

CSC 347 - Concepts of Programming Languages

Strict and Nonstrict Evaluation

Instructor: Stefan Mitsch



Learning Objectives

- ❓ How much of an expression should be evaluated when computing a result?
 - Perform expression evaluation
 - Perform function calls and function argument evaluation



Strict and Nonstrict Operators

- A *strict* construct evaluates all of its operands before it runs
- Name some **strict constructs**
 - Arithmetic operators: `+`, `-`, `*`, `/`, ...
 - Comparison operators: `<`, `<=`, `==`, `!=`, ...
 - Bitwise operators: `|`, `&`, `~`, `^`
 - Function calls: `f(x, y)`



Strict and Nonstrict Operators

- A *strict* construct evaluates all of its operands before it runs
- Name some **non-strict constructs**
 - `e1 && e2` is strict in `e1`, but not `e2`
 - `e1 || e2` is strict in `e1`, but not `e2`
 - `if e1 then e2 else e3` is strict in `e1`, but not `e2` or `e3`



Conditional Expression

Conditional expression `if e1 then e2 else e3`

- Evaluate `e1` ; if `true` then evaluate `e2` , else evaluate `e3`

```
1 for i <- 0 to 10 do
2   val z = (if scala.util.Random.nextInt % 2 == 0
3             then f()
4             else g())
5   println(s"z=$z")
```

- Sometimes `f()` is invoked, sometimes `g()`



Conditionals in C

- Conditional statement: executes `return 1` or `return n*...`, never both

```
1 int fact (int n) {  
2     if (n <= 1) {  
3         return 1;  
4     } else {  
5         return n * fact (n - 1);  
6     }  
7 }
```

- Conditional expression: must be non-strict, otherwise non-terminating recursion

```
1 int fact (int n) {  
2     return (n <= 1) ? 1 : n * fact (n - 1);  
3 }
```

? What Happens?

- Function calls are strict

```
1 def fcond(b: Boolean, t: Int, f: Int) : Int =  
2   if b then t else f  
3  
4 for i <- 1 to 10 do  
5   val z = fcond(scala.util.Random.nextInt % 2 == 0,  
6               f(),  
7               g())  
8   println(s"z=$z")
```

- Now both `f()` and `g()` are invoked

? What Happens?

- *Thunks*: pass-by-name for explicit nonstrict evaluation of function arguments

```
1 def fcond(b: Boolean, t: => Int, f: => Int) : Int =  
2   if b then t else f  
3  
4 for i <- 1 to 10 do  
5   val z = fcond(scala.util.Random.nextInt % 2 == 0,  
6                 f(),  
7                 g())  
8   println(s"z=$z")
```

- Thunks are parameterless functions



Summary

Strict Evaluation

- Evaluates **all operands** to determine value of an expression
 - Arithmetic operators etc.
 - Function arguments

Nonstrict Evaluation

- Evaluates **only necessary operands** to determine value
 - Conditional expression
 - Shortcut evaluation of boolean expressions
- Thunks: request explicit nonstrict evaluation of function arguments