Lecture 5
Sequences & Induction

David Garlan
Carnegie Mellon University

Models of Software Systems
15-671
Fall 1995

The Story So Far

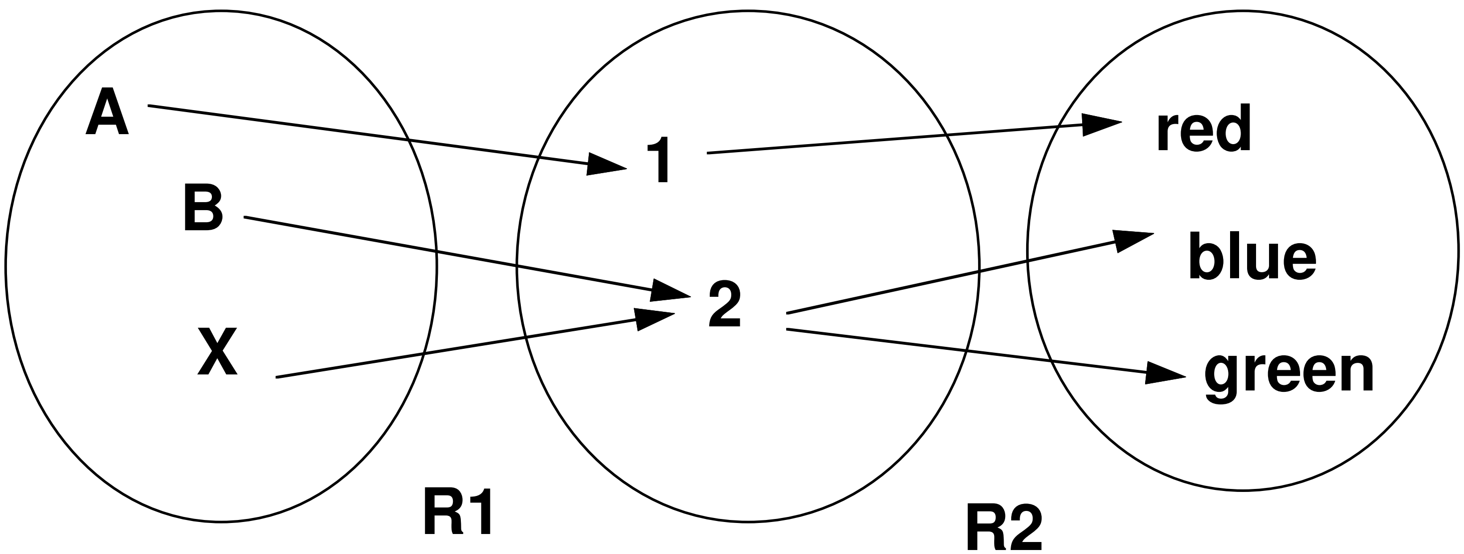
- **Formal System**
- **Logic**
  - > **propositional**
  - > **predicate**
- **Sets**
  - > **Set notation**
  - > **Relations**
  - > **Functions**

This Lecture

- In this lecture we will finish up our treatment of sets, relations, functions
  - > Relational composition
  - > Relations & functions as sets
  - > Using relations to model operations
  - > Lambda notation
- Then we look at a special kind of relation
  - > Sequences
  - > Operations on sequences
- Finally, we review the basic ideas of proof by induction
  - > Over integers
  - > Over other structures

Relational Composition

- If the range of one relation is the domain of another can form the composition (R1;R2)



Relations/Functions as Sets

- Since relations are just sets (of pairs) we can apply set operators to them.
- Example:
  - R1 = {(1,red), (2,blue)}
  - R2 = {(3,green), (2,blue)}
  - R1 $\cup$ R2 = {(1,red), (2,blue), (3,green)}
  - # R1 = 2

Question: is the union/intersection of two functions a function?

More on relations & functions

- **How to turn a relation into a function**

  r: S ™ T   {(2,red), (2,blue), (3,red)}   #r = 3

  f: S ¡ ∏ T   {(2,{red,blue}), (3,{red})}   #f = 2

- **Another notation for pairs (“maps to”)**

  {(2,red),(2,blue),(3,red)} = {2 ¢ red, 2 ¢ blue, 3 ¢ red}

- **Domain & range restriction**

  R == {(2, red),(5, blue),(3, red),(5, pink),(4, azure)}

  Primary == {red, blue, green}

  Even == {n: ˜ • 2n}

  Even ` R = {(2, red), (4, azure)}

  R ~ Primary = {(2, red),(5, blue),(3, red)}

Other ways of defining functions

- We can define a function by giving the rule for calculating its values
- To characterize the "square function" informally we might write

  $f(x) = x^2$

- We use the following form to do this

$$
\begin{array}{|l}
f: A \rightarrow B \\
\hline
\text{predicate over } f
\end{array}
$$

- So here we have

$$
\begin{array}{|l}
f: \mathbb{Z} \rightarrow \mathbb{Z} \\
\hline
\forall x: \mathbb{Z} \bullet f(x) = x^2
\end{array}
$$

Lambda Notation

- We can also use lambda notation
- The general form is

  f = λ x: T $\bullet$ expression

- In this case

  f = λ x: $\mathbb{Z}$ $\bullet$ x^2

- May also use multiple variables

  p

  lus = λ x,y: $\mathbb{Z}$ $\bullet$ x + y

Sequences

- **Sometimes we want to model ordered collections of objects**
  - Sets are inadequate because not ordered and don't allow duplicates
  - Tuples are inadequate because fixed length
  - Queue == {Rob, Peter, Mark, Mark, Matt}
- **A *sequence of X* is simply a function from positive numbers 1 .. N to elements of X**
  - written seq[X]
- **Example:**
  - Queue = <Rob, Peter, Mark, Mark, Matt>
    - = {(1,Rob), (2, Peter) ...}
    - = {1¢ Rob, 2 ¢ Peter, ...}
  - Empty sequence: <>

Operations on Sequences

- Let

  AskedQns == <Rob, Peter, Mark, Mark, Matt>

- Then
  - head(AskedQns) = Rob
  - tail(AskedQns) = <Peter, Mark, Mark, Matt>
  - front(AskedQns) = <Rob, Peter, Mark, Mark>
  - last (AskedQns) = Matt
  - <1,3,1> ^ <3,4> = <1,3,1,3,4>
- Of course, can access individuals:
  - AskedQns(1) = Rob
  - AskedQns(3) = Mark

Mapping a function over a set

- **Suppose**

  f is a function over elements of type X

  s is a sequence of type X

  s: seq[X] ; f: X $\rightarrow$ Y

- **To apply f to every element of s can simply write s composed with f.**

  s ∘ f

  Why?

- **Example:**

  s = <Arn, Angela, Josh, Gary>

  f = {Arn £ 21, Angela £ 22, Josh £ 19, Gary £ 21}

  s ∘ f = <21, 22, 19, 21>

Traces & Restriction

- **Traces**

  ReadEvents == {n: N • read.n}

  = {read.1, read.2, read.3 ...}

  WriteEvents == {n: N • write.n}

  Event == ReadEvents … WriteEvents

  trace: seq[Event]

  trace = <write.7, write.3, read.3, write.20, read.20, read.20>

- **Restriction**

  trace º WriteEvents = <write.7, write.3, write.20>

  trace º ReadEvents = <read.3, read.20, read.20>

Induction

- **Techniques for proving things of the form:**

  Å n: N • P(n)

  1. **Show it holds for all values by enumerating all cases (not promising for infinite sets)**
  2. **Assume not true and derive a contradiction**
  3. **Show P(i) holds for arbitrary "i", and use**

  Å-introduction

- **Alternative: Induction**

  Base case: Show true for n = 0

  Induction step: Show that if it's true for n=k, then it's true for n=k+1

Example

- **Worked out on board**

Structural Induction

- **Induction generalizes beyond natural numbers:**
  - > Show for base case(s)
  - > Show that can prove true for arbitrary entity if know it is true of subentities
- **Example 1: binary trees**
  - > Base case: show holds for tree consisting of a leaf only
  - > Induction step: show that it holds for two sub-tree then it holds for tree built out of them
- **Example 2: sequences**
  - > Base case: show holds for <>
  - > Show that if it holds for sequence s, then it holds for <x> ^ s

Examples

- **Worked out on board**