Let $\quad (\rho, \mu) \in State_h$. Then we obtain for the new edges:

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

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:

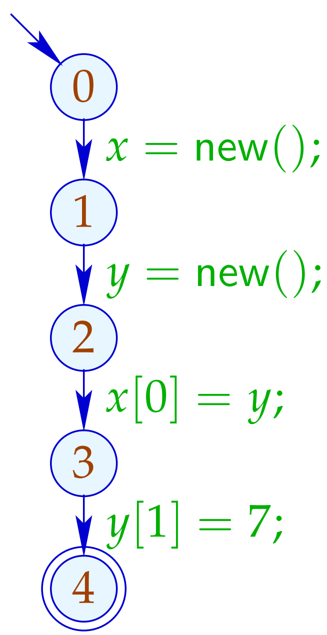
- Distinguish finitely many classes of blocks.
- Collect all addresses of a block into one set!
- Use sets of addresses as abstract values!

$\Longrightarrow$ Points-to-Analysis

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

// complete lattice !!!

... in the Simple Example:



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

The Effects of Edges:

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

Warning:

- 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.

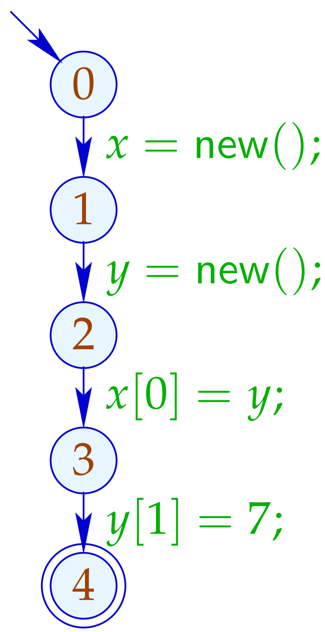
...

- 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 ??

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

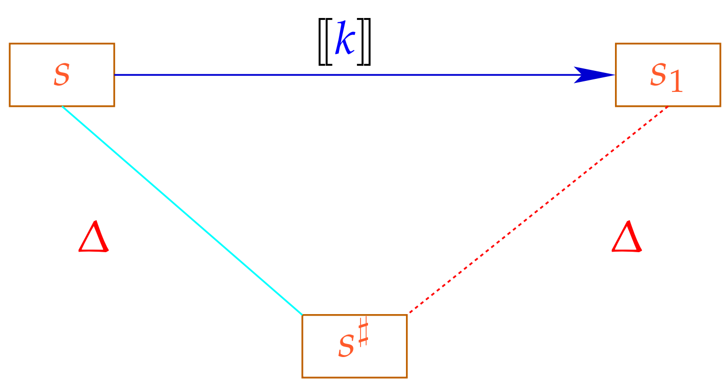
Each edge (u, lab, v) gives rise to constraints:

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

Other edges have no effect :-)

Discussion:

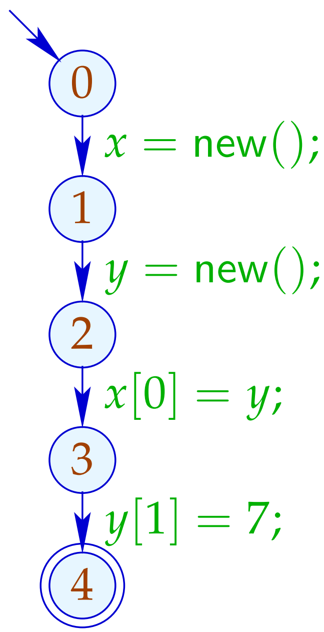
- The resulting constraint system has size $\mathcal{O}(k \cdot n)$ for k abstract addresses and n edges :-(
- The number of necessary iterations is $\mathcal{O}(k + \#\mathit{Vars})$ ...
- The computed information is perhaps still too zu precise !!?
- In order to prove correctness of a solution $s^\sharp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



Alias Analysis 3. Idea:

Determine one equivalence relation $\equiv$ on variables x and memory accesses $y[\,]$ with $s_1 \equiv s_2$ whenever s_1, s_2 may contain the same address at some u_1, u_2

... in the Simple Example:



$$\equiv \;=\; \{\{x\},\; \{y, x[\,]\},\; \{y[\,]\}\}$$

Discussion:

- $\longrightarrow$ We compute a single information fo the whole program.
- $\longrightarrow$ The computation of this information maintains partitions $\pi = \{P_1, \ldots, P_m\}$:-)
- $\longrightarrow$ Individual sets P_i are identified by means of representatives $p_i \in P_i$.
- $\longrightarrow$ The operations on a partition π are:

$$\begin{array}{lcll}
\textsf{find}\,(\pi, p) & = & p_i & \text{if } p \in P_i \\
 & // & \text{returns the representative} & \\
\textsf{union}\,(\pi, p_{i_1}, p_{i_2}) & = & \{P_{i_1} \cup P_{i_2}\} \cup \{P_j \mid i_1 \neq j \neq i_2\} & \\
 & // & \text{unions the represented classes} &
\end{array}$$

$\longrightarrow$ If $x_1, x_2 \in \mathit{Vars}$ are equivalent, then also $x_1[\,]$ and $x_2[\,]$ must be equivalent :-)

$\longrightarrow$ If $P_i \cap \mathit{Vars} \neq \emptyset$, then we choose $p_i \in \mathit{Vars}$. Then we can apply union recursively :

$$
\begin{array}{lllll}
\mathsf{union}^*\,(\pi, q_1, q_2) & = & \mathsf{let} & p_{i_1} = \mathsf{find}\,(\pi, q_1) & \\
 & & & p_{i_2} = \mathsf{find}\,(\pi, q_2) & \\
 & & \mathsf{in} & \multicolumn{2}{l}{\mathsf{if}\ p_{i_1} == p_{i_2}\ \mathsf{then}\ \pi} \\
 & & & \multicolumn{2}{l}{\mathsf{else}\ \ \mathsf{let}\ \pi = \mathsf{union}\,(\pi, p_{i_1}, p_{i_2})} \\
 & & & \multicolumn{2}{l}{\qquad\quad \mathsf{in}\ \ \mathsf{if}\ p_{i_1}, p_{i_2} \in \mathit{Vars}\ \mathsf{then}} \\
 & & & \multicolumn{2}{l}{\qquad\qquad\quad \mathsf{union}^*\,(\pi, p_{i_1}[\,], p_{i_2}[\,])}
\end{array}
$$

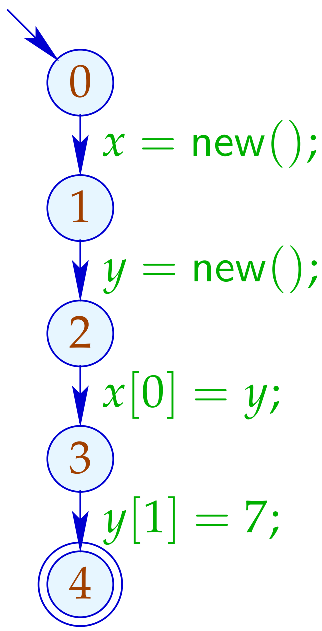
The analysis iterates over all edges once:

$$\pi = \{\{x\}, \{x[\,]\} \mid x \in Vars\};$$
$$\text{forall} \quad k = (\_, lab, \_) \quad \text{do} \quad \pi = [\![lab]\!]^\sharp\,\pi;$$

where:

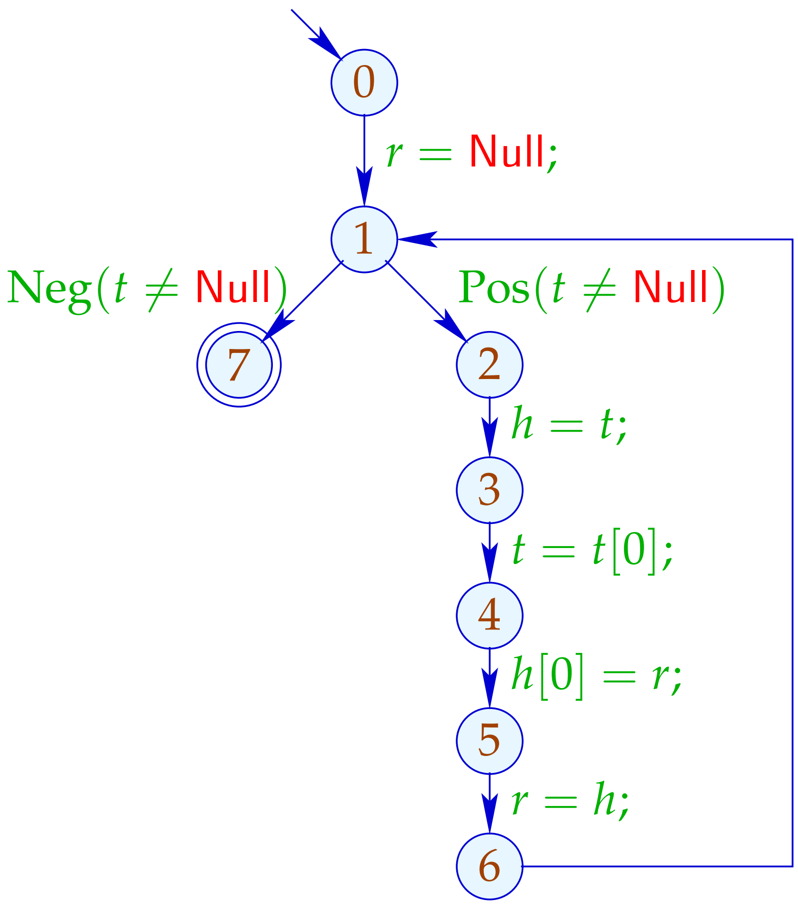
$$\begin{array}{lll}
[\![x = y;]\!]^\sharp\,\pi & = & \text{union}^*\,(\pi, x, y) \\
[\![x = y[e];]\!]^\sharp\,\pi & = & \text{union}^*\,(\pi, x, y[\,]) \\
[\![y[e] = x;]\!]^\sharp\,\pi & = & \text{union}^*\,(\pi, x, y[\,]) \\
[\![lab]\!]^\sharp\,\pi & = & \pi \qquad\qquad \text{otherwise}
\end{array}$$

... in the Simple Example:



	$\{\{x\},\{y\},\{x[\,]\},\{y[\,]\}\}$
$(0,1)$	$\{\{x\},\{y\},\{x[\,]\},\{y[\,]\}\}$
$(1,2)$	$\{\{x\},\{y\},\{x[\,]\},\{y[\,]\}\}$
$(2,3)$	$\{\{x\},\boxed{\{y,x[\,]\}},\{y[\,]\}\}$
$(3,4)$	$\{\{x\},\{y,x[\,]\},\{y[\,]\}\}$

... in the More Complex Example:



	$\{\{h\},\{r\},\{t\},\{h[\,]\},\{t[\,]\}\}$
$(2,3)$	$\{\boxed{\{h,t\}},\{r\},\boxed{\{h[\,],t[\,]\}}\}$
$(3,4)$	$\{\boxed{\{h,t,h[\,],t[\,]\}},\{r\}\}$
$(4,5)$	$\{\boxed{\{h,t,r,h[\,],t[\,]\}}\}$
$(5,6)$	$\{\{h,t,r,h[\,],t[\,]\}\}$

Warning:

In order to find something, we must assume that variables / addresses always receive a value before they are accessed.

Complexity:

we havve:

$\mathcal{O}(\#\,edges + \#\,Vars)$	calls of	union*
$\mathcal{O}(\#\,edges + \#\,Vars)$	calls of	find
$\mathcal{O}(\#\,Vars)$	calls of	union

$\Longrightarrow$ We require efficient Union-Find data-structure :-)