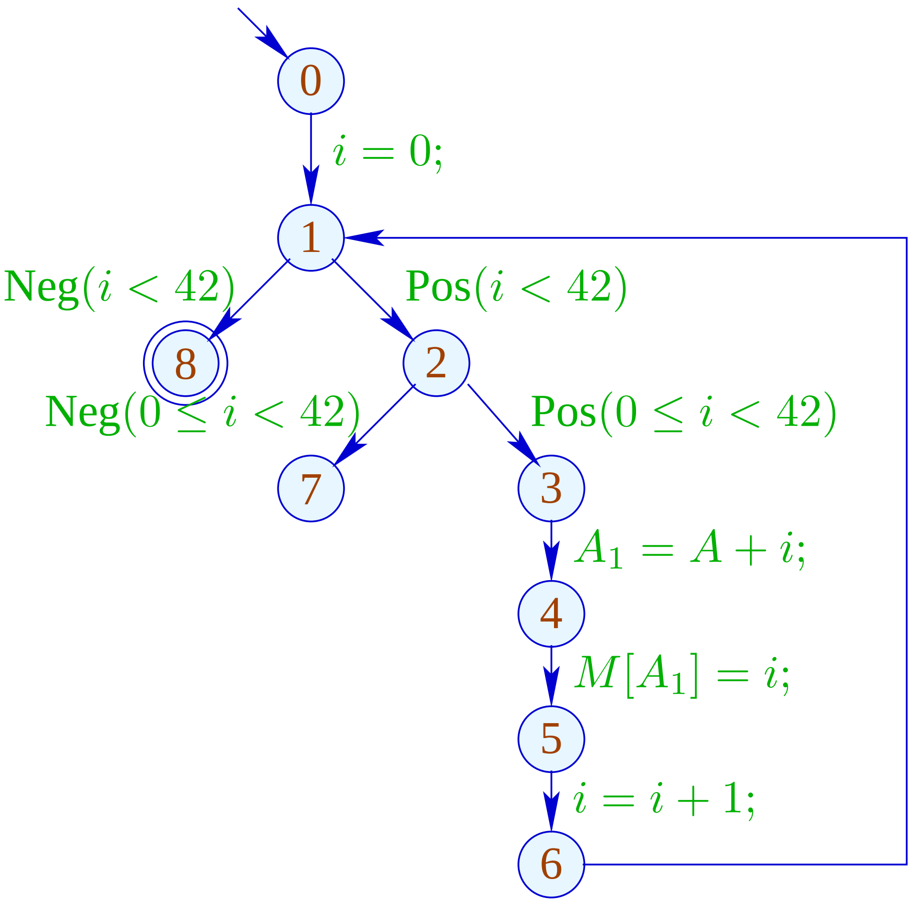
Accelerated Narrowing in the Example:



	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:

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

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

1.6 Pointer Analysis

Questions:

- $\rightarrow$ Are two addresses possibly equal?
- $\rightarrow$ Are two addresses definitively equal?

1.6 Pointer Analysis

Questions:

→ Are two addresses possibly equal? May Alias

→ 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:

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

(2) Values of Variables:

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

- Extend the Effects of Edges:

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

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

(3) Constant Propagation:

- Extend the abstract state by an abstract store M

- Execute accesses to known memory locations!

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

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

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

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:

$$x = \mathsf{new}(); \quad // \text{ allocation of a new block}$$

$$x = y[e]; \quad // \text{ indexed read access to a block}$$

$$y[e_1] = e_2; \quad // \text{ indexed write access to a block}$$

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

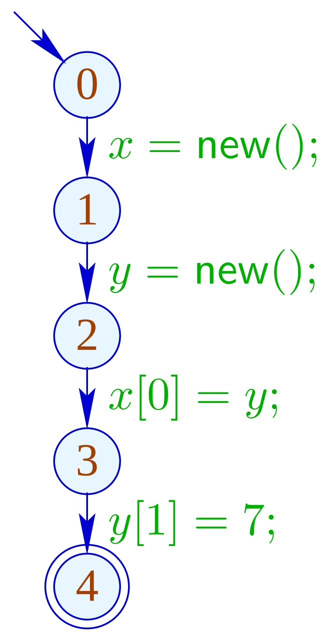
Simple Example:

$x = \textsf{new}();$

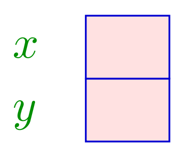
$y = \textsf{new}();$

$x[0] = y;$

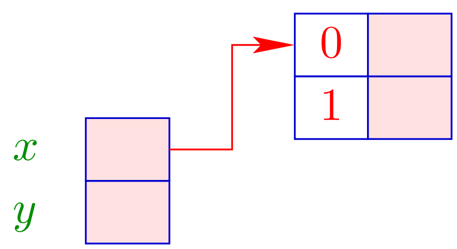
$y[1] = 7;$



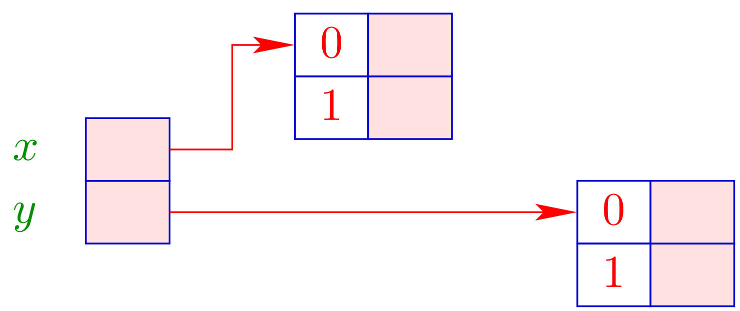
The Semantics:



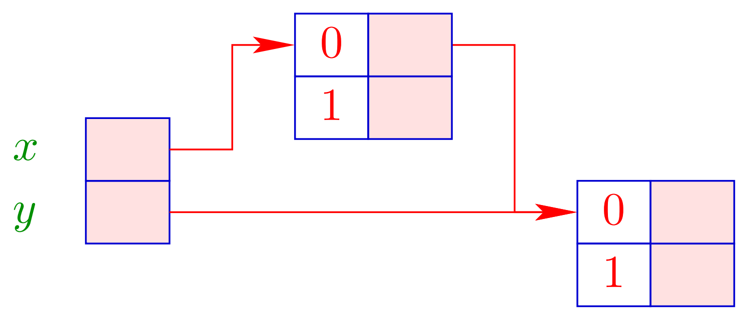
The Semantics:



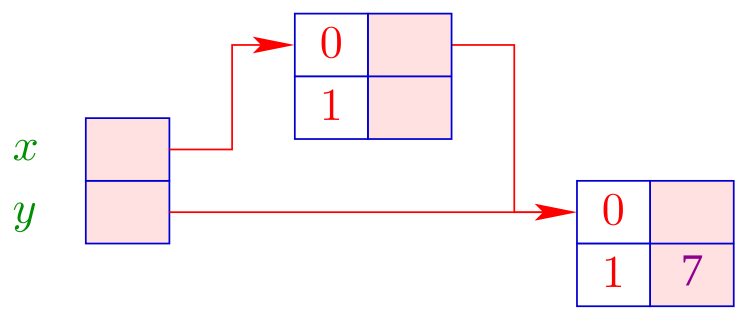
The Semantics:



The Semantics:



The Semantics:



More Complex Example:

$r = \textsf{Null};$

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

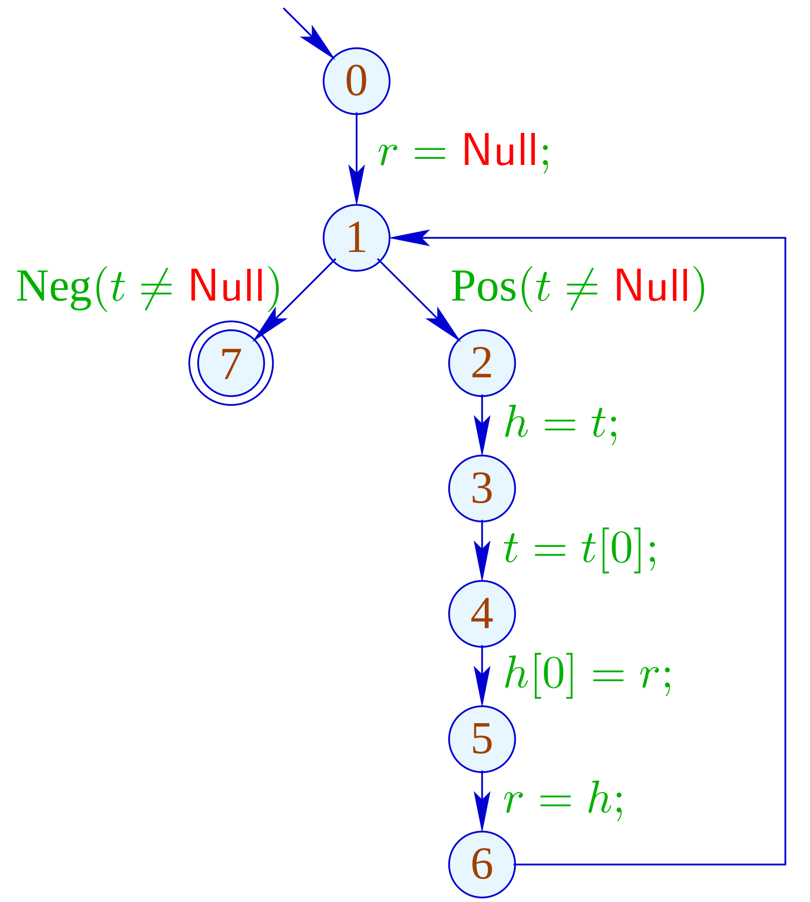
$\quad h = t;$

$\quad t = t[0];$

$\quad h[0] = r;$

$\quad r = h;$

$\}$



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: $0 = \mathsf{Null}$

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

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

Caveat:

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:

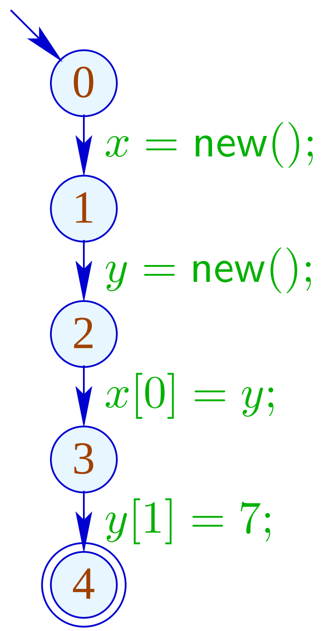
- Distinguish finitely many classes of blocks.
- Collect all addresses of a block into one set!
- Use sets of addresses as abstract values!

$\Longrightarrow$ 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	$\{(0,1)\}$	$\{(1,2)\}$	$\{(1,2)\}$

The Effects of Edges:

$$\begin{array}{lll}
[\![(_,;,_)]\!]^\sharp\,(D,M) & = & (D,M) \\
[\![(_,\mathrm{Pos}(e),_)]\!]^\sharp\,(D,M) & = & (D,M) \\
[\![(_,x=y;,_)]\!]^\sharp\,(D,M) & = & (D\oplus\{x\mapsto D\,y\},M) \\
[\![(_,x=e;,_)]\!]^\sharp\,(D,M) & = & (D\oplus\{x\mapsto\emptyset\},M) \qquad , \qquad e\notin \mathit{Vars} \\
\\
[\![(u,x=\mathsf{new}();,v)]\!]^\sharp\,(D,M) & = & (D\oplus\{x\mapsto\{(u,v)\}\},M) \\
[\![(_,x=y[e];,_)]\!]^\sharp\,(D,M) & = & (D\oplus\{x\mapsto\bigcup\{M(f)\mid f\in D\,y\}\},M) \\
[\![(_,y[e_1]=x;,_)]\!]^\sharp\,(D,M) & = & (D,M\oplus\{f\mapsto(M\,f\cup D\,x)\mid f\in D\,y\})
\end{array}$$

Caveat:

- The value Null has been ignored. Dereferencing of Null or negative indices are not detected :-(
- Destructive updates are only possible for variables, not for blocks in storage!

  $\Longrightarrow$ no information, if not all block entries are initialized before use :-((
- The effects now depend on the edge itself.

  The analysis cannot be proven correct w.r.t. the reference semantics :-(

  In order to prove correctness, we first instrument the concrete semantics with extra information which records where a block has been created.

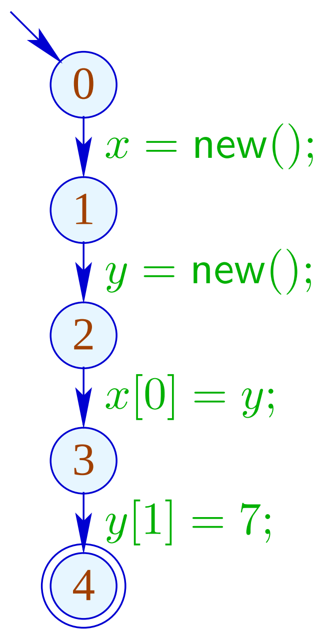
...

- We compute possible points-to information.
- From that, we can extract may-alias information.
- The analysis can be rather expensive — without finding very much :-(
- Separate information for each program point can perhaps be abandoned ??

Alias Analysis 2. Idea:

Compute for each variable and address a value which safely approximates the values at every program point simultaneously !

... in the Simple Example:



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

Each edge (u, lab, v) gives rise to constraints:

lab	$Constraint$
$x = y;$	$\mathcal{P}[x] \quad \supseteq \quad \mathcal{P}[y]$
$x = \textsf{new}();$	$\mathcal{P}[x] \quad \supseteq \quad \{(u, v)\}$
$x = y[e];$	$\mathcal{P}[x] \quad \supseteq \quad \bigcup\{\mathcal{P}[f] \mid f \in \mathcal{P}[y]\}$
$y[e_1] = x;$	$\mathcal{P}[f] \quad \supseteq \quad (f \in \mathcal{P}[y])\,?\,\mathcal{P}[x] \ : \ \emptyset$
	for all $\quad f \in Addr^\sharp$

Other edges have no effect :-)

Discussion:

- The resulting constraint system has size $\mathcal{O}(k \cdot n)$ for k abstract addresses and n edges :-(
- The number of necessary iterations is $\mathcal{O}(k + \# Vars)$ ...
- The computed information is perhaps still too zu precise !!?
- In order to prove correctness of a solution $s^\sharp \in States^\sharp$ we show: