Modular Heap Shape Analysis for Java Programs

Florian Frohn[1] Jürgen Giesl[1]

[1]RWTH Aachen University, Germany

September 19, 2017

AProVE and Java

AProVE: termination and complexity analysis tool for Java

AProVE and Java

AProVE: termination and complexity analysis tool for Java

```
public void add(Object x) {
  List l = this;
  while (l.n != null) {
    l = l.n;
  }
  List ll = new List();
  l.n = ll;
  ll.v = x;
}
```

AProVE and Java

AProVE: termination and complexity analysis tool for Java

```
public void add(Object x) {
  List l = this;
  while (l.n != null) {
    l = l.n;
  }
  List ll = new List();
  l.n = ll;
  ll.v = x;
}
```

- sophisticated heap shape analysis

AProVE and Java

AProVE: termination and complexity analysis tool for Java

```
public void add(Object x) {
  List l = this;
  while (l.n != null) {
    l = l.n;
  }
  List ll = new List();
  l.n = ll;
  ll.v = x;
}
```

- sophisticated heap shape analysis
- lacks **modularity**

AProVE's Approach

```
public void add(Object x) {

  List l = this;


  while (l.n != null) {

    l = l.n;


  }

  List ll = new List();

  l.n = ll;

  ll.v = x;


}
```

AProVE's Approach

```
public void add(Object x) {
⟨this = o1, x = o2 | ε⟩
  List l = this;


  while (l.n != null) {

    l = l.n;


  }

  List ll = new List();

  l.n = ll;

  ll.v = x;


}
```

AProVE's Approach

```
public void add(Object x) {
⟨this = o1, x = o2 | ε⟩
  List l = this;
⟨this = o1, x = o2, l = o1 | ε⟩

  while (l.n != null) {

    l = l.n;



  }

  List ll = new List();

  l.n = ll;

  ll.v = x;


}
```

AProVE's Approach

```
public void add(Object x) {
⟨this = o1, x = o2 | ε⟩
  List l = this;
⟨this = o1, x = o2, l = o1 | ε⟩

  while (l.n != null) {
⟨this = o1, x = o2, l = o1 | ε⟩
    l = l.n;



  }

  List ll = new List();

  l.n = ll;

  ll.v = x;


}
```

AProVE's Approach

```
public void add(Object x) {
```
$\langle \texttt{this} = o_1, \texttt{x} = o_2 \mid \varepsilon \rangle$
```
  List l = this;
```
$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_1 \mid \varepsilon \rangle$
```
  while (l.n != null) {
```
$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_1 \mid \varepsilon \rangle$
```
    l = l.n;
```
$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_3 \mid o_1 \xrightarrow{\texttt{n}} o_3 \rangle$
```
  }

  List ll = new List();

  l.n = ll;

  ll.v = x;


}
```

AProVE's Approach

```
public void add(Object x) {
```
$\langle \texttt{this} = o_1, \texttt{x} = o_2 \mid \varepsilon \rangle$
```
  List l = this;
```
~~$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_1 \mid \varepsilon \rangle$~~
$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_3 \mid o_1 =^{?} o_3, o_1 \not\downarrow o_3 \rangle$
```
  while (l.n != null) {
```
$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_1 \mid \varepsilon \rangle$
```
    l = l.n;
```
$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_3 \mid o_1 \xrightarrow{\texttt{n}} o_3 \rangle$

```
  }

  List ll = new List();

  l.n = ll;

  ll.v = x;


}
```

AProVE's Approach

```
public void add(Object x) {
```
$\langle \texttt{this} = o_1, \texttt{x} = o_2 \mid \varepsilon \rangle$
```
  List l = this;
```
~~$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_1 \mid \varepsilon \rangle$~~
$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_3 \mid o_1 =^{?} o_3, o_1 \not\curlyvee o_3 \rangle$
```
  while (l.n != null) {
```
$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_3 \mid o_1 =^{?} o_3, o_1 \not\curlyvee o_3 \rangle$
```
    l = l.n;
```
$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_3 \mid o_1 \xrightarrow{\texttt{n}} o_3 \rangle$

```
  }

  List ll = new List();

  l.n = ll;

  ll.v = x;


}
```

AProVE's Approach

```
public void add(Object x) {
```
$\langle \texttt{this} = o_1, \texttt{x} = o_2 \mid \varepsilon \rangle$
```
  List l = this;
```
~~$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_1 \mid \varepsilon \rangle$~~

$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_3 \mid o_1 =^{?} o_3, o_1 \not\downarrow o_3 \rangle$
```
  while (l.n != null) {
```
$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_3 \mid o_1 =^{?} o_3, o_1 \not\downarrow o_3 \rangle$
```
    l = l.n;
```
$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_4 \mid o_1 =^{?} o_3, o_1 \not\downarrow o_3, o_3 \xrightarrow{\texttt{n}} o_4, o_1 =^{?} o_4, o_1 \not\downarrow o_4 \rangle$

```
  }

  List ll = new List();

  l.n = ll;

  ll.v = x;


}
```

AProVE's Approach

```
public void add(Object x) {
```
$\langle \texttt{this} = o_1, \texttt{x} = o_2 \mid \varepsilon \rangle$
```
  List l = this;
```
~~$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_1 \mid \varepsilon \rangle$~~
$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_3 \mid o_1 =^{?} o_3, o_1 \not\downarrow o_3 \rangle$
```
  while (l.n != null) {
```
$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_3 \mid o_1 =^{?} o_3, o_1 \not\downarrow o_3 \rangle$
```
    l = l.n;
```
~~$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_4 \mid o_1 =^{?} o_3, o_1 \not\downarrow o_3, o_3 \xrightarrow{\texttt{n}} o_4, o_1 =^{?} o_4, o_1 \not\downarrow o_4 \rangle$~~
$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_4 \mid o_1 =^{?} o_4, o_1 \not\downarrow o_4 \rangle$
```
  }

  List ll = new List();

  l.n = ll;

  ll.v = x;


}
```

AProVE's Approach

```
public void add(Object x) {
```
$\langle \texttt{this} = o_1, \texttt{x} = o_2 \mid \varepsilon \rangle$
```
  List l = this;
```
~~$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_1 \mid \varepsilon \rangle$~~
$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_3 \mid o_1 =^{?} o_3, o_1 \not\downarrow o_3 \rangle$
```
  while (l.n != null) {
```
$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_3 \mid o_1 =^{?} o_3, o_1 \not\downarrow o_3 \rangle$
```
    l = l.n;
```
~~$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_4 \mid o_1 =^{?} o_3, o_1 \not\downarrow o_3, o_3 \xrightarrow{\texttt{n}} o_4, o_1 =^{?} o_4, o_1 \not\downarrow o_4 \rangle$~~
$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_4 \mid o_1 =^{?} o_4, o_1 \not\downarrow o_4 \rangle$
```
  }
```
$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_3 \mid o_1 =^{?} o_3, o_1 \not\downarrow o_3 \rangle$
```
  List ll = new List();

  l.n = ll;

  ll.v = x;


}
```

AProVE's Approach

```
public void add(Object x) {
```
$\langle \texttt{this} = o_1, \texttt{x} = o_2 \mid \varepsilon \rangle$
```
  List l = this;
```
~~$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_1 \mid \varepsilon \rangle$~~
$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_3 \mid o_1 =^{?} o_3, o_1 \not\downarrow o_3 \rangle$
```
  while (l.n != null) {
```
$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_3 \mid o_1 =^{?} o_3, o_1 \not\downarrow o_3 \rangle$
```
    l = l.n;
```
~~$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_4 \mid o_1 =^{?} o_3, o_1 \not\downarrow o_3, o_3 \xrightarrow{\texttt{n}} o_4, o_1 =^{?} o_4, o_1 \not\downarrow o_4 \rangle$~~
$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_4 \mid o_1 =^{?} o_4, o_1 \not\downarrow o_4 \rangle$
```
  }
```
$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_3 \mid o_1 =^{?} o_3, o_1 \not\downarrow o_3 \rangle$
```
  List ll = new List();
```
$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_3, \texttt{ll} = o_5 \mid o_1 =^{?} o_3, o_1 \not\downarrow o_3 \rangle$
```
  l.n = ll;

  ll.v = x;


}
```

AProVE's Approach

```
public void add(Object x) {
```
$\langle \texttt{this} = o_1, \texttt{x} = o_2 \mid \varepsilon \rangle$
```
  List l = this;
```
~~$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_1 \mid \varepsilon \rangle$~~
$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_3 \mid o_1 =^{?} o_3, o_1 \not\downarrow o_3 \rangle$
```
  while (l.n != null) {
```
$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_3 \mid o_1 =^{?} o_3, o_1 \not\downarrow o_3 \rangle$
```
    l = l.n;
```
~~$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_4 \mid o_1 =^{?} o_3, o_1 \not\downarrow o_3, o_3 \xrightarrow{\texttt{n}} o_4, o_1 =^{?} o_4, o_1 \not\downarrow o_4 \rangle$~~
$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_4 \mid o_1 =^{?} o_4, o_1 \not\downarrow o_4 \rangle$
```
  }
```
$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_3 \mid o_1 =^{?} o_3, o_1 \not\downarrow o_3 \rangle$
```
  List ll = new List();
```
$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_3, \texttt{ll} = o_5 \mid o_1 =^{?} o_3, o_1 \not\downarrow o_3 \rangle$
```
  l.n = ll;
```
$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_3, \texttt{ll} = o_5 \mid o_1 =^{?} o_3, o_1 \not\downarrow o_3, o_3 \xrightarrow{\texttt{n}} o_5, \ldots \rangle$
```
  ll.v = x;



}
```

AProVE's Approach

```
public void add(Object x) {
```
$\langle \texttt{this} = o_1, \texttt{x} = o_2 \mid \varepsilon \rangle$
```
  List l = this;
```
~~$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_1 \mid \varepsilon \rangle$~~
$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_3 \mid o_1 =^{?} o_3, o_1 \downarrow o_3 \rangle$
```
  while (l.n != null) {
```
$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_3 \mid o_1 =^{?} o_3, o_1 \downarrow o_3 \rangle$
```
    l = l.n;
```
~~$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_4 \mid o_1 =^{?} o_3, o_1 \downarrow o_3, o_3 \xrightarrow{\texttt{n}} o_4, o_1 =^{?} o_4, o_1 \downarrow o_4 \rangle$~~
$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_4 \mid o_1 =^{?} o_4, o_1 \downarrow o_4 \rangle$
```
  }
```
$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_3 \mid o_1 =^{?} o_3, o_1 \downarrow o_3 \rangle$
```
  List ll = new List();
```
$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_3, \texttt{ll} = o_5 \mid o_1 =^{?} o_3, o_1 \downarrow o_3 \rangle$
```
  l.n = ll;
```
$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_3, \texttt{ll} = o_5 \mid o_1 =^{?} o_3, o_1 \downarrow o_3, o_3 \xrightarrow{\texttt{n}} o_5, \ldots \rangle$
```
  ll.v = x;
```
$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_3, \texttt{ll} = o_5 \mid o_1 =^{?} o_3, o_1 \downarrow o_3, o_3 \xrightarrow{\texttt{n}} o_5, o_5 \xrightarrow{\texttt{v}} o_2, \ldots \rangle$

```
}
```

AProVE's Approach

```
public void add(Object x) {
```
$\langle \texttt{this} = o_1, \texttt{x} = o_2 \mid \varepsilon \rangle$
```
  List l = this;
```
~~$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_1 \mid \varepsilon \rangle$~~
$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_3 \mid o_1 =^{?} o_3, o_1 \downarrow o_3 \rangle$
```
  while (l.n != null) {
```
$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_3 \mid o_1 =^{?} o_3, o_1 \downarrow o_3 \rangle$
```
    l = l.n;
```
~~$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_4 \mid o_1 =^{?} o_3, o_1 \downarrow o_3, o_3 \xrightarrow{\texttt{n}} o_4, o_1 =^{?} o_4, o_1 \downarrow o_4 \rangle$~~
$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_4 \mid o_1 =^{?} o_4, o_1 \downarrow o_4 \rangle$
```
  }
```
$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_3 \mid o_1 =^{?} o_3, o_1 \downarrow o_3 \rangle$
```
  List ll = new List();
```
$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_3, \texttt{ll} = o_5 \mid o_1 =^{?} o_3, o_1 \downarrow o_3 \rangle$
```
  l.n = ll;
```
$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_3, \texttt{ll} = o_5 \mid o_1 =^{?} o_3, o_1 \downarrow o_3, o_3 \xrightarrow{\texttt{n}} o_5, \ldots \rangle$
```
  ll.v = x;
```
$\langle \texttt{this} = o_1, \texttt{x} = o_2, \texttt{l} = o_3, \texttt{ll} = o_5 \mid o_1 =^{?} o_3, o_1 \downarrow o_3, o_3 \xrightarrow{\texttt{n}} o_5, o_5 \xrightarrow{\texttt{v}} o_2, \ldots \rangle$
$\langle \ldots \mid \texttt{this} =^{?} \texttt{l}, \texttt{this} \downarrow \texttt{l}, \texttt{l} \xrightarrow{\texttt{n.v}} \texttt{x} \rangle$
```
}
```

Room for Improvement...

```
public void add(Object x) {
⟨... | ε⟩
  List l = this;
  while (l.n != null) {
    l = l.n;
  }
  List ll = new List();
  l.n = ll;
  ll.v = x;
⟨... | this =? l, this \/ l, l --n.v--> x⟩
}
```

Room for Improvement...

```
public void add(Object x) {
⟨... | ε⟩
  List l = this;
  while (l.n != null) {
    l = l.n;
  }
  List ll = new List();
  l.n = ll;
  ll.v = x;
⟨... | this =? l, this \/ l, l --n.v--> x⟩
}
```

- very complex domain – combined may- and must-analysis!

Room for Improvement...

```
public void add(Object x) {
⟨... | ε⟩
  List l = this;
  while (l.n != null) {
    l = l.n;
  }
  List ll = new List();
  l.n = ll;
  ll.v = x;
⟨... | this =? l, this \/ l, l --n.v--> x⟩
}
```

- very complex domain – combined may- and must-analysis!
- $\lor$ field-insensitive $\curvearrowright$ post-condition too coarse!

Room for Improvement...

```
public void add(Object x) {
⟨... | ε⟩
  List l = this;
  while (l.n != null) {
    l = l.n;
  }
  List ll = new List();
  l.n = ll;
  ll.v = x;
⟨... | this =? l, this \/ l, l --n.v--> x⟩
}
```

- very complex domain – combined may- and must-analysis!
- $\backslash/$ field-insensitive $\curvearrowright$ post-condition too coarse!
- pre-condition?

A Field-Sensitive May-Analysis

```
public void add(Object x) {
  List l = this;
  while (l.n != null) {
    l = l.n;
  }
  List ll = new List();
  l.n = ll;
  ll.v = x;

}
```

A Field-Sensitive May-Analysis

```
public void add(Object x) {
⟨ε⟩
  List l = this;

  while (l.n != null) {

    l = l.n;

  }

  List ll = new List();

  l.n = ll;

  ll.v = x;


}
```

A Field-Sensitive May-Analysis

```
public void add(Object x) {
⟨ε⟩
  List l = this;
⟨this -ε→ ←ε- l⟩
  while (l.n != null) {

    l = l.n;

  }

  List ll = new List();

  l.n = ll;

  ll.v = x;


}
```

A Field-Sensitive May-Analysis

```
public void add(Object x) {
⟨ε⟩
  List l = this;
⟨this -ϵ→ ←ϵ- l⟩
  while (l.n != null) {
⟨this -ϵ→ ←ϵ- l⟩
    l = l.n;


  }


  List ll = new List();

  l.n = ll;

  ll.v = x;



}
```

A Field-Sensitive May-Analysis

```
public void add(Object x) {
⟨ε⟩
  List l = this;
⟨this --ε--> <--ε-- l⟩
  while (l.n != null) {
⟨this --ε--> <--ε-- l⟩
    l = l.n;
⟨this --n--> <--ε-- l⟩
  }

  List ll = new List();

  l.n = ll;

  ll.v = x;


}
```

A Field-Sensitive May-Analysis

```
public void add(Object x) {
⟨ε⟩
  List l = this;
⟨this -ε→ε← l⟩ ⟨this -n?→ε← l⟩
  while (l.n != null) {
⟨this -ε→ε← l⟩
    l = l.n;
⟨this -n→ε← l⟩
  }

  List ll = new List();

  l.n = ll;

  ll.v = x;


}
```

A Field-Sensitive May-Analysis

```
public void add(Object x) {
⟨ε⟩
  List l = this;
⟨this -ε→←ε- l⟩  ⟨this -n?→←ε- l⟩
  while (l.n != null) {
⟨this -n?→←ε- l⟩
    l = l.n;
⟨this -n→←ε- l⟩
  }

  List ll = new List();

  l.n = ll;

  ll.v = x;


}
```

A Field-Sensitive May-Analysis

```
public void add(Object x) {
⟨ε⟩
  List l = this;
⟨this -ε→←ε- l⟩  ⟨this -n?→←ε- l⟩
  while (l.n != null) {
⟨this -n?→←ε- l⟩
    l = l.n;
⟨this -n.n?→←ε- l⟩
  }

  List ll = new List();

  l.n = ll;

  ll.v = x;


}
```

A Field-Sensitive May-Analysis

```
public void add(Object x) {
⟨ε⟩
  List l = this;
⟨this -ε→←ε- l⟩ ⟨this -n?→←ε- l⟩ ⟨this -n*→←ε- l⟩
  while (l.n != null) {
⟨this -n?→←ε- l⟩
    l = l.n;
⟨this -n.n?→←ε- l⟩
  }

  List ll = new List();

  l.n = ll;

  ll.v = x;


}
```

A Field-Sensitive May-Analysis

```
public void add(Object x) {
⟨ε⟩
  List l = this;
⟨this -ε→←ε- l⟩ ⟨this -n?→←ε- l⟩ ⟨this -n*→←ε- l⟩
  while (l.n != null) {
⟨this -n*→←ε- l⟩
    l = l.n;
⟨this -n.n?→←ε- l⟩
  }

  List ll = new List();

  l.n = ll;

  ll.v = x;


}
```

A Field-Sensitive May-Analysis

```
public void add(Object x) {
⟨ε⟩
  List l = this;
⟨this -ε→←ε- l⟩  ⟨this -n?→←ε- l⟩  ⟨this -n*→←ε- l⟩
  while (l.n != null) {
⟨this -n*→←ε- l⟩
    l = l.n;
⟨this -n.n*→←ε- l⟩
  }

  List ll = new List();

  l.n = ll;

  ll.v = x;


}
```

A Field-Sensitive May-Analysis

```
public void add(Object x) {
⟨ε⟩
  List l = this;
⟨this -ε→←ε- l⟩  ⟨this -n?→←ε- l⟩  ⟨this -n*→←ε- l⟩
  while (l.n != null) {
⟨this -n*→←ε- l⟩
    l = l.n;
⟨this -n.n*→←ε- l⟩
  }
⟨this -n*→←ε- l⟩
  List ll = new List();

  l.n = ll;

  ll.v = x;


}
```

A Field-Sensitive May-Analysis

```
public void add(Object x) {
⟨ε⟩
  List l = this;
⟨this -ϵ→ ←ϵ- l⟩ ⟨this -n?→ ←ϵ- l⟩ ⟨this -n*→ ←ϵ- l⟩
  while (l.n != null) {
⟨this -n*→ ←ϵ- l⟩
    l = l.n;
⟨this -n.n*→ ←ϵ- l⟩
  }
⟨this -n*→ ←ϵ- l⟩
  List ll = new List();
⟨this -n*→ ←ϵ- l⟩
  l.n = ll;


  ll.v = x;



}
```

A Field-Sensitive May-Analysis

```
public void add(Object x) {
⟨ε⟩
  List l = this;
⟨this -ϵ→ϵ- l⟩ ⟨this -n?→ϵ- l⟩ ⟨this -n*→ϵ- l⟩
  while (l.n != null) {
⟨this -n*→ϵ- l⟩
    l = l.n;
⟨this -n.n*→ϵ- l⟩
  }
⟨this -n*→ϵ- l⟩
  List ll = new List();
⟨this -n*→ϵ- l⟩
  l.n = ll;
⟨this -n*→ϵ- l, l -n→ϵ- ll⟩
  ll.v = x;



}
```

A Field-Sensitive May-Analysis

```
public void add(Object x) {
```
$\langle \varepsilon \rangle$
```
  List l = this;
```
$\langle \text{this} \xrightarrow{\epsilon}\xleftarrow{\epsilon} \text{l} \rangle$ $\langle \text{this} \xrightarrow{\text{n?}}\xleftarrow{\epsilon} \text{l} \rangle$ $\langle \text{this} \xrightarrow{\text{n}^*}\xleftarrow{\epsilon} \text{l} \rangle$
```
  while (l.n != null) {
```
$\langle \text{this} \xrightarrow{\text{n}^*}\xleftarrow{\epsilon} \text{l} \rangle$
```
    l = l.n;
```
$\langle \text{this} \xrightarrow{\text{n.n}^*}\xleftarrow{\epsilon} \text{l} \rangle$
```
  }
```
$\langle \text{this} \xrightarrow{\text{n}^*}\xleftarrow{\epsilon} \text{l} \rangle$
```
  List ll = new List();
```
$\langle \text{this} \xrightarrow{\text{n}^*}\xleftarrow{\epsilon} \text{l} \rangle$
```
  l.n = ll;
```
$\langle \text{this} \xrightarrow{\text{n}^*}\xleftarrow{\epsilon} \text{l}, \text{l} \xrightarrow{\text{n}}\xleftarrow{\epsilon} \text{ll} \rangle$
```
  ll.v = x;
```
$\langle \text{this} \xrightarrow{\text{n}^*}\xleftarrow{\epsilon} \text{l}, \text{l} \xrightarrow{\text{n}}\xleftarrow{\epsilon} \text{ll}, \text{ll} \xrightarrow{\text{v}}\xleftarrow{\epsilon} \text{x} \rangle$

```
}
```

A Field-Sensitive May-Analysis

```
public void add(Object x) {
```
$\langle \varepsilon \rangle$
```
  List l = this;
```
$\langle \text{this} \xrightarrow{\epsilon}\!\!\!\!\!\!\leftarrow^{\epsilon} \text{l} \rangle$ $\langle \text{this} \xrightarrow{n?}\!\!\!\!\!\!\leftarrow^{\epsilon} \text{l} \rangle$ $\langle \text{this} \xrightarrow{n^*}\!\!\!\!\!\!\leftarrow^{\epsilon} \text{l} \rangle$
```
  while (l.n != null) {
```
$\langle \text{this} \xrightarrow{n^*}\!\!\!\!\!\!\leftarrow^{\epsilon} \text{l} \rangle$
```
    l = l.n;
```
$\langle \text{this} \xrightarrow{n.n^*}\!\!\!\!\!\!\leftarrow^{\epsilon} \text{l} \rangle$
```
  }
```
$\langle \text{this} \xrightarrow{n^*}\!\!\!\!\!\!\leftarrow^{\epsilon} \text{l} \rangle$
```
  List ll = new List();
```
$\langle \text{this} \xrightarrow{n^*}\!\!\!\!\!\!\leftarrow^{\epsilon} \text{l} \rangle$
```
  l.n = ll;
```
$\langle \text{this} \xrightarrow{n^*}\!\!\!\!\!\!\leftarrow^{\epsilon} \text{l}, \text{l} \xrightarrow{n}\!\!\!\!\!\!\leftarrow^{\epsilon} \text{ll} \rangle$
```
  ll.v = x;
```
$\langle \text{this} \xrightarrow{n^*}\!\!\!\!\!\!\leftarrow^{\epsilon} \text{l}, \text{l} \xrightarrow{n}\!\!\!\!\!\!\leftarrow^{\epsilon} \text{ll}, \text{ll} \xrightarrow{v}\!\!\!\!\!\!\leftarrow^{\epsilon} \text{x} \rangle$

$\langle \text{this} \xrightarrow{n^+.v}\!\!\!\!\!\!\leftarrow^{\epsilon} \text{x} \rangle$
```
}
```

Room for Improvement...

```
public void add(Object x) {
⟨ε⟩
  List l = this;
  while (l.n != null) {
    l = l.n;
  }
  List ll = new List();
  l.n = ll;
  ll.v = x;
⟨this --n+.v--> <--ε-- x⟩
}
```

- very complex domain – combined may- and must-analysis!
- $\not\swarrow$ field-insensitive $\curvearrowright$ post-condition too coarse!
- pre-condition?

Room for Improvement...

```
public void add(Object x) {
⟨ε⟩
  List l = this;
  while (l.n != null) {
    l = l.n;
  }
  List ll = new List();
  l.n = ll;
  ll.v = x;
⟨this --n+.v--> <--ε-- x⟩
}
```

- ~~very complex domain – combined may- and must-analysis!~~
  - pure may-analysis
- ↯ field-insensitive ↷ post-condition too coarse!

- pre-condition?

Room for Improvement...

```
public void add(Object x) {
⟨ε⟩
  List l = this;
  while (l.n != null) {
    l = l.n;
  }
  List ll = new List();
  l.n = ll;
  ll.v = x;
⟨this --n+.v--> <--ε-- x⟩
}
```

- ~~very complex domain – combined may- and must-analysis!~~
  - pure may-analysis
- ~~$\not\swarrow$ field-insensitive $\curvearrowright$ post-condition too coarse!~~
  - $\xrightarrow{\pi}\xleftarrow{\tau}$ field-sensitive $\curvearrowright$ precise post-condition
- pre-condition?

Room for Improvement...

```
public void add(Object x) {
⟨ε⟩
  List l = this;
  while (l.n != null) {
    l = l.n;
  }
  List ll = new List();
  l.n = ll;
  ll.v = x;
⟨this --n+.v--> <--ε-- x⟩
}
```

- ~~very complex domain – combined may- and must-analysis!~~
  - pure may-analysis
- ~~$\lightning$ field-insensitive $\curvearrowright$ post-condition too coarse!~~
  - $\xrightarrow{\pi}\xleftarrow{\tau}$ field-sensitive $\curvearrowright$ precise post-condition
- pre-condition?

Room for Improvement...

```
public void add(Object x) {
⟨ε⟩
  List l = this;
  while (l.n != null) {
    l = l.n;
  }
  List ll = new List();
  l.n = ll;
  ll.v = x;
⟨this --n+.v--> <--ε-- x⟩
}
```

- ~~very complex domain – combined may- and must-analysis!~~
  - pure may-analysis
- ~~↯ field-insensitive ↷ post-condition too coarse!~~
  - $\xrightarrow{\pi}\xleftarrow{\tau}$ field-sensitive ↷ precise post-condition
- pre-condition?
  - if $\langle\varepsilon\rangle$ holds before add, $\langle \texttt{this} \xrightarrow{\texttt{n}^+.\texttt{v}}\xleftarrow{\epsilon} \texttt{x}\rangle$ holds after add

Room for Improvement...

```
public void add(Object x) {
⟨ε⟩
  List l = this;
  while (l.n != null) {
    l = l.n;
  }
  List ll = new List();
  l.n = ll;
  ll.v = x;
⟨this --n+.v--> <--ϵ x⟩
}
```

- ~~very complex domain – combined may- and must-analysis!~~
  - pure may-analysis
- ~~field-insensitive ↷ post-condition too coarse!~~
  - $\xrightarrow{\pi}\xleftarrow{\tau}$ field-sensitive ↷ precise post-condition
- pre-condition?
  - ~~if $\langle\varepsilon\rangle$ holds before add, $\langle \texttt{this} \xrightarrow{\texttt{n}^+.\texttt{v}}\xleftarrow{\epsilon} \texttt{x}\rangle$ holds after add~~
  - all side-effects of add are captured by $\langle \texttt{this} \xrightarrow{\texttt{n}^+.\texttt{v}}\xleftarrow{\epsilon} \texttt{x}\rangle$

Current State of Development

- implementation with support for...

Current State of Development

- implementation with support for...
  - non-recursive Java programs

Current State of Development

- implementation with support for...
  - non-recursive Java programs
  - acyclic data-structures

Current State of Development

- implementation with support for...
  - non-recursive Java programs
  - acyclic data-structures
  - method calls

Current State of Development

- implementation with support for...
  - non-recursive Java programs
  - acyclic data-structures
  - method calls
  - arrays

Current State of Development

- implementation with support for...
  - non-recursive Java programs
  - acyclic data-structures
  - method calls
  - arrays
  - static fields

Current State of Development

- implementation with support for...
  - non-recursive Java programs
  - acyclic data-structures
  - method calls
  - arrays
  - static fields
- work in progress

Current State of Development

- implementation with support for...
  - non-recursive Java programs
  - acyclic data-structures
  - method calls
  - arrays
  - static fields
- work in progress
  - recursion

Current State of Development

- implementation with support for...
  - non-recursive Java programs
  - acyclic data-structures
  - method calls
  - arrays
  - static fields
- work in progress
  - recursion
  - ~~loop unrolling~~

Current State of Development

- implementation with support for...
  - non-recursive Java programs
  - acyclic data-structures
  - method calls
  - arrays
  - static fields
- work in progress
  - recursion
  - ~~loop unrolling~~
- TODO

Current State of Development

- implementation with support for...
  - non-recursive Java programs
  - acyclic data-structures
  - method calls
  - arrays
  - static fields
- work in progress
  - recursion
  - ~~loop unrolling~~
- TODO
  - cyclic data-structures

Conclusion

- novel heap shape analysis

Conclusion

- novel heap shape analysis
  - pure may-analysis

Conclusion

- novel heap shape analysis
  - pure may-analysis
  - field-sensitive

Conclusion

- novel heap shape analysis
  - pure may-analysis
  - field-sensitive
  - modular

Conclusion

- novel heap shape analysis
  - pure may-analysis
  - field-sensitive
  - modular
- use cases

Conclusion

- novel heap shape analysis
  - pure may-analysis
  - field-sensitive
  - modular
- use cases
  - termination analysis

Conclusion

- novel heap shape analysis
  - pure may-analysis
  - field-sensitive
  - modular
- use cases
  - termination analysis
  - complexity analysis

Conclusion

- novel heap shape analysis
  - pure may-analysis
  - field-sensitive
  - modular
- use cases
  - termination analysis
  - complexity analysis
  - data-flow analysis

Conclusion

- novel heap shape analysis
  - pure may-analysis
  - field-sensitive
  - modular
- use cases
  - termination analysis
  - complexity analysis
  - data-flow analysis

Conclusion

- novel heap shape analysis
  - pure may-analysis
  - field-sensitive
  - modular
- use cases
  - termination analysis
  - complexity analysis
  - data-flow analysis

Demo!

Preliminary Results

Analyzed property: Are all data-structures trees/DAGs?

300 examples from TPDB 283×same result 8×`HashMap`

tree vs. DAG	1
tree vs. arbitrary	1
DAG vs. arbitrary	6
arbitrary vs. tree	1

Table: ∼2 weeks ago

Preliminary Results

Analyzed property: Are all data-structures trees/DAGs?

300 examples from TPDB

tree vs. DAG	1
tree vs. arbitrary	1
DAG vs. arbitrary	6
arbitrary vs. tree	1

Table: ∼2 weeks ago

286×same result

8×`HashMap`

DAG vs. arbitrary	4
arbitrary vs. tree	1
DAG vs. tree	1

Table: yesterday

Preliminary Results

Analyzed property: Are all data-structures trees/DAGs?

300 examples from TPDB

tree vs. DAG	1
tree vs. arbitrary	1
DAG vs. arbitrary	6
arbitrary vs. tree	1

Table: ∼2 weeks ago

286×same result

8×`HashMap`

DAG vs. arbitrary	4
arbitrary vs. tree	1
DAG vs. tree	1

Table: yesterday

Note

Preliminary Results

Analyzed property: Are all data-structures trees/DAGs?

300 examples from TPDB

tree vs. DAG	1
tree vs. arbitrary	1
DAG vs. arbitrary	6
arbitrary vs. tree	1

Table: ∼2 weeks ago

286×same result

8×`HashMap`

DAG vs. arbitrary	4
arbitrary vs. tree	1
DAG vs. tree	1

Table: yesterday

Note

- AProVE
  - 9 years of development
- new approach
  - 1 year of development

Preliminary Results

Analyzed property: Are all data-structures trees/DAGs?

300 examples from TPDB

tree vs. DAG	1
tree vs. arbitrary	1
DAG vs. arbitrary	6
arbitrary vs. tree	1

Table: ∼2 weeks ago

286×same result

8×`HashMap`

DAG vs. arbitrary	4
arbitrary vs. tree	1
DAG vs. tree	1

Table: yesterday

Note

- AProVE
  - 9 years of development
  - highly optimized for TPDB
- new approach
  - 1 year of development
  - unoptimized

Preliminary Results

Analyzed property: Are all data-structures trees/DAGs?

300 examples from TPDB

tree vs. DAG	1
tree vs. arbitrary	1
DAG vs. arbitrary	6
arbitrary vs. tree	1

Table: ∼2 weeks ago

286×same result

8×`HashMap`

DAG vs. arbitrary	4
arbitrary vs. tree	1
DAG vs. tree	1

Table: yesterday

Note

- AProVE
  - 9 years of development
  - highly optimized for TPDB
  - infers "binary" results
- new approach
  - 1 year of development
  - unoptimized
  - infers detailed results