Dataflow Algorithm: Iterative Approach

```
initialize out[s] = gen(s) for all s
do
   set reiterate to false
   for each statement s
      in(s) = ⨆p∈PRED(s) out[p]
      out(s) = f_s(in(s))
      if out(s) has changed
           set reiterate to true
   end for
while reiterate is true
```

Dataflow Algorithm: Worklist Approach

```
initialize out[s] = in[s] = ⊥ for all s
add all statements to worklist
while worklist not empty
    remove s from worklist
    in[s] = ⊔p∈PRED(s) out[p]
    out[s] = f_s(in[s])
    if out[s] has changed
        add successors of s to worklist
    end if
end while
```

Abstraction Example

```
boolean b = mystery();

if(b) {
    x = 1;
    y = 3;
} else {
    x = 3;
    y = 4;
}

z = x + y;
```

Abstraction Example

```
boolean b = mystery();
< b is true or false; >
if(b) {
    x = 1;
    y = 3;
} else {
    x = 3;
    y = 4;
}

z = x + y;
```

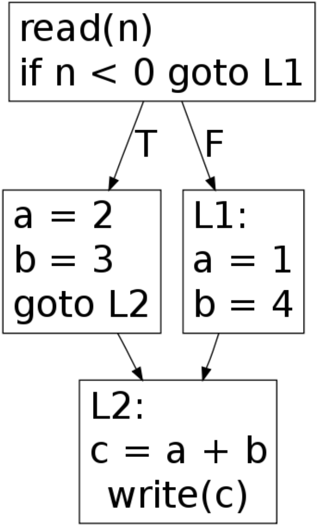
Abstraction Example

```
boolean b = mystery();
< b is true or false; >
if(b) {
    x = 1;
    y = 3;
} else {
    x = 3;
    y = 4;
}
< x is 1 or 3; y is 3 or 4; >
z = x + y;
```

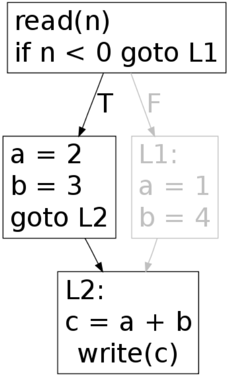
Abstraction Example

```
boolean b = mystery();
< b is true or false; >
if(b) {
    x = 1;
    y = 3;
} else {
    x = 3;
    y = 4;
}
< x is 1 or 3; y is 3 or 4; >
z = x + y;
< z is 4 or 5 or 6 or 7; >
```

Basic Block Graph

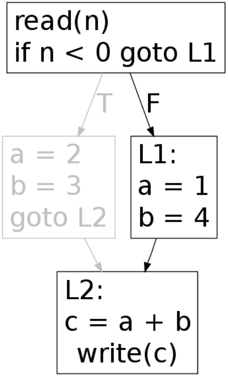


A Path



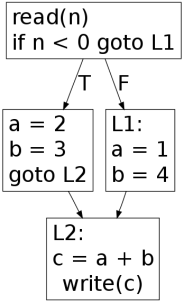
$$f_{\mathrm{write(c)}}(f_{\mathrm{c = a+b}}(f_{\mathrm{b = 3}}(f_{\mathrm{a = 2}}(f_{\mathrm{n < 0}}(f_{\mathrm{read(n)}}(\mathrm{init}))))))$$

Another Path



$$f_{\mathrm{write(c)}}(f_{\mathrm{c=a+b}}(f_{\mathrm{b=4}}(f_{\mathrm{a=1}}(f_{\mathrm{n<0}}(f_{\mathrm{read(n)}}(\mathrm{init}))))))$$

Summarizing Paths



$$f_{\text{write(c)}}\left(f_{\text{c = a+b}}\left(f_{\text{b = 3}}\left(f_{\text{a = 2}}\left(f_{\text{n < 0}}\left(f_{\text{read(n)}}(\text{init})\right)\right)\right)\right)\right)$$

$$\sqcup$$

$$f_{\text{write(c)}}\left(f_{\text{c = a+b}}\left(f_{\text{b = 4}}\left(f_{\text{a = 1}}\left(f_{\text{n < 0}}\left(f_{\text{read(n)}}(\text{init})\right)\right)\right)\right)\right)$$

Definitions

Definition

A partially ordered set (poset) is a set with a binary relation $\sqsubseteq$ that is

- reflexive ($x \sqsubseteq x$),
- transitive ($x \sqsubseteq y \wedge y \sqsubseteq z \implies x \sqsubseteq z$), and
- antisymmetric ($x \sqsubseteq y \wedge y \sqsubseteq x \implies y = x$).

Definitions

Definition

z is an upper bound of x and y if $x \sqsubseteq z$ and $y \sqsubseteq z$.

Definition

z is a least upper bound of x and y if
z is an upper bound of x and y, and
for all upper bounds v of x and y, $z \sqsubseteq v$.

Definition

A lattice is a poset such that for every pair of elements x, y, there exists

- a least upper bound = join = $x \sqcup y$, and
- a greatest lower bound = meet = $x \sqcap y$.

Definitions

Definition

In a complete lattice, $\sqcup$ and $\sqcap$ exist for all (possibly infinite) subsets of elements.

Definition

A bounded lattice contains two elements:

- $\top$ = top such that $\forall x . x \sqsubseteq \top$
- $\perp$ = bottom such that $\forall x . \perp \sqsubseteq x$

Note: all complete lattices are bounded.
Note: all finite lattices are complete.

Definitions

Powerset Lattice

IF F is a set,
THEN the powerset $\mathcal{P}(F)$ with $\sqsubseteq$ defined as $\subseteq$ (or as $\supseteq$) is a lattice.

Definitions

Powerset Lattice

IF F is a set,
THEN the powerset $\mathcal{P}(F)$ with $\sqsubseteq$ defined as $\subseteq$ (or as $\supseteq$) is a lattice.

Product Lattice

IF L_A and L_B are lattices,
THEN their product $L_A \times L_B$ with $\sqsubseteq$ defined as $(a_1, b_1) \sqsubseteq (a_2, b_2)$ if $a_1 \sqsubseteq a_2$ and $b_1 \sqsubseteq b_2$ is also a lattice.

Definitions

Powerset Lattice

IF F is a set,
THEN the powerset $\mathcal{P}(F)$ with $\sqsubseteq$ defined as $\subseteq$ (or as $\supseteq$) is a lattice.

Product Lattice

IF L_A and L_B are lattices,
THEN their product $L_A \times L_B$ with $\sqsubseteq$ defined as $(a_1, b_1) \sqsubseteq (a_2, b_2)$ if $a_1 \sqsubseteq a_2$ and $b_1 \sqsubseteq b_2$ is also a lattice.

Map Lattice

IF F is a set and L is a lattice,
THEN the set of maps $F \rightarrow L$ with $\sqsubseteq$ defined as $m_1 \sqsubseteq m_2$ if $\forall f \in F . m_1(f) \sqsubseteq m_2(f)$ is also a lattice.

Dataflow Framework

- For each statement S in the control-flow graph, define a $f_S : L \to L$.

Dataflow Framework

- For each statement S in the control-flow graph, define a $f_S : L \rightarrow L$.
- For a path $P = S_0 S_1 S_2 \ldots S_n$ through the control-flow graph, define $f_P(x) = f_n(\ldots f_2(f_1(f_0(x))))$.

Dataflow Framework

- For each statement S in the control-flow graph, define a $f_S : L \to L$.
- For a path $P = S_0 S_1 S_2 \ldots S_n$ through the control-flow graph, define $f_P(x) = f_n(\ldots f_2(f_1(f_0(x))))$.
- Goal: find the join-over-all-paths (MOP):

$$\mathrm{MOP}(n, x) = \bigsqcup_{P \text{ is path from } S_0 \text{ to } S_n} f_P(x)$$

Dataflow Framework

- For each statement S in the control-flow graph, define a $f_S : L \to L$.
- For a path $P = S_0 S_1 S_2 \ldots S_n$ through the control-flow graph, define $f_P(x) = f_n(\ldots f_2(f_1(f_0(x))))$.
- Goal: find the join-over-all-paths (MOP):

$$\mathrm{MOP}(n, x) = \bigsqcup_{P \text{ is path from } S_0 \text{ to } S_n} f_P(x)$$

This is undecidable in general. [Kam, Ullman 1977]

Dataflow Framework

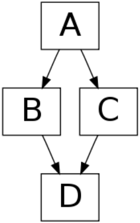
- For each statement S in the control-flow graph, choose a $f_S : L \to L$.
- Goal: For each statement S in the control-flow graph, find $V_{Sin} \in L$ and $V_{Sout} \in L$ satisfying:

$$V_{Sout} = f_S(V_{Sin})$$

$$V_{Sin} = \bigsqcup_{P \in PRED(S)} V_{Pout}$$

- Property: $\mathrm{MOP}(n, x) \sqsubseteq \mathrm{LFP}(n, x)$

MOP vs. fixed point



$$
\begin{aligned}
\text{MOP} &= f_{\text{D}}(f_{\text{B}}(f_{\text{A}}(\text{init}))) \;\sqcup\; f_{\text{D}}(f_{\text{C}}(f_{\text{A}}(\text{init}))) \\
V_{Bout} &= f_{\text{B}}(f_{\text{A}}(\text{init})) \\
V_{Cout} &= f_{\text{C}}(f_{\text{A}}(\text{init})) \\
V_{Din} &= f_{\text{B}}(f_{\text{A}}(\text{init})) \;\sqcup\; f_{\text{C}}(f_{\text{A}}(\text{init})) \\
V_{Dout} &= f_{\text{D}}\left(f_{\text{B}}(f_{\text{A}}(\text{init})) \;\sqcup\; f_{\text{C}}(f_{\text{A}}(\text{init}))\right)
\end{aligned}
$$

Fixed Points

Fixed Point

x is a fixed point of F if $F(x) = x$.

Fixed Points

Fixed Point

x is a fixed point of F if $F(x) = x$.

Monotone Function

A function $f : L_A \to L_B$ is monotone if
$x \sqsubseteq y \implies f(x) \sqsubseteq f(y)$.

Fixed Points

Fixed Point

x is a fixed point of F if $F(x) = x$.

Monotone Function

A function $f : L_A \to L_B$ is monotone if
$x \sqsubseteq y \implies f(x) \sqsubseteq f(y)$.

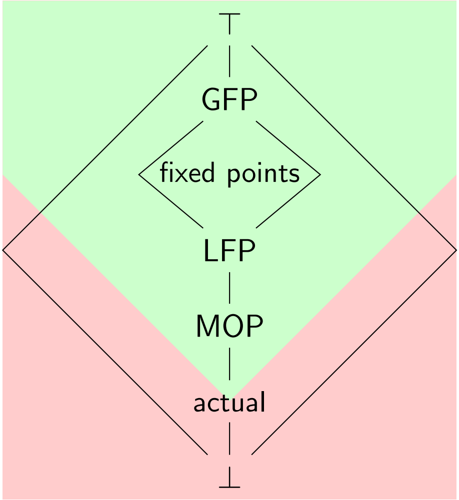
Knaster-Tarski Fixed Point Theorem

IF L is a complete lattice and $f : L \to L$ is monotone,
THEN the set of fixed points of f is a complete sub-lattice.

$$\bigsqcup_{n \geq 0} f^{(n)}(\perp)$$

is the least fixed point of L (i.e. the $\perp$ of the sub-lattice of fixed points).

MOP $\sqsubseteq$ LFP



- Every solution $S \sqsupseteq$ actual is sound.
- MOP $\sqsupseteq$ actual
- LFP $\sqsupseteq$ MOP
- Distributive flow function $\implies$ LFP $=$ MOP

Distributivity

Monotone Function

A function $f : L_A \to L_B$ is monotone if
$x \sqsubseteq y \implies f(x) \sqsubseteq f(y)$.

Theorem

IF f is monotone,
THEN $f(x) \sqcup f(y) \sqsubseteq f(x \sqcup y)$.

Distributive Function

A function $f : L_A \to L_B$ is distributive if
$f(x) \sqcup f(y) = f(x \sqcup y)$.

Sketch of Dataflow Algorithm

1. Define a big product lattice

$$\mathcal{L} = \prod_{s \in \text{statements}} L_{s\ in} \times L_{s\ out}$$

2. Define a big function

$$\mathcal{F} : \mathcal{L} \to \mathcal{L}$$

$$\mathcal{F}(V_{s_1 in}, V_{s_1 out}, \ldots) = \left(\bigsqcup_{p \in PRED(s_1)} V_{p\ out}, f_{s_1}(V_{s_1 in}), \ldots \right)$$

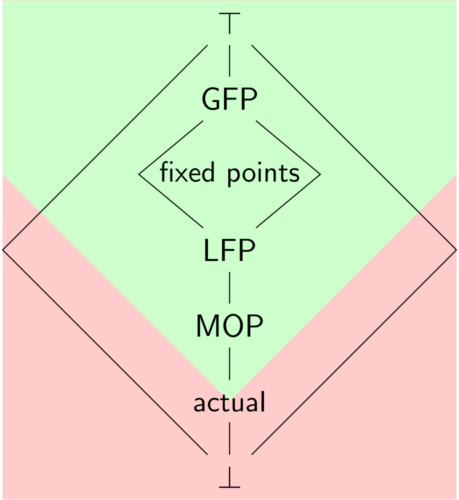
3. Iteratively compute least fixed point

$$\bigsqcup_{n \geq 0} \mathcal{F}^{(n)}(\perp)$$

Designing a Dataflow Analysis

1. Forwards or backwards?
2. What are the lattice elements?
3. Must the property hold on all paths, or must there exist a path?
   (What is the join operator?)
4. On a given path, what are we trying to compute? What are the flow equations?
5. What values hold for program entry points?
6. (What is the initial estimate?)
   It's the unique element $\perp$ such that $\forall x. \perp \sqcup x = x$.

Pessimistic vs. Optimistic Analysis



$$\mathrm{LFP} = \bigsqcup_{n \geq 0} \mathcal{F}^{(n)}(\perp)$$

$$\mathrm{GFP} = \bigsqcap_{n \geq 0} \mathcal{F}^{(n)}(\top)$$

If we start from $\top$ instead of $\perp$, we can stop early before reaching the fixed point, but we may get an imprecise result.