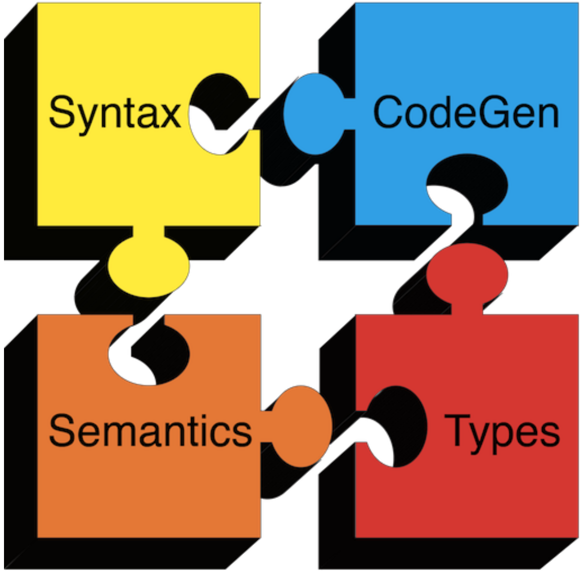

Compilers

Semantics Overview

Alex Aiken

Semantics Overview

- We must specify for every Cool expression what happens when it is evaluated
  - This is the “meaning” of an expression

- The definition of a programming language:
  - The tokens ⇒ lexical analysis
  - The grammar ⇒ syntactic analysis
  - The typing rules ⇒ semantic analysis
  - The evaluation rules

    ⇒ code generation and optimization

Semantics Overview

- We have specified evaluation rules indirectly
  - The compilation of Cool to a stack machine
  - The evaluation rules of the stack machine

- This is a complete description
  - Why isn't it good enough?

Semantics Overview

- Assembly-language descriptions of language implementations have irrelevant detail
  - Whether to use a stack machine or not
  - Which way the stack grows
  - How integers are represented
  - The particular instruction set of the architecture

- We need a complete description
  - But not an overly restrictive specification

Semantics Overview

- Many ways to specify semantics
  - All equally powerful
  - Some more suitable to various tasks than others

- Operational semantics
  - Describes program evaluation via execution rules
    - on an abstract machine
  - Most useful for specifying implementations
  - This is what we use for Cool

Semantics Overview

- Denotational semantics
  - Program's meaning is a mathematical function
- Axiomatic semantics
  - Program behavior described via logical formulae
    - If execution begins in state satisfying X, then it ends in state satisfying Y
    - X, Y formulas
  - Foundation of many program verification systems