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

Metaprogramming: Interpretation

Interpreters
Metaprogramming: Translation

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

Compiler

Java Compiler

(if compare compiled C to compiled Java)
Compilers... whose output is interpreted

Doesn't this look familiar?

Interpreters... that use compilers.

JIT Compilers and Optimization

Virtual Machine Model

- HotSpot JVM
- Jikes RVM
- SpiderMonkey
- v8
- Transmeta
- ...

JIT compiler

Ahead-of-time compiler
Typical Compiler

Source Program
Lexical Analyzer
Syntax Analyzer
Semantic Analyzer
Intermediate Code Generator
Code Optimizer
Code Generator
Target Program

Analysis
Synthesis

How to implement a programming language

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

Interpreter Rule

P-in-L program → L interpreter machine

P machine

Translator Rule

P-in-S program → S-to-T translator machine

P-in-T program

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-translator-in-x86 program (a C compiler written in x86)
• x86 interpreter machine (an x86 computer)

No peaking ahead!

Implementation Derivation Example Solution

We can omit some occurrences of “program” and “machine”:
### Implementation Derivation Are Trees

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

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

### Metaprogramming: Bootstrapping Puzzles

**How can we write Scheme interpreter in Scheme?**

**How can we write a Java-to-x86 compiler in Java?**

### Metacircularity and Bootstrapping

Many examples:
- Lisp in Lisp / Scheme in Scheme/Racket in Racket
- Python in Python: PyPy
- Java in Java: Jikes RVM, Maxine VM
- ...
- C-to-x86 compiler in C
- `eval` construct in languages like Lisp, JavaScript

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: Solution
Suppose you are given:
- Scheme-interpreter-in-Python program
- Python machine
- Scheme-interpreter-in-Scheme program
How do you create a Scheme interpreter machine using the Scheme-interpreter-in-Scheme program?

But why create Scheme interpreter machine #2 when you already have Scheme-interpreter machine #1?

Metacircularity Example 1: More Realistic
Suppose you are given:
- Scheme-subset-interpreter-in-Python program (implements only core Scheme features; no desugaring or other frills)
- Python machine
- Full-Scheme-interpreter-in-Scheme program
How do you create a Full-Scheme interpreter machine using the Full-Scheme-interpreter-in-Scheme program?

Metacircularity Example 2: Problem
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-subset program
How do you compile the C-to-x86-translator-in-C?

But why create C-to-x86-translator-in-x86 program #2 when you already have C-to-x86-translator-in-x86 program #1?

Metacircularity Example 2: Solution
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 program
How do you compile the C-to-x86-translator-in-C?
Metacircularity Example 2: More Realistic

Suppose you are given:
• C-subset-to-x86-translator-in-x86 program (a compiler for a subset of C written in x86)
• x86 interpreter machine (an x86 computer)
• Full-to-x86-translator-in-C-subset program (a compiler for the full C language written in a subset of C)

How do you create a Full-C-to-x86-translator machine?

Full-C-to-x86-translator machine (I)
• Full-C-to-x86-translator-in-x86 program (T)
  • Full-C-to-x86-translator-in-C-subset
  • C-subset-to-x86-translator machine (I)
    • C-subset-to-x86-translator-in-x86 program
    • x86 computer
• x86 computer

A long line of C compilers

C-version_n-to-target_n-translator machine (I)
• C-version_n-to-target_n-translator program in target_n-1 (T)
  • C-version_n-to-target_n-translator program in C-version_n-1
  • C-version_n-1-to-target_n-1 translator machine (I)
    • C-version_n-1-to-target_n-1-translator program in target_n-2 (T)
    • C-version_n-1-to-target_n-1-translator program in C-version_n
    • C-version_n-1-to-target_n-1-translator machine in assembly_0
      • assembly_0 computer
    • target_1 computer
    • target_2 computer
• target_n-2 computer
• target_n-1 computer
• target_n computer

More Metaprogramming in SML

• We’ve already seen PostFix and Intex SML
• A sequences of expression languages implemented in SML that look closer and closer to Racket:
  • Bindex: add naming
  • Valex: add more value types, dynamic type checking, desugaring
  • HOFL: first class function values, closure diagrams

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/emshepten-site/