Lecture 03 – Background + intro AI + dataflow and connection to AI

PROGRAM ANALYSIS & SYNTHESIS

Eran Yahav
Previously...

- structural operational semantics
- trace semantics
  - meaning of a program as the set of its traces
- abstract interpretation
  - abstractions of the trace semantics

Example: Cormac’s PLDI’08

Example: Kuperstein PLDI’11
Today

- abstract interpretation
- some basic math background
  - lattices
  - functions
- Galois connection
- A little more dataflow analysis
  - monotone framework
Trace Semantics

\[ [y := x]^1; \]
\[ [z := 1]^2; \]
\[ \text{while } [y > 0]^3 ( \]
\[ \quad [z := z \times y]^4; \]
\[ \quad [y := y - 1]^5; \]
\[ \) \]
\[ [y := o]^6 \]
\[ \]
\[ < 1, \{ x \mapsto 42, y \mapsto 0, z \mapsto 0 \} > \Rightarrow < 2, \{ x \mapsto 42, y \mapsto 42, z \mapsto 0 \} > \Rightarrow \]
\[ < 3, \{ x \mapsto 42, y \mapsto 42, z \mapsto 1 \} > \Rightarrow < 4, \{ x \mapsto 42, y \mapsto 42, z \mapsto 1 \} > \ldots \]

\[ \]
\[ [y := x]^1 \]
\[ [z := 1]^2 \]
\[ < 1, \{ x \mapsto 73, y \mapsto 0, z \mapsto 0 \} > \Rightarrow < 2, \{ x \mapsto 73, y \mapsto 73, z \mapsto 0 \} > \Rightarrow \]
\[ < 3, \{ x \mapsto 73, y \mapsto 73, z \mapsto 1 \} > \Rightarrow < 4, \{ x \mapsto 73, y \mapsto 73, z \mapsto 1 \} > \ldots \]

\[ \]

\[ \]

- note that input (x) is unknown
- while loop?
- trace semantics is not computable
Abstract Interpretation

\[ C \xrightarrow{[S]} C' \]

\[ \sigma \xrightarrow{[S]} \sigma' \]

Abstract state

Concrete state

Set of states

\[ C \equiv C'' \]
Dataflow Analysis

The assignment \texttt{lab: var := exp} reaches \texttt{lab'} if there is an execution where \texttt{var} was last assigned at \texttt{lab}

(adapted from Nielson, Nielson & Hankin)
Dataflow Analysis

1: y := x;  \{ (x,?), (y,?), (z,?) \}
2: z := 1;  \{ (x,?), (y,1), (z,?) \}
3: while y > 0 {
   4: z := z * y; \{ (x,?), (y,1), (z,2), (y,5), (z,4) \}
   5: y := y - 1 \{ (x,?), (y,5), (z,4) \}
} \{ (x,?), (y,6), (z,2), (z,4) \}
6: y := 0 \{ (x,?), (y,?), (z,4), (y,5) \}

- Reaching Definitions
  - The assignment \texttt{lab: var := exp} reaches \texttt{lab’} if there is an execution where \texttt{var} was last assigned at \texttt{lab}

(adapted from Nielson, Nielson & Hankin)
Dataflow Analysis

- Build control-flow graph
- Assign transfer functions
- Compute fixed point
Control-Flow Graph

1: \( y := x \);
2: \( z := 1 \);
3: while \( y > 0 \) {
4: \( z := z \times y \);
5: \( y := y - 1 \)
}
6: \( y := 0 \)
Transfer Functions

1: \( y := x \)

\[
\text{out}(1) = \text{in}(1) \setminus \{(y,l) | l \in \text{Lab}\} \cup \{(y,1)\}
\]

2: \( z := 1 \)

\[
\text{out}(2) = \text{in}(2) \setminus \{(z,l) | l \in \text{Lab}\} \cup \{(z,2)\}
\]

3: \( y > 0 \)

\[
\text{out}(3) = \text{in}(3)
\]

4: \( z = z \times y \)

\[
\text{out}(4) = \text{in}(4) \setminus \{(z,l) | l \in \text{Lab}\} \cup \{(z,4)\}
\]

5: \( y = y - 1 \)

\[
\text{out}(5) = \text{in}(5) \setminus \{(y,l) | l \in \text{Lab}\} \cup \{(y,5)\}
\]

6: \( y := 0 \)

\[
\text{out}(6) = \text{in}(6) \setminus \{(y,l) | l \in \text{Lab}\} \cup \{(y,6)\}
\]

in(1) = \{(x,?), (y,?), (z,?)\}

in(2) = out(1)

in(3) = out(2) U out(5)

in(4) = out(3)

in(5) = out(4)

in(6) = out(3)
System of Equations

\[
\begin{align*}
\text{in}(1) &= \{(x,?), (y,?), (z,?)\} \\
\text{in}(2) &= \text{out}(1) \\
\text{in}(3) &= \text{out}(2) \cup \text{out}(5) \\
\text{in}(4) &= \text{out}(3) \\
\text{in}(5) &= \text{out}(4) \\
\text{in}(6) &= \text{out}(3) \\
\text{out}(1) &= \text{in}(1) \setminus \{(y,l) \mid l \in \text{Lab}\} \cup \{(y,1)\} \\
\text{out}(2) &= \text{in}(2) \setminus \{(z,l) \mid l \in \text{Lab}\} \cup \{(z,2)\} \\
\text{out}(3) &= \text{in}(3) \\
\text{out}(4) &= \text{in}(4) \setminus \{(z,l) \mid l \in \text{Lab}\} \cup \{(z,4)\} \\
\text{out}(5) &= \text{in}(5) \setminus \{(y,l) \mid l \in \text{Lab}\} \cup \{(y,5)\} \\
\text{out}(6) &= \text{in}(6) \setminus \{(y,l) \mid l \in \text{Lab}\} \cup \{(y,6)\}
\end{align*}
\]

\[
F: (\varnothing (\text{Var} \times \text{Lab}) )^{12} \rightarrow (\varnothing (\text{Var} \times \text{Lab}) )^{12}
\]
System of Equations

| In(1)   | ∅          |
| In(2)   | ∅          |
| In(3)   | ∅          |
| In(4)   | ∅          |
| In(5)   | ∅          |
| In(6)   | ∅          |
| Out(1)  | ∅          |
| Out(2)  | ∅          |
| Out(3)  | ∅          |
| Out(4)  | ∅          |
| Out(5)  | ∅          |
| Out(6)  | ∅          |

\[ F: (\varnothing (\text{Var} \times \text{Lab}))^{12} \rightarrow (\varnothing (\text{Var} \times \text{Lab}))^{12} \]

\[ \text{in}(1) = \{(x,?), (y,?), (z,?)\}, \text{in}(2) = \text{out}(1), \text{in}(3) = \text{out}(2) \cup \text{out}(5), \text{in}(4) = \text{out}(3), \text{in}(5) = \text{out}(4), \text{in}(6) = \text{out}(3) \]

\[ \text{out}(1) = \text{in}(1) \setminus \{(y,l) \mid l \in \text{Lab}\} \cup \{(y,1)\}, \text{out}(2) = \text{in}(2) \setminus \{(z,l) \mid l \in \text{Lab}\} \cup \{(z,2)\} \]

\[ \text{out}(3) = \text{in}(3), \text{out}(4) = \text{in}(4) \setminus \{(z,l) \mid l \in \text{Lab}\} \cup \{(z,4)\} \]

\[ \text{out}(5) = \text{in}(5) \setminus \{(y,l) \mid l \in \text{Lab}\} \cup \{(y,5)\}, \text{out}(6) = \text{in}(6) \setminus \{(y,l) \mid l \in \text{Lab}\} \cup \{(y,6)\} \]
# System of Equations

<table>
<thead>
<tr>
<th>In(1)</th>
<th>RD∅</th>
<th>F(RD∅)</th>
<th>F(F(RD∅))</th>
<th>F(F(F(RD∅))))</th>
<th>F(F(F(F(RD∅))))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ø</td>
<td>∅</td>
<td>{(x,?), (y,?), (z,?)}</td>
<td>==</td>
<td>==</td>
<td>==</td>
</tr>
<tr>
<td>In(2)</td>
<td>Ø</td>
<td>∅</td>
<td>{(y,1)}</td>
<td>{(x,?), (y,1), (z,?)}</td>
<td>==</td>
</tr>
<tr>
<td>In(3)</td>
<td>Ø</td>
<td>∅</td>
<td>{(z,2), (y,5)}</td>
<td>{(z,2), (y,5)}</td>
<td>{(z,2), (z,4), (y,5)}</td>
</tr>
<tr>
<td>In(4)</td>
<td>Ø</td>
<td>∅</td>
<td>∅</td>
<td>∅</td>
<td>{(z,2), (y,5)}</td>
</tr>
<tr>
<td>In(5)</td>
<td>Ø</td>
<td>∅</td>
<td>{(z,4)}</td>
<td>{(z,4)}</td>
<td>{(z,4)}</td>
</tr>
<tr>
<td>In(6)</td>
<td>Ø</td>
<td>∅</td>
<td>∅</td>
<td>∅</td>
<td>{(z,2), (y,5)}</td>
</tr>
<tr>
<td>Out(1)</td>
<td>Ø</td>
<td>{(y,1)}</td>
<td>{(x,?), (y,1), (z,?)}</td>
<td>==</td>
<td>==</td>
</tr>
<tr>
<td>Out(2)</td>
<td>Ø</td>
<td>{(z,2)}</td>
<td>{(z,2)}</td>
<td>{(z,2), (y,1)}</td>
<td>==</td>
</tr>
<tr>
<td>Out(3)</td>
<td>Ø</td>
<td>∅</td>
<td>∅</td>
<td>{(z,2), (y,5)}</td>
<td>{(z,2), (y,5)}</td>
</tr>
<tr>
<td>Out(4)</td>
<td>Ø</td>
<td>{(z,4)}</td>
<td>{(z,4)}</td>
<td>{(z,4)}</td>
<td>{(z,4)}</td>
</tr>
<tr>
<td>Out(5)</td>
<td>Ø</td>
<td>{(y,5)}</td>
<td>{(y,5)}</td>
<td>{(z,4), (y,5)}</td>
<td>{(z,4), (y,5)}</td>
</tr>
<tr>
<td>Out(6)</td>
<td>Ø</td>
<td>{(y,6)}</td>
<td>{(y,6)}</td>
<td>{(y,6)}</td>
<td>{(y,6)}</td>
</tr>
</tbody>
</table>

F: (Ø (Var x Lab))^{12} → (Ø (Var x Lab))^{12}

RD ⊆ RD’ when ∀i: RD(i) ⊆ RD’(i)
Monotonicity

\[ F: (\emptyset (\text{Var} \times \text{Lab}))^{12} \rightarrow (\emptyset (\text{Var} \times \text{Lab}))^{12} \]

<table>
<thead>
<tr>
<th></th>
<th>RD_\emptyset</th>
<th>F(RD_\emptyset)</th>
<th>F(F(RD_\emptyset))</th>
<th>F(F(F(RD_\emptyset)))</th>
<th>F(F(F(F(RD_\emptyset))))</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Out(1)</td>
<td>\emptyset</td>
<td>{(y,1)}</td>
<td>{(x,?), (y,1), (z,?)}</td>
<td>{(y,1)}</td>
<td>...</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ \text{out}(1) = \{(y,1)\} \text{ when } (x,?) \in \text{out}(1) \]
\[ \text{in}(1) \setminus \{(y, l) \mid l \in \text{Lab}\} \cup \{(y,1)\}, \text{ otherwise} \]

(silly example just for illustration)
Convergence
Least Fixed Point

- We will see later why it exists
- For now, mostly informally...

\[ F: \left( \wp(\text{Var} \times \text{Lab}) \right)^{12} \rightarrow \left( \wp(\text{Var} \times \text{Lab}) \right)^{12} \]

\[ \text{RD} \subseteq \text{RD}' \text{ when } \forall i: \text{RD}(i) \subseteq \text{RD}'(i) \]

F is monotone:
\[ \text{RD} \subseteq \text{RD}' \text{ implies that } F(\text{RD}) \subseteq F(\text{RD}') \]

\[ \text{RD}_{\emptyset} = (\emptyset, \emptyset, ..., \emptyset) \]

\[ F(\text{RD}_{\emptyset}), F(F(\text{RD}_{\emptyset})), F(F(F(\text{RD}_{\emptyset}))), \ldots, F^n(\text{RD}_{\emptyset}) \]

\[ F^{n+1}(\text{RD}_{\emptyset}) = F^n(\text{RD}_{\emptyset}) \]
Partially Ordered Sets (posets)

- Partial ordering is a relation
  \( \sqsubseteq : L \times L \rightarrow \{ \text{true, false} \} \) that is:
  - Reflexive (\( \forall l : l \sqsubseteq l \) )
  - Transitive (\( \forall l_1, l_2, l_3 : l_1 \sqsubseteq l_2 \land l_2 \sqsubseteq l_3 \Rightarrow l_1 \sqsubseteq l_3 \) )
  - Anti-symmetric (\( l_1 \sqsubseteq l_2 \land l_2 \sqsubseteq l_1 \Rightarrow l_1 = l_2 \) )

- A partially ordered set \((L, \sqsubseteq)\) is a set \(L\) equipped with a partial ordering \(\sqsubseteq\)

- For example: \((\wp(S), \sqsubseteq)\)

- Captures intuition of implication between facts
  - \(l_1 \sqsubseteq l_2\) will intuitively mean \(l_1 \Rightarrow l_2\)
  - Later, in abstract domains, when \(l_1 \sqsubseteq l_2\), we will say that \(l_1\) is “more precise” than \(l_2\) (represents fewer concrete states)
Bounds in Posets

- Given a poset \((L, \sqsubseteq)\)
- A subset \(Y\) of \(L\) has \(l \in L\) as an upper bound if \(\forall l' \in Y : l' \sqsubseteq l\)
- A subset \(Y\) of \(L\) has \(l \in L\) as a lower bound if \(\forall l' \in Y : l' \sqsupseteq l\)
  (we write \(l' \sqsupseteq l\) when \(l \sqsubseteq l'\))
- A least upper bound (lub) \(l\) of \(Y\) is an upper bound of \(Y\) that satisfies \(l \sqsubseteq l'\) whenever \(l'\) is another upper bound of \(Y\)
- A greatest lower bound (glb) \(l\) of \(Y\) is a lower bound of \(Y\) that satisfies \(l \sqsupseteq l'\) whenever \(l'\) is another lower bound of \(Y\)

- For any subset \(Y \subseteq L\)
  - If the lub exists then it is unique, and denoted by \(\sqcup Y\) (join)
    - We often write \(l_1 \sqcup l_2\) for \(\sqcup \{l_1, l_2\}\)
  - If the glb exists then it is unique, and denoted by \(\sqcap Y\) (meet)
    - We often write \(l_1 \sqcap l_2\) for \(\sqcap \{l_1, l_2\}\)
Complete Lattices

- A complete lattice \((L, \sqsubseteq)\) is a poset such that all subsets have least upper bounds as well as greatest lower bounds.
- In particular
  - \(\bot = \bigsqcup \emptyset = \bigcap L\) is the least element (bottom)
  - \(\top = \bigsqcup L = \bigcap \emptyset\) is the greatest element (top)
Complete Lattices: Examples

\[
\begin{array}{c}
\{1\} \\
\{2\} \\
\{3\} \\
\emptyset \\
\{1,2\} \\
\{1,3\} \\
\{2,3\} \\
\{1,2,3\}
\end{array}
\]
Complete Lattices: Examples

Complete lattice diagram with intervals from $\mathbb{R}$.
Moore Families

- A Moore family is a subset $Y$ of a complete lattice $(L, \sqsubseteq)$ that is closed under greatest lower bounds
  - $\forall Y' \subseteq Y: \bigcap Y' \in Y$
  - Hence, a Moore family is never empty
  - Always contains least element $\bigcap Y$ and a greatest element $\bigcap \emptyset$

- Why do we care?
  - We will see that Moore families correspond to choices of abstractions (as targets of upper closure operators)
  - More on this... later...
Moore Families: Examples
Chains

- In program analysis we will often look at sequences of properties of the form \( l_0 \sqsubseteq l_1 \sqsubseteq \ldots \sqsubseteq l_n \).

- Given a poset \( (L, \sqsubseteq) \) a subset \( Y \subseteq L \) is a chain if every two elements in \( Y \) are ordered:
  - \( \forall l_1, l_2 \in Y: (l_1 \sqsubseteq l_2) \) or \( (l_2 \sqsubseteq l_1) \)
  - When \( Y \) is a finite subset we say that the chain is a finite chain.

- A sequence of elements \( l_0 \sqsubseteq l_1 \sqsubseteq \ldots \in L \) is an ascending chain.

- A sequence of elements \( l_0 \sqsupseteq l_1 \sqsupseteq \ldots \in L \) is a descending chain.

- \( L \) has finite height when all chains in \( L \) are finite.

- A poset \( (L, \sqsubseteq) \) satisfies the ascending chain condition (ACC) if every ascending chain eventually stabilizes, that is:
  \[ \exists s \in \mathbb{N}: \forall n \in \mathbb{N}: n \geq s \Rightarrow l_n = l_s. \]
Chains: Example Posets

(a) $\mathbb{A}_\mathbb{C}$
(b) $\mathbb{I}_\mathbb{C}$

$\mathbb{I}_\mathbb{C}$

$\mathbb{A}_\mathbb{C}$

Finkelstein Height
Properties of Functions

- A function $f: L_1 \rightarrow L_2$ between two posets $(L_1, \sqsubseteq_1)$ and $(L_2, \sqsubseteq_2)$ is
  - Monotone (order-preserving)
    - $\forall l, l' \in L_1: l \sqsubseteq_1 l' \Rightarrow f(l) \sqsubseteq_2 f(l')$
    - Special case $f: L \rightarrow L$, $\forall l, l' \in L: l \sqsubseteq l' \Rightarrow f(l) \sqsubseteq f(l')$

- Distributive (join morphism)
  - $\forall l, l' \in L_1 f(l \sqcup l') = f(l) \sqcup f(l')$
Function Properties: Examples

\[ \{1, 2, 3\} \subseteq \{1, 2\} \]

\[ \{1, 2\} \subseteq \{1\} \leq \{1, 2\} \leq \{1, 2, 3\} \]

\[ f(\{1, 2, 3\}) = \{1, 2\} \]

\[ f(\{1\}, 2) = \{1\}, 2 \]
Function Properties: Examples

{1,2,3} → {1,2,3}

{1,2} → {2,3} → {1,2} → {2}

{2,3} → {1,2,3} → {2,3} → {2}
Function Properties: Examples
Function Properties: Examples

\[ f(\{1,3\}) \cup f(\emptyset) = \{1, 2\} \]
\[ f(\{1,3\} \cup \emptyset) = f(\{1\}) = \{1\} \]
Fixed Points

- Consider a complete lattice $L = (L, \sqsubseteq, \sqcup, \sqcap, \bot, T)$ and a monotone function $f: L \to L$
- An element $l \in L$ is
  - A fixed point iff $f(l) = l$
  - A pre-fixedpoint iff $l \sqsubseteq f(l)$
  - A post-fixedpoint iff $l \sqsupseteq f(l)$

- $\text{Fix}(f) = \text{set of all fixed points}$
- $\text{Ext}(f) = \text{set of all pre-FP}$
- $\text{Red}(f) = \text{set of all post-FP}$
Tarski’s Theorem

Consider a complete lattice \( L = (L, \leq, \cup, \cap, \bot, \top) \) and a monotone function \( f: L \to L \), then \( \text{lfp}(f) \) and \( \text{gfp}(f) \) exist and

- \( \text{lfp}(f) = \bigcap \text{Red}(f) = \bigcap \text{Fix}(f) \)
- \( \text{gfp}(f) = \bigcup \text{Ext}(f) = \bigcup \text{Fix}(f) \)
Kleene’s Fixedpoint Theorem

- Consider a complete lattice \( L = (L, \sqsubseteq, \sqcup, \sqcap, \bot, \top) \) and a continuous function \( f: L \to L \) then

\[
\text{lfp}(f) = \bigsqcup_{n \in \mathbb{N}} f^n(\bot)
\]

- A function \( f \) is continuous if for every increasing chain \( Y \subseteq L \), \( f(\bigsqcup Y) = \bigsqcup \{ f(y) \mid y \in Y \} \)

- Monotone functions on posets satisfying ACC are continuous
  - Every ascending chain eventually stabilizes
    \( l_0 \sqsubseteq l_1 \sqsubseteq \ldots \sqsubseteq l_n = l_{n+1} = \ldots \)
    hence \( l_n \) is the least upper bound of \( \{ l_0, l_1, \ldots, l_n \} \), thus \( f(\bigsqcup Y) = f(l_n) \)
  - From monotonicity of \( f \)
    \( f(l_0) \sqsubseteq f(l_1) \sqsubseteq \ldots \sqsubseteq f(l_n) = f(l_{n+1}) = \ldots \)
    Hence \( f(l_n) \) is the least upper bound of \( \{ f(l_0), f(l_1), \ldots, f(l_n) \} \), thus
    \( \bigsqcup \{ f(y) \mid y \in Y \} = f(l_n) \)
An Algorithm for Computing lfp

- Kleene’s fixed point theorem gives a constructive method for computing the lfp

\[
lfp(f) = \bigsqcup_{n \in \mathbb{N}} f^n(\bot)
\]

[Algorithm]

\[
I = \bot
\]

while \( f(I) \neq I \) do \( I = f(I) \)

- In our case, can compute efficiently using a worklist algorithm for “chaotic iteration”
Finally...
Reaching Definitions...

\( \text{in}(1) = \{(x,?), (y,?), (z,?)\} \)
\( \text{in}(2) = \text{out}(1) \)
\( \text{in}(3) = \text{out}(2) \cup \text{out}(5) \)
\( \text{in}(4) = \text{out}(3) \)
\( \text{in}(5) = \text{out}(4) \)
\( \text{In}(6) = \text{out}(3) \)
\( \text{out}(1) = \text{in}(1) \setminus \{(y,l) \mid l \in \text{Lab}\} \cup \{(y,1)\} \)
\( \text{out}(2) = \text{in}(2) \setminus \{(z,l) \mid l \in \text{Lab}\} \cup \{(z,2)\} \)
\( \text{out}(3) = \text{in}(3) \)
\( \text{out}(4) = \text{in}(4) \setminus \{(z,l) \mid l \in \text{Lab}\} \cup \{(z,4)\} \)
\( \text{out}(5) = \text{in}(5) \setminus \{(y,l) \mid l \in \text{Lab}\} \cup \{(y,5)\} \)
\( \text{out}(6) = \text{in}(6) \setminus \{(y,l) \mid l \in \text{Lab}\} \cup \{(y,6)\} \)

\[ F : (\bigotimes (\text{Var} \times \text{Lab}))^{12} \rightarrow (\bigotimes (\text{Var} \times \text{Lab}))^{12} \]

\[ l = \bot \]
while \( F(l) \neq l \) do \( l = F(l) \)
Algorithm Revisited

- Input: equation system for RD
- Output: lfp(F)
- Algorithm:
  - RD1 = ∅, ..., RD12 = ∅
  - While RDj ≠ Fj(RD1,...,RD12) for some j
    - RDj = Fj(RD1,...,RD12)

- Idea: non-deterministically select which component of RD should make a step
- Can make selection efficiently based on dependencies
  - We have information on what equations depend on Fj that has been updated, need only recompute these on every step
But...

- where did the transfer functions come from?
  - e.g., $\text{out}(1) = \text{in}(1) \setminus \{ (y,l) \mid l \in \text{Lab} \} \cup \{ (y,1) \}$
- how do we know transfer functions really over-approximate the concrete operations?
Trace Semantics

\[
\begin{align*}
[y := x] & \quad [z := 1] \\
[z := 1] & \quad [y := 0] \\
\text{while } [y > 0] & \quad ( [z := z \times y] \\
 & \quad [y := y - 1] ) \\
[y := 0] & \\
\end{align*}
\]

- note that input (x) is unknown
- while loop?
- trace semantics is not computable
Instrumented Trace Semantics

- on every assignment, record the label in which the assignment was performed
Abstract Interpretation

1: y := x;
2: z := 1;
3: while y > 0 {
  4: z := z * y;
  5: y := y − 1
}
6: y := 0

- Instrumented trace semantics
Collecting Semantics

1: \( y := x \)
2: \( z := 1 \)
3: \( y > 0 \)
4: \( z = z \times y \)
5: \( y = y - 1 \)
6: \( y := 0 \)
System of Equations

\[
\begin{align*}
\text{in}(1) &= \{ (x,?) (y,?) (z,?) \} \\
\text{in}(2) &= \text{out}(1) \\
\text{in}(3) &= \text{out}(2) \cup \text{out}(5) \\
\text{in}(4) &= \text{out}(3) \\
\text{in}(5) &= \text{out}(4) \\
\text{in}(6) &= \text{out}(3) \\
\text{out}(1) &= \{ \pi \cdot (y,1) \mid \pi \in \text{in}(1) \} \\
\text{out}(2) &= \{ \pi \cdot (z,2) \mid \pi \in \text{in}(2) \} \\
\text{out}(3) &= \text{in}(3) \\
\text{out}(4) &= \{ \pi \cdot (z,4) \mid \pi \in \text{in}(4) \} \\
\text{out}(5) &= \{ \pi \cdot (y,5) \mid \pi \in \text{in}(5) \} \\
\text{out}(6) &= \{ \pi \cdot (y,6) \mid \pi \in \text{in}(6) \}
\end{align*}
\]

\[\pi \in \text{Trace} = (\text{Var} \times \text{Lab})^*\]

\[G : (\varnothing (\text{Trace}))^{12} \rightarrow (\varnothing (\text{Trace}))^{12}\]
Galois Connection

\[ \alpha(C) \subseteq A \iff C \subseteq \gamma(A) \]
Reaching Definitions: Abstraction

- $\text{DOM}(\pi) = \text{variables in } \pi$
- $\text{SRD}(\pi)(v) = 1$ iff rightmost pair for $v$ in $\pi$ is $(v,1)$
- $\alpha(C) = \{ (v, \text{SRD}(\pi)(v)) \mid v \in \text{DOM}(\pi) \land \pi \in C \}$
- $\gamma(A) = \{ \pi \mid \forall v \in \text{DOM}(\pi) : (v, \text{SRD}(\pi)(v)) \in A \}$

Set of Traces:

- $(x,?) (y,?) (z,?) (y,1) (z,2) (y,6)$
- $(x,?) (y,?) (z,?) (y,1) (z,2) (z,4) (y,5) (y,6)$
- $(x,?) (y,?) (z,?) (y,1) (z,2) (z,4) (y,5) (z,4) (y,5) (y,6)$

Set of (var,lab) pairs:

- $(x,?), (z,2), (y,6)$
- $(x,?), (z,4), (y,6)$
- $(x,?), (z,2), (z,4), (y,6)$
- $(x,?), (z,4), (y,6)$
Reaching Definitions: Concretization

- \( \text{DOM}(\pi) = \text{variables in } \pi \)
- \( \text{SRD}(\pi)(v) = 1 \iff \text{rightmost pair for } v \text{ in } \pi \text{ is } (v,1) \)
- \( \alpha(C) = \{ (v,\text{SRD}(\pi)(v)) \mid v \in \text{DOM}(\pi) \land \pi \in C \} \)
- \( \gamma(A) = \{ \pi \mid \forall v \in \text{DOM}(\pi) : (v,\text{SRD}(\pi)(v)) \in A \} \)
How do we figure out the abstract effect $[S]#$ of a statement $S$?
Sound Abstract Transformer

\[ \alpha \circ [S] \circ \gamma (A) \subseteq [S]^{\#}(A) \]
Soundness of Induced Analysis

\[ \alpha(lfp(G)) \subseteq lfp(\alpha \circ G \circ \gamma) \subseteq lfp(F) \]
Back to a bit of dataflow analysis...
Recap

- Represent properties of a program using a lattice \((L, \sqsubseteq, \sqcup, \sqcap, \bot, \top)\)
- A continuous function \(f: L \rightarrow L\)
  - Monotone function when \(L\) satisfies ACC implies continuous
- Kleene’s fixedpoint theorem
  - \(\text{lfp}(f) = \bigsqcup_{n \in \mathbb{N}} f^n(\bot)\)
- A constructive method for computing the lfp
Some required notation

blocks : Stmt → P(Blocks)
blocks([x := a]lab) = {[x := a]lab}
blocks([skip]lab) = {[skip]lab}
blocks(S₁; S₂) = blocks(S₁) ∪ blocks(S₂)
blocks(if [b]lab then S₁ else S₂) = {[b]lab} ∪ blocks(S₁) ∪ blocks(S₂)
blocks(while [b]lab do S) = {[b]lab} ∪ blocks(S)

FV: (BExp ∪ AExp) → Var
Variables used in an expression

AExp(a) = all non-unit expressions in the arithmetic expression a
similarly AExp(b) for a boolean expression b
Available Expressions Analysis

\[ x := a + b \]
\[ y := a * b \]
while \[ y > a + b \] (  
  \[ a := a + 1 \]  
  \[ x := a + b \]  
)

For each program point, which expressions \textbf{must} have already been computed, and not later modified, on \textbf{all paths} to the program point

\((a+b)\) always available at label 3
Available Expressions Analysis

- Property space
  - $in_{AE}, out_{AE}: \text{Lab} \rightarrow \mathcal{P}(AExp)$
  - Mapping a label to set of arithmetic expressions available at that label

- Dataflow equations
  - Flow equations – how to join incoming dataflow facts
  - Effect equations - given an input set of expressions $S$, what is the effect of a statement
Available Expressions Analysis

- $\text{in}_{AE}(\text{lab}) =$
  - $\emptyset$ when lab is the initial label
  - $\cap \{ \text{out}_{AE}(\text{lab}') | \text{lab}' \in \text{pred}(\text{lab}) \}$ otherwise

- $\text{out}_{AE}(\text{lab}) = ...$

<table>
<thead>
<tr>
<th>Block</th>
<th>out (lab)</th>
</tr>
</thead>
<tbody>
<tr>
<td>$[x := a]^{lab}$</td>
<td>$\text{in}(\text{lab}) \setminus { a' \in \text{AExp}</td>
</tr>
<tr>
<td>$[\text{skip}]^{lab}$</td>
<td>$\text{in}(\text{lab})$</td>
</tr>
<tr>
<td>$[b]^{lab}$</td>
<td>$\text{in}(\text{lab}) \cup \text{AExp}(b)$</td>
</tr>
</tbody>
</table>

From now on going to drop the AE subscript when clear from context
Transfer Functions

1: \( x = a + b \)

- \( \text{out}(1) = \text{in}(1) \cup \{a+b\} \)

2: \( y := a \cdot b \)

- \( \text{out}(2) = \text{in}(2) \cup \{a \cdot b\} \)

3: \( y > a + b \)

- \( \text{out}(3) = \text{in}(3) \cup \{a + b\} \)

4: \( a = a + 1 \)

- \( \text{out}(4) = \text{in}(4) \setminus \{a+b,a \cdot b,a+1\} \)

5: \( x = a + b \)

- \( \text{out}(5) = \text{in}(5) \cup \{a + b\} \)

\[ \begin{align*}
\text{in}(1) &= \emptyset \\
\text{in}(2) &= \text{out}(1) \\
\text{in}(3) &= \text{out}(2) \cap \text{out}(5) \\
\text{in}(4) &= \text{out}(3) \\
\text{in}(5) &= \text{out}(4) \\
\end{align*} \]

\[ \begin{align*}
[x := a+b]^1; \\
[y := a \cdot b]^2; \\
\text{while } [y > a+b]^3 ( \\
\quad [a := a + 1]^4; \\
\quad [x := a + b]^5 \\
) \]
Solution

\[
\text{in}(1) = \emptyset
\]

1: \( x = a+b \)

\[
\text{in}(2) = \text{out}(1) = \{ a + b \}
\]

2: \( y := a \times b \)

\[
\text{out}(2) = \{ a+b, a \times b \} \quad \text{in}(3) = \{ a + b \}
\]

3: \( y > a+b \)

\[
\text{in}(4) = \text{out}(3) = \{ a + b \}
\]

4: \( a = a+1 \)

\[
\text{out}(4) = \emptyset
\]

5: \( x = a+b \)

\[
\text{out}(5) = \{ a+b \}
\]
## Kill/Gen

### Block

<table>
<thead>
<tr>
<th>Block</th>
<th>out (lab)</th>
</tr>
</thead>
<tbody>
<tr>
<td>([x := a]_{\text{lab}})</td>
<td>(\text{in(lab)} \setminus { a' \in \text{AExp} \mid x \in \text{FV}(a') } \cup { a' \in \text{AExp}(a) \mid x \notin \text{FV}(a') } )</td>
</tr>
<tr>
<td>([\text{skip}]_{\text{lab}})</td>
<td>(\text{in(lab)})</td>
</tr>
<tr>
<td>([b]_{\text{lab}})</td>
<td>(\text{in(lab)} \cup \text{AExp}(b))</td>
</tr>
</tbody>
</table>

### Kill/Gen

<table>
<thead>
<tr>
<th>Block</th>
<th>kill</th>
<th>gen</th>
</tr>
</thead>
<tbody>
<tr>
<td>([x := a]_{\text{lab}})</td>
<td>({ a' \in \text{AExp} \mid x \in \text{FV}(a') } )</td>
<td>({ a' \in \text{AExp}(a) \mid x \notin \text{FV}(a') } )</td>
</tr>
<tr>
<td>([\text{skip}]_{\text{lab}})</td>
<td>(\emptyset)</td>
<td>(\emptyset)</td>
</tr>
<tr>
<td>([b]_{\text{lab}})</td>
<td>(\emptyset)</td>
<td>(\text{AExp}(b))</td>
</tr>
</tbody>
</table>

\[
\text{out(lab)} = \text{in(lab)} \setminus \text{kill}(B^{lab}) \cup \text{gen}(B^{lab})
\]

\(B^{lab} = \text{block at label lab}\)
Why solution with largest sets?

1: $z = x + y$
2: true
3: skip

out(1) = in(1) \cup \{x+y\}
in(2) = out(1) \cap out(3)
out(2) = in(2)
in(3) = out(2)
out(3) = in(3)

After simplification: in(2) = in(2) \cap \{x+y\}

Solutions: \{x+y\} or \emptyset
Reaching Definitions Revisited

<table>
<thead>
<tr>
<th>Block</th>
<th>out (lab)</th>
</tr>
</thead>
<tbody>
<tr>
<td>([x := a]_{lab})</td>
<td>(\text{in}(\text{lab}) \setminus {(x, l) \mid l \in \text{Lab}} \cup {(x, \text{lab})})</td>
</tr>
<tr>
<td>([\text{skip}]_{lab})</td>
<td>(\text{in}(\text{lab}))</td>
</tr>
<tr>
<td>([b]_{lab})</td>
<td>(\text{in}(\text{lab}))</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Block</th>
<th>kill</th>
<th>gen</th>
</tr>
</thead>
<tbody>
<tr>
<td>([x := a]_{lab})</td>
<td>({(x, l) \mid l \in \text{Lab}})</td>
<td>({(x, \text{lab})})</td>
</tr>
<tr>
<td>([\text{skip}]_{lab})</td>
<td>(\emptyset)</td>
<td>(\emptyset)</td>
</tr>
<tr>
<td>([b]_{lab})</td>
<td>(\emptyset)</td>
<td>(\emptyset)</td>
</tr>
</tbody>
</table>

For each program point, which assignments may have been made and not overwritten, when program execution reaches this point along some path.
Why solution with smallest sets?

in(1) = \{(x,?), (y,?), (z,?)\}

1: \( z = x+y \)

out(1) = (in(1) \ { (z,?) } ) \cup \{ (z,1) \}

in(2) = out(1) \cup out(3)

2: true

out(2) = in(2)

in(3) = out(2)

3: skip

out(3) = in(3)

After simplification: in(2) = in(2) \cup \{ (x,?), (y,?), (z,1) \}

Many solutions: any superset of \{ (x,?), (y,?), (z,1) \}
Live Variables

[ x :=2]¹;
[y:=4]²;
[x:=1]³;
(if [y>x]⁴ then [z:=y]⁵
else [z:=y*y]⁶);
[x:=z]⁷

For each program point, which variables may be live at the exit from the point.
Live Variables

\[ x := 2 \]
\[ y := 4 \]
\[ x := 1 \]

(if \( y > x \) then \[ z := y \]
else \[ z := y \times y \]);
\[ x := z \]
Live Variables

\[
x := 2
\]
\[
y := 4
\]
\[
x := 1
\]

(if \(y > x\) then \(z := y\) else \(z := y \times y\));

\[
x := z
\]

<table>
<thead>
<tr>
<th>Block</th>
<th>kill</th>
<th>gen</th>
</tr>
</thead>
<tbody>
<tr>
<td>([x := a]^{lab})</td>
<td>{x}</td>
<td>{FV(a)}</td>
</tr>
<tr>
<td>([\text{skip}]^{lab})</td>
<td>\Ø</td>
<td>\Ø</td>
</tr>
<tr>
<td>([b]^{lab})</td>
<td>\Ø</td>
<td>FV(b)</td>
</tr>
</tbody>
</table>

1: \(x := 2\)
2: \(y := 4\)
3: \(x := 1\)
4: \(y > x\)
5: \(z := y\)
6: \(z = y \times y\)
7: \(x := z\)
Live Variables: solution

\[ x := 2^{1}; \]
\[ y := 4^{2}; \]
\[ x := 1^{3}; \]
\( (\text{if } y > x^{4} \text{ then } z := y^{5} \text{ else } z := y \times y^{6}); \)
\[ x := z^{7} \]

\begin{tabular}{|c|c|c|}
\hline
Block & kill & gen \\
\hline
\( [x := a]^{lab} \) & \{ x \} & \{ FV(a) \} \\
\hline
\( [\text{skip}]^{lab} \) & \emptyset & \emptyset \\
\hline
\( [b]^{lab} \) & \emptyset & FV(b) \\
\hline
\end{tabular}
Why solution with smallest set?

After simplification: in(1) = in(1) U \{ x \}

Many solutions: any superset of \{ x \}
Monotone Frameworks

\[
\text{In}(\text{lab}) = \begin{cases} 
\text{Initial} & \text{when } \text{lab} \in \text{Entry labels} \\
\sqcup \{ \text{out}(\text{lab'}) \mid (\text{lab'},\text{lab}) \in \text{CFG edges} \} & \text{otherwise}
\end{cases}
\]

\[
\text{out}(\text{lab}) = f_{\text{lab}}(\text{in}(\text{lab}))
\]

- $\sqcup$ is $\cup$ or $\cap$
- CFG edges go either forward or backwards
- Entry labels are either initial program labels or final program labels (when going backwards)
- Initial is an initial state (or final when going backwards)
- $f_{\text{lab}}$ is the transfer function associated with the blocks $B^{\text{lab}}$
Forward vs. Backward Analyses

1: \( x := 2 \)

\( \{ (x,?), (y,?), (z,?) \} \)

2: \( y := 4 \)

\( \{ (x,1), (y,?), (z,?) \} \)

4: \( y > x \)

\( \{ (x,1), (y,2), (z,?) \} \)

5: \( z := y \)

6: \( z = y*y \)

7: \( x := z \)

\( \{ (x,1), (y,2), (z,?) \} \)
Must vs. May Analyses

- When $\sqcap$ is $\cap$ - must analysis
  - Want largest sets that solve the equation system
  - Properties hold on all paths reaching a label
    (exiting a label, for backwards)

- When $\sqcap$ is $\cup$ - may analysis
  - Want smallest sets that solve the equation system
  - Properties hold at least on one path reaching a label
    (existing a label, for backwards)
Example: Reaching Definition

- \( L = \emptyset (\text{Var} \times \text{Lab}) \) is partially ordered by \( \subseteq \)
- \( \sqcup \) is \( \cup \)
- \( L \) satisfies the Ascending Chain Condition because \( \text{Var} \times \text{Lab} \) is finite (for a given program)
Example: Available Expressions

- \( L = \emptyset (AExp) \) is partially ordered by \( \supseteq \)
- \( \sqcup \) is \( \cap \)
- \( L \) satisfies the Ascending Chain Condition because \( AExp \) is finite (for a given program)
## Analyses Summary

<table>
<thead>
<tr>
<th></th>
<th>Reaching Definitions</th>
<th>Available Expressions</th>
<th>Live Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>( L )</td>
<td>( \wp (\text{Var} \times \text{Lab}) )</td>
<td>( \wp (\text{AExp}) )</td>
<td>( \wp (\text{Var}) )</td>
</tr>
<tr>
<td>( \sqsubseteq )</td>
<td>( \subseteq )</td>
<td>( \supseteq )</td>
<td>( \subseteq )</td>
</tr>
<tr>
<td>( \sqcup )</td>
<td>( \cup )</td>
<td>( \cap )</td>
<td>( \cup )</td>
</tr>
<tr>
<td>( \bot )</td>
<td>( \emptyset )</td>
<td>( \text{AExp} )</td>
<td>( \emptyset )</td>
</tr>
<tr>
<td>Initial</td>
<td>( { (x,?) \mid x \in \text{Var} } )</td>
<td>( \emptyset )</td>
<td>( \emptyset )</td>
</tr>
<tr>
<td>Entry labels</td>
<td>{ init }</td>
<td>{ init }</td>
<td>final</td>
</tr>
<tr>
<td>Direction</td>
<td>Forward</td>
<td>Forward</td>
<td>Backward</td>
</tr>
<tr>
<td>( F )</td>
<td>( { f : L \to L \mid \exists k, g : f(\text{val}) = (\text{val} \setminus k) \cup g } )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( f_{\text{lab}} )</td>
<td>( f_{\text{lab}}(\text{val}) = (\text{val} \setminus \text{kill}) \cup \text{gen} )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Analyses as Monotone Frameworks

- Property space
  - Powerset
  - Clearly a complete lattice

- Transformers
  - Kill/gen form
  - Monotone functions (let’s show it)
Monotonicity of Kill/Gen transformers

- Have to show that \( x \subseteq x' \) implies \( f(x) \subseteq f(x') \)
- Assume \( x \subseteq x' \), then for kill set \( k \) and gen set \( g \)
  \( (x \setminus k) \cup g \subseteq (x' \setminus k) \cup g \)

- Technically, since we want to show it for all functions in \( F \), we also have to show that the set is closed under function composition
Distributivity of Kill/Gen transformers

- Have to show that $f(x \sqcup y) \subseteq f(x) \sqcup f(y)$
- $f(x \sqcup y) = ((x \sqcup y) \setminus k) \cup g$
  = $((x \setminus k) \sqcup (y \setminus k)) \cup g$
  = $(((x \setminus k) \cup g) \sqcup ((y \setminus k) \cup g))$
  = $f(x) \sqcup f(y)$

- Used distributivity of $\sqcup$ and $\cup$
  - Works regardless of whether $\sqcup$ is $\cup$ or $\cap$
Constant Propagation

\[ \sigma \in \text{State} = (\text{Var} \rightarrow Z^T)_\perp \]
Constant Propagation

- $L = ((\text{Var} \rightarrow \mathbb{Z}^T)_\perp, \sqsubseteq)$
- $\sigma_1 \sqsubseteq \sigma_2$ iff $\forall x: \sigma_1(x) \sqsubseteq \sigma_2(x)$
- $\sqsubseteq$ ordering in the $\mathbb{Z}^T_\perp$ lattice

- Examples:
  - $[x \mapsto \perp, y \mapsto 42, z \mapsto \perp] \sqsubseteq [x \mapsto \perp, y \mapsto 42, z \mapsto 73]$
  - $[x \mapsto \perp, y \mapsto 42, z \mapsto 73] \sqsubseteq [x \mapsto \perp, y \mapsto 42, z \mapsto \top]$
Constant Propagation

\[ A: \text{AExp} \rightarrow (\text{State} \rightarrow \mathbb{Z}) \]

\[
A[x]_\sigma = \begin{cases} 
\bot \text{ if } \sigma = \bot \\
\sigma(x) \text{ otherwise}
\end{cases} \quad A[n]_\sigma = \begin{cases} 
\bot \text{ if } \sigma = \bot \\
n \text{ otherwise}
\end{cases}
\]

\[ A[a_1 \text{ op } a_2]_\sigma = A[a_1]_\sigma \text{ op } A[a_2]_\sigma \]

<table>
<thead>
<tr>
<th>Block</th>
<th>out ((\sigma))</th>
</tr>
</thead>
<tbody>
<tr>
<td>([x := a]_{\text{lab}})</td>
<td>(\bot \text{ if } \sigma = \bot)  (\sigma[x \mapsto A[a]_\sigma]) otherwise</td>
</tr>
<tr>
<td>([\text{skip}]_{\text{lab}})</td>
<td>(\sigma)</td>
</tr>
<tr>
<td>([b]_{\text{lab}})</td>
<td>(\sigma)</td>
</tr>
</tbody>
</table>
Example

\[ x := 42 \]
\[ y := 73 \]
\[(\text{if } [?] \text{ then} \]
\[ z := x + y \]
\[ \text{else} \]
\[ z := 12 \]
\[ w := z \]

\[ \begin{align*}
[X & \mapsto \bot, Y \mapsto \bot, Z \mapsto \bot, W \mapsto \bot] \\
[X & \mapsto 42, Y \mapsto \bot, Z \mapsto \bot, W \mapsto \bot] \\
[X & \mapsto 42, Y \mapsto 73, Z \mapsto \bot, W \mapsto \bot] \\
[X & \mapsto 42, Y \mapsto 73, Z \mapsto 115, W \mapsto \bot] \\
[X & \mapsto 42, Y \mapsto 73, Z \mapsto 12, W \mapsto \bot] \\
[X & \mapsto 42, Y \mapsto 73, Z \mapsto \top, W \mapsto 12] \\
[X & \mapsto 42, Y \mapsto 73, Z \mapsto \top, W \mapsto 12]
\end{align*} \]
Constant Propagation is Non Distributive

- Consider the transformer \( f = \lbrack y=x\times x \rbrack \# \)
- Consider two states \( \sigma_1, \sigma_2 \)
  - \( \sigma_1(x) = 1 \)
  - \( \sigma_2(x) = -1 \)

\[
\begin{align*}
(\sigma_1 \sqcup \sigma_2)(x) &= \top \\
f(\sigma_1 \sqcup \sigma_2) &\text{maps } y \text{ to } \top \\
f(\sigma_1) &\text{maps } y \text{ to } 1 \\
f(\sigma_2) &\text{maps } y \text{ to } 1 \\
f(\sigma_1 \sqcup \sigma_2) &\neq f(\sigma_1) \sqcup f(\sigma_2)
\end{align*}
\]