Program Analysis
Lecture 3

Rayna Dimitrova

WS 2016/2017
Tutorials

Tuesday, 15:30, MPI-SWS building (26), room 113
Recap from lecture 2 (While programs)

**arithmetic expressions**, which denote integer values

\[ a ::= k \mid x \mid a_1 + a_2 \mid a_1 - a_2 \mid a_1 \times a_2 \mid a_1 / a_2 \]

\( k \) is an integer constant, \( x \) is a variable

**Boolean expressions**

\[ b ::= a_1 = a_2 \mid a_1 < a_2 \mid \neg b \mid b_1 \land b_2 \mid b_1 \lor b_2 \mid \text{true} \]

**programs**

\[ c ::= [\text{skip}]^l \mid [x := a]^l \mid [\text{read}(x)]^l \mid [\text{print}(a)]^l \mid c_1; c_2 \]

\[ \mid \text{if } [b]^l \text{ then } \{c_1\} \text{ else } \{c_2\} \]

\[ \mid \text{while } [b]^l \text{ do } \{c\} \]

\( x \) is a variable, \( a \) is arithmetic expression, \( b \) is a Boolean expression

Statements are labelled with distinct natural numbers: \([c]^l\) where \( l \in \mathbb{N}_{>0} \). Labelled statements are called **blocks**.
Recap from lecture 2 (CFG)

\[\text{read}(x)\] \[\uparrow\]

\textbf{if} \ [x < 0] \ \{ \\
\hspace{1em} [y := 0]\; \\
\hspace{2em} \textbf{while} \ [x < 0] \ \{ \\
\hspace{3em} [x := x + 1]; \\
\hspace{3em} [y := y + 1] \\
\hspace{2em} \} \\
\hspace{1em} \text{else} \ \\
\hspace{2em} [y := x] \\
\} \\
\] \[\text{print}(y)\] \\

\[\text{read}(x)\] \[\uparrow\] initial

\[\text{read}(x)\] \[\uparrow\] \[x < 0\] \[\uparrow\] [y := 0] \[\uparrow\] [y := x] \[\uparrow\] [x < 0] \[\uparrow\] [x := x + 1] \[\uparrow\] [y := y + 1] \[\uparrow\] [x := x + 1] \[\uparrow\] [y := y + 1] \[\uparrow\] \[\text{print}(y)\] \\

\[\text{read}(x)\] \[\uparrow\] \[\text{print}(y)\] \\

\[\text{read}(x)\] \[\uparrow\] \[\text{print}(y)\] \\

final
Recap from lecture 2 (Monotone framework)

A **dataflow system** is a tuple $S = (G, (D, \leq), i, \{f_b\}_{b \in B})$, where

- $G = (B, E, F)$ is a CFG
- $(D, \leq)$ is a complete lattice that satisfies (ACC)
- $i \in D$ is an initial value for extremal blocks
- $\{f_b\}_{b \in B}$, is a family of monotonic functions $f_b : D \to D$, one for each block in the CFG
Let $S = (G, (D, \leq), i, \{f_b\}_{b \in B})$ be a dataflow system.

We associate with each $b \in B$ a variable $X_b$ with domain $D$.

With $S$ we associate a system of equations that relate the variable $X_b$ for each node $b$ to those of other nodes.

Consider the lattice $(D_S, \leq_S)$, where $D_S = D^{|B|}$ and $(d_1, \ldots, d_{|B|}) \leq_S (d'_1, \ldots, d'_{|B|})$ if and only if $d_b \leq d'_b$ for all $b \in B$.

Define a monotone function $g_S : D_S \rightarrow D_S$ such that a vector $\overline{d} = (d_1, \ldots, d_{|B|})$ is a solution of $S$ if and only if $g_S(\overline{d}) = \overline{d}$. 
Forward versus backward analysis

- **forward analysis** computes information about data that depends on the past of the program execution
- **backward analysis** computes information about data that depends on the future of the program execution

\[
E = \{ \text{initial node} \}
\]

\[
E = \{ \text{final node} \}
\]

For \( b \notin E \)

\[
X_b = \bigsqcup \{ f_p(X_p) \mid p \in \text{pred}(b) \}
\]

\[
X_b = \bigsqcup \{ f_s(X_s) \mid s \in \text{succ}(b) \}
\]
May versus must analysis

- **May analyses** detect properties satisfied by at least one execution path to (or from) the entry (or exit) of a block. The computed information is an overapproximation: it may possibly be true in an actual execution.

- **Must analyses** detect properties satisfied by all paths of execution reaching (or leaving) the entry (or exit) of a block. All values detected by a must analysis are actually reached. The computed information is an underapproximation: it must definitely be true in actual executions.

\[ D = \mathcal{P}(A) \]

For suitably chosen \( A \) with \( \leq \) to be \( \subseteq \)

\( \sqcup \) to be \( \cup \)

\( \bot \) to be \( \emptyset \)

\[ D = \mathcal{P}(A) \]

For suitably chosen \( A \) with \( \leq \) to be \( \supseteq \)

\( \sqcup \) to be \( \cap \)

\( \bot \) to be \( A \)
Transfer functions

The transfer functions definition depends both on the direction (forward or backward) and on the type of the analysis (may or must).

**forward**

\[ f_b(X) := (X \setminus \text{kill}(b)) \cup \text{gen}(b) \]

- **kill(b)**: information invalidated by block \( b \)
- **gen(b)**: information generated by block \( b \)

**backward**

\[ f_b(X) := (X \setminus \text{kill}(b)) \cup \text{gen}(b) \]

- **kill(b)**: information invalidated by block \( b \)
- **gen(b)**: information generated by block \( b \)
A classification of program analyses

<table>
<thead>
<tr>
<th></th>
<th>may</th>
<th>must</th>
</tr>
</thead>
<tbody>
<tr>
<td>forward</td>
<td></td>
<td></td>
</tr>
<tr>
<td>backward</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
A classification of program analyses

<table>
<thead>
<tr>
<th></th>
<th>may</th>
<th>must</th>
</tr>
</thead>
<tbody>
<tr>
<td>forward</td>
<td>reaching definitions</td>
<td>available expressions</td>
</tr>
<tr>
<td>backward</td>
<td>live variables</td>
<td>very busy expressions</td>
</tr>
</tbody>
</table>
Reaching definitions analysis

**Goal:** For each program point, determine which assignments may have been made and not overwritten when this point is reached by the program execution along some execution path.

**Applications:** Establish the correspondence between blocks that produce values and blocks that use them (use-definition chains).

**Classification:** We need a forward analysis that computes information about the executions up to a certain program point. We have to perform a may analysis overapproximating the information from the possible executions. The computed information is guaranteed to include the behaviour of each execution.
Reaching definitions analysis: CFG

\[ \begin{align*}
[x &:= 5]^1; \\
[y &:= 1]^2; \\
\textbf{while } [x > 1]^3 	extbf{ do } \{ \\
& [y := x \times y]^4; \\
& [x := x - 1]^5 \\
\} 
\end{align*} \]

\[ G = (B, E, F), E = \{1\} \]

\[ \begin{array}{c}
\text{initial} \\
\rightarrow [x := 5]^1 \\
\rightarrow \rightarrow [y := 1]^2 \\
\rightarrow \rightarrow \rightarrow [x > 1]^3 \\
\rightarrow \rightarrow \rightarrow \rightarrow [y := x \times y]^4 \\
\rightarrow \rightarrow \rightarrow \rightarrow \rightarrow [x := x - 1]^5
\end{array} \]

\textit{Vars}: the set of variables appearing in the program

\textit{Vars} = \{x, y\}
Reaching definitions analysis: lattice

\((D, \leq) := (\mathcal{P}(\text{Vars} \times (B \cup \{?\})), \subseteq), \bigcup X := \bigcup X, \bot := \emptyset\)

The meaning of \((v, i) \in d \in \mathcal{P}(\text{Vars} \times (B \cup \{?\}))\) is as follows:

- if \(i = ?\), then \(v\) is possibly not initialized yet
- if \(i \in B\), then the last initialization of \(v\) may be in block \(i\)

The lattice satisfies (ACC), since it is finite.
Reaching definitions analysis: transfer functions

initial extremal value \( i := \{(v, ?) \mid v \in Vars\}\)

transfer functions \( f_b : D \to D \)

\[
f_b : P(Vars \times (B \cup \{?\})) \to P(Vars \times (B \cup \{?\}))
\]

\[
f_b(X) := (X \setminus \text{kill}(b)) \cup \text{gen}(b)
\]

- \( \text{kill}(b) \): assignments overwritten in block \( b \)

\[
\text{kill}(b) := \begin{cases} 
\{(v, ?)\} \cup \{(v, b') \mid b' \in B\} & \text{if } b = [v := e]^l \\
\emptyset & \text{or } b = [\text{read}(v)]^l, \\
\emptyset & \text{otherwise.}
\end{cases}
\]

- \( \text{gen}(b) \): assignments generated in block \( b \)

\[
\text{gen}(b) := \begin{cases} 
\{(v, b)\} & \text{if } b = [v := e]^l \\
\emptyset & \text{or } b = [\text{read}(v)]^l, \\
\emptyset & \text{otherwise.}
\end{cases}
\]

The transfer functions are monotonic.
### Example

<table>
<thead>
<tr>
<th>block $b$</th>
<th>kill($b$)</th>
<th>gen($b$)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$[x := 5]^1$</td>
<td>${(x,?), (x,1), (x,5)}$</td>
<td>${(x,1)}$</td>
</tr>
<tr>
<td>$[y := 1]^2$</td>
<td>${(y,?), (y,2), (y,4)}$</td>
<td>${(y,2)}$</td>
</tr>
<tr>
<td>$[x &gt; 1]^3$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>$[y := x \times y]^4$</td>
<td>${(y,?), (y,2), (y,4)}$</td>
<td>${(y,4)}$</td>
</tr>
<tr>
<td>$[x := x - 1]^5$</td>
<td>${(x,?), (x,1), (x,5)}$</td>
<td>${(x,5)}$</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
X_1 &= i \\
X_2 &= f_1(X_1) \\
X_3 &= f_2(X_2) \cup f_5(X_5) \\
X_4 &= f_3(X_3) \\
X_5 &= f_4(X_4)
\end{align*}
\]

\[
\begin{align*}
X_1 &= \{(x,?), (y,?)\} \\
X_2 &= (X_1 \setminus \{(x,?), (x,1), (x,5)\}) \cup \{(x,1)\} \\
X_3 &= (X_2 \setminus \{(y,?), (y,2), (y,4)\}) \cup \{(y,2)\} \cup (X_5 \setminus \{(x,?), (x,1), (x,5)\}) \cup \{(x,5)\} \\
X_4 &= X_3 \\
X_5 &= (X_4 \setminus \{(y,?), (y,2), (y,4)\}) \cup \{(y,4)\}
\end{align*}
\]
Example

\[
\begin{align*}
X_1 &= i = \{(x,?), (y,?)\} \\
X_2 &= f_1(X_1) = (X_1 \setminus \{(x,?), (x,1), (x,5)\}) \cup \{(x,1)\} \\
X_3 &= f_2(X_2) \cup f_5(X_5) = (X_2 \setminus \{(y,?), (y,2), (y,4)\}) \cup \{(y,2)\} \\
        &\quad \cup (X_5 \setminus \{(x,?), (x,1), (x,5)\}) \cup \{(x,5)\} \\
X_4 &= f_3(X_3) = X_3 \\
X_5 &= f_4(X_4) = (X_4 \setminus \{(y,?), (y,2), (y,4)\}) \cup \{(y,4)\}
\end{align*}
\]

compute a solution by computing the least fixpoint of the function

\[
g_S : \mathcal{P}(\text{Vars} \times (B \cup \{?\}))^5 \rightarrow \mathcal{P}(\text{Vars} \times (B \cup \{?\}))^5
\]

in \((\mathcal{P}(\text{Vars} \times (B \cup \{?\}))^5, \subseteq^5)\), starting from \((\emptyset, \emptyset, \emptyset, \emptyset, \emptyset, \emptyset)\)
Example

\[X_1 = i = \{(x,?), (y,?)\}\]
\[X_2 = f_1(X_1) = (X_1 \setminus \{(x,?), (x,1), (x,5)\}) \cup \{(x,1)\}\]
\[X_3 = f_2(X_2) \cup f_5(X_5) = (X_2 \setminus \{(y,?), (y,2), (y,4)\}) \cup \{(y,2)\}\]
\[\cup (X_5 \setminus \{(x,?), (x,1), (x,5)\}) \cup \{(x,5)\}\]
\[X_4 = f_3(X_3) = X_3\]
\[X_5 = f_4(X_4) = (X_4 \setminus \{(y,?), (y,2), (y,4)\}) \cup \{(y,4)\}\]

The least solution is

\[X_1 = \{(x,?), (y,?)\}\]
\[X_2 = \{(y,?), (x,1)\}\]
\[X_3 = \{(x,1), (y,2), (y,4), (x,5)\}\]
\[X_4 = \{(x,1), (y,2), (y,4), (x,5)\}\]
\[X_5 = \{(x,1), (y,4), (x,5)\}\]
Available expressions analysis

**Goal:** For each program point determine which expressions must have already been computed, and not later modified, on all executions reaching this program point.

**Applications:** Avoid the recomputation of expressions.

**Classification:** We need a forward analysis that computes information about the executions up to a certain program point. We have to perform a must analysis underapproximating the information along all possible executions.
Available expressions analysis: CFG

\[
\begin{align*}
  [x & := a + b]^1; \\
  [y & := a * b]^2; \\
  \textbf{while} & [y > a + b]^3 \textbf{ do } \\
  & \{ \\
  & \quad [a := a + 1]^4; \\
  & \quad [x := a + b]^5
  \}
\end{align*}
\]

\[G = (B, E, F), E = \{1\}\]

\[\rightarrow [x := a + b]^1 \quad \text{initial} \]
\[\rightarrow [y := a * b]^2 \]
\[\rightarrow [y > a + b]^3 \]
\[\rightarrow [a := a + 1]^4 \]
\[\rightarrow [x := a + b]^5 \]

\textbf{AExp}: set of non-trivial arithmetic expressions appearing in the program

\textbf{AExp} = \{a+b, a*b, a+1\}

\textbf{Vars}(e) is the set of variables occurring in expression \(e\).
\textbf{AExp}(e) is the set of sub-expressions of expression \(e\).
Available expressions analysis: lattice

\((D, \leq) := (\mathcal{P}(AExp), \supseteq), \bigcup X := \bigcap X, \bot := AExp\)

An element \(d \in \mathcal{P}(AExp)\) denotes a set of expressions.

The lattice satisfies (ACC), since it is finite.
Available expressions analysis: transfer functions

initial extremal value $i := \emptyset$

transfer functions $f_b : D \to D$

$$f_b : \mathcal{P}(AExp) \to \mathcal{P}(AExp)$$

$$f_b(X) := (X \setminus kill(b)) \cup gen(b)$$

- $kill(b)$: expressions whose value is modified by $b$

$$kill(b) := \begin{cases} 
\{ e' \in AExp \mid v \in Vars(e') \} & \text{if } b = [v := e]^l \\
\emptyset & \text{or } b = [\text{read}(v)]^l, \\
\emptyset & \text{otherwise}. 
\end{cases}$$

- $gen(b)$: expressions used and not modified in block $b$

$$gen(b) := \begin{cases} 
\{ e' \in AExp(e) \mid v \notin Vars(e') \} & \text{if } b = [v := e]^l, \\
AExp(e) & \text{if } b = [e]^l \text{ for condition } e, \\
\emptyset & \text{or } b = [\text{print}(e)]^l, \\
\emptyset & \text{otherwise}. 
\end{cases}$$

The transfer functions are monotonic.
### Example

<table>
<thead>
<tr>
<th>block $b$</th>
<th>$kill(b)$</th>
<th>$gen(b)$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$[x := a + b]$</td>
<td>$\emptyset$</td>
<td>${a + b}$</td>
</tr>
<tr>
<td>$[y := a \ast b]$</td>
<td>$\emptyset$</td>
<td>${a \ast b}$</td>
</tr>
<tr>
<td>$[y &gt; a + b]$</td>
<td>$\emptyset$</td>
<td>${a + b}$</td>
</tr>
<tr>
<td>$[a := a + 1]$</td>
<td>${a + b, a \ast b, a + 1}$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>$[x := a + b]$</td>
<td>$\emptyset$</td>
<td>${a + b}$</td>
</tr>
</tbody>
</table>

\[
X_1 = i = \emptyset \\
X_2 = f_1(X_1) = X_1 \cup \{a + b\} \\
X_3 = f_2(X_2) \cap f_5(X_5) = (X_2 \cup \{a \ast b\}) \cap (X_5 \cup \{a + b\}) \\
X_4 = f_3(X_3) = X_3 \cup \{a + b\} \\
X_5 = f_4(X_4) = X_4 \setminus \{a + b, a \ast b, a + 1\}
\]
Example

\[
\begin{align*}
X_1 &= i = \emptyset \\
X_2 &= f_1(X_1) = X_1 \cup \{a + b\} \\
X_3 &= f_2(X_2) \cup f_5(X_5) = (X_2 \cup \{a \ast b\}) \cap (X_5 \cup \{a + b\}) \\
X_4 &= f_3(X_3) = X_3 \cup \{a + b\} \\
X_5 &= f_4(X_4) = X_4 \setminus \{a + b, a \ast b, a + 1\}
\end{align*}
\]

compute a solution by computing the least fixpoint of the function

\[g_S : \mathcal{P}(AExp)^5 \to \mathcal{P}(AExp)^5\]

in \((\mathcal{P}(AExp)^5, \supseteq^5)\) starting at \((AExp, AExp, AExp, AExp, AExp)\)
Example

\[
\begin{align*}
X_1 &= i = \emptyset \\
X_2 &= f_1(X_1) = X_1 \cup \{a + b\} \\
X_3 &= f_2(X_2) \cup f_5(X_5) = (X_2 \cup \{a \ast b\}) \cap (X_5 \cup \{a + b\}) \\
X_4 &= f_3(X_3) = X_3 \cup \{a + b\} \\
X_5 &= f_4(X_4) = X_4 \setminus \{a + b, a \ast b, a + 1\}
\end{align*}
\]

the least solution with respect to \(\supseteq^5\) is

\[
X_1 = X_5 = \emptyset \quad X_2 = X_3 = X_4 = \{a + b\}
\]

We computed the least fixpoint with respect to \(\supseteq^5\), which is also the greatest fixpoint with respect to the dual lattice \((\mathcal{P}(AExp)^5, \subseteq^5)\).