COM S-342

Recitation 10/29/18 – 10/31/18

Today

- Lambda Calculus (what computation this lambda expression represents?)
- Reflang programming

Lambda Encoding

- Functions
  - f1 = (λ (m) (λ (n) (λ(f) (λ(x) (m f (n f x))))
    - ((f1 two) one) = ???
  - f2 = (λ (m) (λ (n) (λ(f) (λ(x) (m (n f) x))))
    - (f2 two) one) = ???
    - (f2 two) three) = ???

Lambda Encoding

- ((f1 two) one)
  - (((λ (m) (λ (n) (λ(f) (λ(x) (m f (n f x)))) two) one)
  - ((λ (n) (λ(f) (λ(x) (two f (n f x))))) one)
  - (λ(f) (λ(x) (two f (one f x)) // remember ((two g) w)?

Lambda Encoding

- ((f1 two) one)
  - (((λ (m) (λ (n) (λ(f) (λ(x) (m f (n f x)))) two) one)
  - ((λ (n) (λ(f) (λ(x) (two f (n f x))))) one)
  - (λ(f) (λ(x) (two f (one f x)) // remember ((two g) w)?
  - (λ(f) (λ(x) (two f (f x)) // remember ((two g) w)?
  - (λ(f) (λ(x) (f (f (f x))
  - = ???

Lambda Encoding

- ((f1 two) one)
  - (((λ (m) (λ (n) (λ(f) (λ(x) (m f (n f x)))) two) one)
  - ((λ (n) (λ(f) (λ(x) (two f (n f x))))) one)
  - (λ(f) (λ(x) (two f (one f x)) // remember ((two g) w)?
  - (λ(f) (λ(x) (two f (f x)) // remember ((two g) w)?
  - (λ(f) (λ(x) (f (f (f x))
  - = three
- ((f1 three) two)) = ???

Lambda Encoding

- (f2 two) one) = ???
- (f2 two) three) = ???

Lambda Encoding

- f1 = (plus m n)
- f2 = (mult m n)

Lambda Encoding

- You can encode
  - If conditions (e.g. ite)
  - Boolean functions (e.g. or asked in hw)
  - Etc.
- When we ask what computation this represents, is just what computation does this encode?

RefLang

- We add extensions to our language to support side effects
- These extensions focus on reading and writing memory locations
- We need two concepts and definitions
  - Heap: memory reserved for dynamic alloc
  - References: locations in the heap

Reference Expressions

- Represent malloc expressions
- (ref 1): stores the value 1 in a fresh location
- (free (ref 1)): deallocate the location for (ref 1)
- (deref (ref 1)): dereference a previously allocated memory location
  - $ (define loc (ref 3))
  - $ (deref loc)
  - $ 3

Reference Expressions

- (set! loc v): mutates the value of location *loc*, assigning value *v*
  - *$ (define loc (ref 5))*
  - *$ (deref loc)*
  - *$ 5*
  - *$ (set! loc 10)*
  - *$ (deref loc)*
  - *$ 10*

Examples RefLang

- $ (define loc (ref 5))
- $ (deref loc)
- $ 5
- $ (free loc)
- $ (deref loc)
- $ Error: null

Examples RefLang

- $ (define l1 (ref 10))
- $ (define l2 (ref 20))
- $ l2
- $??
- $ (free l2)
- $ (l2)
- $??
- $ (set! L2 20)
- $ 20
- $ (deref l2)
- $??
- $ (define l2 (ref 30)
- $ l2
- $??

Examples RefLang

- $ (define l1 (ref 10))
- $ (define l2 (ref 20))
- $ l2
- $ 1
- $ (free l2)
- $ (l2)
- $ Error: null
- $ (set! L2 20)
- $ 20
- $ (deref l2)
- $ 20
- $ (define l2 (ref 30)
- $ l2
- $ 30