

Demand-Driven Context-Sensitive Alias Analysis for Java

Dacong (Tony) Yan

Guoqing (Harry) Xu

Atanas Rountev

Ohio State University

Alias Analysis

- Many static analysis tools need highly precise and efficiently-computed alias information
- Desirable properties
 - Demand-driven: query “could x and y alias?”
 - Client-driven: client-defined time budget for a query
- Typical solution: use points-to analysis
 - Compute the objects that x may point to; same for y ;
are there common objects?
- Goal: answer alias queries precisely and efficiently, without computing the complete points-to sets

Redundancy in Points-to Analysis

Redundancy in Points-to Analysis

```
m(a) {  
    b = a;  
    x = a.f;  
    y = b.f;  
}
```

Redundancy in Points-to Analysis

```
m(a) {
```

```
  b = a;
```

```
  x = a.f;
```

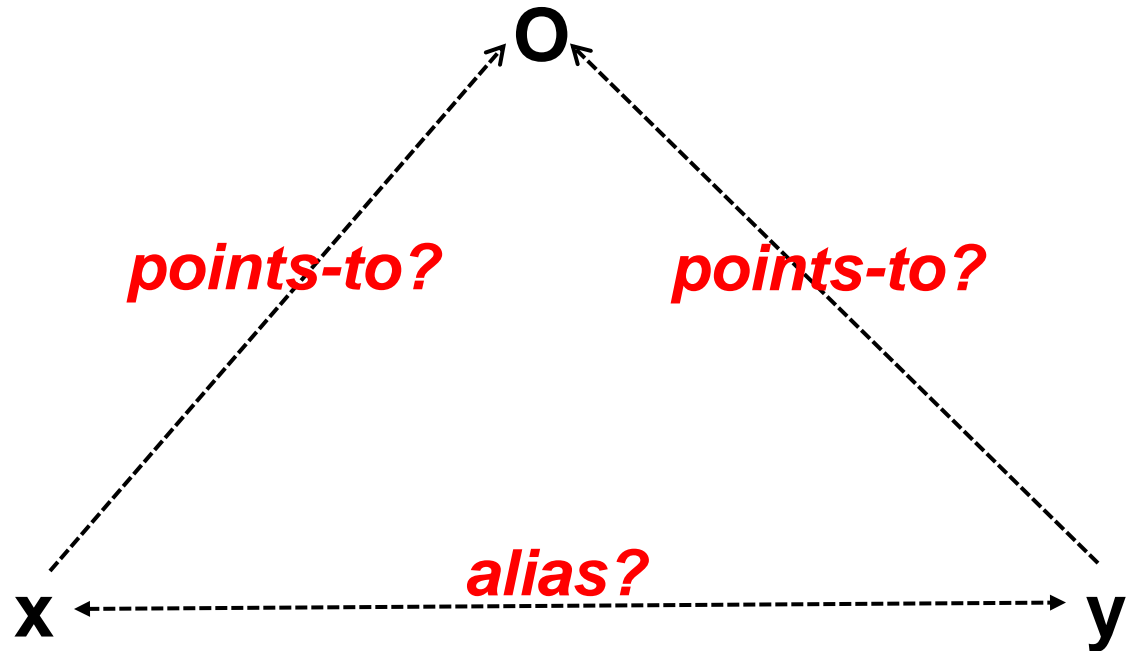
```
  y = b.f;
```

```
}
```



Redundancy in Points-to Analysis

```
m(a) {  
    b = a;  
    x = a.f;  
    y = b.f;  
}
```



Redundancy in Points-to Analysis

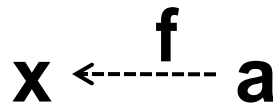
m(a) {

b = a;

x = a.f;

y = b.f;

}



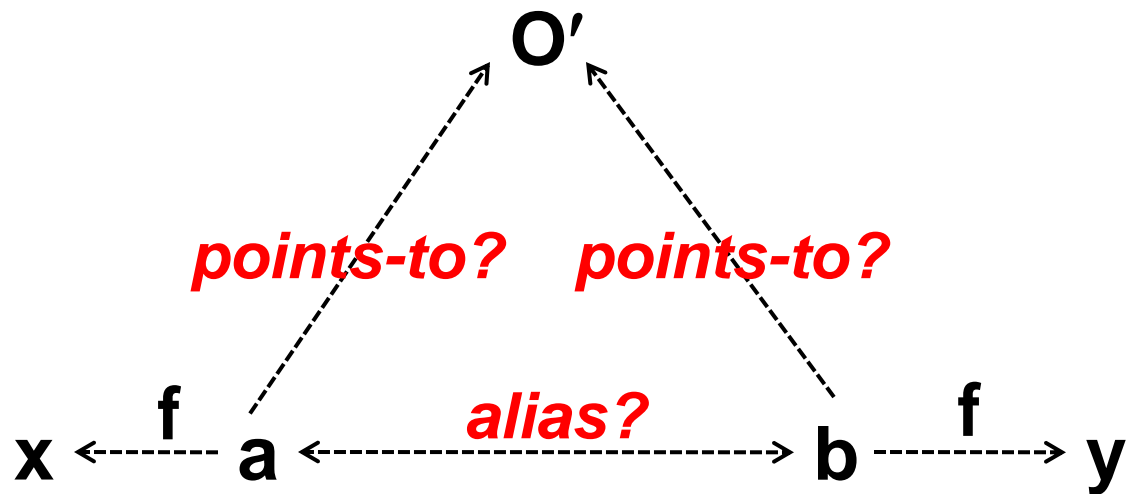
Redundancy in Points-to Analysis

```
m(a) {  
    b = a;  
    x = a.f;  
    y = b.f;  
}
```



Redundancy in Points-to Analysis

```
m(a) {  
    b = a;  
    x = a.f;  
    y = b.f;  
}
```



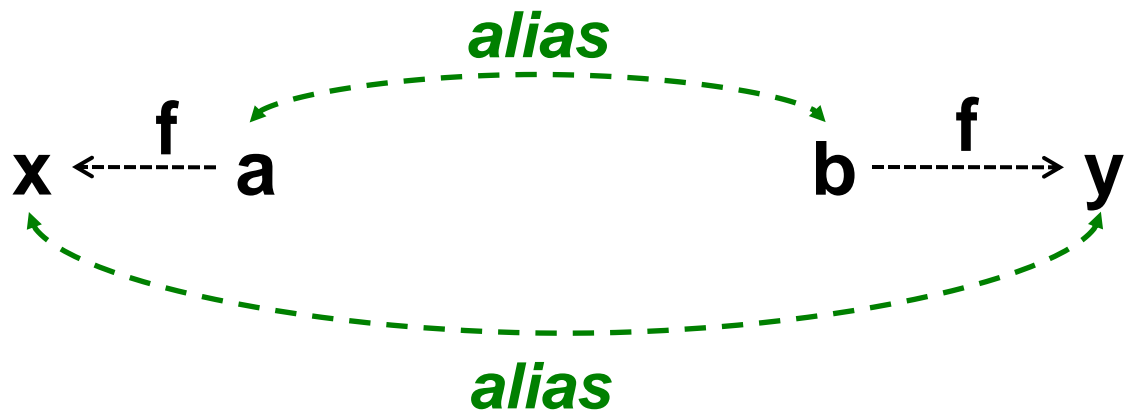
Redundancy in Points-to Analysis

```
m(a) {  
    b = a;  
    x = a.f;  
    y = b.f;  
}
```



Redundancy in Points-to Analysis

```
m(a) {  
    b = a;  
    x = a.f;  
    y = b.f;  
}
```



Our Approach

- Alias analysis
 - Demand-driven and client-driven
 - Field-sensitive and calling-context-sensitive
 - Does not require complete points-to set computation
 - Better performance through [method summaries](#)
- Symbolic Points-to Graph
 - A specialized program representation that enables the demand-driven alias analysis
 - Facilitates computation and use of method summaries

Program Representation

- Intraprocedural Symbolic Points-To Graph
 - Introduce a *symbolic object node* s for each
 - formal parameter
 - field read $a.fld$
 - a call site that returns a reference-typed value
 - Compute intraprocedural points-to relationships

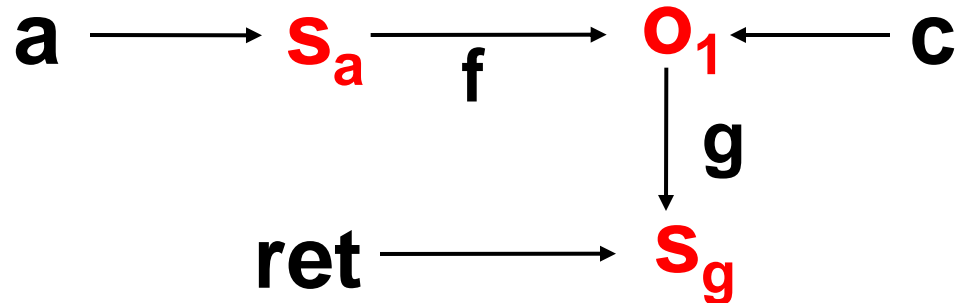
m(a) {

c = new ...; // o₁

a.f = c;

return c.g;

}



Interprocedural Symbolic Points-To Graph

- Connect the intraprocedural graphs using **entry** and **exit** edges

Interprocedural Symbolic Points-To Graph

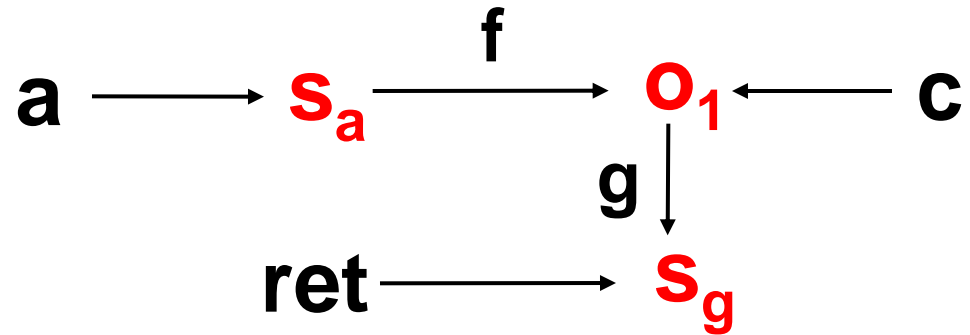
- Connect the intraprocedural graphs using **entry** and **exit** edges

```
m(a) {  
  c = new ...; // o1  
  a.f = c;  
  return c.g;  
}
```

Interprocedural Symbolic Points-To Graph

- Connect the intraprocedural graphs using **entry** and **exit** edges

```
m(a) {  
  c = new ...; // o1  
  a.f = c;  
  return c.g;  
}
```

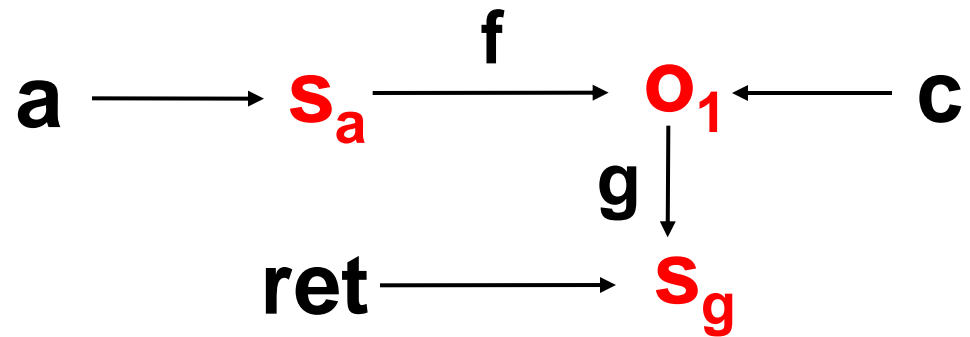


Interprocedural Symbolic Points-To Graph

- Connect the intraprocedural graphs using **entry** and **exit** edges

```
m(a) {  
  c = new ...; // o1  
  a.f = c;  
  return c.g;  
}
```

```
d = new ...; // o2  
b = m(d); // call m
```

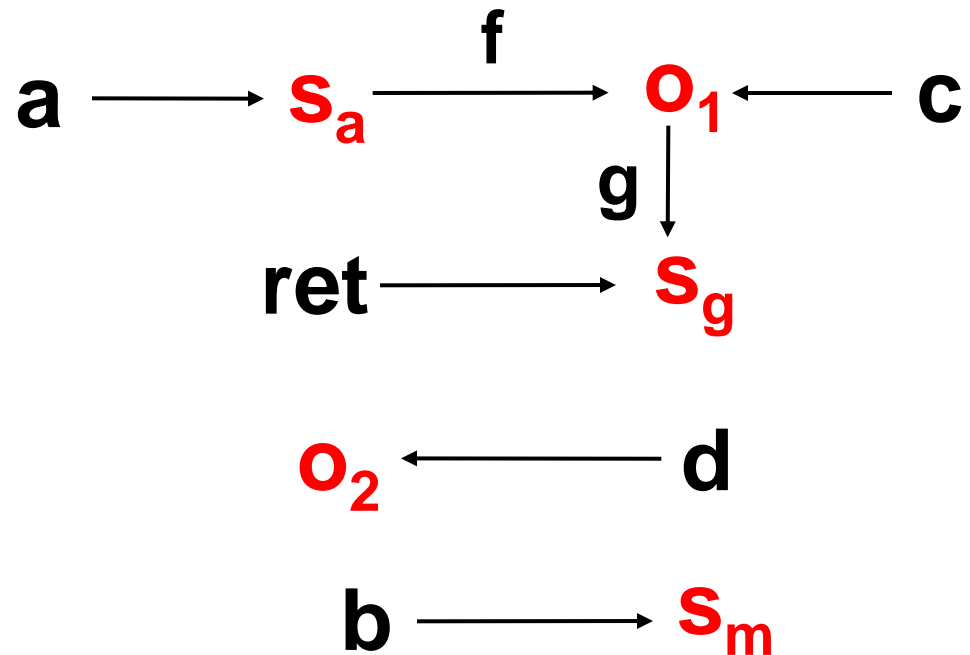


Interprocedural Symbolic Points-To Graph

- Connect the intraprocedural graphs using **entry** and **exit** edges

```
m(a) {  
  c = new ...; // o1  
  a.f = c;  
  return c.g;  
}
```

```
d = new ...; // o2  
b = m(d); // call m
```

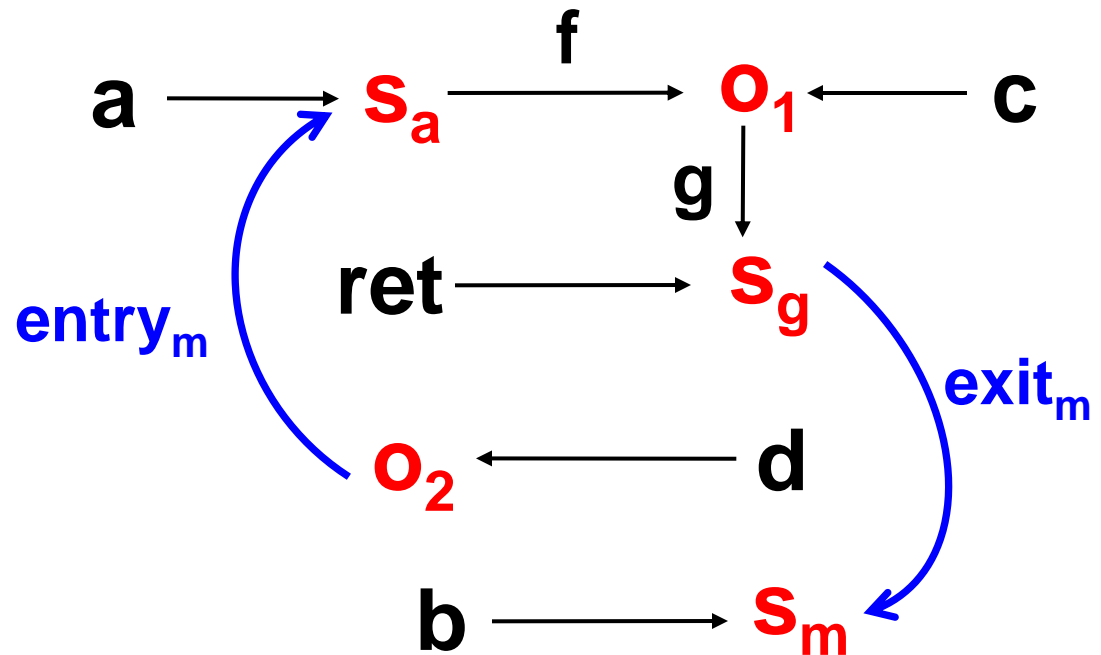


Interprocedural Symbolic Points-To Graph

- Connect the intraprocedural graphs using **entry** and **exit** edges

```
m(a) {  
  c = new ...; // o1  
  a.f = c;  
  return c.g;  
}
```

```
d = new ...; // o2  
b = m(d); // call m
```

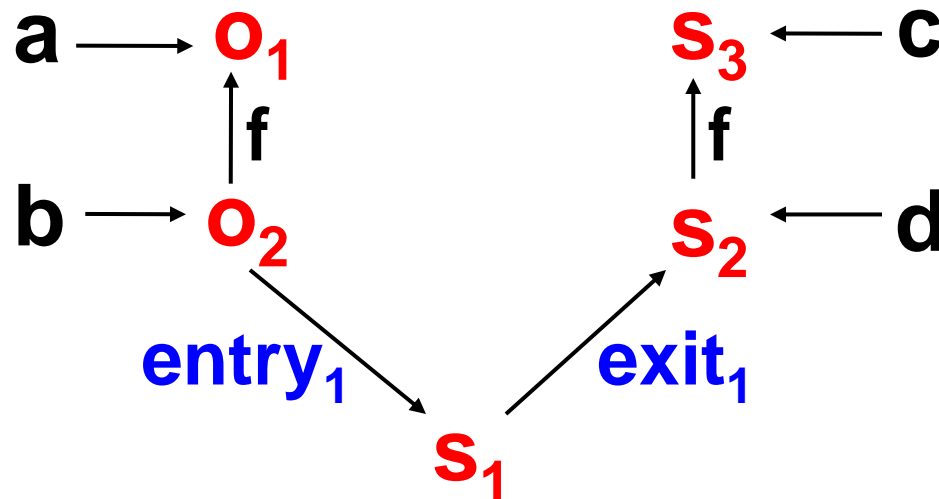


Alias Analysis Formulation

- Field-sensitivity and calling-context-sensitivity: matched **points-to edges** and **entry/exit edges**
- Traverses the object nodes (including symbolic ones) and the edges between them

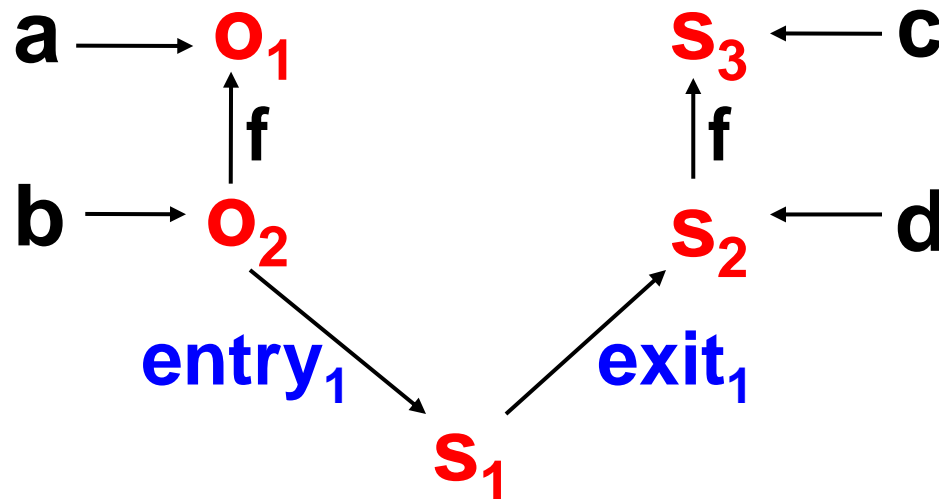
Alias Analysis Formulation

- Field-sensitivity and calling-context-sensitivity: matched **points-to edges** and **entry/exit edges**
- Traverses the object nodes (including symbolic ones) and the edges between them



Alias Analysis Formulation

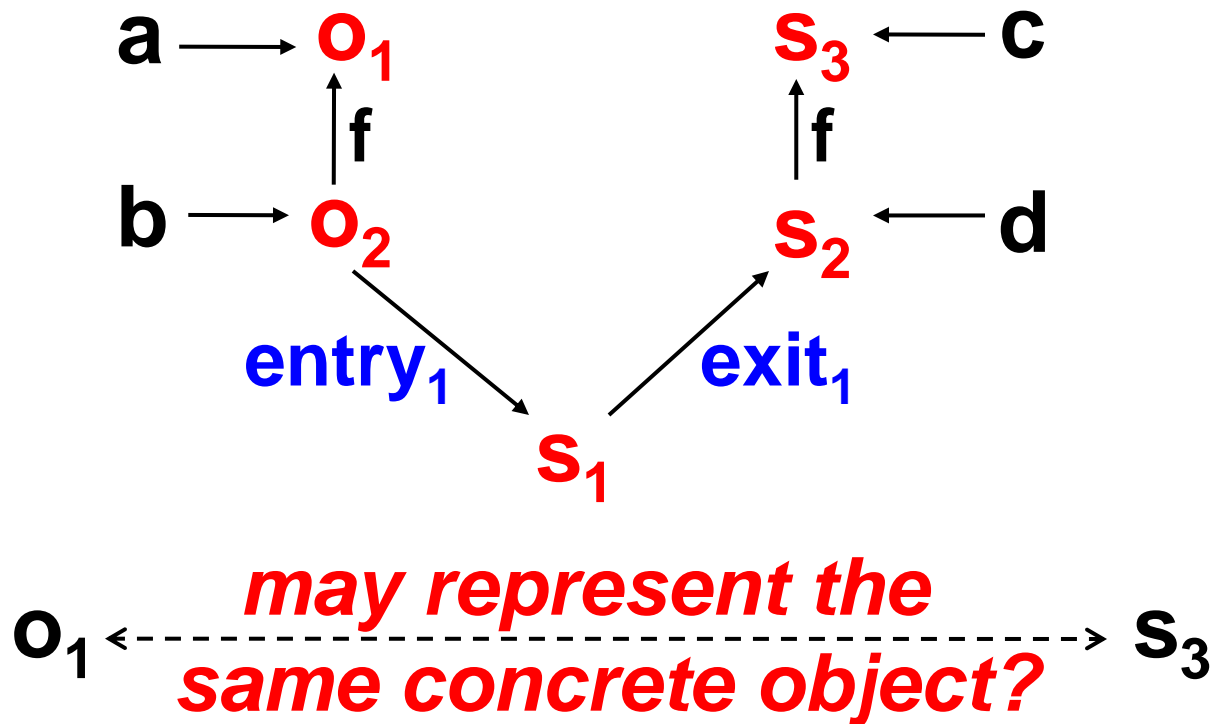
- Field-sensitivity and calling-context-sensitivity: matched **points-to edges** and **entry/exit edges**
- Traverses the object nodes (including symbolic ones) and the edges between them



a *alias?* **c**

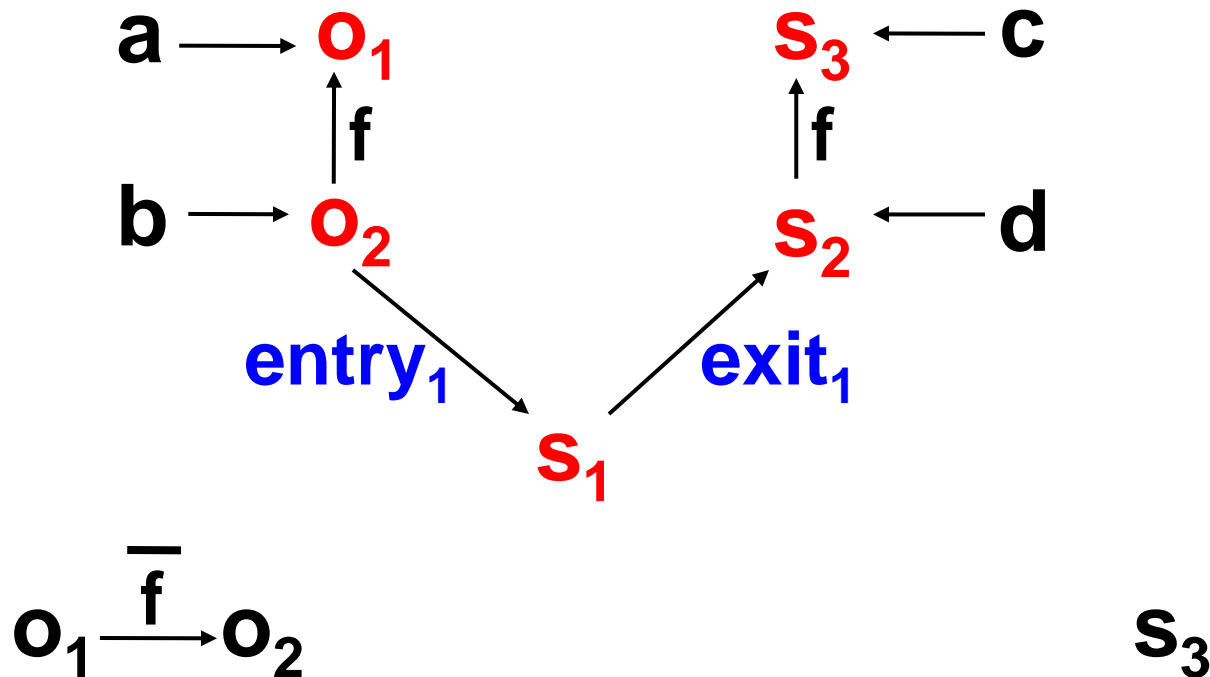
Alias Analysis Formulation

- Field-sensitivity and calling-context-sensitivity: matched **points-to edges** and **entry/exit edges**
- Traverses the object nodes (including symbolic ones) and the edges between them



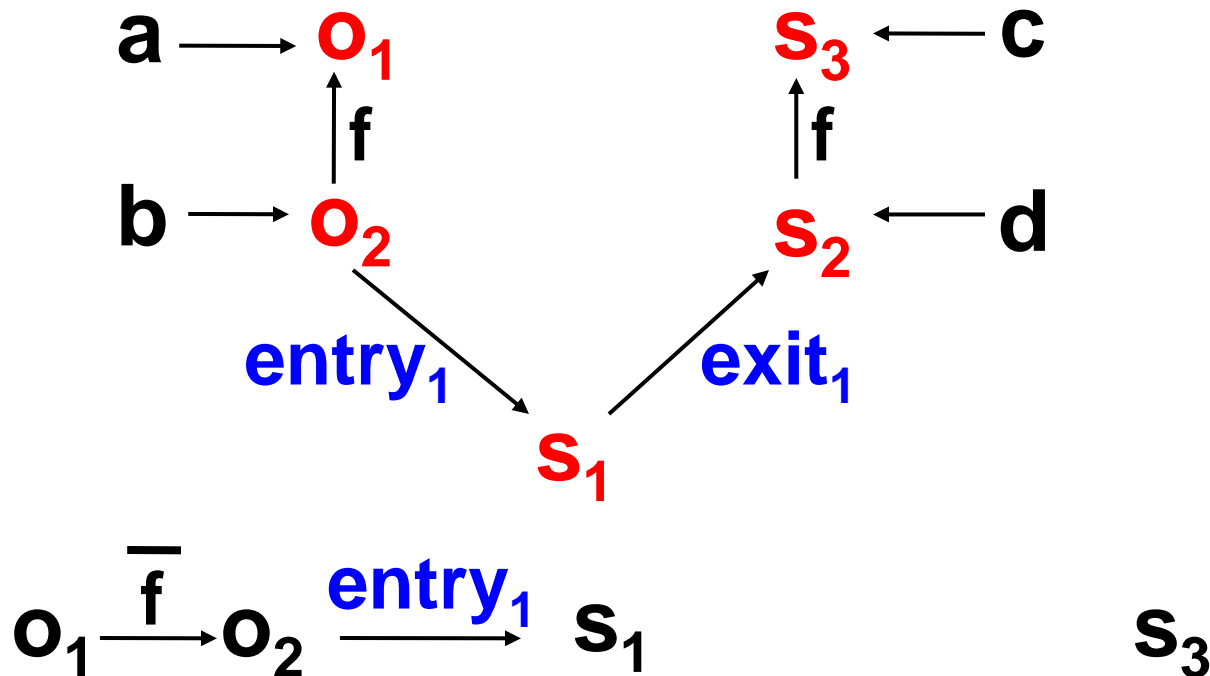
Alias Analysis Formulation

- Field-sensitivity and calling-context-sensitivity: matched **points-to edges** and **entry/exit edges**
- Traverses the object nodes (including symbolic ones) and the edges between them



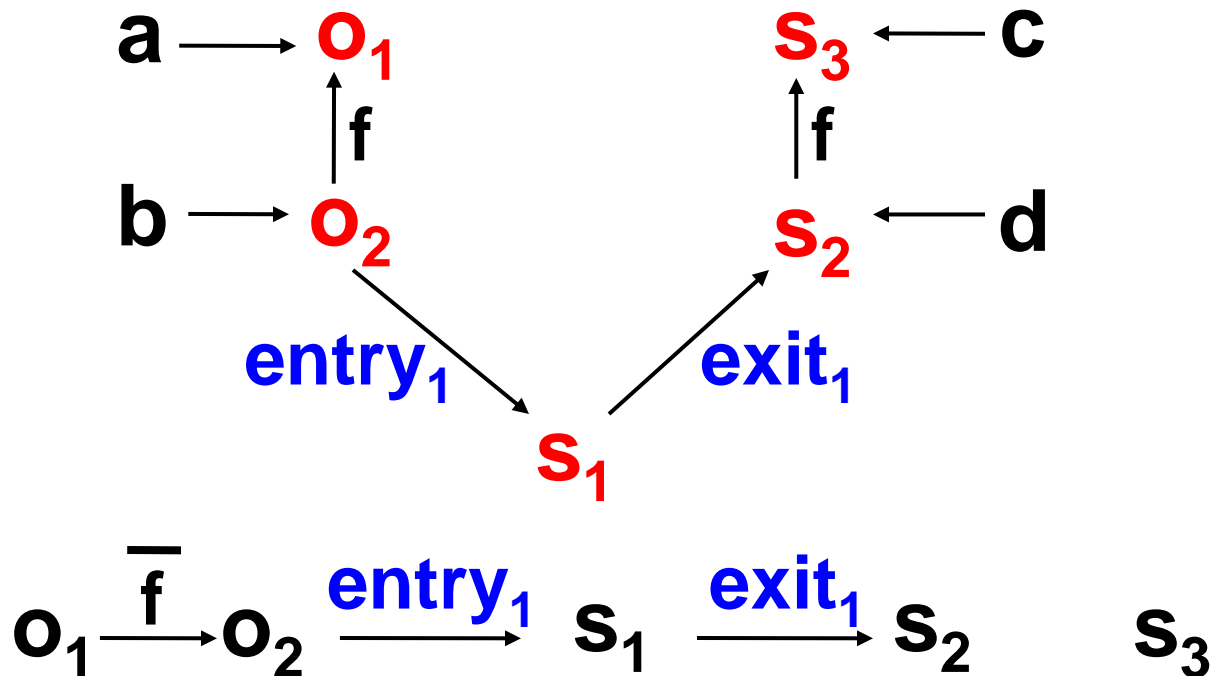
Alias Analysis Formulation

- Field-sensitivity and calling-context-sensitivity: matched **points-to edges** and **entry/exit edges**
- Traverses the object nodes (including symbolic ones) and the edges between them



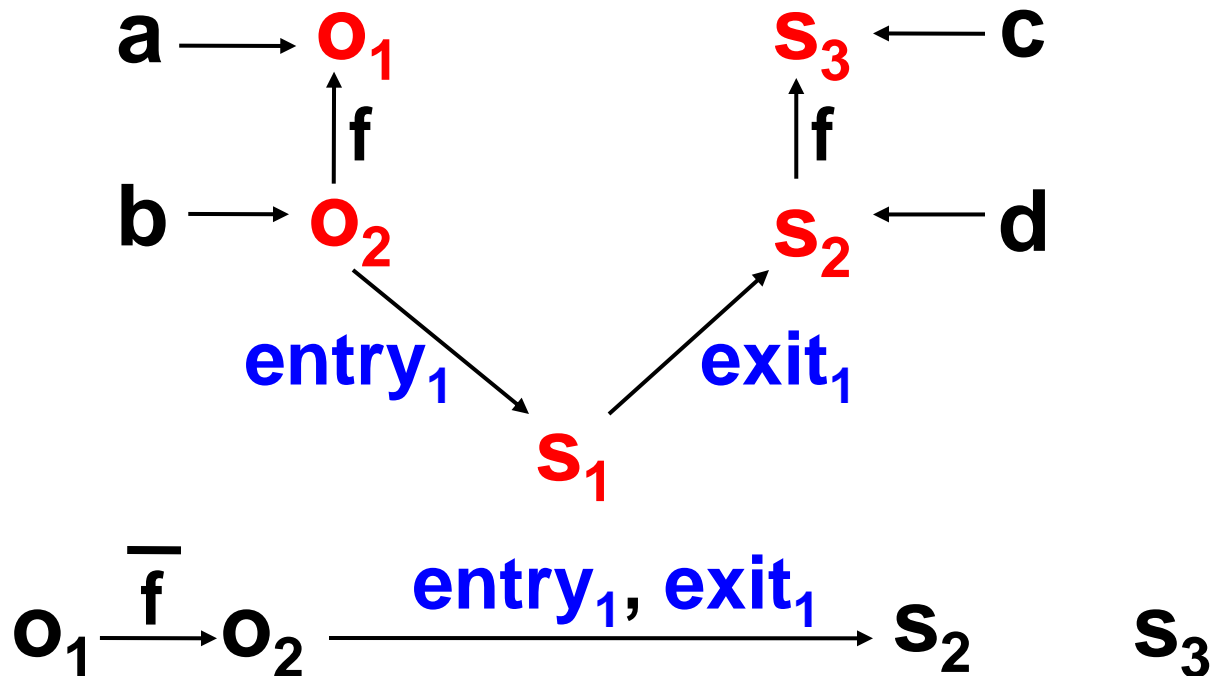
Alias Analysis Formulation

- Field-sensitivity and calling-context-sensitivity: matched **points-to edges** and **entry/exit edges**
- Traverses the object nodes (including symbolic ones) and the edges between them



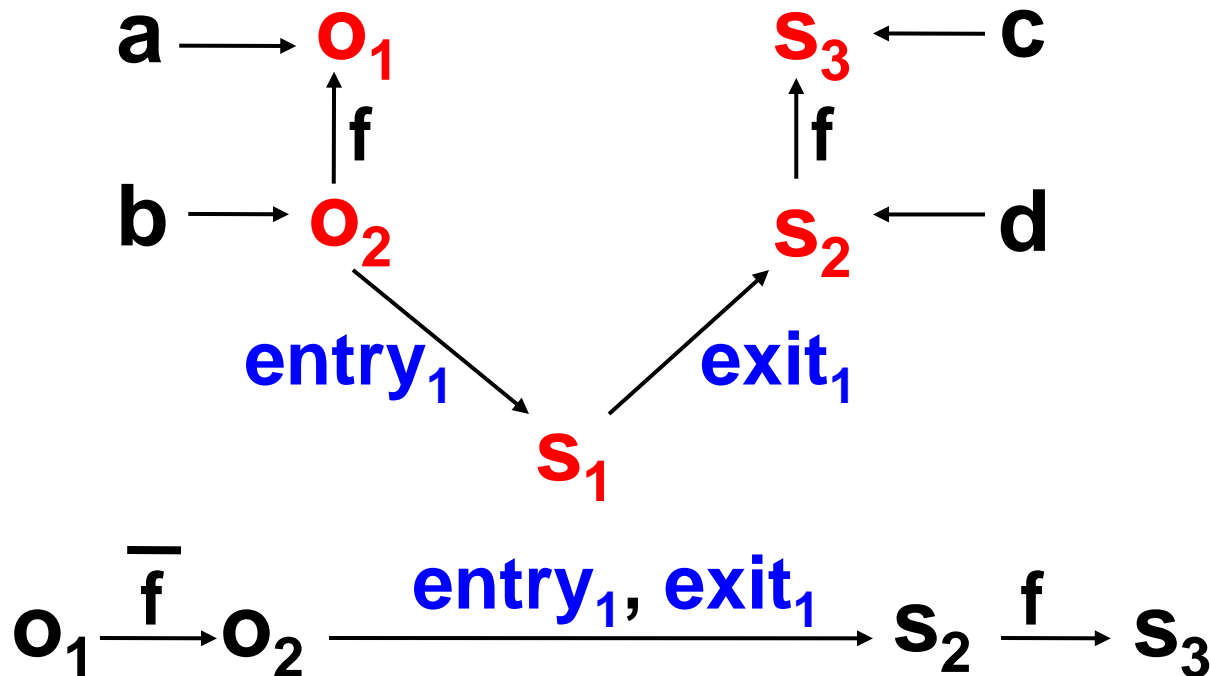
Alias Analysis Formulation

- Field-sensitivity and calling-context-sensitivity: matched **points-to edges** and **entry/exit edges**
- Traverses the object nodes (including symbolic ones) and the edges between them



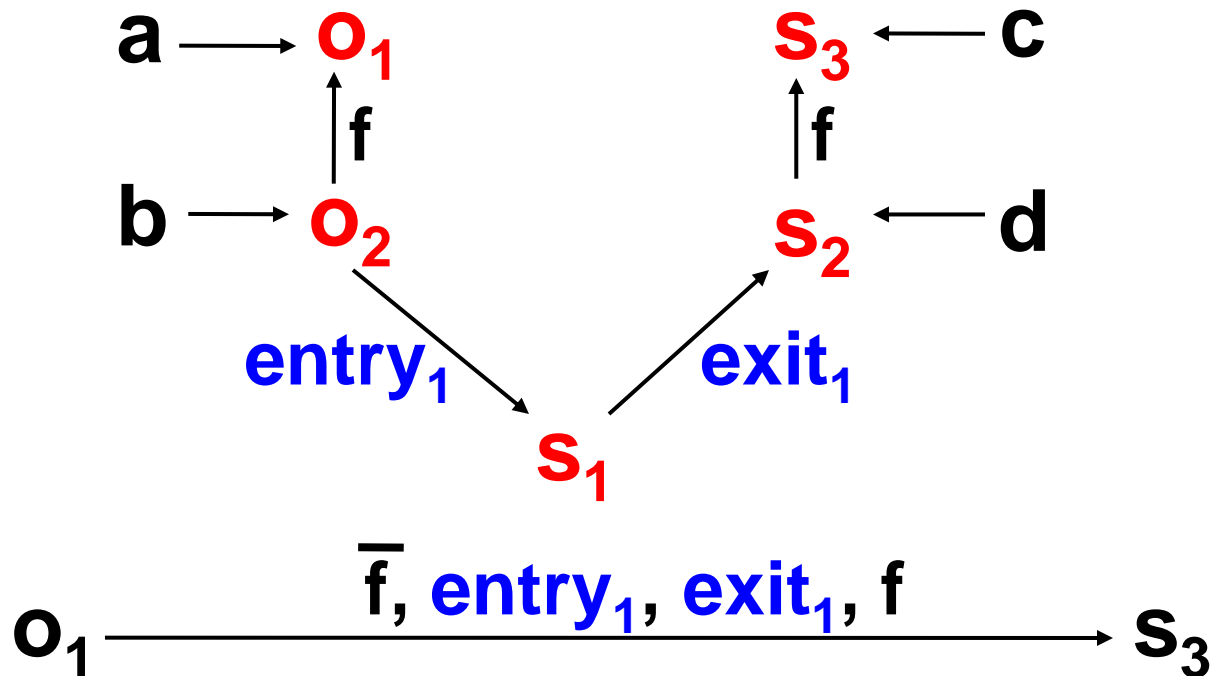
Alias Analysis Formulation

- Field-sensitivity and calling-context-sensitivity: matched **points-to edges** and **entry/exit edges**
- Traverses the object nodes (including symbolic ones) and the edges between them



Alias Analysis Formulation

- Field-sensitivity and calling-context-sensitivity: matched **points-to edges** and **entry/exit edges**
- Traverses the object nodes (including symbolic ones) and the edges between them



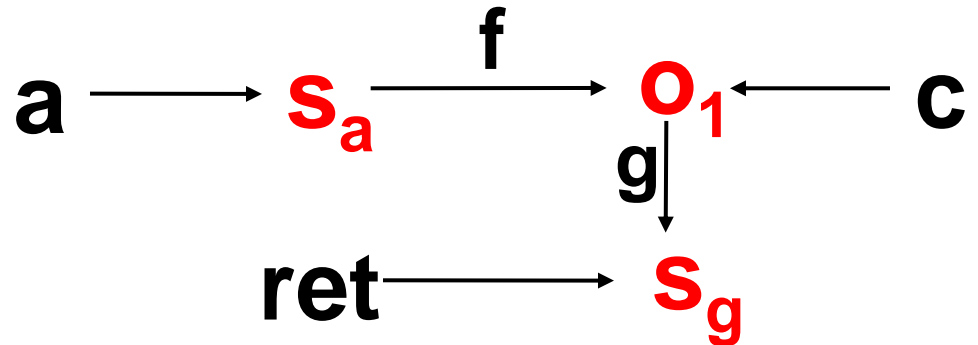
Using Method Summaries

- Several reachability strings for a selected method
 - Each string is a sequence of field points-to edges between boundary objects of the method

Using Method Summaries

- Several reachability strings for a selected method
 - Each string is a sequence of field points-to edges between boundary objects of the method

```
m(a) {  
  c = new ...; // o1  
  a.f = c;  
  return c.g;  
}
```



Using Method Summaries

- Several reachability strings for a selected method
 - Each string is a sequence of field points-to edges between boundary objects of the method

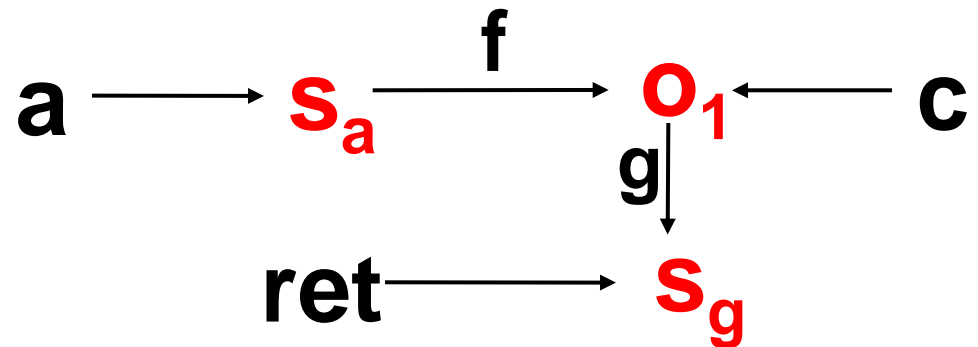
```
m(a) {
```

```
c = new ...; // o1
```

```
a.f = c;
```

```
return c.g;
```

```
}
```



```
summary(m): Sa  $\xrightarrow{f, g}$  Sg
```

Using Method Summaries

- Several reachability strings for a selected method
 - Each string is a sequence of field points-to edges between boundary objects of the method
- Selecting methods for summarization
 - Compute a summary for a method only if it is invoked from many different call sites

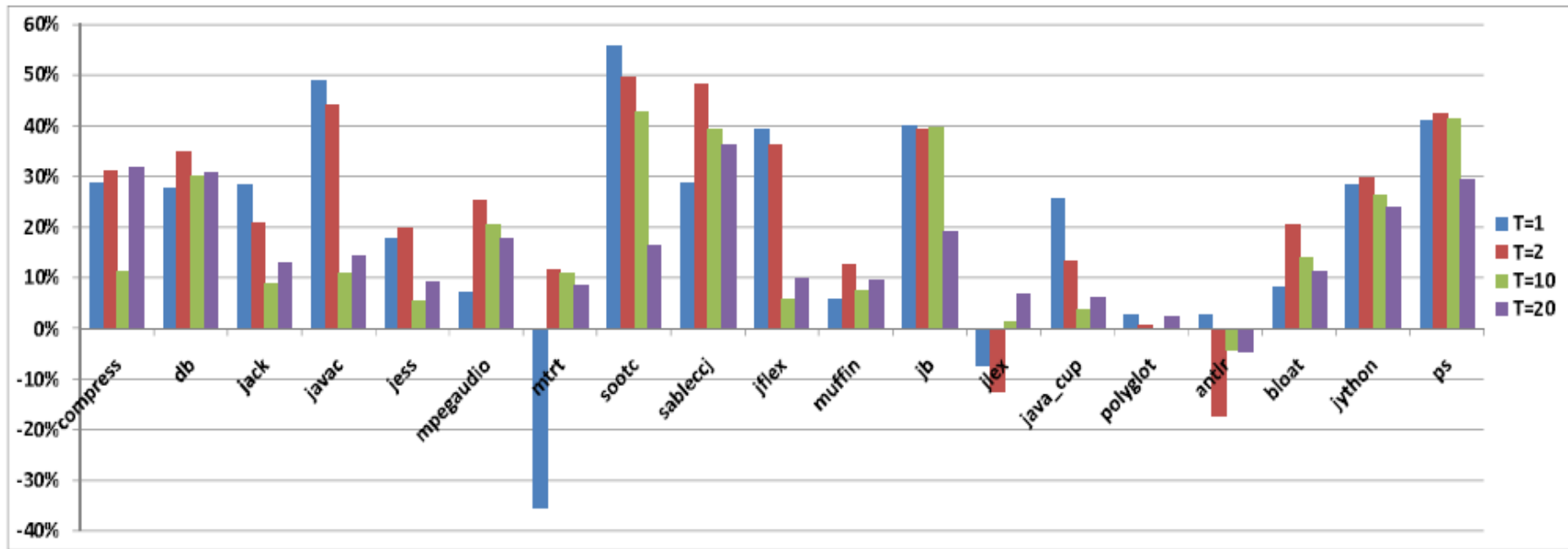
Using Method Summaries

- Several reachability strings for a selected method
 - Each string is a sequence of field points-to edges between boundary objects of the method
- Selecting methods for summarization
 - Compute a summary for a method only if it is invoked from many different call sites
 - Summarization ratio for method **m**
 - the number of incoming call graph edges of **m**, divided by the average number of incoming call graph edges for all methods

Experimental Evaluation

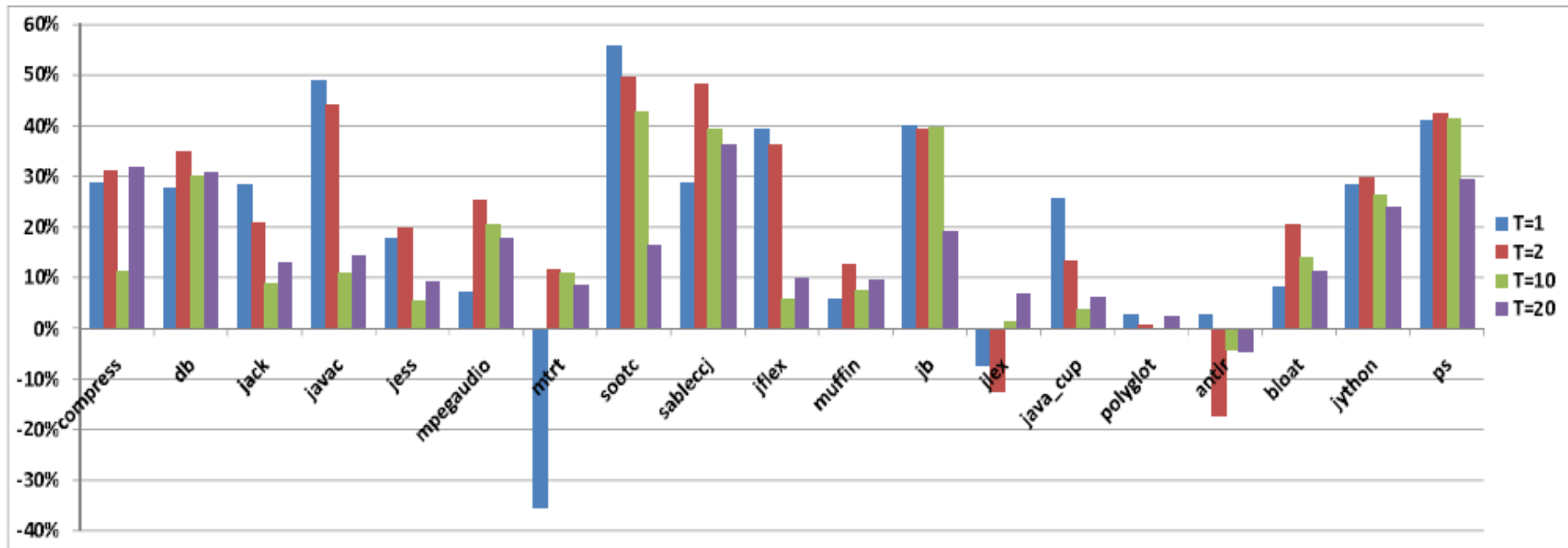
- 19 Java programs
 - Number of methods in the whole-program call graph ranging from 2344 to 8789
- **Experiment 1**: compare the precision with a field and context-sensitive, demand-driven, client-driven points-to analysis [PLDI'06]
 - Under the same time budget per alias query
 - Result 1: for 14 programs, the number of alias pairs is lower when using our analysis
 - Result 2: summarization leads to better precision
- **Experiment 2**: compare running times with and without method summaries

Running Time Reduction When Using Method Summaries



$$\text{Running Time Reduction} = (\text{RT}_{no-summ} - \text{RT}_{summ}) / \text{RT}_{no-summ}$$

Running Time Reduction When Using Method Summaries



$$\text{Running Time Reduction} = (\text{RT}_{no-summ} - \text{RT}_{summ}) / \text{RT}_{no-summ}$$

24% average reduction with threshold T=2

Conclusions

- A demand-driven alias analysis
 - Answers alias queries directly, without computing the complete points-to sets
 - Selects methods for online summarization to reduce analysis running time
 - Outperforms a highly precise state-of-the-art points-to analysis

Thank you