Flow-Sensitive Pointer Analysis
flow-sensitive analysis

• Basic data-flow analysis

• Track points-to graph at each point in the program
  - Graphs ordered according to subset inclusion on edges of the graph

• All that’s left: what are the transfer functions?
notation: points-to sets

• Can think of a points-to graph as a set of **points-to sets**

• $pt(x) =$ the set of nodes that $x$ points to $=$ the targets of edges that have $x$ as a source

• $pt(ptr) = \{x, y\}$

• $pt(x) = \{z\}$
Suppose \( S \) and \( S' \) are set-valued variables: \( S = \{x, y\} \) and \( S' = \{x, z\} \)

- \( S' \leftarrow S \) **strong update**
  - \( S' \) has a new value of whatever is in \( S \)
  - \( S' = \{x, y\} \)

- \( S' \cup S \) **weak update**
  - Add whatever is in \( S \) to \( S' \)
  - \( S' = \{x, y, z\} \)
dataflow equations

- Forward analysis (points-to information is about what has already happened)
- Use $\sqcup$ at merges (points-to information is may information)
- Transfer functions: $G$ is graph before statement, $G'$ is graph after
dataflow equations

• Forward analysis (points-to information is about what has already happened)

• Use △ at merges (points-to information is may information)

• Transfer functions: $G$ is graph before statement, $G'$ is graph after

\[ G' = G \text{ with } pt(x) \leftarrow \{y\} \]
dataflow equations

- Forward analysis (points-to information is about what has already happened)
- Use $\sqcup$ at merges (points-to information is $may$ information)
- Transfer functions: $G$ is graph before statement, $G'$ is graph after

$\quad G' = G$ with $pt(x) \leftarrow pt(y)$

\[ \text{copy} \quad x = y \]
dataflow equations

- Forward analysis (points-to information is about what has already happened)
- Use $\sqcup$ at merges (points-to information is may information)
- Transfer functions: $G$ is graph before statement, $G'$ is graph after

$$G' = G \text{ with } pt(x) \leftarrow \bigcup_{a \in pt(y)} pt(a)$$

load
$x = * y$

Diagram:

```
X -> Y -> W -> B
X -> Y -> Z -> A
X -> Z -> A
X -> W -> B
```
**dataflow equations**

- Forward analysis (points-to information is about what has already happened)

- Use $\sqcup$ at merges (points-to information is *may* information)

- Transfer functions: $G$ is graph before statement, $G'$ is graph after

\[
G' = G \text{ with } \forall a \in pt(x). pt(a) \leftarrow pt(y)
\]
dataflow equations

• Forward analysis (points-to information is about what has already happened)

• Use \( \sqcup \) at merges (points-to information is may information)

• Transfer functions: \( G \) is graph before statement, \( G' \) is graph after

\[
\begin{align*}
G' &= G \text{ with } \forall a \in \text{pt}(x). pt(a) \sqcup \leftarrow pt(y) \\
\end{align*}
\]

**store**

\[ ^* x = y \]

Weak update! why?
weak vs. strong updates

• Strong update
  • At an assignment, you know what variables are being written to
  • Can remove points-to information coming in to the statement

• Weak update
  • *x = … means “whatever x points to should be updated”
  • At runtime, only one variable is written to, but at analysis time, we don’t know which one
  • Each variable may be written to, but we cannot safely remove any information
loads and stores as paths

- One way to keep this straight is to think of loads and stores as multiple paths through the program, one path per thing the pointer points to.

```
load
x = *y

//pts(y) = {a, b}
if (…)
  x = a
else
  x = b
```

```
store
*x = y

//pts(x) = {a, b}
if (…)
  a = y
else
  b = y
```
next: flow-insensitive pointer analysis