

Compositional Pointer and Escape Analysis for Java Programs

John Whaley and Martin Rinard

Presented by Chen Fu
Rutgers

Points-to Analysis

- Flow Sensitive
 - Intraprocedural
 - Do “kill”
 - “Summary” points-to graph for each method.
 - “Outside” nodes/edges to represent interprocedural/unknown points-to relation.
 - Interprocedural
 - Merge the caller/callee points-to graph at call sites.
 - How to get initial call graph?
- Context Sensitive?
 - Maybe
- Emphasis on escape analysis.

Inside Nodes/Edges

- Inside nodes/edges
 - Just like those in Anderson's points-to

Outside Nodes/Edges

- Outside Nodes:
 - One of more (static) objects
 - Created outside the current scope.
 - Referenced in the current scope
 - Parameter
 - Global
 - Field Load
 - Return ??
- Outside edges
 - Edges established outside the current scope.
 - Discovered in the current scope.

Algorithm

- Introduce each outside node for
 - Each reference parameter (inside edge),
 - Static field accessed (outside edge).
- Run analysis similar in Anderson's points-to except:
 - Do kill, need iterate,
 - Difference in field load.
- At call sites
 - Merge callee's summary with current points-to graph.

Escape

- An object escape a method if reachable from:
 - A parameter
 - A static field
 - A thread object
 - An actual parameter of an not-yet-known method.

Figure 1

```
class complex{
    double x,y;
    complex(double a, double b) { x = a; y = b; }

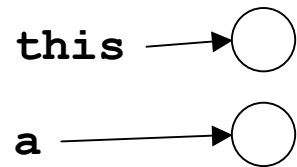
    complex multiply(complex a) {
        complex product = new complex(x*a.x - y*a.y, x*a.y + y*a.x);
        return product;
    }

    complex multiply(complex a){
        complex sum = new complex(x+a.x, y+a.y);
        return sum;
    }

    complex multiplyAdd(complex a, complex b) {
        complex product = a.multiply(b);
        complex sum = this.add(product);
        return sum;
    }
}
```

Figure 1 – Cont.

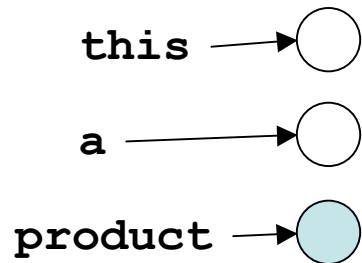
```
complex multiply(complex a) {  
    complex product = new();  
    return product;  
}
```



Graph Initialization

Figure 1 – Cont.

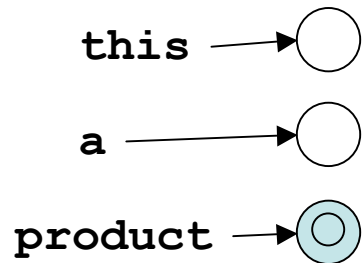
```
complex multiply(complex a) {  
    complex product = new();  
    return product;  
}
```



Graph Initialization
New Object

Figure 1 – Cont.

```
complex multiply(complex a) {  
    complex product = new();  
    return product;  
}
```



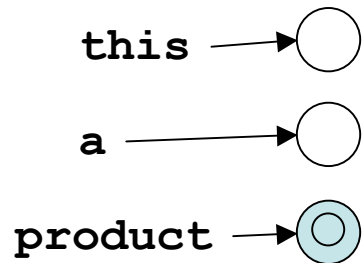
Graph Initialization

New Object

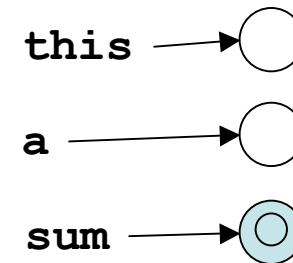
Return

Figure 1 – Cont.

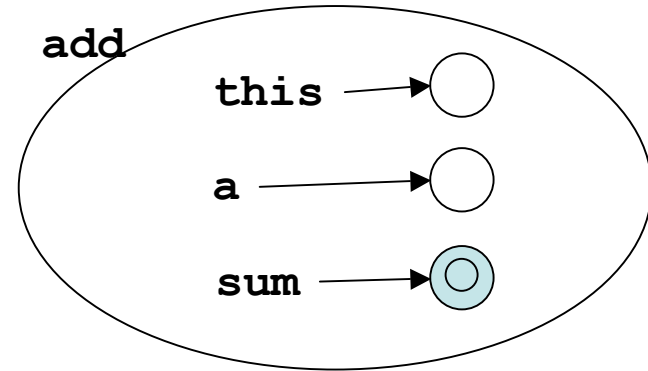
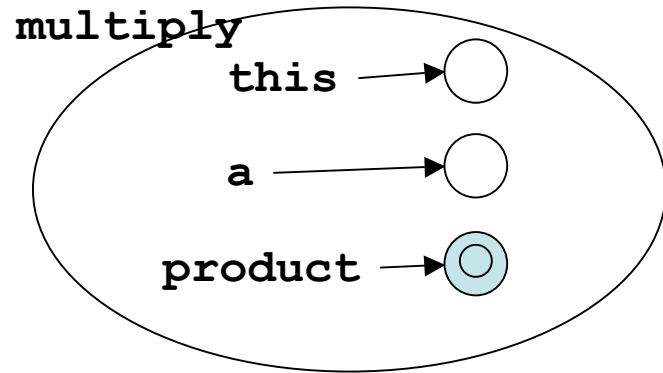
```
complex multiply(complex a) {  
    complex product = new();  
    return product;  
}
```



```
complex add(complex a) {  
    complex sum = new complex();  
    return sum;  
}
```

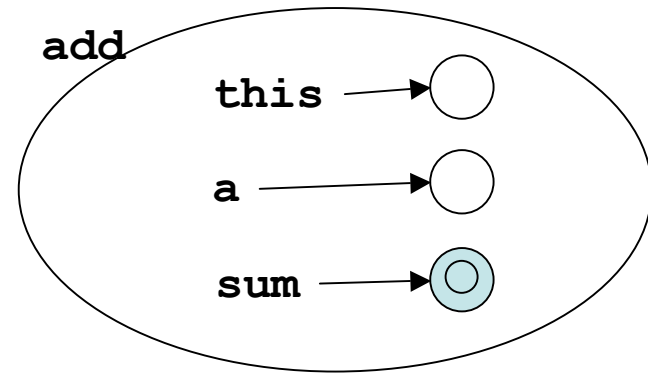
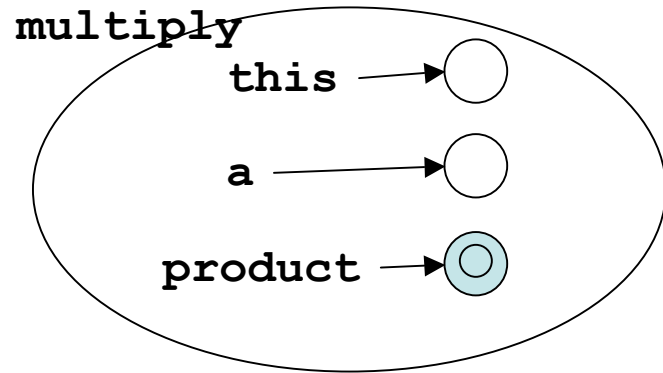


Graph Initialization
New Object
Return



```
complex multiplyAdd(complex a, complex b) {  
    complex product = a.multiply(b);  
    complex sum = this.add(product);  
    return sum;  
}
```

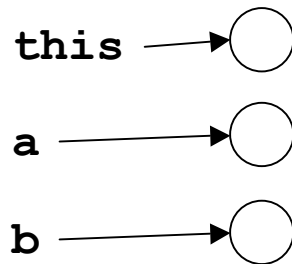
Graph merging at call site



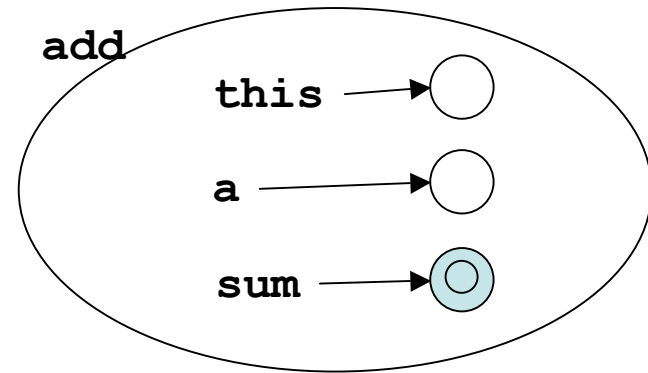
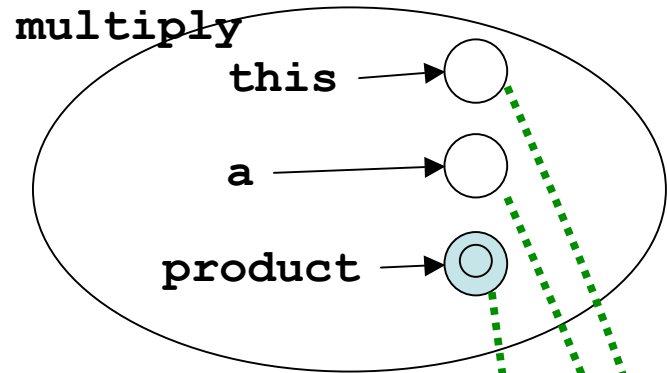
```

complex multiplyAdd(complex a, complex b) {
  complex product = a.multiply(b);
  complex sum = this.add(product);
  return sum;
}

```



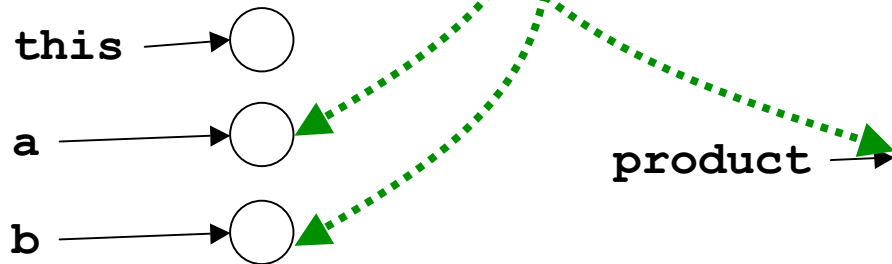
Graph merging at call site



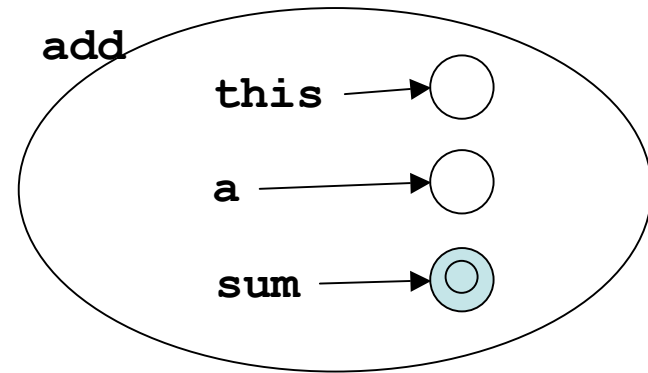
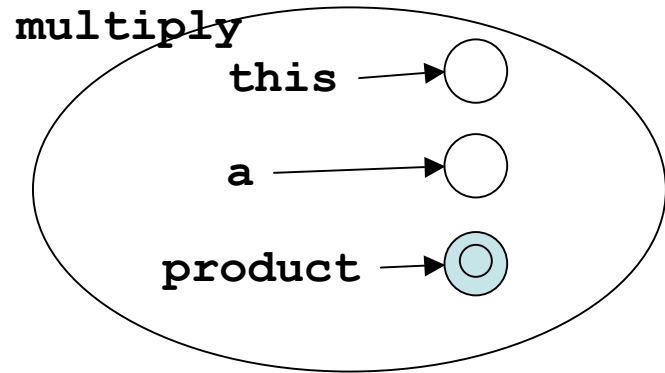
```

complex multiplyAdd(complex a, complex b) {
  complex product = a.multiply(b);
  complex sum = this.add(product);
  return sum;
}

```



Graph merging at call site



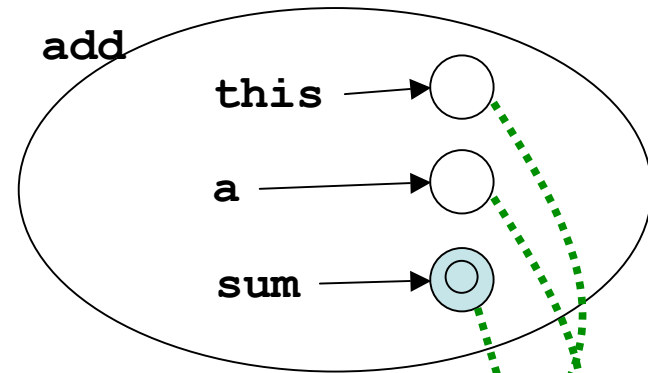
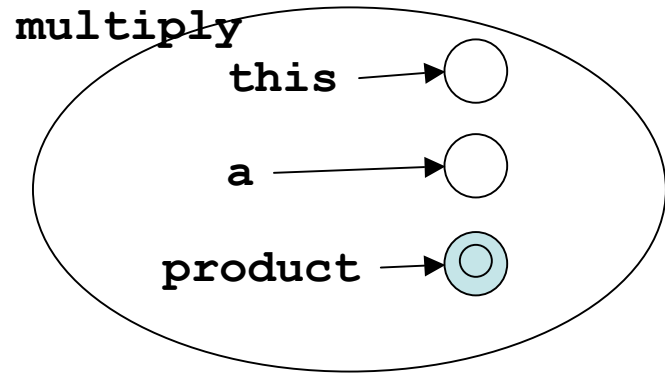
```

complex multiplyAdd(complex a, complex b) {
  complex product = a.multiply(b);
  complex sum = this.add(product);
  return sum;
}

```



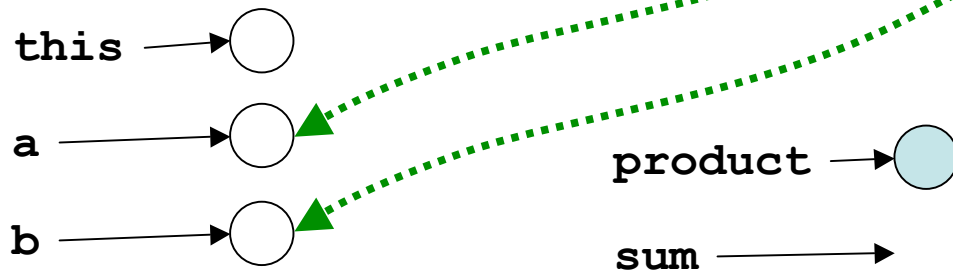
Graph merging at call site



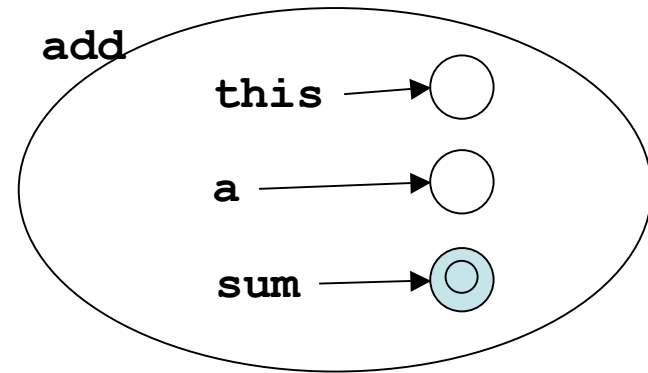
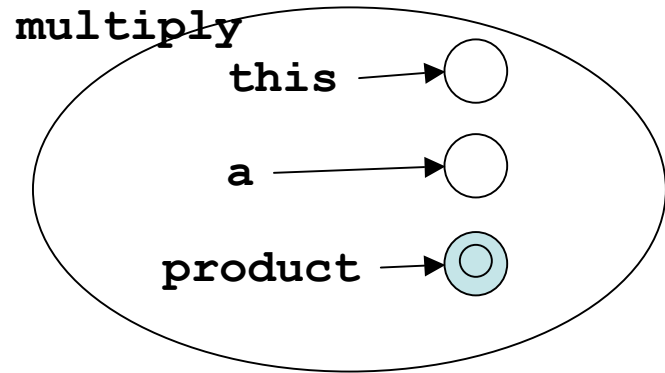
```

complex multiplyAdd(complex a, complex b) {
  complex product = a.multiply(b);
  complex sum = this.add(product);
  return sum;
}

```



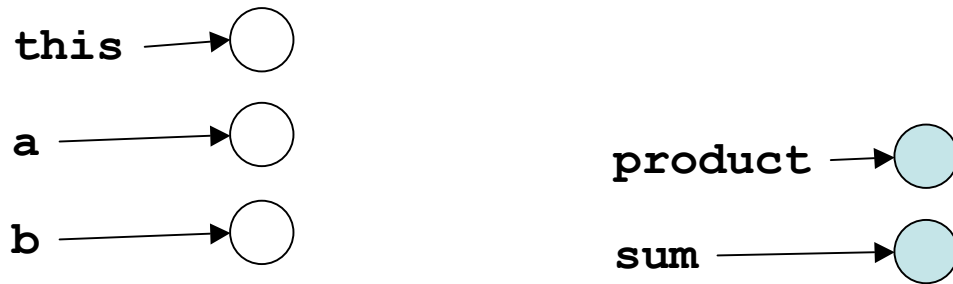
Graph merging at call site



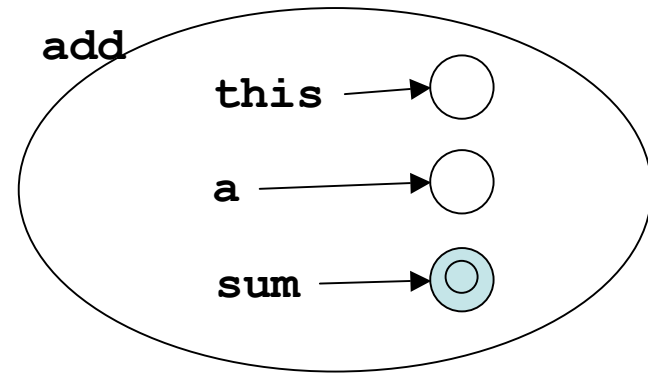
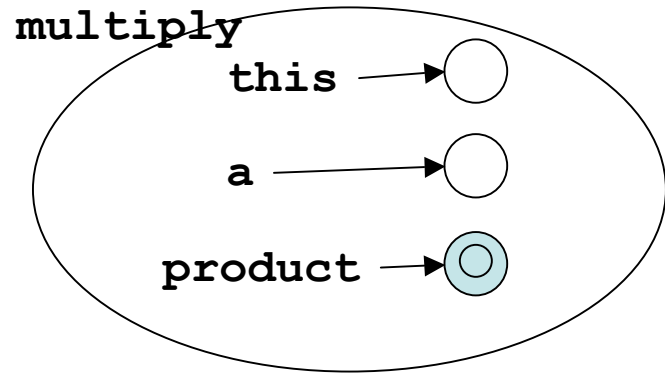
```

complex multiplyAdd(complex a, complex b) {
  complex product = a.multiply(b);
  complex sum = this.add(product);
  return sum;
}

```



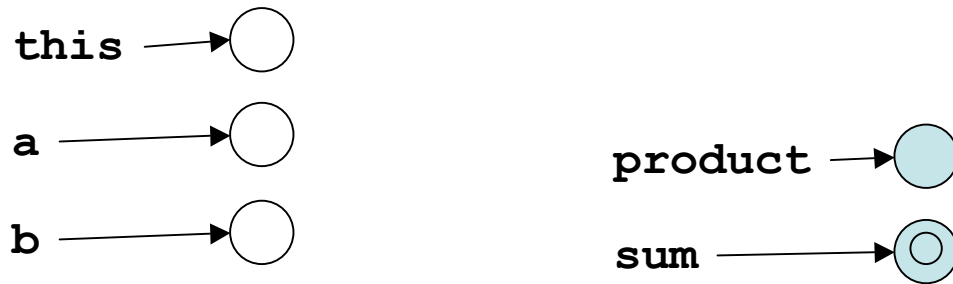
Graph merging at call site



```

complex multiplyAdd(complex a, complex b) {
  complex product = a.multiply(b);
  complex sum = this.add(product);
  return sum;
}

```



Graph merging at call site

Figure 4

```
<init>(Object e, multisetElement n) {  
    this.element = e;  
    this.next = n;  
}
```

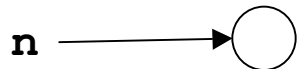
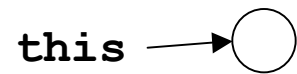


Figure 4

```
<init>(Object e, multisetElement n) {  
    this.element = e;  
    this.next = n;  
}
```

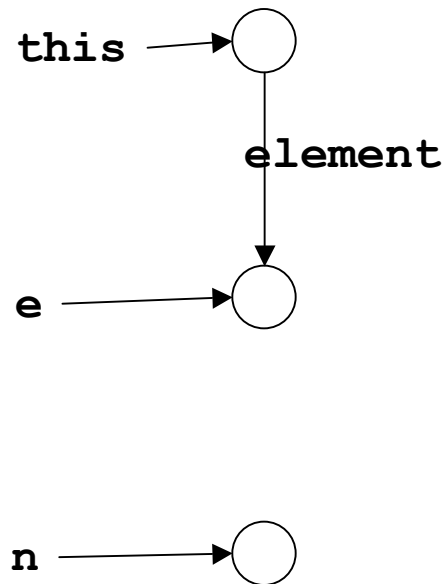


Figure 4

```
<init>(Object e, multisetElement n) {  
    this.element = e;  
    this.next = n;  
}
```

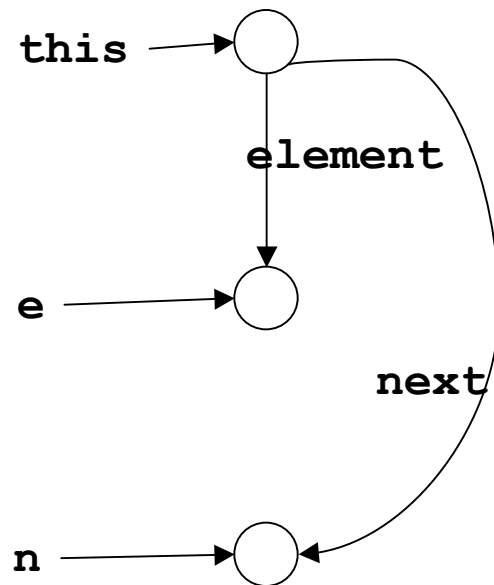
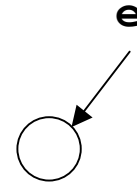
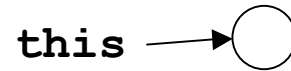
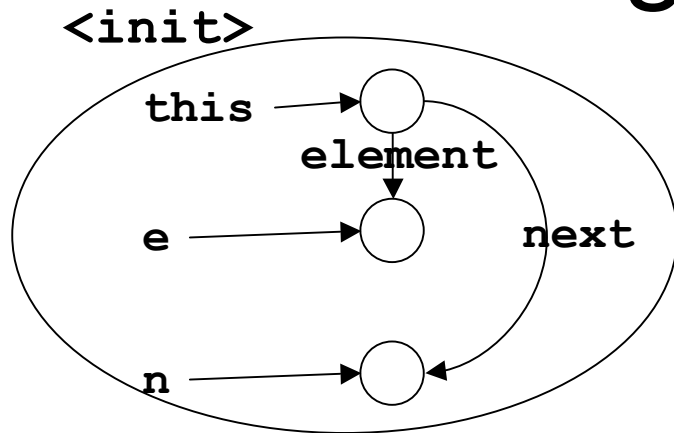
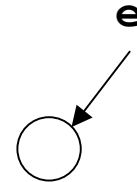
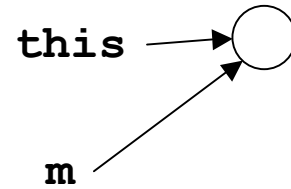
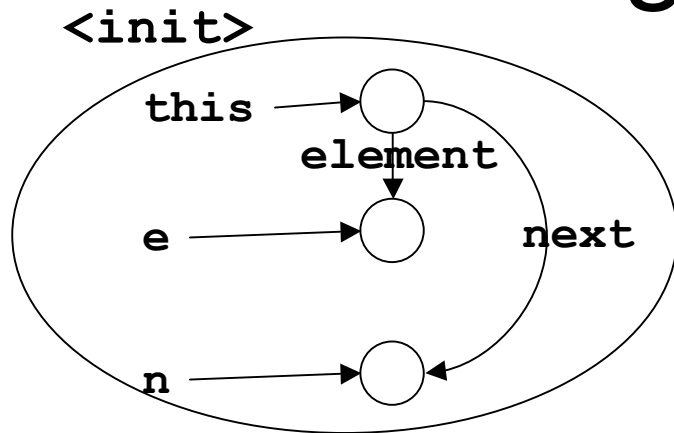


Figure 4 – cont.



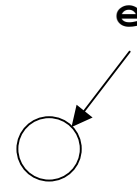
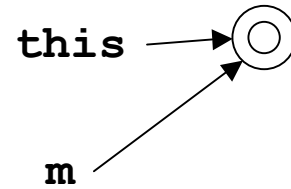
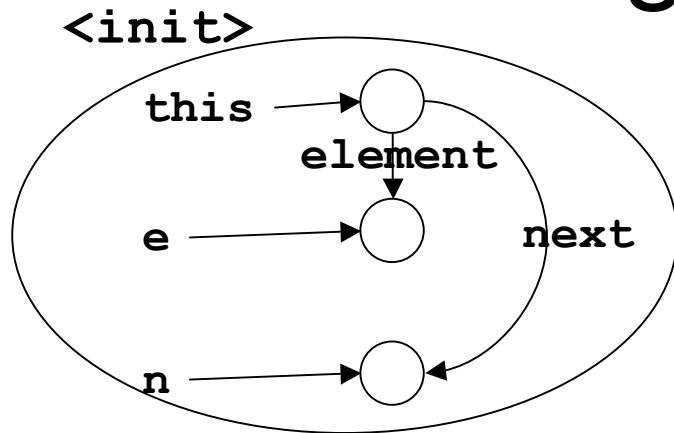
```
multisetElement insert(Object e){
  multisetElement m = this
  while (..) {
    if (..) return this;
    m = m.next;
  }
  return new multisetElement(e, this);
}
```

Figure 4 – cont.



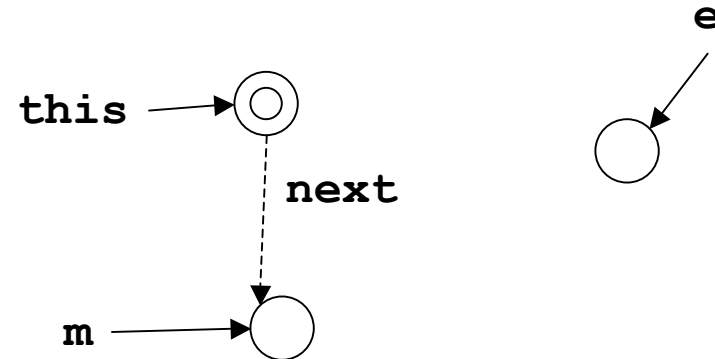
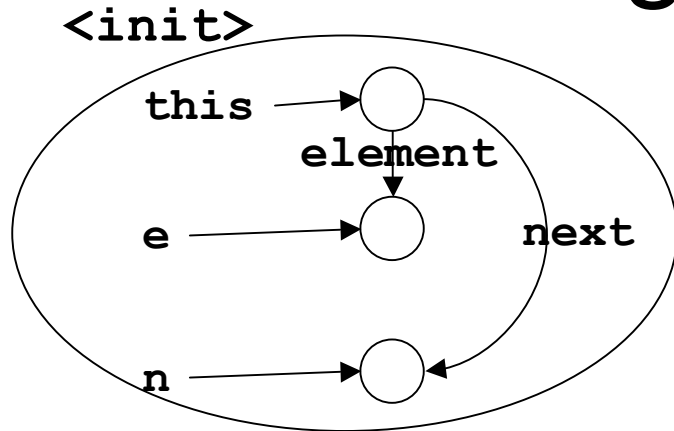
```
multisetElement insert(Object e){
  multisetElement m = this
  while (..) {
    if (..) return this;
    m = m.next;
  }
  return new multisetElement(e, this);
}
```

Figure 4 – cont.



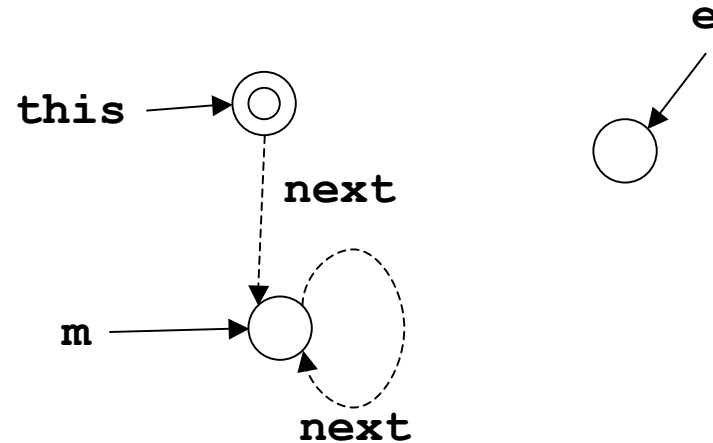
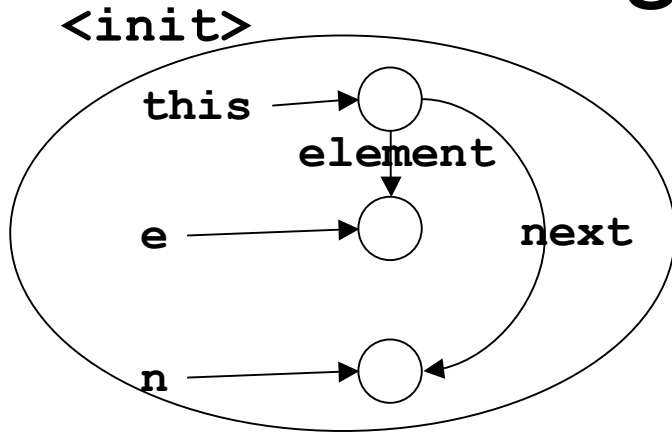
```
multisetElement insert(Object e){
  multisetElement m = this
  while (..) {
    if (..) return this;
    m = m.next;
  }
  return new multisetElement(e, this);
}
```


Figure 4 – cont.



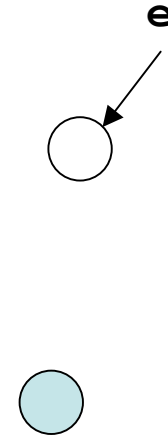
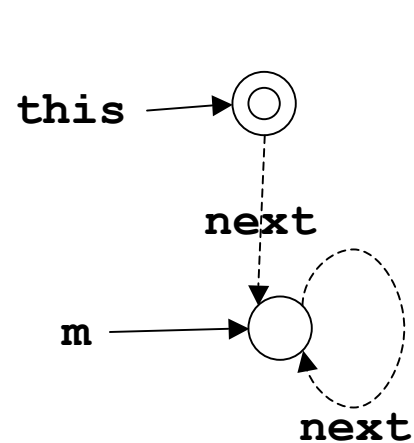
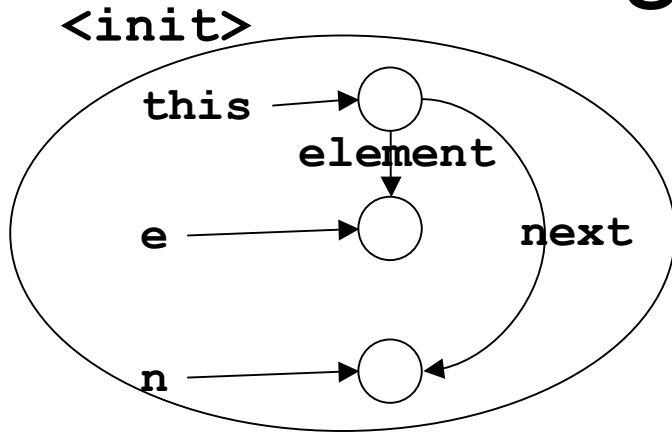
```
multisetElement insert(Object e){
  multisetElement m = this
  while (..) {
    if (..) return this;
    m = m.next;
  }
  return new multisetElement(e, this);
}
```

Figure 4 – cont.



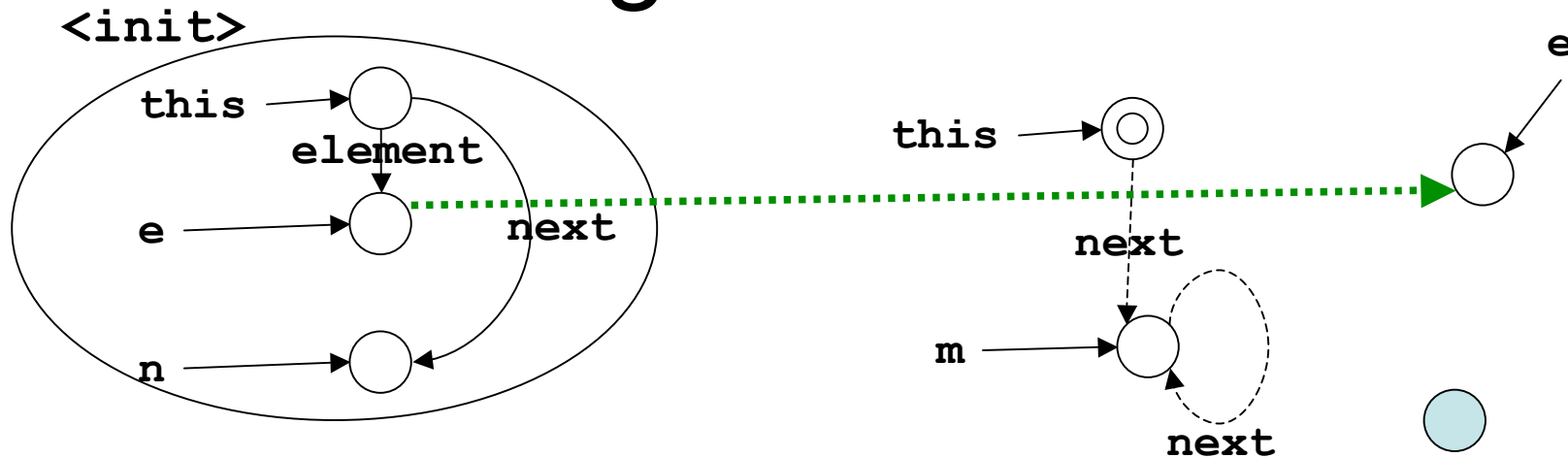
```
multisetElement insert(Object e){  
    multisetElement m = this  
    while (..) {  
        if (..) return this;  
        m = m.next;  
    }  
    return new multisetElement(e, this);  
}
```

Figure 4 – cont.



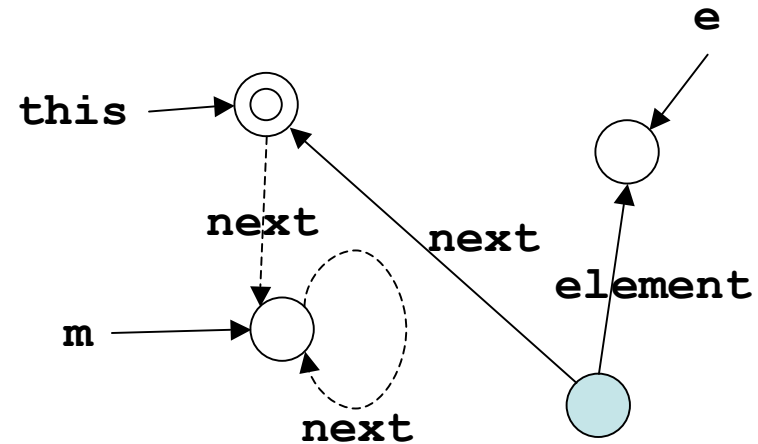
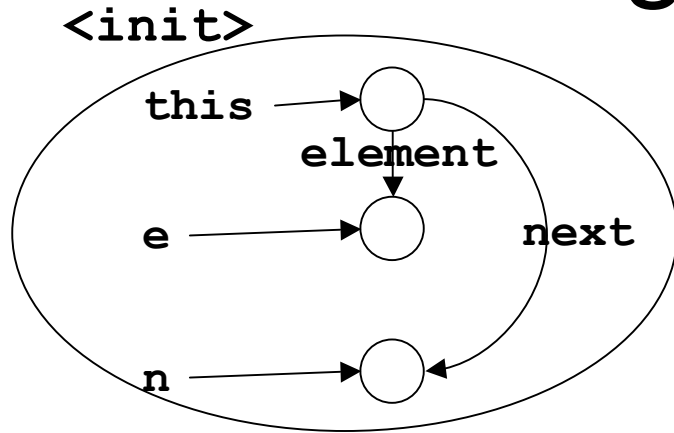
```
multisetElement insert(Object e){
    multisetElement m = this
    while (..) {
        if (..) return this;
        m = m.next;
    }
    return new multisetElement(e, this);
}
```

Figure 4 – cont.



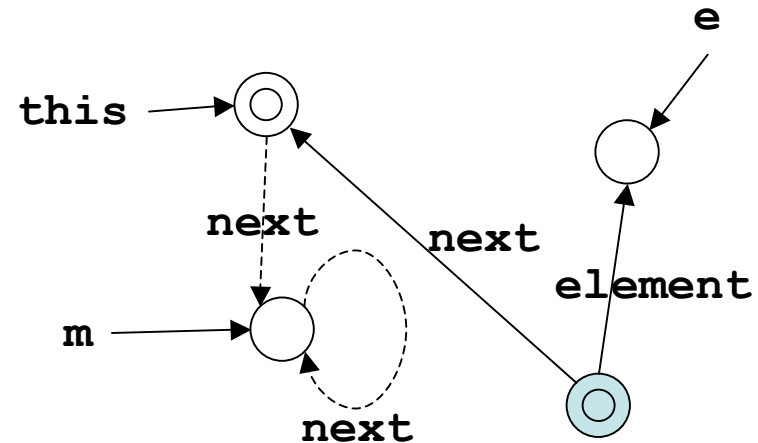
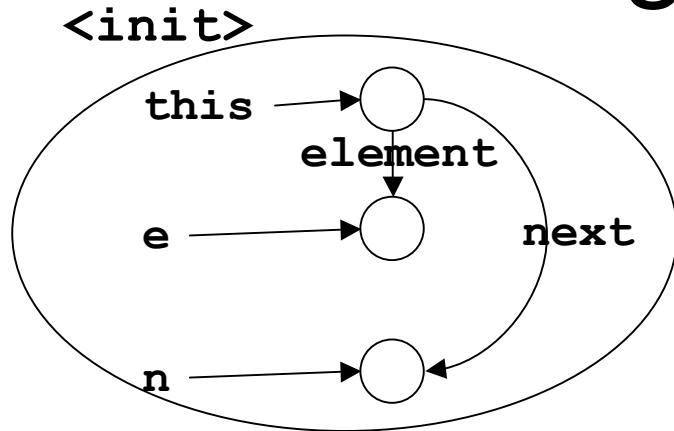
```
multisetElement insert(Object e){
    multisetElement m = this
    while (..) {
        if (..) return this;
        m = m.next;
    }
    return new multisetElement(e, this);
}
```

Figure 4 – cont.



```
multisetElement insert(Object e){
    multisetElement m = this
    while (..) {
        if (..) return this;
        m = m.next;
    }
    return new multisetElement(e, this);
}
```

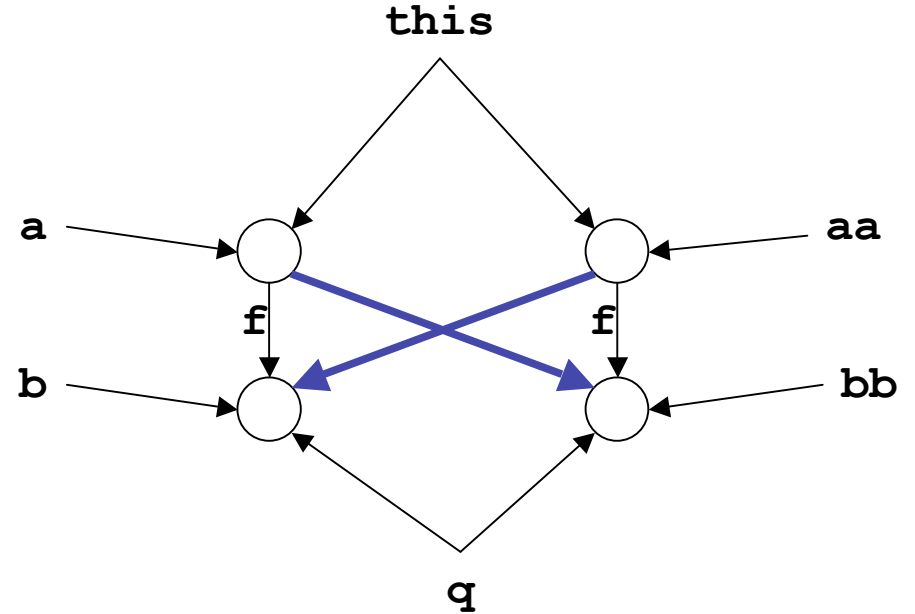
Figure 4 – cont.



```
multisetElement insert(Object e){
  multisetElement m = this
  while (..) {
    if (..) return this;
    m = m.next;
  }
  return new multisetElement(e, this);
}
```

Context Sensitive?

```
class A{
    X f;
    void m(X q){
        this.f = q;
    }
}
. . .
A a = new A(); //o1
b = new X();   //o2
a.m(b);
. . .
A aa = new A(); //o3
bb = new X();   //o4
aa.m(bb);
```



Context Sensitive?

```
class A{  
    X f;  
    void m(X q){  
        this.f = q;  
    }  
}
```

```
. . .  
A a = new A(); //o1  
b = new X();   //o2  
a.m(b);
```

```
. . .  
A aa = new A(); //o3  
bb = new X();   //o4  
aa.m(bb);
```


Context Sensitive?

```
class A{  
    X f;  
    void m(X q){  
        this.f = q;  
    }  
}
```

```
. . .  
A a = new A(); //o1  
b = new X();   //o2  
a.m(b);
```

```
. . .  
A aa = new A(); //o3  
bb = new X();   //o4  
aa.m(bb);
```

Context Sensitive?

```
class A{
    X f;
    void m(X q){
        this.f = q;
    }
}
```

this → ○

q → ○

```
. . .
A a = new A(); //o1
b = new X();   //o2
a.m(b);
```

```
. . .
A aa = new A(); //o3
bb = new X();   //o4
aa.m(bb);
```

Context Sensitive?

```
class A{  
    X f;  
    void m(X q){  
        this.f = q;  
    }  
}
```

this → ○

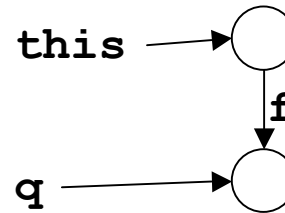
q → ○

```
. . .  
A a = new A(); //o1  
b = new X(); //o2  
a.m(b);
```

```
. . .  
A aa = new A(); //o3  
bb = new X(); //o4  
aa.m(bb);
```

Context Sensitive?

```
class A{  
    X f;  
    void m(X q){  
        this.f = q;  
    }  
}
```

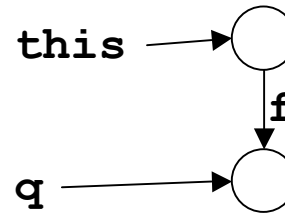


```
. . .  
A a = new A(); //o1  
b = new X();   //o2  
a.m(b);
```

```
. . .  
A aa = new A(); //o3  
bb = new X();   //o4  
aa.m(bb);
```

Context Sensitive?

```
class A{  
    X f;  
    void m(X q){  
        this.f = q;  
    }  
}
```

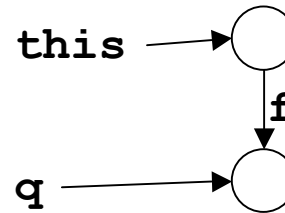


```
. . .  
A a = new A(); //o1  
b = new X(); //o2  
a.m(b);
```

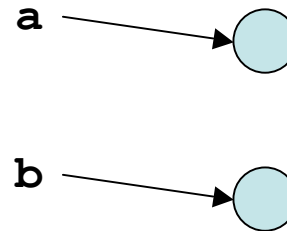
```
. . .  
A aa = new A(); //o3  
bb = new X(); //o4  
aa.m(bb);
```

Context Sensitive?

```
class A{  
    X f;  
    void m(X q){  
        this.f = q;  
    }  
}
```



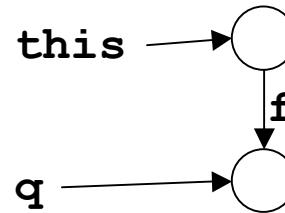
```
. . .  
A a = new A(); //o1  
b = new X();   //o2  
a.m(b);
```



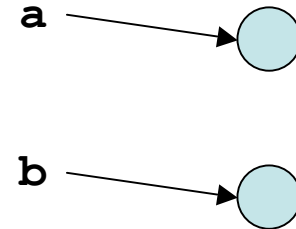
```
. . .  
A aa = new A(); //o3  
bb = new X();   //o4  
aa.m(bb);
```

Context Sensitive?

```
class A{  
    X f;  
    void m(X q){  
        this.f = q;  
    }  
}
```



```
. . .  
A a = new A(); //o1  
b = new X(); //o2  
a.m(b);
```



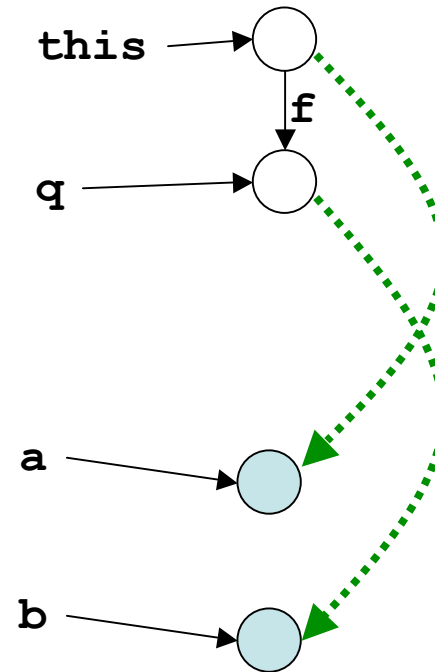
```
. . .  
A aa = new A(); //o3  
bb = new X(); //o4  
aa.m(bb);
```

Context Sensitive?

```
class A{  
    X f;  
    void m(X q){  
        this.f = q;  
    }  
}
```

```
. . .  
A a = new A(); //o1  
b = new X();   //o2  
a.m(b);
```

```
. . .  
A aa = new A(); //o3  
bb = new X();   //o4  
aa.m(bb);
```

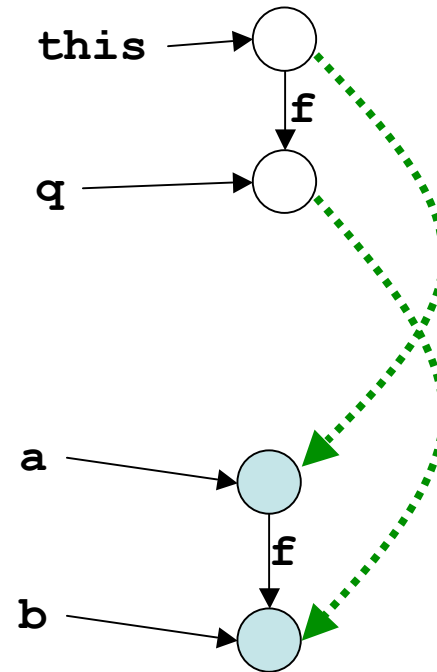


Context Sensitive?

```
class A{  
    X f;  
    void m(X q){  
        this.f = q;  
    }  
}
```

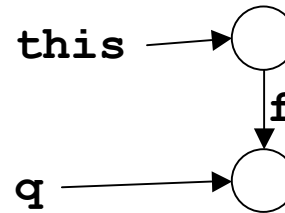
```
. . .  
A a = new A(); //o1  
b = new X();   //o2  
a.m(b);
```

```
. . .  
A aa = new A(); //o3  
bb = new X();   //o4  
aa.m(bb);
```

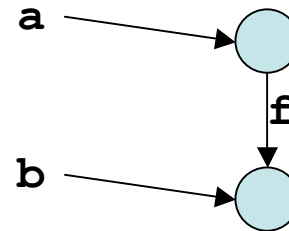


Context Sensitive?

```
class A{  
    X f;  
    void m(X q){  
        this.f = q;  
    }  
}
```



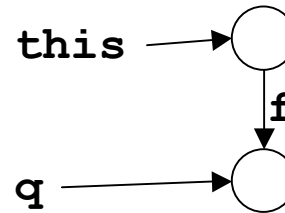
```
. . .  
A a = new A(); //o1  
b = new X();   //o2  
a.m(b);
```



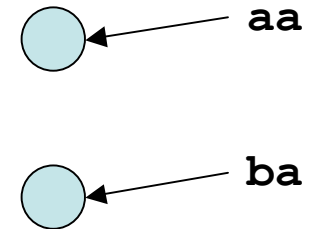
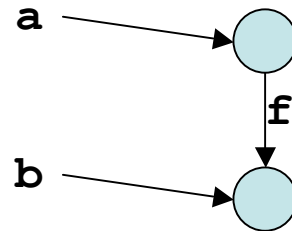
```
. . .  
A aa = new A(); //o3  
bb = new X();   //o4  
aa.m(bb);
```

Context Sensitive?

```
class A{  
    X f;  
    void m(X q){  
        this.f = q;  
    }  
}
```



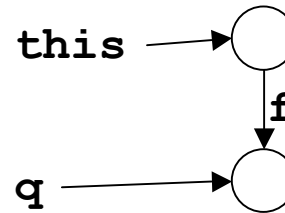
```
. . .  
A a = new A(); //o1  
b = new X();   //o2  
a.m(b);
```



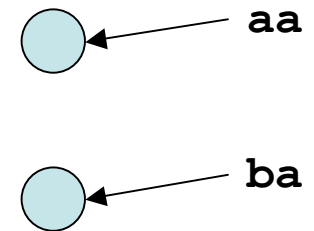
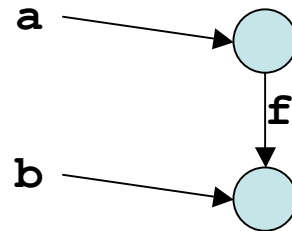
```
. . .  
A aa = new A(); //o3  
bb = new X();   //o4  
aa.m(bb);
```

Context Sensitive?

```
class A{  
    X f;  
    void m(X q){  
        this.f = q;  
    }  
}
```



```
. . .  
A a = new A(); //o1  
b = new X(); //o2  
a.m(b);
```



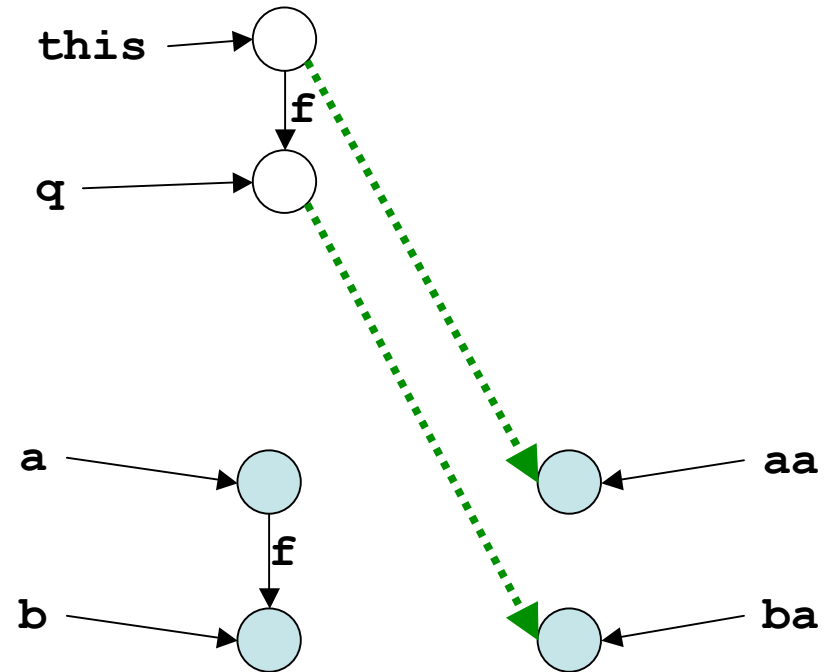
```
A aa = new A(); //o3  
bb = new X(); //o4  
aa.m(bb);
```

Context Sensitive?

```
class A{  
    X f;  
    void m(X q){  
        this.f = q;  
    }  
}
```

```
. . .  
A a = new A(); //o1  
b = new X();   //o2  
a.m(b);
```

```
. . .  
A aa = new A(); //o3  
bb = new X();   //o4  
aa.m(bb);
```

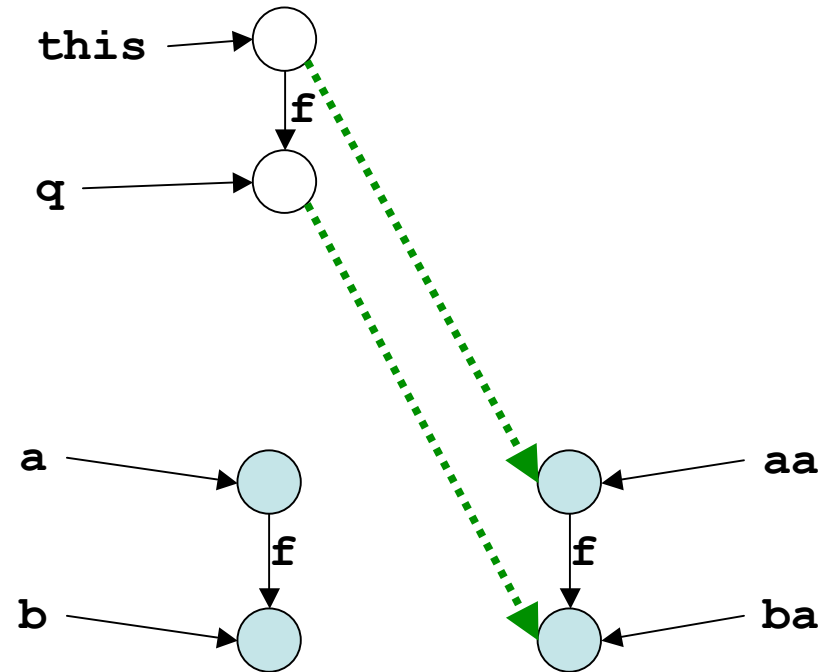


Context Sensitive?

```
class A{  
    X f;  
    void m(X q){  
        this.f = q;  
    }  
}
```

```
. . .  
A a = new A(); //o1  
b = new X(); //o2  
a.m(b);
```

```
. . .  
A aa = new A(); //o3  
bb = new X(); //o4  
aa.m(bb);
```



Contribution

- An idea to summarize the points-to effect of any partial program including:
 - Algorithm of summarizing
 - Algorithm of merging.
 - Usage Example.

Questions

- Why flow sensitive?
 - No comparison in the experiment section.
- What's wrong with Jikes RVM in the benchmark?
- Complexity?