Theme 1: Abstract Reasoning

Lecture 1: Abstract Data Types & Recursive Functions

Matthieu Sozeau

Inria & Paris Diderot University – Paris 7

January 2014
http://www.pps.univ-paris-diderot.fr/~sozeau/teaching/MVF-2014-lecture1.pdf

Data manipulation

- Programs transform data
- They implement functions between inputs and outputs
- Examples of data domains: Booleans, Characters, Integers, Reals, Strings, Lists, Trees, etc.
- A function has a type (domain and co-domain):

$$f : D_1 \times \cdots \times D_n \to D$$

- Examples:

$$\begin{array}{rcl} \wedge & : & \text{Boolean} \times \text{Boolean} \to \text{Boolean} \\ + & : & \text{Nat} \times \text{Nat} \to \text{Nat} \\ \text{Sort} & : & \text{List[Nat]} \to \text{List[Nat]} \end{array}$$

- Types must be given precisely. This avoids many errors.

Defining Functions

- Finite data domains: Enumeration of its values
- Example:

$$\begin{array}{rcl} 0 \wedge 0 & = & 0 \\ 0 \wedge 1 & = & 0 \\ 1 \wedge 0 & = & 0 \\ 1 \wedge 1 & = & 1 \end{array}$$

- Not always practical, but possible in theory
- A more compact definition using a conditional construct:
  $x \wedge y = ($ `if` $x = 0$ `then` 0 `else` $y)$
- How to write functions over infinite domains ?
- We need more powerful constructs
- We need to give a structure to infinite data domains

Inductive Definition of (Potentially Infinite) Sets

- An element (object) is either basic or constructed from other objects.
- A set is defined by a set of constants and a set of constructors
- Example: The set Nat of natural numbers
  - Constant:
    $$0 : \text{Nat}$$
  - Constructor:
    $$s : \text{Nat} \to \text{Nat}$$
- Example of elements of Nat:

$$0, s(0), s(s(0)), s(s(s(0))), \ldots$$

- Notation: n abbreviates $s^n(0)$

The General Schema

- Given a set of constants $C = \{c_1, \dots, c_m\}$,
- Given a set of constructors of the form $: D^n \times A \to D$
- The set of element of D is the smallest set such that:
  - $C \subseteq D$
  - For every constructor $: D^n \times A \to D$, for every $d_1, \dots, d_n \in D$, and every $a \in A$, $(d_1, \dots, d_n, a) \in D$

The Domain of Lists

- Examples of lists:
  - $[2; 5; 8; 5]$ list of natural numbers
  - $[p; a; r; i; s]$ list of characters
  - $[[0; 2]; [2; 5; 2; 0]]$ list of lists of natural numbers
- The domain $List[?]$ parametrized by a domain $?$:
  - Constant:

$$[] : List[?]$$

  - Left-concatenation:

$$: \ ? \times List[?] \to List[?]$$

- Examples:
  - $0 \quad [] = [0]$
  - $2 \quad (5 \quad (8 \quad (5 \quad []))) = 2 \quad 5 \quad 8 \quad 5 \quad [] = [2; 5; 8; 5]$
  - $(0 \quad []) \quad [] = [[0]]$
  - $[] \quad [] = [[]] \neq []$
  - $(0 \quad []) \quad ((2 \quad []) \quad []) = [[0]; [2]]$

Defining functions over inductively defined sets

Let $f : Nat \to D$. Define $f(x)$, for every $x \in Nat$.

- Case spitting using the structure of the elements
  - $f(0) = ?$
  - $f(s(x)) = ?$
- Inductive definition (Recursion)

  Define $f(s(x))$ assuming that we know how to compute $f(x)$

- Similar to proofs using structural induction

  Prove $P(0)$, and prove that $P(s(x))$ holds assuming $P(x)$.

Recursion: An Example

- Addition $+ : Nat \times Nat \to Nat$
- Recursive definition

$$\begin{aligned} 0 + x &= x \\ s(x_1) + x_2 &= s(x_1 + x_2) \end{aligned}$$

- Computation

$$\begin{aligned} s(s(0)) + s(0) &= s(s(0) + s(0)) \\ &= s(s(0 + s(0))) \\ &= s(s(s(0))) \end{aligned}$$

Recursion: Another Example

- Append function @ : List[?] List[?] ! List[?]
- Example: [2; 5; 7]@[1; 5] = [2; 5; 7; 1; 5]
- Recursive definition

$$\begin{array}{rcl} []@ & = & ‘ \\ (a\ \ ‘_1)@_2 & = & a\ \ (‘_1@_2) \end{array}$$

- Computation:

$$\begin{array}{rcl} (2\ \ 5\ \ 7\ \ [])@(1\ \ 5\ \ []) & = & 2\ \ ((5\ \ 7\ \ [])@(1\ \ 5\ \ [])) \\ & = & 2\ \ 5\ \ ((7\ \ [])@(1\ \ 5\ \ [])) \\ & = & 2\ \ 5\ \ 7\ \ ([]@(1\ \ 5\ \ [])) \\ & = & 2\ \ 5\ \ 7\ \ 1\ \ 5\ \ [] \end{array}$$

Composition: Functions can call other functions

- Multiplication : Nat Nat ! Nat
- Recursive definition

$$\begin{array}{rcl} 0\ \ x & = & 0 \\ s(x_1)\ \ x_2 & = & x_2 + (\ x_1\ \ x_2) \end{array}$$

- Computation

$$\begin{array}{rcl} s^2(0)\ \ s^3(0) & = & s^3(0) + (\ s(0)\ \ s^3(0)) \\ & = & s^3(0) + (\ s^3(0) + (0\ \ s^3(0))) \\ & = & s^3(0) + (\ s^3(0) + 0) \\ & = & s(s^2(0)) + (\ s^3(0) + 0) = s(s^2(0) + \ s^3(0) + 0) \\ & = & s(s(s(0)) + (\ s^3(0) + 0)) \\ & = & s(s(s(0) + (\ s^3(0) + 0))) \\ & = & s(s(s(0 + (\ s^3(0) + 0)))) \\ & = & s(s(s(s^3(0) + 0))) \\ & = & s(s(s(s(s(s(0)))))) = \ s^6(0) \end{array}$$

Composition: Another Example

- Factorial function fact : Nat ! Nat
- Recursive definition

$$\begin{array}{rcl} \mathrm{fact}(0) & = & s(0) \\ \mathrm{fact}(s(x)) & = & s(x)\ \ \mathrm{fact}(x) \end{array}$$

- Computation

$$\begin{array}{rcl} \mathrm{fact}(s(s(0))) & = & s(s(0))\ \ \mathrm{fact}(s(0)) \\ & = & s(s(0))\ \ (s(0)\ \ \mathrm{fact}(0)) \\ & = & s(s(0))\ \ (s(0)\ \ s(0)) \\ & = & s(0)\ \ s(0) + \ s(0)\ \ (s(0)\ \ s(0)) \\ & = & s(0)\ \ s(0) + \ s(0)\ \ s(0) + 0\ \ (s(0)\ \ s(0)) \\ & = & s(0)\ \ s(0) + \ s(0)\ \ s(0) + 0\ \ (s(0)\ \ s(0)) \\ & = & s(0) + \ s(0) \\ & = & s(s(0)) \end{array}$$

Composition: Yet Another Example

- Reverse function Rev : List[?] ! List[?]
- Example: Rev([2; 5; 2; 1]) = [1; 2; 5; 2]
- Recursive definition:

$$\begin{array}{rcl} \mathrm{Rev}([]) & = & [] \\ \mathrm{Rev}(a\ \ ‘\ ) & = & \mathrm{Rev}(‘\ )@[a] \end{array}$$

- Computation

$$\begin{array}{rcl} \mathrm{Rev}([2; 5; 1]) & = & \mathrm{Rev}([5; 1])@[2] \\ & = & (\ \mathrm{Rev}([1])@[5])@[2] \\ & = & ((\ \mathrm{Rev}([])@[1])@[5])@[2] \\ & = & ([1]@[5])@[2] \\ & = & [1; 5]@[2] \\ & ::: & \\ & = & [1; 5; 2] \end{array}$$

Functions between different domains

- The Length function $|\ |$: List[?] $\to$ Nat

$$|[]| = 0$$
$$|a\ \ \ell| = s(|\ell|)$$

- Sum of the elements : List[Nat] $\to$ Nat

$$([]) = 0$$
$$(n\ \ \ell) = n + (\ell)$$

Inductive definition of functions: A General Schema

Let $f : D\ \ E \to F$.

- For every constant $c \in D$ and every $e \in E$, define $f(c, e)$ (as an element of F)
- For every constructor $: D^n\ \ A \to D$, for every $e \in E$, define $f(\ (x_1,\ \ , x_n, a), e)$ using a and $f(x_1, e),\ \ , f(x_n, e)$.

Proving facts about functions

- Neutral element:

$$\forall x \in \mathrm{Nat} : x\ \ s(0) = s(0)\ \ x = x$$

- Commutativity:

$$\forall x, y \in \mathrm{Nat} : x + y = y + x$$

- Associativity:

$$\forall x, y, z \in \mathrm{Nat} : x + (y + z) = (x + y) + z$$

- Distributivity:

$$\forall x, y, z \in \mathrm{Nat} : x\ \ (y + z) = (x\ \ y) + (x\ \ z)$$

- Idempotence:

$$\forall \ell \in \mathrm{List}[?] : \mathrm{Rev}(\mathrm{Rev}(\ell)) = \ell$$

- Kind of distributivity:

$$\forall \ell_1, \ell_2 \in \mathrm{List}[?] : \mathrm{Rev}(\ell_1 @ \ell_2) = \mathrm{Rev}(\ell_2) @ \mathrm{Rev}(\ell_1)$$

Structural Induction

Let $c_1, \ldots, c_m$ be the constants, and let $_1, \ldots, _n$ be the constructors.

$$\frac{\begin{array}{c} P(c_1) \\ \ldots \\ P(c_m) \\ \bigwedge_{i=1}^{K_1} P(x_i) \Rightarrow P(\ _1(x_1,\ \ x_{K_1})) \\ \ldots \\ \bigwedge_{i=1}^{K_n} P(x_i) \Rightarrow P(\ _n(x_1,\ \ x_{K_n})) \end{array}}{\forall x : P(x)}$$

Proving Neutrality of 1 for

$$8x\ 2\ \mathrm{Nat}: x \quad s(0) = s(0) \quad x = x$$

- Case $x = 0$.
  - $0 \quad s(0) = 0$
  - $s(0) \quad 0 = 0 + 0 \quad 0 = 0 \quad 0 = 0$
- Case $x = s(x')$. Induction Hypothesis: $x' \quad s(0) = s(0) \quad x' = x'$
  - $s(x^\theta) \quad s(0) = s(0) + (x^\theta \quad s(0)) = s(0) + x^\theta = s(0 + x^\theta) = s(x^\theta)$
  - $s(0) \quad s(x^\theta) = s(x^\theta) + (0 \quad s(x^\theta)) = s(x^\theta + (0 \quad s(x^\theta)) = s(x^\theta + 0) = s(x^\theta)$

Proving Commutativity of +

$$8x; y\ 2\ \mathrm{Nat}: x + y = y + x$$

- Case $x = 0$. $) \quad x + y = 0 + y = y$
  $\rightsquigarrow 8y\ 2\ \mathrm{Nat}: y = y + 0$?
  - Case $y = 0$: $y + 0 = 0 + 0 = 0$
  - Case $y = s(y^\theta)$:
    - Induction hypothesis: $y' = y' + 0$
    - $y + 0 = s(y') + 0 = s(y' + 0) = s(y') = y$
- Case $x = s(x')$. Induction Hypothesis: $8z\ 2\ \mathrm{Nat}: x' + z = z + x'$
  $\rightsquigarrow 8y\ 2\ \mathrm{Nat}: s(x') + y = y + s(x')$?
  - Case $y = 0$: $s(x^\theta) + 0 = s(x^\theta + 0) = s(0 + x^\theta) = s(x^\theta) = 0 + s(x^\theta)$
  - Case $y = s(y^\theta)$:
    - Induction hypothesis: $s(x') + y' = y' + s(x')$
    - $s(x') + s(y') = s(x' + s(y')) = s(s(y') + x') = s(s(y' + x'))$
    - $s(y') + s(x') = s(y' + s(x')) = s(s(x') + y') = s(s(x' + y'))$
    - $s(s(x' + y')) = s(s(y' + x'))$

Summary

- The first step in defining a function is to define its type (its domain and its co-domain).
- Infinite data domain can be defined inductively (set of constants and a set of constructors).
- Functions over infinite data domains by reasoning on the inductive structure of the data domains.
- Facts about recursive functions can be proved by reasoning on the inductive structure of the data domains.