<) = \{(a,b) \in N^2 \mid a \text{ is smaller than } b\}$

Operations on Relations

- Suppose $R \subseteq A \times B$ and $T \subseteq B \times C$, then
  - $R^- = \{(b,a) \mid (a,b) \in R\} \subseteq B \times A$
    - *reverse of R*
  - $R \circ T = \{ (a,c) \mid \exists b \text{ s.t. } (a,b) \in R \text{ and } (b,c) \in T\} \subseteq A \times C$
    - *composition of R and T*

Properties of Binary Relations

- Let R be a binary relation on set S, i.e., $R \subseteq S \times S$.
  - R is reflexive iff for all a in S, $(a,a) \in R$.
  - R is symmetric iff for all a, b in R,
    if $(a,b) \in R$ then $(b,a) \in R$ (i.e. $R^- = R$)
  - R is transitive iff for all a, b, c in R,
    if $(a,b) \in R$ and $(b,c) \in R$ then $(a,c) \in R$
    i.e., $R \circ R \subseteq R$
- R is an equivalence relation if it is reflexive, symmetric, and transitive.

Functions

- … are special binary relations $R \subseteq A \times B$ such that if $(a,b) \in R$ and $(a,b') \in R$, then $b = b'$.
  - Unique element of range associated to each element of domain of R
  - $\text{domain}(R) = \{a \in A \mid \exists c \text{ such that } (a,c) \in R\} \subseteq A$
  - $\text{range}(R) = \{b \mid \exists a \text{ such that } (a,b) \in R\} \subseteq B$
  - Typically write $f(a) = b$ if $(a,b) \in f$ and $f: A \rightarrow B$
  - Function composition defined similar to relations
    - $g \circ f = \{(a,c) \mid \exists b \text{ s.t. } f(a) = b \text{ and } g(b) = c\}$

Oops, actually reverse of definition for relations!

Functions are Key to Computation

- Mathematicians define everything from sets, computer scientists define everything from functions, *including sets.*
- Let $R \subseteq U$ be a set, then the characteristic function of R, written f_R, is a function from U to Boolean s.t. $f_R(a)$ = true iff $a \in R$
  - Easy to go back and forth between R and f_R
  - We'll use relations at lowest levels, but use functions to combine them!

A Different View of Functions

- A function can be seen as instructions for computation.
- In this view $f(x) = (x + 1)^2$ is *not* the same as $g(x) = x^2 + 2x + 1$
- They represent different algorithms that nevertheless return the same value
- Extensional vs. intensional view

Lambda Calculus

Lambda Calculus

- Invented by Alonzo Church as model of computation.
  - Equivalent (& earlier than) Turing machines.
- Used as a tool to specify semantics of programming languages
  - ... and as of 1970's, natural language.
  - We will use as a way of composing meanings in natural languages.

Defining Functions

- In math *and* LISP *and* ML:
  - f(n) = n * n
  - (define (f n) (* n n))
  - (define f (lambda (n) (* n n)))
  - *in ML:* val f = fn n => n * n;
- In lambda calculus
  - λn. n * n
  - ((λn. n * n) 12) ⇒ 144

What Operations on Functions?

- If f is a function, can apply f to an argument b
  - f(b)
- If e is an expression then can form a function, by abstracting out a variable: λx.e
  - For example, λn. m + n, and then λm. λn. m + n

Questions?