Lecture 13
Scoping & Binding

Denoted values

- **Variables**
  - References: \[(f x y)\]
  - Declarations:
    - \[\text{lambda } (x)\ (x + 3)\]
    - \[\text{let } (x = (y + 7))\ (x + 3)\]

- **Semantics**
  - Binding
  - Scope
Denoted values

- Variables
  - References
    - \( f \times y \)
  - Declarations
    - \( \lambda x (x + 3) \)
    - \( \text{let } (x + y) (x + 3) \)

- Semantics
  - Binding
  - Scope

we need rules to define scoping
Scoping

- **Static scoping**
  - Declarations and references can be matched without code execution
  - Search “outward”

- **Dynamic scoping**
  - Declarations and references are matched during code execution

```plaintext
(let ((x 3) (y 4))
  (+ (let ((x (+ y 5)))
      (* x y))
  x)
```

Calling this `x1`, `x2`, and `x3`.

Concepts

- **Shadowing**
- **Holes**
- **Extent**
  - Duration of the binding
- **Contour diagram**
  - Helps resolving bindings
- **Lexical depth**
Another example

How are the binding rules defined?

How are the binding rules defined?

\[
\begin{align*}
\text{(lambda } (x \ y) & ) \\
\text{(let } ((z (+ x y))) & ) \\
\text{(lambda } (x \ z) & ) \\
\text{(let } ((x \ (let \ ((x \ (let \ ((y \ 11)) \ (z (+ x y z)) \]))) \ (z (+ x y z)))) & ) \\
\end{align*}
\]