universität innsbruck


# Program Analysis

Georg Moser

cbr.uibk.ac.at

# Summary of Last Lecture

**Definition (type and effect system)**

a type and effect system is conceivable as a combination of a

- effect system
- annotated type system

in an effect system we have judgements of the form

$$S : \Sigma \xrightarrow{\phi} \Sigma$$

where $\phi$ represents the effect of the execution of $S$; in an annotated type system we have

$$S : \Sigma_1 \to \Sigma_2$$

describing that state properties have been transformed

# Data Flow Analysis

# Intraprocedural Analysis

**Definition (initial and final labels)**

we define mappings init: **Stmt** $\to$ **Lab** and final: **Stmt** $\to \mathcal{P}(\textbf{Lab})$ as follows

$$\begin{aligned} \mathrm{init}([x := a]^\ell) &= \ell & \mathrm{final}([x := a]^\ell) &= \{\ell\} \\ \mathrm{init}([\textbf{skip}]^\ell) &= \ell & \mathrm{final}([\textbf{skip}]^\ell) &= \{\ell\} \\ \mathrm{init}(S_1; S_2) &= \mathrm{init}(S_1) & \mathrm{final}(S_1; S_2) &= \mathrm{final}(S_2) \\ \mathrm{init}(\textbf{if } [b]^\ell \textbf{ then } S_1 \textbf{ else } S_2) &= \ell & \mathrm{final}(\textbf{if } [b]^\ell \textbf{ then } S_1 \textbf{ else } S_2) &= \mathrm{final}(S_1) \cup \mathrm{final}(S_2) \\ \mathrm{init}(\textbf{while})[b]^\ell \textbf{ do } S) &= \ell & \mathrm{final}(\textbf{while})[b]^\ell \textbf{ do } S) &= \{\ell\} \end{aligned}$$

## Definition (blocks)

we define the mapping blocks: $\textbf{Stmt} \to \mathcal{P}(\textbf{Blocks})$ as follows

$$\text{blocks}([x := a]^\ell) = \{[x := a]^\ell\}$$

$$\text{blocks}([\textbf{skip}]^\ell) = \{[\textbf{skip}]^\ell\}$$

$$\text{blocks}(S_1; S_2) = \text{blocks}(S_1) \cup \text{blocks}(S_2)$$

$$\text{blocks}(\textbf{if } [b]^\ell \textbf{ then } S_1 \textbf{ else } S_2) = \{[b]^\ell\} \cup \text{blocks}(S_1) \cup \text{blocks}(S_2)$$

$$\text{blocks}(\textbf{while})[b]^\ell \textbf{ do } S) = \{[b]^\ell\} \cup \text{blocks}(S)$$

and the set of labels in a program

$$\text{labels}(S) = \{\ell \mid [B]^\ell \in \text{blocks}(S)\}$$

## Definition (flows and reverse flows)

we define the function flow: $\textbf{Stmt} \to \mathcal{P}(\textbf{Lab} \times \textbf{Lab})$ as follows

$$\begin{aligned}
\mathsf{flow}([x := a]^\ell) &= \varnothing \\
\mathsf{flow}([\textbf{skip}]^\ell) &= \varnothing \\
\mathsf{flow}(S_1; S_2) &= \mathsf{flow}(S_1) \cup \mathsf{flow}(S_2) \cup \{(\ell, \mathsf{init}(S_2)) \mid \ell \in \mathsf{final}(S_1)\} \\
\mathsf{flow}(\textbf{if } [b]^\ell \textbf{ then } S_1 \textbf{ else } S_2) &= \mathsf{flow}(S_1) \cup \mathsf{flow}(S_2) \cup \{(\ell, \mathsf{init}(S_1))\} \cup \{(\ell, \mathsf{init}(S_2))\} \\
\mathsf{flow}(\textbf{while})[b]^\ell \textbf{ do } S) &= \mathsf{flow}(S) \cup \{(\ell, \mathsf{init}(S))\} \cup \{(\ell', \ell) \mid \ell' \in \mathsf{final}(S)\}
\end{aligned}$$

## Definition (flows and reverse flows)

we define the function flow: $\textbf{Stmt} \to \mathcal{P}(\textbf{Lab} \times \textbf{Lab})$ as follows

$$\mathsf{flow}([x := a]^\ell) = \varnothing$$

$$\mathsf{flow}([\textbf{skip}]^\ell) = \varnothing$$

$$\mathsf{flow}(S_1; S_2) = \mathsf{flow}(S_1) \cup \mathsf{flow}(S_2) \cup \{(\ell, \mathsf{init}(S_2)) \mid \ell \in \mathsf{final}(S_1)\}$$

$$\mathsf{flow}(\textbf{if } [b]^\ell \textbf{ then } S_1 \textbf{ else } S_2) = \mathsf{flow}(S_1) \cup \mathsf{flow}(S_2) \cup \{(\ell, \mathsf{init}(S_1))\} \cup \{(\ell, \mathsf{init}(S_2))\}$$

$$\mathsf{flow}(\textbf{while})[b]^\ell \textbf{ do } S) = \mathsf{flow}(S) \cup \{(\ell, \mathsf{init}(S))\} \cup \{(\ell', \ell) \mid \ell' \in \mathsf{final}(S)\}$$

## Example

$$[z := 1]^1; \textbf{while } [x > 0]^2 \textbf{ do } ([z := z * y]^3; [x := x - 1]^4)$$

## Definition (reverse flows)

we define the function $\mathsf{flow}^{\mathsf{R}}\colon \textbf{Stmt} \to \mathcal{P}(\textbf{Lab} \times \textbf{Lab})$ as follows

$$\mathsf{flow}^{\mathsf{R}}(S) = \{(\ell, \ell') \mid (\ell', \ell) \in \mathsf{flow}(S)\}$$

## Definition (reverse flows)

we define the function $\mathsf{flow}^{\mathsf{R}} \colon \mathbf{Stmt} \to \mathcal{P}(\mathbf{Lab} \times \mathbf{Lab})$ as follows

$$\mathsf{flow}^{\mathsf{R}}(S) = \{(\ell, \ell') \mid (\ell', \ell) \in \mathsf{flow}(S)\}$$

## Convention

- we use $S_*$ to represent the program of interest
- $\mathbf{Lab}_*$, $\mathsf{Var}_*$, $\mathbf{Blocks}_*$ refer to the (finite) labels, variables, blocks in the program of interest
- $\mathbf{AExp}_*$ represents the non-trivial arithemetic subexpressions in $S_*$
- we also write $\mathsf{FV}(a)$ for the set of variables occuring in $a$

# Available Expression Analysis

*For each program point, which expression must have already been computed, and not later modified, on all paths to the program pont*

# Available Expression Analysis

*For each program point, which expression must have already been computed, and not later modified, on all paths to the program pont*

**Example**

$$[x := a + b]^1; [y := a * b]^2; \textbf{while } [y > a + b]^3 \textbf{ do } ([a := a + 1]^4; [x := a + b]^5)$$

the expression $a + b$ is available every time the execution reaches label 3

## Example (continued)

$$\begin{aligned}
\mathrm{kill}_{\mathrm{AE}}([x := a]^\ell) &= \{a' \in \mathbf{AExp}_* \mid x \in \mathrm{FV}(a')\} \\
\mathrm{kill}_{\mathrm{AE}}([\mathbf{skip}]^\ell) &= \varnothing \\
\mathrm{kill}_{\mathrm{AE}}([b]^\ell) &= \varnothing \\[1ex]
\mathrm{gen}_{\mathrm{AE}}([x := a]^\ell) &= \{a' \in \mathbf{AExp}(a) \mid x \notin \mathrm{FV}(a')\} \\
\mathrm{gen}_{\mathrm{AE}}([\mathbf{skip}]^\ell) &= \varnothing \\
\mathrm{gen}_{\mathrm{AE}}([b]^\ell) &= \mathbf{AExp}(b) \\[1ex]
\mathrm{AE}_{\mathrm{entry}}(\ell) &= \begin{cases} \varnothing & \text{if } \ell = \mathrm{init}(S_*) \\ \bigcap\{\mathrm{AE}_{\mathrm{exit}}(\ell') \mid (\ell', \ell) \in \mathrm{flow}(S_*)\} & \text{otherwise} \end{cases} \\
\mathrm{AE}_{\mathrm{exit}}(\ell) &= \Big(\mathrm{AE}_{\mathrm{entry}}(\ell) \setminus \mathrm{kill}_{\mathrm{AE}}(B^\ell)\Big) \cup \mathrm{gen}_{\mathrm{AE}}(B^\ell)
\end{aligned}$$

where $B^\ell \in \mathrm{blocks}(S_*)$