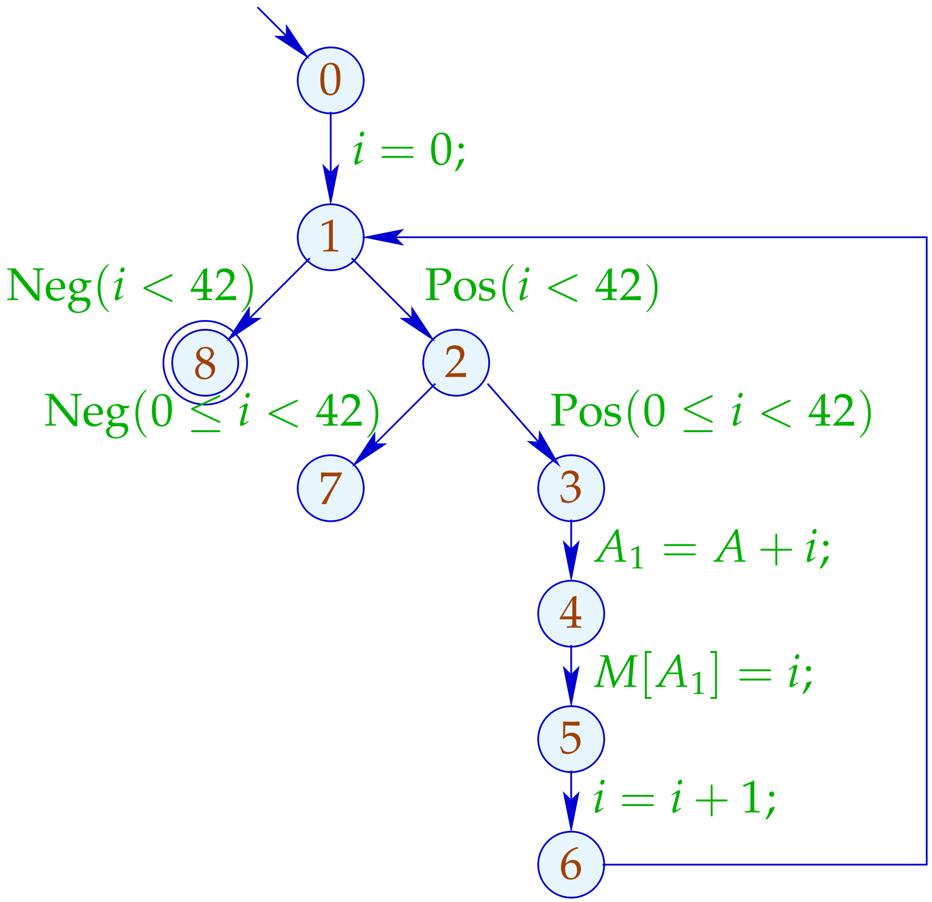
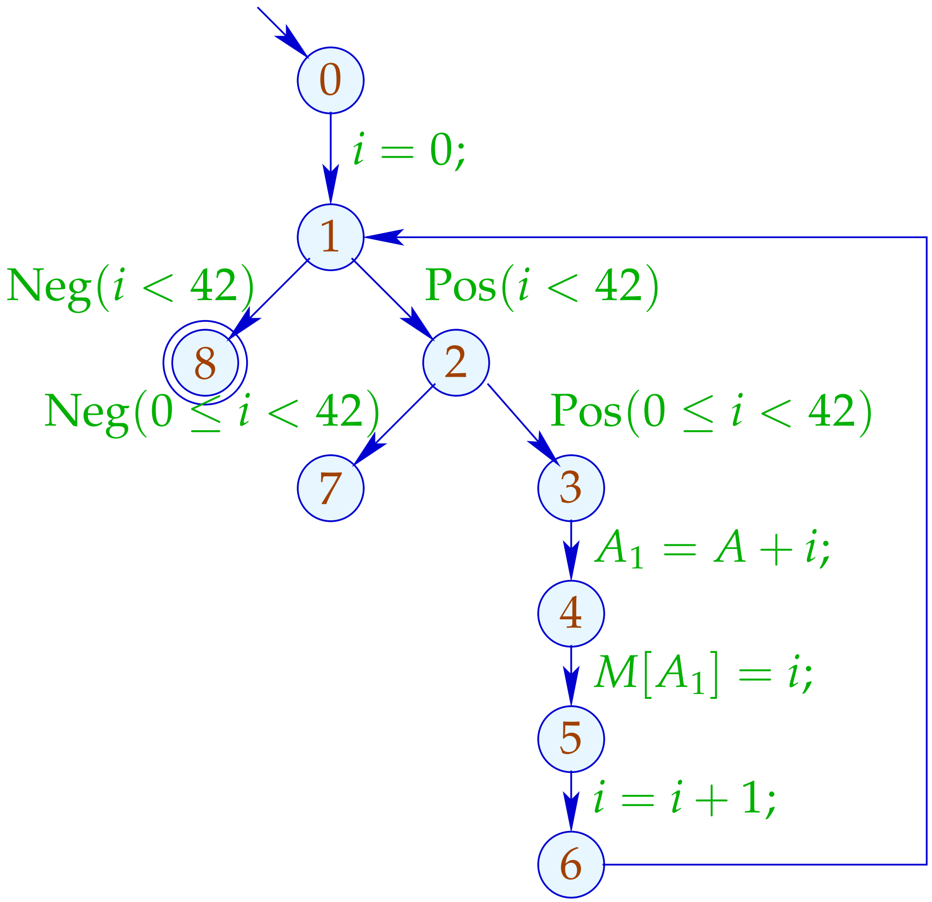
Narrowing Iteration in the Example:



	0		1	
	l	u	l	u
0	$-\infty$	$+\infty$	$-\infty$	$+\infty$
1	0	$+\infty$	0	$+\infty$
2	0	$+\infty$	0	41
3	0	$+\infty$	0	41
4	0	$+\infty$	0	41
5	0	$+\infty$	0	41
6	1	$+\infty$	1	42
7	42	$+\infty$	$\perp$	
8	42	$+\infty$	42	$+\infty$

Narrowing Iteration in the Example:



	0		1		2	
	l	u	l	u	l	u
0	$-\infty$	$+\infty$	$-\infty$	$+\infty$	$-\infty$	$+\infty$
1	0	$+\infty$	0	$+\infty$	0	42
2	0	$+\infty$	0	41	0	41
3	0	$+\infty$	0	41	0	41
4	0	$+\infty$	0	41	0	41
5	0	$+\infty$	0	41	0	41
6	1	$+\infty$	1	42	1	42
7	42	$+\infty$	$\perp$		$\perp$	
8	42	$+\infty$	42	$+\infty$	42	42

Discussion:

- $\longrightarrow$ We start with a safe approximation.
- $\longrightarrow$ We find that the inner check is redundant :-)
- $\longrightarrow$ We find that at exit from the loop, always $i = 42$:-))
- $\longrightarrow$ It was not necessary to construct an optimal loop separator :-)))

Last Question:

Do we have to accept that narrowing may not terminate ???

4. Idea: Accelerated Narrowing

Assume that we have a solution $\underline{x} = (x_1, \ldots, x_n)$ of the system of constraints:

$$x_i \sqsupseteq f_i(x_1, \ldots, x_n) \ , \quad i = 1, \ldots, n \tag{1}$$

Then consider the system of equations:

$$x_i = x_i \sqcap f_i(x_1, \ldots, x_n) \ , \quad i = 1, \ldots, n \tag{4}$$

Obviously, we have for monotonic f_i : $H^k \underline{x} = F^k \underline{x}$:-)

where $H(x_1, \ldots, x_n) = (y_1, \ldots, y_n) \ , \quad y_i = x_i \sqcap f_i(x_1, \ldots, x_n)$.

In (4) , we replace $\sqcap$ durch by the novel operator $\bar{\sqcap}$ where:

$$a_1 \sqcap a_2 \sqsubseteq a_1 \bar{\sqcap} a_2 \sqsubseteq a_1$$

... for Interval Analysis:

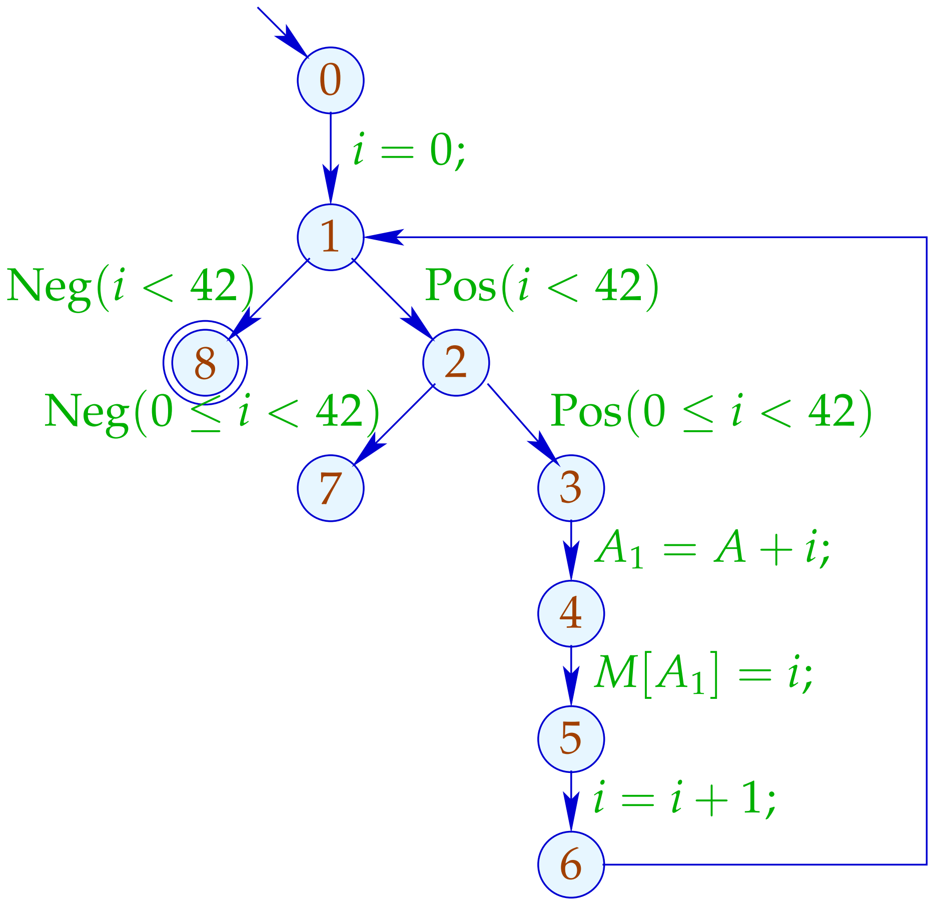
We preserve finite interval bounds :-)

Therefore, $\perp \sqcap\!\!\!\!\!\!\!\!\;\;\; D \;=\; D \sqcap \perp \;=\; \perp$ and for $D_1 \neq \perp \neq D_2$:

$$
\begin{array}{rcl}
(D_1 \sqcap D_2)\, x & = & (D_1\, x) \sqcap (D_2\, x) \qquad \text{where} \\
[l_1, u_1] \sqcap [l_2, u_2] & = & [l, u] \qquad \text{mit} \\
l & = & \begin{cases} l_2 & \text{if} \quad l_1 = -\infty \\ l_1 & \text{otherwise} \end{cases} \\
u & = & \begin{cases} u_2 & \text{if} \quad u_1 = \infty \\ u_1 & \text{otherwise} \end{cases}
\end{array}
$$

$\Longrightarrow$ $\sqcap$ is not commutative !!!

Accelerated Narrowing in the Example:



	0		1		2	
	l	u	l	u	l	u
0	$-\infty$	$+\infty$	$-\infty$	$+\infty$	$-\infty$	$+\infty$
1	0	$+\infty$	0	$+\infty$	0	42
2	0	$+\infty$	0	41	0	41
3	0	$+\infty$	0	41	0	41
4	0	$+\infty$	0	41	0	41
5	0	$+\infty$	0	41	0	41
6	1	$+\infty$	1	42	1	42
7	42	$+\infty$	$\perp$		$\perp$	
8	42	$+\infty$	42	$+\infty$	42	42

Discussion:

- $\longrightarrow$ Warning: Widening also returns for non-monotonic f_i a solution. Narrowing is only applicable to monotonic f_i !!
- $\longrightarrow$ In the example, accelerated narrowing already returns the optimal result :-)
- $\longrightarrow$ If the operator $\sqcap\!\!\!\!\!\!\;\,\overline{}$ only allows for finitely many improvements of values, we may execute narrowing until stabilization.
- $\longrightarrow$ In case of interval analysis these are at most:

$$\#\mathit{points} \cdot (1 + 2 \cdot \#\mathit{Vars})$$

1.6 Pointer Analysis

Questions:

→ Are two addresses possibly equal?

→ Are two addresses definitively equal?

1.6 Pointer Analysis

Questions:

$\rightarrow$ Are two addresses possibly equal? May Alias

$\rightarrow$ Are two addresses definitively equal? Must Alias

$\Longrightarrow$ Alias Analysis

The analyses so far **without** alias information:

(1) Available Expressions:

- Extend the set $Expr$ of expressions by occurring loads $M[e]$.

- Extend the Effects of Edges:

$$
\begin{array}{lll}
[\![x = e;]\!]^\sharp\, A & = & (A \cup \{e\})\backslash Expr_x \\
[\![x = M[e];]\!]^\sharp\, A & = & (A \cup \{e, M[e]\})\backslash Expr_x \\
[\![M[e_1] = e_2;]\!]^\sharp\, A & = & (A \cup \{e_1, e_2\})\backslash Loads
\end{array}
$$

(2) Values of Variables:

- Extend the set $Expr$ of expressions by occurring loads $M[e]$.

- Extend the Effects of Edges:

$$[\![x = M[e];]\!]^\sharp\, V\, e' = \begin{cases} \{x\} & \text{if} \quad e' = M[e] \\ \emptyset & \text{if} \quad e' = e \\ V\, e' \backslash \{x\} & \text{otherwise} \end{cases}$$

$$[\![M[e_1] = e_2;]\!]^\sharp\, V\, e' = \begin{cases} \emptyset & \text{if} \quad e' \in \{e_1, e_2\} \\ V\, e' & \text{otherwise} \end{cases}$$

(3) Constant Propagation:

- Extend the abstract state by an abstract store M
- Execute accesses to known memory locations!

$$[\![x = M[e];]\!]^\sharp\,(D, M) \;=\; \begin{cases} (D \oplus \{x \mapsto M\,a\}, M) & \text{if} \\ & [\![e]\!]^\sharp\,D = a \sqsubset \top \\ (D \oplus \{x \mapsto \top\}, M) & \text{otherwise} \end{cases}$$

$$[\![M[e_1] = e_2;]\!]^\sharp\,(D, M) \;=\; \begin{cases} (D, M \oplus \{a \mapsto [\![e_2]\!]^\sharp D\}) & \text{if} \\ & [\![e_1]\!]^\sharp\,D = a \sqsubset \top \\ (D, \underline{\top}) & \text{otherwise} \qquad \text{where} \end{cases}$$

$$\underline{\top}\,a \;=\; \top \qquad\qquad (a \in \mathbb{N})$$

Problems:

- Addresses are from $\mathbb{N}$:-(

  There are no infinite strictly ascending chains, but ...
- Exact addresses at compile-time are rarely known :-(
- At the same program point, typically different addresses are accessed ...
- Storing at an unknown address destroys all information M :-(

$\Longrightarrow$ constant propagation fails :-(

$\Longrightarrow$ memory accesses/pointers kill precision :-(

Simplification:

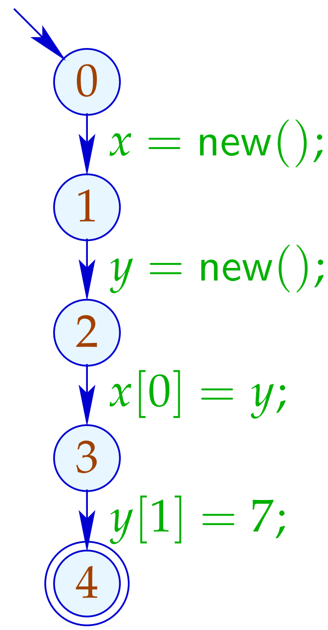
- We consider pointers to the beginning of blocks A which allow indexed accesses $A[i]$:-)
- We ignore well-typedness of the blocks.
- New statements:

$$x = \mathsf{new}(); \quad // \quad \text{allocation of a new block}$$
$$x = y[e]; \quad // \quad \text{indexed read access to a block}$$
$$y[e_1] = e_2; \quad // \quad \text{indexed write access to a block}$$

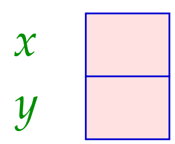
- Blocks are possibly infinite :-)
- For simplicity, all pointers point to the beginning of a block.

Simple Example:

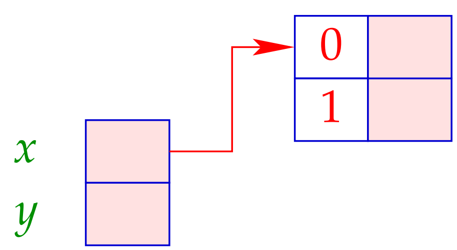
$$x = \mathsf{new}();$$
$$y = \mathsf{new}();$$
$$x[0] = y;$$
$$y[1] = 7;$$



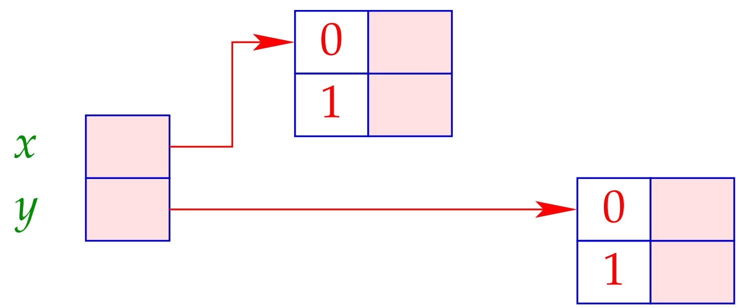
The Semantics:



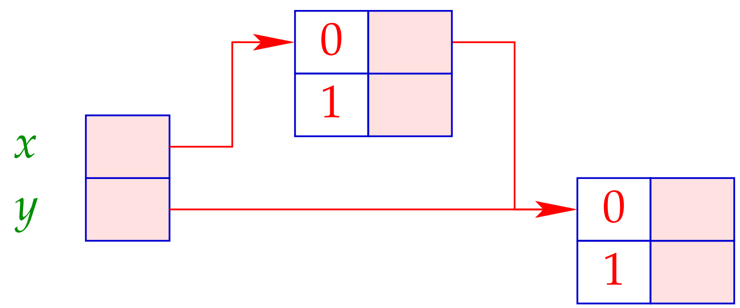
The Semantics:



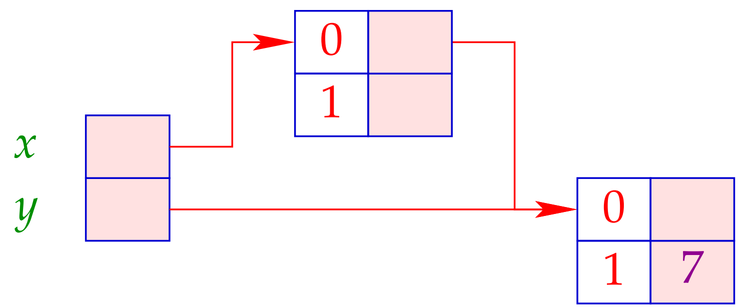
The Semantics:



The Semantics:



The Semantics:



More Complex Example:

$r = \text{Null};$

$\text{while } (t \neq \text{Null}) \ \{$

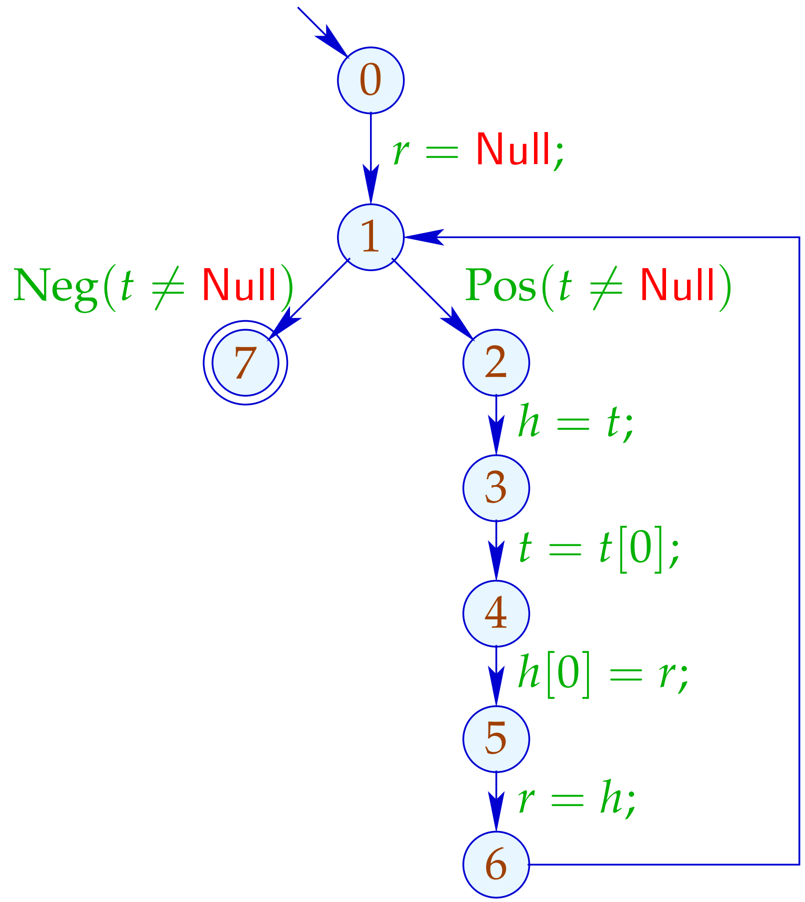
$\quad h = t;$

$\quad t = t[0];$

$\quad h[0] = r;$

$\quad r = h;$

$\}$



Concrete Semantics:

A store consists of a finite collection of blocks.

After h new-operations we obtain:

$$\begin{array}{lcll}
\mathit{Addr}_h & = & \{\mathsf{ref}\ a \mid 0 \leq a < h\} & // \quad \text{addresses} \\
\mathit{Val}_h & = & \mathit{Addr}_h \cup \mathbb{Z} & // \quad \text{values} \\
\mathit{Store}_h & = & (\mathit{Addr}_h \times \mathbb{N}_0) \to \mathit{Val}_h & // \quad \text{store} \\
\mathit{State}_h & = & (\mathit{Vars} \to \mathit{Val}_h) \times \mathit{Store}_h & // \quad \text{states}
\end{array}$$

For simplicity, we set: $\quad 0 = \mathsf{Null}$

Let $\quad (\rho, \mu) \in \mathit{State}_h$. Then we obtain for the new edges:

$$
\begin{array}{lcl}
[\![x = \mathsf{new}();]\!]\ (\rho, \mu) & = & (\rho \oplus \{x \mapsto \mathsf{ref}\ h\}, \\
 & & \ \mu \oplus \{(\mathsf{ref}\ h, i) \mapsto 0, (i \in \mathbb{N}_0) \\
[\![x = y[e];]\!]\ (\rho, \mu) & = & (\rho \oplus \{x \mapsto \mu\,(\rho\, y, [\![e]\!]\, \rho)\}, \mu) \\
[\![y[e_1] = e_2;]\!]\ (\rho, \mu) & = & (\rho, \mu \oplus \{(\rho\, y, [\![e_1]\!]\, \rho) \mapsto \rho\, [\![e_2]\!]\, \rho\})
\end{array}
$$

Warning:

This semantics is too detailled in that it computes with absolute Addresses. Accordingly, the two programs:

$$x = \mathsf{new}(); \qquad\qquad y = \mathsf{new}();$$
$$y = \mathsf{new}(); \qquad\qquad x = \mathsf{new}();$$

are not considered as equivalent !!?

Possible Solution:

Define equivalence only up to permutation of addresses :-)

Alias Analysis 1. Idea:

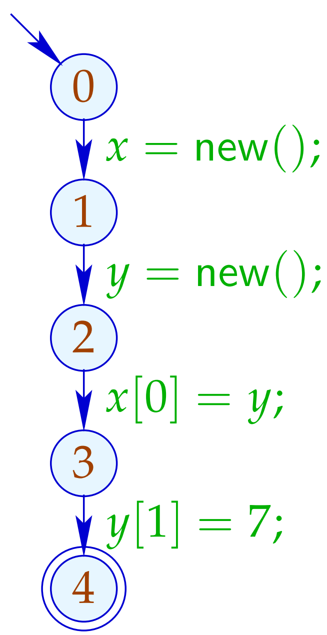
- Distinguish finitely many classes of blocks.
- Collect all addresses of a block into one set!
- Use sets of addresses as abstract values!

$\Longrightarrow$ Points-to-Analysis

$$
\begin{array}{lcll}
\mathit{Addr}^\sharp & = & \mathit{Edges} & \text{// creation edges} \\
\mathit{Val}^\sharp & = & 2^{\mathit{Addr}^\sharp} & \text{// abstract values} \\
\mathit{Store}^\sharp & = & \mathit{Addr}^\sharp \to \mathit{Val}^\sharp & \text{// abstract store} \\
\mathit{State}^\sharp & = & (\mathit{Vars} \to \mathit{Val}^\sharp) \times \mathit{Store}^\sharp & \text{// abstract states}
\end{array}
$$

// complete lattice !!!

... in the Simple Example:



	x	y	$(0,1)$
0	$\emptyset$	$\emptyset$	$\emptyset$
1	$\{(0,1)\}$	$\emptyset$	$\emptyset$
2	$\{(0,1)\}$	$\{(1,2)\}$	$\emptyset$
3	$\{(0,1)\}$	$\{(1,2)\}$	$\{(1,2)\}$
4	$\{(0,1)\}$	$\{(1,2)\}$	$\{(1,2)\}$