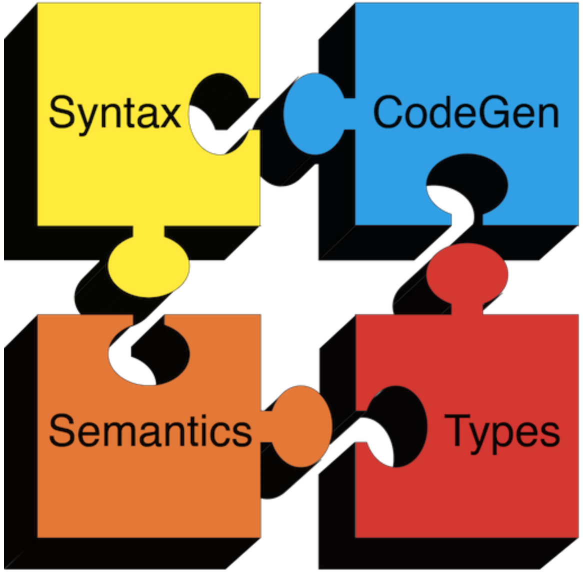

Compilers

Typing Methods

Alex Aiken

Typing Methods

$$\frac{\begin{array}{c} O \vdash e_0 : T_0 \\ O \vdash e_1 : T_1 \\ . \; . \; . \\ O \vdash e_n : T_n \end{array}}{O \vdash e_0.f(e_1, \ldots ,e_n) : ?} \; [\text{Dispatch}]$$

Typing Methods

- In Cool, method and object identifiers live in different name spaces
  - A method foo and an object foo can coexist in the same scope

- In the type rules, this is reflected by a separate mapping M for method signatures

$$M(C,f) = (T_1,\ldots T_n,T_{n+1})$$

means in class C there is a method f

$$f(x_1{:}T_1,\ldots,x_n{:}T_n){:}\ T_{n+1}$$

Typing Methods

$$\frac{\begin{array}{c} O, M \vdash e_0 : T_0 \\ O, M \vdash e_1 : T_1 \\ \cdots \\ O, M \vdash e_n : T_n \\ M(T_0, f) = (T_{1'}, \ldots T_{n'}, T_{n+1}) \\ T_i \leq T_{i'} \text{ for } 1 \leq i \leq n \end{array}}{O, M \vdash e_0.f(e_1, \ldots, e_n) : T_{n+1}} \; \text{[Dispatch]}$$

Typing Methods

$$\begin{array}{c}
O, M \vdash \underline{e_0 : T_0} \\
O, M \vdash e_1 : T_1 \\
\ldots \\
O, M \vdash e_n : T_n \\
\underline{T_0 \leq T} \\
\underline{M(T,f) = (T_1', \ldots T_n', \boxed{T_{n+1}})} \\
\underline{T_i \leq T_i'}, \text{ for } 1 \leq i \leq n \\
\hline
O, M \vdash \underline{e_0}\,\underline{@T}.f(e1, \ldots, e_n) : \boxed{T_{n+1}}
\end{array}
\quad \text{[Static Dispatch]}$$

Typing Methods

Given the class definitions and method declaration at right, which of the following are valid types for the variables in the statement below?

```
z <- x.setCenter(y)
```

- □ x: Rect, y: Object, z: Bool
- □ x: Circle, y: Point, z: Bool
- □ x: Object, y: Object, z: Object
- □ x: Shape, y: Point, z: Bool

```
Class Object
Class Bool inherits Object
Class Point inherits Object
Class Line inherits Object
Class Shape inherits Object {
  setCenter(p: Point): Bool {
    …
  };
  …
};
Class Quad inherits Shape
Class Circle inherits Shape
Class Rect inherits Quad
Class Square inherits Rect
```

Typing Methods

- The method environment must be added to all rules
- In most cases, M is passed down but not actually used
  - Only the dispatch rules use M

$$\frac{O,M \vdash e_1 : \text{Int} \quad O,M \vdash e_2 : \text{Int}}{O,M \vdash e_1 + e_2 : \text{Int}} \text{[Add]}$$

Typing Methods

- For some cases involving SELF_TYPE, we need to know the class in which an expression appears

- The full type environment for COOL:
  - A mapping O giving types to object id's
  - A mapping M giving types to methods
  - The current class C

Typing Methods

The form of a *sentence* in the logic is

$$\underline{O},\underline{M},\underline{C} \vdash \underline{e}: \underline{T}$$

Example:

$$\left[\frac{O,M,C \vdash e_1 : \text{Int} \quad O,M,C \vdash e_2 : \text{Int}}{O,M,C \vdash e_1 + e_2 : \text{Int}}\right. \quad \text{[Add]}$$

Typing Methods

- The rules in this lecture are COOL-specific
  - Some other languages have very different rules

- General themes
  - Type rules are defined on the structure of expressions
  - Types of variables are modeled by an environment

- Warning: Type rules are very compact!