Background 4: Interprocedural Analysis

So far, we can analyze each procedure separately.

$\rightarrow$ The costs are moderate :-)

$\rightarrow$ The methods also work in presence of separate compilation :-)

$\rightarrow$ At procedure calls, we must assume the worst case :-(

$\rightarrow$ Constant propagation only works for local constants :-((

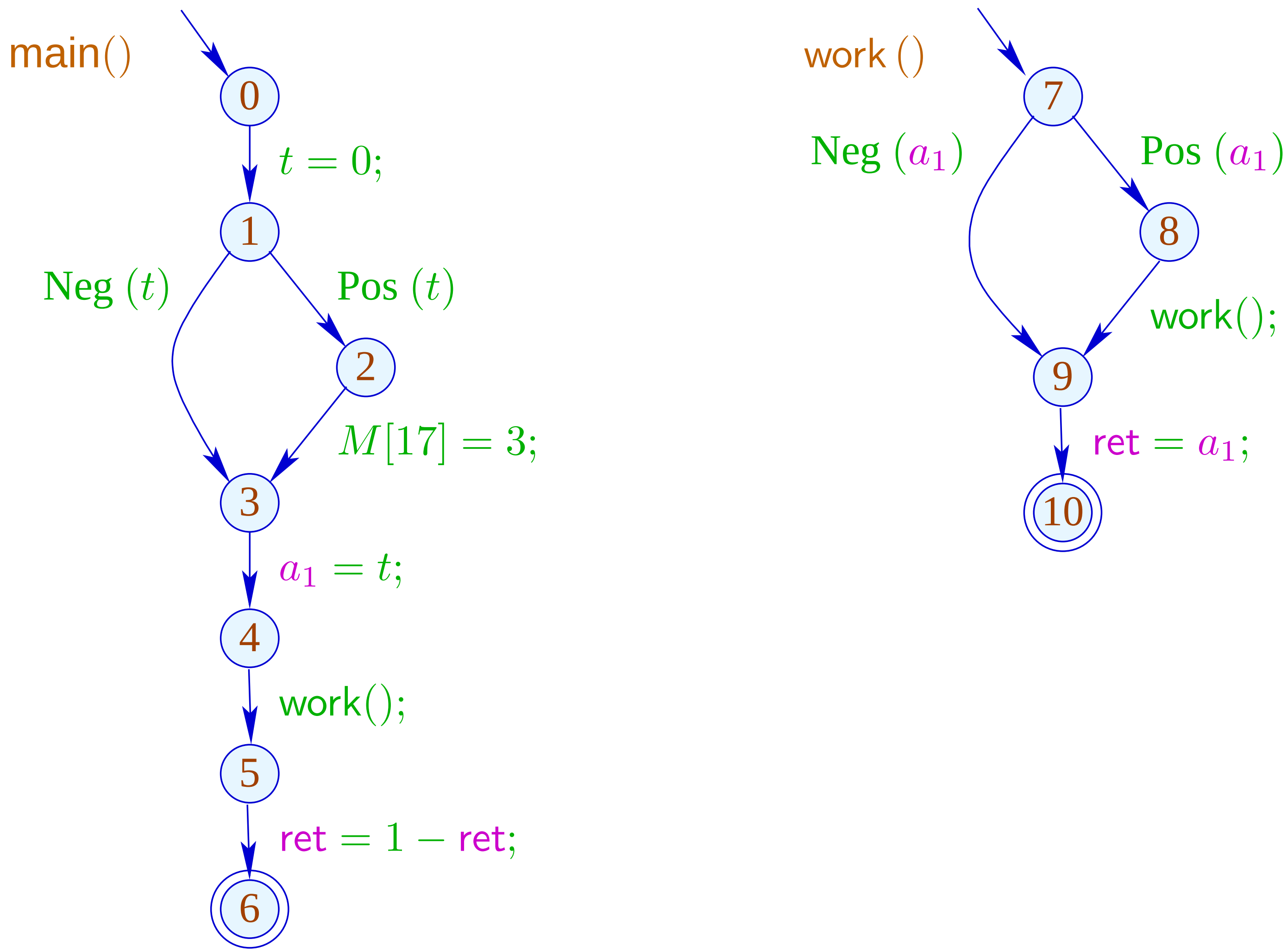
Question:

How can recursive programs be analyzed ???

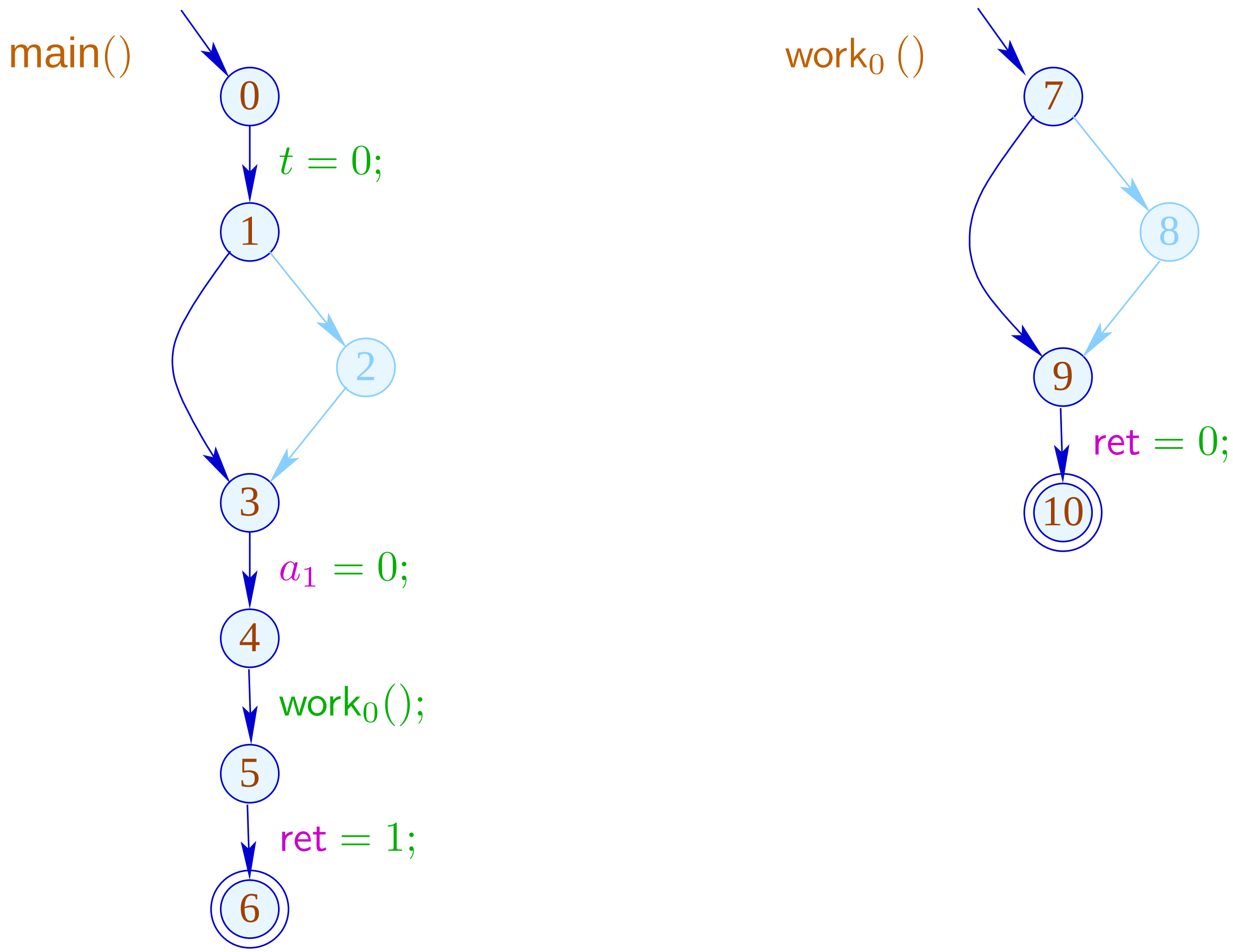
Example: Constant Propagation

$$
\begin{array}{ll}
\mathsf{main}()\ \{\ \mathbf{int}\ t; & \mathsf{work}()\ \{ \\
\quad t = 0; & \quad \mathsf{if}\ (a_1)\ \mathsf{work}(); \\
\quad \mathsf{if}\ (t)\ M[17] = 3; & \quad \mathsf{ret} = a_1; \\
\quad a_1 = t; & \} \\
\quad \mathsf{work}\,(); & \\
\quad \mathsf{ret} = 1 - \mathsf{ret}; & \\
\} &
\end{array}
$$

Example: Constant Propagation



Example: Constant Propagation



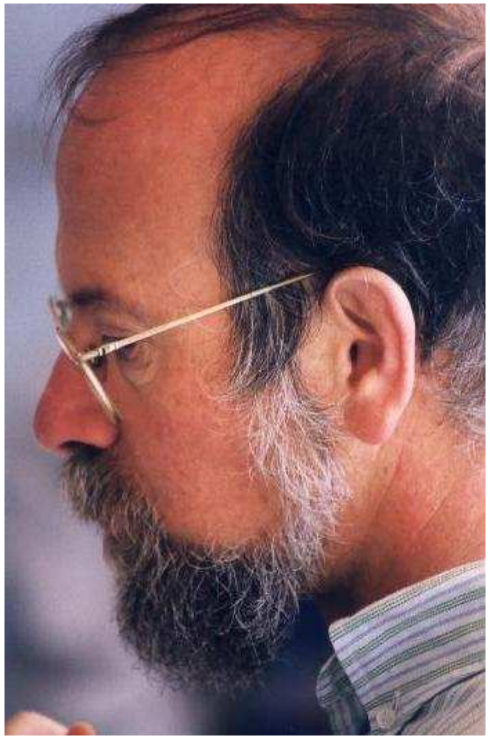
(1) Functional Approach:

Let $\mathbb{D}$ denote a complete lattice of (abstract) states.

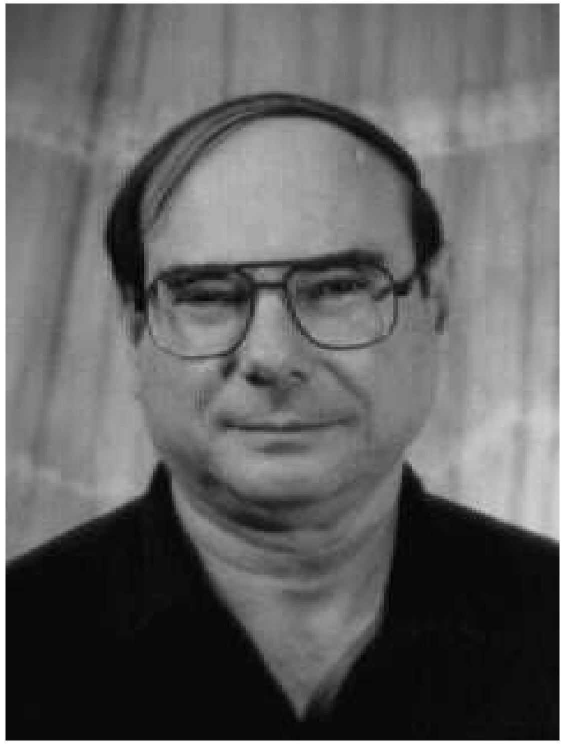
Idea:

Represent the effect of $f()$ by a function:

$$[\![f]\!]^\sharp \; : \; \mathbb{D} \to \mathbb{D}$$



Micha Sharir, Tel Aviv University



Amir Pnueli, Weizmann Institute

In order to determine the effect of a call edge $k = (u, f\,();, v)$ we require abstract functions:

$$\begin{array}{lll} \mathsf{enter}^\sharp & : & \mathbb{D} \to \mathbb{D} \\ \mathsf{combine}^\sharp & : & \mathbb{D}^2 \to \mathbb{D} \end{array}$$

Then we define:

$$[\![k]\!]^\sharp\, D \;=\; \mathsf{combine}^\sharp\, (D, [\![f]\!]^\sharp\, (\mathsf{enter}^\sharp\, D))$$

... for Constant Propagation:

$$\mathbb{D} = (Vars \to \mathbb{Z}^\top)_\perp$$

$$\textsf{enter}^\sharp\; D = \begin{cases} \perp & \text{if} \quad D = \perp \\ D|_{Globals} \oplus \{x \mapsto 0 \mid x \in Locals\} & \text{otherwise} \end{cases}$$

$$\textsf{combine}^\sharp\,(D_1, D_2) = \begin{cases} \perp & \text{if} \quad D_1 = \perp \vee D_2 = \perp \\ D_1|_{Locals} \oplus D_2|_{Globals} & \text{otherwise} \end{cases}$$

The effects $[\![f]\!]^\sharp$ then can be determined by a system of constraints over the complete lattice $\mathbb{D} \to \mathbb{D}$:

$$
\begin{array}{llll}
[\![v]\!]^\sharp & \sqsupseteq & \mathsf{Id} & v \quad \text{entry point} \\
[\![v]\!]^\sharp & \sqsupseteq & [\![k]\!]^\sharp \circ [\![u]\!]^\sharp & k = (u, \_, v) \quad \text{edge} \\
[\![f]\!]^\sharp & \sqsupseteq & [\![stop_f]\!]^\sharp & stop_f \quad \text{end point of} \quad f
\end{array}
$$

$[\![v]\!]^\sharp \;:\; \mathbb{D} \to \mathbb{D}$ describes the effect of all prefixes of computation forests w of a procedure which lead from the entry point to v :-)

Problems:

- How can we represent functions $f : \mathbb{D} \to \mathbb{D}$???
- If $\#\mathbb{D} = \infty$, then $\mathbb{D} \to \mathbb{D}$ has infinite strictly increasing chains :-(

Simplification: Copy-Constants

$\to$ Conditions are interpreted as $;$:-)

$\to$ Only assignments $x = e;$ with $e \in \mathit{Vars} \cup \mathbb{Z}$ are treated exactly :-)

Observation:

$\rightarrow$ The effects of assignments are:

$$[\![x = e;]\!]^\sharp\; D \;=\; \begin{cases} D \oplus \{x \mapsto c\} & \text{if} \quad e = c \in \mathbb{Z} \\ D \oplus \{x \mapsto (D\; y)\} & \text{if} \quad e = y \in \mathit{Vars} \\ D \oplus \{x \mapsto \top\} & \text{otherwise} \end{cases}$$

$\rightarrow$ Let $\mathbb{V}$ denote the (finite !!!) set of constant right-hand sides. Then variables may only take values from $\mathbb{V}^\top$:-))

$\rightarrow$ The occurring effects can be taken from

$$\mathbb{D}_f \to \mathbb{D}_f \qquad \text{with} \qquad \mathbb{D}_f = (\mathit{Vars} \to \mathbb{V}^\top)_\perp$$

$\rightarrow$ The complete lattice is huge, but finite !!!

Improvement:

$\rightarrow$ Not all functions from $\mathbb{D}_f \to \mathbb{D}_f$ will occur :-)

$\rightarrow$ All occurring functions $\lambda D. \perp \; \neq \; M$ are of the form:

$$M \;=\; \{x \mapsto (b_x \sqcup \bigsqcup_{y \in I_x} y) \mid x \in \mathit{Vars}\} \qquad \text{where:}$$
$$M\; D \;=\; \{x \mapsto (b_x \sqcup \bigsqcup_{y \in I_x} D\; y) \mid x \in \mathit{Vars}\} \qquad \text{für} \quad D \neq \perp$$

$\rightarrow$ Let $\mathbb{M}$ denote the set of all these functions. Then for $M_1, M_2 \in \mathbb{M}$ $(M_1 \neq \lambda\, D. \perp \neq M_2)$:

$$(M_1 \sqcup M_2)\; x \;=\; (M_1\; x) \sqcup (M_2\; x)$$

$\rightarrow$ For $k = \#\mathit{Vars}$, $\mathbb{M}$ has height $\mathcal{O}(k^2)$:-)

Improvement (Cont.):

→ Also, composition can be directly implemented:

$$\begin{array}{rcll} (M_1 \circ M_2)\; x & = & b' \sqcup \bigsqcup_{y \in I'} y & \text{with} \\ b' & = & b \sqcup \bigsqcup_{z \in I} b_z & \\ I' & = & \bigcup_{z \in I} I_z & \text{where} \\ M_1\; x & = & b \sqcup \bigsqcup_{y \in I} y & \\ M_2\; z & = & b_z \sqcup \bigsqcup_{y \in I_z} y & \end{array}$$

→ The effects of assignments then are:

$$[\![x = e;]\!]^\sharp \quad = \quad \begin{cases} \mathsf{Id}_{\mathit{Vars}} \oplus \{x \mapsto c\} & \text{if} \quad e = c \in \mathbb{Z} \\ \mathsf{Id}_{\mathit{Vars}} \oplus \{x \mapsto y\} & \text{if} \quad e = y \in \mathit{Vars} \\ \mathsf{Id}_{\mathit{Vars}} \oplus \{x \mapsto \top\} & \text{otherwise} \end{cases}$$

... in the Example:

$$\begin{array}{lll} [\![t = 0;]\!]^\sharp & = & \{a_1 \mapsto a_1, \mathsf{ret} \mapsto \mathsf{ret}, \boxed{t \mapsto 0}\} \\ [\![a_1 = t;]\!]^\sharp & = & \{\boxed{a_1 \mapsto t}, \mathsf{ret} \mapsto \mathsf{ret}, t \mapsto t\} \end{array}$$

In order to implement the analysis, we additionally must construct the effect of a call $k = (\_, f\,();, \_)$ from the effect of a procedure f :

$$\begin{array}{lll} [\![k]\!]^\sharp & = & H\,([\![f]\!]^\sharp) \qquad\qquad \text{where:} \\ H\,(M) & = & \mathsf{Id}|_{Locals} \oplus (M \circ \mathsf{enter}^\sharp)|_{Globals} \\ \mathsf{enter}^\sharp\, x & = & \begin{cases} x & \text{if} \quad x \in Globals \\ 0 & \text{otherwise} \end{cases} \end{array}$$