SICP notes

Mark Tropin

September-October 2023

DISCLAIMER: This is not a summary of SICP (Structure and Interpretation of Computer Programs). These are my personal notes, i.e. non-obvious things mentioned in the book that I find interesting.

Chapter 1.

To evaluate a combination,

- evaluate the operator
- evaluate the operands
- apply the operator to the operands

It is important to note that:

- EVALUATE is recursive.
  - We evaluate the operator and the operands using the same logic.
- EVALUATE means "reduce to a value".
  - Operators are reduced to built-in instructions ($\oplus$, $\otimes$, ...) that correspond to machine instructions (CPU instruction set). Operands are reduced to numeric values (by applying operators to combinations or looking up their value in the environment).
    $\Longrightarrow$ "value" means primitive. "value" means irreducible.
    No more evaluation rules apply. See TAPL theorems 3.5.7, 3.5.8.

p.12, footnote 14:

This is how `let` works. `Let` basically adds a new symbol to the *local* environment. It is used in evaluating the final expression in a block of code. This value (because it is local) is then destroyed.

End of section 1.1.4: You can't tell a user-defined compound procedure from a primitive. That is the function of the environment: to make user-defined definitions usable like primitives. It is essentially abstracting away definitions by *pointing* to them via names.

Evaluation strategies.

Section 1.1.5 defines normal-order evaluation and applicative-order evaluation. In Scala we have *call-by-name* and *call-by-value*.

- call-by-value $\Longleftrightarrow$ applicative-order evaluation
- call-by-name $\Longleftrightarrow$ normal-order evaluation
  - applicative-order evaluation $\Rightarrow$ "evaluate the arguments and then apply"
  - normal-order evaluation $\Rightarrow$ "fully expand and then reduce"

TODO:

- describe linear recursion and linear iterative process
- describe bound variables and free variables
- describe lambda semantics and let semantics
- describe interesting footnote at the end of the chapter