Reference Analysis - 3

- Parameterized object-sensitive analysis
  - Milanova and Ryder, ICSM'05
- Comparisons of object-sensitive analysis with 1-CFA
  - Two papers differ in their conclusions
    - Liang et.al Paste'05
    - Lhotak et.al CC'06

Parameterized CS Analysis ICSM'05

- Presents a parameterized framework for context-sensitive analysis
  - E.G., Context-sensitive object naming by allocation site with context or by using abstract class objects for some allocation sites
  - Describes different sorts of context sensitivity reflected in constraint annotations

Parameterized CS Analysis ICSM'05

- **Empirical investigations**
  - **Compares 2 kinds of object-sensitive analyses on side-effects and downcast safety problems**
  - **FullObjSens: uses object-sensitive object naming**
  - **ObjSens:**
    - **if #alloc sites for class A> 50, then use abstract Aobj as object representation, rather than alloc site**
    - **Only analyze calls through *this* context-sensitively**
  - **Results:**
    - **Running times and memory usage is comparable**
    - **Side-effect analysis determines set of objects modified by each statement**
    - **Downcast safety analysis determines which downcasts are provably safe**

Milanova & Ryder, ICSM'05

Program	(1) Program Size		(2) Analysis Cost				(3) Side-effect Analysis				(4) Downcast Safety	
	Classes	Methods	*FieldSens*		*ObjSens*		*FieldSens*		*ObjSens*		*FieldSens*	*ObjSens*
			Time	Mem	Time	Mem	1-3	≥10	1-3	≥10		
proxy	565	3283	4.8	35.1	5.3	34.8	19%	75%	76%	10%	24%	67%
compress	568	3316	8.3	39.6	10.1	40.1	23%	73%	68%	23%	24%	71%
db	565	3339	9.2	40.6	10.6	42.5	20%	76%	66%	25%	24%	74%
jb	574	3393	6.0	36.7	5.8	36.9	16%	80%	73%	12%	12%	44%
echo	577	3544	18.7	49.2	44.9	66.2	24%	69%	63%	26%	18%	43%
raytrace	582	3451	7.8	42.2	10.8	46.1	23%	72%	67%	24%	23%	71%
mtrt	582	3451	9.4	42.1	11.3	46.2	23%	72%	67%	24%	23%	71%
jtar	618	3583	16.8	50.3	24.4	58.9	19%	74%	62%	25%	17%	44%
jlex	578	3381	6.7	39.8	7.3	40.6	18%	79%	57%	10%	22%	78%
javacup	581	3564	23.2	55.8	21.2	58.5	14%	83%	54%	9%	9%	85%
rabbit	615	3770	9.1	46.2	11.7	45.6	20%	76%	48%	17%	23%	68%
jack	613	3573	28.7	54.8	24.9	56.7	17%	80%	54%	38%	15%	63%
jflex	608	3692	28.5	63.5	30.3	66.4	18%	78%	64%	13%	4%	62%
jess	715	3973	35.8	59.4	87.5	61.0	16%	79%	63%	29%	20%	73%
mpegaudio	608	3531	11.6	44.0	10.4	48.4	23%	78%	67%	24%	23%	68%
jjtree	620	4078	8.6	46.8	32.1	64.4	8%	90%	32%	42%	65%	80%
sablecc	864	5151	34.5	78.5	51.2	75.3	20%	77%	67%	20%	33%	47%
javac	730	4470	100.5	110.0	168.5	129.0	14%	83%	38%	42%	12%	36%
creature	626	3881	64.3	94.3	105.5	124.8	19%	79%	55%	32%	18%	33%
mindterm	686	4420	37.2	78.5	51.5	90.5	20%	73%	57%	30%	25%	47%
soot	1214	5669	139.4	117.8	115.9	117.9	31%	73%	46%	40%	17%	25%
muffin	894	5253	120.7	133.9	115.1	149.7	16%	80%	45%	49%	13%	35%
javacc	615	4198	99.6	96.6	93.4	101.9	10%	89%	29%	22%	6%	59%
Average							18%	78%	57%	36%	20%	58%

Table 1. Java programs and analysis results.

Empirical Comparisons-Paste'05

D. Liang, M. Pennings, MJ Harrold, "Evaluating the Impact of Context Sensitivity on Andersen's Algorithm for Java Programs", PASTE'05

- Object-sensitive vs 1-CFA with Andersen
- Good comparison examples given of the abstraction choices in the algorithms
- Context-sensitive object naming
  - Uses k-level of calls (context-bounded) or k receiver names (name-bounded) to differentiate object creation sites in different calling contexts
- Details
  - Builds models for collections and maps
  - User-supplied info about reflection
  - Compares static solutions to dynamically observed points-to's

Empirical Comparisons Paste'05

- Experiments on 12 Java programs with 8-310 classes and 89-2025 methods
- Three studies
  - Relative precision of static results to dynamically measured object receivers at callsites
    - Showed that context-sensitive approaches can achieve significant precision over context-insensitive ones
  - Use of context-sensitive naming schemes can lead to significant gains in precision
    - Using context-insensitive object naming with context-sensitive (call-site) Andersen degrades precision

Empirical Comparisons Paste'05

- Precision per allocation site
  - Aggregating precision information for all objects at the same allocation site
    - Again gains showed for context-sensitive analysis
- Conclusions
  - "Crucial to distinguish the instances allocated at an allocation site under different contexts in a context-sensitive analysis" (section 4.4)

Empirical Comparisons CC'06

- Reports on a comparison of 4 different context-sensitive analyses
  - Context-insensitive points-to
  - Call-site-string-based points-to
  - Receiver-object-based points-to
  - Cloning-based points-to (ZCWL, PLDI'04)
  - Run on same 16 benchmarks
  - Implemented on the same framework (JEDD in Soot)
  - Combined with context-sensitive object naming schemes
  - Effectiveness measured on devirtualization, redundant cast removal, call graph size
- *Bottom line:* object-sensitive analysis shown to be superior, in terms of scalability and precision

"Context-sensitive analysis - Is it worth it?", O. Lhotak, L. Hendren, CC'06

Context-sensitive Points-to Algorithms in Study CC'06

- Informal algorithm is flow- and context-insensitive
- Call-site-string-based uses a string of the *k* most recent actual call sites on the runtime stack as the 'calling context'
- Receiver object-based (object-sensitive) uses the sequence of the *k* most recent receiver objects as the 'calling context'
- Cloning-based (with BDDs) actually makes one copy per method instantiation
  - Corresponding to call edges that DO NOT participate in a cycle in the context-insensitive call graph (ZCWL, PLDI'04)

"Context-sensitive analysis - Is it worth it?", O. Lhotak, L. Hendren, CC'06

Questions to answer

1. Which contexts are actually useful to improve analysis precision?
   - How often contexts have identical points-to info?
   - How much context can be saved for practical cost?
   - Does more context help precision?
2. Why can BBDs do so well in representing large numbers of contexts?
   - How poorly would non-BDD representations do for context-sensitive analyses?
3. How well do the algorithms do on client problems?
   - Call graph construction, devirtualization, unnecessary cast elimination

"Context-sensitive analysis - Is it worth it?", O. Lhotak, L. Hendren, CC'06

Findings - #Contexts

Benchmark	insens.	object-sensitive 1	2	3	1H	call site 1	2	1H	ZCWL
compress	2596	13.7	113	1517	13.4	6.5	237	6.5	2.9×10^4
db	2613	13.7	115	1555	13.4	6.5	236	6.5	7.9×10^4
jack	2869	13.8	156	1872	13.2	6.8	220	6.8	2.7×10^7
javac	3780	15.8	297	13289	15.6	8.4	244	8.4	
jess	3216	19.0	305	5394	18.6	6.7	207	6.7	6.1×10^6
mpegaudio	2793	13.0	107	1419	12.7	6.3	221	6.3	4.4×10^5
mtrt	2738	13.3	108	1447	13.1	6.6	226	6.6	1.2×10^5
soot-c	4837	11.1	168	4010	10.9	8.2	198	8.2	
sablecc-j	5608	10.8	116	1792	10.5	5.5	126	5.5	
polyglot	5616	11.7	149	2011	11.2	7.1	144	7.1	10130
antlr	3897	15.0	309	8110	14.7	9.6	191	9.6	4.8×10^9
bloat	5237	14.3	291		14.0	8.9	159	8.9	3.0×10^8
chart	7069	22.3	500		21.9	7.0	335		
jython	4401	18.8	384		18.3	6.7	162	6.7	2.1×10^{15}
pmd	7219	13.4	283	5607	12.9	6.6	239	6.6	
ps	3874	13.3	271	24967	13.1	9.0	224	9.0	2.0×10^8

Table II: Total number of abstract contexts

"Context-sensitive analysis - Is it worth it?", O. Lhotak, L. Hendren, CC'06

Findings - #Equiv Contexts

- **Given <m1,c1> and <m1,c2>, if every local reference has same points-to set in these 2 contexts, they are *equivalent***
- **Found many equivalent abstract contexts in the data**
- **In general, there are more equiv classes of contexts with ObjSens than with CallSite abstractions**
  - **Expect better precision from this**
- **In both ObjSens and CallSite, increasing k increases the #equiv classes only slightly while increasing the absolute #contexts significantly (little precision improvement for a large cost)**
- **#contexts of ZCWL is very small because of the merges on the large SCCs in the benchmark initial call graphs; effectively ZCWL models much of the call graph context-insensitively**

"Context-sensitive analysis - Is it worth it?", O. Lhotak, L. Hendren, CC'06

Findings - #Equiv Contexts

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

Table III: Number of equivalence classes of abstract contexts

"Context-sensitive analysis - Is it worth it?", O. Lhotak, L. Hendren, CC'06

#Distinct Points-to Sets

- **Found fairly equivalent numbers of distinct points-to sets across all algorithms with all levels of context.**
- **Means the problem for a non-BDD solution procedure for context-sensitive analysis is not points-to set size, but rather how to efficiently store contexts.**

"Context-sensitive analysis - Is it worth it?", O. Lhotak, L. Hendren, CC'06

Call Graph Construction

- **Idea: construct context-sensitive call graphs, project away their contexts and then compare results (otherwise, cannot compare different context abstractions)**
  1. **Measure set of reachable methods from program entries**
  2. **Measure set of call site possible targets**
- **Results**
  - **Little difference in #1 between ObjSens and CallSite**
  - **Little difference in number of call graph edges eminating from application methods**
  - **Better devirtualization with ObjSens than with CallSite**

"Context-sensitive analysis - Is it worth it?", O. Lhotak, L. Hendren, CC'06

#Polymorphic Call Sites

		object-sensitive				call site		
Benchmark	insens.	1	2	3	1H	1	2	1H
compress	**3**	3	3	3	3	3	3	3
db	5	**4**	4	4	4	5	4	5
jack	25	23	23	23	**22**	24	23	24
javac	737	**720**	720	720	720	720	720	720
jess	**45**	45	45	45	45	45	45	45
mpegaudio	27	**24**	24	24	24	24	24	24
mtrt	9	**7**	7	7	7	8	8	8
soot-c	983	**913**	913	913	913	938	913	938
sablecc-j	450	325	325	325	**301**	380	325	380
polyglot	744	592	592	592	**585**	592	592	592
antlr	**843**	843	843	843	843	843	843	843
bloat	1079	962	962		**961**	962	962	962
chart	254	235	235		**214**	235	235	
jython	347	347	347		**346**	347	347	347
pmd	1224	1193	1193	1193	**1163**	1205	1205	1205
ps	304	303	303	303	**300**	303	303	303

Table VII: Total number of potentially polymorphic call sites in benchmark (non-library) code

"Context-sensitive analysis - Is it worth it?", O. Lhotak, L. Hendren, CC'06

Cast Safety

- **When can we use points-to analysis to eliminate unnecessary casts?**
  - **ObjSens is often significantly more precise than CallSite**
  - **Precision is further improved with context-sensitive object naming as shown in polyglot below**

Benchmark	insens.	object-sensitive 1	2	3	1H	call site 1	2	1H	ZCWL
soot-c	955	932	932	932	**878**	932	932	932	
sablecc-j	375	369	369	369	**331**	370	370	370	
polyglot	3539	3307	3306	3306	**1017**	3526	3443	3526	3318
antlr	295	275	275	275	**237**	276	275	276	276

Table VIII: Number of casts potentially failing at run time

"Context-sensitive analysis - Is it worth it?", O. Lhotak, L. Hendren, CC'06

Conclusions CC'06

- **Interesting empirical study of effectiveness of different context-sensitive algorithms**
- **Object-sensitive contexts seem more effective than call-site contexts in precision and scalability**
- **Context-sensitive object naming (with ObjSens contexts) help precision when containers and maps are involved**
- **Claim that a non-BDD implementation of 1-ObjSens analysis should be possible**
- **Claim that such an implementation with context-sensitive object naming would require new improvements in data structures and algorithms to be practical**