Programming Languages and Types

Exercise 11

Yi Dai

January 31, 2013

1 λ-Calculus vs. λ-Based Language

Many languages have claimed to be based on λ-calculus. Being λ-calculus-based does *not* mean they include λ-calculus as a subset or they offer every feature λ-calculus offers. Actually there are quite some differences.

1. λ-calculus uses *static scoping*. This sounds strange because nowadays almost all languages (whether functional or imperative) that supports functions (whether higher-order or just first-order), use static scoping. However, this point deserves an emphasis because the first functional language that claimed to be λ-calculus, mamely *Lisp*, ironically uses *dynamic scoping*. This *bug* or *mis-feature* were not discovered for over two decades until Scheme came out. Then after over one decade, static scoping was finnally written into the Common Lisp ANSI standard. What a historical lession!

2. λ-calculus supports *symbolic computation* while most λ-calculus-based languages do not.[1] Symbolic computation allows a free variable to compute to itself if there is no binding associated to it. So in λ-calculus, the variable x evaluates to itself. Whereas in most functional languages, you get a run-time error complaining that the variable is *unbound*.

3. λ-calculus features full β-reduction. In other words, evaluation goes under *lambda*. Whereas in most functional language, evaluation stops

[1] Actually, most languages do not support symbolic computation. Exceptions may be those languages that were designed for AI, such as Lisp and Prolog. Though their support is via a new data type for symbols, which is kinda *indirect*.

in front of a *lambda*. Instead of a *lambda*-abstraction with a fully-reduced body, you get a closure. Although we gain some efficiency, we also miss part of the beauty of λ-calculus. For example, we cannot see how Church-numerals work.

2 Nameless λ-Calculus

2.1 α-Conversion: Pros and Cons

2.2 λ-Calculus Using De-Bruijn Indices

2.3 Evaluation Semantics for λ_{bv}

3 Substitution vs. Environment

3.1 Substitution-Based Evaluation Semantics for λ_{ao}

3.2 Environment-Based Evaluation Semantics for λ_{bn}

4 Computational λ-Calculus

4.1 λ-Calculus Extended with Monads

4.2 Typing λ_c