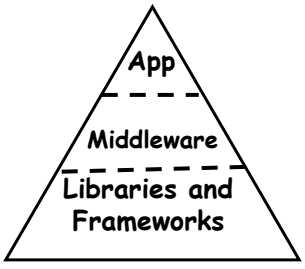
Blended Program Analysis

Barbara G. Ryder
Virginia Tech

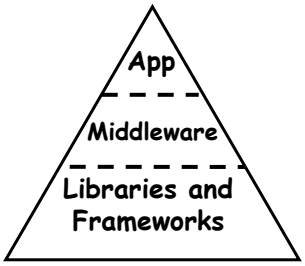
Collaborators: Bruno Dufour (Rutgers) & Gary Sevitsky (IBM Research); Funded by IBM Open Collaborative Research Program and NSF 08-0811518

Framework-based Applications



- Application is an iceberg
  - Bulk of the code in libraries and frameworks
  - Genre not commonly addressed by research community
  - E.g., financial planning services, e-commerce sites, online reservation systems, Tomcat-based systems software
- Programs are not just large, but are more complex in interactions between frameworks
- Performance problems span multiple layers

Framework-based Applications



- Software characteristics
  - Not amenable to static analyses
    - Not scalable -- too complex
  - Not amenable to dynamic analysis
    - Too intrusive to execution for production codes
  - Applications main function often is data transformation
- Goal: design analyses for performance diagnosis of these systems

Outline

- Motivation
- Blended analysis paradigm
- Blended escape analysis
  - Example
  - Explanations of performance problems
  - Newest empirical results
- Related work
- Summary and future work

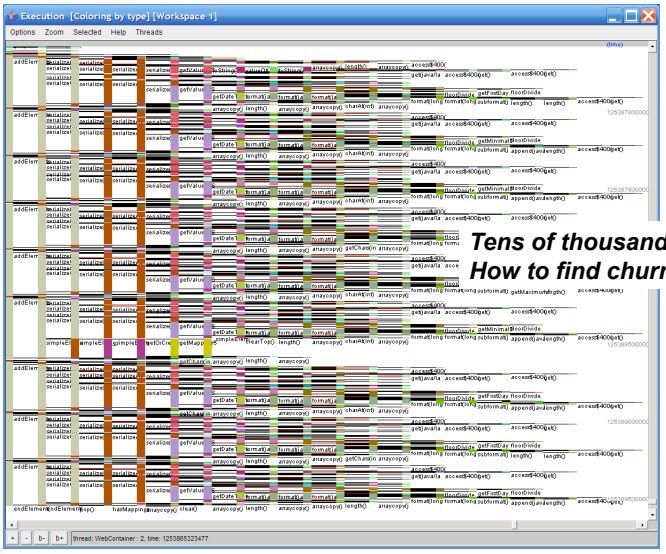
Initial Goals

- Devise new analyses to aid performance diagnosis
- Gather data about the characteristics of these important practical applications
  - To enable code specialization, better benchmark selection, establishment of API 'best practices'
- Design initial experiments to test ideas
  - Problem: overuse of temporaries or object churn
  - Q: can we identify object churn through analysis?

Eliminating Object Churn

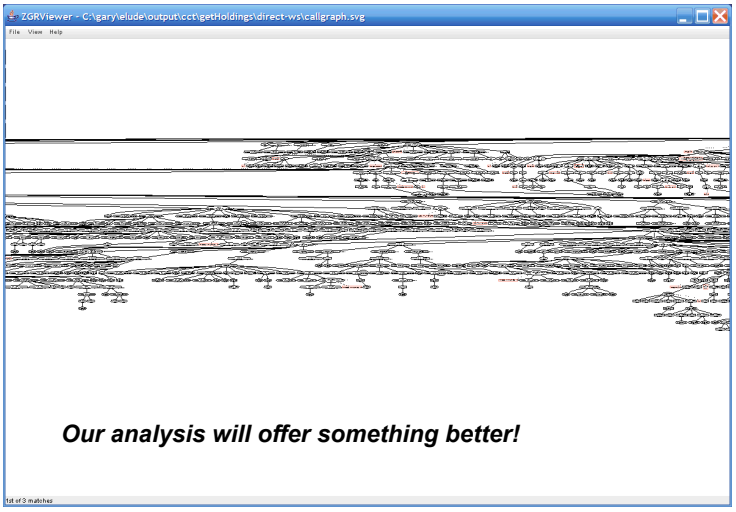
- Identify temporary objects
  - Need to approximate "object lifetime"
- Identify execution contexts with excessive use of temporaries
  - Based on total number of *instances*
  - Not same as finding often-executed allocation sites
- Elimination strategies
  - Optimize the use of frameworks and libraries together
  - Introduce caching for temporary data structures
  - Code specialization
- Can help understand construction of longer-lived data

Current Practice: Jinsight Trace of HoldingDataBean_Ser.serialize()



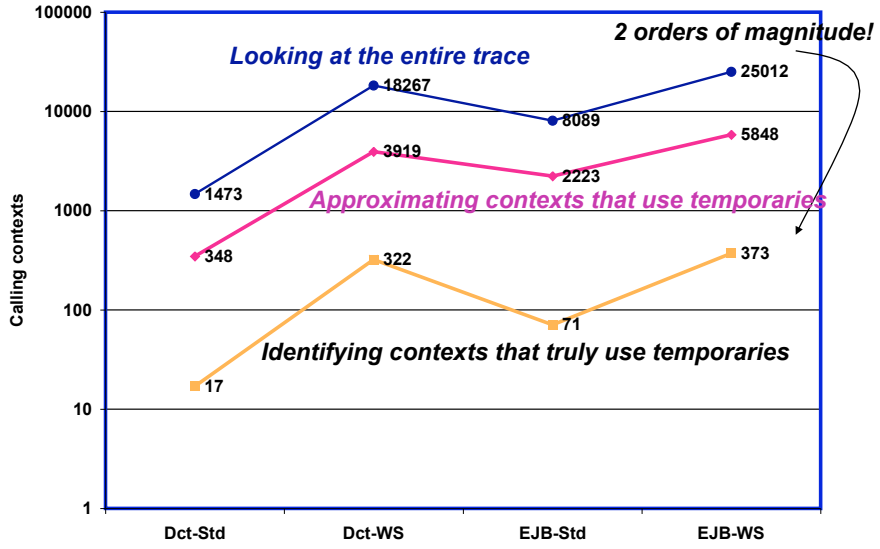
Tens of thousands of calls
How to find churn locality?

Optimized calling tree of trace from HoldingDataBean_Ser.serialize()



Our analysis will offer something better!

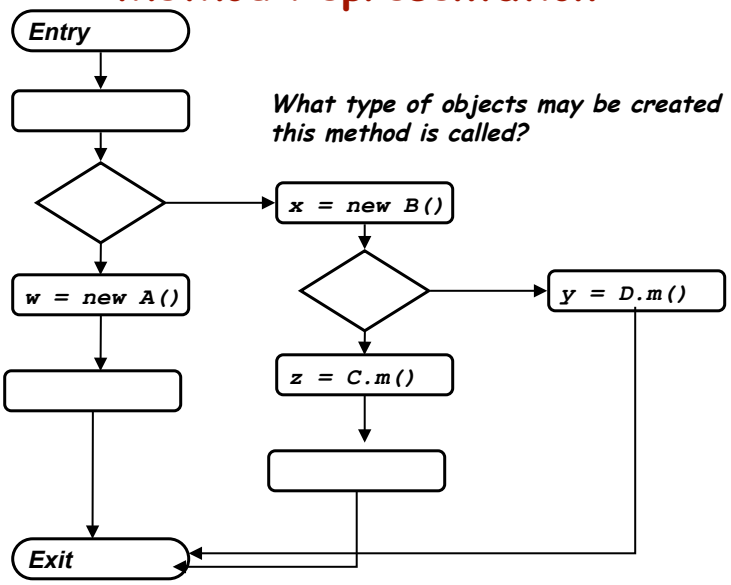
Blended Analysis - Scalability



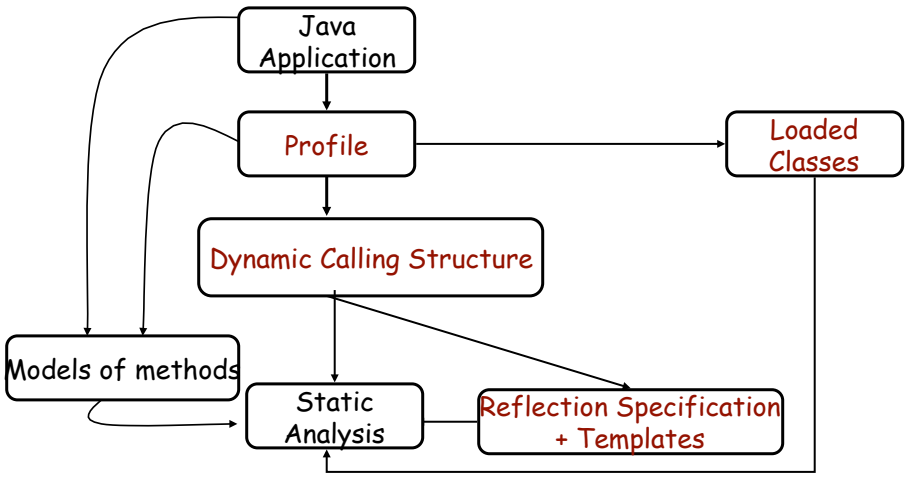
Outline

- Motivation
- Blended analysis paradigm
- Blended escape analysis
  - Example
  - Explanations of performance problems
  - Newest empirical results
- Related work
- Summary and future work

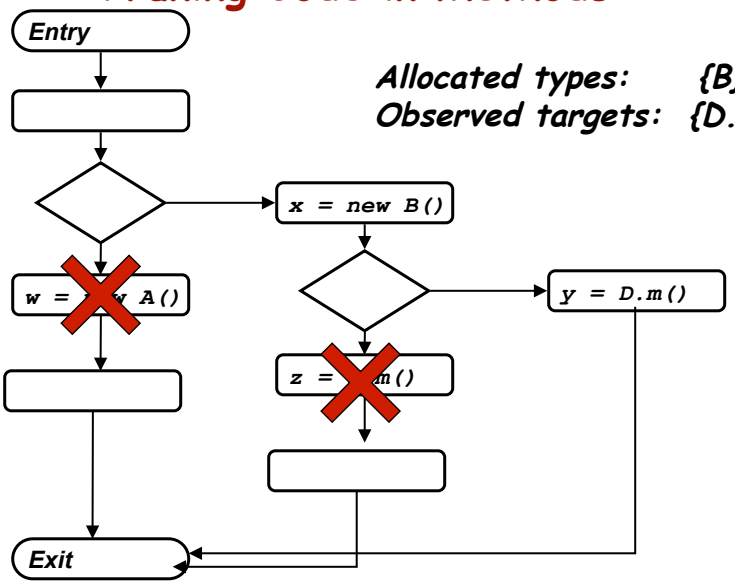
Method Representation



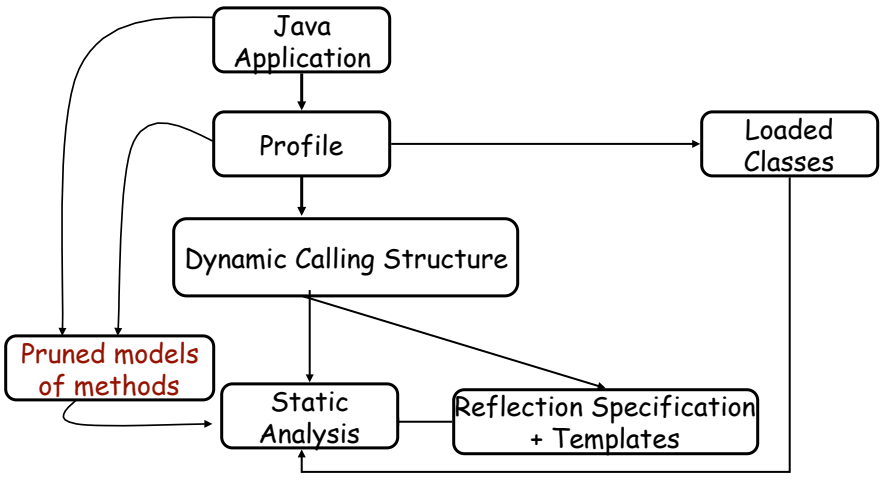
Blended Analysis Paradigm



Pruning Code in Methods



Blended Analysis Paradigm



Outline

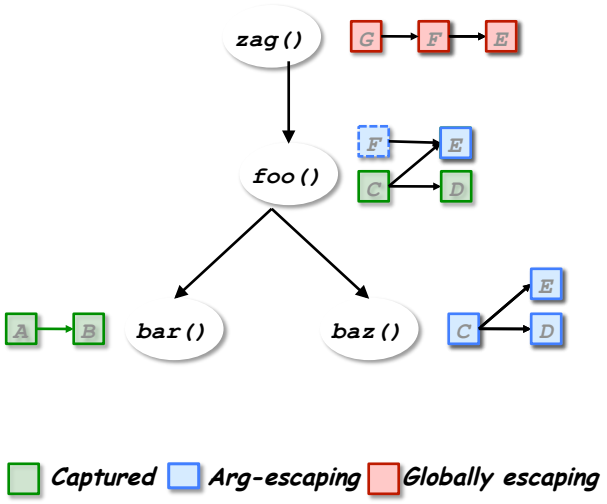
- Motivation
- Blended analysis paradigm
- Blended escape analysis
  - Example
  - Explanations of performance problems
  - Newest empirical results
- Related work
- Summary
- Future work

Escape Analysis

Choi et. al, TOPLAS'03

- Determines escape property of an object (i.e., an allocation site):
  - *Captured* (not escaping)
  - *Arg-escaping* (escaping through an argument)
  - *Globally escaping*
- Builds connection graph for each method
  - Shows points-to relations between object fields and references
  - Shows escape state of each object

Escape analysis




```
void bar() {
   a = new A();
   a.x = new B();
}
C baz() {
   c = new C();
   c.y = new D();
   c.z = new E();
   return c;
}
void foo(F f) {
   C c = baz();
   f.w = c.z;
}
void zag() {
   F f = new F();
   foo(f);
   G.global = f;
}
```

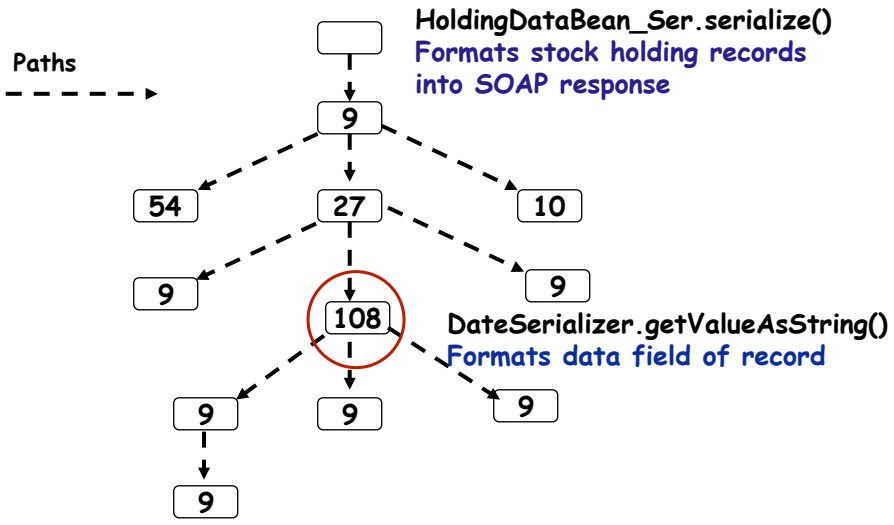
Disposition

A
B
C
D
E
F
G

Outline

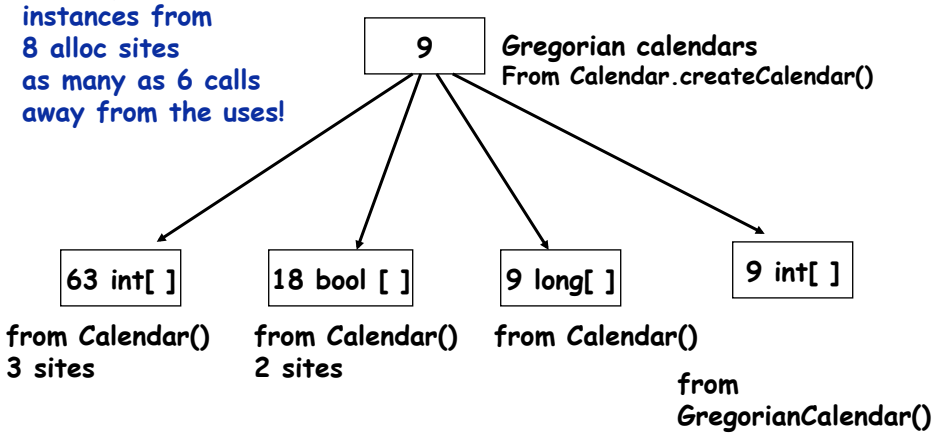
- Motivation
- Blended analysis paradigm
- Blended escape analysis
  - Example
  - Explanations of performance problems
  - Newest empirical results
- Related work
- Summary and future work

Calling Contexts with Lots of Temporaries

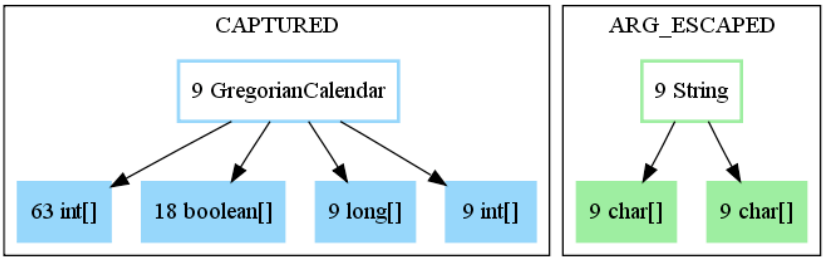


Reduced Connection Graph for DateSerializer.getValueAsString()

108 captured instances from 8 alloc sites as many as 6 calls away from the uses!



Visualized Results



DateSerializer.getValueAsString()

Outline

- Motivation
- Blended analysis paradigm
- Blended escape analysis
  - Example
  - Retrieving explanations of performance problems
  - Newest empirical results
- Related work
- Summary and future work

Experiments [ISSTA'07, FSE'08]

- Elude
  - Prototype built in WALA, uses Jinsight traces
- Benchmarks -Trade 6.0.1; Websphere Application Server 6.0.0.1; DB2 v8.2.0
  - Traced a single transaction
  - 4 configurations of Trade 6 depend on mode choices
    - Run-time mode (DB): Direct, EJB
    - Access mode: Standard, WebServices
  - Eclipse JDT Compiler 3.1.0
- Machine: Intel Core Duo 1.8Ghz, 3GB RAM, Linux 2.6 kernel

Size Comparison for Benchmarks

Benchmark (First 4 rows are Trade)	Allocated Types	Allocated Instances	Methods	Calls	Max Stack Depth
Direct/Std	30	186	710	4 484	26
Direct/WS	166	5 522	3 308	127 794	53
EJB/Std	82	1 751	1 978	60 936	62
EJB/WS	210	7 088	4 479	184 288	72
JDT Compiler	168	53 191	1 411	1 081 927	53

Metrics

Designed new metrics for blended escape analysis

- Measure effectiveness of pruning
  - Scalability of analysis – % of blocks in methods pruned
  - Precision improvement not observed in disposition metric

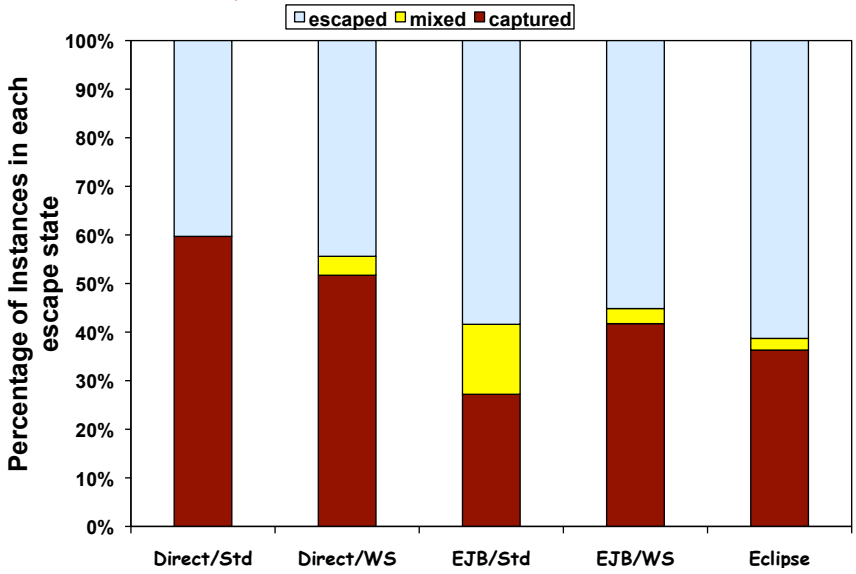
Scalability: %blocks pruned

Benchmark	Pruned BBs	Running Time (h:m:s)		Speedup
		Orig	Pruned	
Direct/Std	42.9%	0:00:19	0:00:16	1.2
Direct/WS	36.1%	0:06:38	0:03:17	2.0
EJB/Std	41.1%	0:02:40	0:02:02	1.18
EJB/WS	38.3%	N/A	18:33:13	N/A
Eclipse JDT	25.5%	N/A	6:09:15	N/A
Average	36.8%			1.6

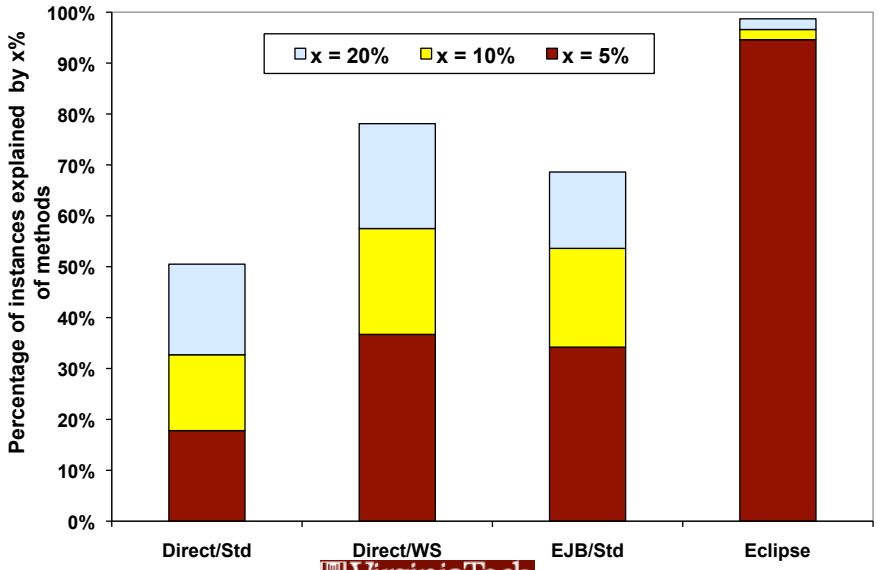
Metrics

- Measure usage of temporaries
  - Disposition- categorizes instances as globally: escaping, captured, mixed
  - Concentration- measures locality of temporary usage
  - Capturing depth- # calls between temporary creation and capture

Disposition of Instances



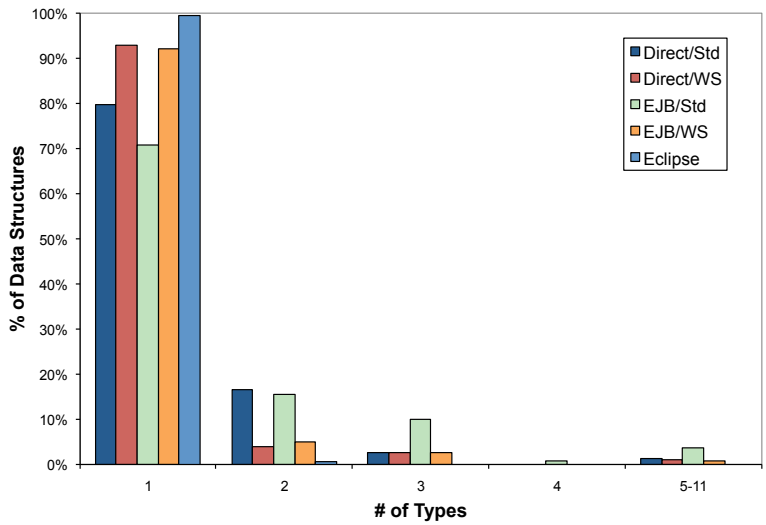
Concentration of Instances



Metrics

- Estimate temporary data structure complexity
  - # of types in data structures
  - # of allocating methods for objects in a data structure
  - Height of data structure
  - Maximum capturing distance

of Types in Data Structures



Outline

- Motivation
- Blended analysis paradigm
- Blended escape analysis
  - Example
  - Explanations of performance problems
  - Newest empirical results
- Related work
- Summary and future work

Related Work on Framework-based Systems

- Framework-based systems
  - Profiling execution, Ammons et. al. [ECOOP'04]
  - Characterizing where execution time is spent, Srinivas & Srinivasan [FSE '05]
  - Characterizing data structures in Java, Mitchell & Sevitsky [ECOOP '03], Mitchell [ECOOP '06], Blackburn et. al. [OOPSLA'06], Buytaert et. al. [ACES'05]
  - Characterizing data transformations, Mitchell et. al. [ECOOP '06]

Related Work on Analysis

- Often static analysis used to direct placement of instrumentation for dynamic analysis for efficiency
- Some previous uses of dynamic analysis to "direct" static:
  - Hybrid slicing, Gupta et.al. [TOSEM 1997]
  - Optimize model checking, Groce et.al. [TACAS'06]
  - Dynamic points-to in slicing, Mock et.al. [FSE'02]
  - Parameter mutability analysis, Artzi et.al. [MIT TR, 9/2006]

Outline

- Motivation
- Blended analysis paradigm
- Blended escape analysis
  - Example
  - Explanations of performance problems
  - Newest empirical results
- Related work
- Summary and future work

Summary

- New blended analysis paradigm
  - Combines dynamic program info with static analysis
  - Aimed at framework-intensive applications
  - Obtains high precision at reasonable cost
  - Algorithm aimed at greater scalability & precision, and better data structure characterization
- Problem studied: performance understanding of object churn
  - Novel use of escape analysis
  - Algorithm plus empirical results
  - New metrics to characterize usage of temporaries

Future Work

- Enhanced tooling
  - Visualization of connection graphs
  - Experiments with more precise calling structure representations
  - Integration into interactive tools

Future Work

- Explore wider applicability of blended analyses
  - Blended security analyses
    - Permissions
    - Information flow (i.e., taint)
  - Blended value-flow
    - Semantic exploration of specific test executions
    - Help with debugging