1.6 Pointer Analysis

Questions

→ Are two addresses possibly equal? May Alias
→ Are two addresses definitively equal? Must Alias

⇒ 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:

$$[x = c] A = (A \cup \{c\}) \setminus Expr$$
$$[x = M[e]] A = (A \cup \{e, M[e]\}) \setminus Expr$$
$$[M[e_1] = e_2] A = (A \cup \{e_1, e_2\}) \setminus Loads$$
Values of Variables:

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

- Extend the Effects of Edges:

\[
[x = M[e];] \mathcal{E} V e' = \begin{cases}
\{x\} & \text{if } e' = M[e] \\
\emptyset & \text{if } e' = e \\
V e' \setminus \{x\} & \text{otherwise}
\end{cases}
\]

\[
[M[e_1] = e_2]; \mathcal{E} V e' = \begin{cases}
\emptyset & \text{if } e' \in \{e_1, e_2\} \\
V e' & \text{otherwise}
\end{cases}
\]

The analyses so far without alias information

Available Expressions:

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

- Extend the Effects of Edges:

\[
[x = e]; \mathcal{E} A = (A \cup \{e\}) \setminus \textit{Expr}_g
\]

\[
[x = M[e];] \mathcal{E} A = (A \cup \{e, M[e]\}) \setminus \textit{Expr}_g
\]

\[
[M[e_1] = e_2]; \mathcal{E} A = (A \cup \{e_1, e_2\}) \setminus \textit{Loads}
\]

Constant Propagation:

- Extend the abstract state by an abstract store $M$.

- Execute accesses to known memory locations!

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

\[
[M[e_1] = e_2]; \mathcal{D} (D, M) = \begin{cases}
[D \oplus \{a \mapsto [e_2]D\}] & \text{if } [e_2]D = a \sqsubset T \\
(D, \top) & \text{otherwise}
\end{cases}
\]

\[\top a = \top \quad (a \in \mathbb{N})\]

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 $M$.

\[\text{constant propagation fails} \quad \text{memory accesses/pointers kill precision}\]
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 $M$.

$\implies$ constant propagation fails
$\implies$ 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:
  
  $x = \text{new}(); // \text{allocation of a new block}$
  
  $x = y[e]; // \text{indexed read access to a block}$
  
  $y[e_1] = e_2; // \text{indexed write access to a block}$

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

The Semantics
More Complex Example

\[ r = \text{Null}; \]
while \( t \neq \text{Null} \) \{
\[ h = t; \]
\[ t = t[0]; \]
\[ h[0] = r; \]
\[ r = h; \]
\}

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

\[
[x = \text{new}()] (\rho, \mu) = (\rho \oplus \{ x \mapsto \text{ref } h \}, \mu \oplus \{(\text{ref } h, i) \mapsto 0 \mid i \in \mathbb{N}_0\})
\]
\[
x[y/x] (\rho, \mu) = (\rho \oplus \{ x \mapsto \mu (\rho, [y/x]), \mu \})
\]
\[
y[e_1] = e_2 \] (\rho, \mu) = (\rho, \mu \oplus \{ (\rho, [e_1/x]) \mapsto [e_2/x] \})
\]

Concrete Semantics

A store consists of a finite collection of blocks.

After \( h \) new-operations we obtain:

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

For simplicity, we set: \( 0 = \text{Null} \)
Let \((\rho, \mu) \in \text{States}^d\). Then we obtain for the new edges:

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

\[
[x = y[e];] \ (\rho, \mu) = (\rho \uplus \{x \mapsto \mu (p, y, [e] \rho), \mu\} \\
[y[c_1] = c_2] \ (\rho, \mu) = (\rho, \mu \uplus \{([\rho, y, c_1] \rho) \mapsto [c_2] \rho\})
\]

### Caveat

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

\[
x = \text{new}; \\
y = \text{new};
\]

\[
x = \text{new}; \\
y = \text{new}; \\
x = \text{new};
\]

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!

\[ \Longrightarrow \text{ Points-to-Analysis} \]

\[ \text{Addr}^d = \text{Edges} \quad \text{Va}^d = 2^{\text{Addr}^d} \quad \text{Store}^d = \text{Addr}^d \rightarrow \text{Va}^d \]

\[ \text{State}^d = (\text{Vars} \rightarrow \text{Va}^d) \times \text{Store}^d \]

// complete lattice !!!
Aliasing 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!

\[ \text{Addr}^t = \text{Edges} \quad \text{// creation edges} \]
\[ \text{Val}^t = 2^{\text{Addr}^t} \quad \text{// abstract values} \]
\[ \text{Store}^t = \text{Addr}^t \rightarrow \text{Val}^t \quad \text{// abstract store} \]
\[ \text{State}^t = (\text{Vars} \rightarrow \text{Val}^t) \times \text{Store}^t \quad \text{// abstract states} \]

// complete lattice !!!

... in the Simple Example

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>x = new();</td>
<td>y = new();</td>
</tr>
<tr>
<td>1</td>
<td>x[0] = y;</td>
<td>y[1] = 7;</td>
</tr>
<tr>
<td>2</td>
<td>x[0] = y;</td>
<td>y[1] = 7;</td>
</tr>
<tr>
<td>3</td>
<td>x[0] = y;</td>
<td>y[1] = 7;</td>
</tr>
<tr>
<td>4</td>
<td>x[0] = y;</td>
<td>y[1] = 7;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>x</th>
<th>y</th>
<th>(0, 1)</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

The Effects of Edges

\[ [(\_, \_ \_ \_)]^t (D, M) = (D, M) \]
\[ [(\_, \text{Pos}(e), \_)]^t (D, M) = (D, M) \]
\[ [(\_, x = y, \_)]^t (D, M) = (D \oplus \{x \rightarrow D y\}, M) \]
\[ [(\_, x = e, \_)]^t (D, M) = (D \oplus \{x \rightarrow \emptyset\}, M) \quad , \quad e \not\in \text{Vars} \]
\[ [(u, x = \text{new}(), v)]^t (D, M) = (D \oplus \{x \rightarrow \{(u, v)\}\}, M) \]
\[ [(\_, x = v[e], \_)]^t (D, M) = (D \oplus \{x \rightarrow \bigcup\{M(f) \mid f \in D y\}\}, M) \]
\[ [(\_, y[e_1] = x, \_)]^t (D, M) = (D, M \oplus \{f \rightarrow (M f \cup D x) \mid f \in D y\}) \]

\[ \uparrow \quad \uparrow \]

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!

\[ \Rightarrow \text{no information, if not all block entries are initialized before use.} \]
- 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 \textbf{instrument} the concrete semantics with extra information which records where a block has been created.
The Effects of Edges

\[
\begin{align*}
[[\_, \_]]^I(D, M) & = (D, M) \\
[[\_, \text{Pos}(e), \_]]^I(D, M) & = (D, M) \\
[[\_, x = y, \_]]^I(D, M) & = (D \oplus \{x \mapsto D\overline{y}\}, M) \\
[[\_, x = e, \_]]^I(D, M) & = (D \oplus \{x \mapsto \emptyset\}, M), \quad e \notin \text{Vars} \\
[[u, x = \text{new}(e); v]]^I(D, M) & = (D \oplus \{x \mapsto \{(u, v)\}\}, M) \\
[[\_, x = y[e], \_]]^I(D, M) & = (D \oplus \{x \mapsto \bigcup\{M(f) \mid f \in D\overline{y}\}\}, M) \\
[[\_, y[e_1] = x, \_]]^I(D, M) & = (D, M \oplus \{f \mapsto (M \cup D) f \mid f \in D\overline{y}\})
\end{align*}
\]

Caveat

- The value \textbf{Null} has been ignored. Dereferencing \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 \textbf{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

\[
\begin{align*}
\text{program} & \quad \text{state} \\
1. & \quad x = \text{new}(); \quad \{0, 1\} \\
2. & \quad y = \text{new}(); \quad \{0, 1, 2\} \\
3. & \quad x[0] = y; \quad \{0, 1\} \\
4. & \quad y[1] = 7; \quad \emptyset
\end{align*}
\]

Discussion

• The resulting constraint system has size \(O(k \cdot n)\) for \(k\) abstract addresses and \(n\) edges.
• The number of necessary iterations is \(O(k(k + \#Vars))\) ...
• The computed information is perhaps still too zu precise !??
• In order to prove correctness of a solution \(s^1 \in \text{States}^1\) we show:

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>(P[x] \supseteq P[y])</td>
</tr>
<tr>
<td>(x = \text{new}();)</td>
<td>(P[x] \supseteq {u, v})</td>
</tr>
<tr>
<td>(x = y[e];)</td>
<td>(P[x] \supseteq \bigcup {P[f] \mid f \in P[y]})</td>
</tr>
<tr>
<td>(y[e] = x;)</td>
<td>(P[f] \supseteq {f \in P[y]} \cup P[x] : \emptyset ) for all (f \in Addr^1)</td>
</tr>
</tbody>
</table>

Other edges have no effect.