Program Analysis - 1

Speaker: Xie Li

July 20, 2021

Overview

Sources:

- Principles of Program Analysis, Nielson 1999.
- Slides of Yingfei Xiong.

Contents:

1. Introduction (**Part of**).
2. **Data Flow Analysis**.
3. Constraint Based Analysis.
4. Abstract Interpretation.
5. Type and Effect System.

Data Flow Analysis: Preliminaries

- **Preliminaries on Partial Ordered Sets**.
- Reaching Definition Analysis.
- Live Variables Analysis.
- Theoretical Properties.

Preliminaries

Definition (Partial Order Set)

- 偏序是一个二元组(S, ⊑)，其中S是一个集合，⊑是一个定义在S上的二元关系，并且满足如下性质：
  - 自反性：$\forall a \in S: a \sqsubseteq a$
  - 传递性：$\forall x, y, z \in S: x \sqsubseteq y \wedge y \sqsubseteq z \Rightarrow x \sqsubseteq z$
  - 非对称性：$x \sqsubseteq y \wedge y \sqsubseteq x \Rightarrow x = y$

- *Upper bound*: A subset Y of S has $l \in S$ as upper bound if $\forall l' \in Y. l' \sqsubseteq l$.
- *Greatest upper bound*: $l \in S$ is the greatest upper bound of Y iff forall upper bounds $l_0 \in S$, $l \sqsubseteq l_0$.
- Similar definition for lower bound and least lower bound.

We use $\sqcap Y$ and $\sqcup Y$ to represent the greatest lower bound (meet) and least lower bound (join) of Y.
$l_1 \sqcap l_2, l_1 \sqcup l_2$.

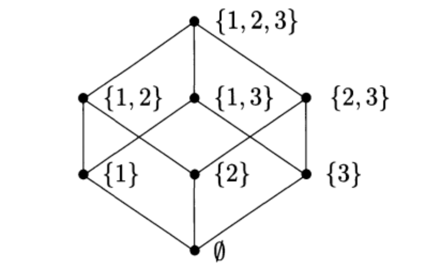
Complete Lattice

Definition (Complete Lattice)

A *complete lattice* $L = (L, \sqsubseteq) = (L, \sqsubseteq, \sqcup, \sqcap, \top, \bot)$.

Example

A complete lattice defined on the superset of $\{1, 2, 3\}$ where the partial order is $\subseteq$.



Monotone Function

Definition (Monotone Function)

A *monotone function* is a function $f : L_1 \to L_2$ between partially ordered sets $(L_1, \sqsubseteq_1)$ and $(L_2, \sqsubseteq_2)$ s.t.

$$\forall l, l' \in L_1.(l \sqsubseteq_1 l' \implies f(l) \sqsubseteq_2 f(l'))$$

Example

The monotone (increasing) function from $(\mathbb{R}, \leq)$ to $(\mathbb{R}, \leq)$.

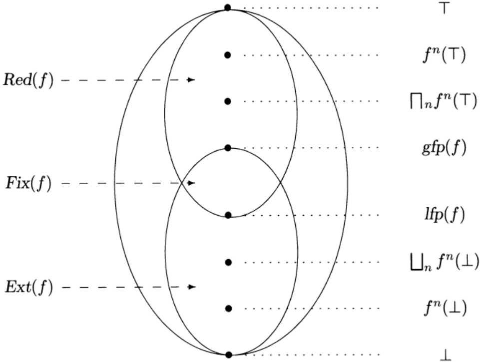
Fixed Points

Given a complete lattice $(L, \sqsubseteq, \sqcup, \sqcap, \top, \bot)$,

$$Red(f) = \{l \mid f(l) \sqsubseteq l\}$$
$$Ext(f) = \{l \mid f(l) \sqsupseteq l\}$$
$$Fix(f) = \{l \mid f(l) = l\}$$

Least fix-point and greatest fix-point:

$$gfp(f) = \bigsqcup Fix(f)$$
$$lfp(f) = \bigsqcap Fix(f)$$



Tarski Fixed Point Theorem

Proposition A.10
Let $L = (L, \sqsubseteq, \bigsqcup, \bigsqcap, \bot, \top)$ be a complete lattice. If $f : L \to L$ is a monotone function then $lfp(f)$ and $gfp(f)$ satisfy:

$$\begin{aligned} lfp(f) &= \bigsqcap Red(f) \in Fix(f) \\ gfp(f) &= \bigsqcup Ext(f) \in Fix(f) \end{aligned}$$

Proof.

$$f(l_0) \sqsubseteq f(l) \sqsubseteq l \text{ for all } l \in Red(f)$$

Data Flow Analysis

- Preliminaries on Partial Ordered Sets.
- **Reaching Definition Analysis**.
- Live Variables Analysis.
- Theoretical Properties.

Basic Notations

$$\begin{array}{rcll} a & \in & \mathbf{AExp} & \text{arithmetic expressions} \\ b & \in & \mathbf{BExp} & \text{boolean expressions} \\ S & \in & \mathbf{Stmt} & \text{statements} \end{array} \qquad \begin{array}{rcll} x, y & \in & \mathbf{Var} & \text{variables} \\ n & \in & \mathbf{Num} & \text{numerals} \\ \ell & \in & \mathbf{Lab} & \text{labels} \end{array}$$

$$\begin{array}{rcll} op_a & \in & \mathbf{Op}_a & \text{arithmetic operators} \\ op_b & \in & \mathbf{Op}_b & \text{boolean operators} \\ op_r & \in & \mathbf{Op}_r & \text{relational operators} \end{array}$$

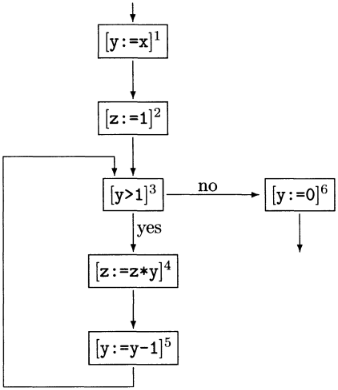
The syntax of the language is given by the following *abstract syntax*:

$$\begin{array}{rcl} a & ::= & x \mid n \mid a_1 \ op_a \ a_2 \\ b & ::= & \texttt{true} \mid \texttt{false} \mid \texttt{not}\ b \mid b_1 \ op_b \ b_2 \mid a_1 \ op_r \ a_2 \\ S & ::= & [x := a]^\ell \mid [\texttt{skip}]^\ell \mid S_1; S_2 \mid \\ & & \texttt{if}\ [b]^\ell\ \texttt{then}\ S_1\ \texttt{else}\ S_2 \mid \texttt{while}\ [b]^\ell\ \texttt{do}\ S \end{array}$$

Example of Program

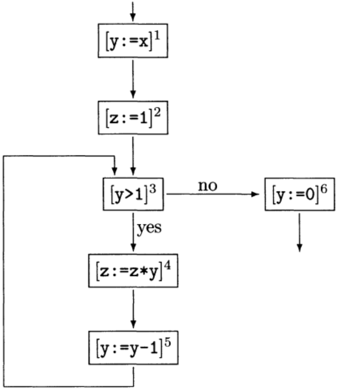
Example 1.1 An example of a program written in this language is the following which computes the factorial of the number stored in **x** and leaves the result in **z**:

$$[\texttt{y:=x}]^1;\ [\texttt{z:=1}]^2;\ \texttt{while}\ [\texttt{y>1}]^3\ \texttt{do}\ ([\texttt{z:=z*y}]^4;\ [\texttt{y:=y-1}]^5);\ [\texttt{y:=0}]^6 \qquad \blacksquare$$



Other Notations

$$init, final, blocks, labels, flow, flow^R$$



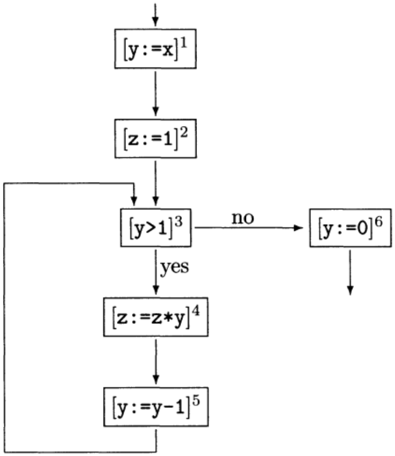
Reaching Definition Analysis

Reaching definition: An assignment of the form $[x := a]^l$ *may reach* a certain program point if there is an execution of the program where x was last assigned at value at l when the program point is reached.

Program points: exits and entries of elementary blocks.

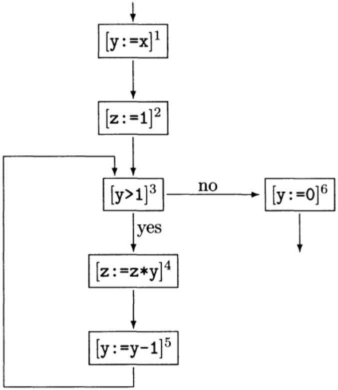
Problem: For each program points, find a set of pairs like (x, l) to represent the assignment that may reach the program points.

Notations: $(x, ?), (x, l)$



Data Flow Analysis: Equational Approach

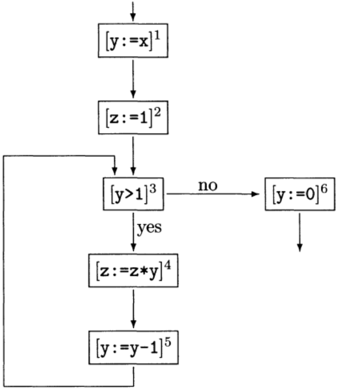
$$
\begin{aligned}
\mathsf{RD}_{exit}(1) &= (\mathsf{RD}_{entry}(1)\backslash\{(\mathrm{y},\ell) \mid \ell \in \mathbf{Lab}\}) \cup \{(\mathrm{y},1)\} \\
\mathsf{RD}_{exit}(2) &= (\mathsf{RD}_{entry}(2)\backslash\{(\mathrm{z},\ell) \mid \ell \in \mathbf{Lab}\}) \cup \{(\mathrm{z},2)\} \\
\mathsf{RD}_{exit}(3) &= \mathsf{RD}_{entry}(3) \\
\mathsf{RD}_{exit}(4) &= (\mathsf{RD}_{entry}(4)\backslash\{(\mathrm{z},\ell) \mid \ell \in \mathbf{Lab}\}) \cup \{(\mathrm{z},4)\} \\
\mathsf{RD}_{exit}(5) &= (\mathsf{RD}_{entry}(5)\backslash\{(\mathrm{y},\ell) \mid \ell \in \mathbf{Lab}\}) \cup \{(\mathrm{y},5)\} \\
\mathsf{RD}_{exit}(6) &= (\mathsf{RD}_{entry}(6)\backslash\{(\mathrm{y},\ell) \mid \ell \in \mathbf{Lab}\}) \cup \{(\mathrm{y},6)\}
\end{aligned}
$$



Data Flow Analysis: Equational Approach

$$\begin{aligned}
\mathsf{RD}_{entry}(2) &= \mathsf{RD}_{exit}(1)\\[1em]
\mathsf{RD}_{entry}(3) &= \mathsf{RD}_{exit}(2) \cup \mathsf{RD}_{exit}(5)\\
\mathsf{RD}_{entry}(4) &= \mathsf{RD}_{exit}(3)\\
\mathsf{RD}_{entry}(5) &= \mathsf{RD}_{exit}(4)\\
\mathsf{RD}_{entry}(6) &= \mathsf{RD}_{exit}(3)
\end{aligned}$$

$$\mathsf{RD}_{entry}(1) = \{(\mathtt{x}, ?), (\mathtt{y}, ?), (\mathtt{z}, ?)\}$$



How these equations generated?

Forward analysis.

kill and *gen* functions

$$
\begin{aligned}
kill_{\mathsf{RD}}([x := a]^{\ell}) &= \{(x, ?)\} \\
&\quad \cup \{(x, \ell') \mid B^{\ell'} \text{ is an assignment to } x \text{ in } S_\star\} \\
kill_{\mathsf{RD}}([\texttt{skip}]^{\ell}) &= \emptyset \\
kill_{\mathsf{RD}}([b]^{\ell}) &= \emptyset \\
gen_{\mathsf{RD}}([x := a]^{\ell}) &= \{(x, \ell)\} \\
gen_{\mathsf{RD}}([\texttt{skip}]^{\ell}) &= \emptyset \\
gen_{\mathsf{RD}}([b]^{\ell}) &= \emptyset
\end{aligned}
$$

data flow equations: $\mathsf{RD}^{=}$

$$
\begin{aligned}
\mathsf{RD}_{entry}(\ell) &= \begin{cases} \{(x, ?) \mid x \in FV(S_\star)\} & \text{if } \ell = init(S_\star) \\ \bigcup\{\mathsf{RD}_{exit}(\ell') \mid (\ell', \ell) \in flow(S_\star)\} & \text{otherwise} \end{cases} \\
\mathsf{RD}_{exit}(\ell) &= (\mathsf{RD}_{entry}(\ell) \backslash kill_{\mathsf{RD}}(B^{\ell})) \cup gen_{\mathsf{RD}}(B^{\ell}) \\
&\quad \text{where } B^{\ell} \in blocks(S_\star)
\end{aligned}
$$

Solve the equation system.

For the above program we obtain twelve sets at the program points:

$$\vec{\mathsf{RD}} = (\mathsf{RD}_{entry}(1), \mathsf{RD}_{exit}(1), \ldots, \mathsf{RD}_{entry}(6), \mathsf{RD}_{exit}(6))$$

We can regard one step execution of the program as a function F on $\vec{\mathsf{RD}}$. i.e.

$$\vec{\mathsf{RD}} = F(\vec{\mathsf{RD}})$$

$$F(\vec{\mathsf{RD}}) = (F_{entry}(1)(\vec{\mathsf{RD}}), F_{exit}(1)(\vec{\mathsf{RD}}), \cdots, F_{entry}(6)(\vec{\mathsf{RD}}), F_{exit}(6)(\vec{\mathsf{RD}}))$$

where e.g.:

$$F_{entry}(3)(\cdots, \mathsf{RD}_{exit}(2), \cdots, \mathsf{RD}_{exit}(5), \cdots) = \mathsf{RD}_{exit}(2) \cup \mathsf{RD}_{exit}(5)$$

Solve the equation system.

Definition of the function F:

$$F : (\mathcal{P}(\mathbf{Var}_\star \times \mathbf{Lab}_\star))^{12} \rightarrow (\mathcal{P}(\mathbf{Var}_\star \times \mathbf{Lab}_\star))^{12}$$

Partial order for the complete lattice:

$$\vec{\mathsf{RD}} \sqsubseteq \vec{\mathsf{RD}}' \quad \text{iff} \quad \forall i : \mathsf{RD}_i \subseteq \mathsf{RD}'_i$$

F is a monotone function:

$$\vec{\mathsf{RD}} \sqsubseteq \vec{\mathsf{RD}}' \;\; \text{implies} \;\; F(\vec{\mathsf{RD}}) \sqsubseteq F(\vec{\mathsf{RD}}')$$

Fix-point of F is the least solution to the equation system.

$$F^{n+1}(\vec{\emptyset}) = F^{n}(\vec{\emptyset})$$

Data Flow Analysis: Live Variable Analysis

- Reaching Definition Analysis.
- Preliminaries on Partial Ordered Sets.
- **Live Variables Analysis**.
- Theoretical Properties.

Live Variable Analysis

Live variable: varDef $\longrightarrow$ varUsed
Along the path, there is no redefinition to the variable.
Then we call it live at the exit of the program.
Problem: for each program point, which variable *may* be live at the exit from the point.

Example

```
int main(){
    int x = 10;
    int y = 11;
    int z = x + 1;
    return z;
}
```

Live variable analysis is useful in dead code elimination.

Live Variable Analysis

Backward analysis. Smallest solution.

kill and *gen* functions

$$
\begin{aligned}
\mathit{kill}_{\mathsf{LV}}([x := a]^\ell) &= \{x\} \\
\mathit{kill}_{\mathsf{LV}}([\mathtt{skip}]^\ell) &= \emptyset \\
\mathit{kill}_{\mathsf{LV}}([b]^\ell) &= \emptyset \\
\\
\mathit{gen}_{\mathsf{LV}}([x := a]^\ell) &= FV(a) \\
\mathit{gen}_{\mathsf{LV}}([\mathtt{skip}]^\ell) &= \emptyset \\
\mathit{gen}_{\mathsf{LV}}([b]^\ell) &= FV(b)
\end{aligned}
$$

data flow equations: $\mathsf{LV}^=$

$$
\begin{aligned}
\mathsf{LV}_{exit}(\ell) &= \begin{cases} \emptyset & \text{if } \ell \in \mathit{final}(S_\star) \\ \bigcup\{\mathsf{LV}_{entry}(\ell') \mid (\ell', \ell) \in \mathit{flow}^R(S_\star)\} & \text{otherwise} \end{cases} \\
\mathsf{LV}_{entry}(\ell) &= (\mathsf{LV}_{exit}(\ell) \backslash \mathit{kill}_{\mathsf{LV}}(B^\ell)) \cup \mathit{gen}_{\mathsf{LV}}(B^\ell) \\
& \text{where } B^\ell \in \mathit{blocks}(S_\star)
\end{aligned}
$$

Table 2.4: Live Variables Analysis.

The Difference between May and Must

$$(\texttt{while } [\texttt{x>1}]^{\ell} \texttt{ do } [\texttt{skip}]^{\ell'}); [\texttt{x:=x+1}]^{\ell''}$$

$$\begin{aligned}
\mathsf{LV}_{entry}(\ell) &= \mathsf{LV}_{exit}(\ell) \cup \{\mathrm{x}\} \\
\mathsf{LV}_{entry}(\ell') &= \mathsf{LV}_{exit}(\ell')
\end{aligned}$$

$$\begin{aligned}
\mathsf{LV}_{entry}(\ell'') &= \{\mathrm{x}\} \\
\mathsf{LV}_{exit}(\ell) &= \mathsf{LV}_{entry}(\ell') \cup \mathsf{LV}_{entry}(\ell'') \\
\mathsf{LV}_{exit}(\ell') &= \mathsf{LV}_{entry}(\ell) \\
\mathsf{LV}_{exit}(\ell'') &= \emptyset
\end{aligned}$$

After some calculations:

$$\mathsf{LV}_{exit}(\ell) = \mathsf{LV}_{exit}(\ell) \cup \{\mathrm{x}\}$$

The Difference between May and Must

数据流分析单调框架

- 一个控制流图(V, E)
- 一个有限高度的半格$(S,\sqcap)$
- 一个entry的初值I
- 一组结点转换函数，对任意$v \in V - entry$存在一个结点转换函数f_v

- 注意：对于逆向分析，变换控制流图方向再应用单调框架即可

数据流分析实现算法

$DATA_{entry} = I$

$\forall v \in (V - entry): DATA_v \leftarrow \top$

$ToVisit \leftarrow V - entry$

While(ToVisit.size > 0) {

  v ← ToVisit中任意结点

  ToVisit -= v

  $MEET_v \leftarrow \sqcap_{w \in pred(v)} DATA_w$

  $If(DATA_V \neq f_v(MEET_v))$ $ToVisit \cup= succ(v)$

  $DATA_v \leftarrow f_v(MEET_v)$

}

Data Flow Analysis: Live Variable Analysis

- Reaching Definition Analysis.
- Preliminaries on Partial Ordered Sets.
- Live Variables Analysis.
- **Theoretical Properties**.

Theoretical Properties