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 = new();  // allocation of a new block
  x = y[垛];  // indexed read access to a block
  y[垛1] = e2; // indexed write access to a block
  ```
- Blocks are possibly infinite :-(
- For simplicity, all pointers point to the beginning of a block.

Simple Example:

```lang
x = new();
y = new();
x[0] = y;
y[1] = 7;
```

The Semantics:

```lang
x   y
```
The Semantics:

```latex
x \quad 0 \quad 1
y
```

The Semantics:

```latex
x \quad 0 \quad 1
y
```

More Complex Example:

```latex
r = \textbf{Null};
\textbf{while} (t \neq \textbf{Null}) \{ \\
\quad h = t; \\
\quad t = t[0]; \\
\quad h[0] = r; \\
\quad r = h;
\}
```

More Complex Example:

```latex
r = \textbf{Null};
\textbf{while} (t \neq \textbf{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:

\[
\begin{align*}
\text{Addr}_h &= \{ \text{ref } a \mid 0 \leq a < h \} & \text{// addresses} \\
\text{Val}_h &= \text{Addr}_h \cup \mathbb{Z} & \text{// values} \\
\text{Store}_h &= (\text{Addr}_h \times \mathbb{N}_0) \rightarrow \text{Val}_h & \text{// store} \\
\text{State}_h &= (\text{Vars} \rightarrow \text{Val}_h) \times \text{Store}_h & \text{// states}
\end{align*}
\]

For simplicity, we set: \( 0 = \text{Null} \)

Caveat:

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

\[
\begin{align*}
& x = \text{new}(); \\
& y = \text{new}(); \\
& y = \text{new}(); \\
& x = \text{new}();
\end{align*}
\]

are not considered as equivalent !?!

Possible Solution:

Define equivalence only up to permutation of addresses \( :-) \)

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

\[
\begin{align*}
[x = \text{new}()] (\rho, \mu) &= (\rho \cup \{ x \mapsto \text{ref } h \}, \mu) \\
\mu \cup \{ (\text{ref } h, i) \mapsto (\mu \cup \{ i \in \mathbb{N}_0 \}) \} \\
[y[e_1] = e_2] (\rho, \mu) &= (\rho \cup \{ x \mapsto \mu (y[e_1], \rho) \}, \mu)
\end{align*}
\]

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

\[
\begin{align*}
\text{Addr}^3 &= \text{Edges} & \text{// creation edges} \\
\text{Val}^3 &= 2^{\text{Addr}^3} & \text{// abstract values} \\
\text{Store}^3 &= \text{Addr}^3 \rightarrow \text{Val}^3 & \text{// abstract store} \\
\text{State}^3 &= (\text{Vars} \rightarrow \text{Val}^3) \times \text{Store}^3 & \text{// complete lattice !!!}
\end{align*}
\]
Let \((\rho, \mu) \in \text{State}_h\). Then we obtain for the new edges:

\[
[x = \text{new}()]\ (\rho, \mu) = (\rho \oplus \{x \mapsto \text{ref } h\}, \\
\mu \oplus \{(\text{ref } h, i) \mapsto 0 \mid i \in \mathbb{N}_0\})
\]

\[
[x = y[e];] \ (\rho, \mu) = (\rho \oplus \{x \mapsto \mu(y, [e]\rho), \mu\}, \\
\mu \oplus \{(\rho, y, [e]\rho) \mapsto [e2]\rho\})
\]

... in the Simple Example:

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Caveat:

- The value \textbf{Null} has been ignored. Dereferencing of \textbf{Null} or negative indices are not detected :-(

- \textbf{Destructive updates} are only possible for variables, not for blocks in storage!

  \[\implies\] 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.
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!
  - 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:

```
0  x = new();
1  y = new();
2  x[0] = y;
3  y[1] = 7;
```

<table>
<thead>
<tr>
<th>Value</th>
<th>Set</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>(0,1)</td>
</tr>
<tr>
<td>y</td>
<td>(1,2)</td>
</tr>
<tr>
<td>x[0]</td>
<td>(0,1)</td>
</tr>
<tr>
<td>x[1]</td>
<td>(1,2)</td>
</tr>
<tr>
<td>y[1]</td>
<td>0</td>
</tr>
</tbody>
</table>
Each edge \((u, lab, v)\) gives rise to constraints:

<table>
<thead>
<tr>
<th>lab</th>
<th>Constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x = y;)</td>
<td>(\mathcal{P}[x] \supseteq \mathcal{P}[y])</td>
</tr>
<tr>
<td>(x = \text{new}();)</td>
<td>(\mathcal{P}[x] \supseteq {(u, v)})</td>
</tr>
<tr>
<td>(x = y[e];)</td>
<td>(\mathcal{P}[x] \supseteq \bigcup{\mathcal{P}[f] \mid f \in \mathcal{P}[y]})</td>
</tr>
<tr>
<td>(y[e_1] = x;)</td>
<td>(\mathcal{P}[f] \supseteq {f \in \mathcal{P}[y] : \mathcal{P}[x] : \emptyset} ) for all (f \in Add^4)</td>
</tr>
</tbody>
</table>

Other edges have no effect :-)

---

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

<table>
<thead>
<tr>
<th>lab</th>
<th>Constraint</th>
</tr>
</thead>
<tbody>
<tr>
<td>(x = y;)</td>
<td>(\mathcal{P}[x] \supseteq \mathcal{P}[y])</td>
</tr>
<tr>
<td>(x = \text{new}();)</td>
<td>(\mathcal{P}[x] \supseteq {(u, v)})</td>
</tr>
<tr>
<td>(x = y[e];)</td>
<td>(\mathcal{P}[x] \supseteq \bigcup{\mathcal{P}[f] \mid f \in \mathcal{P}[y]})</td>
</tr>
<tr>
<td>(y[e_1] = x;)</td>
<td>(\mathcal{P}[f] \supseteq {f \in \mathcal{P}[y] : \mathcal{P}[x] : \emptyset} ) for all (f \in Add^4)</td>
</tr>
</tbody>
</table>

Other edges have no effect :-)}