Class Analysis Workshop

Programming Research Lab. / SNU / Korea

January 17, 2009

Contents

- Introduction
- Historical Overview
- Algorithm Overview
  - XTA / 0-CFA
  - k-CFA / Object-Sensitive
  - EPA / Refinement-Based
- Conclusion

RO
PAS

Introduction

Wontae Choi@ropas

Class Analysis Workshop
January 17, 2009

RO
PAS

What is it?

to extract RTTI
from source code
statically

RTTI : runtime type information

RO
PAS

What is it?

example

```
Parent * a;

if(....)
    a = new Child1();
else
    a = new Child2();

a->do();
```

RO
PAS

What is it?

example

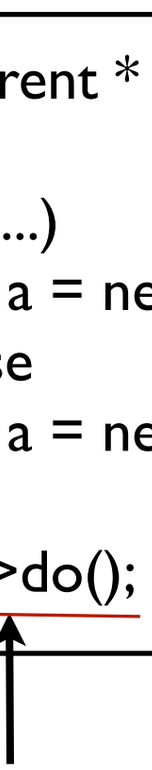
```
Parent * a;

if(....)
    a = new Child1();
else
    a = new Child2();

a->do();
```

RO
PAS

What is it?

example

```
Parent * a;

if(....)
    a = new Child1();
else
    a = new Child2();

a->do();
```



What is it?

example

```
Parent * a;

if(....)
    a = new Child1();
else
    a = new Child2();

a->do();
```

what is runtime type of a?

RO
PAS

What is it?

example

```
Parent * a;

if(....)
    a = new Child1();
else
    a = new Child2();

a->do();
```

what is runtime type of a?

Answer = {Child1, Child2}

RO
PAS

What is it?

Similar things

Class Analysis : extract RTTI

Alias Analysis : extract aliased variables

Point-to Analysis : extract referencing location

ROPAS

Where to use?

Virtual Function Resolution (Compiler)

Call Graph Construction (Analysis)

Other Analysis (Analysis)

RO
PAS

How to evaluate?
What is measure?

- size of points-to set
- # of call-edge
- # of resolved virtual call

- client analysis

RO
PAS

Historical Overview

Part I
context-insensitive family

CHA / RTA / XTA / Anderson / 0-CFA

RO
PAS

Example #1

cf Barbara G. Ryder, ETAPS 03

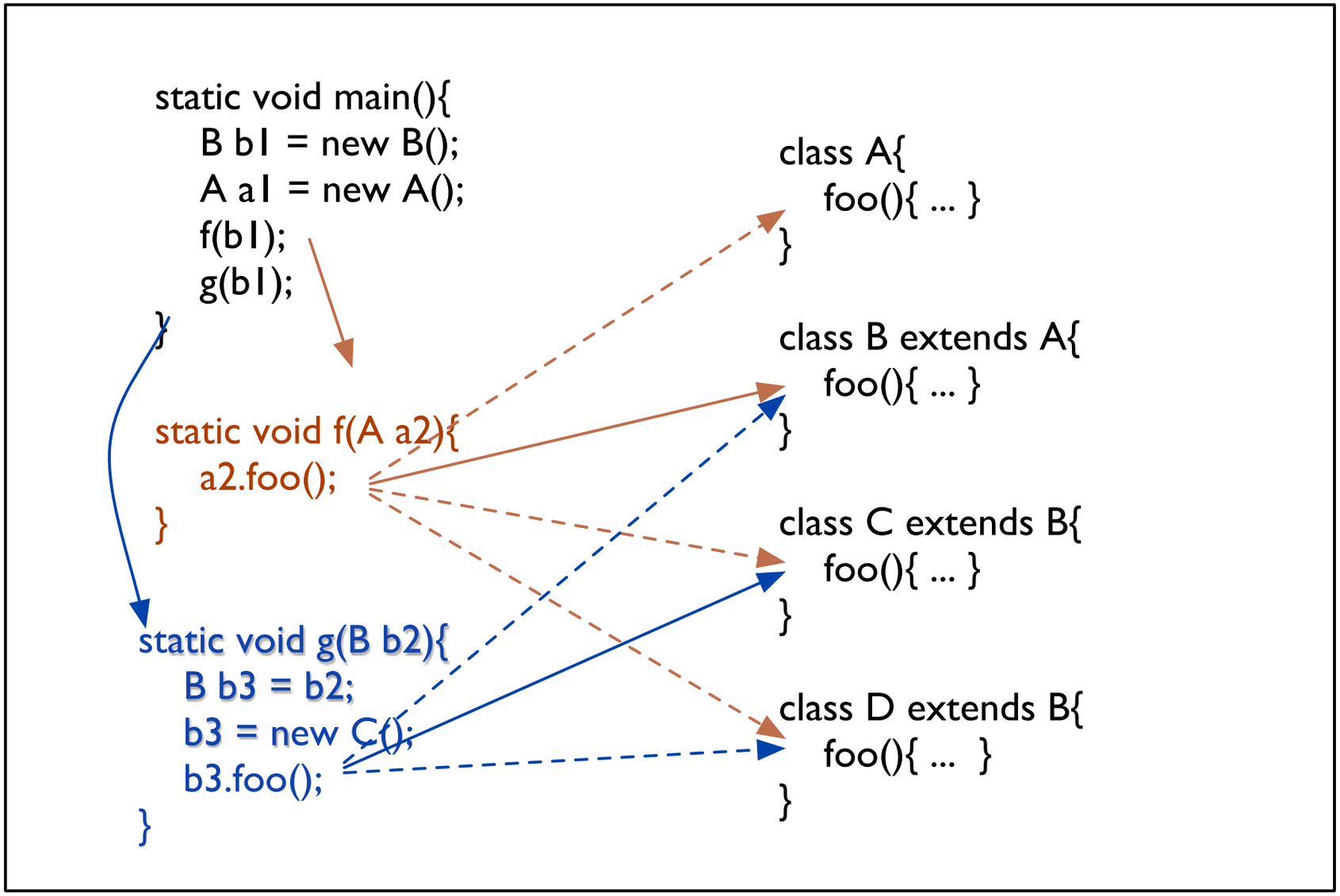
```
static void main(){
    B b1 = new B();
    A a1 = new A();
    f(b1);
    g(b1);
}

static void f(A a2){
    a2.foo();
}

static void g(B b2){
    B b3 = b2;
    b3 = new C();
    b3.foo();
}
```

```
class A{
    foo(){ ... }
}

class B extends A{
    foo(){ ... }
}

class C extends B{
    foo(){ ... }
}

class D extends B{
    foo(){ ... }
}
```

RO PAS

CHA

cf Barbara G. Ryder, ETAPS 03



J. Dean, D. Grove, C. Chambers,
"Optimization of OO Programs using Class Hierarchy", ECOOP 95

RO
PAS

CHA

Class Hierarchy Analysis

- calculate call graph using static type
- seems stupid
- resolve average 10-50% of Virtual Call
- recommend : working with other analysis
- O(n)

J. Dean, D. Grove, C. Chambers,
"Optimization of OO Programs using Class Hierarchy", ECOOP 95

RO
PAS

CHA

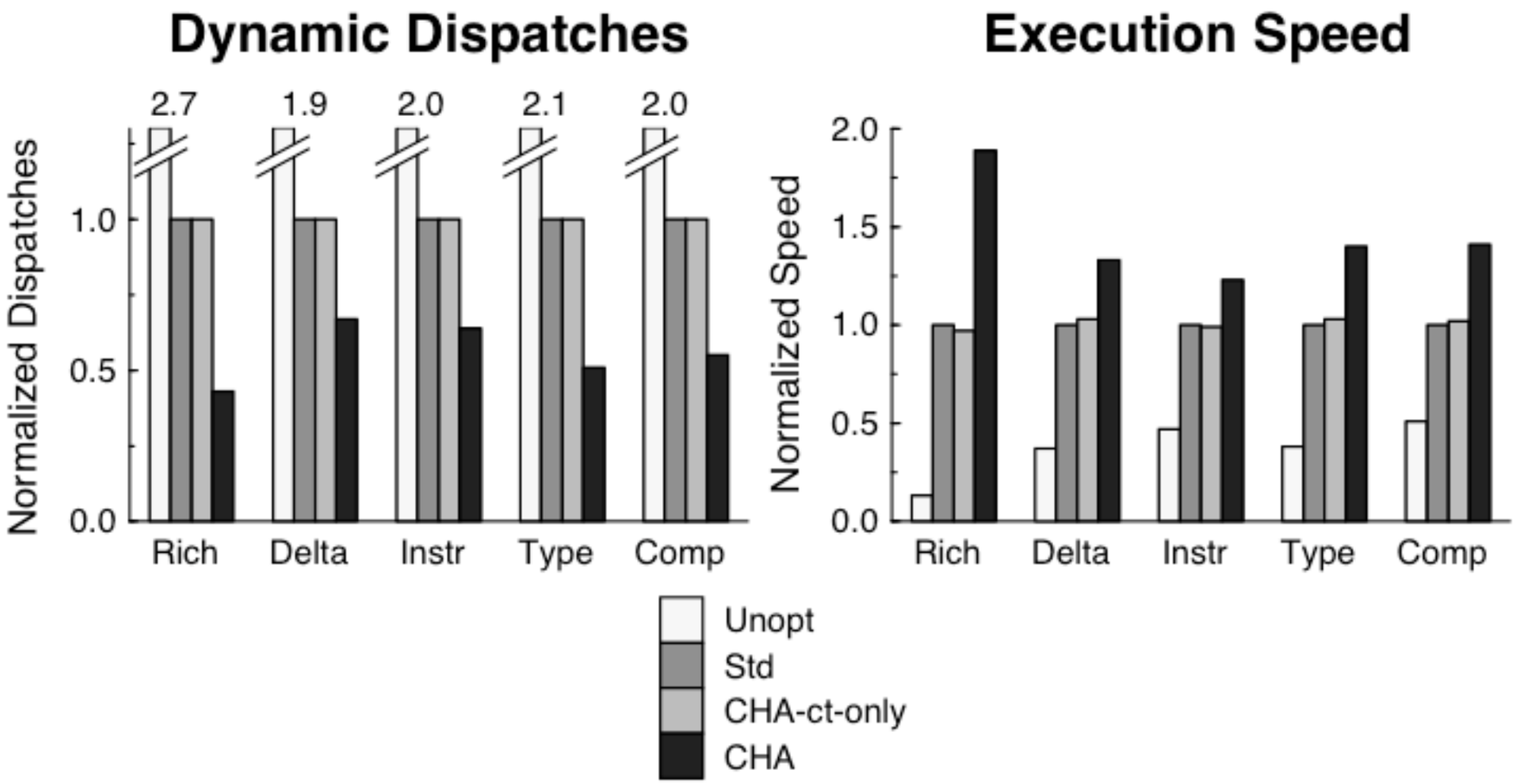


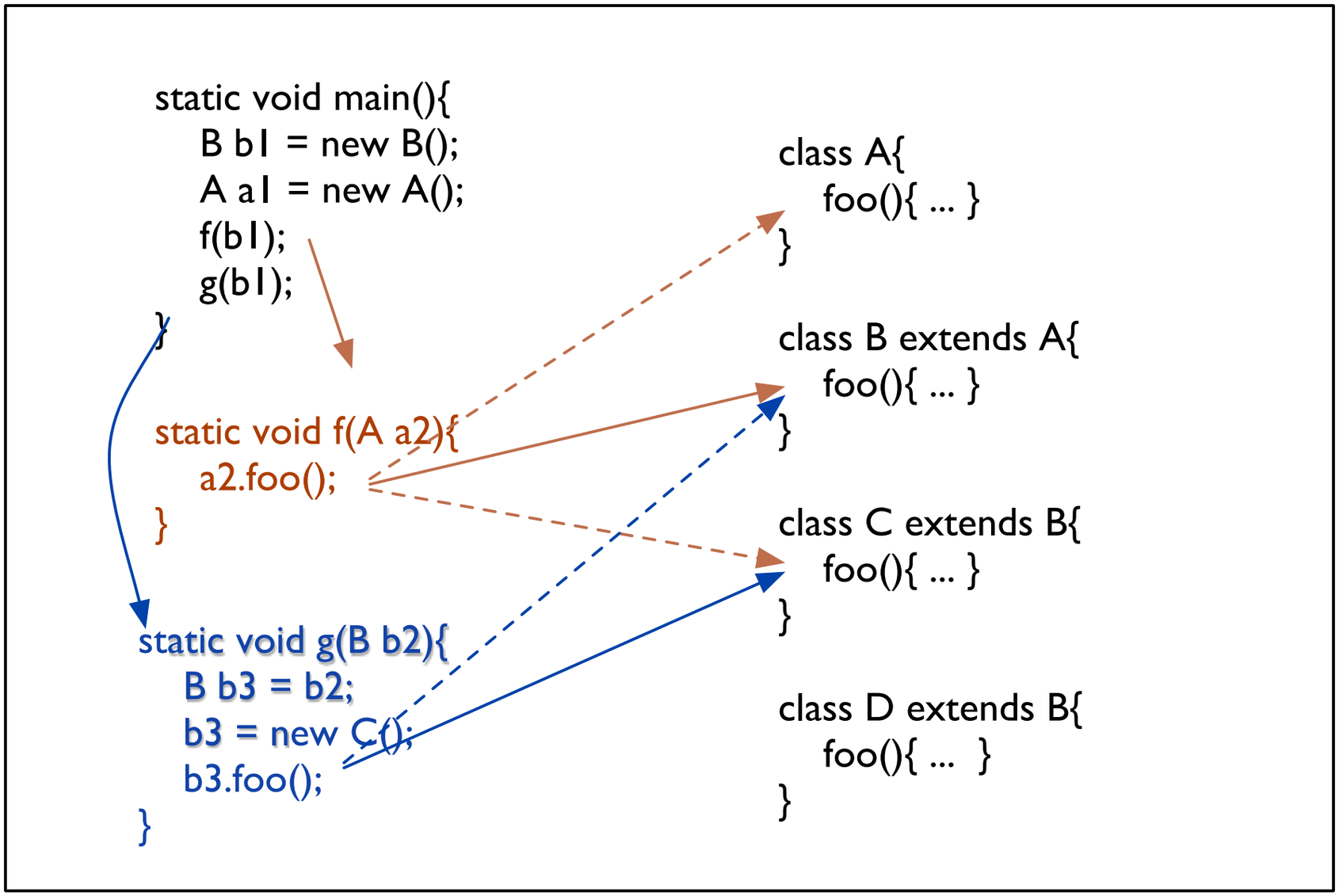
Figure 1: Number of dynamic dispatches and execution speed

chart from [Dean'95]

RO PAS

RTA

cf Barbara G. Ryder, ETAPS 03



D. Bacon and P. Sweeny,
"Fast Static Analysis of C++ Virtual Function Calls", OOPSLA'96

ROPAS

RTA

Rapid Type Analysis

- calculate instantiated class set :“I”
- calculate call graph using “I” and CHA
- resolve 10-90% of virtual call
- O(n)

D. Bacon and P. Sweeny,
“Fast Static Analysis of C++ Virtual Function Calls”, OOPSLA’96

RO
PAS

RTA

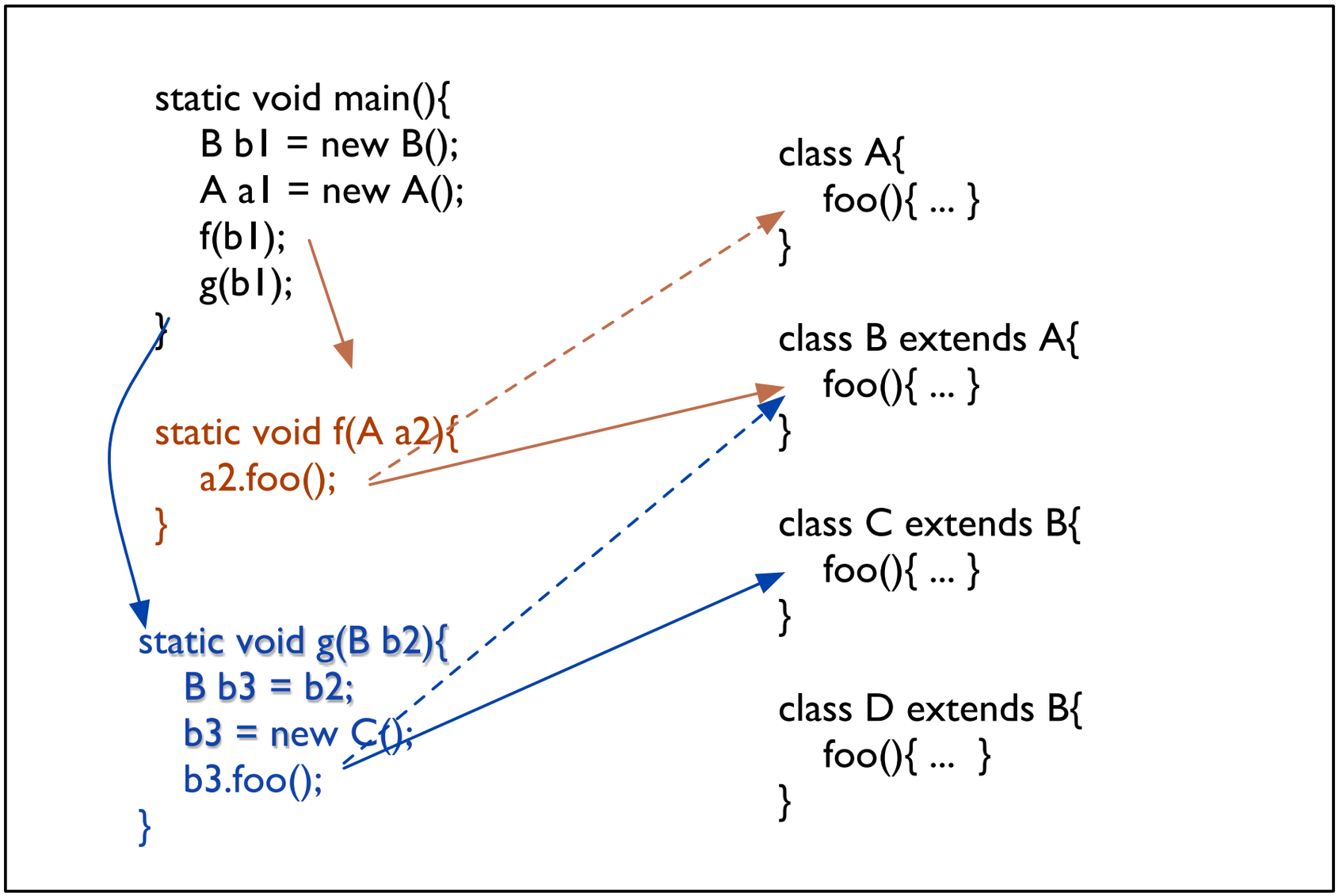
		Whole Application					
		Nodes Removed	*(%tot.)*	Edges Removed	*(%tot.)*	Callsites Resolved *(%poly)*	*(%tot.)*
`raytrace`	pRTA	808	(46%)	3585	(39%)	292 (77%)	(4.2%)
	oRTA	884	(51%)	4128	(45%)	300 (79%)	(4.3%)
	DTA	925	(53%)	4375	(47%)	304 (80%)	(4.4%)
	VTA	1026	(59%)	5200	(56%)	342 (90%)	(4.9%)
	oRTA+VTA	1031	(59%)	5242	(57%)	342 (90%)	(4.9%)
	VTA+VTA	1026	(59%)	5200	(56%)	342 (90%)	(4.9%)
`compress`	pRTA	814	(51%)	3664	(45%)	293 (79%)	(5.0%)
	oRTA	890	(56%)	4207	(52%)	301 (81%)	(5.2%)
	DTA	926	(58%)	4418	(55%)	303 (82%)	(5.2%)
	VTA	1033	(65%)	5214	(65%)	344 (93%)	(5.9%)
	oRTA+VTA	1039	(65%)	5256	(65%)	346 (93%)	(5.9%)
	VTA+VTA	1033	(65%)	5214	(65%)	344 (93%)	(5.9%)
`jack`	pRTA	820	(44%)	3763	(34%)	313 (40%)	(3.9%)
	oRTA	896	(48%)	4306	(39%)	321 (41%)	(4.0%)
	DTA	924	(50%)	4475	(41%)	323 (41%)	(4.1%)
	VTA	1027	(55%)	5719	(52%)	734 (94%)	(9.2%)
	oRTA+VTA	1033	(55%)	5769	(53%)	735 (94%)	(9.2%)
	VTA+VTA	1027	(55%)	5719	(52%)	734 (94%)	(9.2%)

table from [Sundaresan'00]

RO PAS

XTA

cf Barbara G. Ryder, ETAPS 03



Tip and Palsberg, "Scalable Propagation-based Call Graph Construction Algorithms", OOPSLA'00

RO PAS

XTA

- natural extension of RTA
- one class set per method
- one class set per field
- propagate set through function call and return
- small improvement
- O(n^3)

Tip and Palsberg, "Scalable Propagation-based Call Graph Construction Algorithms", OOPSLA'00

RO
PAS

XTA

benchmark	RTA	MTA	FTA	XTA	(RTA-MTA)/RTA	(RTA-FTA)/RTA	(RTA-XTA)/RTA
Hanoi	400	386	382	379	3.5%	4.5%	5.3%
Ice Browser	1,594	1,594	1,593	1,588	0.0%	0.1%	0.4%
mBird	8,061	8,036	7,772	7,760	0.3%	3.6%	3.7%
Cindy	15,457	14,729	11,331	10,967	4.7%	26.7%	29.0%
CindyApplet	5,223	4,990	4,399	4,347	4.5%	15.8%	16.8%
eSuite Sheet	7,171	7,149	7,093	7,071	0.3%	1.1%	1.4%
eSuite Chart	14,669	14,648	13,857	13,771	0.1%	5.5%	6.1%
javaFig 1.43	5,128	5,064	4,963	4,961	1.2%	3.2%	3.3%
BLOAT	19,384	18,772	16,704	16,672	3.2%	13.8%	14.0%
JAX 6.3	7,053	7,018	6,904	6,895	0.5%	2.1%	2.2%
javac	13,154	13,154	13,115	13,113	0.0%	0.3%	0.3%
Res. System	46,130	45,944	44,792	44,412	0.4%	2.9%	3.7%
AVERAGE					1.6%	6.6%	7.2%

Number of edges in the call graph(RTA/XTA)

table from [Tip'00]

RO
PAS

Anderson style

cf Barbara G. Ryder, ETAPS 03

```
static void main(){
    B b1 = new B();
    A a1 = new A();
    f(b1);
    g(b1);
}

static void f(A a2){
    a2.foo();
}

static void g(B b2){
    B b3 = b2;
    b3 = new C();
    b3.foo();
}
```

```
class A{
    foo(){ ... }
}

class B extends A{
    foo(){ ... }
}

class C extends B{
    foo(){ ... }
}

class D extends B{
    foo(){ ... }
}
```

L. Andersen, "Program Analysis and Specialization of the C Programming Language", PhD. thesis

Rountev, A Milanova, B. Ryder, "Points-to Analysis for Java Using Annotated Constraints", OOPSLA'01

RO PAS

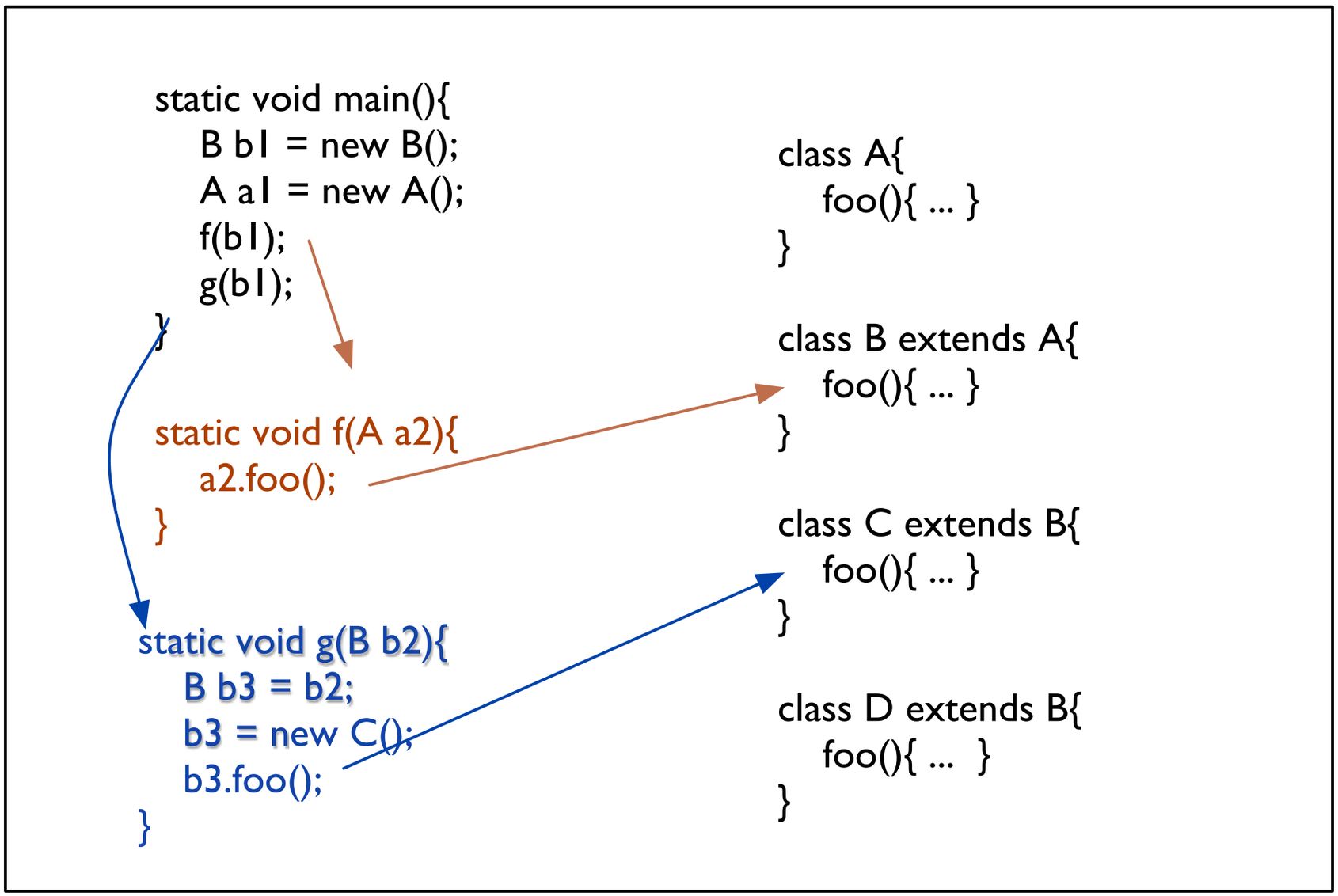
Anderson Style

- natural extension of XTA
- one class set per variable
- one class set per field
- distinguish object by allocation site
- O(n^3)

RO
PAS

0-CFA

cf Barbara G. Ryder, ETAPS 03



Shivers, PLDI'88

Chambers , POPL'98

RO PAS

0-CFA

- flow-sensitive
- most precise in context-insensitive family
- (variable#) class set per program point
- O(n^3)
- expensive!

RO
PAS

What to use?

- [Tip00]
  - Virtual Call Resolution : RTA

- [Sundaresan00][Lhotak06]
  - Call Graph Construction : Andersen Style

RO
PAS

Historical Overview

Part 2
context-sensitive family

k-Context / Object-Sensitive

RO
PAS

Example #2

cf Barbara G. Ryder, ETAPS 03

```
static void main(){
    D d1 = new D();
    if (...)
        (d1.f(new B())).g();
    else
        (d1.f(new C())).g();
}

public class D{
    public A f (A a1){ return a1; }
}
```

```
public class A{
    X xx;
    A (X xa){ this.xx = xa;}
}

publix class B{
    B(X xb){C3 : super(xb);}
    publix X f() { return this.xx;}
    static void main(){
        X x1, x2;
        B b1 = new B(new Y());
        B b2 = new B(new Z());
        x1 = b1.f();
        x1.g();
        x2.b2.f();
        x2.g();
}
```

context-insensetive family cannot handle

RO PAS

k-call-site-sensitive

cf Barbara G. Ryder, ETAPS 03

```
static void main(){
    D d1 = new D();
    if (...)
        (d1.f(new B())).g();
    else
        (d1.f(new C())).g();
}

public class D{
    public A f (A a1){ return a1; }
}
```

```
public class A{
    X xx;
    A (X xa){ this.xx = xa;}
}

publix class B{
    B(X xb){C3 : super(xb);}
    publix X f() { return this.xx;}
    static void main(){
        X x1, x2;
        B b1 = new B(new Y());
        B b2 = new B(new Z());
        x1 = b1.f();
        x1.g();
        x2.b2.f();
        x2.g();
}
```

John Plevyak, Andrew A. Chien, "Precise Concrete Type Inference for Object-Oriented Languages", OOPSLA'94

John Whaley, Monica S. Lam, "Cloning-Based Context-Sensitive Pointer Alias Analysis Using Binary Decision Diagrams", PLDI'04

RO PAS

k-call-site-sensitive

- unlimited call-site-string
- collapse SCC to one big function node
- flow-insensitive analysis

- using BDD to control context blow up
- implement Datalog to BDD compiler

RO
PAS

k-call-site-sensitive

Benchmark	insens.	object-sensitive 1	2	3	1H	call site 1	2	1H	ZCWL
compress	2597	8.4	9.9	11.3	12.1	2.4	3.9	4.9	3.3
db	2614	8.5	9.9	11.4	12.1	2.4	3.9	5.0	3.3
jack	2870	8.6	10.2	11.6	11.9	2.4	3.9	5.0	3.4
javac	3781	10.4	17.7	33.8	14.3	2.7	5.3	5.4	
jess	3217	8.9	10.6	12.0	13.9	2.6	4.2	5.0	3.9
mpegaudio	2794	8.1	9.4	10.8	11.5	2.4	3.8	4.8	3.3
mtrt	2739	8.3	9.7	11.1	11.8	2.5	4.0	4.9	3.4
soot-c	4838	7.1	13.7	18.4	9.8	2.6	4.2	4.8	
sablecc-j	5609	6.9	8.4	9.6	9.5	2.3	3.6	3.9	
polyglot	5617	7.9	9.4	10.8	10.2	2.4	3.7	4.7	3.3
antlr	3898	9.4	12.1	13.8	13.2	2.5	4.1	5.2	4.3
bloat	5238	10.2	44.6		12.9	2.8	4.9	5.2	6.7
chart	7070	10.0	17.4		18.2	2.7	4.8		
jython	4402	9.9	55.9		15.6	2.5	4.3	4.6	4.0
pmd	7220	7.6	14.6	17.0	11.0	2.4	4.2	4.2	
ps	3875	8.7	9.9	11.0	12.0	2.6	4.0	5.2	4.4

Note: columns after the second column show multiples of the context-insensitive number.

Table 3. Number of equivalence classes of abstract contexts

graph from [Lhotak06]

RO PAS

k-call-site-sensitive

- [Lhotak,06]
  - evan less precise than 2-call-site-sensitive
  - because of collapsed SCC

Object-Sensitive

cf Barbara G. Ryder, ETAPS 03

```
static void main(){
    D d1 = new D();
    if (...)
        (d1.f(new B())).g();
    else
        (d1.f(new C())).g();
}

public class D{
    public A f (A a1){ return a1; }
}
```

```
public class A{
    X xx;
    A (X xa){ this.xx = xa;}
}

publix class B{
    B(X xb){C3 : super(xb);}
    publix X f() { return this.xx;}
    static void main(){
        X x1, x2;
        B b1 = new B(new Y());
        B b2 = new B(new Z());
        x1 = b1.f();
        x1.g();
        x2.b2.f();
        x2.g();
}
```

Ana Milanova, Barbara G. Ryder, "Annotated Inclusion Constraints for Precise Flow Analysis", ICSM'03

Ana Milanova, "Light Context-Sensitive Points-to Anlaysis for Java", PASTE'07

RO PAS

Object-Sensitive

- use receiver argument as context
- kind of bounded CPA [Agesen'95]
- about 2 time slower than Andersen Style
- more precise than k-call-site-sensitive

RO
PAS

Comparison

[Lothak'06]

1. Object-Sensitivity + Heap Cloning
2. Object-Sensitivity
3. Call-Site + Heap Cloning
4. Call-Site
5. k-call-site
6. Adersen Style (context-insensitive)

Observe

[Lothak'06]

Object-Sensitive + Heap Cloning is best

Anderson Style : not very bad

Unrealizable Path caused by bounded-context spoils precision

RO
PAS

Modern Movement

- *goal : defeat Unrealizable Path*

- Equivalent Context [Xu08]
- Refinement-Based [Sridhan06]

RO
PAS

Algorithm Overview

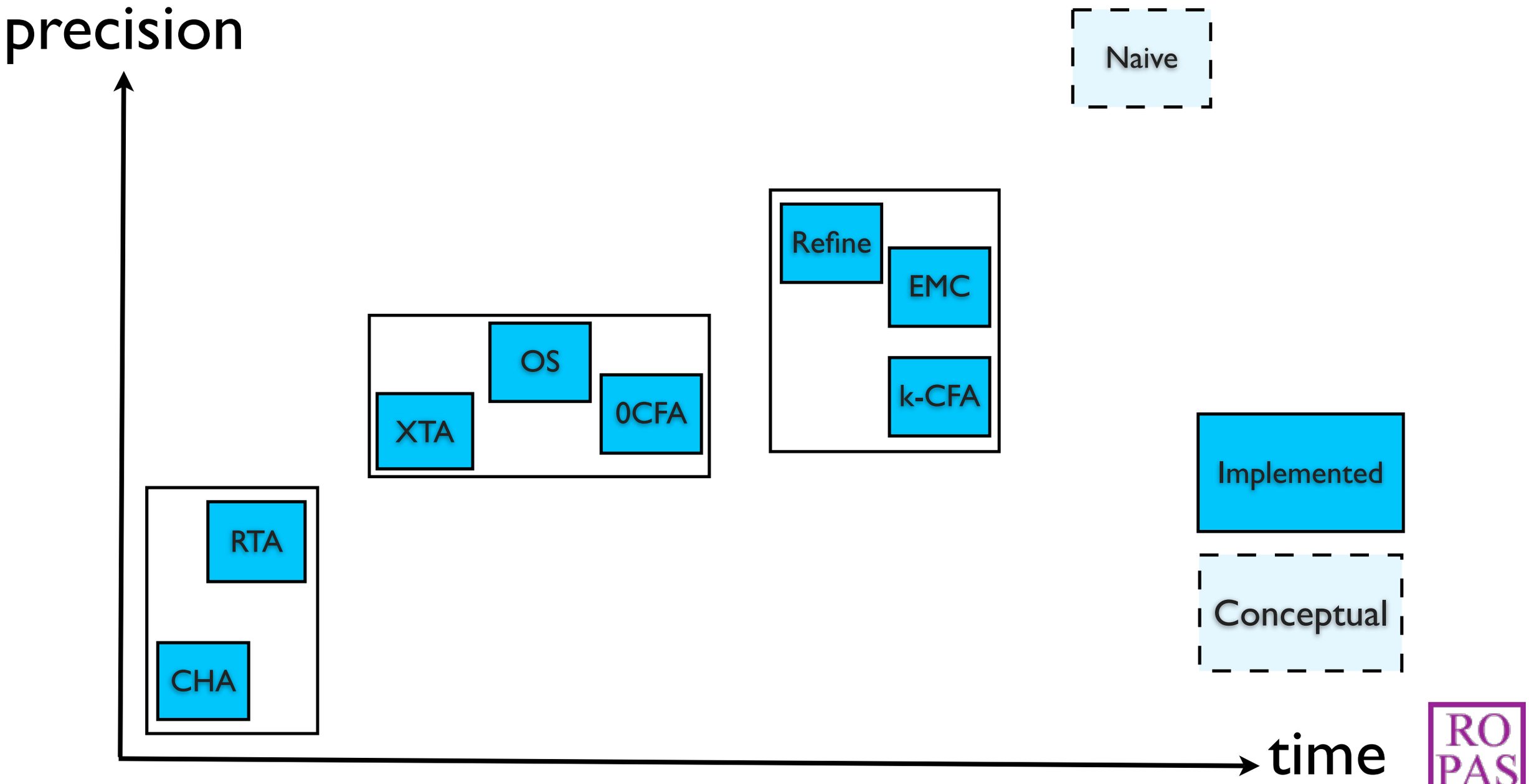
RO
PAS

Contents

- CHA/RTA/XTA (정가을)
- 0-CFA (고윤석)
- Refinement-Based (공순호)
- Equivalent-Context (오학주)

Conclusion

Categorize


precision
Naive
Refine
EMC
k-CFA
OS
XTA
0CFA
RTA
CHA
Implemented
Conceptual
time
RO
PAS

What to use?

- Virtual Function Resolution :

  RTA + Anderson Style

- Call Graph Construction : Object-Sensitive

- Pointer Characteristic Analysis :

  Refinement-Based

RO
PAS

Other Issues

- Demand-Driven?
- Unification-Based Analysis?

RO
PAS

Thank you

Reference

- [Agesn95] O. Agsen. "The cartesian product algorithm : Simple and precise type inference of parameteric polymorphis", In European Conference on Object Oriented Programming, pages 2-26, 1995

- L. O. Andersen, "Program analysis and specialization for the C programming language". Ph.D thesis, DIKU, University of Copenhagen, 1994

- D.Bacon and P. Sweeney, "Fast Static Analysis of C++ Virtual Function Calls", OOPLSA'96

- [Dean95] J.Dean, D.Grove, C.Chambers, "Optimization of OO Programs Using Static Class Hierarchy", ECOOP'95

- D.Grove and C.Chambers. "A Framework for call graph construction algorithms." ACM Transactions on Programming Languages and Syatems (TOPLAS), 23(6), 2001.

RO
PAS

References

- A. Milanova, A Rountev, and B.G. Ryder. "Parameterized object-sensitivity for points-to and side-effect analysis for java." In International Symposium on Software Testing and Analysis, 2002.

- A. Milanova, Barbara G. Ryder. "Annotated Inclusion Constraints for Precise Flow Analysis", Proceedings of the 21st IEEE International, 2005

- A. Milanova. "Light Context-Sensitive Points-to Analysis for Java", PASTE'07

- J. Plevyak and A.Chien. "Precise concrete type inference for object-oriented languages." OOSPLA'94

- [Lhotak07] Ondřej Lhoták and Laurie Hendren, "Context-Sensitive Points-to Analysis: Is It Worth It?", LNCS, Springer, 2007

- [Sundaresan00] Vijay Sundaresn, Laurie Hendren, Chrislain Ranafimahefa, Reja Vallee-Rai, Ptrick Lam, Etenne Gagnon and Charles Godin. "Practical Virtual Method Call Resolution for Java", Conference on Object Oriented Programming System Languages and Applications, Proceedings of the 15th ACM SIGPLAN conference on Object-orientd programming, systems, languages, and applications, 2000

RO
PAS

References

- [Sridharan06] M.Sridharan, R.Bodik, "Refinement-Based Context-Sensitive Analysis for Java", PLDI'06

- M.Sharir and A.Pnueli. "Two approaches to interprocedural data-flow analysis", In S.Muchnick and N.Jones, editors, Program Flow Analysis : Theory and Applications, Prentice Hall, 1981

- Barbara G.Ryder. "Dimension in Reference Analysis of Object-Oriented Programming Languages", LNCS Springer, 2003

- [Tip00] F. Tip and J.Palsberg. "Scalable Propagation-based Call Graph Construcion Algorithm", OOPSLA'00

- J.Whaley and M.Lam. "Cloning-Based Context-Sensitive Pointer Analysis Using Binary Decision Diagrams", PLDI'04

- [Xu08] Guoqing Xu, Atanas Rountev. "Merging Equivalent Contexts for Scalable Heap-Cloning-Based Context-SEnsitive Points-to Anlyasis", ISSTA'08

RO
PAS