Java

13 2 22

12

Java

Java

Calling-Context

Java

GUI

(Program Slice)

Calling-Context

(GUI)

1

P (*Program Slice*) P n
v n v P
Mark Weiser [1] ,

.

(C)
Calling-Context[4] Calling-
Context Calling-Context

Calling-Context

Calling-Context
(4)

(
,*GUI*) (5) GUI

2 3 Java
4
5
6

2

(*Program Slicing*) P s
v ,v s w
w $\mathcal{Q}$
(*Slice*) Mark Weiser[1]

.

2.1

Phase 1: (

Phase 2: (*Program Dependence Graph,PDG*)

Phase 3: PDG

2.1.1 Phase 1:

Phase1
([6]) 2

- p s_1 s_2
  s_1 s_2 *(Control Dependence CD)*
  - s_1
  - s_2 s_1
- p s_1 s_2 3
  s_1 s_2 v *(Data Dependence DD)*
  - s_1 v
  - s_2 v

– s_1 s_2 s_1 s_2
v

[1](2.2
[?](2.3)
DC [3](2.4)

2.1.2 Phase 2:

Phase2 Phase1
(Program Dependence Graph PDG) PDG

,
(
)
[4, 6]

2.1.3 Phase 3:

Phase3 Phase2 PDG

PDG

2.2

2.2.1

[5] [4, 6]
(PDG)

-

PDG PDG
if while

(Reaching Definitions RD)

PDG t RD v s $\langle v, s \rangle$

1. s v
2. s t v

t RD $\langle v, s \rangle$ t v s t
(DD(s, v, t))

s t s t
(CD(s, t)) if while

PDG

( )

PDG PDG

RD PDG

[6].

PDG G G E

S v V

1. $V \leftarrow \{n | n$ S $\}$
2. $N \leftarrow \{n | \langle v, n \rangle \in RD_{in}(S)\}$
3. $N \neq \phi$
   - $l \in N$
   - $N \leftarrow N - \{l\}$
   - $V \leftarrow V \cup \{l\}$
   - $N \leftarrow N \cup \{k | (k \notin V) \wedge (k \xrightarrow{\star} l \in E \vee k \dashrightarrow l) \wedge (k$ *g–in* $) \wedge (k$ *para–in* $)\}($ $k \xrightarrow{\star} l$ k l $)$
   - k *para–in* *para–in*
     $N \leftarrow N \cup \{k | (k \notin V) \wedge (k \xrightarrow{\star} l \in E \vee k \dashrightarrow l)\}($ $k \xrightarrow{\star} l$
     k l $)$

2.3

2.3.1

p p [?]

1. **Definition-Use DU**

p =Def(q, w) q w p

$p < r < q$ r

w p

Ins(p) p Def(p) p Ins(p)

Use(p) p Ins(p)

p $q(p < q)$

DU(p, q) $w \in$ Use(q) p =Def(q, w)

DU(p, q) p w q

$w \in$ Use(q)

DU$w(p, q)$

2. **Test-Control TC**

t CtlExec(t)

CtlExec(t) ={ $s|$CD(s, t)}

t t CtlExec(t)

p $q(p < q)$ TC(p, q)

p=max{ $i|i < q$ Ins$(i) \in$ CtlExec(Ins(q))}

TC(p, q) q Ins(q)

p Ins(p)

q Ins(q)

q TC(p, q)

q Ins(q)

v p

r $\langle v, p \rangle$ p

$VarREF(p)$

$$VarREF(p) \equiv \{ \langle v, q \rangle \mid \mathrm{DU}v(q, p) \}$$

p q Control(p)

P

$C = (x, r, V)$ x, r, V

- x P
- r P x
- V P

P $C = (x, r, V)$

1. DS $\leftarrow \phi$
   CalcDsObj $\leftarrow \phi$
   CalcDsObj

2. DS $\leftarrow$ {Ins(r)}
   CalcDsObj $\leftarrow$ { q | $v \in$ V q=Def(r, v)}

3. $p \in$ CalcDsObj
   CalcDsObj $\leftarrow$ { $q \mid VarREF(p) \cup$ Contorol(p)} $\cup$ CalcDsObj
   DS $\leftarrow$ DS $\cup$ {Ins(p)}
   CalcDsObj $\leftarrow$ CalcDsObj - {p}

4. CalcDsObj 3 CalcDsObj DS
   P $C = (x, r, V)$

2.4 Dependence-Cache

(

)

( )

( )

DC [?]

2.4.1 Dependence-Cache

s v v (t)

$DD(t, v, s)$ v

s $DDS(s)$

$DDS(s)$ 2 (“ v” “s

”)

v $DefS(v)$ v

$DefS$

(1) s $DDS(s) = \phi$

(2) s

- s v [1] $DDS(s) \leftarrow DDS(s) \cup (v, DefS(v))$
- s v $DefS(v) = s$

[1]

$**v$ 2 v $*v$ $**v$

$*v$ v

$DDS(s)$ $DDS(s) =$

$\{(v,t)|DD(t,v,s)$ $\}$

(3) DDS PDG s

- $DDS(s) \neq \phi$
  - $DDS(s) \leftarrow DDS(s) - \{(v,t)\}$
  - $t \xrightarrow{v} s$
- s t $s \dashrightarrow t$

2.5

DC 1

[3]

: $\geq$ DC $\geq$

: $>$ DC $>$

: $>$ DC $>$

1:

			DC
PDG			

C $(23,d)$

1 2 DC 2

C

```
 1:  #include <stdio.h>
 2:  #define SIZE 5
 3:
 4:  int cube(int x) {
 5:      return x*x*x;
 6:  }
 7:
 8:  void main(void)
 9:  {
10:      int a[SIZE];
11:      int b[SIZE];
12:      int c, d, i;
13:
14:      a[0] = 0;
15:      a[1] = -1;
16:      a[2] = 2;
17:      a[3] = -3;
18:      a[4] = 4;
19:
20:      for (i=0;i<SIZE;i++) {
21:          b[i] = a[i];
22:      }
23:
24:      printf("Input: ");
25:      scanf("%d", &c);
26:
27:      if (c >= SIZE) {
28:          c = c % SIZE;
29:      }
30:
31:      d = cube(b[c]);
32:
33:      if (d < 0) {
34:          d = -1 * d;
35:      }
36:
37:      printf("%d\n", d);
38:  }
```

```
 1:  #include <stdio.h>
 2:  #define SIZE 5

 4:  int cube(int x) {
 5:      return x*x*x;
 6:  }

 8:  void main(void)
 9:  {
10:      int a[SIZE];
11:      int b[SIZE];
12:      int c, d, i;

14:      a[0] = 0;
15:      a[1] = -1;
16:      a[2] = 2;
17:      a[3] = -3;
18:      a[4] = 4;

20:      for (i=0;i<SIZE;i++) {
21:          b[i] = a[i];
22:      }


25:      scanf("%d", &c);

27:      if (c >= SIZE) {
28:          c = c % SIZE;
29:      }

31:      d = cube(b[c]);

33:      if (d < 0) {
34:          d = -1 * d;
35:      }

37:      printf("%d\n", d);
38:  }
```

1: C $(23, d)$ 1

```
 1:  #include <stdio.h>
 2:  #define SIZE 5

 4:  int cube(int x) {
 5:      return x*x*x;
 6:  }

 8:  void main(void)
 9:  {
10:      int a[SIZE];
11:      int b[SIZE];
12:      int c, d, i;



16:      a[2] = 2;



20:      for (i=0;i<SIZE;i++) {
21:          b[i] = a[i];
22:      }


25:      scanf("%d", &c);







31:      d = cube(b[c]);









37:      printf("%d\n", d);
38:  }
```

DC

```
 1:  #include <stdio.h>
 2:  #define SIZE 5

 4:  int cube(int x) {
 5:      return x*x*x;
 6:  }

 8:  void main(void)
 9:  {
10:      int a[SIZE];
11:      int b[SIZE];
12:      int c, d, i;

14:      a[0] = 0;
15:      a[1] = -1;
16:      a[2] = 2;
17:      a[3] = -3;
18:      a[4] = 4;

20:      for (i=0;i<SIZE;i++) {
21:          b[i] = a[i];
22:      }


25:      scanf("%d", &c);







31:      d = cube(b[c]);









37:      printf("%d\n", d);
38:  }
```

2: C $(23, d)$ 2

3 Java

C Java, C++

Java[12, 13, 15] Java

3.1

Java

Java

DC [10] DC **Calling-Context**[4](4.2

4

(3)

1.

2.

PDG

3. GUI

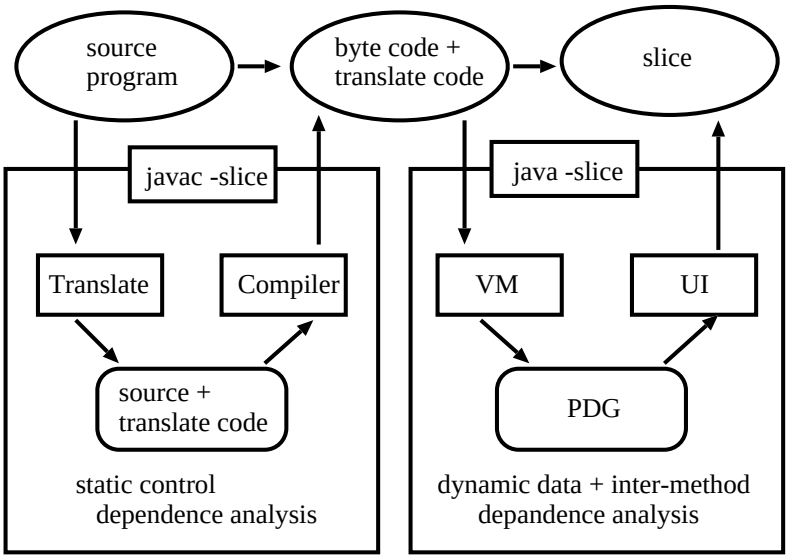
3.2

(4)

(*GUI*)(5)

4 5

GUI


source
program
byte code +
translate code
slice
javac -slice
java -slice
Translate
Compiler
VM
UI
source +
translate code
PDG
static control
dependence analysis
dynamic data + inter-method
depandence analysis

3: Java

4

Calling-Context[4]

4.1

-

PDG

PDG

[?]

2

2:

global-def-in	/
global-def-out	/
global-out	/ ( )
parameter-in	/
parameter-out	/ ( )
actual-in	/ /
actual-out	/ /
func-return	
func-exit	

4.2 Calling-Context

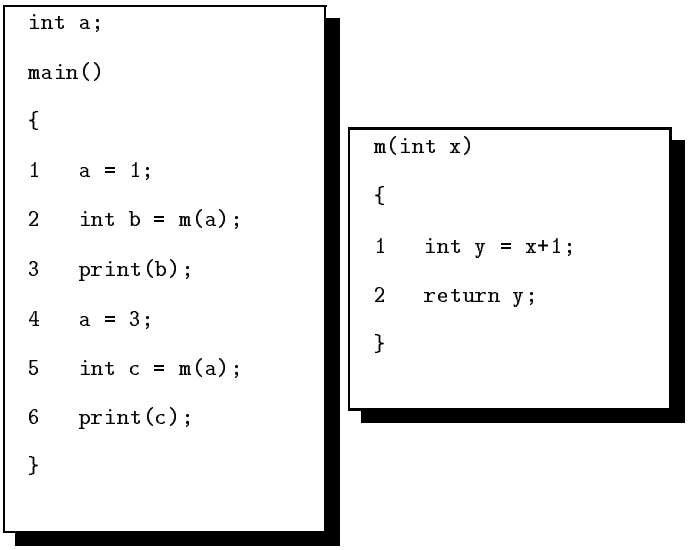
Calling-Context[4]

4

4 main 6 c Calling-Context

5 m main 5

Calling-Context 5 m main



4: Calling-Context

2,5

1,2

Java

Java

.

Calling-Context

4.3

Calling-Context

PDG

(index)

PDG

PDG Calling-

Context $index = 0$

(CALL

RETURN) index

PDG CALL

RETURN

4.4 PDG index( ) 4.5 index PDG

4.4 Add Method-Indexed-Edge

Add Method-Indexed-Edge PDG

index retun (return)

PDG index

index

5 Java [2] Add Method-Indexed-Edge PDG 7

```
int a = 1;  // global
main(){
   int b = 1;
   int c = m(b);
   print(c);
   if (c > 5){;
      int d = m(c);
      print(d);
    }
}
```

```
m(int x)
{
   int y = n(x+a);
   return y;
}
```

```
n(int s)
{
   int t = s*2;
   return t;
}
```

5: Java

[2] Java ,System.out.println(c) print(c)

ADD METHOD-INDEXED-EDGE

s PDG n_s

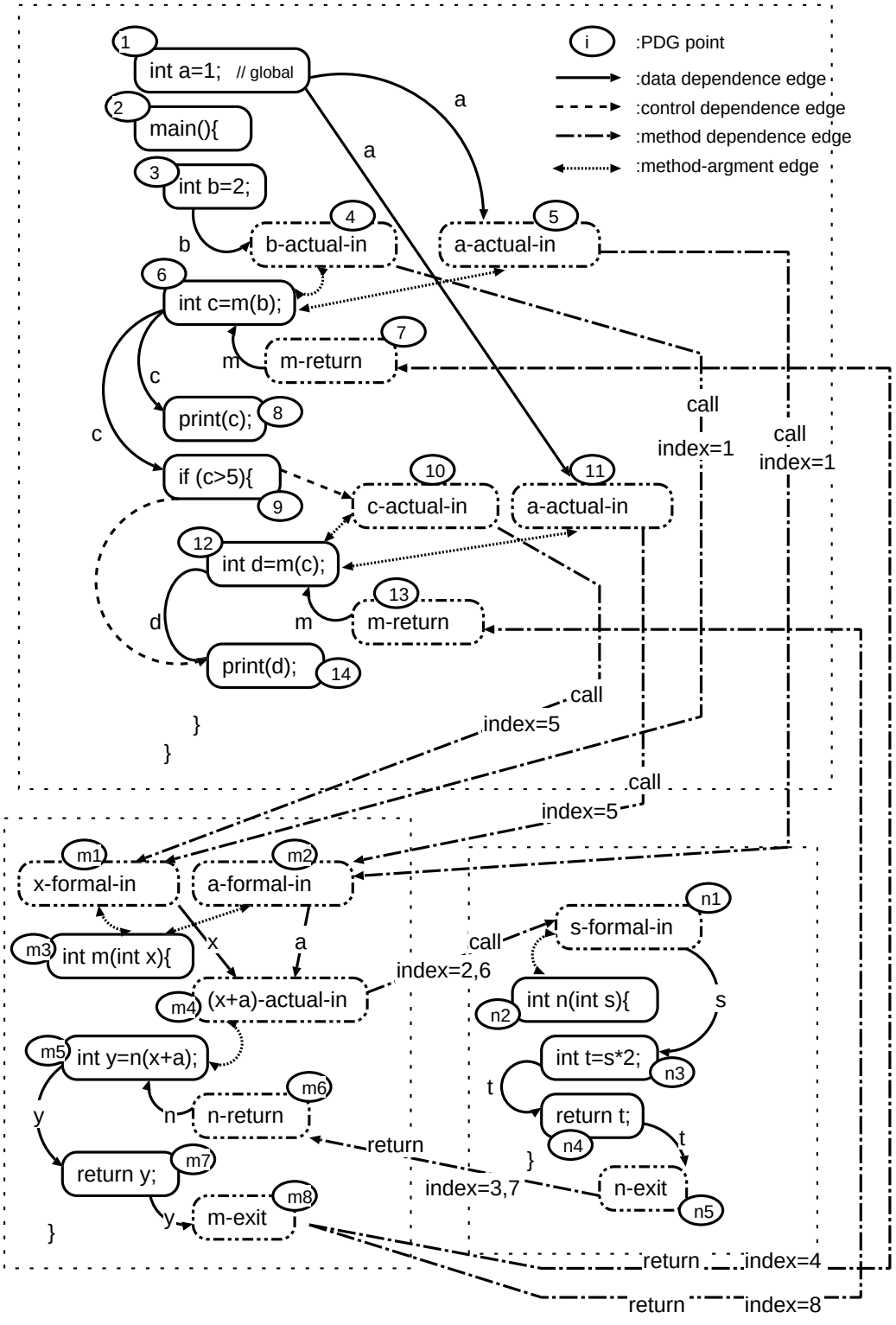
Java

DD-Edge index PDG

PDG DD-Edge index

```
(1) call_number = 1
(2) while then begin
(3)   if s t x then begin
(4)   n_t ← n_s x
(5)   end
(6)   else if s t then begin
(7)     n_t ← n_s x
(8)   end
(9)   else if s then begin
(10)    call_number = call_number + 1
(11)    while then begin
(12)      index = call_number
(13)    end
(14)  end
(15)  else if s return then begin
(16)    call_number = call_number + 1
(17)    while then begin
(18)      index = call_number parameter return
(19)    end
(20)  end
(21) end
```

6: ADD METHOD-INDEXED-EDGE


i :PDG point
:data dependence edge
:control dependence edge
:method dependence edge
:method-argment edge
1 int a=1; // global
2 main(){
3 int b=2;
4 b-actual-in
5 a-actual-in
6 int c=m(b);
7 m-return
print(c); 8
if (c>5){ 9
10 c-actual-in
11 a-actual-in
12 int d=m(c);
13 m-return
print(d); 14
}
}
a
b
c
m
d
call
index=1
index=5
m1 x-formal-in
m2 a-formal-in
m3 int m(int x){
x
a
m4 (x+a)-actual-in
index=2,6
m5 int y=n(x+a);
n
m6 n-return
return
y
return y; m7
m8 m-exit
}
n1 s-formal-in
s
n2 int n(int s){
int t=s*2; n3
t
return t; n4
}
n-exit n5
index=3,7
return
index=4
index=8

7:

4.5 Trace Indexed-Edge PDG

Trace Indexed-Edge PDG

index PDG

Calling-Context CALL RETURN

index index

$index = n$ RETURN RETUN

CALL CALL CALL

index 1

RETURN CALL index

$index <= n$ **index** **CALL**

$index = 13$ RETURN M

M CALL

CALL $index = 4, 7, 10$

RETURN CALL $index = 10$ CALL

CALL

RETURN index n RETURN CALL

index m $m == n$ $m == n - 1$ CALL

RETURN $m < n - 1$ CALL

RETURN

$m <= x < n - 1$ x

index

index

HashTabale $index = n$

HashTable[n]

$index = n$

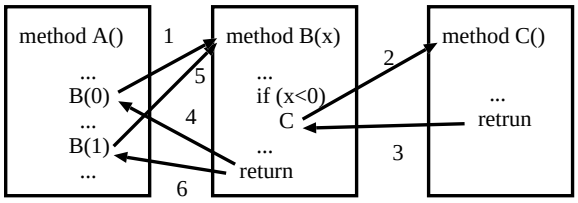
HashTable[n]

PDG

8

8 method A $index = 6$ RETURN

method B . RETURN CALL



8:

$index = 5$ $index = 3$

$index = 4$ RETURN

$index = 3$ method C

index

CALL CALL index

CALL index index CALL

2

PDG s s

$index == n$ RETURN s

- t
  - index
  - index CALL $CallIndex \leftarrow CallIndex \cup \{index\}$
  - $index = m$ $m == n$ $m == n - 1$

    $HashTable[index] \leftarrow HashTable[index] \cup \{t\}$

    $TraceIndex \leftarrow TraceIndex \cup \{index\}$
  - i$index = m$ $m < n - 1$

    $HashTable[index] \leftarrow HashTable[index] \cup \{t\}$ $CheckIndex \leftarrow CheckIndex \cup \{index\}$

  * CALL CheckIndex $m <= x < n - 1$
    x TraceIndex
  – index index n

- t
  – n
    CallIndex $m <= n$ index m
    CheckIndex $m <= x < n - 1$ x TraceIndex

CallIndex TraceIndex CheckIndex CALL index
index
index

9 Trace Indexed-Edge PDG
10 Check PDG-Point Trace Indexed-Edge PDG
Check PDG-Point

TRACE INDEXED-EDGE PDG

PDG DD-Edge index

HashTable[i],

index TraceIndex

index CheckIndex

v(=PDG s)

DD-Edge index PDG

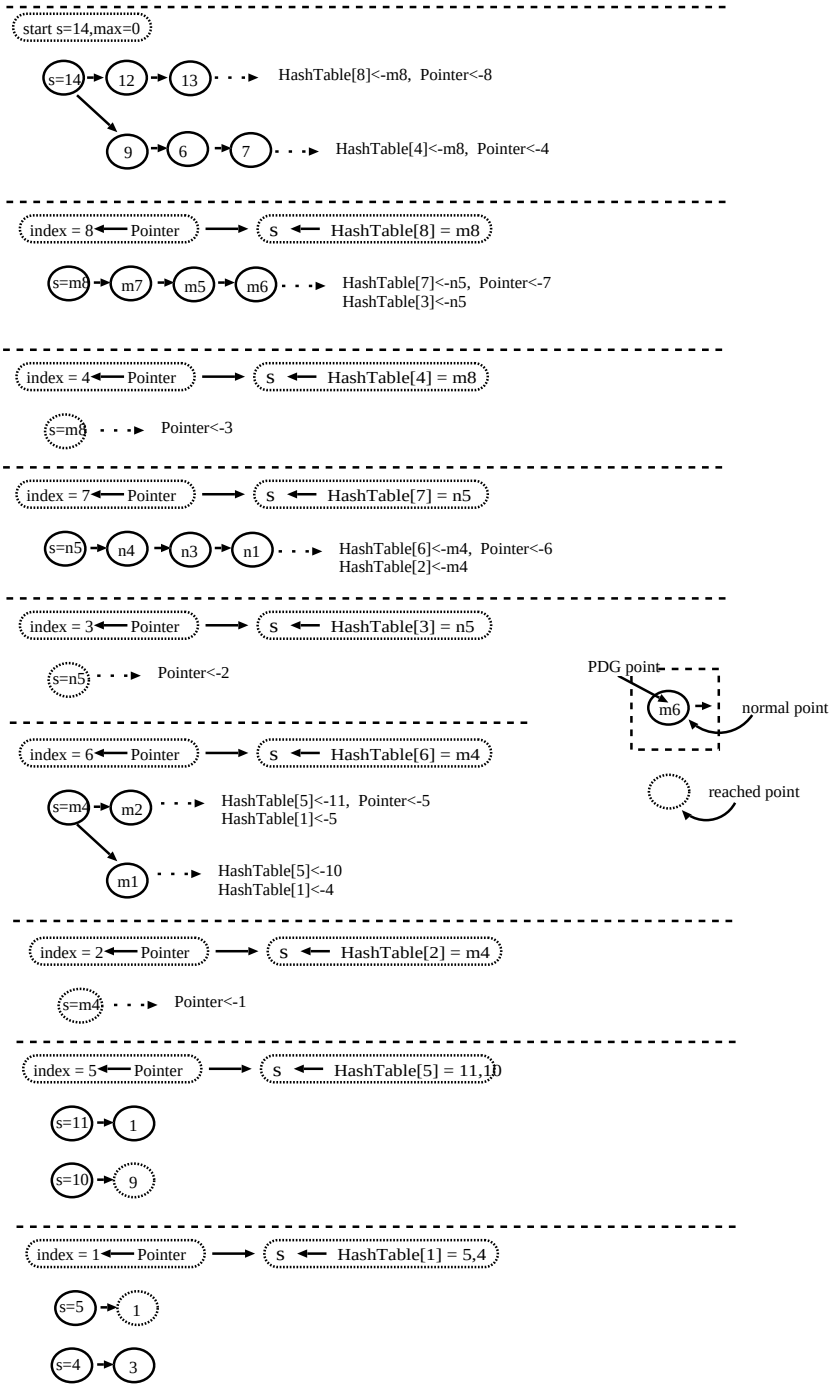
(1) **while** s t **then begin**
(3) **if** s PDG e index **then begin**
(4) **if** e v **then begin**
(5) Check PDG(t,0)
(6) **end**
(7) **else if** e index **then begin**
(8) Check PDG(t,index)
(9) **end**
(10) **while** TraceIndex **then begin**
(11) TraceIndex i
(12) **while** HashTable[i] **then begin**
(13) HashTable[i] s
(14) Check PDG-Point(s,i)
(15) **end**
(16) **end**
(17) T

9: TRACE INDEXED-EDGE PDG

CHECK PDG-POINT
PDG DD-Edge index
CALL index CallIndex
PDG s, index max
T
PDG

Algorithm Check PDG-Point(s,max)

(1) $T \leftarrow T \cup \{s\}$
(2) **while** s **then begin**
(3) **if** s PDG e index **then begin**
(4) **if** t T **then begin**
(5) Check PDG(t,max)
(6) **end**
(7) **else if** t T $max \neq 0$**then begin**
(8) $\{n \mid n, \forall a \subseteq CallIndex \wedge a <= n <= max\}$ n
$\{m \mid \forall m \subseteq CheckIndex \wedge n <= m <= max\}$ m
$TraceIndex \leftarrow TraceIndex \cup \{m\}$
(9) **end**
(10) **end**
(11) **else if** e index **then begin**
(12) **if** t T **then begin**
(13) **if** CALL **then begin**
(14) $CallIndex \cup \{index\}$
(15) **end**
(16) **if** $index == max - 1$ $max == 0$ **then begin**
(17) $HashTable[index] \leftarrow HashTable[index] \cup \{t\}$ $TraceIndex \leftarrow TraceIndex \cup \{index\}$
(18) **end**
(19) **else if** $index < max - 1$ **then**
(20) $HashTable[index] \leftarrow HashTable[index] \cup \{t\}$
(21) **if** CALL **then**
(22) $\{m \mid \forall m \subseteq CheckIndex \wedge n <= m <= max\}$ m
$TraceIndex \leftarrow TraceIndex \cup \{m\}$
(23) **end**
(24) **end**
(25) **end**
(26) **else if** t T **then begin**
(27) $\{n \mid n, \forall a \subseteq CallIndex \wedge a <= n <= max\}$ n
$\{m \mid \forall m \subseteq CheckIndex \wedge n <= m <= max\}$ m
$TraceIndex \leftarrow TraceIndex \cup \{m\}$
(28) **end**
(29) **end**
(30) **end**


start s=14,max=0
s=14
12
13
HashTable[8]<-m8, Pointer<-8
9
6
7
HashTable[4]<-m8, Pointer<-4
index = 8
Pointer
s
HashTable[8] = m8
s=m8
m7
m5
m6
HashTable[7]<-n5, Pointer<-7
HashTable[3]<-n5
index = 4
Pointer
s
HashTable[4] = m8
s=m8
Pointer<-3
index = 7
Pointer
s
HashTable[7] = n5
s=n5
n4
n3
n1
HashTable[6]<-m4, Pointer<-6
HashTable[2]<-m4
index = 3
Pointer
s
HashTable[3] = n5
s=n5
Pointer<-2
index = 6
Pointer
s
HashTable[6] = m4
s=m4
m2
HashTable[5]<-11, Pointer<-5
HashTable[1]<-5
m1
HashTable[5]<-10
HashTable[1]<-4
index = 2
Pointer
s
HashTable[2] = m4
s=m4
Pointer<-1
index = 5
Pointer
s
HashTable[5] = 11,10
s=11
1
s=10
9
index = 1
Pointer
s
HashTable[1] = 5,4
s=5
1
s=4
3
PDG point
m6
normal point
reached point

11:

4.6

Java

- (method-indexed Slice)
- (all-indexed Slice)
- (DC Slice)

3

-
-
-

Calling-

Context

4.6.1

4.6.1

PDG index

PDG

index

12

TRACE ALL-INDEXED PDG

index

PDG s, index max

PDG

Algorithm Trace all-indexed PDG(s,max)

(1) **if** s T **then begin**

(2) $T \leftarrow T \cup \{s\}$

(3) **end**

(4) **while** s **then begin**

(5) **if** $index < max$ index **then begin**

(6) index m Trace all-indexed PDG(t,m)

(7) **end**

(8) **end**

12: TRACE ALL-INDEXED PDG

4.6.2

- (21 3)

3

3:

	method-indexed Slice	all-indexed Slice	DC Slice
( )	9	8	13
index	6	20	-
PDG (ms)	3072	3566	2795
PDG (ms)	583	784	942

Calling-Context

PDG

- (32) 4

4:

	method-indexed Slice	all-indexed Slice	DC Slice
( )	14	14	14
index	0	80	-
PDG (ms)	2769	3155	2718
PDG (ms)	1101	1230	1060

-

(241) 5

5:

	method-indexed Slice	all-indexed Slice	DC Slice
( )	42	40	83
index	3316	19680	-
PDG (ms)	55817	331256	15016
PDG (ms)	7052	1275834	3489

1

1 (

)

1

1

4.6.3

Calling-Context

5

3 Java
(GUI)

5.1

(Prgram Slice)[1] SeeSoft[7]
AT&T SeeSlice[8] SeeSlice
[9]

- ( )
-
-
-
-

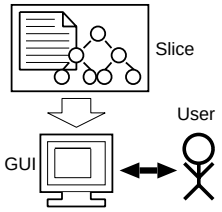
SeeSlice
Java

(13 GUI)

1:

2: ( )

3: (
)



13: ( )

5.2

5.2

Java GUI [5, 11] Java[12, 13, 14] 3100 Java

5.2.1

Java Java

( )

Java ⇒ Execute (Compile)

14

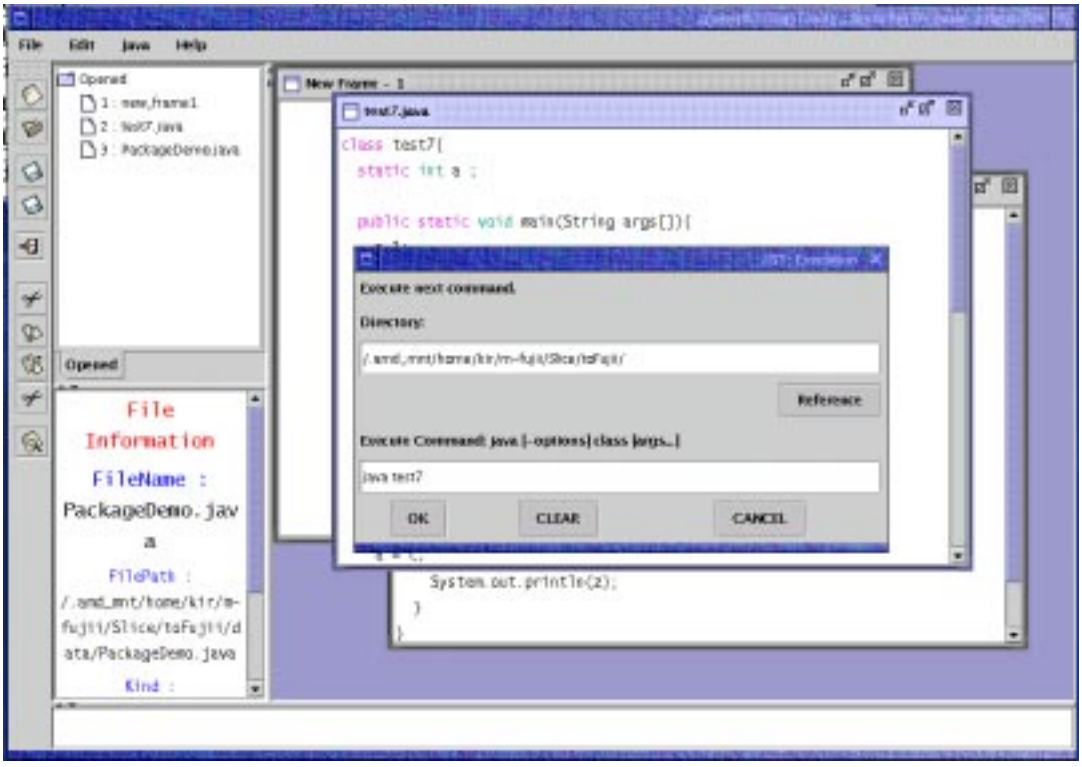
Reference

14

Execute Command OK

5.2.2

(15)



14:

Java ⇒ Slice ⇒ Translate

Java

( )

PDG

Java ⇒ Slice ⇒ Get PDG

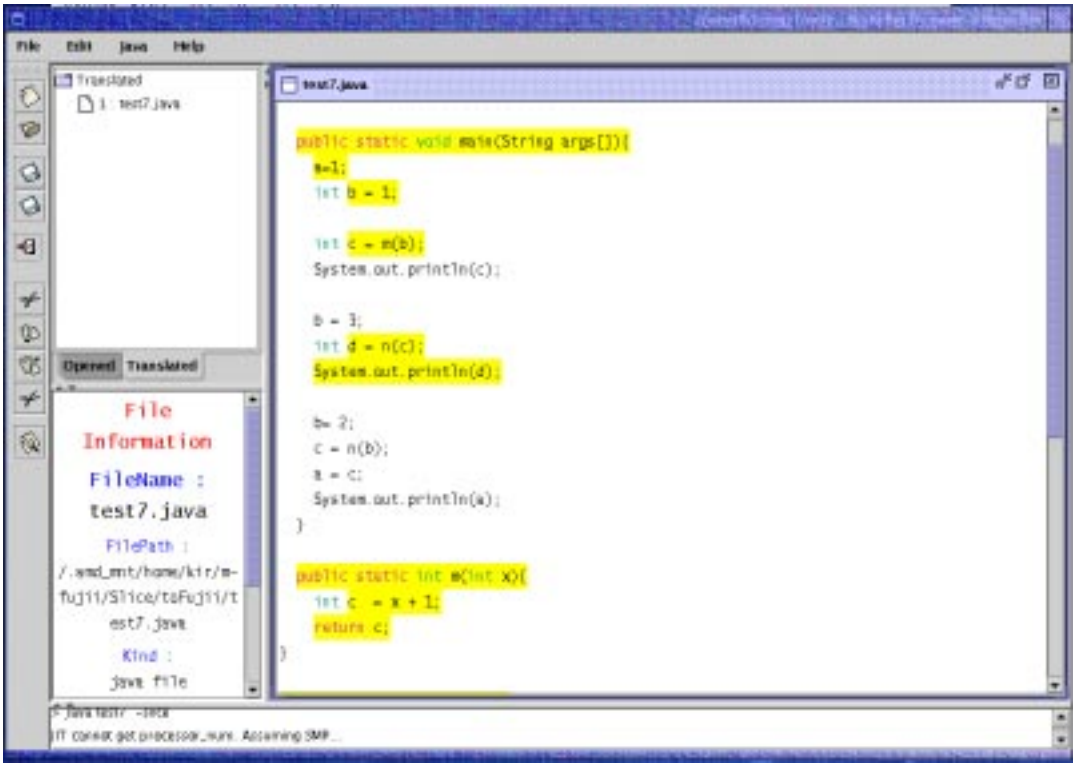
Complete Translate.Get PDG at once?

PDG

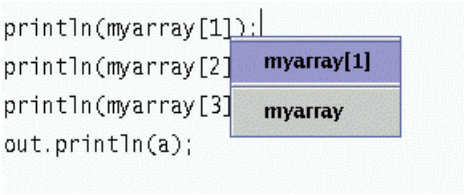
Java ⇒ Slice ⇒ Criterion

(16)

PDG 16


File
Edit
Java
Help
Translated
1 test7.java
Opened
Translated
File
Information
FileName :
test7.java
FilePath :
/.amd_mnt/home/kir/m-fuji1/Slice/toFuji1/test7.java
Kind :
java file
test7.java
public static void main(String args[]){
a=1;
int b = 1;
int c = m(b);
System.out.println(c);
b = 3;
int d = n(c);
System.out.println(d);
b= 2;
c = n(b);
a = c;
System.out.println(a);
}
public static int m(int x){
int c = x + 1;
return c;
}

15:


println(myarray[1]);
println(myarray[2]
println(myarray[3]
out.println(a);
myarray[1]
myarray

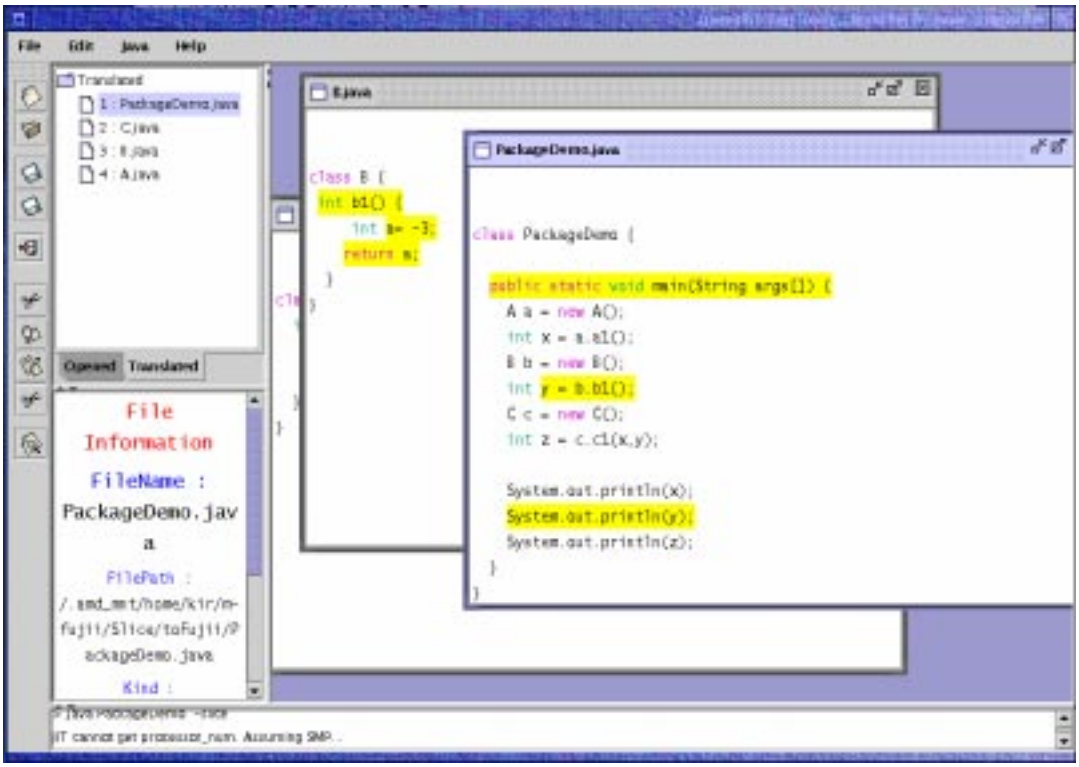
16:

5.2.3

Java

GUI

17



17:

6

Java

Calling-Context

Java

-
  - –
  - –
-
  - –

[1] Weiser M. : "Program Slicing", *Proceedings of the Fifth International Conference on Software Engineering*, pp. 439–449, 1981.

[2] Agrawal H. and Horgan, J. : "Dynamic Program Slicing", *SIGPLAN Notices, Vol. 25, No. 6*, pp. 246–256, 1990.

[3] Ashida Y., Ohata F. and Inoue K. : "Slicing Methods Using Static and Dynamic Information", *Proceedings of the 6th Asia Pacific Software Engineering Conference (APSEC'99)*, pp. 344–350, December 1999.

[4] Horwitz S. and Reps T. : "The Use of Program Dependence Graphs in Software Engineering", *Proceedings of the 14th International Conference on Software Engineering*, pp. 392–411, 1992.

[5] , , : " ", , *Vol. 37, No. 4*, pp. 536–545, 1996.

[6] , , , : " ", , *Vol. J78-D-I, No. 1*, pp. 11–22, 1995.

[7] Steffen G., Eick E., and Sumner Jr.: " Seesoft - a tool for visualizing line-oriented software statistics", *IEEE Trans. on Software Engineering*, Vol. 18, No. 11, pp.957–968, 1992.

[8] Thomas B., and Stephen G.E. : "Visualizing Program Slices", *in Proceedings of the 1994 IEEE Symposium on Visual Languages*, pp.288–295, 1994.

[9] : " ", *9* , 1998.

[10] : " Java ", , 2001.

[11] : "JAVA ", , 2000.

[12] Gosling J., Joy B. and Steele G., [ ] : "The Java ", Addison-Wesley Publishers Japan, Ltd., 1997.

[13] Joseph O., [ ] : “ Java”, SHOEISHA, 1999.

[14] Satyaraj P., [ ] : “ JavaSwing ”, SOFTBANK publishing, 1999.

[15] Java : “Java ”, , 2000.