Lecture 7: Data-Flow Analysis: Soundness and Precision

17-355/17-665/17-819: Program Analysis

Rohan Padhye

September 18, 2025

* Course materials developed with Jonathan Aldrich and Claire Le Goues

WHILE3ADDDR Semantics

$$\frac{P(n) = x := m}{P \vdash \langle E, n \rangle \leadsto \langle E[x \mapsto m], n+1 \rangle} \; \textit{step-const}$$

$$\frac{P[n] = x := y}{P \vdash \langle E, n \rangle \leadsto \langle E[x \mapsto E(y)], n+1 \rangle} \; \textit{step-copy}$$

$$\frac{P(n) = x := y \; op \; z \quad E(y) \; \mathbf{op} \; E(z) = m}{P \vdash \langle E, n \rangle \leadsto \langle E[x \mapsto m], n+1 \rangle} \; \textit{step-arith}$$

$$\frac{P(n) = \text{goto } m}{P \vdash \langle E, n \rangle \leadsto \langle E, m \rangle} \; \textit{step-goto}$$

$$\frac{P(n) = \text{if } x \; op_r \; 0 \text{ goto } m \quad E(x) \; \mathbf{op_r} \; 0 = true}{P \vdash \langle E, n \rangle \leadsto \langle E, m \rangle} \; \textit{step-iftrue}$$

$$\frac{P(n) = \text{if } x \; op_r \; 0 \text{ goto } m \quad E(x) \; \mathbf{op_r} \; 0 = false}{P \vdash \langle E, n \rangle \leadsto \langle E, n+1 \rangle} \; \textit{step-iffalse}$$

Program Trace

A trace T of a program P is a potentially infinite sequence $\{c_0, c_1, ...\}$ of program configurations, where $c_0 = E_0, 1$ is called the initial configuration, and for every $i \geqslant 0$ we have $P \vdash c_i \rightsquigarrow c_{i+1}$

Program Traces

```
1: if x = 0 goto 4
2: y := 0
3: goto 5
4: y := 1
5: x := x + y
```

Exercise: Write two program traces starting from $c_0 = \langle\{x \mapsto 0, y \mapsto 1\}, 1\rangle$ and $c_0 = \langle\{x \mapsto 1, y \mapsto 0\}, 1\rangle$

Soundness

The result $\langle \sigma_n \mid n \in P \rangle$ of a program analysis running on program P is sound iff, for all traces T of P, for all i such that $0 \leqslant i < \mathit{length}(T)$, $\alpha(c_i) \sqsubseteq \sigma_{n_i}$

Local Soundness

A flow function f is *locally sound* iff $P \vdash c_i \rightsquigarrow c_{i+1}$ and $\alpha(c_i) \sqsubseteq \sigma_{n_i}$ and $f[\![P[n_i]]\!](\sigma_{n_i}) = \sigma_{n_{i+1}}$ implies $\alpha(c_{i+1}) \sqsubseteq \sigma_{n_{i+1}}$

Fixed Point

Let $\mathcal{F}(\langle \sigma_1, \sigma_2, ...\sigma_{|P|} \rangle) = \langle \sigma'_1, \sigma'_2, ...\sigma'_{|P|} \rangle$ be a composite flow function for program P, such that:

$$\begin{aligned} \forall 1 < i \leqslant |\mathrm{P}| : \sigma'_i &= \left(\bigsqcup_{j \in \mathtt{preds}(i)} f[\![P[j]]\!](\sigma_j) \right) \\ \sigma'_1 &= \left(\bigsqcup_{j \in \mathtt{preds}(1)} f[\![P[j]]\!](\sigma_j) \right) \sqcup \sigma_0 \end{aligned}$$

where σ_0 is the initial dataflow information. Then, a dataflow analysis result $\Sigma = \langle \sigma_1, \sigma_2, ...\sigma_{|P|} \rangle$ is a *fixed point* iff $\mathcal{F}(\Sigma) = \Sigma$.

Worklist Algorithm [Kam & Ullman'76]

```
worklist = $\varnothing$
for Node n in cfg
    input[n] = output[n] = $\bot$
    add n to worklist
output[programStart] = initialDataflowInformation

while worklist is not empty
    take a Node n off the worklist
    input[n] = $\sqcup_{k \in preds(n)}$ output[k]
    newOutput = flow(n, input[n])
    if newOutput $\neq$ output[n]
        output[n] = newOutput
        for Node j in succs(n)
            add j to worklist
```

Worklist Algorithm Terminates at Fixed Point

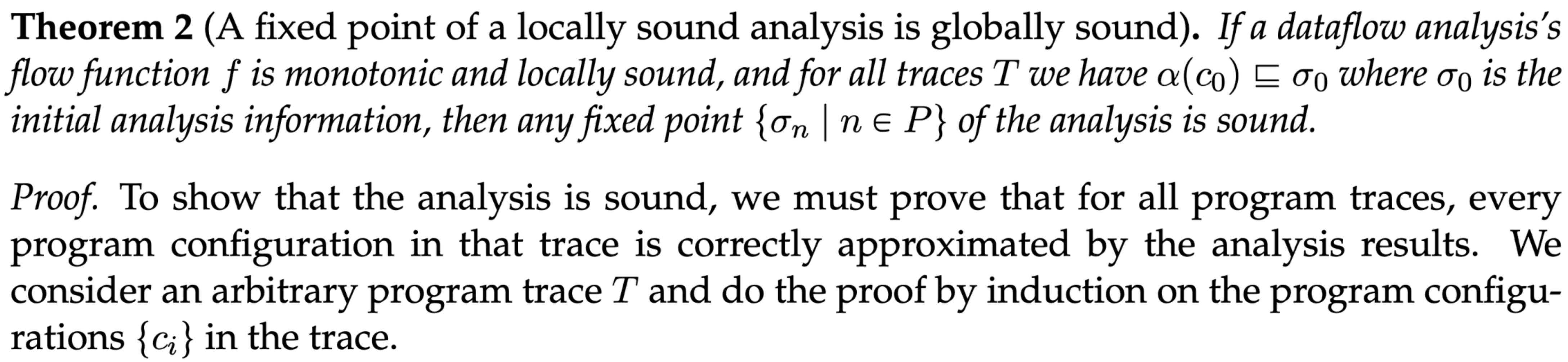
At the fixed point, we therefore have the following equations satisfied:

$$\begin{aligned} \sigma_0 &\sqsubseteq \sigma_1 \\ \forall i \in P : \left(\bigsqcup_{j \in \mathtt{preds}(i)} f[\![P[j]]\!](\sigma_j) \right) &\sqsubseteq \sigma_i \end{aligned}$$

The worklist algorithm shown above computes a fixed point when it terminates. We can prove this by showing that the following loop invariant is maintained:

$$\forall i \;.\; (\exists j \in \mathit{preds}(i) \text{ } \mathit{such\ that}\ f[\![P[j]]\!](\sigma_j) \not\sqsubseteq \sigma_i) \Rightarrow i \in \mathtt{worklist}$$

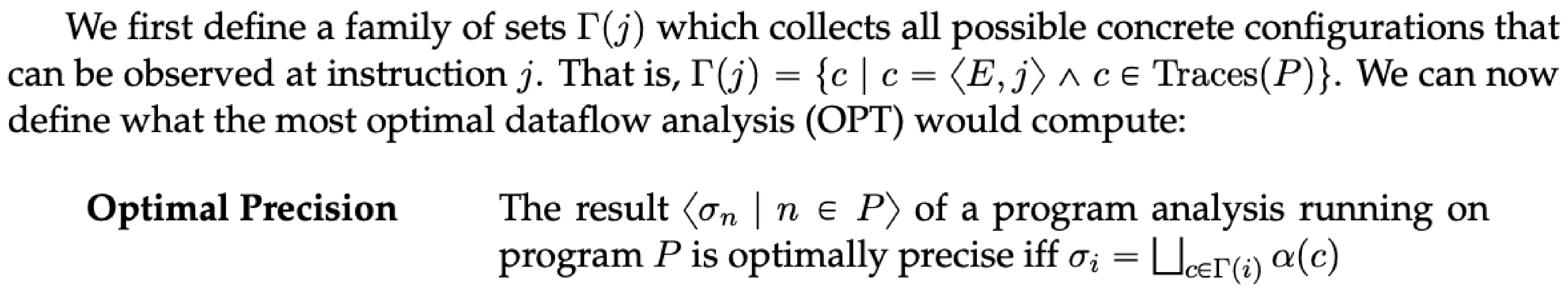
Fixed Point Theorem



Theorem 2 (A fixed point of a locally sound analysis is globally sound). *If a dataflow analysis's flow function f is monotonic and locally sound, and for all traces T we have $\alpha(c_0) \sqsubseteq \sigma_0$ where σ_0 is the initial analysis information, then any fixed point $\{\sigma_n \mid n \in P\}$ of the analysis is sound.*

Proof. To show that the analysis is sound, we must prove that for all program traces, every program configuration in that trace is correctly approximated by the analysis results. We consider an arbitrary program trace T and do the proof by induction on the program configurations $\{c_i\}$ in the trace.

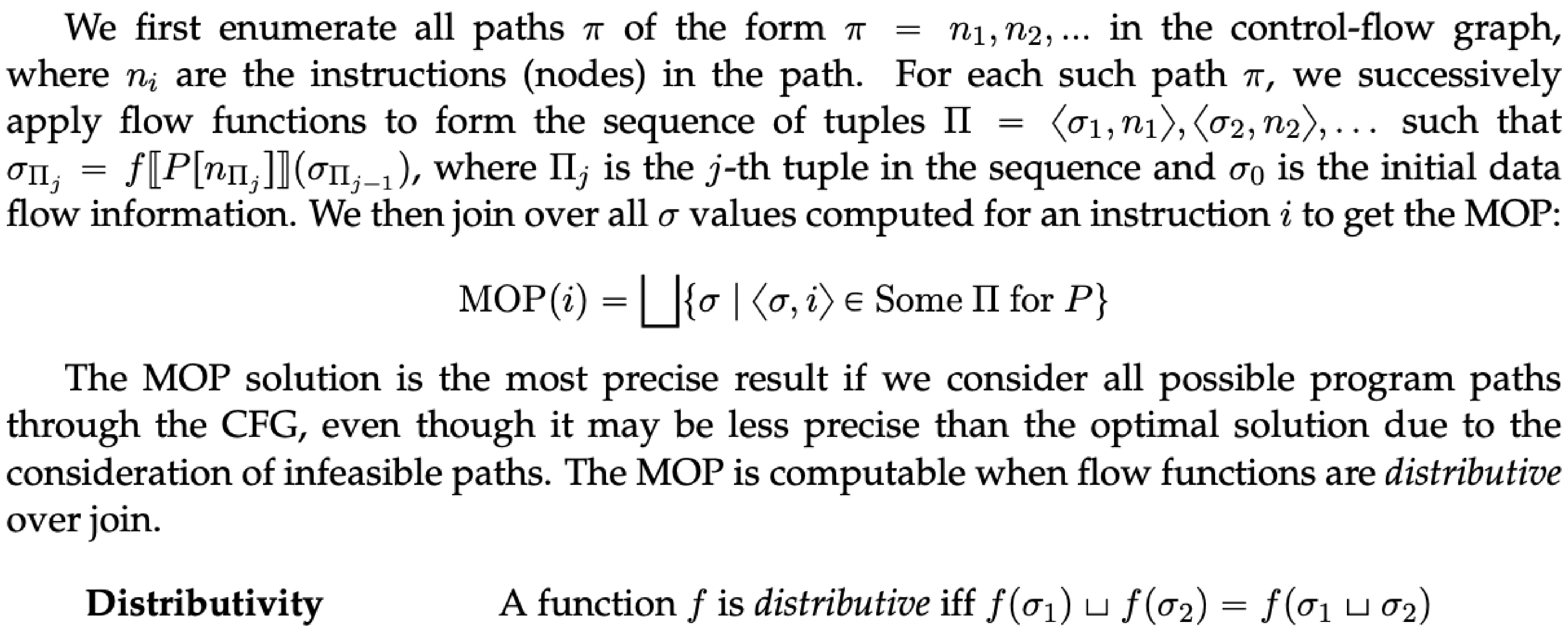
Optimal Precision (OPT)



We first define a family of sets $\Gamma(j)$ which collects all possible concrete configurations that can be observed at instruction j. That is, $\Gamma(j) = \{c \mid c = \langle E, j\rangle \wedge c \in \text{Traces}(P)\}$. We can now define what the most optimal dataflow analysis (OPT) would compute:

Optimal Precision The result $\langle \sigma_n \mid n \in P\rangle$ of a program analysis running on program P is optimally precise iff $\sigma_i = \bigsqcup_{c \in \Gamma(i)} \alpha(c)$

Merge Over Paths (MOP)



We first enumerate all paths π of the form $\pi = n_1, n_2, \ldots$ in the control-flow graph, where n_i are the instructions (nodes) in the path. For each such path π, we successively apply flow functions to form the sequence of tuples $\Pi = \langle\sigma_1, n_1\rangle, \langle\sigma_2, n_2\rangle, \ldots$ such that $\sigma_{\Pi_j} = f[\![P[n_{\Pi_j}]]\!](\sigma_{\Pi_{j-1}})$, where Π_j is the j-th tuple in the sequence and σ_0 is the initial data flow information. We then join over all σ values computed for an instruction i to get the MOP:

$$\text{MOP}(i) = \bigsqcup\{\sigma \mid \langle\sigma, i\rangle \in \text{Some } \Pi \text{ for } P\}$$

The MOP solution is the most precise result if we consider all possible program paths through the CFG, even though it may be less precise than the optimal solution due to the consideration of infeasible paths. The MOP is computable when flow functions are *distributive* over join.

Distributivity A function f is *distributive* iff $f(\sigma_1) \sqcup f(\sigma_2) = f(\sigma_1 \sqcup \sigma_2)$

Least Fixed Point (LFP)

The least fixed point solution of a composite flow function $\mathcal{F}$ is the fixed-point result Σ^* such that $\mathcal{F}(\Sigma^*) = \Sigma^*$ and $\forall \Sigma : (\mathcal{F}(\Sigma) = \Sigma) \Rightarrow (\Sigma^* \sqsubseteq \Sigma)$.

Relative Precision

$$\mathrm{OPT} \sqsubseteq \mathrm{MOP} \sqsubseteq \mathrm{LFP} \sqsubseteq \mathrm{FP} \sqsubseteq \top$$