1 Language Primer

citations for this history are fragmented across the internet; should consolidate some kernel of citation from it

Factor is a rather young language created by Slava Pestov in September of 2003. Its first incarnation targeted the Java Virtual Machine (JVM) as an embedded scripting language for a game. As such, its feature set was minimal. Factor has since evolved into a general-purpose programming language, gaining new features and redesigning old ones as necessary for larger programs. Today's implementation sports an extensive standard library and has moved away from the JVM in favor of native code generation. In this section, we cover the basic syntax and semantics of Factor for those unfamiliar with the language. This should be just enough to understand the later material in this thesis. More thorough documentation can be found via Factor's website.

figure out section / chapter names

cite chapter?

cite factor-code.org

1.1 Stack-Based Programming

Like Reverse Polish Notation (RPN) calculators, Factor's essential evaluation model uses a global **stack** upon which operands are pushed before operators are called. This lends itself to **postfix** notation, in which operators are written after their operands. For example, instead of `1 + 2`, we write `1 2 +`.

figure out how terms will work

figure out how terms will work

- Basic syntax & semantics
  - Postfix example (RPN with a stack, `1 2 +` sort of thing)
  - Other stack-based languages
  - Parsing algorithm
    - "
    - token
      - ordinary word (terminology, incl. "vocabulary")
      - parsing word
    - number
  - Example ordinary words: stack shufflers
  - Example parsing words: quotations
- Concatenative programming
  - "Pipeline code"
  - Whitespace = function composition
  - Point-free style
  - Origin of name "Factor"
- Stack effects
  - Basic stack effects: stack shufflers illustrated
  - Complex stack effects: row polymorphism & types

    - Stack checker
- Control flow
    - Combinators
        * if
        * each
        * while
    - Dataflow combinators
        * Dip/keep
        * Cleave
        * Spread
        * Apply
- Object system
    - tuples
    - generics & methods
- Libraries & metaprogramming
    - Results of evolution
    - locals?
    - fry?
    - macros?
    - functors?
    - ffi?