SLE 2010

Automated Selective Caching for Reference Attribute Grammars

Emma Söderberg Görel Hedin

Department of Computer Science
Lund University

October 12th 2010

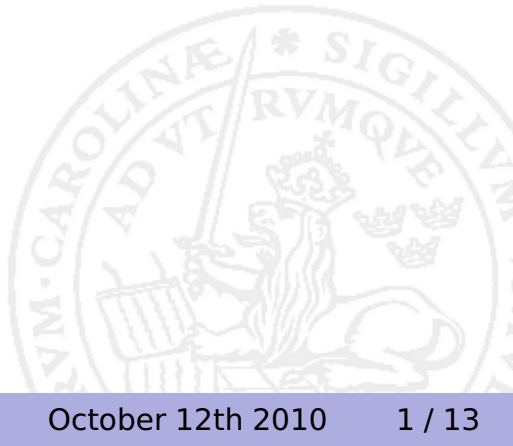
Overview

RAGs : High-level compiler specification

Objective : Speed up generated compiler

Approach : Profiling of the genenerated compiler

Outline : RAGs $\rightarrow$
Caching $\rightarrow$
Profiling $\rightarrow$
Evaluation $\rightarrow$
Conclusions



Reference Attribute Grammars

Abstract Grammar:
Block = Stmt*;
Stmt = Block|Decl|Use;

Abstract Syntax Tree

- *Grammar*
- *Attributes*
  - *Defined by equations*
- *References*
  - *Super-imposed graphs*
  - *On-demand evaluation*

Attributes:
int Use.a = b.c;
Decl Use.b = ...;
int Decl.c = ...;

Reference Attribute Grammars

Abstract Grammar:
Block = Stmt*;
Stmt = Block|Decl|Use;

Abstract Syntax Tree

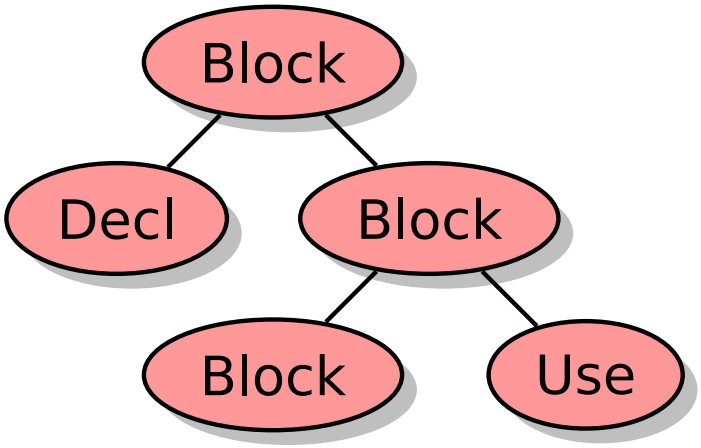
- *Grammar*
- *Attributes*
  - *Defined by equations*
- *References*
  - *Super-imposed graphs*
  - *On-demand evaluation*

Attributes:
int Use.a = b.c;
Decl Use.b = ...;
int Decl.c = ...;

Reference Attribute Grammars

Abstract Grammar:
Block = Stmt*;
Stmt = Block|Decl|Use;

Abstract Syntax Tree



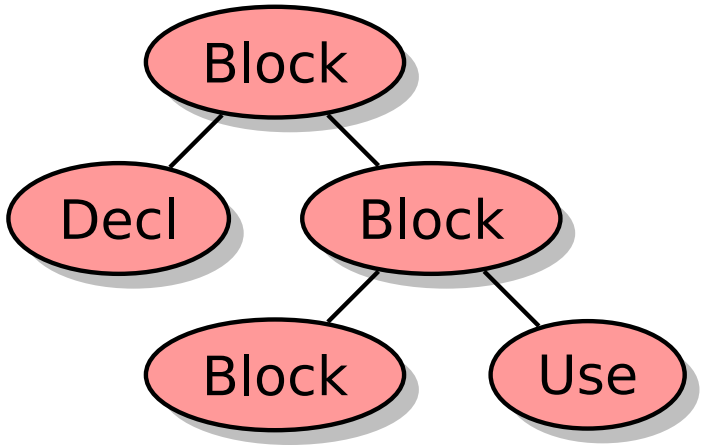
- *Grammar*
- *Attributes*
  - *Defined by equations*
- *References*
  - *Super-imposed graphs*
  - *On-demand evaluation*

Attributes:
int Use.a = b.c;
Decl Use.b = ...;
int Decl.c = ...;

Reference Attribute Grammars

Abstract Grammar:
Block = Stmt*;
Stmt = Block|Decl|Use;

Abstract Syntax Tree



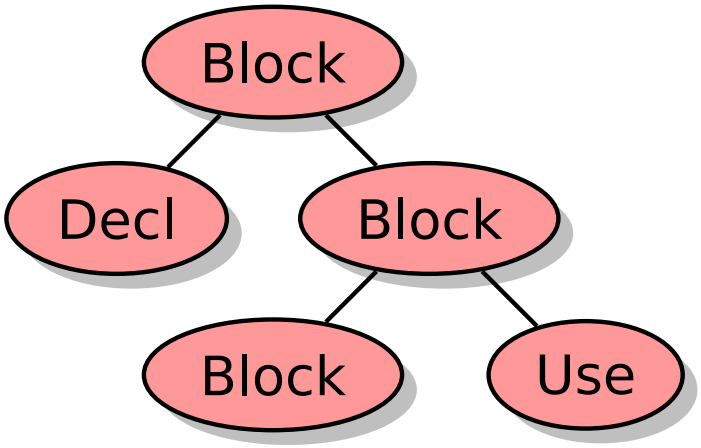
- *Grammar*
- *Attributes*
  - *Defined by equations*
- *References*
  - *Super-imposed graphs*
  - *On-demand evaluation*

Attributes:
int Use.a = b.c;
Decl Use.b = ...;
int Decl.c = ...;

Reference Attribute Grammars

Abstract Grammar:
Block = Stmt*;
Stmt = Block|Decl|Use;

Abstract Syntax Tree



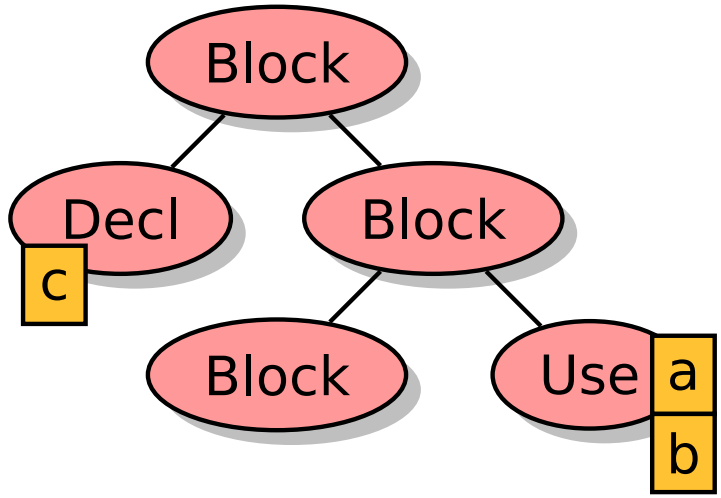
- *Grammar*
- *Attributes*
  - *Defined by equations*
- *References*
  - *Super-imposed graphs*
  - *On-demand evaluation*

Attributes:
int Use.a
Decl Use.b
int Decl.c

Reference Attribute Grammars

Abstract Grammar:
Block = Stmt*;
Stmt = Block|Decl|Use;

Abstract Syntax Tree



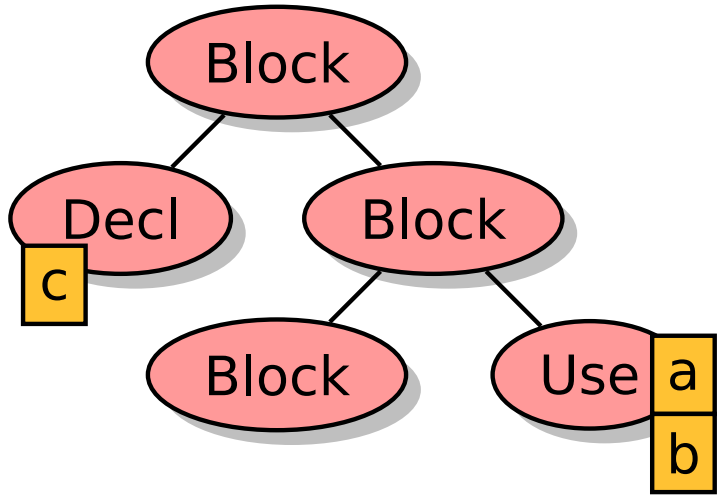
- *Grammar*
- *Attributes*
  - *Defined by equations*
- *References*
  - *Super-imposed graphs*
  - *On-demand evaluation*

Attributes:
int Use.a
Decl Use.b
int Decl.c

Reference Attribute Grammars

Abstract Grammar:
Block = Stmt*;
Stmt = Block|Decl|Use;

Abstract Syntax Tree



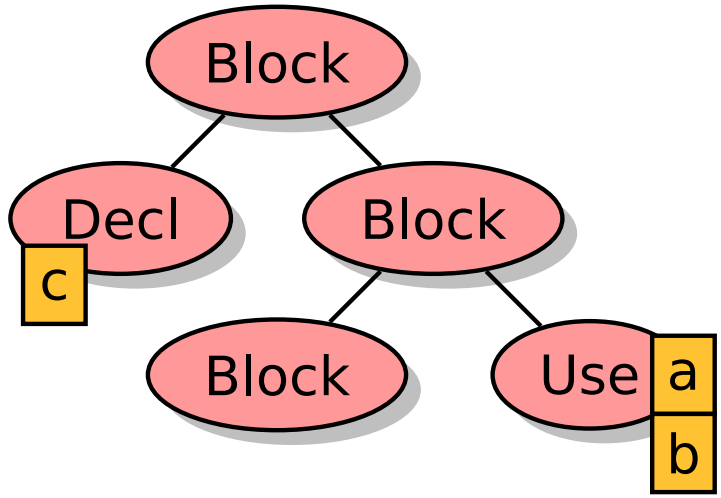
- *Grammar*
- *Attributes*
  - *Defined by equations*
- *References*
  - *Super-imposed graphs*
  - *On-demand evaluation*

Attributes:
int Use.a
Decl Use.b
int Decl.c

Reference Attribute Grammars

Abstract Grammar:
Block = Stmt*;
Stmt = Block|Decl|Use;

Abstract Syntax Tree



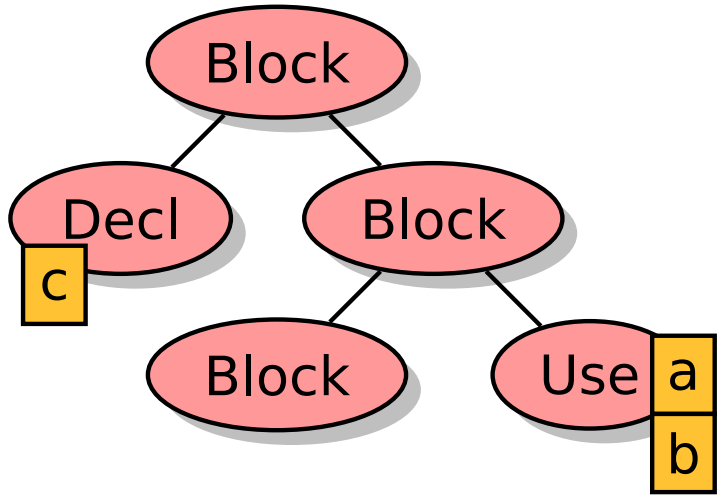
- *Grammar*
- *Attributes*
  - *Defined by equations*
- *References*
  - *Super-imposed graphs*
  - *On-demand evaluation*

Attributes:
int Use.a = b.c;
Decl Use.b = ...;
int Decl.c = ...;

Reference Attribute Grammars

Abstract Grammar:
Block = Stmt*;
Stmt = Block|Decl|Use;

Abstract Syntax Tree



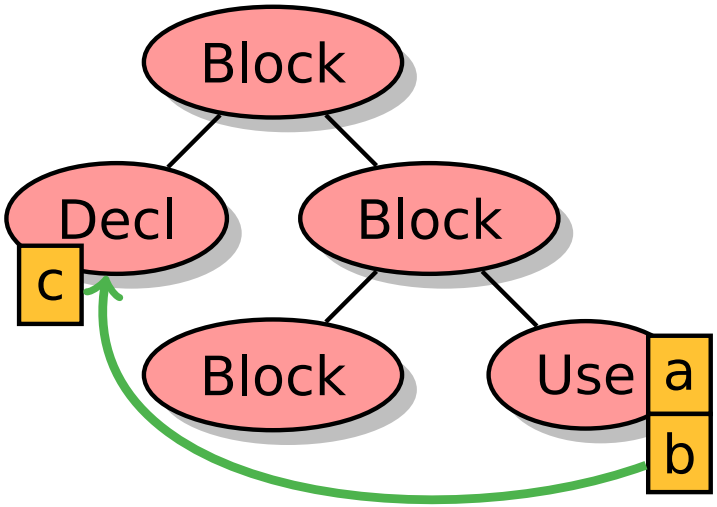
- *Grammar*
- *Attributes*
  - *Defined by equations*
- *References*
  - *Super-imposed graphs*
  - *On-demand evaluation*

Attributes:
int Use.a = b.c;
Decl Use.b = ...;
int Decl.c = ...;

Reference Attribute Grammars

Abstract Grammar:
Block = Stmt*;
Stmt = Block|Decl|Use;

Abstract Syntax Tree



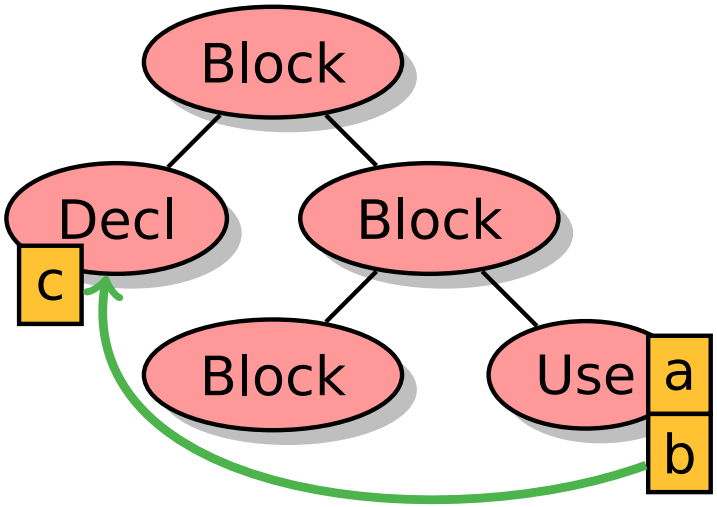
- *Grammar*
- *Attributes*
  - *Defined by equations*
- *References*
  - *Super-imposed graphs*
  - *On-demand evaluation*

Attributes:
int Use.a = b.c;
Decl Use.b = ...;
int Decl.c = ...;

Reference Attribute Grammars

Abstract Grammar:
Block = Stmt*;
Stmt = Block|Decl|Use;

Abstract Syntax Tree



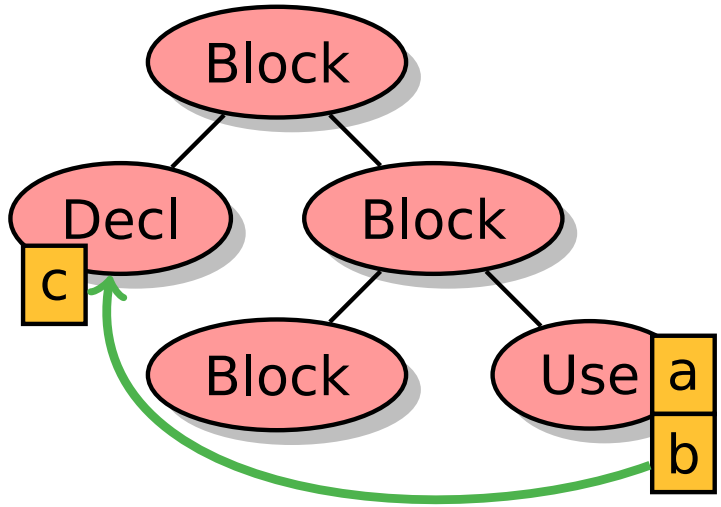
- *Grammar*
- *Attributes*
  - *Defined by equations*
- *References*
  - *Super-imposed graphs*
  - *On-demand evaluation*

Attributes:
int Use.a = b.c;
Decl Use.b = ...;
int Decl.c = ...;

Reference Attribute Grammars

Abstract Grammar:
Block = Stmt*;
Stmt = Block|Decl|Use;

Abstract Syntax Tree



- *Grammar*
- *Attributes*
  - *Defined by equations*
- *References*
  - *Super-imposed graphs*
  - *On-demand evaluation*

Attributes:
int Use.a = b.c;
Decl Use.b = ...;
int Decl.c = ...;

Attribute Evaluation

Without Caching

```
class Node {
  int a() {
    return b().c();
  }
}
```

With Caching

```
class Node {
  boolean a_cached = false;
  int a_value;
  int a() {
    if (! a_cached) {
      a_value = b().c();
      a_cached = true;
    }
    return a_value;
  }
}
```

JastAdd Caching Scheme

- *Example: a = b.c*
- *Break even?*

Attribute Evaluation

Without Caching

```
class Node {
  int a() {
    return b().c();
  }
}
```

With Caching

```
class Node {
  boolean a_cached = false;
  int a_value;
  int a() {
    if (! a_cached) {
      a_value = b().c();
      a_cached = true;
    }
    return a_value;
  }
}
```

JastAdd Caching Scheme

- *Example: a = b.c*
- *Break even?*

Attribute Evaluation

Without Caching

```
class Node {
  int a() {
    return b().c();
  }
}
```

Exponential!

With Caching

```
class Node {
  boolean a_cached = false;
  int a_value;
  int a() {
    if (! a_cached) {
      a_value = b().c();
      a_cached = true;
    }
    return a_value;
  }
}
```

JastAdd Caching Scheme

- *Example: a = b.c*
- *Break even?*

Attribute Evaluation

Without Caching

```
class Node {
  int a() {
    return b().c();
  }
}
```

Exponential!

With Caching

```
class Node {
  boolean a_cached = false;
  int a_value;
  int a() {
    if (! a_cached) {
      a_value = b().c();
      a_cached = true;
    }
    return a_value;
  }
}
```

JastAdd Caching Scheme

- *Example: a = b.c*
- *Break even?*

Attribute Evaluation

Without Caching

```
class Node {
  int a() {
    return b().c();
  }
}
```

Exponential!

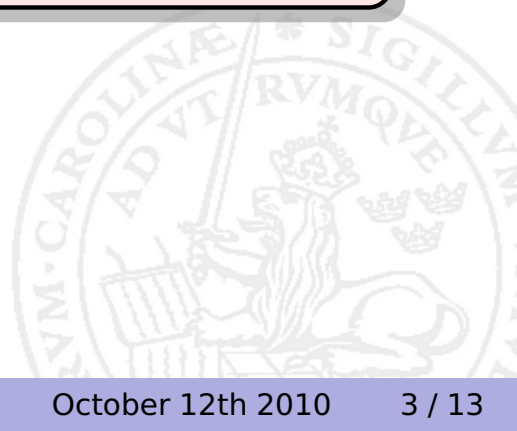
With Caching

```
class Node {
  boolean a_cached = false;
  int a_value;
  int a() {
    if (! a_cached) {
      a_value = b().c();
      a_cached = true;
    }
    return a_value;
  }
}
```

Linear!

JastAdd Caching Scheme

- *Example: a = b.c*
- *Break even?*



Attribute Evaluation

Without Caching

```
class Node {
  int a() {
    return b().c();
  }
}
```

Exponential!

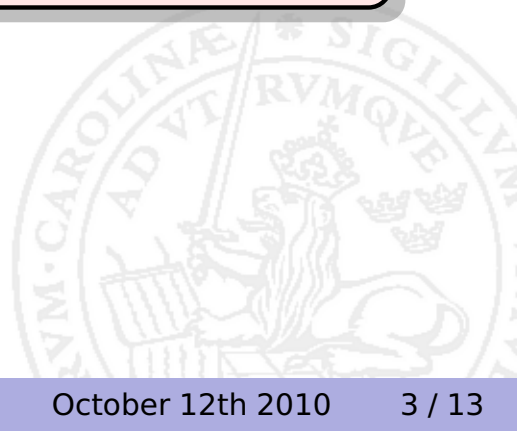
With Caching

```
class Node {
  boolean a_cached = false;
  int a_value;
  int a() {
    if (! a_cached) {
      a_value = b().c();
      a_cached = true;
    }
    return a_value;
  }
}
```

Linear!

JastAdd Caching Scheme

- *Example: a = b.c*
- *Break even?*



Attribute Evaluation

Without Caching

```
class Node {
  int a() {
    return b().c();
  }
}
```

Exponential!

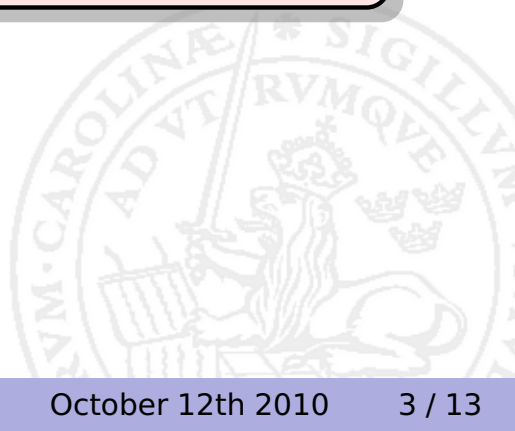
With Caching

```
class Node {
  boolean a_cached = false;
  int a_value;
  int a() {
    if (! a_cached) {
      a_value = b().c();
      a_cached = true;
    }
    return a_value;
  }
}
```

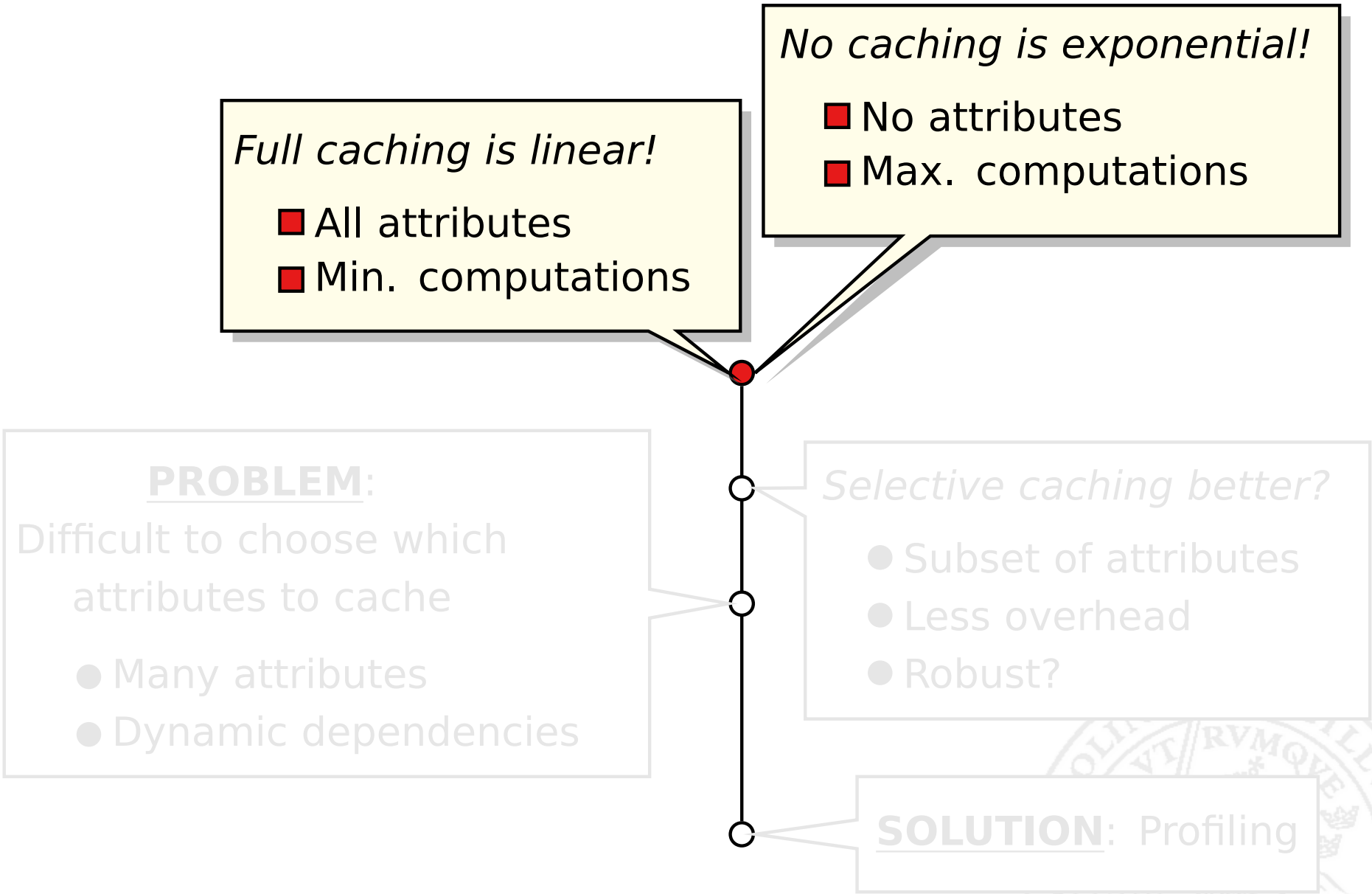
Linear!

JastAdd Caching Scheme

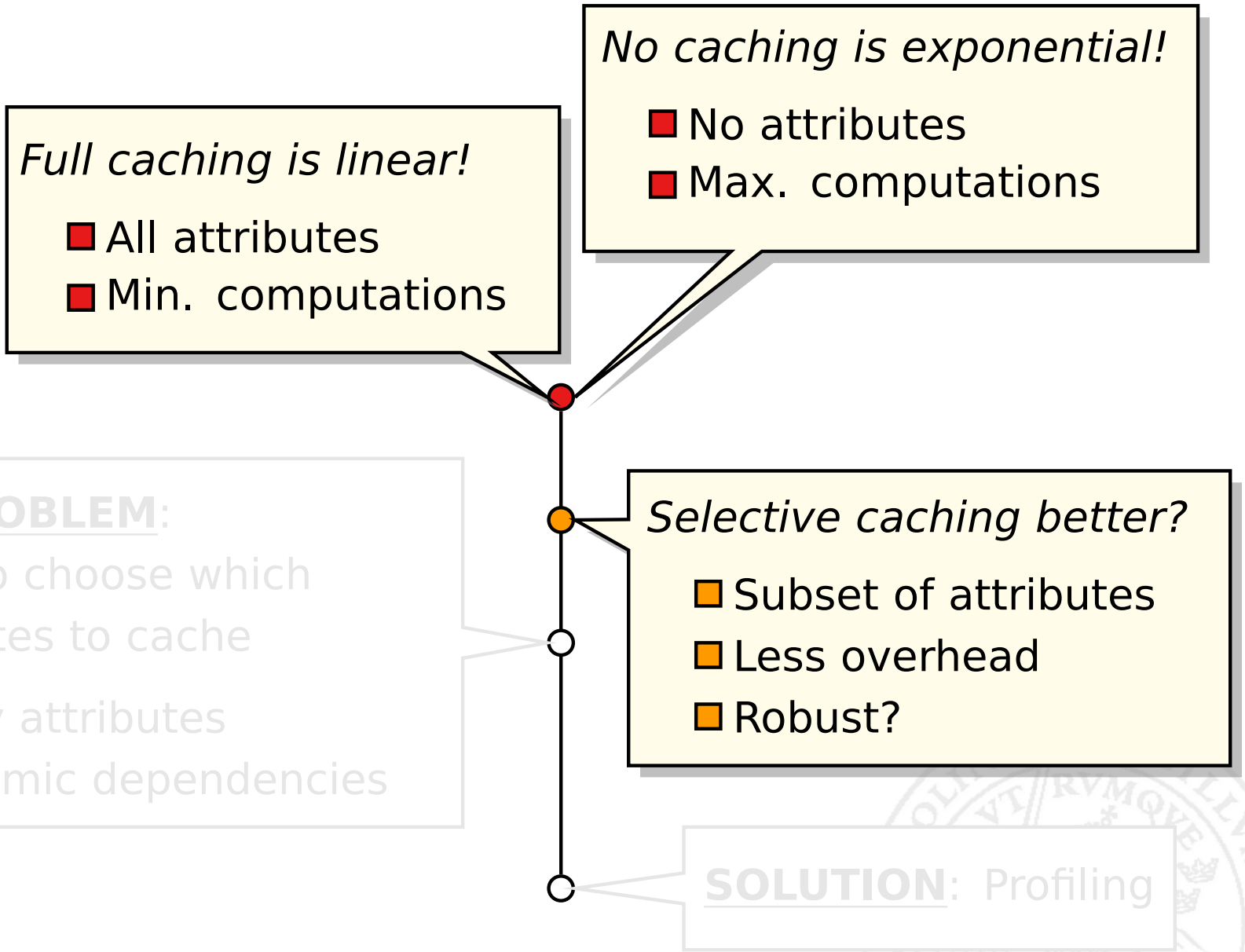
- *Example: a = b.c*
- *Break even?*



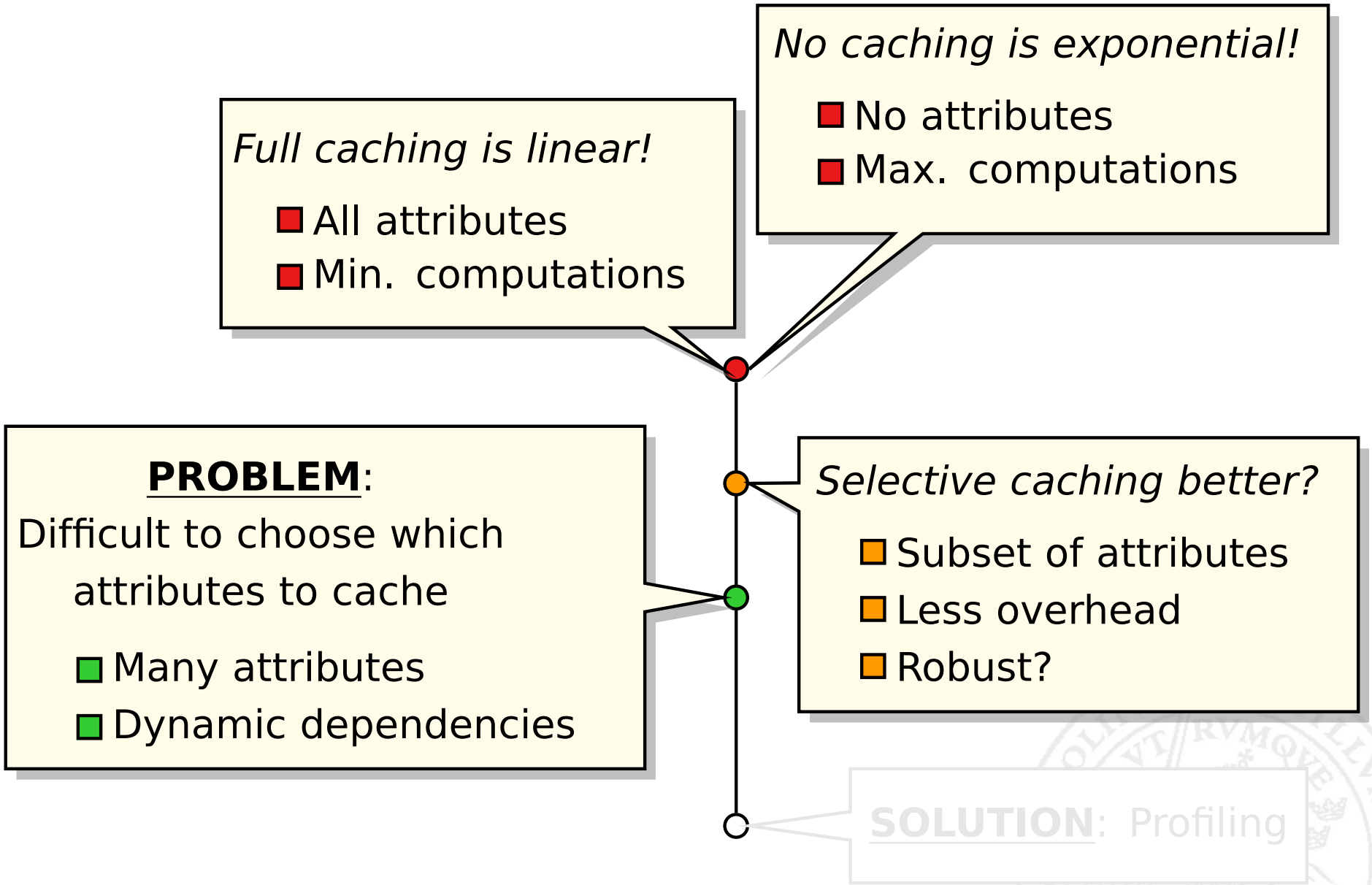
Caching of RAGs



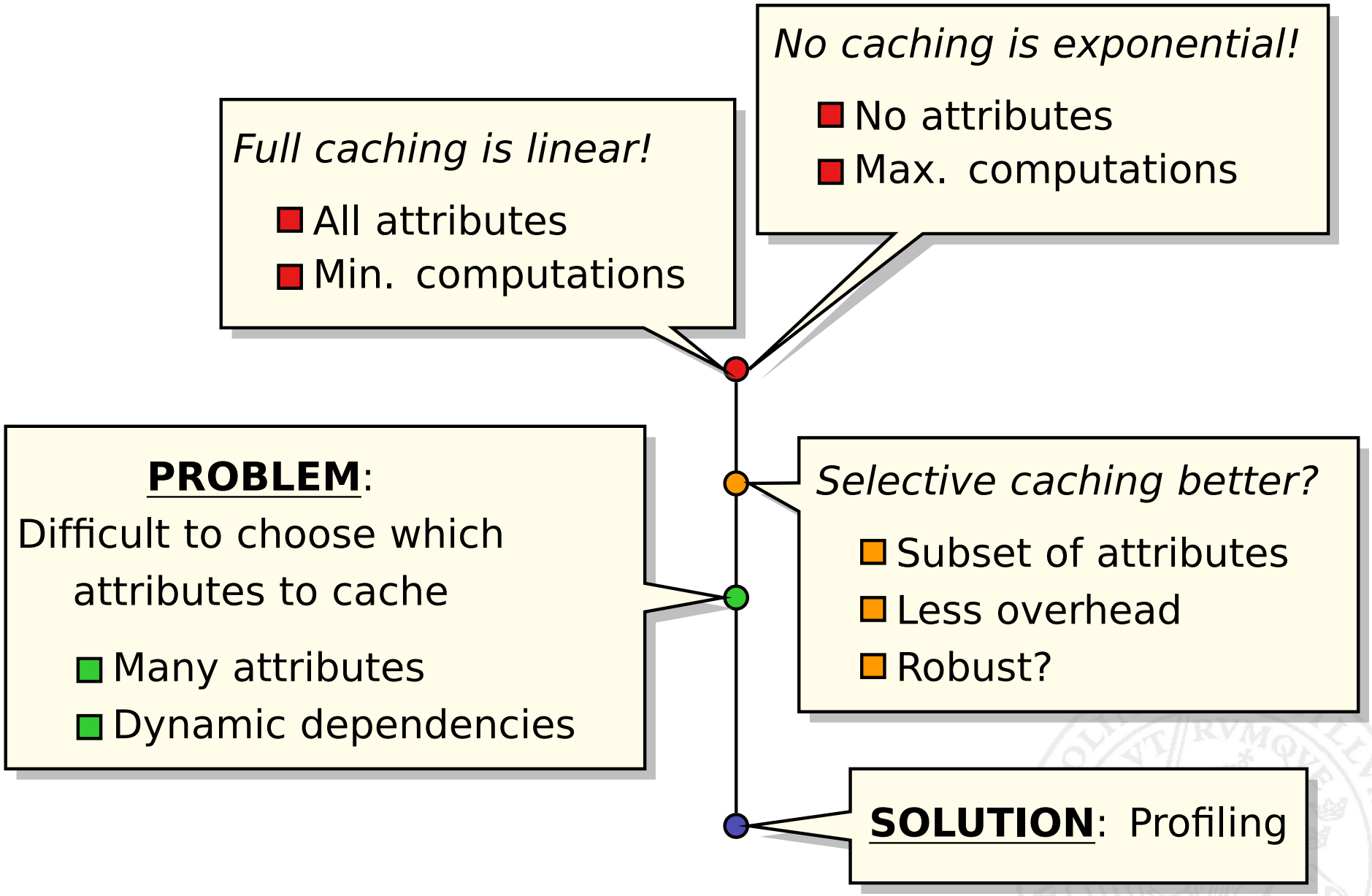
Caching of RAGs



Caching of RAGs



Caching of RAGs



Profiling

Attributes

$a = b.c \quad c = d \quad e = ..$

$b = d \quad d = e \quad f = ..$

Attribute Instances

$a : \{ a \quad , a \quad \}$

$b : \{ b \quad , b \quad \}$

$c : \{ c \quad \}$

$d : \{ d \quad \}$

$e : \{ e \quad \}$

$f : \quad \emptyset$

- *Ref. Attr. Grammar + Program*
- *Call Graph*
- *Total calls*
- *When to cache?*
  - *used?*
  - *calls > 1?*

Attribute Instance Call Graph

Profiling

Attributes

$a = b.c \quad c = d \quad e = ..$

$b = d \quad d = e \quad f = ..$

- *Ref. Attr. Grammar + Program*
- *Call Graph*
- *Total calls*
- *When to cache?*
  - *used?*
  - *calls > 1?*

Attribute Instances

$a : \{ a \quad , a \quad \}$

$b : \{ b \quad , b \quad \}$

$c : \{ c \quad \}$

$d : \{ d \quad \}$

$e : \{ e \quad \}$

$f : \quad \emptyset$

Attribute Instance Call Graph

Profiling

Attributes

$a = b.c \quad c = d \quad e = ..$

$b = d \quad d = e \quad f = ..$

Attribute Instances

$a : \{ a_I , a_{II} \}$

$b : \{ b_I , b_{II} \}$

$c : \{ c_I \}$

$d : \{ d_I \}$

$e : \{ e_I \}$

$f : \emptyset$

- *Ref. Attr. Grammar + Program*
- *Call Graph*
- *Total calls*
- *When to cache?*
  - *used?*
  - *calls > 1?*

Attribute Instance Call Graph

Profiling

Attributes

$a = b.c \quad c = d \quad e = ..$

$b = d \quad d = e \quad f = ..$

Attribute Instances

$a : \{ a_I , a_{II} \}$

$b : \{ b_I , b_{II} \}$

$c : \{ c_I \}$

$d : \{ d_I \}$

$e : \{ e_I \}$

$f : \emptyset$

- *Ref. Attr. Grammar + Program*
- *Call Graph*
- *Total calls*
- *When to cache?*
  - *used?*
  - *calls > 1?*

Attribute Instance Call Graph

Profiling

Attributes

$a = b.c \quad c = d \quad e = ..$

$b = d \quad d = e \quad f = ..$

- *Ref. Attr. Grammar + Program*
- *Call Graph*
  - *Total calls*
  - *When to cache?*
    - *used?*
    - *calls > 1?*

Attribute Instances

$a : \{ a_I , a_{II} \}$

$b : \{ b_I , b_{II} \}$

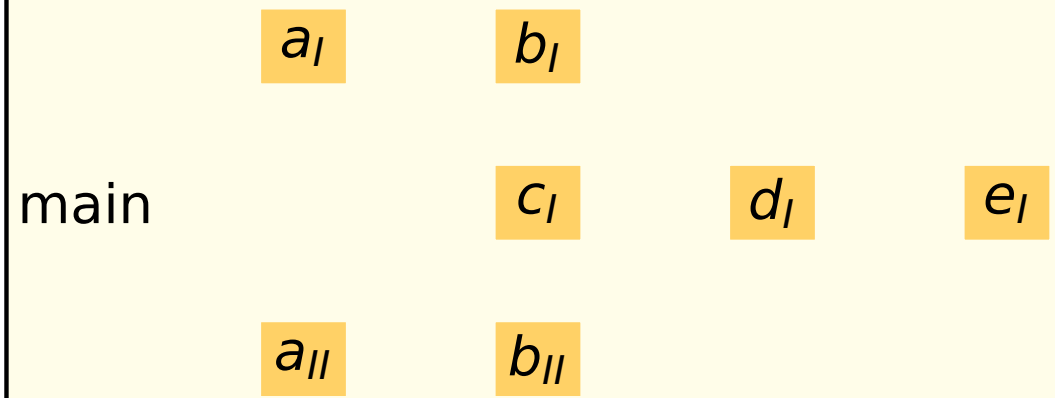
$c : \{ c_I \}$

$d : \{ d_I \}$

$e : \{ e_I \}$

$f : \emptyset$

Attribute Instance Call Graph



Profiling

Attributes

$a = b.c \quad c = d \quad e = ..$

$b = d \quad d = e \quad f = ..$

Attribute Instances

$a : \{ a_I , a_{II} \}$

$b : \{ b_I , b_{II} \}$

$c : \{ c_I \}$

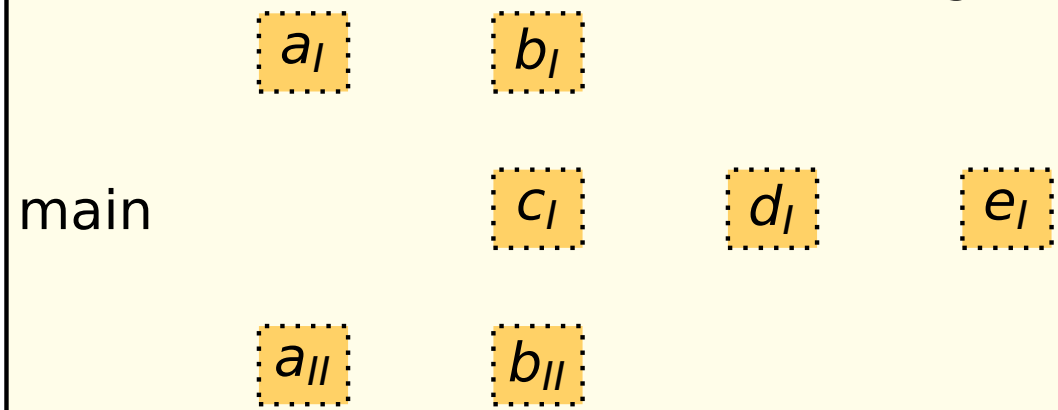
$d : \{ d_I \}$

$e : \{ e_I \}$

$f : \emptyset$

- *Ref. Attr. Grammar + Program*
- *Call Graph*
- *Total calls*
- *When to cache?*
  - *used?*
  - *calls > 1?*

Attribute Instance Call Graph
(no caching)



Profiling

Attributes

$a = b.c \quad c = d \quad e = ..$

$b = d \quad d = e \quad f = ..$

- *Ref. Attr. Grammar + Program*
- *Call Graph*
- *Total calls*
- *When to cache?*
  - *used?*
  - *calls > 1?*

Attribute Instances

$a : \{ a_I , a_{II} \}$

$b : \{ b_I , b_{II} \}$

$c : \{ c_I \}$

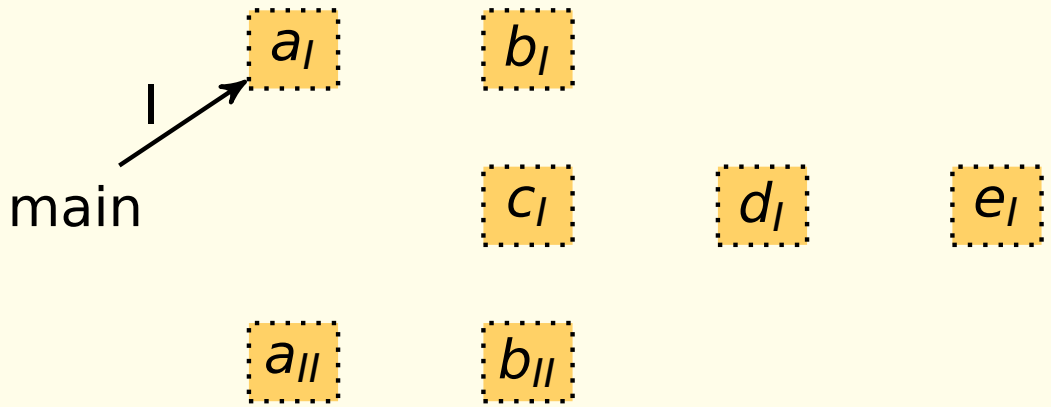
$d : \{ d_I \}$

$e : \{ e_I \}$

$f : \emptyset$

Attribute Instance Call Graph

(no caching)



Profiling

Attributes

$a = b.c \quad c = d \quad e = ..$

$b = d \quad d = e \quad f = ..$

Attribute Instances

$a : \{ a_I , a_{II} \}$

$b : \{ b_I , b_{II} \}$

$c : \{ c_I \}$

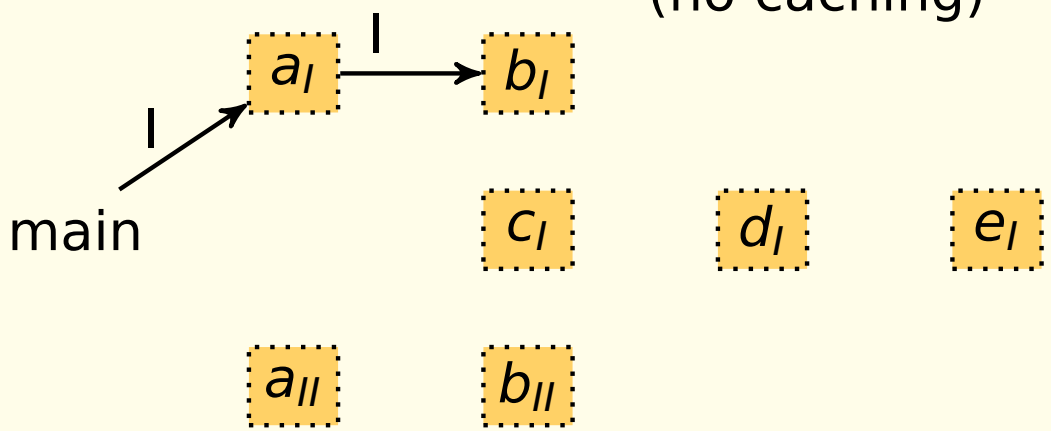
$d : \{ d_I \}$

$e : \{ e_I \}$

$f : \emptyset$

- *Ref. Attr. Grammar + Program*
- *Call Graph*
- *Total calls*
- *When to cache?*
  - *used?*
  - *calls > 1?*

Attribute Instance Call Graph
(no caching)



Profiling

Attributes

$a = b.c \quad c = d \quad e = ..$

$b = d \quad d = e \quad f = ..$

Attribute Instances

$a : \{ a_I , a_{II} \}$

$b : \{ b_I , b_{II} \}$

$c : \{ c_I \}$

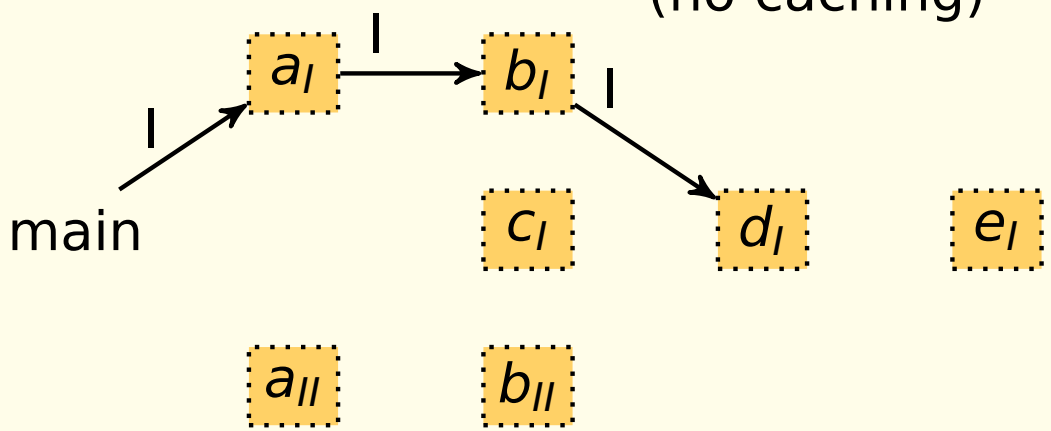
$d : \{ d_I \}$

$e : \{ e_I \}$

$f : \emptyset$

- *Ref. Attr. Grammar + Program*
- *Call Graph*
- *Total calls*
- *When to cache?*
  - *used?*
  - *calls > 1?*

Attribute Instance Call Graph
(no caching)



Profiling

Attributes

$a = b.c \quad c = d \quad e = ..$

$b = d \quad d = e \quad f = ..$

- *Ref. Attr. Grammar + Program*
- *Call Graph*
- *Total calls*
- *When to cache?*
  - *used?*
  - *calls > 1?*

Attribute Instances

$a : \{ a_I , a_{II} \}$

$b : \{ b_I , b_{II} \}$

$c : \{ c_I \}$

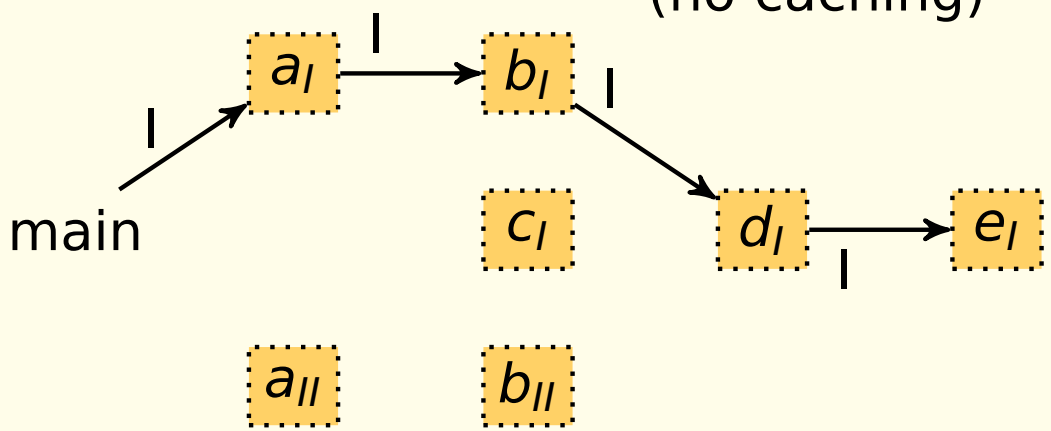
$d : \{ d_I \}$

$e : \{ e_I \}$

$f : \emptyset$

Attribute Instance Call Graph

(no caching)



Profiling

Attributes

$a = b.c \quad c = d \quad e = ..$

$b = d \quad d = e \quad f = ..$

Attribute Instances

$a : \{ a_I , a_{II} \}$

$b : \{ b_I , b_{II} \}$

$c : \{ c_I \}$

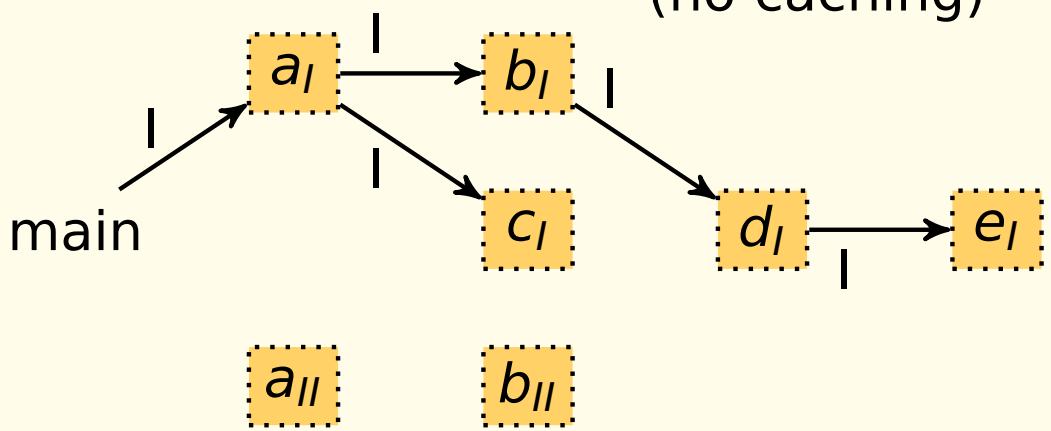
$d : \{ d_I \}$

$e : \{ e_I \}$

$f : \emptyset$

- *Ref. Attr. Grammar + Program*
- *Call Graph*
- *Total calls*
- *When to cache?*
  - *used?*
  - *calls > 1?*

Attribute Instance Call Graph
(no caching)



Profiling

Attributes

$a = b.c \quad c = d \quad e = ..$

$b = d \quad d = e \quad f = ..$

- *Ref. Attr. Grammar + Program*
- *Call Graph*
- *Total calls*
- *When to cache?*
  - *used?*
  - *calls > 1?*

Attribute Instances

$a : \{ a_I , a_{II} \}$

$b : \{ b_I , b_{II} \}$

$c : \{ c_I \}$

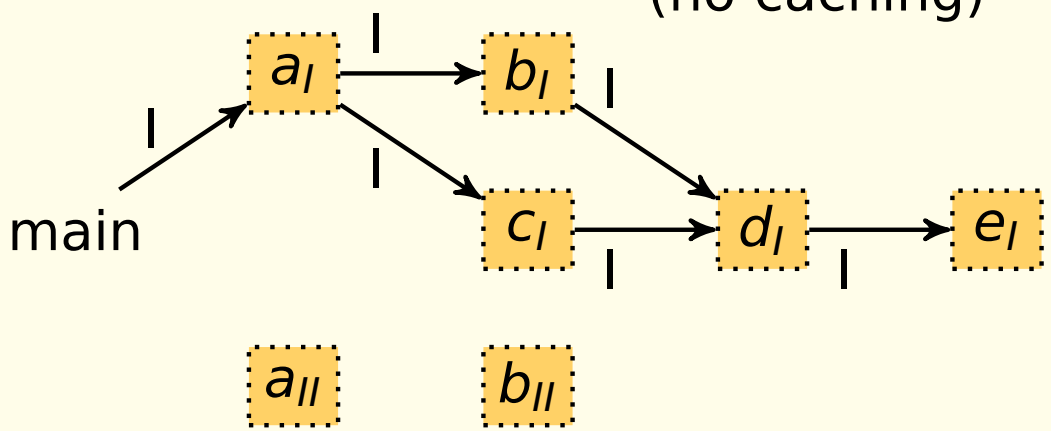
$d : \{ d_I \}$

$e : \{ e_I \}$

$f : \emptyset$

Attribute Instance Call Graph

(no caching)



Profiling

Attributes

$a = b.c \quad c = d \quad e = ..$

$b = d \quad d = e \quad f = ..$

- *Ref. Attr. Grammar + Program*
- *Call Graph*
- *Total calls*
- *When to cache?*
  - *used?*
  - *calls > 1?*

Attribute Instances

$a : \{ a_I , a_{II} \}$

$b : \{ b_I , b_{II} \}$

$c : \{ c_I \}$

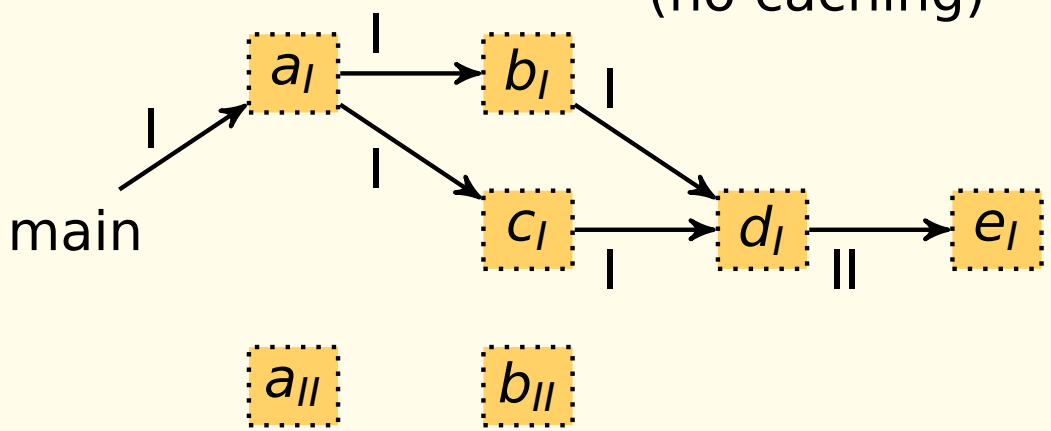
$d : \{ d_I \}$

$e : \{ e_I \}$

$f : \emptyset$

Attribute Instance Call Graph

(no caching)



Profiling

Attributes

$a = b.c \quad c = d \quad e = ..$

$b = d \quad d = e \quad f = ..$

Attribute Instances

$a : \{ a_I , a_{II} \}$

$b : \{ b_I , b_{II} \}$

$c : \{ c_I \}$

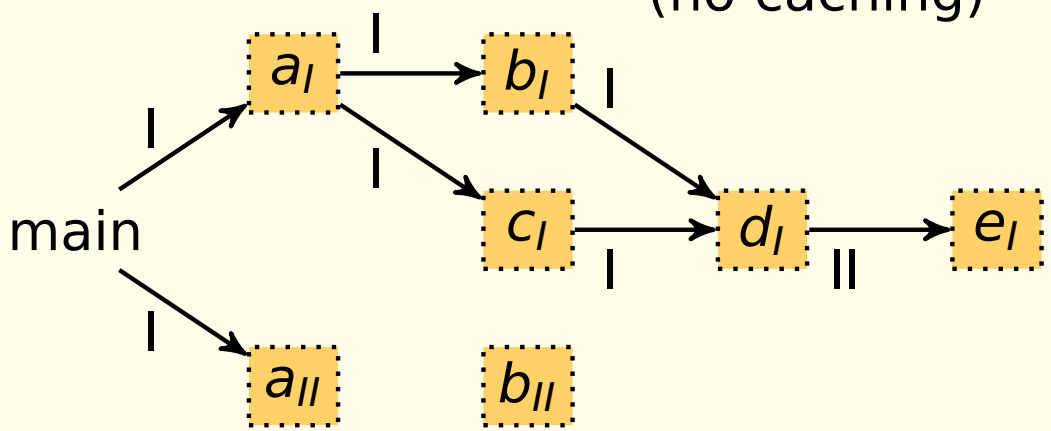
$d : \{ d_I \}$

$e : \{ e_I \}$

$f : \emptyset$

- *Ref. Attr. Grammar + Program*
- *Call Graph*
- *Total calls*
- *When to cache?*
  - *used?*
  - *calls > 1?*

Attribute Instance Call Graph
(no caching)



Profiling

Attributes

$a = b.c \quad c = d \quad e = ..$

$b = d \quad d = e \quad f = ..$

Attribute Instances

$a : \{ a_I , a_{II} \}$

$b : \{ b_I , b_{II} \}$

$c : \{ c_I \}$

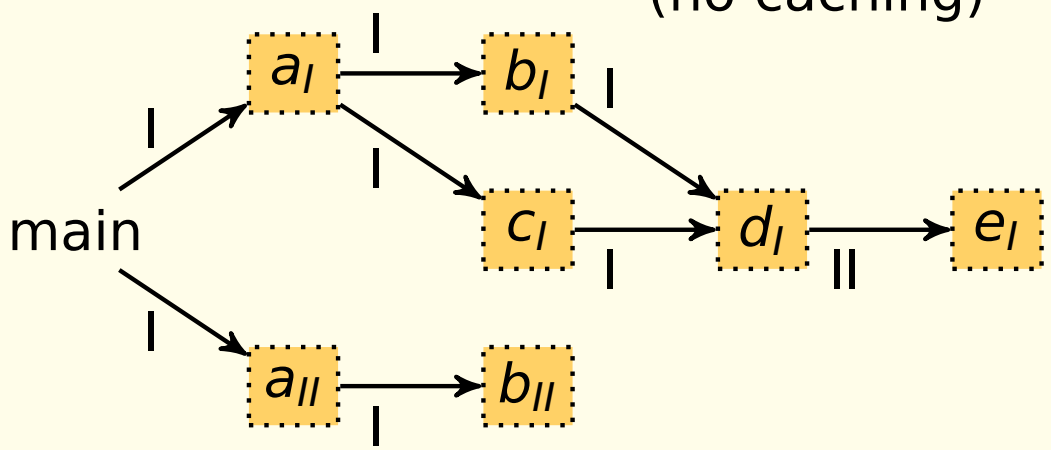
$d : \{ d_I \}$

$e : \{ e_I \}$

$f : \emptyset$

- *Ref. Attr. Grammar + Program*
- *Call Graph*
- *Total calls*
- *When to cache?*
  - *used?*
  - *calls > 1?*

Attribute Instance Call Graph
(no caching)



Profiling

Attributes

$a = b.c \quad c = d \quad e = ..$

$b = d \quad d = e \quad f = ..$

- *Ref. Attr. Grammar + Program*
- *Call Graph*
- *Total calls*
- *When to cache?*
  - *used?*
  - *calls > 1?*

Attribute Instances

$a : \{ a_I , a_{II} \}$

$b : \{ b_I , b_{II} \}$

$c : \{ c_I \}$

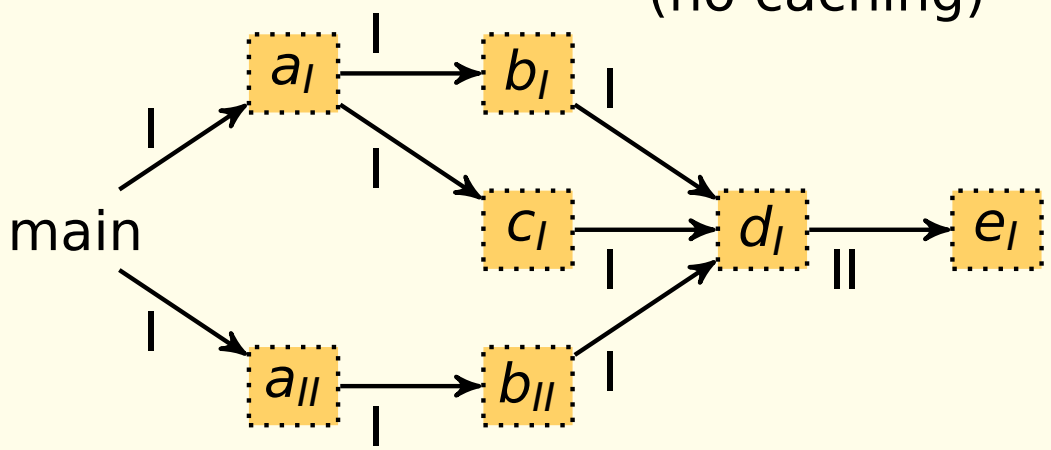
$d : \{ d_I \}$

$e : \{ e_I \}$

$f : \emptyset$

Attribute Instance Call Graph

(no caching)



Profiling

Attributes

$a = b.c \quad c = d \quad e = ..$

$b = d \quad d = e \quad f = ..$

Attribute Instances

$a : \{ a_I , a_{II} \}$

$b : \{ b_I , b_{II} \}$

$c : \{ c_I \}$

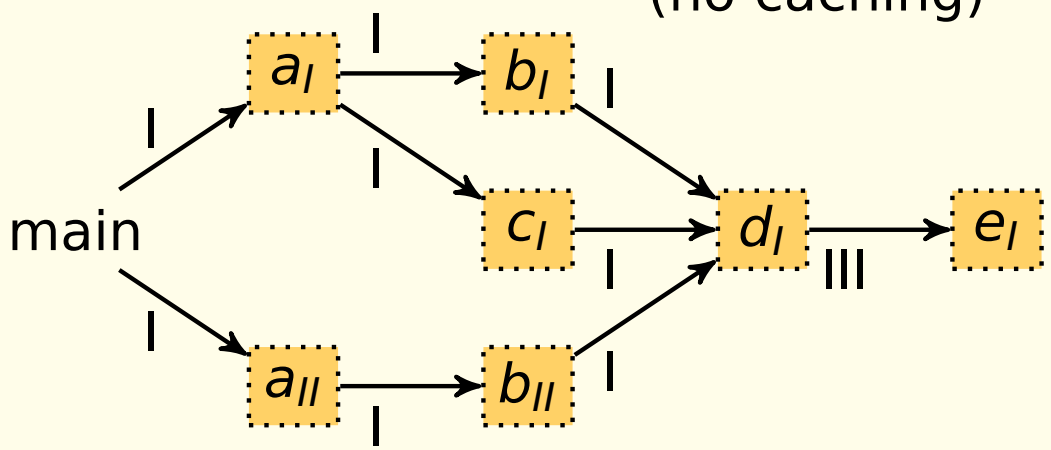
$d : \{ d_I \}$

$e : \{ e_I \}$

$f : \emptyset$

- *Ref. Attr. Grammar + Program*
- *Call Graph*
- *Total calls*
- *When to cache?*
  - *used?*
  - *calls > 1?*

Attribute Instance Call Graph
(no caching)



Profiling

Attributes

$a = b.c \quad c = d \quad e = ..$

$b = d \quad d = e \quad f = ..$

- *Ref. Attr. Grammar + Program*
- *Call Graph*
- *Total calls*
- *When to cache?*
  - *used?*
  - *calls > 1?*

Attribute Instances

$a : \{ a_I , a_{II} \}$

$b : \{ b_I , b_{II} \}$

$c : \{ c_I \}$

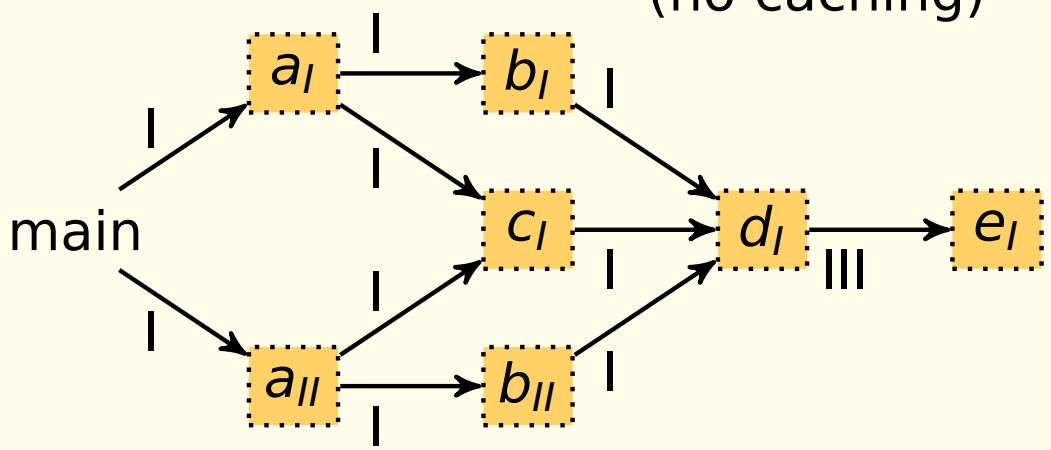
$d : \{ d_I \}$

$e : \{ e_I \}$

$f : \emptyset$

Attribute Instance Call Graph

(no caching)



Profiling

Attributes

$a = b.c \quad c = d \quad e = ..$

$b = d \quad d = e \quad f = ..$

Attribute Instances

$a : \{ a_I , a_{II} \}$

$b : \{ b_I , b_{II} \}$

$c : \{ c_I \}$

$d : \{ d_I \}$

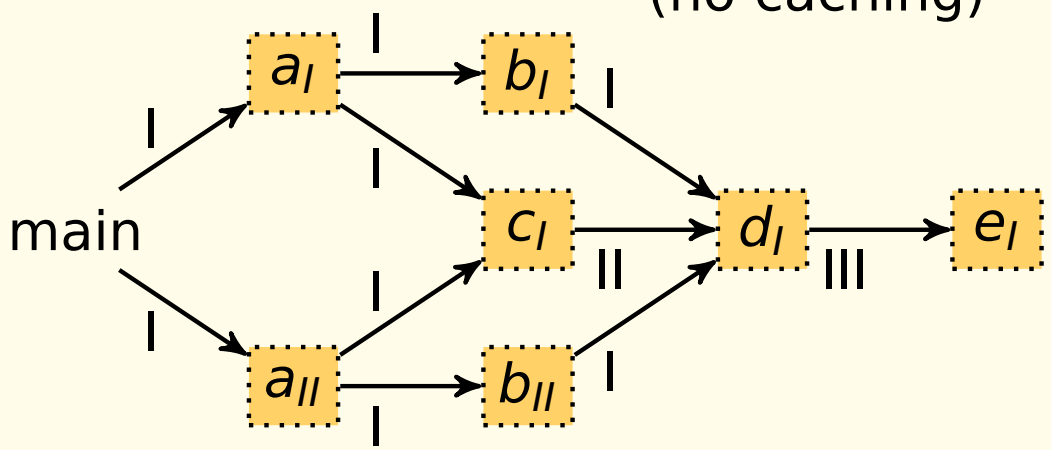
$e : \{ e_I \}$

$f : \emptyset$

- *Ref. Attr. Grammar + Program*
- *Call Graph*
- *Total calls*
- *When to cache?*
  - *used?*
  - *calls > 1?*

Attribute Instance Call Graph

(no caching)



Profiling

Attributes

$a = b.c \quad c = d \quad e = ..$

$b = d \quad d = e \quad f = ..$

Attribute Instances

$a : \{ a_I , a_{II} \}$

$b : \{ b_I , b_{II} \}$

$c : \{ c_I \}$

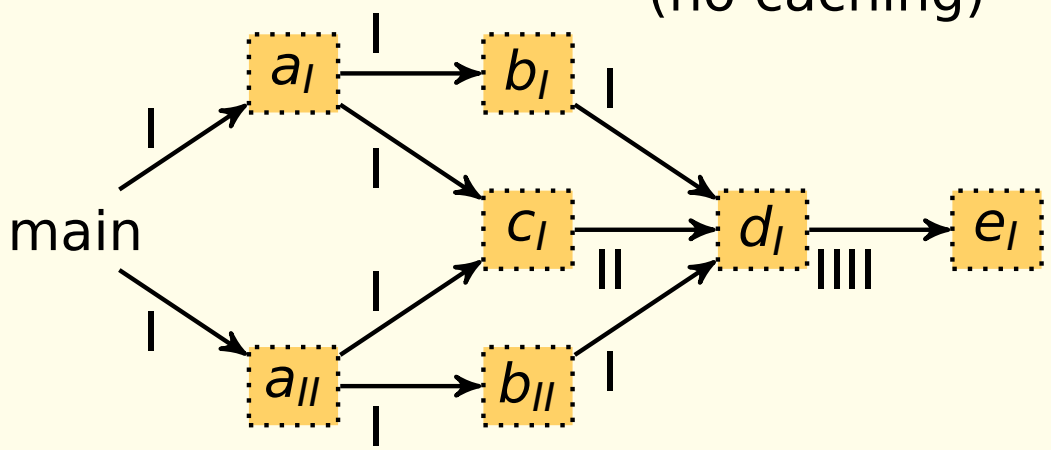
$d : \{ d_I \}$

$e : \{ e_I \}$

$f : \emptyset$

- *Ref. Attr. Grammar + Program*
- *Call Graph*
- *Total calls*
- *When to cache?*
  - *used?*
  - *calls > 1?*

Attribute Instance Call Graph
(no caching)



Profiling

Attributes

$a = b.c \quad c = d \quad e = ..$

$b = d \quad d = e \quad f = ..$

- *Ref. Attr. Grammar + Program*
- *Call Graph*
- *Total calls*
- *When to cache?*
  - *used?*
  - *calls > 1?*

Attribute Instances

$a : \{ a_I , a_{II} \}$

$b : \{ b_I , b_{II} \}$

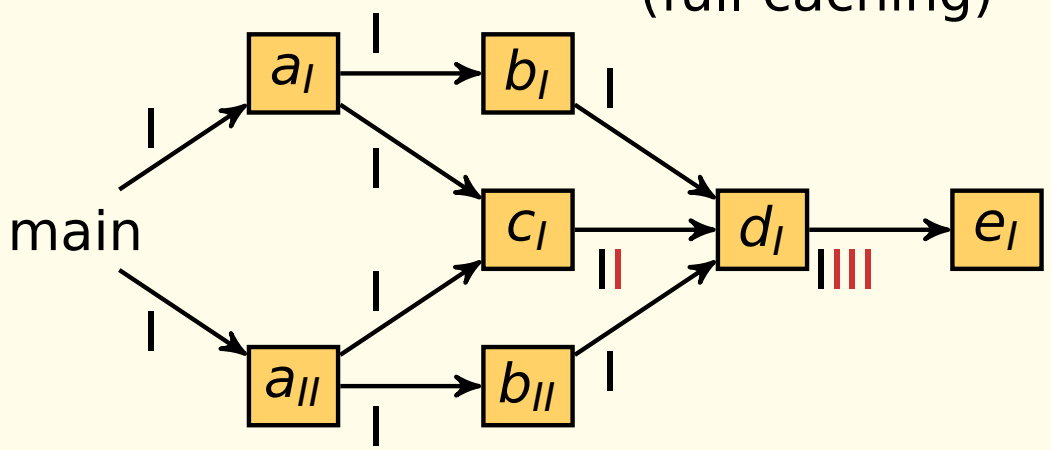
$c : \{ c_I \}$

$d : \{ d_I \}$

$e : \{ e_I \}$

$f : \emptyset$

Attribute Instance Call Graph
(full caching)



Profiling

Attributes

$a = b.c \quad c = d \quad e = ..$

$b = d \quad d = e \quad f = ..$

Attribute Instances

$a : \{ a_I , a_{II} \}$

$b : \{ b_I , b_{II} \}$

$c : \{ c_I \}$

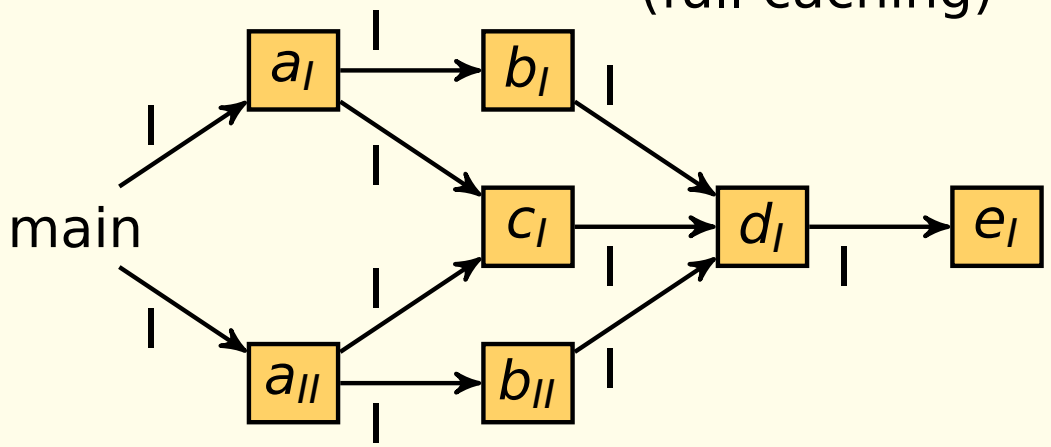
$d : \{ d_I \}$

$e : \{ e_I \}$

$f : \emptyset$

- *Ref. Attr. Grammar + Program*
- *Call Graph*
- *Total calls*
- *When to cache?*
  - *used?*
  - *calls > 1?*

Attribute Instance Call Graph
(full caching)



Profiling

Attributes

$a = b.c \quad c = d \quad e = ..$

$b = d \quad d = e \quad f = ..$

- *Ref. Attr. Grammar + Program*
- *Call Graph*
- *Total calls*
- *When to cache?*
  - *used?*
  - *calls > 1?*

Attribute Instances

$a : \{ a_I , a_{II} \}$

$b : \{ b_I , b_{II} \}$

$c : \{ c_I \}$

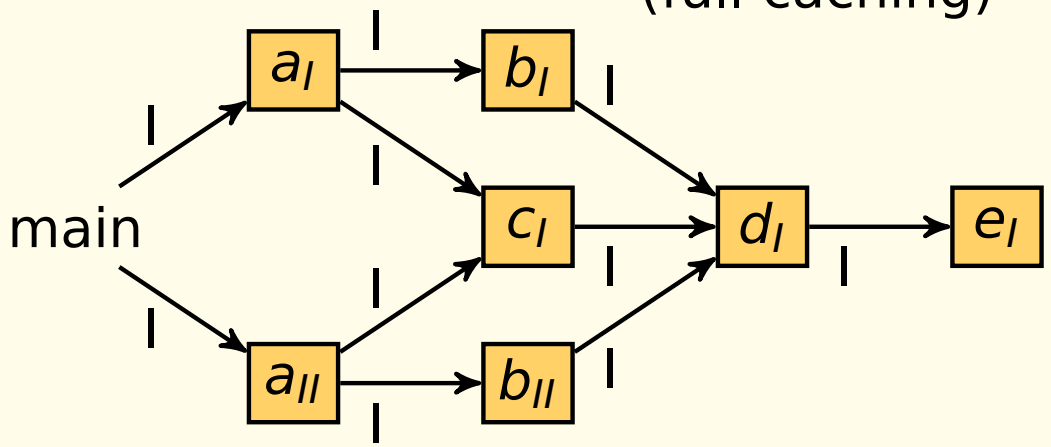
$d : \{ d_I \}$

$e : \{ e_I \}$

$f : \emptyset$

Attribute Instance Call Graph

(full caching)



Profiling

Attributes

$a = b.c \quad c = d \quad e = ..$

$b = d \quad d = e \quad f = ..$

- *Ref. Attr. Grammar + Program*
- *Call Graph*
- *Total calls*
- *When to cache?*
  - *used?*
  - *calls > 1?*

Attribute Instances

$a : \{ a_I^{1} , a_{II}^{1} \}$

$b : \{ b_I^{1} , b_{II}^{1} \}$

$c : \{ c_I^{2} \}$

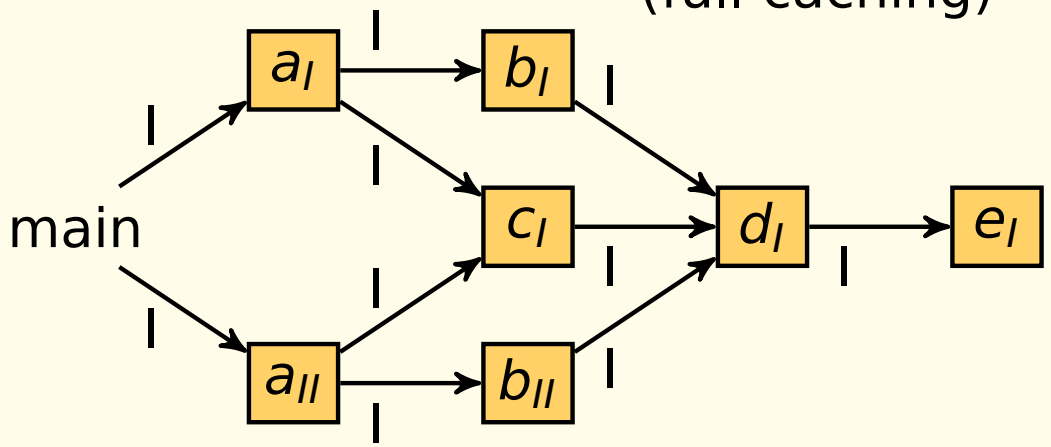
$d : \{ d_I^{3} \}$

$e : \{ e_I^{1} \}$

$f : \emptyset$

Attribute Instance Call Graph

(full caching)



Profiling

Attributes

$a = b.c \quad c = d \quad e = ..$

$b = d \quad d = e \quad f = ..$

- *Ref. Attr. Grammar + Program*
- *Call Graph*
- *Total calls*
- *When to cache?*
  - *used?*
  - *calls > 1?*

Attribute Instances

$a : \{ a_I^1 , a_{II}^1 \}$

$b : \{ b_I^1 , b_{II}^1 \}$

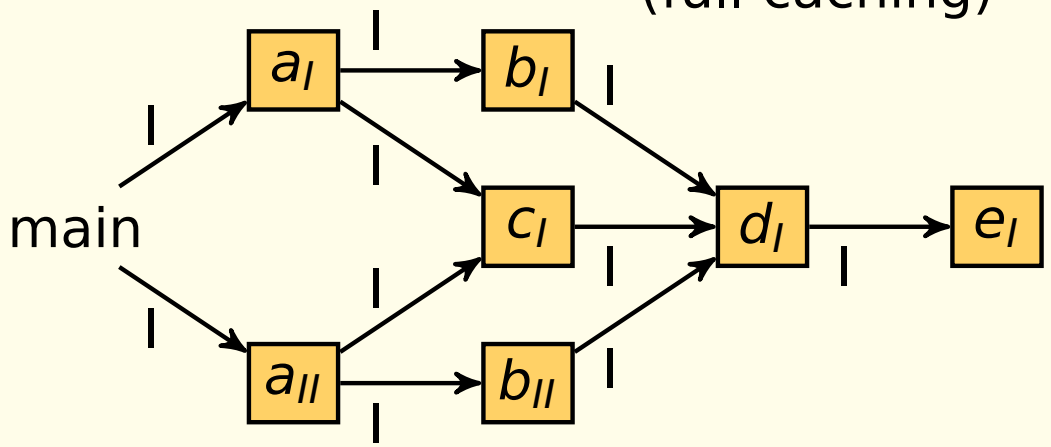
$c : \{ c_I^2 \}$

$d : \{ d_I^3 \}$

$e : \{ e_I^1 \}$

$f : \emptyset$

Attribute Instance Call Graph
(full caching)



Profiling

Attributes

$a = b.c \quad c = d \quad e = ..$

$b = d \quad d = e \quad f = ..$

- *Ref. Attr. Grammar + Program*
- *Call Graph*
- *Total calls*
- *When to cache?*
  - *used?*
  - *calls > 1?*

Attribute Instances

$a : \{ a_I^1 , a_{II}^1 \}$

$b : \{ b_I^1 , b_{II}^1 \}$

$c : \{ c_I^2 \}$

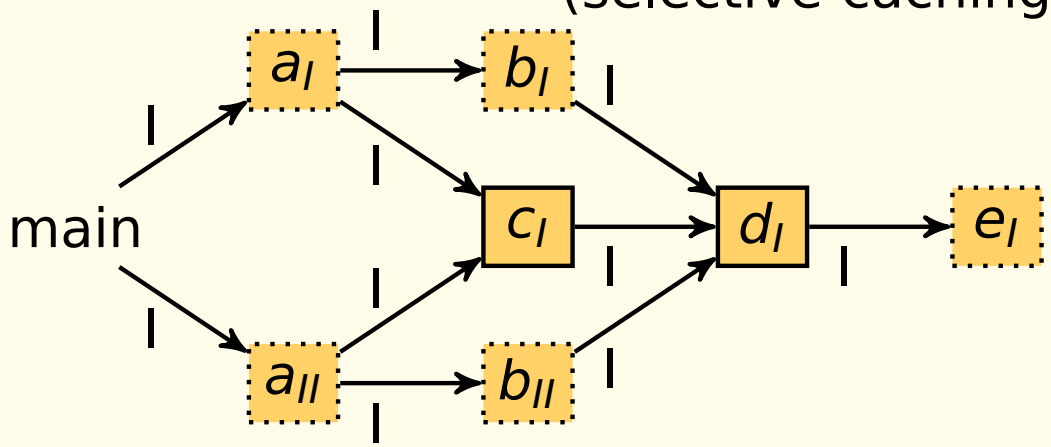
$d : \{ d_I^3 \}$

$e : \{ e_I^1 \}$

$f : \emptyset$

Attribute Instance Call Graph

(selective caching)



Attribute Sets

Static info.

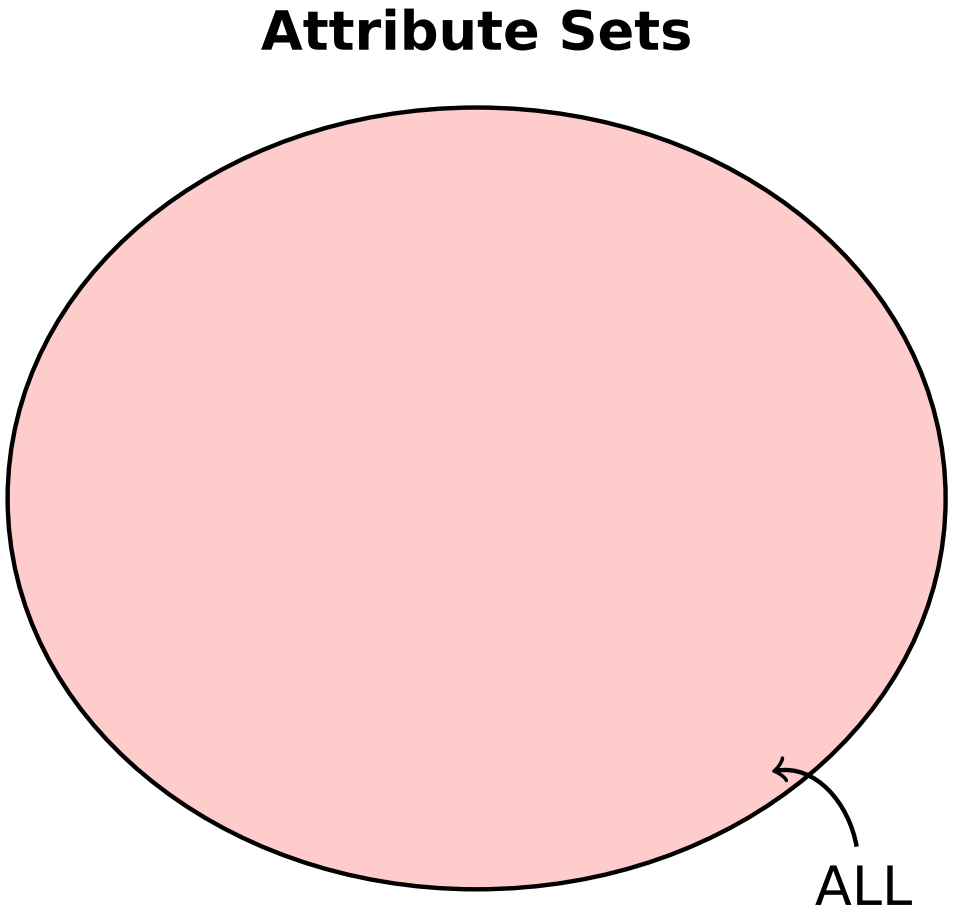
- *ALL*
- *PARAM, NONPARAM*
- *PRE*

Profiling info.

- *USED, UNUSED*
- *ONE, MANY*

Configurations

- *ALL - ONE*
- *USED - ONE*



Attribute Sets

Static info.

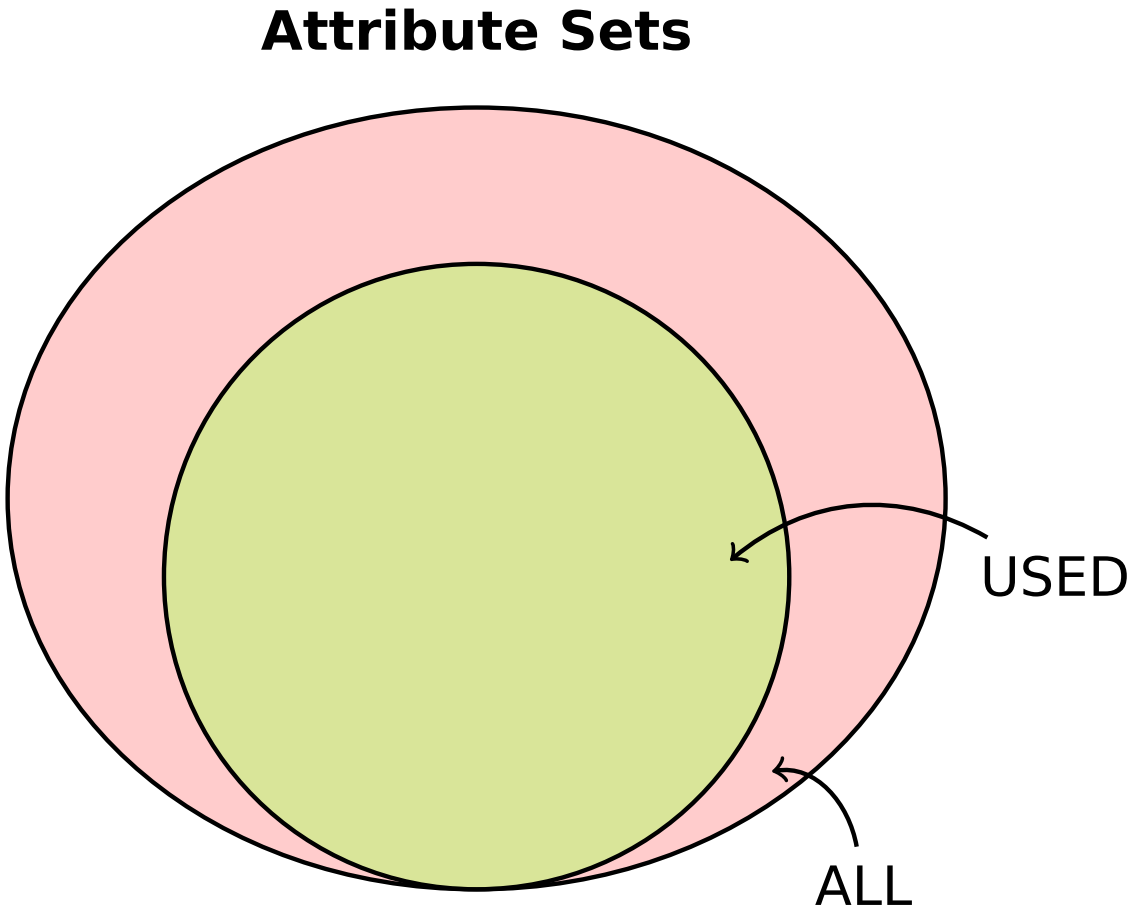
- *ALL*
- *PARAM, NONPARAM*
- *PRE*

Profiling info.

- *USED, UNUSED*
- *ONE, MANY*

Configurations

- *ALL - ONE*
- *USED - ONE*



Attribute Sets

Static info.

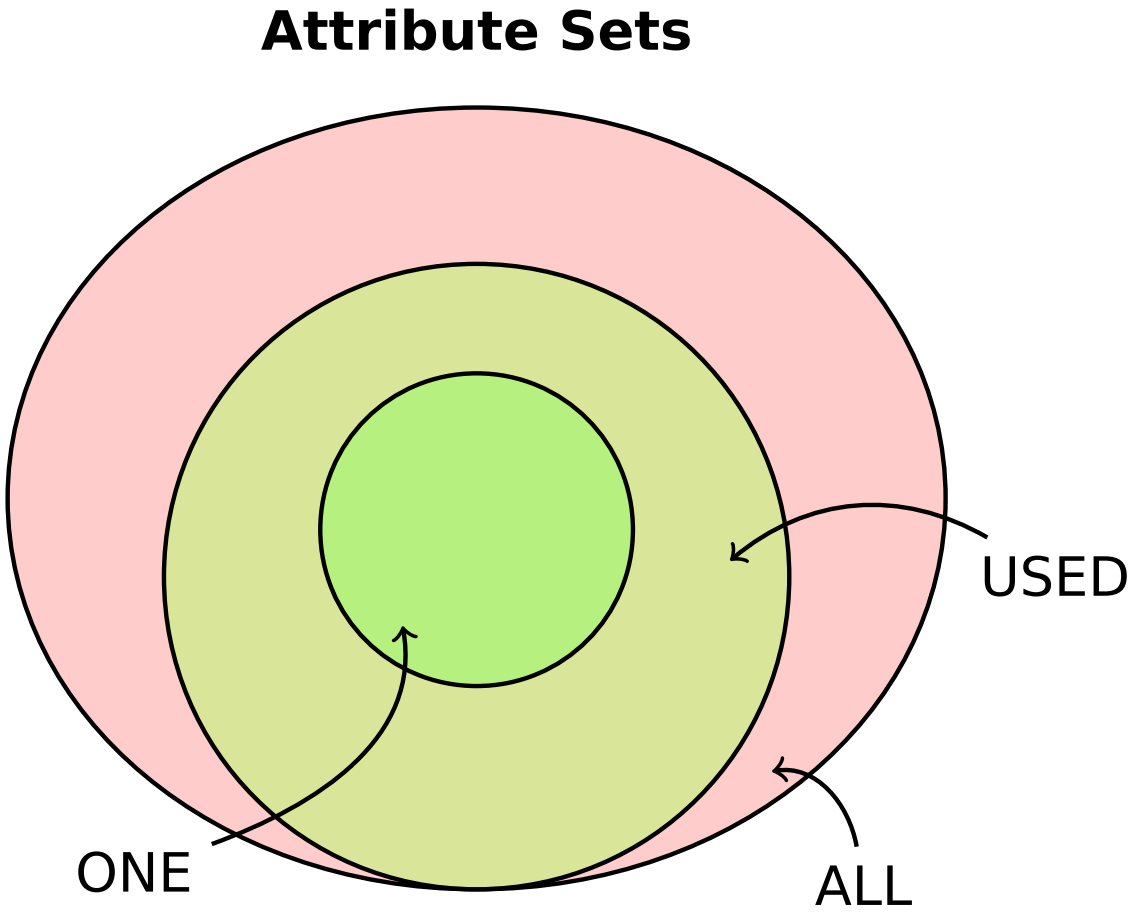
- *ALL*
- *PARAM, NONPARAM*
- *PRE*

Profiling info.

- *USED, UNUSED*
- *ONE, MANY*

Configurations

- *ALL - ONE*
- *USED - ONE*



Attribute Sets

Static info.

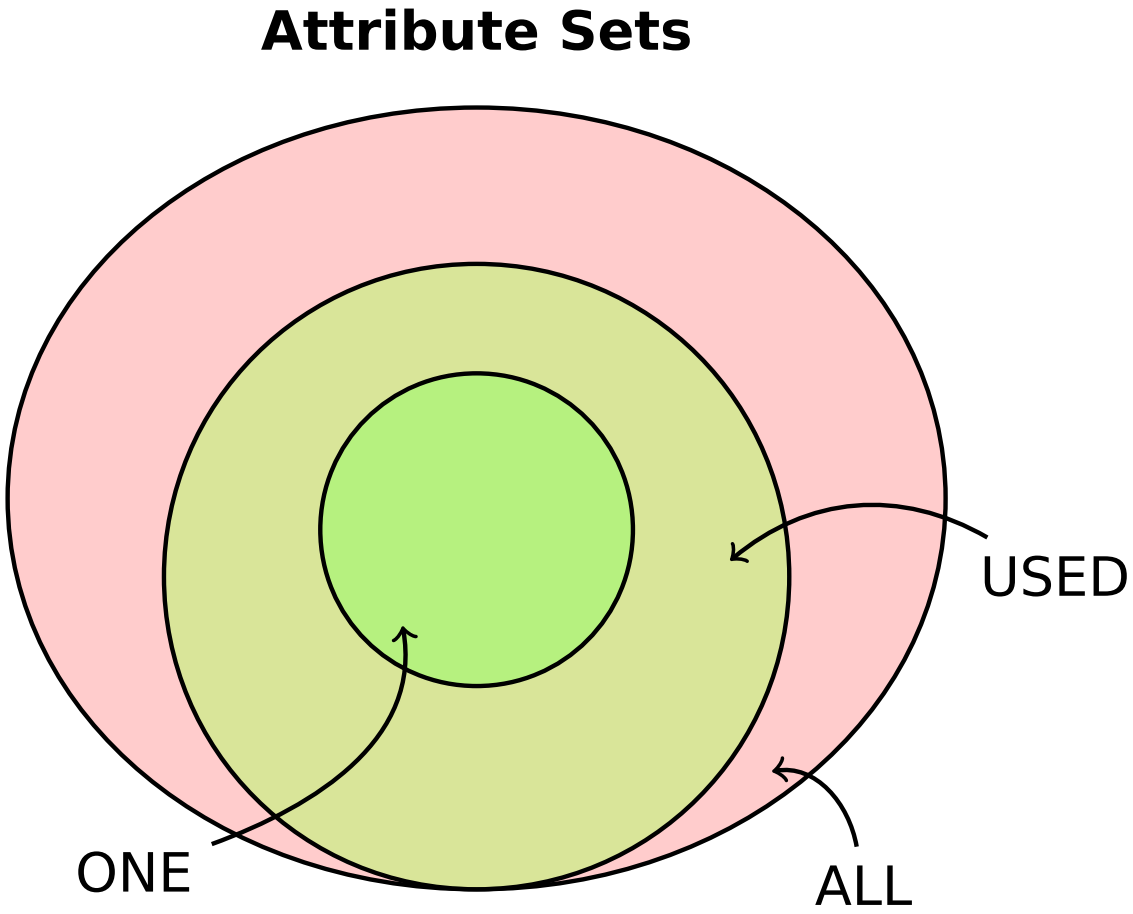
- *ALL*
- *PARAM, NONPARAM*
- *PRE*

Profiling info.

- *USED, UNUSED*
- *ONE, MANY*

Configurations

- *ALL - ONE*
- *USED - ONE*



Attribute Sets

Static info.

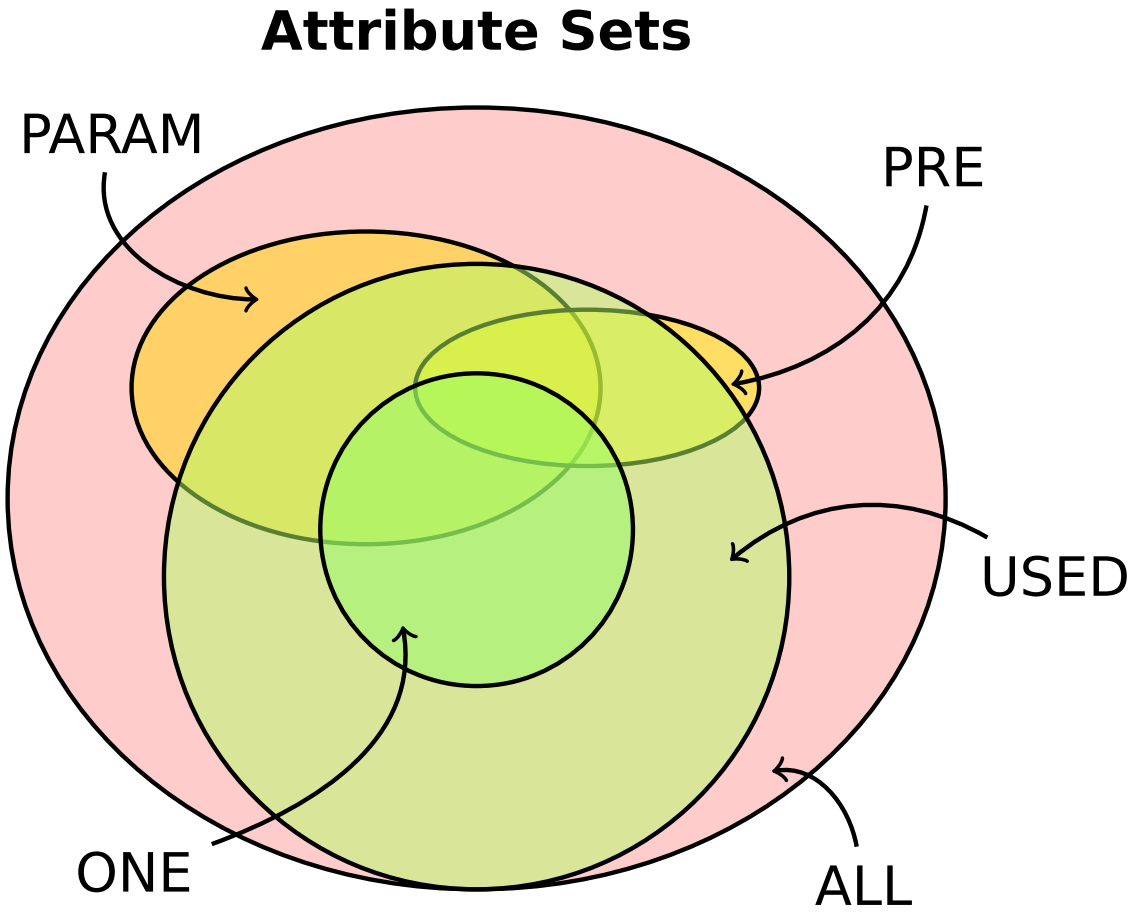
- *ALL*
- *PARAM, NONPARAM*
- *PRE*

Profiling info.

- *USED, UNUSED*
- *ONE, MANY*

Configurations

- *ALL - ONE*
- *USED - ONE*

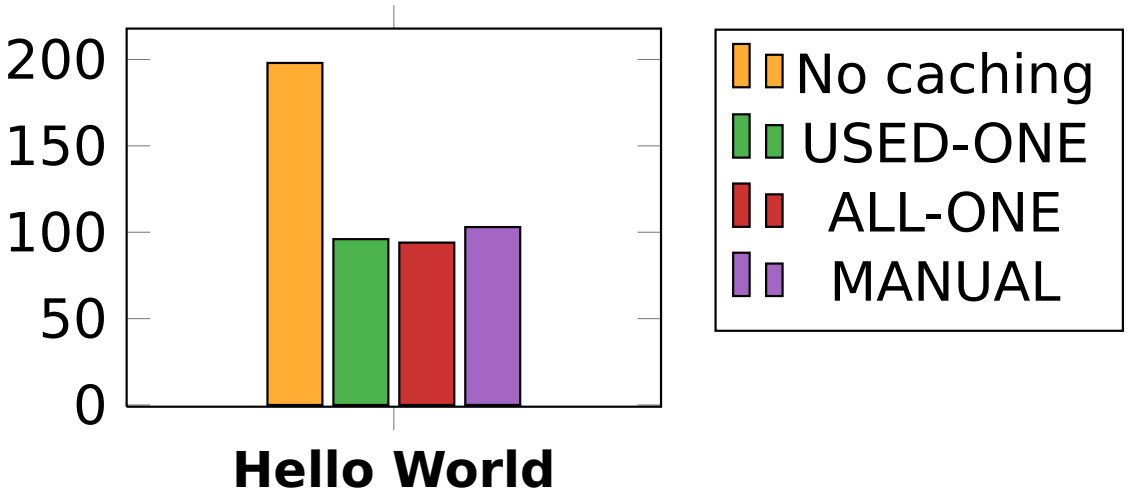


Evaluation

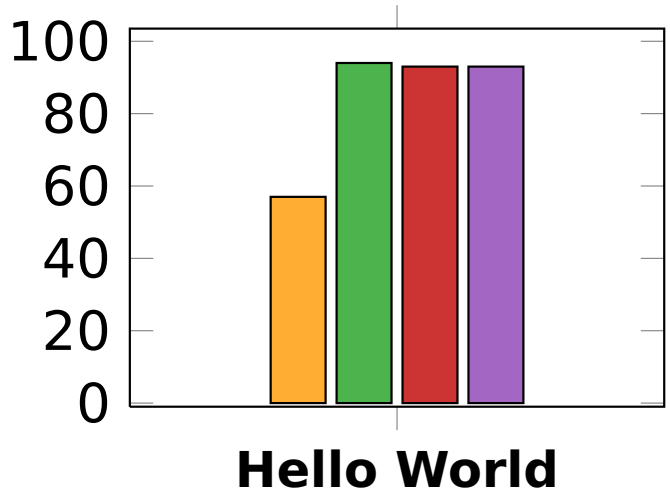
- **Two profiling approaches**: ALL-ONE, USED-ONE
  - Robustness? (slower than ALL?)
  - Performance? (time,memory)
- **A RAG-based Java compiler**: JastAddJ
  - |ALL|=787, Expert configuration (|MANUAL|=281)
  - **Profiling input**: Hello World, Jacks, DaCapo
  - **Test input**: Hello World, DaCapo
- **A set of experiments**:
  - A hello world / no caching, 29%
  - B profiling with a test suite (Jacks), 80%
  - C profiling with benchmarks (DaCapo), 61-81%, 84%
  - D both B and C

Results: Experiment A (No caching)

Execution Time (% of ALL)



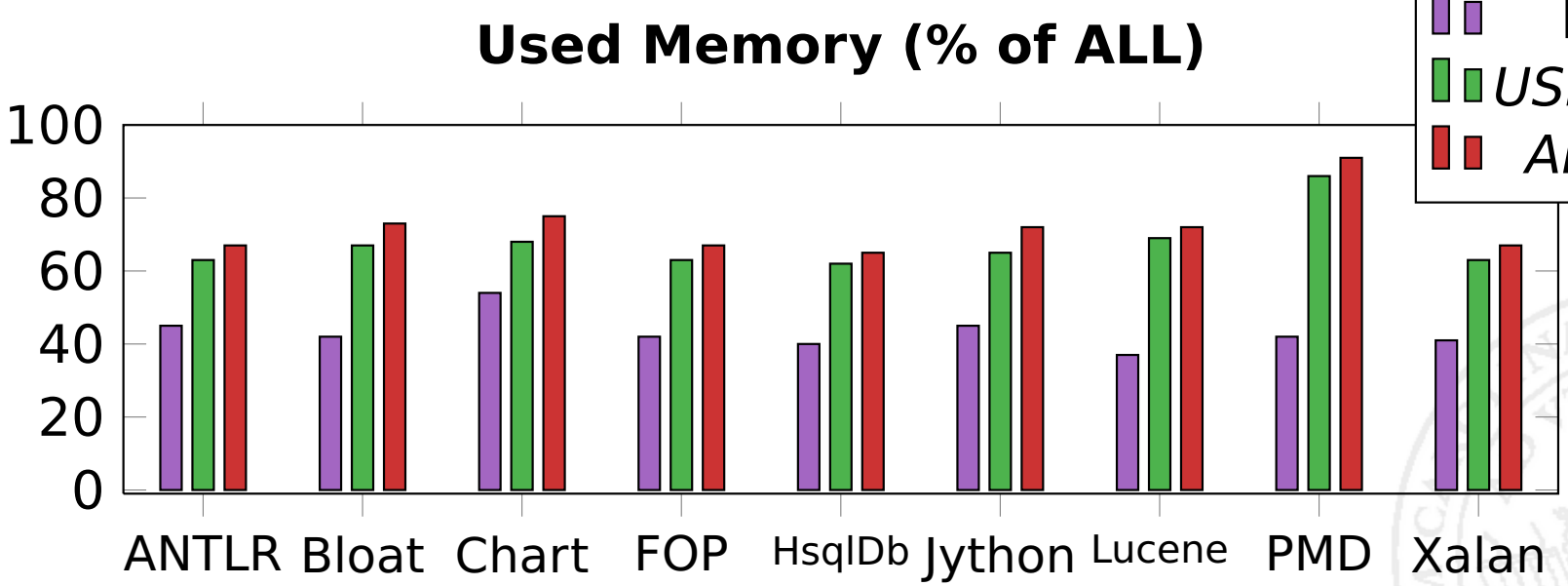
Used Memory (% of ALL)

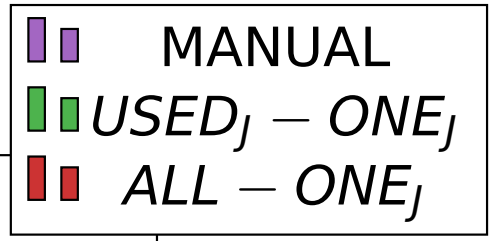


ALL: 47 ms, 15 Mb

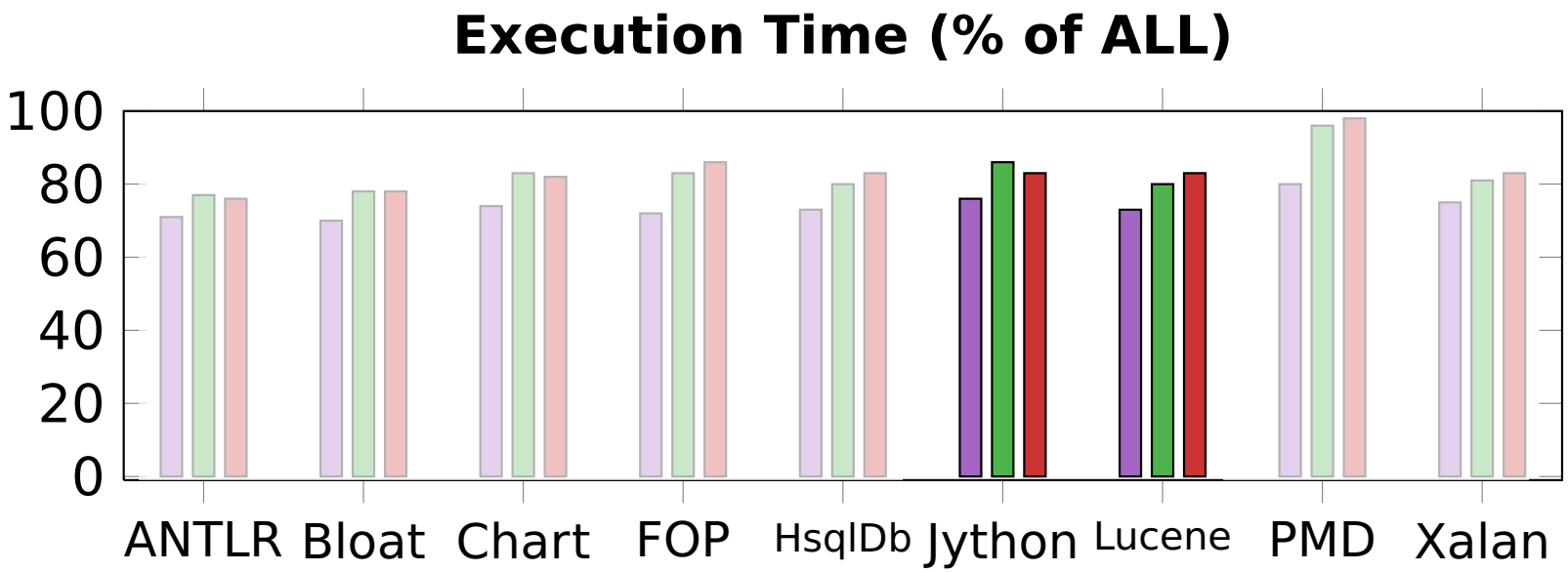
No caching: 93 ms, 8.6 Mb

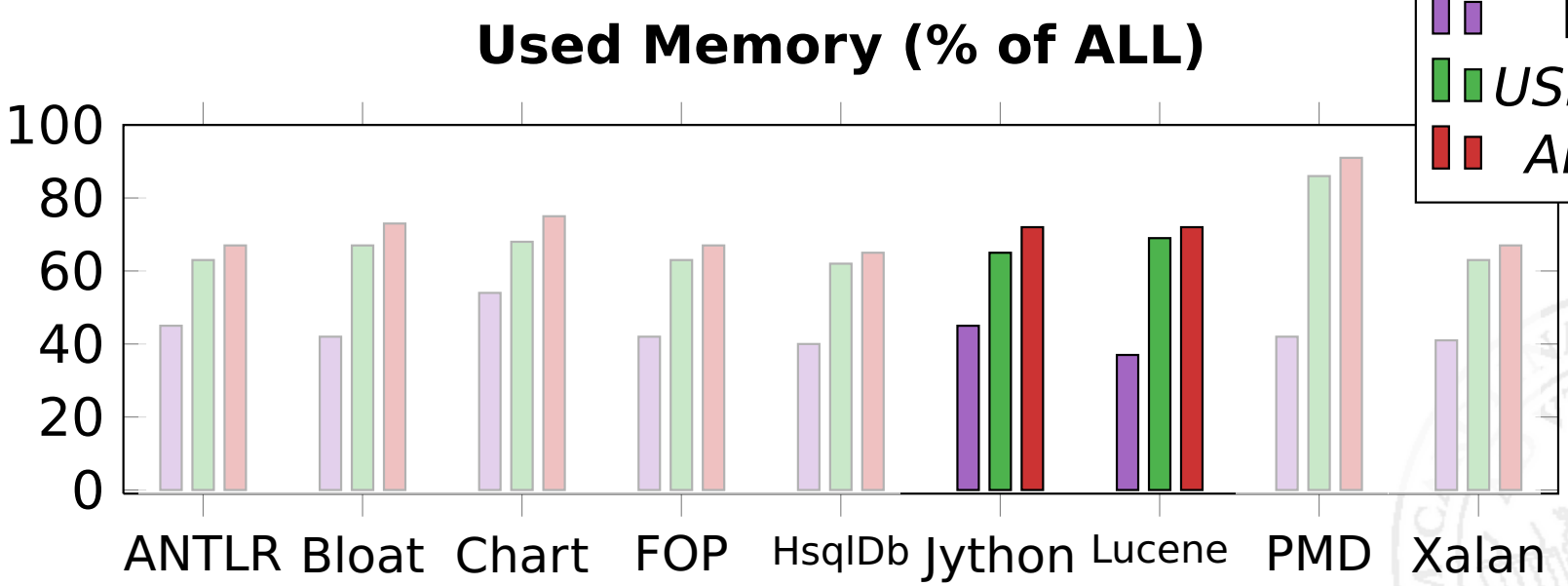
Results: Experiment B (J - Jacks test suite)

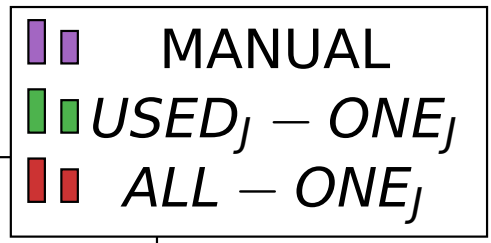




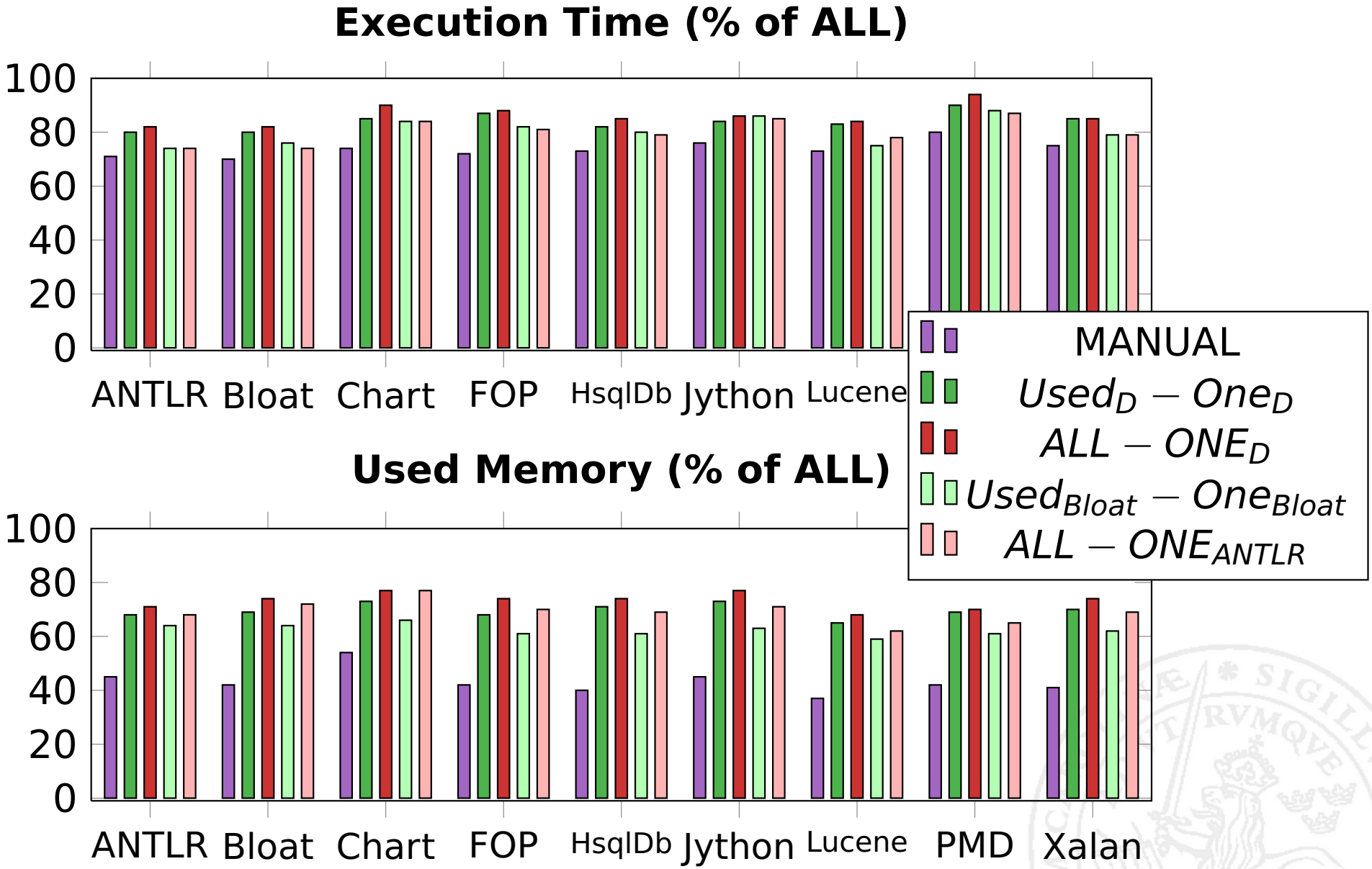
Results: Experiment B (J - Jacks test suite)



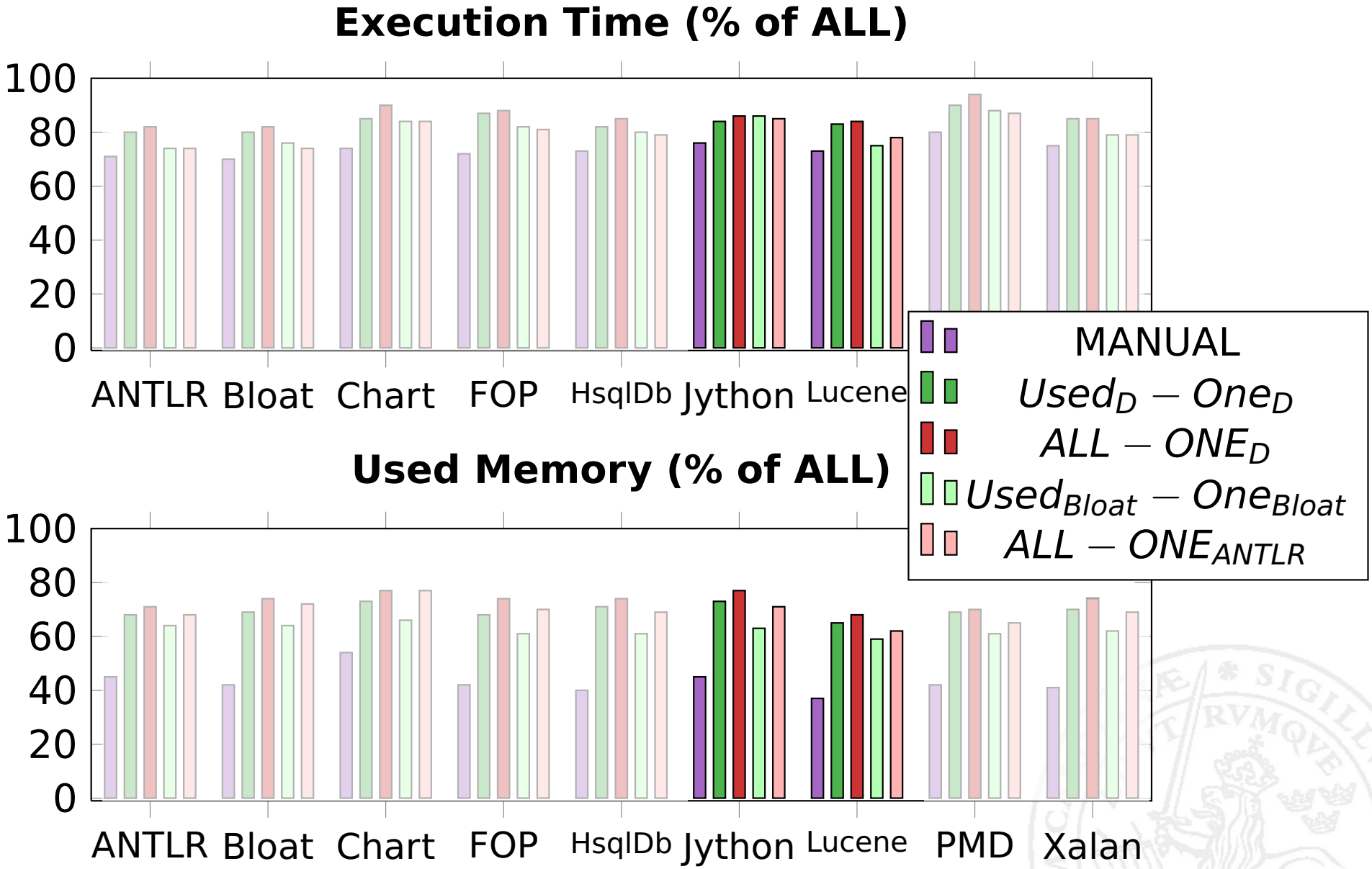




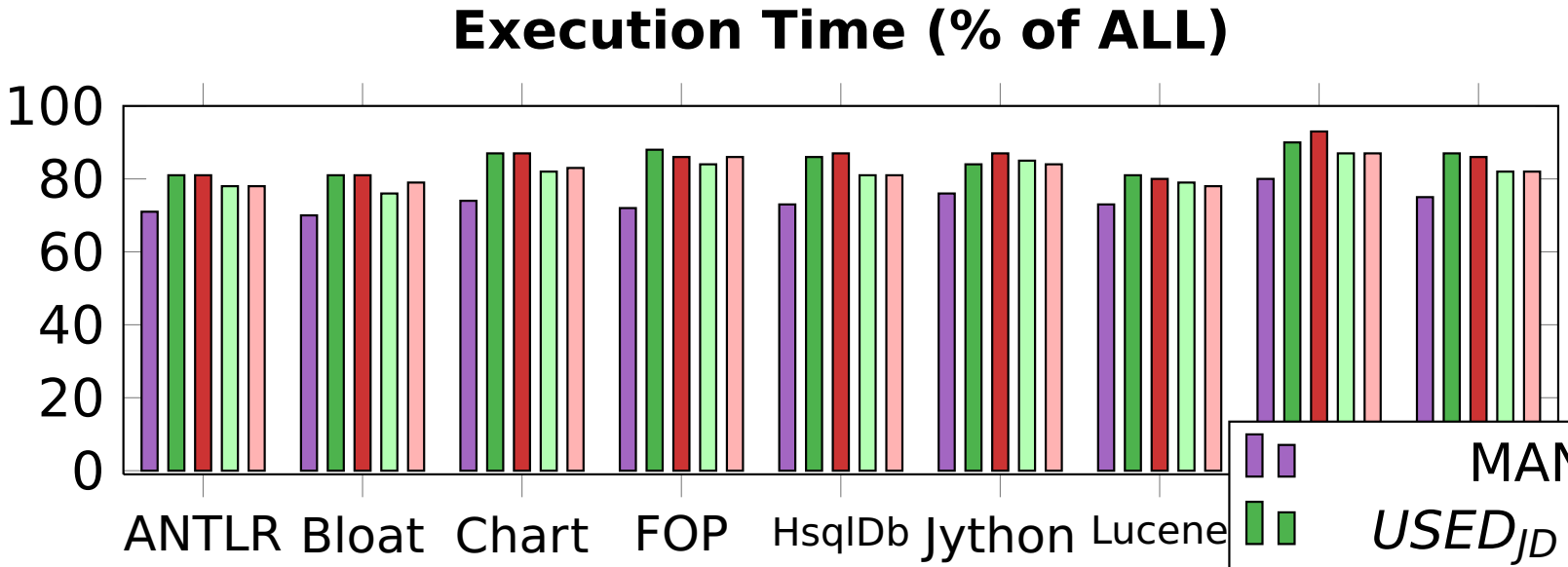
Results: Experiment C (D - DaCapo benchmarks)

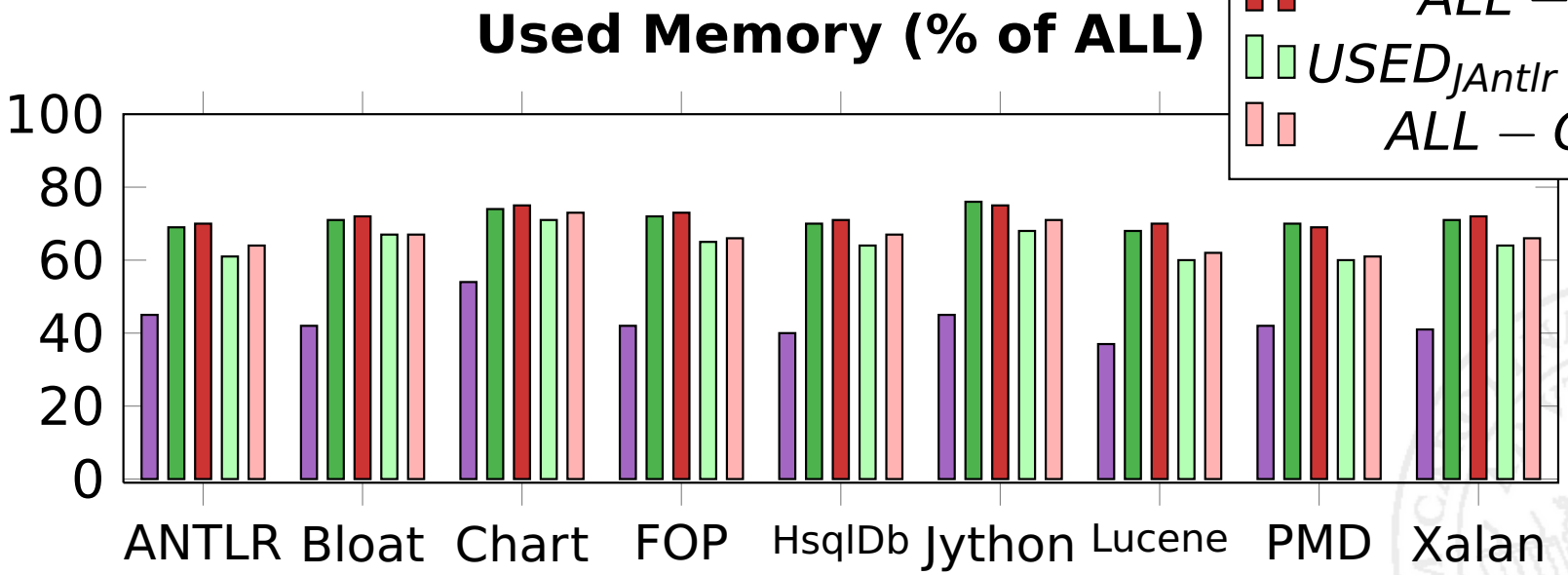


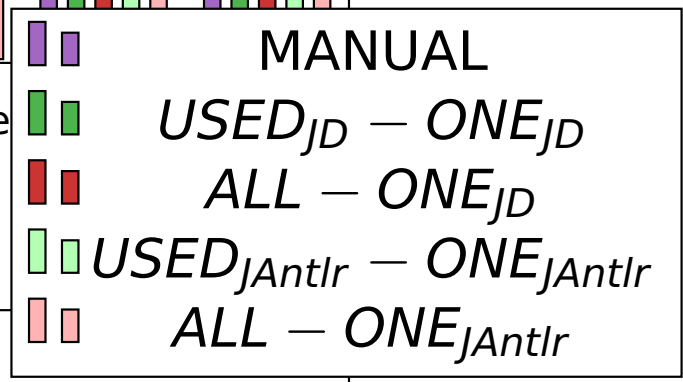
Results: Experiment C (D - DaCapo benchmarks)



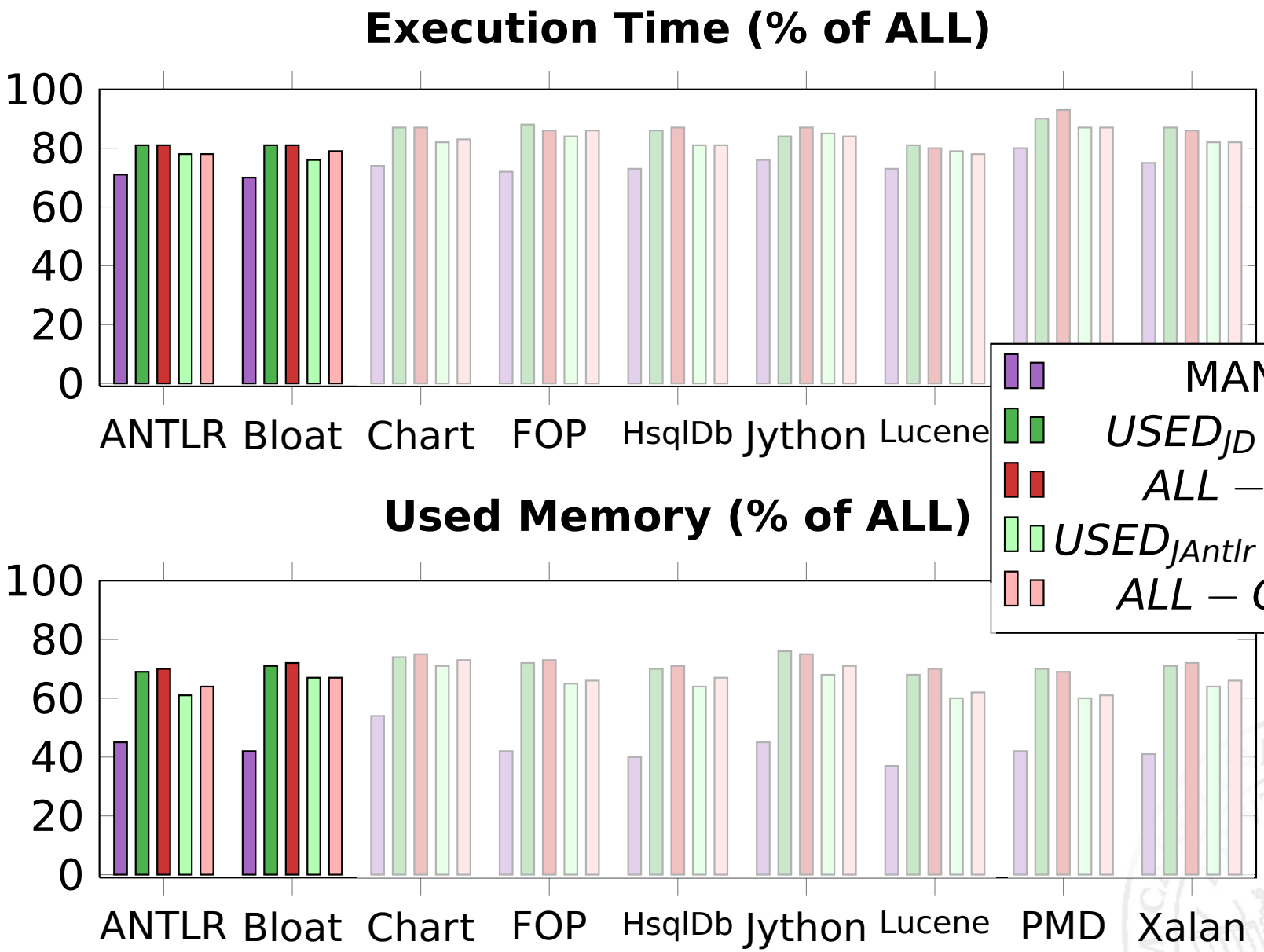
Results: Experiment D (Jacks + DaCapo)







Results: Experiment D (Jacks + DaCapo)



Conclusions/Future Work

Conclusions

- Average speed-up: 20%
- MANUAL: Possible lower bound (26%)
- Small difference ALL-ONE, USED-ONE
- We recommend ALL-ONE

Future Work

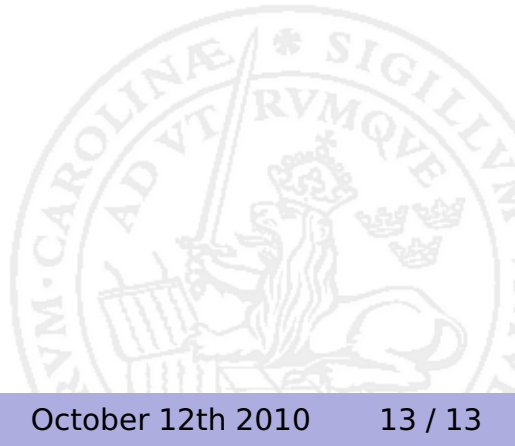
- How do we reach MANUAL?
- Results for other compilers

You can soon try out the profiler (JACO) here:

`http://svn.cs.lth.se/trac/jastadd-caching`

Thank you!

Questions?



Experiment Setup

Result **EXPERIMENT** (*pInput*, *tInput*)

pInput : Profiling input
tInput : Test input for measurement

```
1 Compiler comp ← BUILD-COMPILER (full-caching, tracing);
2 Config   conf ← PROFILE(pInput);
3          comp ← BUILD-COMPILER (conf);
4 Result   res  ← MEASURE-PERFORMANCE (comp, tInput);
```