

3007 Final Exam Review

William Findlay

April 22, 2018

Contents

1	Definitions	3
1.1	Imperative vs Declarative	3
1.1.1	Imperative	3
1.1.2	Declarative	3
1.2	Scope vs Visibility	3
1.2.1	Scope	3
1.2.2	Visibility	3
1.3	Applicative Order Evaluation vs Normal Order Evaluation	4
1.3.1	Applicative Order Evaluation	4
1.3.2	Normal Order Evaluation	4

1 Definitions

1.1 Imperative vs Declarative

1.1.1 Imperative

- Series of instructions
- Iterative functions
- Command driven, statement oriented
- Procedural
 - C
 - Pascal
 - Assembly
- Object oriented
 - C++
 - Java

1.1.2 Declarative

- No side effects
- Focus on relations
- “What to get” instead of “How to get”
- Order of statements *shouldn't* matter
- Examples:
 - SQL
 - Prolog
 - Regex

1.2 Scope vs Visibility

1.2.1 Scope

- The set of expressions for which the variable *exists*
- In lexical scoping
 - variables in the scope we were *defined* in
 - and local variables
 - who uses this?
 - * C-family languages
 - * Scheme
 - * Algol
- In dynamic scoping
 - variables in the scope we were *called* in
 - and local variables
 - who uses this?
 - * early LISP
 - * APL
 - * BASH

1.2.2 Visibility

- The set of expressions for which the variable *can be reached*

- If we **declare a local variable** with the *same name* as a variable in enclosing scope
 - that enclosing scope variable is now hidden
 - all references to *name* are to our locally scoped variable instead

1.3 Applicative Order Evaluation vs Normal Order Evaluation

1.3.1 Applicative Order Evaluation

- **Strict evaluation**
- Evaluate an expression *before* it is passed in as an argument
 - go as deep as you can until you hit primitives, then evaluate and go back
 - as deep into the nest as possible and work backwards

```
(double (* (+ 1 3) 4))
(double (* 4 4))
(double 16)
(* 16 2)
32
```

1.3.2 Normal Order Evaluation

- **Lazy evaluation**
- Evaluate an expression *only* when its value is needed
 - first **expand**, then **reduce**

```
(double (* (+ 1 3) 4))
(* (* (+ 1 3) 4) 2)
(* (* 4 4) 2)
(* 16 2)
32
```