A CFL-Reachability Formulation of Callsite-Sensitive Pointer Analysis with Built-in On-the-Fly Call Graph Construction

Dongjie He[1,2], Jingbo Lu[1,3], and Jingling Xue[1]

SECTREND

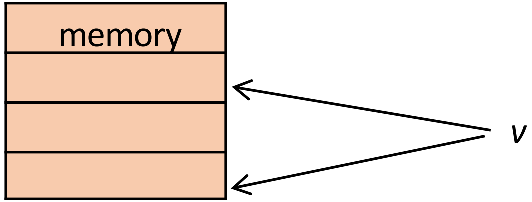
Contributions

- A new CFL-reachability formulation of *k*CFA (call-site sensitive pointer analysis) for Java, supporting on-the-fly callgraph construction.
  - $L_D \cap L_c \cap L_R$

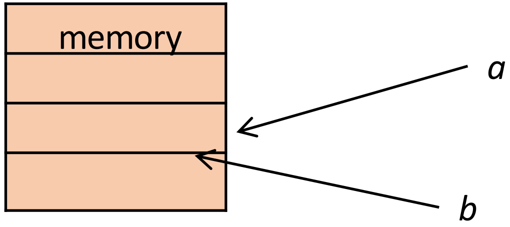
- P3Ctx: the first precision-preserving selective context sensitivity technique to accelerate *k*CFA.

A Brief Introduction to Pointer Analysis

- Programs (in C/C++/Java, …) are full of **pointers** or **references**
- Answer the following two problems



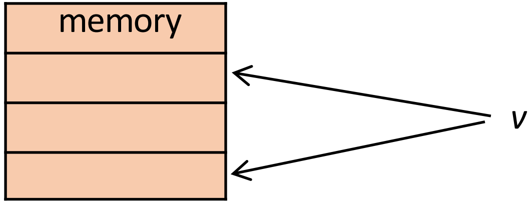
(1) What can a pointer point to?



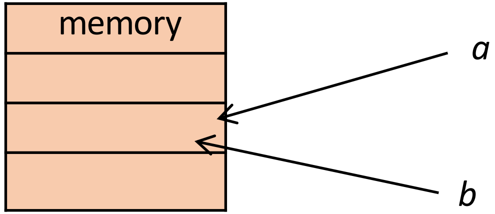
(2) Can *a* and *b* be aliases?

A Brief Introduction to Pointer Analysis

- Programs (in C/C++/Java, …) are full of **pointers** or **references**
- Answer the following two problems



(1) What can a pointer point to?



(2) Can *a* and *b* be aliases?

- Foundation of many Static Program Analysis

Compiler Optimization | Call-graph Construction | Program Verification | Program Understanding | Bug Detection …

- Implementations in many popular frameworks

Qilin Soot WALA (T. J. WATSON LIBRARIES FOR ANALYSIS) DOOP SVF

Motivation

- *k*CFA has two formulations:

Andersen-style Inclusion-based formulation for *k*CFA

$$\frac{\mathtt{x} = \mathbf{new}\ \mathtt{T}\ //\ O \quad ctx \in \mathtt{MethodCtx(M)}}{\langle O, \lceil ctx \rceil_{hk} \rangle \in \mathtt{PTS}(\mathtt{x}, ctx)} \ [\text{I-NEW}]$$

$$\frac{\mathtt{x} = \mathtt{y} \quad ctx \in \mathtt{MethodCtx(M)}}{\mathtt{PTS}(\mathtt{y}, ctx) \subseteq \mathtt{PTS}(\mathtt{x}, ctx)} \ [\text{I-ASSIGN}]$$

$$\frac{\mathtt{x} = \mathtt{y.f} \quad ctx \in \mathtt{MethodCtx(M)} \quad \langle O, htx \rangle \in \mathtt{PTS}(\mathtt{y}, ctx)}{\mathtt{PTS}(O.\mathtt{f}, htx) \subseteq \mathtt{PTS}(\mathtt{x}, ctx)} \ [\text{I-LOAD}]$$

$$\frac{\mathtt{x.f} = \mathtt{y} \quad ctx \in \mathtt{MethodCtx(M)} \quad \langle O, htx \rangle \in \mathtt{PTS}(\mathtt{x}, ctx)}{\mathtt{PTS}(\mathtt{y}, ctx) \subseteq \mathtt{PTS}(O.\mathtt{f}, htx)} \ [\text{I-STORE}]$$

$$\frac{\mathtt{x} = \mathtt{m}(a_1, \ldots, a_n)\ //\ \mathtt{c} \quad ctx \in \mathtt{MethodCtx(M)} \quad ctx' = \lceil \mathtt{c} :: ctx \rceil_k}{ctx' \in \mathtt{MethodCtx(m)} \quad \mathtt{PTS}(\mathtt{ret}^{\mathtt{m}}, ctx') \subseteq \mathtt{PTS}(\mathtt{x}, ctx) \quad \forall i \in [1, n] : \mathtt{PTS}(a_i, ctx) \subseteq \mathtt{PTS}(p_i^{\mathtt{m}}, ctx')} \ [\text{I-SCALL}]$$

$$\frac{\mathtt{x} = \mathtt{r.m}(a_1, \ldots, a_n)\ //\ \mathtt{c} \quad ctx \in \mathtt{MethodCtx(M)} \quad \langle O, htx \rangle \in \mathtt{PTS}(\mathtt{r}, ctx) \quad \mathtt{t} = \mathtt{DynTypeOf}(O) \quad \mathtt{m}' = \mathtt{dispatch(c, t)} \quad ctx' = \lceil \mathtt{c} :: ctx \rceil_k}{ctx' \in \mathtt{MethodCtx(m')} \quad \mathtt{PTS}(\mathtt{ret}^{\mathtt{m}'}, ctx') \subseteq \mathtt{PTS}(\mathtt{x}, ctx) \quad \langle O, htx \rangle \in \mathtt{PTS}(\mathtt{this}^{\mathtt{m}'}, ctx') \quad \forall i \in [1, n] : \mathtt{PTS}(a_i, ctx) \subseteq \mathtt{PTS}(p_i^{\mathtt{m}'}, ctx')} \ [\text{I-VCALL}]$$

Widely adopted in frameworks like Soot/Spark, WALA, DOOP, QILIN, …

Sridharan's CFL-reachability formulation for *k*CFA

$$L_{FC} = L_F \cap L_C$$

$$L_F: \begin{array}{rcl}
\textsf{flowsto} & \longrightarrow & \textsf{new}\ \textsf{flows}^* \\
\textsf{flows} & \longrightarrow & \textsf{assign} \mid \textsf{store[f]}\ \textsf{alias}\ \textsf{load[f]} \\
\textsf{alias} & \longrightarrow & \overline{\textsf{flowsto}}\ \textsf{flowsto} \\
\overline{\textsf{flowsto}} & \longrightarrow & \overline{\textsf{flows}}^*\ \overline{\textsf{new}} \\
\overline{\textsf{flows}} & \longrightarrow & \overline{\textsf{assign}} \mid \overline{\textsf{load[f]}}\ \textsf{alias}\ \overline{\textsf{store[f]}}
\end{array}$$

$$L_C: \begin{array}{rcl}
\textsf{realizable} & \longrightarrow & \textsf{exit}\ \textsf{entry} \\
\textsf{exit} & \longrightarrow & \textsf{exit}\ \textsf{balanced} \mid \textsf{exit}\ \check{c} \mid \epsilon \\
\textsf{entry} & \longrightarrow & \textsf{entry}\ \textsf{balanced} \mid \textsf{entry}\ \hat{c} \mid \epsilon \\
\textsf{balanced} & \longrightarrow & \textsf{balanced}\ \textsf{balanced} \mid \hat{c}\ \textsf{balanced}\ \check{c} \mid \epsilon
\end{array}$$

PAG construction rules

$$\frac{\mathtt{x} = \mathbf{new}\ \mathtt{T}\ //\ O}{O \xrightarrow{\textsf{new}} \mathtt{x}}\ [\text{P-NEW}] \qquad \frac{\mathtt{x} = \mathtt{y}}{\mathtt{y} \xrightarrow{\textsf{assign}} \mathtt{x}}\ [\text{P-ASSIGN}] \qquad \frac{\mathtt{x} = \mathtt{y.f}}{\mathtt{y} \xrightarrow{\textsf{load[f]}} \mathtt{x}}\ [\text{P-LOAD}]$$

$$\frac{\mathtt{x.f} = \mathtt{y}}{\mathtt{y} \xrightarrow{\textsf{store[f]}} \mathtt{x}}\ [\text{P-STORE}] \qquad \frac{\mathtt{x} = \mathtt{m}(a_1, \ldots, a_n)\ //\ \mathtt{c}}{\forall\, i \in [1, n] : a_i \xrightarrow[\hat{c}]{\textsf{assign}} p_i^{\mathtt{m}} \quad \mathtt{ret}^{\mathtt{m}} \xrightarrow[\check{c}]{\textsf{assign}} \mathtt{x}}\ [\text{P-SCALL}]$$

$$\frac{\mathtt{x} = \mathtt{r.m}(a_1, \ldots, a_n)\ //\ \mathtt{c} \quad \mathtt{m}' \text{ is a target of this callsite}}{\mathtt{r} \xrightarrow[\hat{c}]{\textsf{assign}} \mathtt{this}^{\mathtt{m}'} \quad \mathtt{ret}^{\mathtt{m}'} \xrightarrow[\check{c}]{\textsf{assign}} \mathtt{x} \quad \forall\, i \in [1, n] : a_i \xrightarrow[\hat{c}]{\textsf{assign}} p_i^{\mathtt{m}'}}\ [\text{P-VCALL}]$$

Inverse edge

Call dispatch

Motivation

- *k*CFA has two formulations:
  - Andersen-style inclusion-based formulation
  - Sridharan's CFL-reachability formulation $L_F \cap L_c$
- The two formulation are not equally precise.
  - The 2nd is less precise than the 1st.

Some examples to show precision loss in L_{FC}

Using a (Most Precise) Call Graph Constructed on the Fly or in Advance

```
class E {
  void foo(G p) {
    Object v = p.g;
}}
class F extends E {
  void foo(G q) { }
}
class G { Object g; }
G w = new G(); // G1
if (...) {
  E e1 = new E(); // E1
  w.g = e1;
} else {
  F f1 = new F(); // F1
  w.g = f1;
}
E x = w.g;
x.foo(null); // c
```

Spurious value flow in L_{FC} formulation

$$\mathtt{E1} \xrightarrow{\textsf{new}} \mathtt{e1} \xrightarrow{\textsf{store}[\mathsf{g}]} \mathtt{w} \xrightarrow{\overline{\textsf{new}}} \mathtt{G1} \xrightarrow{\textsf{new}} \mathtt{w} \xrightarrow{\textsf{load}[\mathsf{g}]} \mathtt{x} \xrightarrow[\hat{c}]{\textsf{assign}} \mathtt{this}^{\mathtt{E:foo()}}$$

$$\boxed{\mathtt{F1} \xrightarrow{\textsf{new}} \mathtt{f1} \xrightarrow{\textsf{store}[\mathsf{g}]} \mathtt{w} \xrightarrow{\overline{\textsf{new}}} \mathtt{G1} \xrightarrow{\textsf{new}} \mathtt{w} \xrightarrow{\textsf{load}[\mathsf{g}]} \mathtt{x} \xrightarrow[\hat{c}]{\textsf{assign}} \mathtt{this}^{\mathtt{E:foo()}}}$$

Some examples to show precision loss in L_{FC}

Using a (Most Precise) Call Graph Constructed on the Fly or in Advance

```
class E {
  void foo(G p) {
    Object v = p.g;
}}
class F extends E {
  void foo(G q) { }
}
class G { Object g; }
G w = new G(); // G1
if (...) {
  E e1 = new E(); // E1
  w.g = e1;
} else {
  F f1 = new F(); // F1
  w.g = f1;
}
E x = w.g;
x.foo(null); // c
```

Spurious value flow in L_{FC} formulation

$$\mathtt{E1} \xrightarrow{\text{new}} \mathtt{e1} \xrightarrow{\text{store[g]}} \mathtt{w} \xrightarrow{\overline{\text{new}}} \mathtt{G1} \xrightarrow{\text{new}} \mathtt{w} \xrightarrow{\text{load[g]}} \mathtt{x} \xrightarrow[\hat{c}]{\text{assign}} \mathtt{this}^{\mathtt{E:foo()}}$$

$$\boxed{\mathtt{F1} \xrightarrow{\text{new}} \mathtt{f1} \xrightarrow{\text{store[g]}} \mathtt{w} \xrightarrow{\overline{\text{new}}} \mathtt{G1} \xrightarrow{\text{new}} \mathtt{w} \xrightarrow{\text{load[g]}} \mathtt{x} \xrightarrow[\hat{c}]{\text{assign}} \mathtt{this}^{\mathtt{E:foo()}}}$$

L_{FC} also leads applications built upon it to lose precision, e.g., Selectx

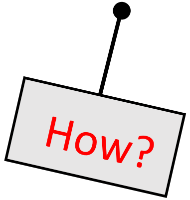
Jingbo Lu, Dongjie He, and Jingling Xue. Selective Context-Sensitivity for *k*CFA with CFL-Reachability. 28th International Static Analysis Symposium (SAS'21).

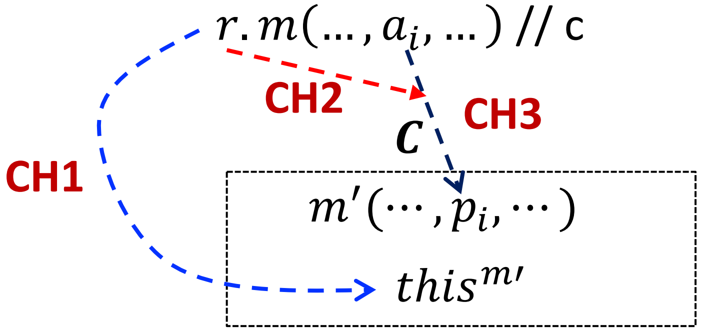
Motivation

- *k*CFA has two formulations:
  - Andersen-style inclusion-based formulation
  - Sridharan's CFL-reachability formulation $L_F \cap L_c$
- The two formulation are not equally precise.
  - The 2^{nd} is less precise than the 1^{st}.
- We aim to develop a precision-preserving selective context sensitivity technique to accelerate *k*CFA.

L_{DCR}: A New CFL-Reachability Formulation for kCFA

- L_{DCR} supports built-in on-the-fly call graph construction





Challenges

- **CH1**: how to pass r to $this^{m'}$?
- **CH2**: how to establish a CFL-reachability path from a_i to p_i under C while associating to r to trigger dynamic dispatch?
- **CH3**: How to pass a_i to p_i without changing context C?

$L_{DCR} = L_D \cap L_C \cap L_R$ (explained shortly)

We address these challenges by formulating $\boldsymbol{L_{DCR}}$ as $L_D \cap L_C \cap L_R$

$$
L_D: \quad
\begin{array}{rcl}
\mathsf{flowsto} & \longrightarrow & \mathsf{new}[\mathtt{t}]\ (\mathsf{flows} \mid \mathsf{dispatch}[\mathtt{t}])^* \\
\mathsf{flows} & \longrightarrow & \mathsf{assign} \mid \mathsf{store}[\mathtt{f}]\ \mathsf{alias}\ \mathsf{load}[\mathtt{f}] \\
\mathsf{alias} & \longrightarrow & \overline{\mathsf{flowsto}}\ \mathsf{flowsto} \\
\overline{\mathsf{flowsto}} & \longrightarrow & (\overline{\mathsf{dispatch}[\mathtt{t}]} \mid \overline{\mathsf{flows}})^*\ \overline{\mathsf{new}[\mathtt{t}]} \\
\overline{\mathsf{flows}} & \longrightarrow & \overline{\mathsf{assign}} \mid \overline{\mathsf{load}[\mathtt{f}]}\ \mathsf{alias}\ \overline{\mathsf{store}[\mathtt{f}]}
\end{array}
$$

$$
L_C: \quad
\begin{array}{rcl}
\mathsf{realizable} & \longrightarrow & \mathsf{exit}\ \mathsf{entry} \\
\mathsf{exit} & \longrightarrow & \mathsf{exit}\ \mathsf{balanced} \mid \mathsf{exit}\ \check{c} \mid \epsilon \\
\mathsf{entry} & \longrightarrow & \mathsf{entry}\ \mathsf{balanced} \mid \mathsf{entry}\ \hat{c} \mid \epsilon \\
\mathsf{balanced} & \longrightarrow & \mathsf{balanced}\ \mathsf{balanced} \mid \hat{c}\ \mathsf{balanced}\ \check{c} \mid \epsilon
\end{array}
$$

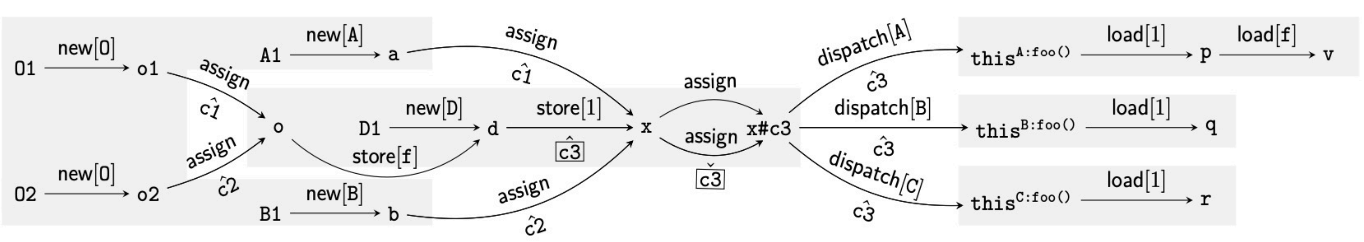
$$
L_R: \quad
\begin{array}{rcl}
\mathsf{recoveredCtx} & \longrightarrow & \mathsf{recoveredCtx}\ \hat{c} \mid \mathsf{recoveredCtx}\ \check{c} \mid \mathsf{recoveredCtx}\ \mathsf{siteRecovered} \mid \epsilon \\
\mathsf{siteRecovered} & \longrightarrow & \hat{\boxed{c}}\ \mathsf{ctxRecovered}\ \check{\boxed{c}} \\
\mathsf{ctxRecovered} & \longrightarrow & \mathsf{matched}\ \mathsf{ctxRecovered} \mid \mathsf{ctxRecovered}\ \mathsf{matched} \mid \check{c}\ \mathsf{ctxRecovered}\ \hat{c} \mid \epsilon \\
\mathsf{matched} & \longrightarrow & \mathsf{matched}\ \mathsf{matched} \mid \hat{c}\ \mathsf{matched}\ \check{c} \mid \mathsf{siteRecovered} \mid \epsilon
\end{array}
$$

New PAG Construction Rules for $L_{DCR} = L_D \cap L_C \cap L_R$

$$\frac{\texttt{x} = \textbf{new}\ \texttt{T}\ //\ O}{O \xrightarrow{\mathsf{new}[\texttt{T}]} \texttt{x}}\ \text{[C-NEW]} \qquad \frac{\texttt{M is an instance method}}{\texttt{this}^{\texttt{M}} \xrightarrow{\mathsf{load}[i]} p_i^{\texttt{M}}}\ \text{[C-PARAM]} \qquad \frac{\texttt{M is an instance method}}{\texttt{ret}^{\texttt{M}} \xrightarrow{\mathsf{store}[0]} \texttt{this}^{\texttt{M}}}\ \text{[C-RET]}$$

$$\frac{\texttt{x} = \texttt{r.m}(a_1, \ldots, a_n)\ //\ \texttt{c} \quad \texttt{t} <: \texttt{DeclTypeOf(r)} \quad \texttt{m}' = \texttt{dispatch(c,t)}}{\forall\, i \in [1,n] : a_i \xrightarrow[\boxed{\check{c}}]{\mathsf{store}[i]} \texttt{r} \quad \texttt{r} \xrightarrow[\boxed{\check{c}}]{\mathsf{load}[0]} \texttt{x} \quad \texttt{r} \xrightarrow{\mathsf{assign}} \texttt{r\#c} \quad \texttt{r} \xrightarrow[\boxed{\check{c}}]{\mathsf{assign}} \texttt{r\#c} \quad \texttt{r\#c} \xrightarrow[\hat{c}]{\mathsf{dispatch}[\texttt{t}]} \texttt{this}^{\texttt{m}'}}\ \text{[C-VCALL]}$$

```
class A {
  void foo(D p) {
    Object v = p.f;
  }
}
class B extends A {
  void foo(D q) { }
}
class C extends A {
  void foo(D r) {}
}
class D { Object f; }
class O { }
static void bar(A x, O o) {
  D d = new D(); // D1
  d.f = o;
  x.foo(d); // c3
}
static void main() {
  O o1 = new O(); // O1
  O o2 = new O(); // O2
  A a = new A(); // A1
  A b = new B(); // B1
  bar(a, o1); // c1
  bar(b, o2); // c2
}
```



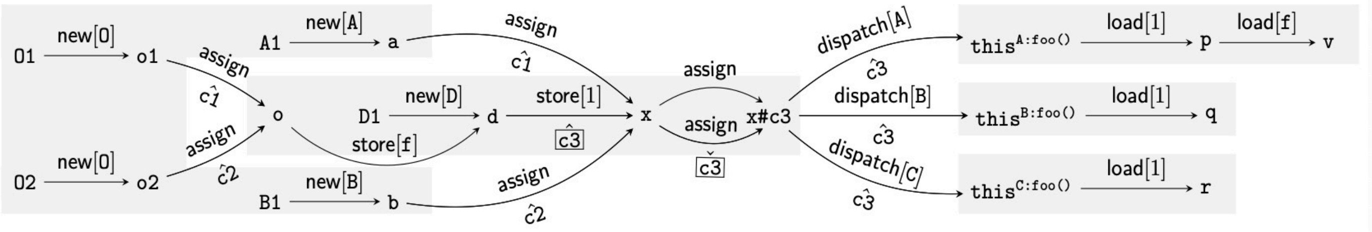
Address CH1 and CH2 with L_D and L_C

$$
L_D: \quad
\begin{array}{rcl}
\mathsf{flowsto} & \longrightarrow & \mathsf{new[t]}\ (\mathsf{flows} \mid \mathsf{dispatch[t]})^* \\
\mathsf{flows} & \longrightarrow & \mathsf{assign} \mid \mathsf{store[f]}\ \mathsf{alias}\ \mathsf{load[f]} \\
\mathsf{alias} & \longrightarrow & \overline{\mathsf{flowsto}}\ \mathsf{flowsto} \\
\overline{\mathsf{flowsto}} & \longrightarrow & (\overline{\mathsf{dispatch[t]}} \mid \overline{\mathsf{flows}})^*\ \overline{\mathsf{new[t]}} \\
\overline{\mathsf{flows}} & \longrightarrow & \overline{\mathsf{assign}} \mid \overline{\mathsf{load[f]}}\ \mathsf{alias}\ \overline{\mathsf{store[f]}}
\end{array}
$$

- address **CH1** by requiring new[t] match dispatch[t]

$$\mathrm{A1} \xrightarrow{\mathrm{new[A]}} a \xrightarrow{\mathrm{assign}} x \xrightarrow{\mathrm{assign}} x\#c3 \xrightarrow{\mathrm{dispatch[A]}} this^{\mathrm{A:foo()}}$$

$$\mathrm{B1} \xrightarrow{\mathrm{new[B]}} b \xrightarrow{\mathrm{assign}} x \xrightarrow{\mathrm{assign}} x\#c3 \xrightarrow{\mathrm{dispatch[A]}} this^{\mathrm{A:foo()}}$$



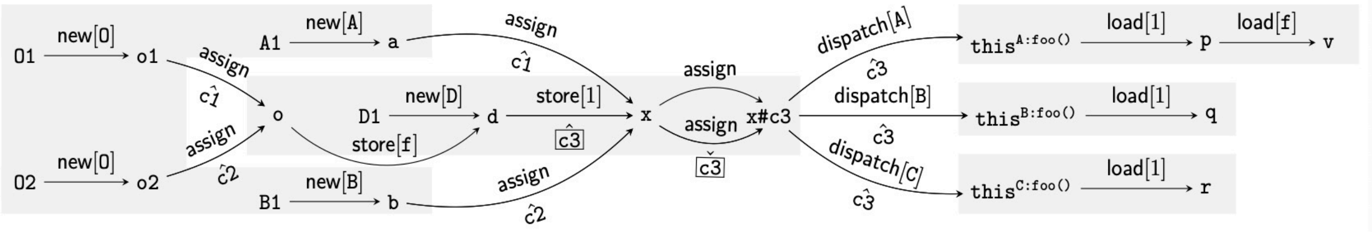
Address CH1 and CH2 with L_D and L_C

$$
L_D: \quad
\begin{array}{rcl}
\mathsf{flowsto} & \longrightarrow & \mathsf{new[t]}\ (\mathsf{flows} \mid \mathsf{dispatch[t]})^* \\
\mathsf{flows} & \longrightarrow & \mathsf{assign} \mid \mathsf{store[f]}\ \mathsf{alias}\ \mathsf{load[f]} \\
\mathsf{alias} & \longrightarrow & \overline{\mathsf{flowsto}}\ \mathsf{flowsto} \\
\overline{\mathsf{flowsto}} & \longrightarrow & (\overline{\mathsf{dispatch[t]}} \mid \overline{\mathsf{flows}})^*\ \overline{\mathsf{new[t]}} \\
\overline{\mathsf{flows}} & \longrightarrow & \overline{\mathsf{assign}} \mid \overline{\mathsf{load[f]}}\ \mathsf{alias}\ \overline{\mathsf{store[f]}}
\end{array}
$$

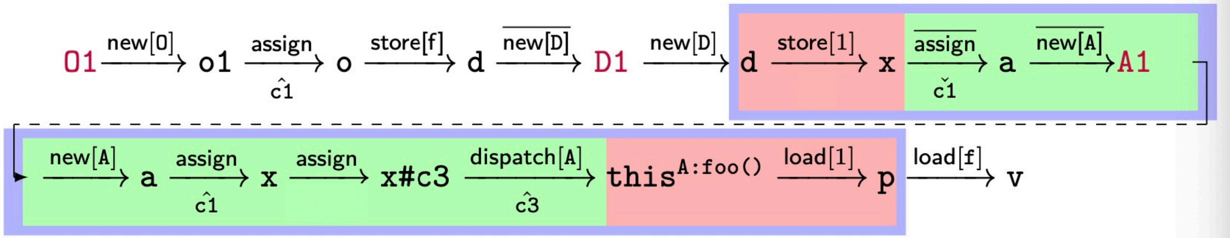
- address **CH1** by requiring new[t] match dispatch[t]

✓ $A1 \xrightarrow{\text{new[A]}} a \xrightarrow{\text{assign}} x \xrightarrow{\text{assign}} x\#c3 \xrightarrow{\text{dispatch[A]}} this^{\text{A:foo()}}$

✗ $B1 \xrightarrow{\text{new[B]}} b \xrightarrow{\text{assign}} x \xrightarrow{\text{assign}} x\#c3 \xrightarrow{\text{dispatch[A]}} this^{\text{A:foo()}}$



- address **CH2** by modeling parameter passing as stores and loads, and also enforce L_C



$$L_{DC} = L_D \cap L_C$$

O2 cannot flow to v

L_{DC} is not precise

Imprecision of L_{DC} caused by an incorrect dispatch site

```
static void main() {
  H h = new H(); // H1
  I i1 = new I(); // I1
  I i2 = new I(); // I2
  h.m(i1); // c4
  h.n(i2); // c5
}
class I {}
class H {
    void m(Object p) { ... }
    void n(Object q) { ... }
}
```

$$\mathtt{I1} \xrightarrow{\text{new}[\mathtt{I}]} \mathtt{i1} \xrightarrow{\text{store}[1]} \mathtt{h} \xrightarrow{\overline{\text{new}[\mathtt{H}]}} \mathtt{H1} \xrightarrow{\text{new}[\mathtt{H}]} \mathtt{h} \xrightarrow{\text{assign}} \mathtt{h\#c4} \xrightarrow[\hat{c4}]{\text{dispatch}[\mathtt{H}]} \mathtt{this}^{\mathtt{m}} \xrightarrow{\text{load}[1]} \mathtt{p}$$

$$\mathtt{I1} \xrightarrow{\text{new}[\mathtt{I}]} \mathtt{i1} \xrightarrow{\text{store}[1]} \mathtt{h} \xrightarrow{\overline{\text{new}[\mathtt{H}]}} \mathtt{H1} \xrightarrow{\text{new}[\mathtt{H}]} \mathtt{h} \xrightarrow{\text{assign}} \mathtt{h\#c5} \xrightarrow[\hat{c5}]{\text{dispatch}[\mathtt{H}]} \mathtt{this}^{\mathtt{n}} \xrightarrow{\text{load}[1]} \mathtt{q}$$

Spurious

L_{DC} is not precise

Imprecision of L_{DC} caused by an incorrect dispatch context

```
static void main() {
  J j1 = new J(); // J1
  K k1 = new K(); // K1
  K k2 = new K(); // K2
  K v1 = wid(j1, k1); // c6
  K v2 = wid(j1, k2); // c7
}
class K { }
class J {
  K id(K p) {
     return p;
}}
static K wid(J j, K k) {
  K v = j.id(k); // c8
  return v;
}
```

$$\mathtt{K1} \xrightarrow{\mathsf{new}[K]} \mathtt{k1} \xrightarrow[\hat{c}6]{\mathsf{assign}} \mathtt{k} \xrightarrow[\boxed{\hat{c}8}]{\mathsf{store}[1]} \mathtt{j} \xrightarrow[\check{c}6]{\overline{\mathsf{assign}}} \mathtt{j1} \xrightarrow{\overline{\mathsf{new}[J]}} \mathtt{J1} \xrightarrow{\mathsf{new}[J]} \mathtt{j1} \xrightarrow[\hat{c}6]{\mathsf{assign}} \mathtt{j} \xrightarrow[\boxed{\check{c}8}]{\mathsf{assign}} \mathtt{j\#c8} \xrightarrow[\hat{c}8]{\mathsf{dispatch}[J]} \mathtt{this}^{\mathtt{id}} \xrightarrow{\mathsf{load}[1]}$$

$$\mathtt{p} \xrightarrow{\mathsf{store}[0]} \mathtt{this}^{\mathtt{id}} \xrightarrow[\check{c}8]{\overline{\mathsf{dispatch}[J]}} \mathtt{j\#c8} \xrightarrow[\boxed{\hat{c}8}]{\overline{\mathsf{assign}}} \mathtt{j} \xrightarrow[\check{c}6]{\overline{\mathsf{assign}}} \mathtt{j1} \xrightarrow{\overline{\mathsf{new}[J]}} \mathtt{J1} \xrightarrow{\mathsf{new}[J]} \mathtt{j1} \xrightarrow[\hat{c}6]{\mathsf{assign}} \mathtt{j} \xrightarrow[\boxed{\check{c}8}]{\mathsf{load}[0]} \mathtt{v} \xrightarrow[\check{c}6]{\mathsf{assign}} \mathtt{v1}$$

$$\mathtt{K1} \xrightarrow{\mathsf{new}[K]} \mathtt{k1} \xrightarrow[\hat{c}6]{\mathsf{assign}} \mathtt{k} \xrightarrow[\boxed{\hat{c}8}]{\mathsf{store}[1]} \mathtt{j} \xrightarrow[\check{c}6]{\overline{\mathsf{assign}}} \mathtt{j1} \xrightarrow{\overline{\mathsf{new}[J]}} \mathtt{J1} \xrightarrow{\mathsf{new}[J]} \mathtt{j1} \xrightarrow[\hat{c}7]{\mathsf{assign}} \mathtt{j} \xrightarrow[\boxed{\check{c}8}]{\mathsf{assign}} \mathtt{j\#c8} \xrightarrow[\hat{c}8]{\mathsf{dispatch}[J]} \mathtt{this}^{\mathtt{id}} \xrightarrow{\mathsf{load}[1]}$$

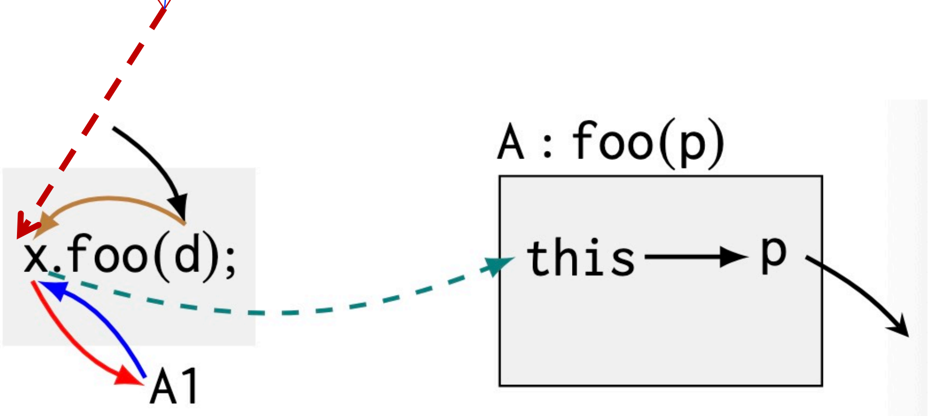
$$\mathtt{p} \xrightarrow{\mathsf{store}[0]} \mathtt{this}^{\mathtt{id}} \xrightarrow[\check{c}8]{\overline{\mathsf{dispatch}[J]}} \mathtt{j\#c8} \xrightarrow[\boxed{\hat{c}8}]{\overline{\mathsf{assign}}} \mathtt{j} \xrightarrow[\check{c}7]{\overline{\mathsf{assign}}} \mathtt{j1} \xrightarrow{\overline{\mathsf{new}[J]}} \mathtt{J1} \xrightarrow{\mathsf{new}[J]} \mathtt{j1} \xrightarrow[\hat{c}7]{\mathsf{assign}} \mathtt{j} \xrightarrow[\boxed{\check{c}8}]{\mathsf{load}[0]} \mathtt{v} \xrightarrow[\check{c}7]{\mathsf{assign}} \mathtt{v2}$$

Spurious

Address CH3 by also enforcing L_R

$$\text{recoveredCtx} \longrightarrow \text{recoveredCtx}\ \hat{c} \mid \text{recoveredCtx}\ \check{c} \mid \text{recoveredCtx siteRecovered} \mid \epsilon$$

$$L_R: \begin{array}{rcl} \text{siteRecovered} & \longrightarrow & \hat{c}\ \boxed{\text{ctxRecovered}}\ \check{c} \\ \text{ctxRecovered} & \longrightarrow & \text{matched ctxRecovered} \mid \text{ctxRecovered matched} \mid \check{c}\ \text{ctxRecovered}\ \hat{c} \mid \epsilon \\ \text{matched} & \longrightarrow & \text{matched matched} \mid \hat{c}\ \text{matched}\ \check{c} \mid \text{siteRecovered} \mid \epsilon \end{array}$$



Ensure come back to the same callsite under the same context.

L_{DCR} is precise

Eliminate imprecision of L_{DC} caused by an incorrect dispatch site

```
static void main() {
  H h = new H(); // H1
  I i1 = new I(); // I1
  I i2 = new I(); // I2
  h.m(i1); // c4
  h.n(i2); // c5
}
class I {}
class H {
  void m(Object p) { ... }
  void n(Object q) { ... }
}
```

$$\mathtt{I1} \xrightarrow{\text{new}[\mathtt{I}]} \mathtt{i1} \xrightarrow[\boxed{\hat{\mathtt{c4}}}]{\text{store}[1]} \mathtt{h} \xrightarrow{\overline{\text{new}[\mathtt{H}]}} \mathtt{H1} \xrightarrow{\text{new}[\mathtt{H}]} \mathtt{h} \xrightarrow[\boxed{\check{\mathtt{c4}}}]{\text{assign}} \mathtt{h\#c4} \xrightarrow[\hat{\mathtt{c4}}]{\text{dispatch}[\mathtt{H}]} \mathtt{this}^{\mathtt{m}} \xrightarrow{\text{load}[1]} \mathtt{p}$$

$$\mathtt{I1} \xrightarrow{\text{new}[\mathtt{I}]} \mathtt{i1} \xrightarrow[\boxed{\hat{\mathtt{c4}}}]{\text{store}[1]} \mathtt{h} \xrightarrow{\overline{\text{new}[\mathtt{H}]}} \mathtt{H1} \xrightarrow{\text{new}[\mathtt{H}]} \mathtt{h} \xrightarrow[\boxed{\check{\mathtt{c5}}}]{\text{assign}} \mathtt{h\#c5} \xrightarrow[\hat{\mathtt{c5}}]{\text{dispatch}[\mathtt{H}]} \mathtt{this}^{\mathtt{n}} \xrightarrow{\text{load}[1]} \mathtt{q}$$

Infeasible as $\boxed{\hat{\mathtt{c4}}}$ does not match $\boxed{\check{\mathtt{c5}}}$

L_{DCR} is precise

Eliminate imprecision of L_{DC} caused by an incorrect dispatch context

```
static void main() {
  J j1 = new J(); // J1
  K k1 = new K(); // K1
  K k2 = new K(); // K2
  K v1 = wid(j1, k1); // c6
  K v2 = wid(j1, k2); // c7
}
class K { }
class J {
  K id(K p) {
     return p;
}}
static K wid(J j, K k) {
  K v = j.id(k); // c8
  return v;
}
```

$$\mathtt{K1} \xrightarrow{\mathsf{new[K]}} \mathtt{k1} \xrightarrow[\hat{c}6]{\mathsf{assign}} \mathtt{k} \xrightarrow[\boxed{\hat{c}8}]{\mathsf{store[1]}} \mathtt{j} \xrightarrow[\check{c}6]{\overline{\mathsf{assign}}} \mathtt{j1} \xrightarrow{\overline{\mathsf{new[J]}}} \mathtt{J1} \xrightarrow{\mathsf{new[J]}} \mathtt{j1} \xrightarrow[\hat{c}6]{\mathsf{assign}} \mathtt{j} \xrightarrow[\boxed{\check{c}8}]{\mathsf{assign}} \mathtt{j\#c8} \xrightarrow[\hat{c}8]{\mathsf{dispatch[J]}} \mathtt{this}^{\mathtt{id}} \xrightarrow{\mathsf{load[1]}}$$

$$\mathtt{p} \xrightarrow{\mathsf{store[0]}} \mathtt{this}^{\mathtt{id}} \xrightarrow[\check{c}8]{\overline{\mathsf{dispatch[J]}}} \mathtt{j\#c8} \xrightarrow[\boxed{\hat{c}8}]{\overline{\mathsf{assign}}} \mathtt{j} \xrightarrow[\check{c}6]{\overline{\mathsf{assign}}} \mathtt{j1} \xrightarrow{\overline{\mathsf{new[J]}}} \mathtt{J1} \xrightarrow{\mathsf{new[J]}} \mathtt{j1} \xrightarrow[\hat{c}6]{\mathsf{assign}} \mathtt{j} \xrightarrow[\boxed{\check{c}8}]{\mathsf{load[0]}} \mathtt{v} \xrightarrow[\check{c}6]{\mathsf{assign}} \mathtt{v1}$$

$$\mathtt{K1} \xrightarrow{\mathsf{new[K]}} \mathtt{k1} \xrightarrow[\hat{c}6]{\mathsf{assign}} \mathtt{k} \xrightarrow[\boxed{\hat{c}8}]{\mathsf{store[1]}} \mathtt{j} \xrightarrow[\check{c}6]{\overline{\mathsf{assign}}} \mathtt{j1} \xrightarrow{\overline{\mathsf{new[J]}}} \mathtt{J1} \xrightarrow{\mathsf{new[J]}} \mathtt{j1} \xrightarrow[\hat{c}7]{\mathsf{assign}} \mathtt{j} \xrightarrow[\boxed{\check{c}8}]{\mathsf{assign}} \mathtt{j\#c8} \xrightarrow[\hat{c}8]{\mathsf{dispatch[J]}} \mathtt{this}^{\mathtt{id}} \xrightarrow{\mathsf{load[1]}}$$

$$\mathtt{p} \xrightarrow{\mathsf{store[0]}} \mathtt{this}^{\mathtt{id}} \xrightarrow[\check{c}8]{\overline{\mathsf{dispatch[J]}}} \mathtt{j\#c8} \xrightarrow[\boxed{\hat{c}8}]{\overline{\mathsf{assign}}} \mathtt{j} \xrightarrow[\check{c}7]{\overline{\mathsf{assign}}} \mathtt{j1} \xrightarrow{\overline{\mathsf{new[J]}}} \mathtt{J1} \xrightarrow{\mathsf{new[J]}} \mathtt{j1} \xrightarrow[\hat{c}7]{\mathsf{assign}} \mathtt{j} \xrightarrow[\boxed{\check{c}8}]{\mathsf{load[0]}} \mathtt{v} \xrightarrow[\check{c}7]{\mathsf{assign}} \mathtt{v2}$$

Infeasible as $\boxed{\hat{c}8}\ \check{c}6\ \hat{c}7\ \boxed{\check{c}8}$ does not belong to L_R

L_{DCR}

- A new CFL-reachability formulation for *k*CFA with built-in callgraph construction

- Demonstrating that *k*CFA is a special kind of context-sensitive language, i.e., the intersection of multiple CFLs.

P3Ctx: An L_{DCR}-based technique for accelerating *k*CFA

Selective Context-Sensitivity

Only apply context-sensitivity to **precision-critical** variables/objects

Criterion of precision critical nodes

$$\texttt{CS-C1} : L_F(p_{O,n,v}) \in L_F$$

$$\texttt{CS-C2} : L_C(p_{O,n}) \in L_C \wedge L_C(p_{n,v}) \in L_C$$

$$\texttt{CS-C3} : L_C^{\mathsf{en}}(p_{O,n}) \neq \epsilon \wedge L_C^{\mathsf{ex}}(p_{n,v}) \neq \epsilon$$

Like L_{FC}, L_{DCR} is also undecidable. Need Regularization.

Regularize L_{DCR}

Regularize L_R to L_R^r:

$$L_R: \begin{array}{rcl} \text{recoveredCtx} & \longrightarrow & \text{recoveredCtx } \hat{c} \mid \text{recoveredCtx } \check{c} \mid \text{recoveredCtx siteRecovered} \mid \epsilon \\ \text{siteRecovered} & \longrightarrow & \boxed{\hat{c}} \text{ ctxRecovered } \boxed{\check{c}} \\ \text{ctxRecovered} & \longrightarrow & \text{matched ctxRecovered} \mid \text{ctxRecovered matched} \mid \check{c} \text{ ctxRecovered } \hat{c} \mid \epsilon \\ \text{matched} & \longrightarrow & \text{matched matched} \mid \hat{c} \text{ matched } \check{c} \mid \text{siteRecovered} \mid \epsilon \end{array}$$

$$\Downarrow$$

$$L_R^r: \quad \text{recoveredCtx} \longrightarrow \text{recoveredCtx } \hat{c} \mid \text{recoveredCtx } \check{c} \mid \text{recoveredCtx } \boxed{\hat{c}} \mid \text{recoveredCtx } \boxed{\check{c}} \mid \epsilon$$

$$L_D \cap L_C \cap L_R^r = L_D \cap L_C = L_{DC}$$

Regularize L_{DCR}

Regularize L_D to L_D^r:

L_D:

$$
\begin{array}{lcl}
\text{flowsto} & \longrightarrow & \text{new}[t]\ (\text{flows} \mid \text{dispatch}[t])^* \\
\text{flows} & \longrightarrow & \text{assign} \mid \text{store}[f]\ \text{alias}\ \text{load}[f] \\
\text{alias} & \longrightarrow & \overline{\text{flowsto}}\ \text{flowsto} \\
\overline{\text{flowsto}} & \longrightarrow & (\overline{\text{dispatch}[t]} \mid \overline{\text{flows}})^*\ \overline{\text{new}[t]} \\
\overline{\text{flows}} & \longrightarrow & \overline{\text{assign}} \mid \overline{\text{load}[f]}\ \text{alias}\ \overline{\text{store}[f]}
\end{array}
$$

disregard field sensitivity

$$
\begin{array}{lcl}
\text{flowsto} & \longrightarrow & \text{new}\ (\text{flows} \mid \text{dispatch})^* \\
\text{flows} & \longrightarrow & \text{assign} \mid \text{store}\ \overline{\text{flowsto}}\ \text{flowsto}\ \text{load} \\
\overline{\text{flowsto}} & \longrightarrow & (\overline{\text{dispatch}} \mid \overline{\text{flows}})^*\ \overline{\text{new}} \\
\overline{\text{flows}} & \longrightarrow & \overline{\text{assign}} \mid \overline{\text{load}}\ \overline{\text{flowsto}}\ \text{flowsto}\ \overline{\text{store}}
\end{array}
$$

interpret **dispatch** as **assign**

$$
\begin{array}{lcl}
\text{flowsto} & \longrightarrow & \text{new}\ \text{flows}^* \\
\overline{\text{flowsto}} & \longrightarrow & \overline{\text{flows}}^*\ \overline{\text{new}} \\
\text{flows} & \longrightarrow & \text{assign} \mid \text{store}\ \overline{\text{flowsto}}\ \text{flowsto}\ \text{load} \\
\overline{\text{flows}} & \longrightarrow & \overline{\text{assign}} \mid \overline{\text{load}}\ \overline{\text{flowsto}}\ \text{flowsto}\ \overline{\text{store}}
\end{array}
$$

no longer require **load** to match **store**

$$
\begin{array}{lcl}
\text{flowsto} & \longrightarrow & \text{new}\ \text{flows}^* \\
\overline{\text{flowsto}} & \longrightarrow & \overline{\text{flows}}^*\ \overline{\text{new}} \\
\text{flows} & \longrightarrow & \text{assign} \mid \text{store}\ \overline{\text{flowsto}}\ \text{flowsto} \\
\overline{\text{flows}} & \longrightarrow & \overline{\text{assign}} \mid \overline{\text{flowsto}}\ \text{flowsto}\ \overline{\text{store}}
\end{array}
$$

no longer distinguish **store** and $\overline{\textbf{store}}$

L_D^r:

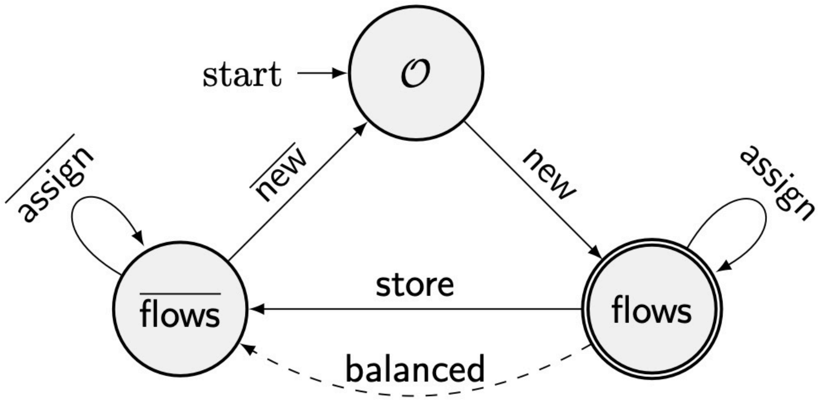
$$
\begin{array}{lcl}
\text{flowsto} & \longrightarrow & \text{new}\ \text{flows}^* \\
\overline{\text{flowsto}} & \longrightarrow & \overline{\text{flows}}^*\ \overline{\text{new}} \\
\text{flows} & \longrightarrow & \text{assign} \mid \text{store}\ \overline{\text{assign}}^*\ \overline{\text{new}}\ \text{new} \\
\overline{\text{flows}} & \longrightarrow & \overline{\text{assign}} \mid \overline{\text{new}}\ \text{new}\ \text{assign}^*\ \text{store}
\end{array}
$$

Regularize L_{DCR}

We keep L_C unchanged.

$$L_{DCR} = L_D \cap L_C \cap L_R \Longrightarrow L_D^r \cap L_C \cap L_R^r = L_D^r \cap L_C$$

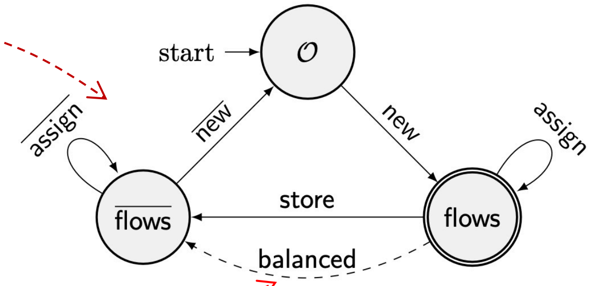
L_D^r is equivalent to the following DFA (Deterministic Finite Automata):



P3Ctx: An L_{DCR}-based technique for accelerating *k*CFA

L_D^r

$$\begin{aligned}
&\texttt{CS-C1}: L_F(p_{O,n,v}) \in L_F \\
&\texttt{CS-C2}: L_C(p_{O,n}) \in L_C \wedge L_C(p_{n,v}) \in L_C \\
&\texttt{CS-C3}: L_C^{\mathsf{en}}(p_{O,n}) \neq \epsilon \wedge L_C^{\mathsf{ex}}(p_{n,v}) \neq \epsilon
\end{aligned}$$

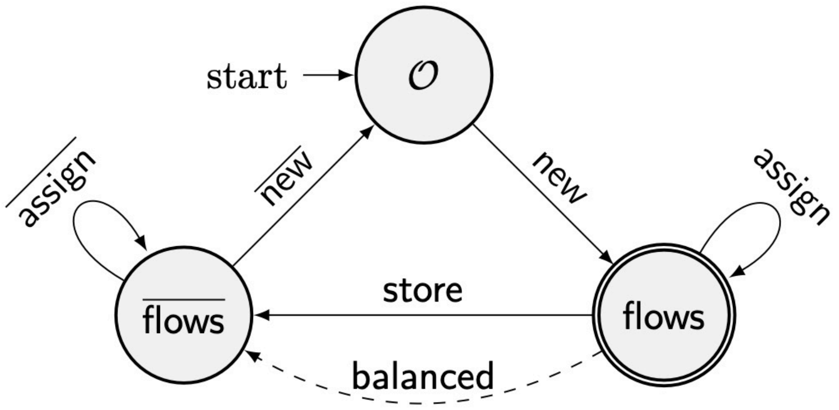


Verify selection criterion CS-C1, CS-C2, and CS-C3 over-approximately

- Replace L_F with L_D^r
- Use balanced edges to ensure **CS-C2**
- Assume every ***this*** variable are not **null** (some objects can flow to it).
- Replace CS-C1 and CS-C3 with the following condition: (n in method M)

$$\langle \texttt{this}^{\texttt{M}}, \mathsf{flows} \rangle \rightarrowtail^{+} \langle n, q \rangle \rightarrowtail^{+} \langle \texttt{this}^{\texttt{M}}, \overline{\mathsf{flows}} \rangle$$

P3Ctx: An L_{DCR}-based technique for accelerating *k*CFA



$$\langle \texttt{this}^{\texttt{M}}, \mathsf{flows} \rangle \rightarrowtail^{+} \langle n, q \rangle \rightarrowtail^{+} \langle \texttt{this}^{\texttt{M}}, \overline{\mathsf{flows}} \rangle$$

$\Downarrow$ equivalent to

$$n \in R(\mathcal{O}) \quad \vee \quad n \in R(\mathsf{flows}) \cap R(\overline{\mathsf{flows}})$$

where $n \in R(s)$ means $\langle \text{this}^{\text{M}}, \text{flows} \rangle \rightarrowtail^{+} \langle n, s \rangle$ for some M.

The DFA has two properties:

- **PROP-O**. Let O be an object created in a method M. Then $\langle \texttt{this}^{\texttt{M}}, \mathsf{flows} \rangle \rightarrowtail^{+} \langle O, \mathcal{O} \rangle \iff \langle O, \mathcal{O} \rangle \rightarrowtail^{+} \langle \texttt{this}^{\texttt{M}}, \overline{\mathsf{flows}} \rangle$ always holds.
- **PROP-V**. Let v be a variable defined in a method M. Then $\langle \texttt{this}^{\texttt{M}}, \mathsf{flows} \rangle \rightarrowtail^{+} \langle v, q \rangle \iff \langle v, \overline{q} \rangle \rightarrowtail^{+} \langle \texttt{this}^{\texttt{M}}, \overline{\mathsf{flows}} \rangle$ always holds, where $q \in \{\mathsf{flows}, \overline{\mathsf{flows}}\}$ (since v is a variable).

P3Ctx: An L_{DCR}-based technique for accelerating *k*CFA

To verify $n \in R(\mathcal{O}) \quad \vee \quad n \in R(\mathsf{flows}) \cap R(\overline{\mathsf{flows}})$

We compute R using following rules over PAG $= (N, E)$

$$\frac{n_1 \xrightarrow[\hat{c}]{} \mathtt{this}^{\mathtt{M}} \in E}{\mathtt{this}^{\mathtt{M}} \in R(\mathsf{flows}) \quad \mathsf{flows} \in R^{-1}(\mathtt{this}^{\mathtt{M}})} \quad \text{[F-INIT]}$$

$$\frac{n_1 \xrightarrow{\ell} n_2 \in E \quad q_1 \in R^{-1}(n_1) \quad \delta(q_1, \ell) = q_2}{n_2 \in R(q_2) \quad q_2 \in R^{-1}(n_2)} \quad \text{[F-PROPA]}$$

$$\frac{n_1 \xrightarrow[\hat{c}]{} \mathtt{this}^{\mathtt{M}} \in E \quad \mathtt{this}^{\mathtt{M}} \xrightarrow[\check{c}]{} n_2 \in E \quad \overline{\mathsf{flows}} \in R^{-1}(\mathtt{this}^{\mathtt{M}})}{n_1 \xrightarrow{\mathsf{balanced}} n_2 \in E} \quad \text{[F-SUM]}$$

Theorem (Precision-preserving): kCFA produces exactly the same points-to information when performed with selective context-sensitivity under P3Ctx.

Implementation

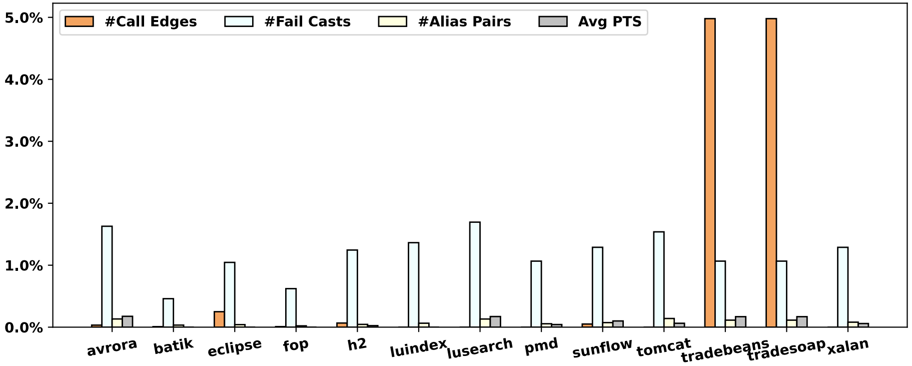
P3Ctx is implemented on top of SelectX in about 500 LOC.

Artifact (including source): https://zenodo.org/records/11061892

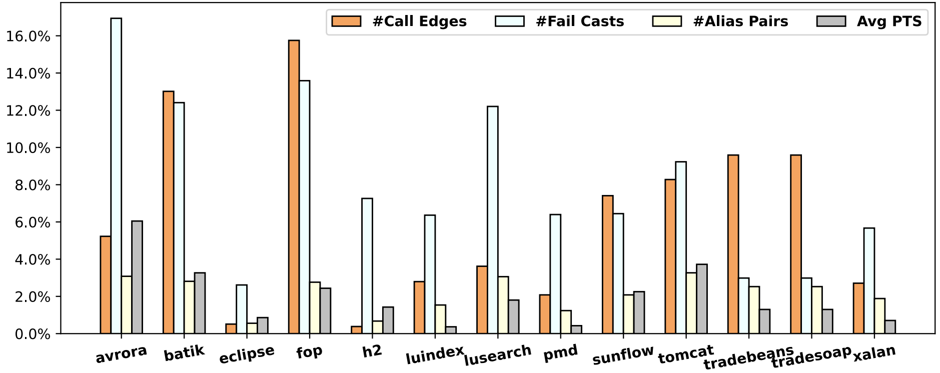
Evaluation: Settings

- Machine: Intel® Xeon® W-2245 3.90GHz, 512GB RAM
- OS: Ubuntu 20.04.3 LTS (Focal Fossa)
- **Baselines**: SelectX (SAS’21), Zipper (OOPSLA’18), kCFA
- **Benchmarks**: 13 benchmarks from the latest **DaCapo benchmark suite**
- Java library: JRE1.8.0_31

Evaluation: Precision



Precision loss of Selectx-guided 2CFA



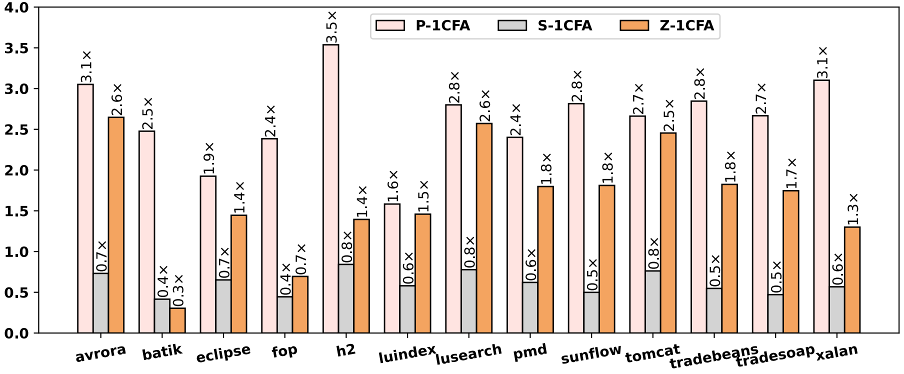
Precision loss of Zipper-guided 2CFA

P3Ctx is precision preserving.

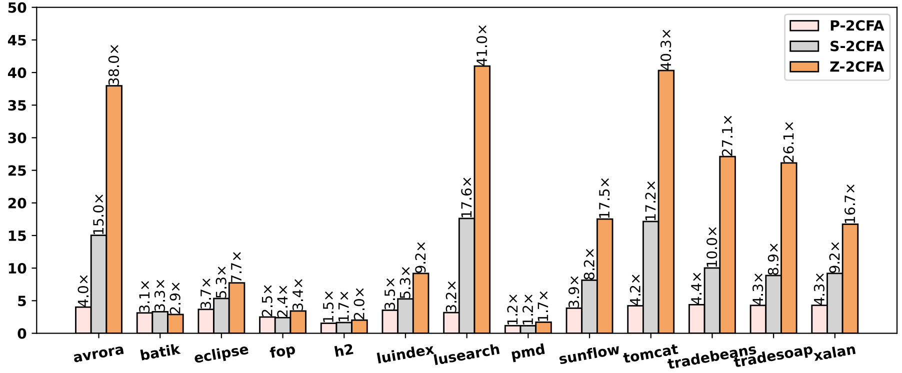
Precision Loss: P3Ctx < SelectX < Zipper

Evaluation: Speedups

We compute speedups by considering all analysis time including pre-analysis time.



Speedups over 1CFA



Speedups over 2CFA

For 1CFA (most widely used): P3Ctx > Zipper >SelectX

For 2CFA: Zipper >SelectX >P3Ctx

No one can make 3CFA scalable

Summary

Contribution 1: $L_{DCR} = L_D \cap L_c \cap L_R$

- a new CFL-reachability formulation for kCFA with built-in callgraph construction.
- show that *k*CFA is a special kind of context-sensitive language

Contribution 2: $P3Ctx$

- the first precision-preserving acceleration technique for kCFA.

Q & A

- ❑ Please refer to our paper for more technical details!

- ❑ Contact:

hedongjie15@gmail.com