Dataflow Algorithm: Iterative Approach

initialize out[s] = gen(s) for all s
do
    set reiterate to false
    for each statement s
        in(s) = \bigcup_{p \in \text{PRED}(s)} \text{out}[p]
        out(s) = f_s(in(s))
        if out(s) has changed
            set reiterate to true
    end for
while reiterate is true
initialize $out[s] = in[s] = \bot$ for all $s$

add all statements to worklist

while worklist not empty
  remove $s$ from worklist
  $in[s] = \bigcup_{p \in \text{PRED}(s)} out[p]$
  $out[s] = f_s(in[s])$
  if $out[s]$ has changed
    add successors of $s$ to worklist
  end if
end while
boolean b = mystery();

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

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

z = x + y;
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

```java
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

read(n)
if n < 0 goto L1

T
a = 2
b = 3
goto L2

F
L1:
a = 1
b = 4

L2:
c = a + b
write(c)
A Path

\[ f_{\text{write}(c)}(f_c = a+b)(f_b = 3)(f_a = 2)(f_n < 0)(f_{\text{read}(n)}(\text{init}))) \]
Another Path

\[ f_{\text{write}(c)}(f_c = a + b( f_b = 4( f_a = 1( f_n < 0( f_{\text{read}(n)}(\text{init}))))))) \]
Summarizing Paths

\[ f_{\text{write}}(c) \left( f_c = a + b \left( f_b = 3 \left( f_a = 2 \left( f_n < 0 \left( f_{\text{read}}(n)(\text{init}) \right) \right) \right) \right) \right) \]

\[ \square \]

\[ f_{\text{write}}(c) \left( f_c = a + b \left( f_b = 4 \left( f_a = 1 \left( f_n < 0 \left( f_{\text{read}}(n)(\text{init}) \right) \right) \right) \right) \right) \]
A partially ordered set (poset) is a set with a binary relation \( \sqsubseteq \) that is

- reflexive \( (x \sqsubseteq x) \),
- transitive \( (x \sqsubseteq y \land y \sqsubseteq z \implies x \sqsubseteq z) \), and
- antisymmetric \( (x \sqsubseteq y \land y \sqsubseteq x \implies y = x) \).
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 $= \text{join} = x \sqcup y$, and
- a greatest lower bound $= \text{meet} = x \sqcap y$. 
Definitions

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

A **bounded** lattice contains two elements:

- $\top =$ top such that $\forall x. x \subseteq \top$
- $\bot =$ bottom such that $\forall x. \bot \subseteq x$

Note: all complete lattices are bounded.

Note: all finite lattices are complete.
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 $F \to L$ 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.
For each statement $S$ in the control-flow graph, define a $f_S : L \rightarrow L$. 
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):

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

This is undecidable in general. [Kam, Ullman 1977]
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 $f_P(x) = f_n(\ldots f_2(f_1(f_0(x))))$.

Goal: find the join-over-all-paths (MOP):

$$\text{MOP}(n, x) = \bigcup_{P \text{ is path from } S_0 \text{ to } S_n} f_P(x)$$
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 $f_P(x) = f_n(\ldots f_2(f_1(f_0(x))))$.

Goal: find the join-over-all-paths (MOP):

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

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: $\text{MOP}(n, x) \subseteq \text{LFP}(n, x)$
MOP vs. fixed point

\[ \text{MOP} = f_D(f_B(f_A(\text{init}))) \sqcup f_D(f_C(f_A(\text{init}))) \]

\[ V_{Bout} = f_B(f_A(\text{init})) \]

\[ V_{Cout} = f_C(f_A(\text{init})) \]

\[ V_{Din} = f_B(f_A(\text{init})) \sqcup f_C(f_A(\text{init})) \]

\[ V_{Dout} = f_D(f_B(f_A(\text{init}))) \sqcup f_C(f_A(\text{init}))) \]
**Fixed Point**

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

---

**Monotone Function**

A function $f: L \rightarrow L$ is monotone if $x \sqsubseteq y \implies f(x) \sqsubseteq f(y)$.

**Knaster-Tarski Fixed Point Theorem**

If $L$ is a complete lattice and $f: L \rightarrow L$ is monotone, then the set of fixed points of $f$ is a complete sub-lattice.

$\bigoplus_{n \geq 0} f^{(n)}(\bot)$ is the least fixed point of $L$ (i.e., the $\bot$ of the sub-lattice of fixed points).
Fixed Point

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

Monotone Function

A function $f : L_A \rightarrow L_B$ is **monotone** if $x \sqsubseteq y \implies f(x) \sqsubseteq f(y)$.
Fixed Point

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

Monotone Function

A function $f : L_A \rightarrow L_B$ is monotone if

$x \sqsubseteq y \implies f(x) \sqsubseteq f(y)$.

Knaster-Tarski Fixed Point Theorem

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

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

is the least fixed point of $L$ (i.e. the $\bot$ of the sub-lattice of fixed points).
Every solution $S \sqsubseteq \text{actual}$ is sound.

- $\text{MOP} \sqsubseteq \text{actual}$
- $\text{LFP} \sqsubseteq \text{MOP}$
- Distributive flow function $\implies \text{LFP} = \text{MOP}$
Distributivity

Monotone Function

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

Theorem

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

Distributive Function

A function $f : L_A \rightarrow L_B$ is distributive if $f(x) \sqcup f(y) = f(x \sqcup y)$.
Sketch of Dataflow Algorithm

1. Define a big product lattice

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

2. Define a big function

\[ \mathcal{F} : \mathcal{L} \to \mathcal{L} \]

\[ \mathcal{F}(V_{s1\text{in}}, V_{s1\text{out}}, \ldots) = \left( \bigsqcup_{p \in PRED(s_1)} V_{p \text{ out}}, f_{s_1}(V_{s1\text{in}}), \ldots \right) \]

3. Iteratively compute least fixed point

\[ \bigsqcup_{n \geq 0} \mathcal{F}^{(n)}(\bot) \]
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 \( \bot \) such that \( \forall x. \bot \sqcup x = x \).
Pessimistic vs. Optimistic Analysis

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

\[
LFP = \bigsqcup_{n \geq 0} \mathcal{F}(n)(\bot)
\]

\[
GFP = \bigsqcap_{n \geq 0} \mathcal{F}(n)(\top)
\]