Structure and Interpretation of Computer Programs: Learning Notes

Florian Legendre

April 24, 2020

Contents

Chapter 1

Building Abstractions with Procedures

Process / Program / Data. Processes manipulate data by following rules given by a program. Programming language = like words of a sorcerer when conjuring. Symbolic expressions carefully composed.

> Novice must learn to understand and anticipate the consequences of their conjuring. Even small errors in programs can have complex and unanticipated consequences. [...] Master software engineers have the ability to organize programs so they can be reasonably sure that the resulting processes will perform the tasks intended. They can visualize the behavior of their systems in advance. They know how to structure programs so that unanticipated problems do not lead to catastrophic consequences, and when problems do arise, they can debug their programs. Well-designed computational systems [...] are designed in a modular manner, so that the parts can be constructed, replaced and debugged separately.[1]

What this quote indicates, in my opinion, is a mindset to acquire in order to be able to make reliable, professionnal programs. This mindset involves:

- Understanding the consequences of what we're writing so that we can anticipate possible problems
- Having methods to be reasonably sure of what will happen
- Structuring programs so that unanticipated problems do not lead to catastrophic consequences
- Implementing ways to help debugging in case of such problems
- Structuring programs so that they are easier to maintain

Language = made to express a set of thoughts. Natural language = everyday thoughts, mathematics = descriptions of quantitative phenomenas, programming language (here scheme lisp) = procedural thoughts.

Significant lisp (LISt Processing) features according to the authors:

- procedure (= description of a process, $\simeq$ subroutine or function?) can be represented and manipulated as data. $\Rightarrow$ possibility to blur the distinction between "passive" data and "active" processes. $\Rightarrow$ new interesting techniques relying on this lack of distinction.
- Because procedures are data, programs can be data too $\Rightarrow$ we can write interpreters and compilers

1.1 The Elements of Programming

Bibliography

[1] Harold Abelson and Gerald Jay Sussman with Julie Sussman. *Structure and Interpretation of Computer Programs*. Ed. by The MIT Press. 2nd edition. 1996.