On the Usefulness of Liveness for Garbage Collection and Leak Detection

Martin Hirzel, Amer Diwan, and Antony Hosking

{hirzel,diwan}@cs.colorado.edu hosking@cs.purdue.edu

ECOOP June 2001 Budapest, Hungary

What is Liveness?

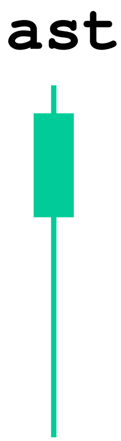
- A variable is *live* if its value will be used in the future.

…

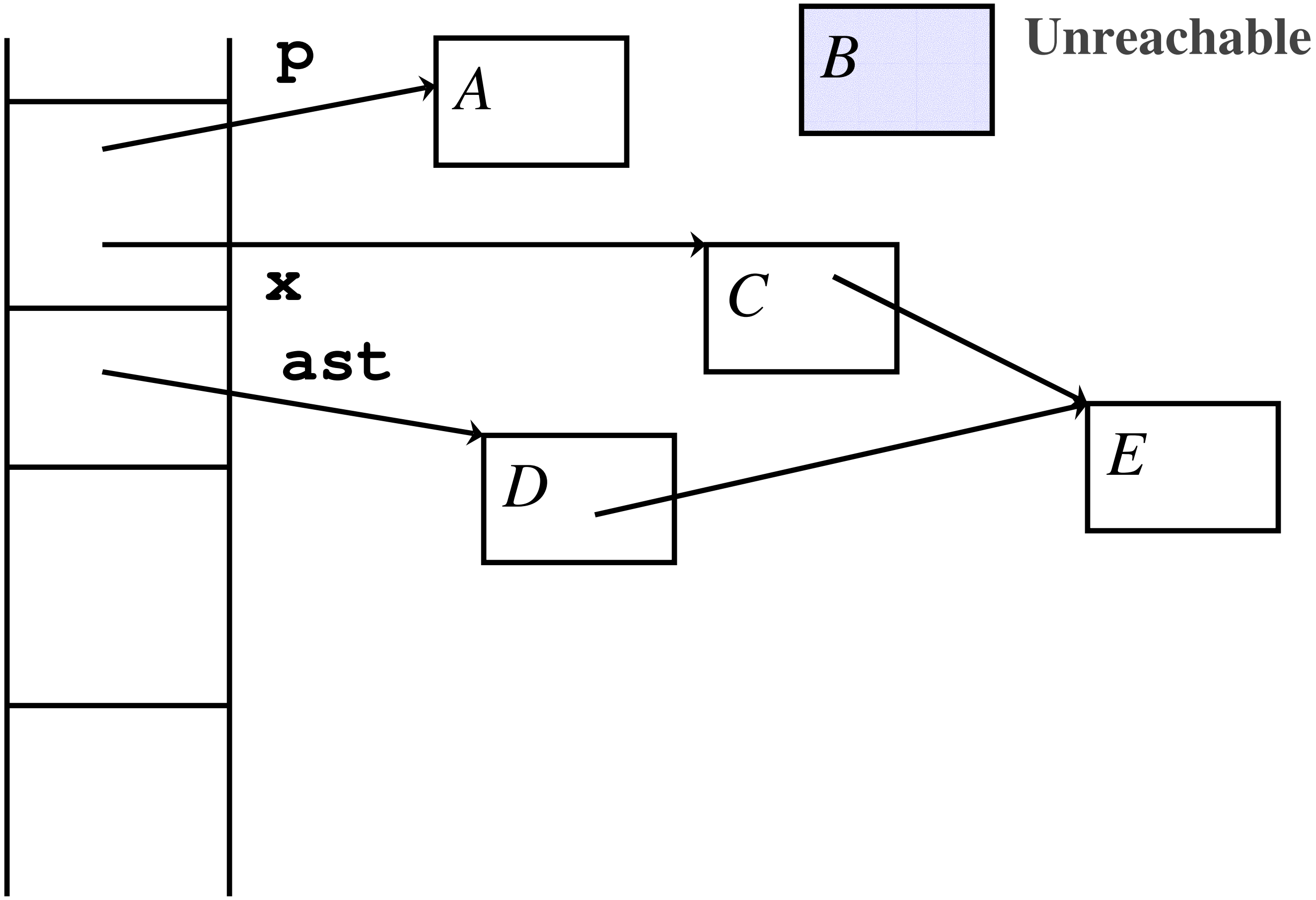
```
Tree *ast = parse();
Cfg *cfg = translate(ast);
```

⟨ code that does not use the value of `ast` ⟩

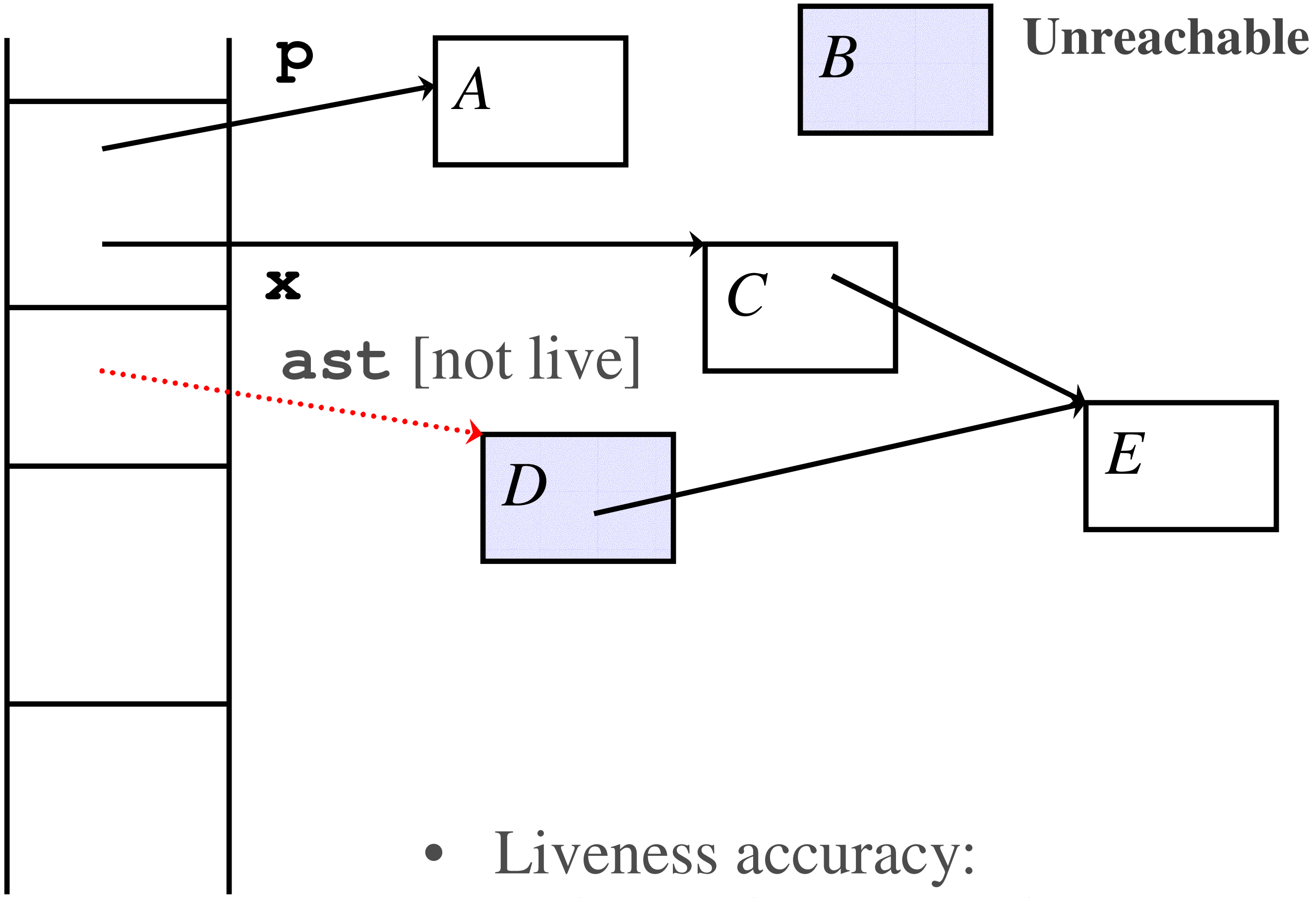
`...`



Accuracy and Reachability

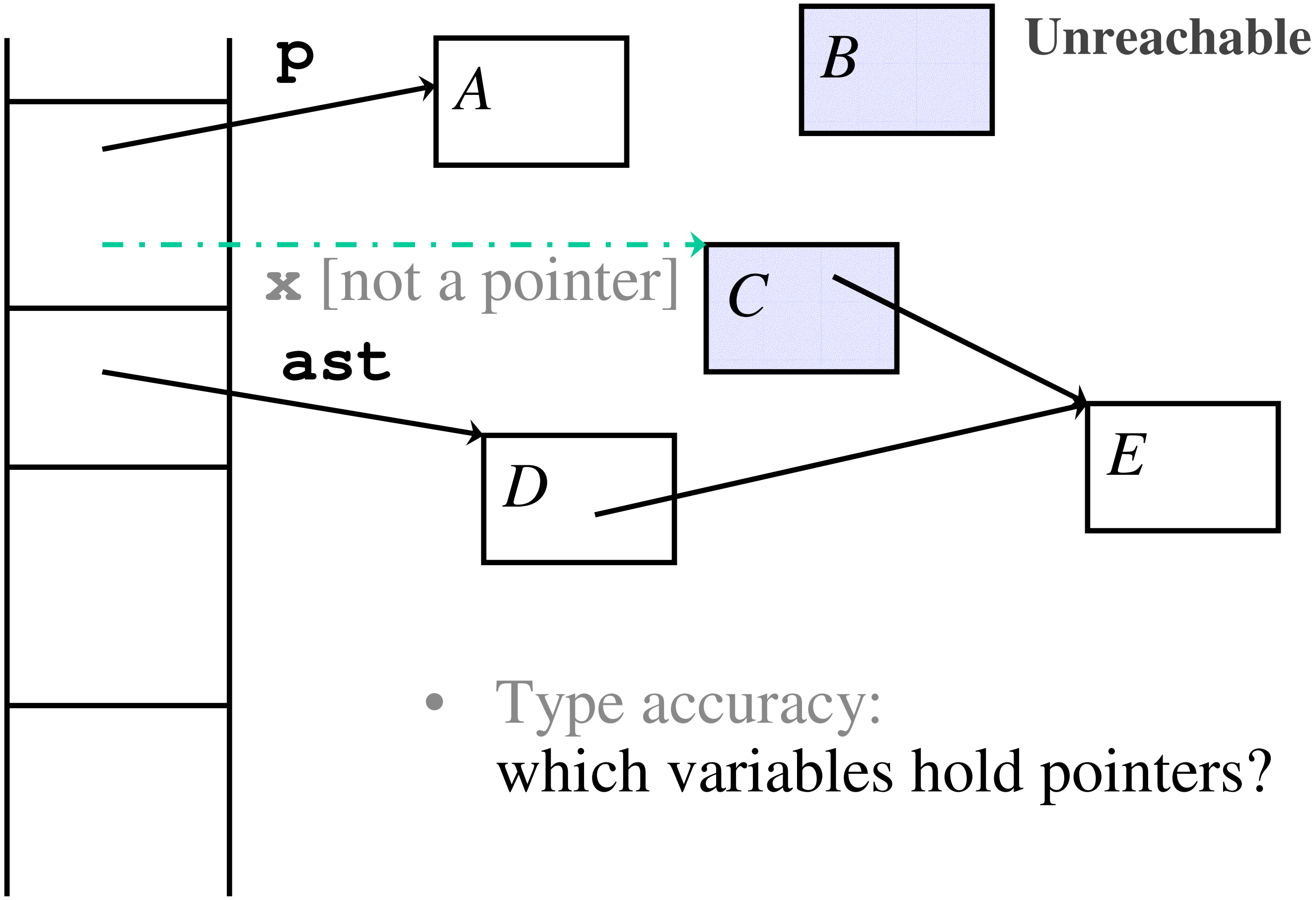


Accuracy and Reachability



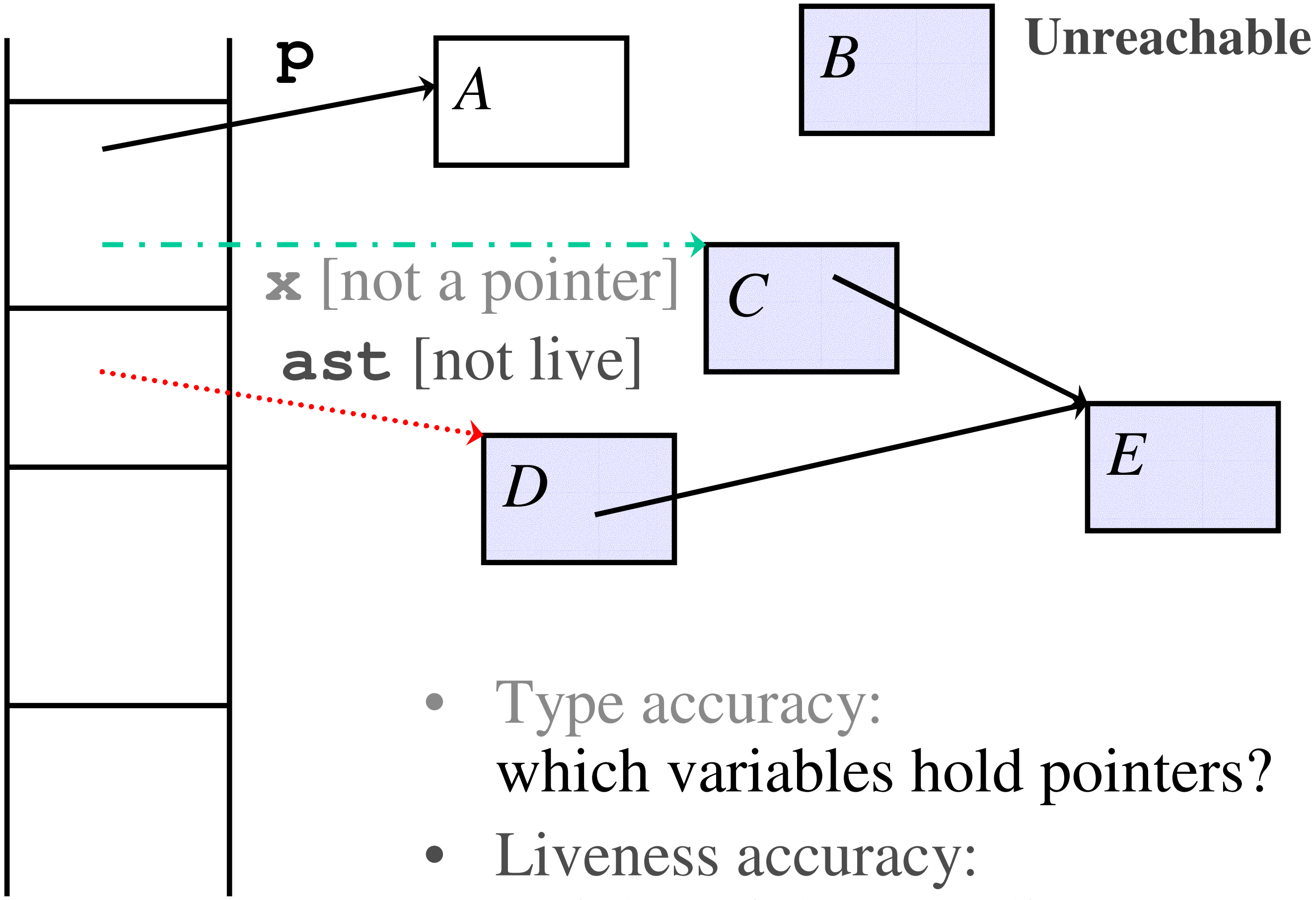
- Liveness accuracy:
  which variables are live?

Accuracy and Reachability



- Type accuracy: which variables hold pointers?

Accuracy and Reachability

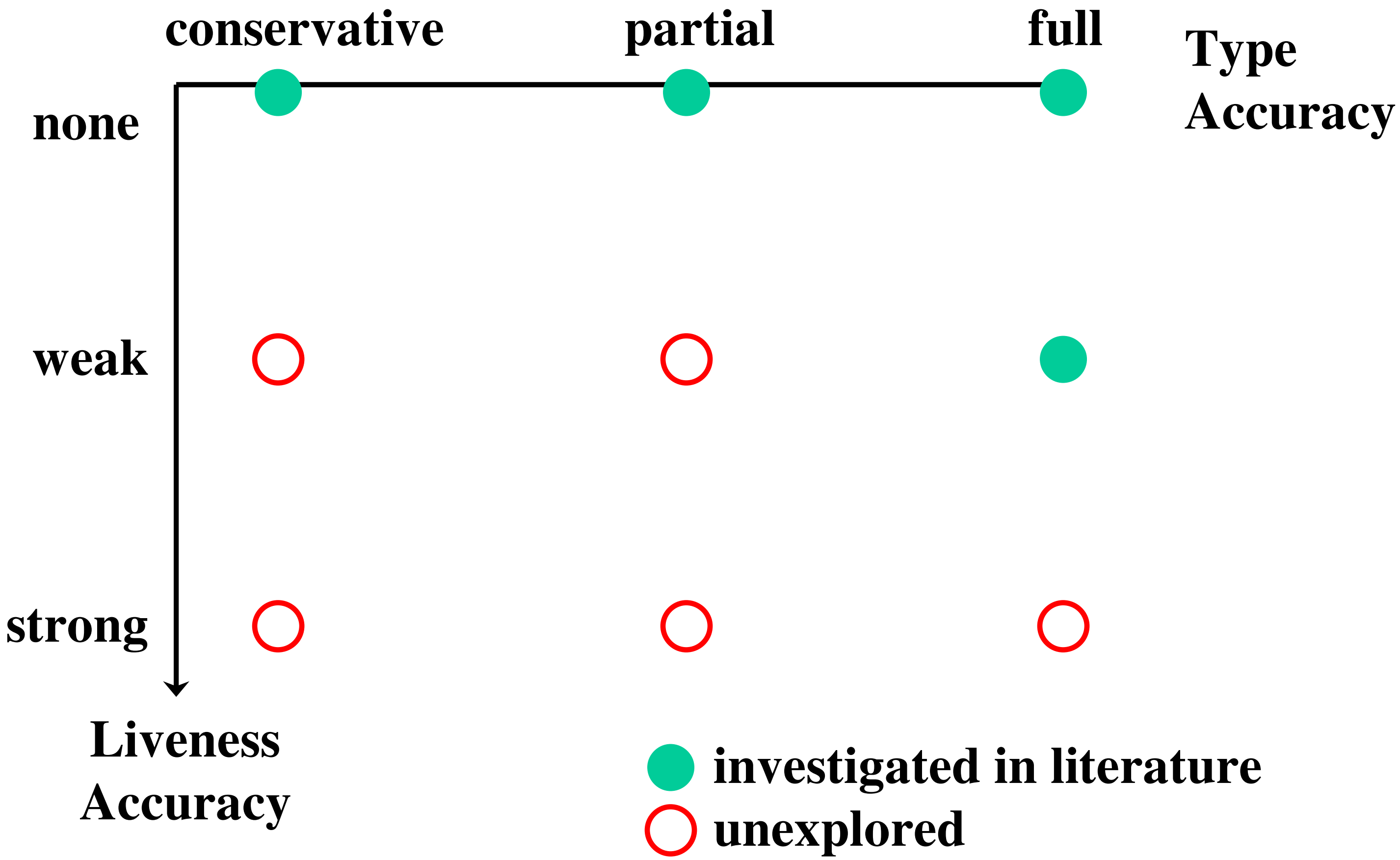


- Type accuracy:
  which variables hold pointers?
- Liveness accuracy:
  which variables are live?

The Questions

- Can liveness accuracy benefit reachability traversals?
  - Useful for garbage collection?
  - Useful for leak detection?
- What kind of analysis is necessary?

Motivation

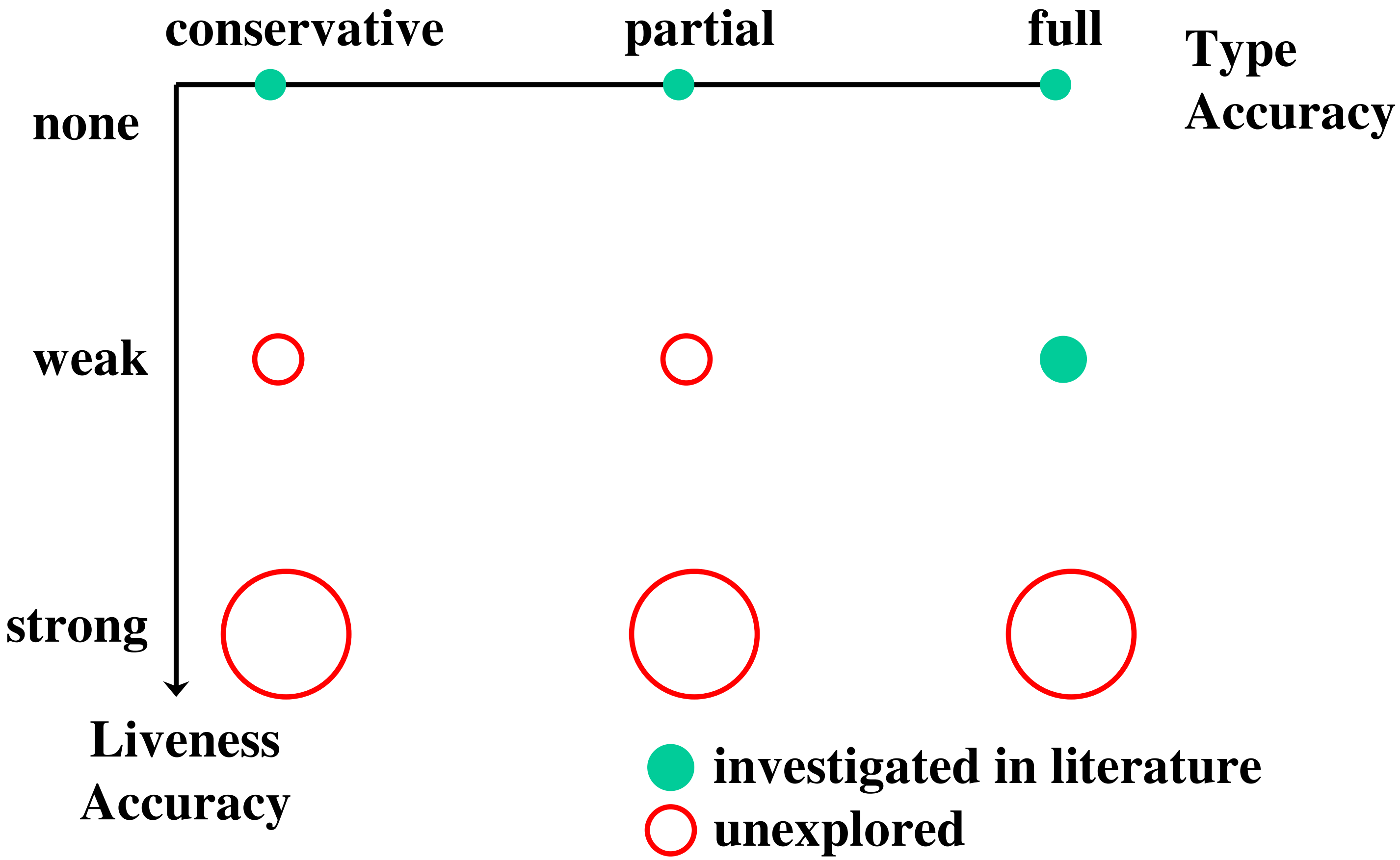


Liveness Accuracy \ Type Accuracy	conservative	partial	full
none	●	●	●
weak	○	○	●
strong	○	○	○

● investigated in literature

○ unexplored

Results in a Nutshell



Type Accuracy

Liveness Accuracy

	conservative	partial	full
none	●	●	●
weak	○	○	●
strong	○	○	○

● investigated in literature

○ unexplored

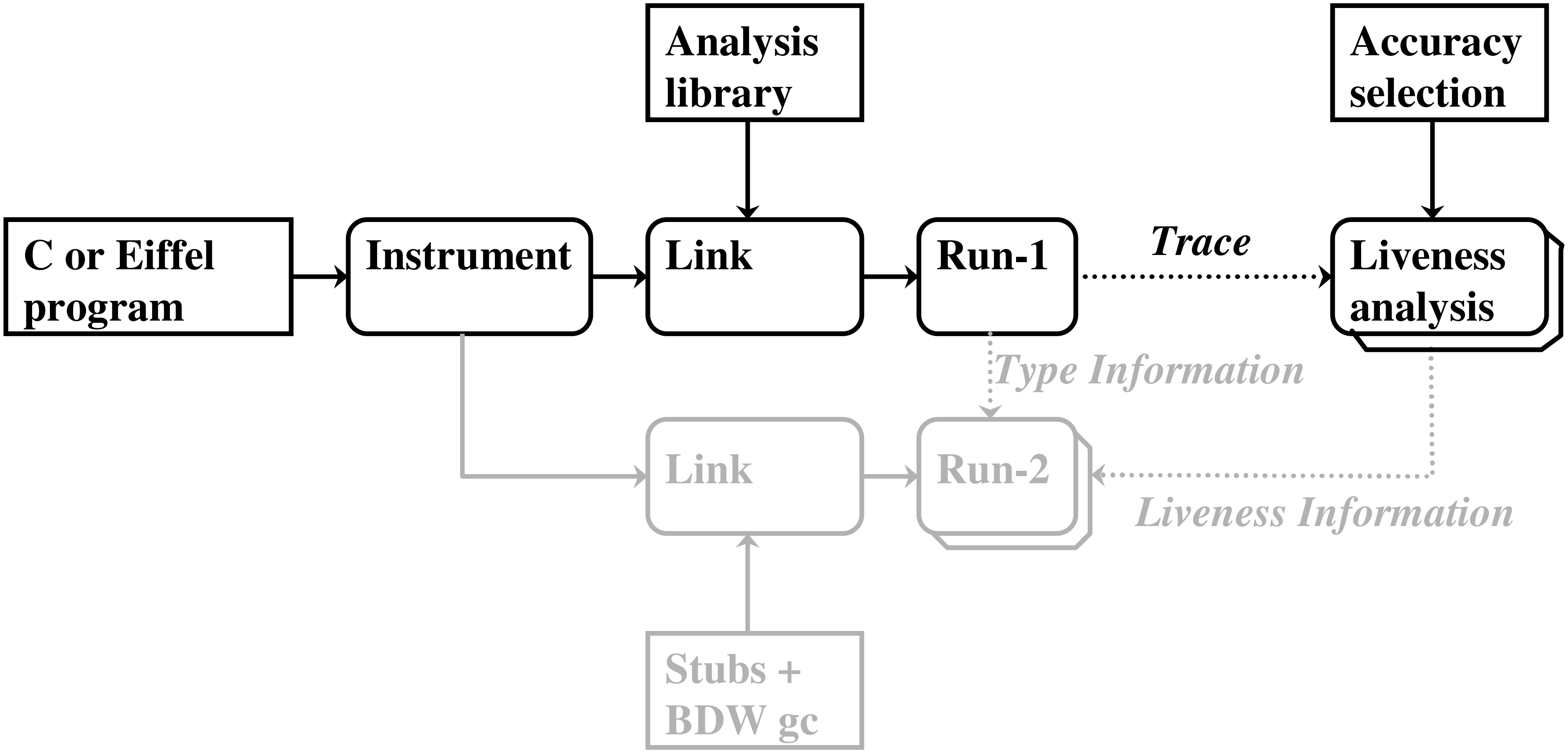
Outline

Introduction	• Motivation • Preview of results
Methodology	• Obtaining liveness information • Metrics
Results	• Reachable heap given various accuracy schemes
Conclusion	• Related work • Summary

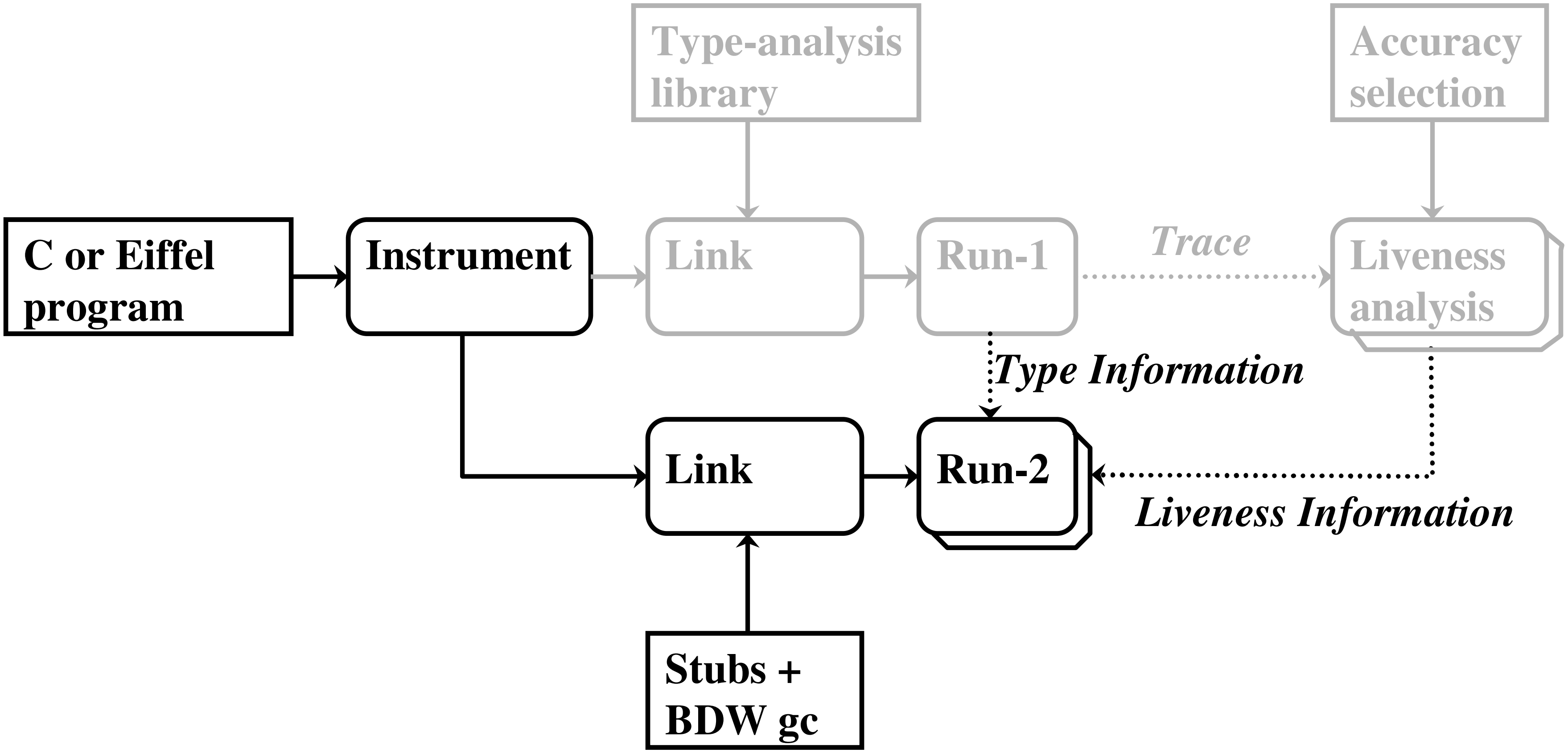
Liveness Approaches

- Static analysis
  - Compiler-analysis of source code
  - Disadvantage: difficult, cost + benefits unclear
- Dynamic analysis
  - Trace-based analysis
  - Disadvantage: two runs needed, limit study

Infrastructure for Experiments



Infrastructure for Experiments



Generating the Trace

Original Code

```
x = malloc(4*N);


i = 0;

while(i < N){
  y = x + 4*i;

  *y = i;


  i++;

}
```

Instrumented Code

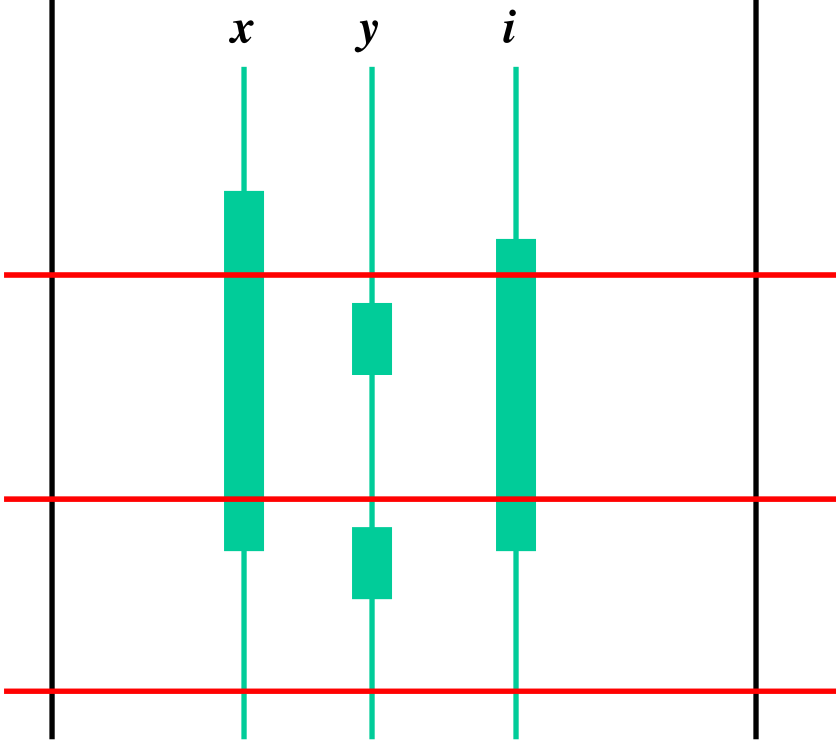
```
x = malloc(4*N);
note_allocation();
note_assign(&x);
i = 0;
note_assign(&i);
while(i < N){
  y = x + 4*i;
  note_assign(&y,&x,&i);
  *y = i;
  note_use(&y);
  note_assign(y,&i);
  i++;
  note_assign(&i,&i);
}
```

Trace

…
allocation(91)
assign(x)
assign(i)
assign(y, x, i)
use(y)
assign(91.0, i)
assign(i, i)
assign(y, x, i)
use(y)
assign(91.4, i)
…

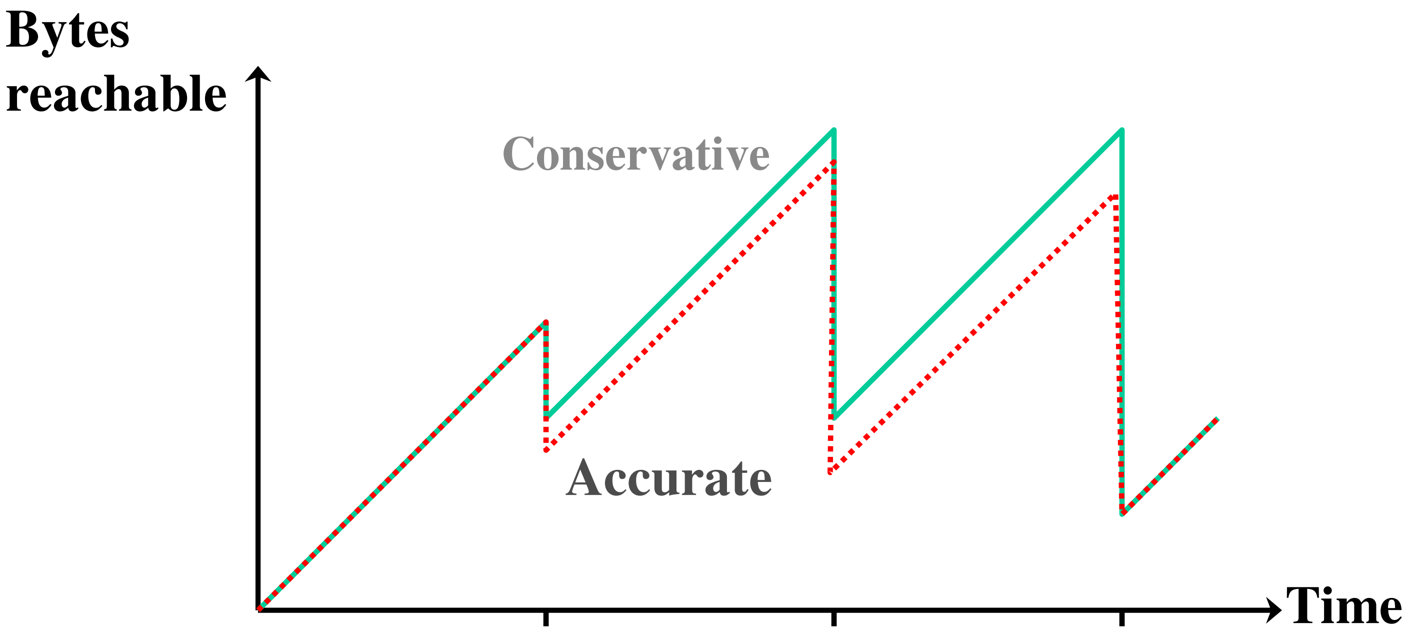
Analyzing the Trace

Trace	Simulated Liveness State	Resulting Information

…
allocation(91)
assign(x)
assign(i)
assign(y, x, i)
use(y)
assign(91.0, i)
assign(i, i)
assign(y, x, i)
use(y)
assign(91.4, i)
…
		T1: {***x*, *i***}
		T1: {***x*, *i***}
		T2: { }

"A variable is *live* if its value will be used in the future."

Usefulness Metric for Accuracy



Reachability traversal:	1	2	3
Reachability reduction:	10%	20%	0%

⇒ **The accurate scheme reduced reachability by 10% on average.**

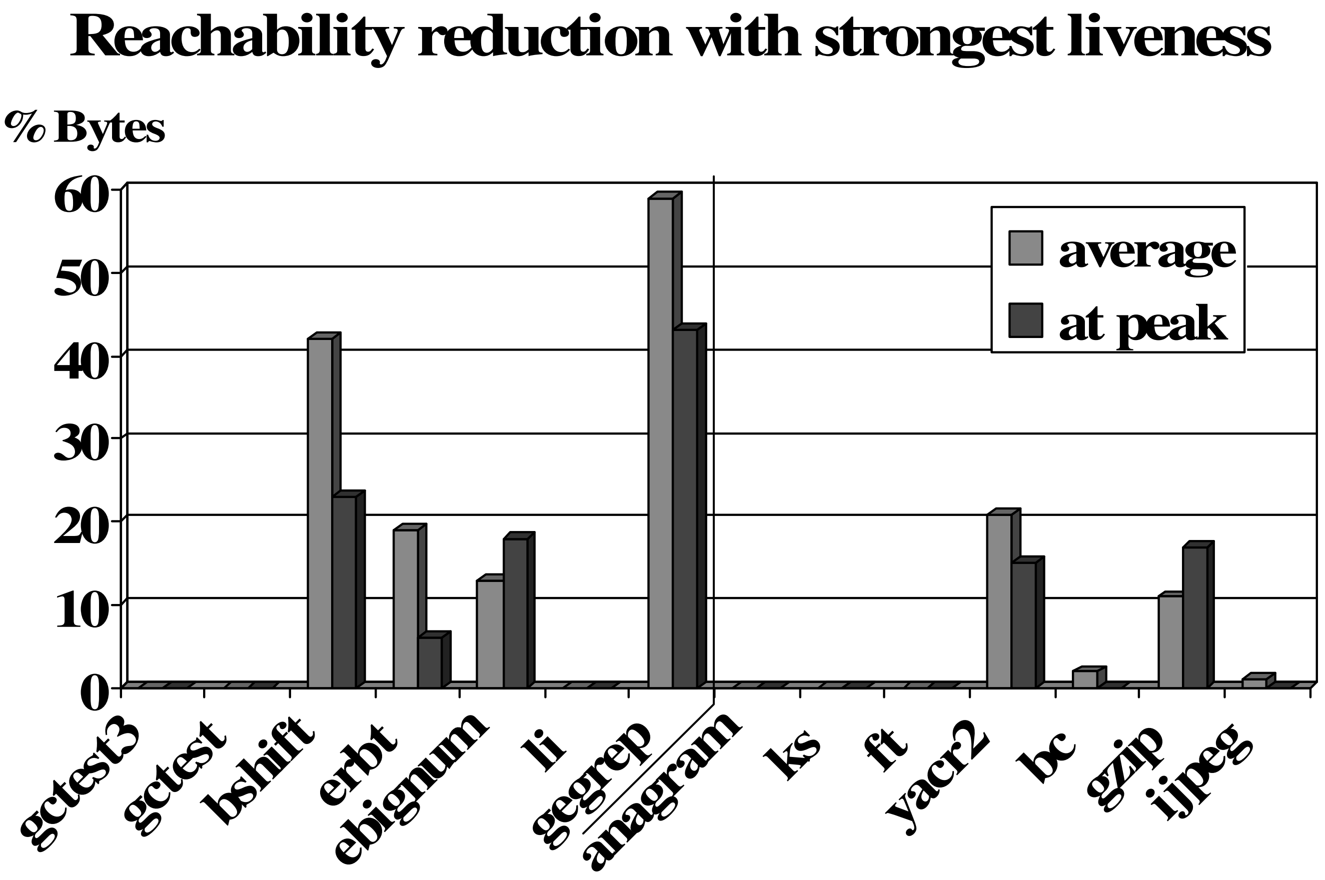
Outline

Introduction	• Motivation • Preview of results
Methodology	• Obtaining liveness information • Metrics
Results	• Reachable heap given various levels of accuracy
Conclusion	• Related work • Summary

Benchmarks

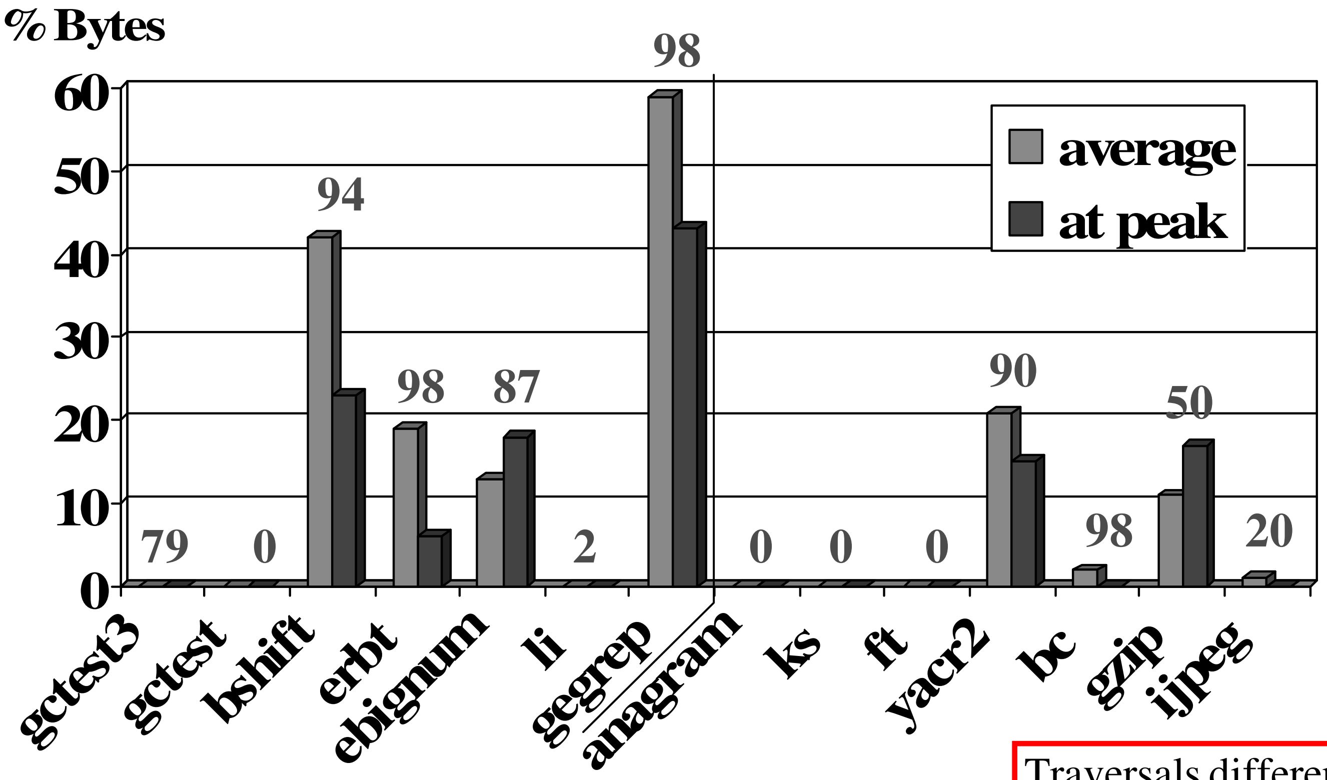
Name	Language	Lines of Code	Total allocation [Bytes]	Author/Source
Programs written with GC in mind:				
gctest3	C	85	2 200 004	Bartlett
gctest	C	196	1 123 180	Bartlett
bshift	Eiffel	350	28 700	Hirzel
erbt	Eiffel	927	222 300	Durian
ebignum	Eiffel	3 137	109 548	Hillion
li	C	7 597	9 030 872	Spec95
gegrep	Eiffel	17 185	106 392	Bezault
Programs with explicit deallocation:				
anagram	C	647	259 512	Austin
ks	C	782	7 920	Austin
ft	C	2 156	166 832	Austin
yacr2	C	3 979	41 380	Austin
bc	C	7 308	12 382 400	Austin
gzip	C	8 163	14 180	GNU
ijpeg	C	31 211	148 664	Spec95

Usefulness of Liveness



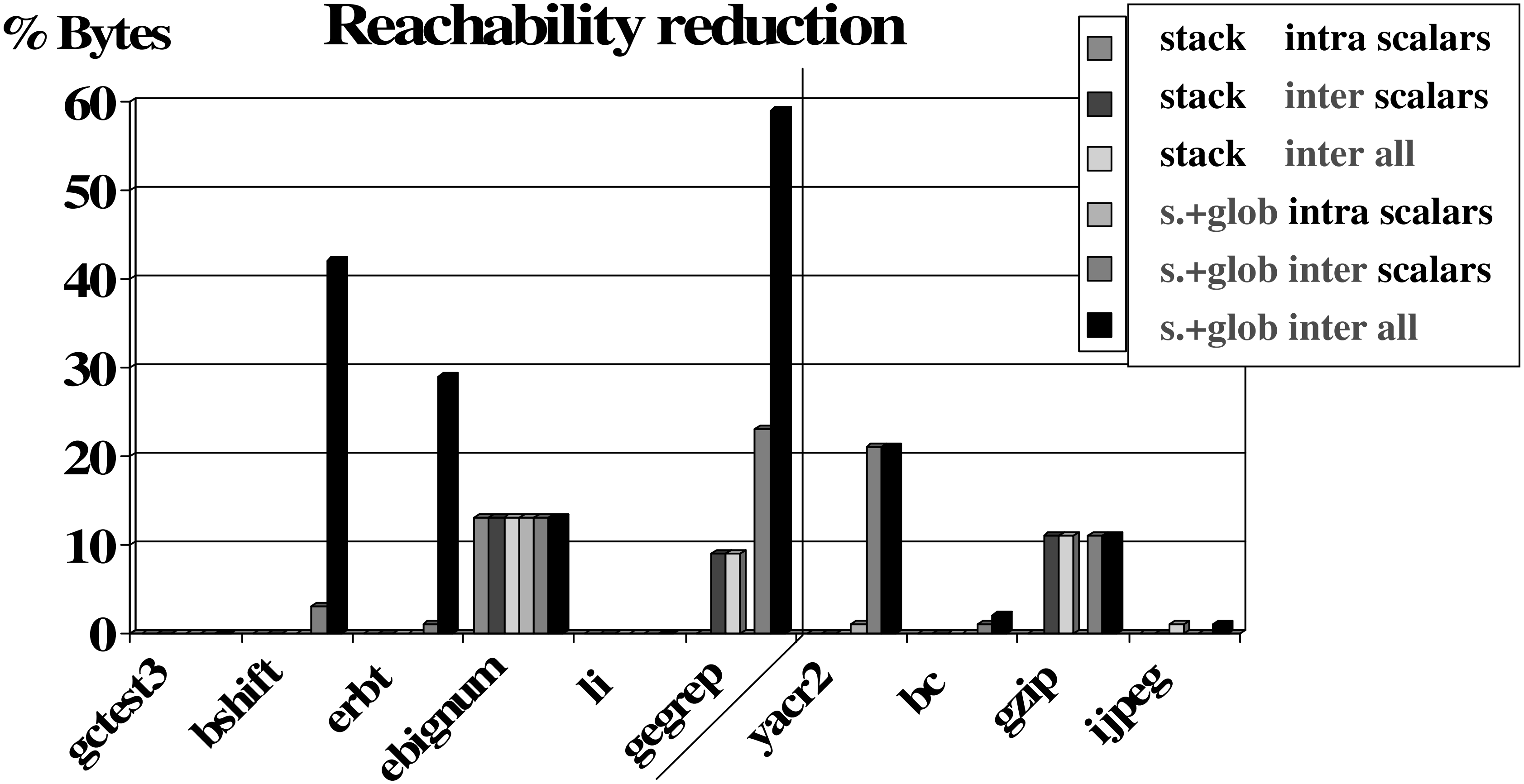
Usefulness of Liveness

Reachability reduction with strongest liveness

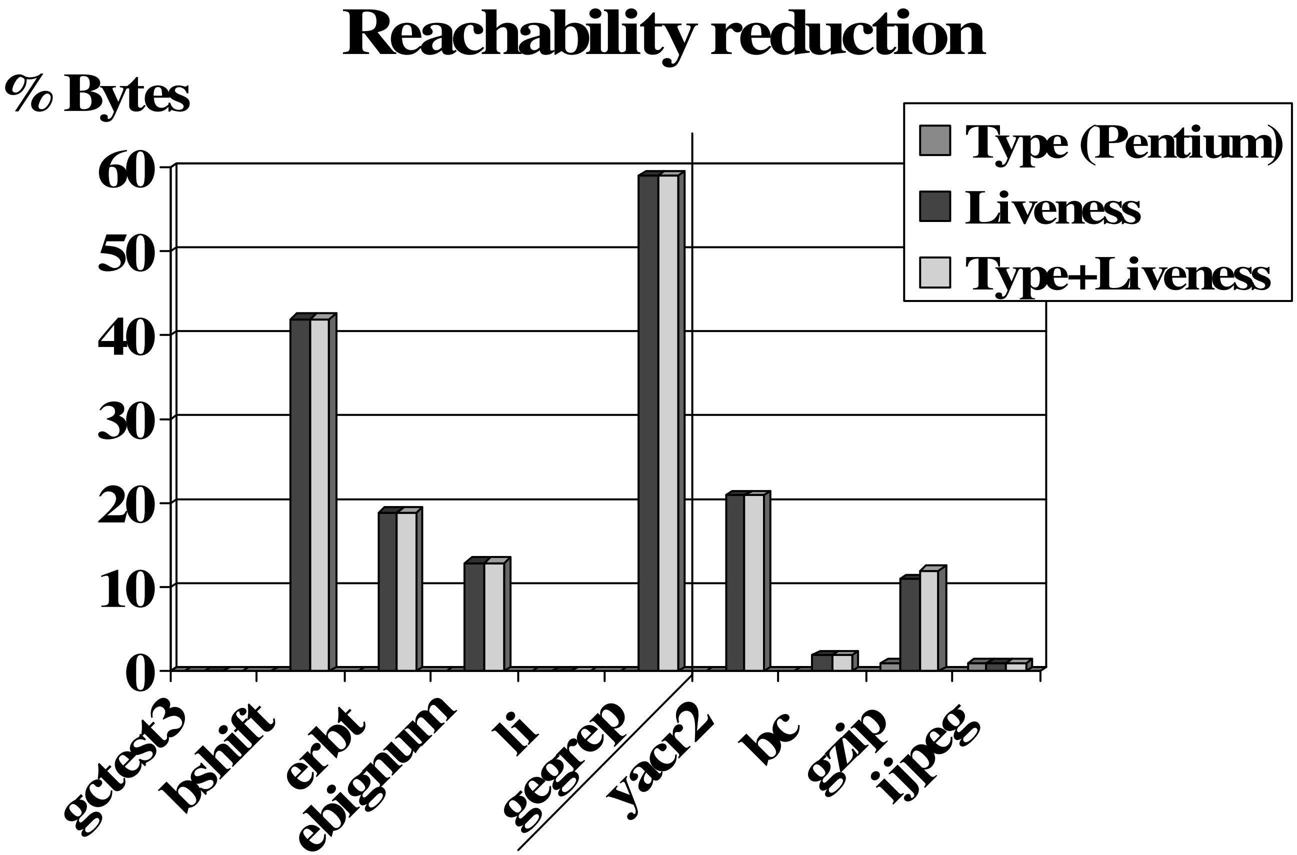


$$\frac{\text{Traversals different}}{\text{Num traversals}}\%$$

Different Levels of Liveness



Type versus Liveness Accuracy



Validation

- Comparing liveness information found in different runs
  - For how many locations did the obtained liveness information differ?

Benchmark	Stack % different	Global % different
gegrep	0.7	0.0
yacr2	2.7	0.0
gzip	1.3	2.2

Outline

Introduction	• Motivation • Preview of results
Methodology	• Obtaining liveness information • Metrics
Results	• Reachable heap given various levels of accuracy
Conclusion	• Related work • Summary

Related Work

- Evaluating Accuracy
  - Hirzel, Diwan: On the type accuracy of garbage collection. ISMM 2000.
  - Shaham, Kolodner, Sagiv: On the effectiveness of GC in Java. ISMM 2000.
- Implementing Accuracy
  - [Bartlett1988] [DiwanMossHudson1992] [SmithMorrisett1998] [Zorn1993] [AgesenDetlefsMoss1998] …

Summary

- Liveness accuracy can be very useful for reachability traversals.
- Strong analyses are necessary to reach significantly fewer Bytes.
- Type accuracy was not very useful in these experiments.