The Goal

- Many problems in programming languages can be formulated as the solution to a set of mutually recursive equations.

The Goal

- Many problems in programming languages can be formulated as the solution to a set of mutually recursive equations.
- Given a set D and functions $f, g: D \times D \rightarrow D$, solve

$$x = f(x, y)$$
$$y = g(x, y)$$

The Goal

- Many problems in programming languages can be formulated as the solution to a set of mutually recursive equations.
- Given a set D and functions $f, g: D \times D \to D$, solve

$$x = f(x, y)$$
$$y = g(x, y)$$

- Underlying questions
  - What assumptions on D, f, and g are sufficient to ensure that such a system of equations has a solution?
  - If such a system has multiple solutions, which solution do we really want?
  - How do we compute that solution?

Example 1: LL(1) Parsing Table

- The computation of $NULLABLE$, $FIRST$, and $FOLLOW$ can be formulated in terms of solving such a system of equations.
  - $FIRST(\varepsilon) = \{\varepsilon\}$.
  - $\forall t \in T, FIRST(t) = \{t\}$.
  - $FIRST(Y_1 \cdots Y_k) = FIRST(Y_1) +_1 \cdots +_1 FIRST(Y_k)$.
  - $(A \to Y_1 \cdots Y_k \in P) \wedge \big(f \in FIRST(Y_1 \cdots Y_k)\big) \Longrightarrow f \in FIRST(A)$.
  - $\$ \in FOLLOW(S)$.
  - $(A \to X_1 \cdots X_k B Y_1 \cdots Y_m \in P) \wedge \big(t \in FIRST(Y_1 \cdots Y_m) +_1 FOLLOW(A)\big) \Longrightarrow t \in FOLLOW(B)$.

Example 1: LL(1) Parsing Table

S→ A$

A → BC | x

B → t | ε

C → v | ε

- The computation of $NULLABLE$, $FIRST$, and $FOLLOW$ can be formulated in terms of solving such a system of equations.
  - $FIRST(\varepsilon) = \{\varepsilon\}$.
  - $\forall t \in T, FIRST(t) = \{t\}$.
  - $FIRST(Y_1 \cdots Y_k) = FIRST(Y_1) +_1 \cdots +_1 FIRST(Y_k)$.
  - $(A \to Y_1 \cdots Y_k \in P) \wedge \left(f \in FIRST(Y_1 \cdots Y_k)\right) \Longrightarrow f \in FIRST(A)$.
  - $\$ \in FOLLOW(S)$.
  - $(A \to X_1 \cdots X_k B Y_1 \cdots Y_m \in P) \wedge \left(t \in FIRST(Y_1 \cdots Y_m) +_1 FOLLOW(A)\right) \Longrightarrow t \in FOLLOW(B)$.

NULLABLE = {A,B,C}
FIRST(A)={x,t,v,ε}
FIRST(B)={t,ε}
FIRST(C)={v,ε}
FIRST(S)={x,t,v,$}
FOLLOW(A)={$}
FOLLOW(B)={v,$}
FOLLOW(C)={$}

FOLLOW(A) = FOLLOW(A) U {$}
FOLLOW(B) = FOLLOW(B) U {v}
FOLLOW(B) = FOLLOW(B) U FOLLOW(A)
FOLLOW(C) = FOLLOW(C) U FOLLOW(A)

FOLLOW(A) = FOLLOW(A) U {$}
FOLLOW(B) = FOLLOW(B) U {v} U FOLLOW(A)
FOLLOW(C) = FOLLOW(C) U FOLLOW(A)

Example 2: Live Variables

- The computation of the In and Out sets for basic blocks can be formulated in terms of solving such a system of equations.
- Given the CFG, assemble the full system of equations.
  - Compute use and def sets for each instruction I and each basic block B.
  - For each instruction I:
    $$In(I) = \big(Out(I) \setminus \mathrm{def}(I)\big) \cup \mathrm{use}(I).$$
  - For each basic block B:
    $$Out(B) = \bigcup_{B' \in \mathrm{succ}(B)} In(B').$$

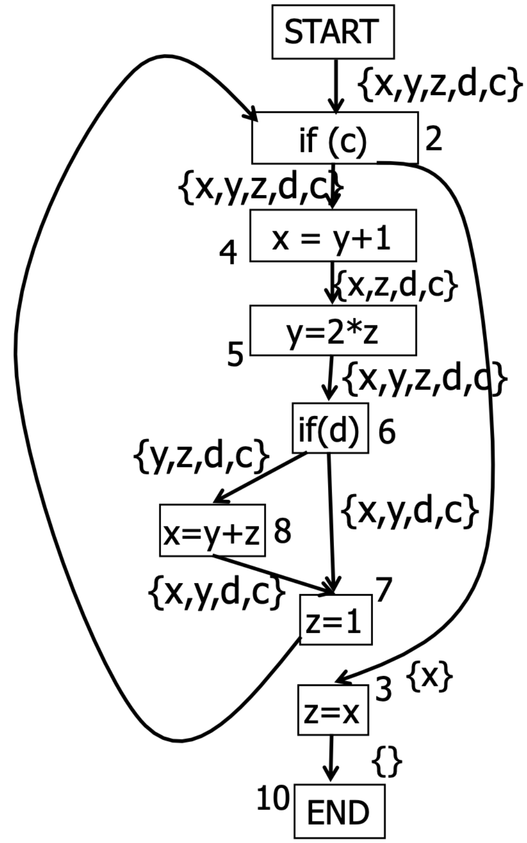
Example 2: Live Variables

- The computation of the In and Out sets for basic blocks can be formulated in terms of solving such a system of equations.
- Given the CFG, assemble the full system of equations.
  - Compute use and def sets for each instruction I and each basic block B.
  - For each instruction I:

    $$In(I) = \left(Out(I) \setminus \mathrm{def}(I)\right) \cup \mathrm{use}(I).$$

  - For each basic block B:

    $$Out(B) = \cup_{B' \in \mathrm{succ}(B)} In(B').$$



L10 = { }

L3 = {x} U (L10- {z})

L2 = L4 U L3 U {c}

L7 = L2 – {z}

L8 = {y,z} U (L7 – {x})

L6 = L8 U L7 U {d}

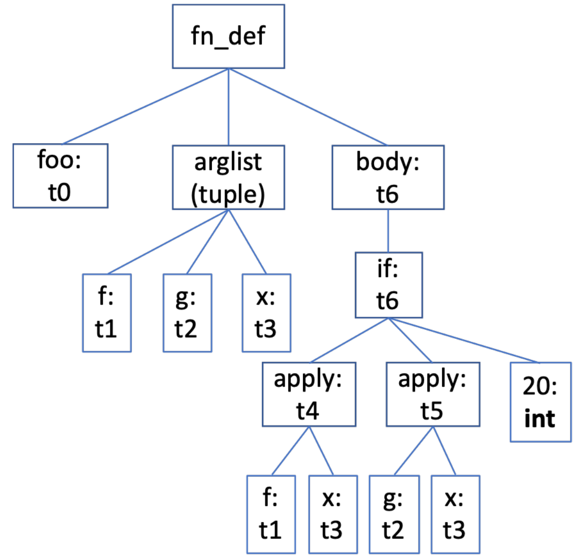
L5 = {z} U (L6 – {y})

L4 = {y} U (L5 – {x})

Example 3: Type Inference

- The computation of *polytypes* can be formulated in terms of solving such a system of equations.
- Form a system of equations among unknown *type variables,* following the language's typing rules "in reverse".
  - For the judgment $\frac{E \vdash f : T \rightarrow U, E \vdash e : T}{E \vdash f(e) : U}$, if $f, e, \mathbf{apply}$ have been assigned symbolic type names t_1, t_2, t_3, then add the equation $t_1 = t_2 \rightarrow t_3$.
  - For the judgment $\frac{E \vdash e_0 : \mathbf{bool}, E \vdash e_1 : T, E \vdash e_2 : T}{E \vdash \mathbf{if}\ e_0\ \mathbf{then}\ e_1\ \mathbf{else}\ e_2}$, if $e_0, e_1, e_2, \mathbf{if}$ have been assigned symbolic type names t_0, t_1, t_2, t_3, then add the equations $\{t_0 = \mathbf{bool}, t_1 = t_2, t_3 = t_1\}$.

Example 3: Type Inference



- The computation of *polytypes* can be formulated in terms of solving such a system of equations.
- Form a system of equations among unknown *type variables,* following the language's typing rules "in reverse".
  - For the judgment $\frac{E\,\vdash f : T \to U, E \vdash e : T}{E \vdash f(e) : U}$, if $f, e, \mathbf{apply}$ have been assigned symbolic type names t_1, t_2, t_3, then add the equation $t_1 = t_2 \to t_3$.
  - For the judgment $\frac{E\,\vdash e_0 : \mathbf{bool}, E \vdash e_1 : T, E \vdash e_2 : T}{E \vdash \mathbf{if}\ e_0\ \mathbf{then}\ e_1\ \mathbf{else}\ e_2}$, if $e_0, e_1, e_2, \mathbf{if}$ have been assigned symbolic type names t_0, t_1, t_2, t_3, then add the equations $\{t_0 = \mathbf{bool}, t_1 = t_2, t_3 = t_1\}$.

t0 = t1 * t2 * t3 -> t6

t4 = **bool**
t5 = **int**
t6 = t5

t1 = t3 -> t4

t2 = t3 -> t5

t0 = ((t3 -> **bool**) * (t3 -> **int**) * t3) -> **int**

Game Plan

1. Finite partially-ordered set D with least element $\perp$
   Function $f: D \to D$
   Monotonic function $f: D \to D$

2. Fixpoints of monotonic function $f: D \to D$
   - Least fixpoint

3. Solving equation $x = f(x)$
   - Least solution is least fixpoint of f

4. Generalization to when D has a greatest element $\top$
   - Least and greatest solutions to equation $x = f(x)$
   - Generalization to systems of equations

5. Semi-lattices and lattices