# Iterative Solving Framework

## Recap

Every dataflow analysis is a five-tuple $(L, D, F_n, \sqcap, \top)$

Iterative solving (forward analysis)

1. Initialize $out[n] \coloneqq \top$, for each CFG node $n$
2. Repeat until convergence:
   For each CFG node $n$,
   $$out[n] \coloneqq F_n\left(\prod_{n'<n} out[n']\right)$$

## A more abstract way to think about it

$N$ nodes in CFG
Start from $(\top, \top, \ldots, \top) \in L^N$
Result is $(\ell_{n_1}, \ell_{n_2}, \ldots \ell_{n_N}) \in L^N$
Global transfer function $F : L^N \to L^N$

$$F(\ell_{n_1}, \ell_{n_2}, \ldots \ell_{n_N}) \coloneqq \left(F_{n_1}\left(\prod_{n'<n_1} \ell_{n'}\right), F_{n_2}\left(\prod_{n'<n_2} \ell_{n'}\right), \ldots, F_{n_N}\left(\prod_{n'<n_N} \ell_{n'}\right)\right)$$

Iterative solving

1. Initialize $X \coloneqq (\top, \top, \ldots, \top) \in L^N$
2. Repeat until convergence:
   $X \coloneqq F(X)$

$X_0 = (\top, \cdots, \top)$
$X_{i+1} = F(X_i)$
Convergence in the k-th iteration if
$X_{k+1} = F(X_k) = X_k$

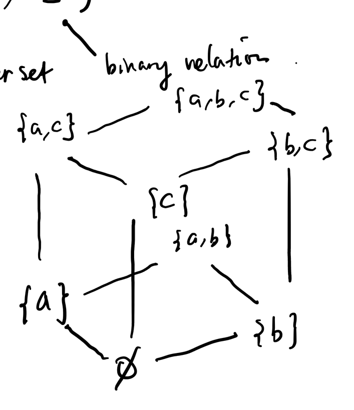
Condition 1. L is a partial order. (poset)

Reflexive $\ell \sqsubseteq \ell$
Transitive $\ell_1 \sqsubseteq \ell_2 \wedge \ell_2 \sqsubseteq \ell_3 \Rightarrow \ell_1 \sqsubseteq \ell_3$
Anti-symmetric $\ell_1 \sqsubseteq \ell_2 \wedge \ell_2 \sqsubseteq \ell_1 \Rightarrow \ell_1 = \ell_2$

$\ell_1 \sqsubseteq \ell_2$ : $\ell_2$ at least as informative as $\ell_1$

Example. $(2^S, \subseteq)$ is a partial order.
carrier set
binary relation

Hasse diagram




$S = \{a, b, c\}$
height = 3

If $(L, \sqsubseteq)$ is a partial order, then $(L, \sqsupseteq)$ is a partial order

Condition 1. L is a lower-semi lattice
= p.o.set + meet operator GLB

GLB. $\ell_1 \sqcap \ell_2 \sqsubseteq \ell_1 \wedge \ell_1 \sqcap \ell_2 \sqsubseteq \ell_2$
$\forall \ell_3,\ \ell_3 \sqsubseteq \ell_1 \wedge \ell_3 \sqsubseteq \ell_2 \Rightarrow \ell_3 \sqsubseteq \ell_1 \sqcap \ell_2$

LVA. $\sqcap = \cup$ RSA $\sqcap = \vee$ ACA $\sqcap = \cap$

Condition 1. L is a lower-semi lattice w/ a top element.

$X_0 = (\top, \cdots, \top)$
$X_{i+1} = F(X_i) = F(F(X_{i-1})) = F^2(X_{i-1}) = \cdots = F^{i+1}(X_0)$

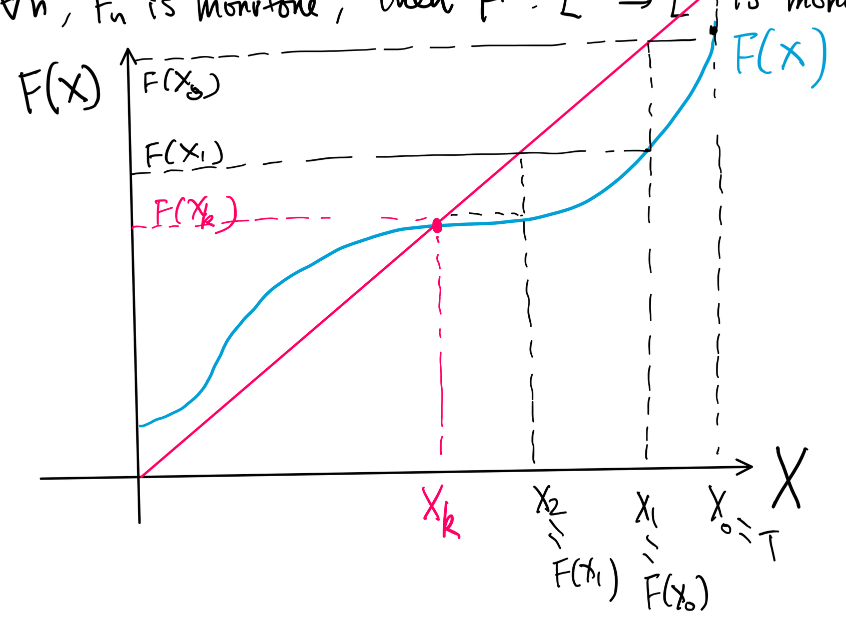
$L =$ boolean lattice. $F = \neg$
$X_0 = true$ $X_1 = F(X_0) = false$ $X_2 = F(X_1) = true$ ~

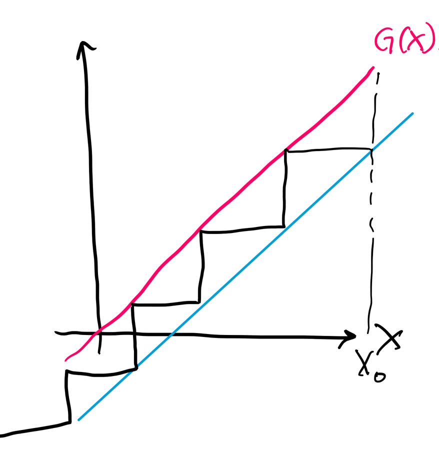
Condition 2. F is monotone
$\forall \ell_1 \sqsubseteq \ell_2,\ F(\ell_1) \sqsubseteq F(\ell_2)$

LVA. $F_n(\ell) = use[n] \cup (\ell \setminus def[n])$

$G(X) \coloneqq X$

If $\forall n$, $F_n$ is monotone, then $F: L^N \to L^N$ is monotone.




Condition 3 L has finite height. ⇒ every descending chain has a minimum.

$\ell_0 \sqsupset \ell_1 \sqsupset \ell_2 \sqsupset \cdots$

L has height $h$ ⇒ $L^N$ has height $N \cdot h$

Termination. $X_0 = (\top, \cdots, \top)$ $X_{i+1} = F(X_i)$
$X_1 \sqsubseteq X_0$
$X_2 = F(X_1) \sqsubseteq F(X_0) = X_1$
$X_3 = F(X_2) \sqsubseteq F(X_1) = X_2$
$\vdots$
$X_{i+1} \sqsubseteq X_i$

$X_0 \sqsupseteq X_1 \sqsupseteq X_2 \cdots \sqsupseteq X_i \sqsupseteq X_{i+1} \sqsupseteq \cdots$

$\exists k \le Nh$ $X_0 \sqsupset X_1 \sqsupset X_2 \cdots \sqsupset X_k = X_{k+1} = X_{k+2} = \cdots$

$F^{Nh}(\top)$

Quality of solution Want: $F^{Nh}(\top)$ is the greatest fixed point

If $X = F(X)$, then $X \sqsubseteq F^{Nh}(\top)$

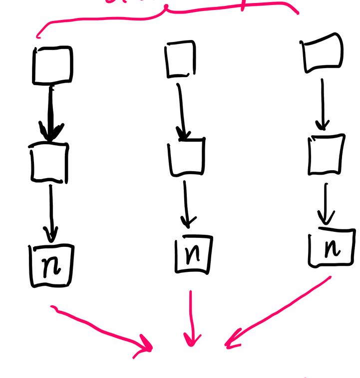
Proof.
$X \sqsubseteq \top$
$X = F(X) \sqsubseteq F(\top)$
$X = F(X) \sqsubseteq F(F(\top)) = F^2(\top)$
$\vdots$
$X = F(X) \sqsubseteq F^{Nh}(\top)$

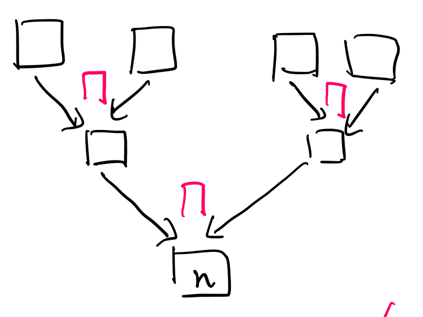
---

Goal. approximate run-time behaviors at compile time.

Question. Conditions under which dataflow analysis loses no info?

execution paths




$\prod_{\cdots n'' n' n \text{ is a path}} F_n(F_{n'}(F_{n''} \cdots (\top) \cdots))$

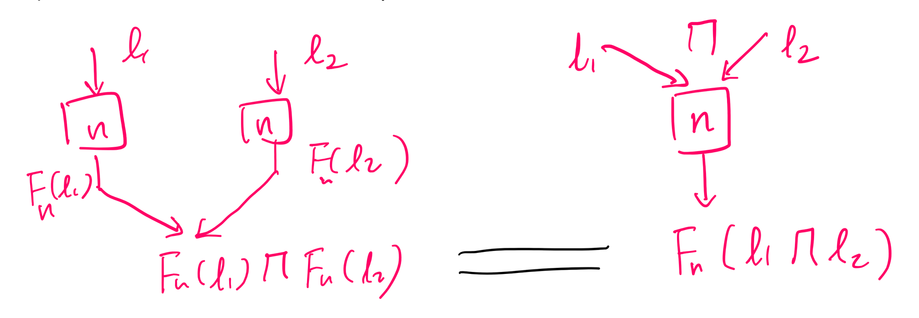
ideal solution
MOP solution

$out[n] = F_n\left(\prod_{n' < n} F_{n'}\left(\prod_{n'' < n'} \cdots F_1(\top)\right)\right)$

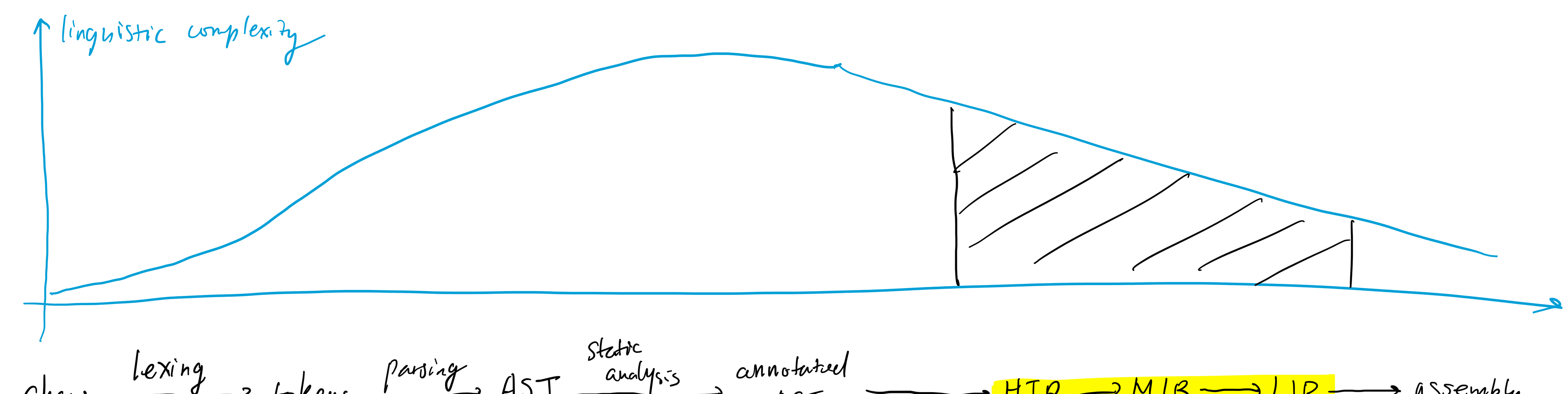
dataflow solution.

$F_n$ are distributive. ⇒ two solutions are equal.

$F_n(\ell_1) \sqcap F_n(\ell_2) = F_n(\ell_1 \sqcap \ell_2)$




---




Chars $\xrightarrow{\text{lexing}}$ tokens $\xrightarrow{\text{parsing}}$ AST $\xrightarrow{\text{static analysis}}$ annotated ASTs $\longrightarrow$ HIR $\longrightarrow$ MIR $\longrightarrow$ LIR $\longrightarrow$ assembly

sequences trees sequences

### Why IRs? Or, IR Design Goals

- Simplicity
- Language independence
- Machine independence
- Support optimization + code generation

Popular IRs JVM LLVM RTL Wasm

Approaches

1. Stack-based IR (JVM, Wasm)
2. quadruples, aka 3-address code. (LLVM)
   $x_1 \leftarrow x_2 \text{ op } x_3$
3. A-normal form ≈ quadruples + higher-order control
   CPS
4. This course: low-level AST w/ explicit mem operation.