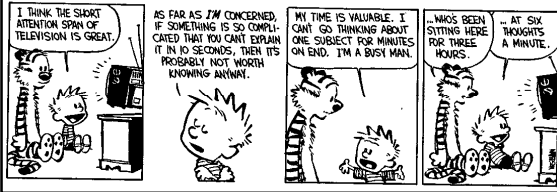


Parametric Polymorphism

Prof. Evan Chang
Meeting 23, CSCI 3155, Fall 2009



Announcements

- Midterms returned today in recitation
- HW8 due Thu at 11:55pm
- PROJ1 and HW6 grades posted
 - PROJ1 redo: interview before the last class (12/10)

2

Review Parameter Passing

One-Slide Summary

- **Procedures/functions/subprograms** allow code to be reused.
- On a call, **formal parameters** used by the callee are bound to **actual parameters** (or arguments) provided by the caller.
- In **eager evaluation**, actuals are evaluated before a call. In **lazy evaluation**, they are evaluated only when used.
- **Pass-by-value, pass-by-reference, pass-by-value-result** are parameter passing modes in an eager setting. **Pass-by-name** gives rise to lazy evaluation.

4

Pass-by-value

"value is copied in"

```
fun add (x,y) = x+y  
add (3-4, 5) →  
add (-1, 5) →  
[x→-1, y→5] x+y
```

```
val a = 24  
add (a+2, 1) → add (26, 1)
```

5

Pass-by-value-result

*pass in a value on call
and pass out value on return*

```
void f(inout int x) {  
    x = x + 1;  
    z = 42;  
}  
int y = 0  
f(y):  
print y
```

6

Pass-by-reference

simulate it
"pass address by value"

```
void f(int *x) { *x = *x + 1; }
int a = 0;
f(&a);
```

a 0x2 x 1

```
void f(int *x, int *y) {
  *y = 1; *x = *x + 1;
}
int a = 0;
f(&a, &a);
```

prints 2 a 0x2 a * 2

7

Pass-by-reference (C++)

```
void f(int &x, int &y) { x = x + 1; }
int a;
f(a, a);
```

a →

8

Exercises

- Write a MYSTERY program to distinguish between call-by-value and call-by-reference

```
VAR a : INTEGER;
PROCEDURE f(y : INTEGER) =
  BEGIN
    y := 1;
  END;
BEGIN
  a := 0;
  f(a);
  PRINT a;
```

if 0 then value
if 1 then ref / value-result

9

Exercises

- Write a MYSTERY program to distinguish between call-by-reference and call-by-value-result

```
VAR a : INTEGER;
PROCEDURE f(x:INT, y:INT) =
  BEGIN
    y := 10;
    x = x + 1;
  END;
BEGIN
  a := f(a, a);
  PRINT a;
END
```

reference value-result

10

Lazy evaluation: Pass-by-name

Jan: "don't evaluate argument to value, pass the code to callee"

→ won't evaluate if argument not used

```
fun f x = 1 fun loopforever() = loopforever()
f(loopforever()) → / (lazy evaluation)
f(loopforever()) → ... → ... (eager evaluation)


11


```

Lazy evaluation: Pass-by-name

12

Exercise

- Write a MYSTERY program to distinguish between call-by-name and call-by-value

13

Functional Languages: Parameter Passing

$$(\lambda x. e_1) v_2 \rightarrow [v_2/x]e_1 \quad \text{Eager}$$

$$\begin{array}{l} (\lambda x. e_1) e_2 \rightarrow (\lambda x. e_1) e_2' \quad \text{if } e_2 \rightarrow e_2' \\ e_1 e_2 \rightarrow e_1' e_2 \quad \text{if } e_1 \rightarrow e_1' \end{array}$$

$$\begin{array}{l} (\lambda x. e_1) e_2 \rightarrow [e_2/x]e_1 \quad \text{Lazy} \\ e_1 e_2 \rightarrow e_1' e_2 \quad \text{if } e_1 \rightarrow e_1' \end{array}$$

14

For Next Time

- Reading
- Forum comment
- HW8

15