SICP - conclusion

April 18, 2008 at 16:54 Tags SICP

I've completed the SICP reading project, which I began on June 19th, 2007. In hindsight, it seems like a very long time (almost a year!), during which this was my main hacking project at home. So, in this conclusion I want to examine my progress, and compare the result with my initial goals.

The original plan was:

1. Read the book
2. See all the video lectures by Sussman and Abelson themselves
3. Do most of the interesting exercises in the book
4. Do some of the larger projects listed here and here

Let's see how I've managed:

1. Done
2. Done
3. This is the point I'm most happy with. I've completed the vast majority of the exercises in SICP and posted them online. Of the 356 exercises in the book, I've skipped about 20, so I've completed 94% of the exercises – which is far above my initial plans.
4. I didn't do this, but there's no need, having completed so many of the exercises. I feel I gained a completely solid understanding of the material, and won't gain much by doing the projects.

Also, I originally planned to reimplement all the code do all the exercises in Common Lisp. Later I changed my mind and decided to use PLT Scheme for some of them. Eventually, I've been using both languages interchangeably, which is a good thing, as I got some practice with both.

Here are some numerical statistics that emphasize the magnitude of this endeavor:

1. I've written 52 blog posts (not including this one) in the SICP category, spread over 10 months.
2. The combined total length of my posts (including code snippets) is 66,265 words. For comparison, Jack London's "Son of the Wolf" is 50K words long.
3. Some of the large projects from the book I've re-implemented in wholeness: a constraint propagation solver, an evaluator (interpreter) for Scheme, a generic object-oriented arithmetic package including complex and polynomial arithmetic, Huffman encoding, an interpreter for a logic programming language similar to Prolog (as a DSL on top of Lisp), a picture-language interpreter, a simulator for digital circuits, a symbolic differentiation package, a virtual machine for a simplified pseudo-assembly DSL, an interpreter for Scheme written in this pseudo-assembly, and finally, a compiler for Scheme that spits out pseudo-assembly code.
4. Counting with the cloc tool (Count Lines Of Code), the total physical LOC count[1] for the code I've written during this time: 7,300 LOC of Common Lisp, 4,100 LOC of Scheme. If you prefer raw LOCs[2], it's 10,800 LOC of Common Lisp, 5,600 LOC of Scheme. So this is more than 10K LOC of Lisp code, any way you look at it, which is a lot, since Lisp is a very expressive high-level functional language.

A word on SICP and its exercises. Abelson and Sussman have created a masterpiece, a book initially written in the 1980s, and that still hasn't lost one bit of relevance. Everything it contains must, **must** be learned and understood by any aspiring programmer. It teaches algorithms and data structures, good programming style, provides some contact with large systems, experimenting with their implementation and modification. You will learn about functional programming, imperative programming, object-oriented programming in it. You will learn how to implement interpreters, compilers, arithmetic systems, simulators, a whole virtual machine in it, and much more.

The exercises of SICP are essential to understanding. Looking back at the work I've done on the book, I don't think I would have understood it near so well without doing a lot of the exercises. There's only so much material that can be gained from reading. Getting your hands dirty with code is essential to true understanding. And the authors of SICP brought this concept to perfection, with their excellent exercises, that are an unreplaceable companion to the book. Although they're not 100% perfect, for the most part the exercises are very well thought out and tuned to aid

understanding and practice writing parts of large systems.

[1] Physical LOC: lines of code, excluding comments and blank lines. To test my Common Lisp and Scheme code counts with `cloc.pl`, I used these commands:

```
cloc.pl --force-lang="Lisp",lisp *.lisp
cloc.pl --force-lang="Lisp",scm *.scm
```

[2] Including comments and blank lines, as output by `wc -l`.

For comments, please send me an email.

Back to top