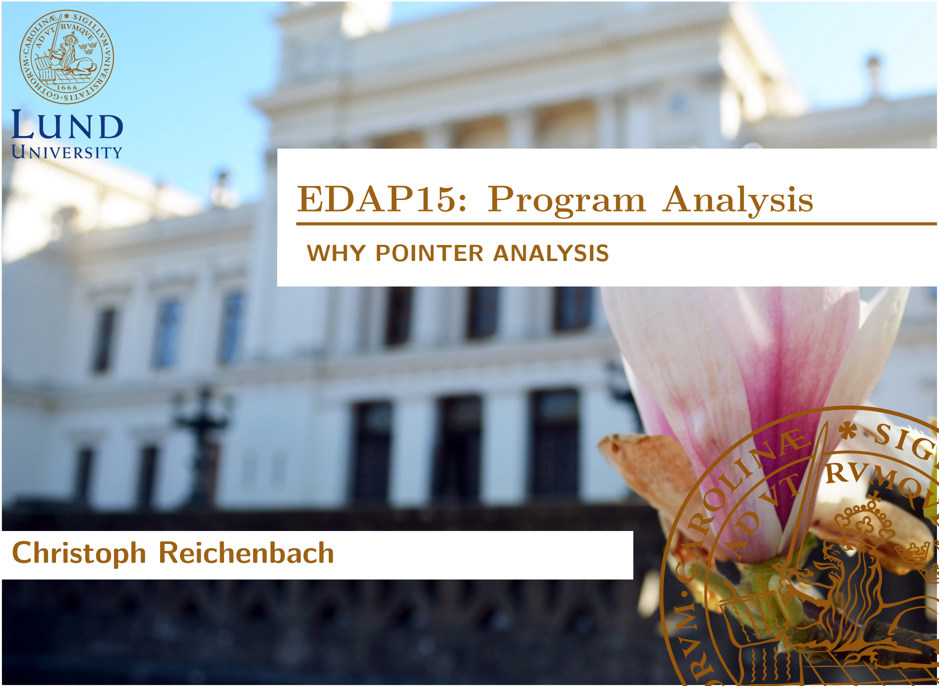


EDAP15: Program Analysis

WHY POINTER ANALYSIS

Christoph Reichenbach

Our Memory Modelling Until Now

- Our analyses so far have considered:
  - Static Variables
  - Local (stack-dynamic) Variables
  - (Stack-dynamic) parameters

Our Memory Modelling Until Now

- Our analyses so far have considered:
  - Static Variables
  - Local (stack-dynamic) Variables
  - (Stack-dynamic) parameters

Missing: heap variables!

Example Program

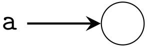
Example

```
a := new();
a.x := null;
b := a;
b.x := new();
a.x.y := 1;
c := new();
c.x := new();
c.x.x := a;
c := a.x;
// A
```

Example Program

Example

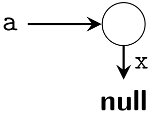
```
a := new();   // ⇐
a.x := null;
b := a;
b.x := new();
a.x.y := 1;
c := new();
c.x := new();
c.x.x := a;
c := a.x;
// A
```



Example Program

Example

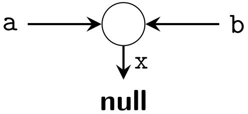
```
a := new();
a.x := null;  // ⇐
b := a;
b.x := new();
a.x.y := 1;
c := new();
c.x := new();
c.x.x := a;
c := a.x;
// A
```



Example Program

Example

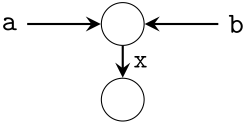
```
a := new();
a.x := null;
b := a;         // ⇐
b.x := new();
a.x.y := 1;
c := new();
c.x := new();
c.x.x := a;
c := a.x;
// A
```



Example Program

Example

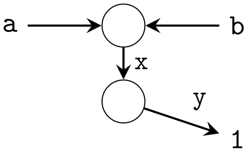
```
a := new();
a.x := null;
b := a;
b.x := new(); // ⇐
a.x.y := 1;
c := new();
c.x := new();
c.x.x := a;
c := a.x;
// A
```



Example Program

Example

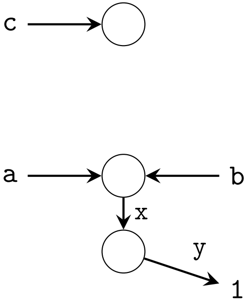
```
a := new();
a.x := null;
b := a;
b.x := new();
a.x.y := 1;    // ⇐
c := new();
c.x := new();
c.x.x := a;
c := a.x;
// A
```



Example Program

Example

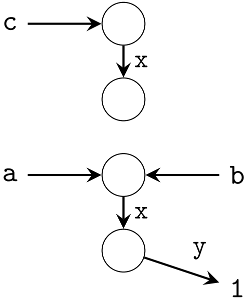
```
a := new();
a.x := null;
b := a;
b.x := new();
a.x.y := 1;
c := new();    // ⇐
c.x := new();
c.x.x := a;
c := a.x;
// A
```



Example Program

Example

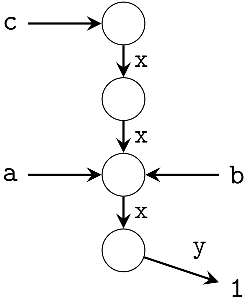
```
a := new();
a.x := null;
b := a;
b.x := new();
a.x.y := 1;
c := new();
c.x := new(); // ⇐
c.x.x := a;
c := a.x;
// A
```



Example Program

Example

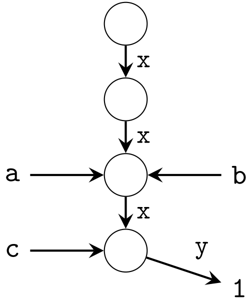
```
a := new();
a.x := null;
b := a;
b.x := new();
a.x.y := 1;
c := new();
c.x := new();
c.x.x := a;    // ⇐
c := a.x;
// A
```



Example Program

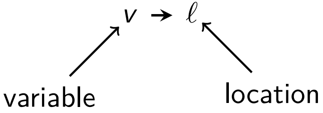
Example

```
a := new();
a.x := null;
b := a;
b.x := new();
a.x.y := 1;
c := new();
c.x := new();
c.x.x := a;
c := a.x;        // ⇐
// A
```



Concrete Heap Graph

“v points to ℓ”

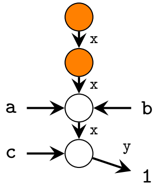


$v \rightarrow \ell$

variable

location

- Heap graph connects memory locations
- Represents all heap-allocated objects and their points-to relationships
- Edges labelled with field names
- Some objects not reachable from variables



Aliasing

Example

```
a := new();
a.x := null;
b := a;
b.x := new();
a.x.y := 1;
c := new();
c.x := new();
c.x.x := a;
c := a.x;
// A
```

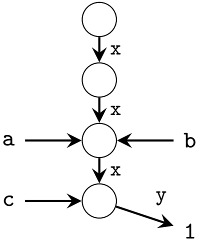
Aliases at // A:

- a and b represent the same object

⇒ a and b are *aliased*

$$a \overset{alias}{=\!=} b$$

- c and a.x are *aliased*



Aliasing

Example

```
a := new();
a.x := null;
b := a;
b.x := new();
a.x.y := 1;
c := new();
c.x := new();
c.x.x := a;
c := a.x;
// A
```

Aliases at // A:

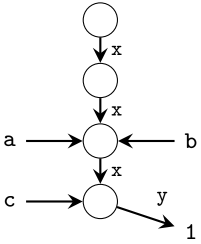
- a and b represent the same object

⇒ a and b are *aliased*

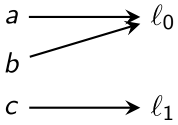
$$\mathtt{a} \overset{\textit{alias}}{=\!=} \mathtt{b}$$

⇒ a.x and b.x are *aliased*

- c and a.x and b.x are *aliased*



Pointer Analysis



- *Points-To Analysis*:
  - Analyse *heap usage*
  - Which *variables* may/must point to which *heap locations*?

$$a \rightarrow \ell_0$$

- *Alias Analysis*:
  - Analyse *address sharing*
  - Which *pair/set of variables* may/must point to the same address?

$$a \xlongequal{\mathit{alias}} b$$

Summary: Pointer Analysis

- Class of analyses to model dynamic heap allocation
- **Points-To Analysis**: computes mapping
  - From *variables*
  - To *pointees* (other variables)
  - More general than Alias Analysis
- **Alias Analysis**: computes
  - *Sharing information* between variables
  - Implicitly produced by points-to analysis

$$a \overset{alias}{=\!=} b \iff a \rightarrow \ell \leftarrow b$$