Given a relation $e \Rightarrow v$

Deterministic : $\forall e.\ e \Rightarrow v_1$ and $e \Rightarrow v_2$ only if $v_1 = v_2$ — at most one v for each e

Normalizing : $\forall e.\ \exists v.\ e \Rightarrow v$. — at least one v for each e

Fb

Example Fn Val:

Syntax

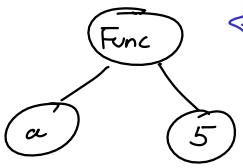
$v ::= 0 \mid 1 \mid -1 \mid 2 \mid -2 \mid \ldots \mid \text{True} \mid \text{False} \mid \text{Function } x \to e$

$e ::= v \mid e + e \mid e - e \mid e \text{ And } e \mid e \text{ Or } e \mid \text{Not } e \mid \text{If } e \text{ Then } e \text{ Else } e \mid e = e \mid e\ e \mid (e)$

$x ::=$ (all variable names)

Concrete Syntax → Function a → 5

Abstract Syntax



AST ↓↓ → abstract syntax tree

Operational Semantics

Let $e \Rightarrow v$ be a relation defined by the rules

Value $\dfrac{}{v \Rightarrow v}$ And $\dfrac{e_1 \Rightarrow v_1 \quad e_2 \Rightarrow v_2 \quad v \text{ is the logical conjuction of } v_1 \text{ and } v_2}{e_1 \text{ And } e_2 \Rightarrow v}$

Or .... Not .... Plus $\dfrac{e_1 \Rightarrow v_1 \quad e_2 \Rightarrow v_2 \quad v \text{ is the arithmetic sum of } v_1 \text{ and } v_2}{e_1 + e_2 \Rightarrow v}$

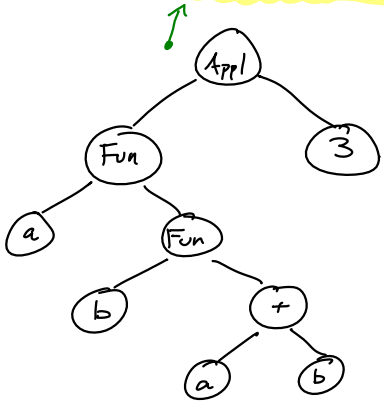
Minus .... Equals ....

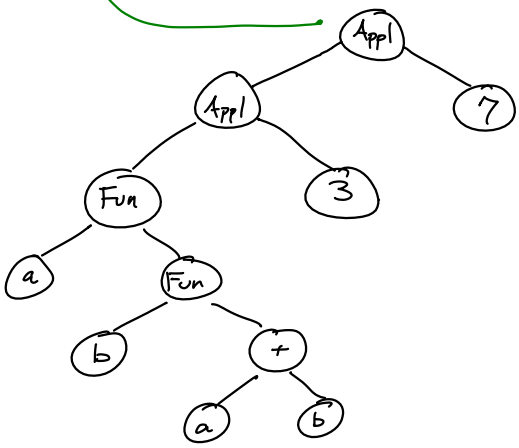
Functions' Semantics

(Function a → a) 5 ⟹ 5

((Function a → Function b → a + b) 3) 7 ⟹ 10

(Function a → Function b → a + b) 3 ⟹ Function b → 3 + b





(Function c → Function c → c) 5 ⟹ Function c → 5
Function c → c

```
int x = 0;
if (b) {
  int x = 1;
  printf("%d", x);
}
```

The inner variable "shadows" the outer variable.

"Substitution" of form

$e[v/x] = e'$