How to implement a programming language

**Interpretation**

An **interpreter** written in the **implementation language** reads a program written in the **source language** and **evaluates** it.

**Translation (a.k.a. compilation)**

An **translator** (a.k.a. **compiler**) written in the **implementation language** reads a program written in the **source language** and **translates** it to an equivalent program in the **target language**.

But now we need implementations of:

- **implementation language**
- **target language**

---

**Lisp/Racket and Implementation (2)**

Interpretation, Translation, and everything in between

- Programs as Data
- If time: Implementing Racket in Racket
  - hands-on
  - how Lisp was first implemented

How to implement a programming language

Can describe by deriving a “proof” of the implementation using these inference rules:

<table>
<thead>
<tr>
<th>Interpreter Rule</th>
<th>P-in-L program</th>
<th>L interpreter machine</th>
<th>P machine</th>
</tr>
</thead>
<tbody>
<tr>
<td>Translator Rule</td>
<td>P-in-S program</td>
<td>S-to-T translator machine</td>
<td>P-in-T program</td>
</tr>
</tbody>
</table>
Implementation Derivation Example

Prove how to implement a "251 web page machine" using:
- 251-web-page-in-HTML program (a web page written in HTML)
- HTML-interpreter-in-C program (a web browser written in C)
- C-to-x86-compiler-in-x86 program (a C compiler written in x86)
- x86 interpreter machine (an x86 computer)

No peaking ahead!

Implementation Derivation Are Trees

And so we can represent them as nested structures, like nested bulleted lists:

- 251 web page machine (I)
  - 251-web-page-in-HTML program
  - HTML interpreter machine (I)
    - HTML-interpreter-in-x86 program (T)
      - HTML-interpreter-in-C program
      - C-to-x86 compiler machine (I)
        - C-to-x86-compiler-in-x86 program
        - x86 computer
      - x86 computer

Implementation Derivation Example Solution

No peaking ahead!

Metacircularity and Bootstrapping

Many examples:
- Lisp in Lisp / Racket in Racket: eval
- Python in Python: PyPy
- Java in Java: Jikes RVM, Maxine VM
- ...
- C-to-x86 compiler in C

How can this be possible?

Key insights to bootstrapping:
- The first implementation of a language cannot be in itself, but must be in some other language.
- Once you have one implementation of a language, you can implement it in itself.
Metacircularity Example 1

Suppose you are given:
- Racket-in-SML interpreter
- SML machine
- Racket-in-Racket interpreter
How do you run the Racket-in-Racket interpreter?

Metacircularity Example 2

Suppose you are given:
- C-to-x86-translator-in-x86 program (a C compiler written in x86)
- x86 interpreter machine (an x86 computer)
- C-to-x86-translator-in-C
How do you compile the C-to-x86-translator-in-C?

Compiler

```
if (x == 0) {
    x = x + 1;
}
...
```

Typical Compiler

```
cmp (1000), $0
bne L
add (1000), $1
L:
...
```
Interpreters

Interpreters

Interpreters vs Compilers

Interpreters

No work ahead of time
Incremental
maybe inefficient

Compilers

All work ahead of time
See whole program (or more of program)
Time and resources for analysis and optimization

Compilers... whose output is interpreted

Java Compiler

Java Compiler

if (x == 0) {
  x = x + 1;
}

... 

load 0
ifne L
load 0
inc
store 0
L:
...

(compare compiled C to compiled Java)
Interpreters... that use compilers.

JIT Compilers and Optimization

Virtual Machine Model

Remember: language != implementation

- Easy to confuse "the way this language is usually implemented" or "the implementation I use" with "the language itself."
- Java and Racket can be compiled to x86
- C can be interpreted in Racket
- x86 can be compiled to JavaScript
- Can we compile C/C++ to Javascript? [http://kripken.github.io/emscripten-site/]
Next Topic: Metaprogramming in SML

- PostFix in SML (see `postfix.sml`)
- A sequences of expression languages implemented in SML that look closer and closer to Racket:
  - Intex
  - Bindex
  - Valex
  - HOFL (higher-order functional language)