Caches in WCET Analysis

Jan Reineke

Department of Computer Science
Saarland University
Saarbrücken, Germany

ARTIST Summer School in Europe 2009
Autrans, France
September 7-11, 2009
Outline

1  Caches

2  Cache Analysis for Least-Recently-Used

3  Beyond Least-Recently-Used
   - Predictability Metrics
   - Relative Competitiveness
   - Sensitivity – Caches and Measurement-Based Timing Analysis

4  Summary
Outline

1 Caches

2 Cache Analysis for Least-Recently-Used

3 Beyond Least-Recently-Used
   - Predictability Metrics
   - Relative Competitiveness
   - Sensitivity – Caches and Measurement-Based Timing Analysis

4 Summary
Caches

- Small but very fast memories that buffer part of the main memory
- Bridge the gap between speed of CPU and main memory

Why caches work: principle of locality
  - spatial: e.g. in sequential instructions, accessing arrays
  - temporal: e.g. in loops
Caches

- Small but very fast memories that buffer part of the main memory
- Bridge the gap between speed of CPU and main memory

Why caches work: *principle of locality*
- spatial: e.g. in sequential instructions, accessing arrays
- temporal: e.g. in loops
Caches

- Small but very fast memories that buffer part of the main memory
- Bridge the gap between speed of CPU and main memory

Why caches work: *principle of locality*

- spatial: e.g. in sequential instructions, accessing arrays
- temporal: e.g. in loops
Caches

- Small but very fast memories that buffer part of the main memory
- Bridge the gap between speed of CPU and main memory

Why caches work: *principle of locality*
- spatial: e.g. in sequential instructions, accessing arrays
- temporal: e.g. in loops
Caches

- Small but very fast memories that buffer part of the main memory
- Bridge the gap between speed of CPU and main memory

![Diagram of CPU, Cache, and Main Memory]

- Why caches work: *principle of locality*
  - spatial: e.g. in sequential instructions, accessing arrays
  - temporal: e.g. in loops
Caches

- Small but very fast memories that buffer part of the main memory
- Bridge the gap between speed of CPU and main memory

![Diagram of CPU, Cache, and Main Memory]

- Capacity: 32 KB
- Latency: 3 cycles
- 2 MB
- 100 cycles

Why caches work: *principle of locality*

- Spatial: e.g. in sequential instructions, accessing arrays
- Temporal: e.g. in loops
Caches

- Small but very fast memories that buffer part of the main memory
- Bridge the gap between speed of CPU and main memory

Why caches work: *principle of locality*
  - spatial: e.g. in sequential instructions, accessing arrays
  - temporal: e.g. in loops
Caches

- Small but very fast memories that buffer part of the main memory
- Bridge the gap between speed of CPU and main memory

**Why caches work: principle of locality**
- spatial: e.g. in sequential instructions, accessing arrays
- temporal: e.g. in loops
Caches

- Small but very fast memories that buffer part of the main memory
- Bridge the gap between speed of CPU and main memory

Why caches work: *principle of locality*
  - spatial: e.g. in sequential instructions, accessing arrays
  - temporal: e.g. in loops
Fully-Associative Caches

Address:

Tag Block

.offset

Address:\ B b
b
k s 

log\text{2}(s) log\text{2}(8*b)

\bullet \bullet \bullet

Tag Data Block

Tag Data Block

Tag Data Block

\ldots

=k = \text{associativity}

MUX

Data

Yes: Hit!

No: Miss!

\text{MUX}

=k = \text{associativity}

=?
Set-Associative Caches

Special cases:
- direct-mapped cache: only one line per cache set
- fully-associative cache: only one cache set
Cache Replacement Policies

- Least-Recently-Used (LRU) used in Intel Pentium I and MIPS 24K/34K
- First-In First-Out (FIFO or Round-Robin) used in Motorola PowerPC 56x, Intel XScale, ARM9, ARM11
- Pseudo-LRU (PLRU) used in Intel Pentium II-IV and PowerPC 75x
- Most-Recently-Used (MRU) as described in literature

Each cache set is treated independently:

→ Set-associative caches are compositions of fully-associative caches.
Outline

1 Caches

2 Cache Analysis for Least-Recently-Used

3 Beyond Least-Recently-Used
   - Predictability Metrics
   - Relative Competitiveness
   - Sensitivity – Caches and Measurement-Based Timing Analysis

4 Summary
Cache Analysis

Two types of cache analyses:

1. **Local guarantees**: classification of individual accesses
   - May-Analysis $\rightarrow$ Overapproximates cache contents
   - Must-Analysis $\rightarrow$ Underapproximates cache contents

2. **Global guarantees**: bounds on cache hits/misses

- Cache analyses almost exclusively for LRU
- In practice: FIFO, PLRU, …
Challenges for Cache Analysis

Always a cache hit/always a miss?

- Initial cache contents unknown.
- Different paths lead to these points.
- Cannot resolve address of \( z \).
Challenges for Cache Analysis

Always a cache hit/always a miss?

1. Initial cache contents unknown.
2. Different paths lead to these points.
3. Cannot resolve address of $z$.
Deriving Invariants about Cache States using Abstract Interpretation

Collecting Semantics = set of states at each program point that any execution may encounter there

Two approximations:

- Collecting Semantics uncomputable
- \( \subseteq \) Cache Semantics computable
- \( \subseteq \) \( \gamma \) (Abstract Cache Sem.) efficiently computable
Deriving Invariants about Cache States using Abstract Interpretation

Collecting Semantics =
set of states at each program point that any execution may encounter there

Two approximations:

Collecting Semantics         uncomputable
⊆ Cache Semantics            computable
⊆ \( \gamma \) (Abstract Cache Sem.) efficiently computable
Deriving Invariants about Cache States using Abstract Interpretation

Collecting Semantics = set of states at each program point that any execution may encounter there

Two approximations:

- Collecting Semantics uncomputable
- Cache Semantics computable
- \( \subseteq \gamma\) (Abstract Cache Sem.) efficiently computable
Deriving Invariants about Cache States using Abstract Interpretation

Collecting Semantics = set of states at each program point that any execution may encounter there

Two approximations:

Collecting Semantics \subseteq Cache Semantics \subseteq \gamma(\text{Abstract Cache Sem.})

uncomputable
computable
efficiently computable
Deriving Invariants about Cache States using Abstract Interpretation

Collecting Semantics = set of states at each program point that any execution may encounter there

Two approximations:

- Collecting Semantics uncomputable
- Cache Semantics computable
- $\subseteq \gamma$(Abstract Cache Sem.) efficiently computable
Least-Recently-Used (LRU): Concrete Behavior

“Cache Miss”:

```
z
y
x
t
```

LRU has notion of age

“Cache Hit”:

```
z
y
s
t
```

```
s
z
y
t
```
LRU: Must-Analysis: Abstract Domain

- Used to predict *cache hits*.
- Maintains *upper bounds on ages* of memory blocks.
- Upper bound $\leq$ associativity $\rightarrow$ memory block definitely cached.

### Example

**Abstract state:**

<table>
<thead>
<tr>
<th>State</th>
<th>Age 0</th>
<th>Age 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>{x}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>{}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>{s,t}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>{}</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

...and its interpretation:

Describes the set of all concrete cache states in which $x$, $s$, and $t$ occur,

- $x$ with an age of 0,
- $s$ and $t$ with an age not older than 2.

$$\gamma([\{x\}, \{\}, \{s, t\}, \{\}]) = \{[x, s, t, a], [x, t, s, a], [x, s, t, b], \ldots\}$$
Sound Update – Local Consistency

Abstract Update

\[ (\text{must}) \rightarrow (\text{must}') \]

\[ \gamma \]

\[ \text{concrete cache states} \]

\[ \text{Lifted Concrete Update} \]

\[ \gamma \]

\[ \text{concrete cache states} \]
LRU: Must-Analysis: Update

“Potential Cache Miss”:

```
{x}
{}
{s,t}
{}
```

```
{z}
{}
{s,t}
{}
```

“Definite Cache Hit”:

```
{x}
{}
{s,t}
{}
```

```
{s}
{}
{t}
{}
```

Why does \( t \) not age in the second case?
LRU: Must-Analysis: Join

Need to combine information where control-flow merges.

Join should be conservative:

- $\gamma(A) \subseteq \gamma(A \sqcup B)$
- $\gamma(B) \subseteq \gamma(A \sqcup B)$

```
\begin{array}{c}
\{a\} \\
\{} \\
\{c,f\} \\
\{d\}
\end{array}
\sqcup
\begin{array}{c}
\{c\} \\
\{e\} \\
\{a\} \\
\{d\}
\end{array}
\sqcup
\begin{array}{c}
\{} \\
\{} \\
\{a,c\} \\
\{d\}
\end{array}
```

“Intersection + Maximal Age”
LRU: Must-Analysis: Join

Need to combine information where control-flow merges.

Join should be conservative:

- $\gamma(A) \subseteq \gamma(A \cup B)$
- $\gamma(B) \subseteq \gamma(A \cup B)$

```
{a}  \{c\}  {\{}  \\
{}    {e}    {\{}  \\
{c,f} {a}    {a,c}  \\
{d}   {d}    {d}   
```

“Intersection + Maximal Age”
LRU: Must-Analysis: Join

Need to combine information where control-flow merges.

Join should be conservative:

- \( \gamma(A) \subseteq \gamma(A \sqcup B) \)
- \( \gamma(B) \subseteq \gamma(A \sqcup B) \)

"Intersection + Maximal Age"
LRU: Must-Analysis: Join

Need to combine information where control-flow merges.

Join should be conservative:
- \( \gamma(A) \subseteq \gamma(A \cup B) \)
- \( \gamma(B) \subseteq \gamma(A \cup B) \)

```
\begin{array}{c}
\{a\} \\
\{\} \\
\{c,f\} \\
\{d\}
\end{array} \quad \sqcup \quad \begin{array}{c}
\{c\} \\
\{e\} \\
\{a\} \\
\{d\}
\end{array} \quad \begin{array}{c}
\{\} \\
\{\} \\
\{a,c\} \\
\{d\}
\end{array}
```

“Intersection + Maximal Age”
LRU: Must-Analysis: Join

Need to combine information where control-flow merges.

Join should be conservative:

- \( \gamma(A) \subseteq \gamma(A \sqcup B) \)
- \( \gamma(B) \subseteq \gamma(A \sqcup B) \)

```plaintext
{a}  {c}  {}

{e}  {a}  {a,c}

d}  {d}  {d}
```

“Intersection + Maximal Age”
LRU: Must-Analysis: Join

Need to combine information where control-flow merges.

Join should be conservative:

- $\gamma(A) \subseteq \gamma(A \sqcup B)$
- $\gamma(B) \subseteq \gamma(A \sqcup B)$

```
\begin{array}{c}
\{a\} \\
\{\} \\
\{c,f\} \\
\{d\}
\end{array}
\sqcup
\begin{array}{c}
\{c\} \\
\{e\} \\
\{a\} \\
\{d\}
\end{array}
\sqcup
\begin{array}{c}
\{\} \\
\{\} \\
\{a,c\} \\
\{d\}
\end{array}
```

“Intersection + Maximal Age”
LRU: Must-Analysis: Join

Need to combine information where control-flow merges.

Join should be conservative:

- \( \gamma(A) \subseteq \gamma(A \sqcup B) \)
- \( \gamma(B) \subseteq \gamma(A \sqcup B) \)

```
\[
\begin{array}{c}
\{a\} \\
\{\} \\
\{c,f\} \\
\{d\}
\end{array}
\quad \sqcup \\
\begin{array}{c}
\{c\} \\
\{e\} \\
\{a\} \\
\{d\}
\end{array}
\quad \sqcup \\
\begin{array}{c}
\{\} \\
\{\} \\
\{a,c\} \\
\{d\}
\end{array}
\]
```

“Intersection + Maximal Age”

How many memory blocks can be in the must-cache?
Example: Must-Analysis

entry \[ \{\}, \{\}, \{\}, \{\} \]

A \perp B \perp C \perp D \perp exit
Example: Must-Analysis

\[
\text{entry} \quad [\{\}, \{\}, \{\}, \{\}, \{\}]
\]

\[
\perp \sqcup [\{\}, \{\}, \{\}, \{\}, \{\}] = [\{\}, \{\}, \{\}, \{\}, \{\}]
\]

\[
\text{exit} \quad \perp
\]
Example: Must-Analysis

\[
\text{entry } [\{\}, \{\}, \{\}, \{\}, \{\}]
\]

\[
\perp \sqcup [\{\}, \{\}, \{\}, \{\}, \{\}] = [\{\}, \{\}, \{\}, \{\}, \{\}]
\]

\[
[A], \{\}, \{\}, \{\}, \{\}]
\]

\[
[A], \{\}, \{\}, \{\}, \{\}]
\]

\[
[A], \{\}, \{\}, \{\}, \{\}]
\]

\[
[A], \{\}, \{\}, \{\}, \{\}]
\]

\[
[A], \{\}, \{\}, \{\}, \{\}]
\]

\[
[A], \{\}, \{\}, \{\}, \{\}]
\]

\[
[A], \{\}, \{\}, \{\}, \{\}]
\]

\[
[A], \{\}, \{\}, \{\}, \{\}]
\]

\[
[A], \{\}, \{\}, \{\}, \{\}]
\]

\[
[A], \{\}, \{\}, \{\}, \{\}]
\]

\[
[A], \{\}, \{\}, \{\}, \{\}]
\]

\[
[A], \{\}, \{\}, \{\}, \{\}]
\]

\[
[A], \{\}, \{\}, \{\}, \{\}]
\]

\[
[A], \{\}, \{\}, \{\}, \{\}]
\]
Example: Must-Analysis

\[
\begin{align*}
\text{entry} & \quad [\{\}, \{\}, \{\}, \{\}] \\
\bot \sqcup [\{\}, \{\}, \{\}, \{\}] & = [\{\}, \{\}, \{\}, \{\}] \\
[\{A\}, \{\}, \{\}, \{\}] & \quad [\{A\}, \{\}, \{\}, \{\}] \\
B & \quad C \\
\sqcup [\{B\}, \{A\}, \{\}, \{\}] & \sqcup [\{C\}, \{A\}, \{\}, \{\}] = [\{\}, \{A\}, \{\}, \{\}] \\
D & \quad \text{exit} \\
\bot & \quad \bot
\end{align*}
\]
Example: Must-Analysis

$$[\{\}, \{\}, \{\}, \{\}]$$

entry

$$[\{D\}, \{\}, \{A\}, \{\}] \sqcup [\{\}, \{\}, \{\}, \{\}] = [\{\}, \{\}, \{\}, \{\}]$$

[A]

$$[\{A\}, \{\}, \{\}, \{\}]$$

B

$$[\{B\}, \{A\}, \{\}, \{\}] \sqcup [\{C\}, \{A\}, \{\}, \{\}] = [\{\}, \{A\}, \{\}, \{\}]$$

C

$$[\{A\}, \{\}, \{\}, \{\}]$$

D

$$[\{D\}, \{\}, \{A\}, \{\}]$$

exit

No cache hits can be predicted :-(
Context-Sensitive Analysis/Virtual Loop-Unrolling

- **Problem:**
  - The first iteration of a loop will always result in cache misses.
  - Similarly for the first execution of a function.

- **Solution:**
  - Virtually Unroll Loops: Distinguish the first iteration from others
  - Distinguish function calls by calling context.

Virtually unrolling the loop once:

- **Accesses to** $A$ **and** $D$ **are provably hits after the first iteration**

- **Accesses to** $B$ **and** $C$ **can still not be classified. Within each execution of the loop, they may only miss once.**

  $\rightarrow$ Persistence Analysis
LRU: May-Analysis: Abstract Domain

- Used to predict *cache misses*.
- Maintains *lower bounds on ages* of memory blocks.
- Lower bound \( \geq \) associativity

\[ \rightarrow \text{memory block definitely not cached.} \]

**Example**

<table>
<thead>
<tr>
<th>Abstract state:</th>
<th>{x,y}</th>
<th>{}</th>
<th>{s,t}</th>
<th>{u}</th>
</tr>
</thead>
<tbody>
<tr>
<td>age 0</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>age 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ \gamma([\{x, y\}, \{\}, \{s, t\}, \{u\}]) = \{[x, y, s, t], [y, x, s, t], [x, y, s, u], \ldots \} \]

\[ \ldots \text{and its interpretation:} \]

Describes the set of all concrete cache states in which no memory blocks except \( x, y, s, t, \) and \( u \) occur,

- \( x \) and \( y \) with an age of at least 0,
- \( s \) and \( t \) with an age of at least 2,
- \( u \) with an age of at least 3.
LRU: May-Analysis: Update

“Definite Cache Miss”:

“Potential Cache Hit”:

Why does $t$ age in the second case?
LRU: May-Analysis: Join

Need to combine information where control-flow merges.

Join should be conservative:

- $\gamma(A) \subseteq \gamma(A \sqcup B)$
- $\gamma(B) \subseteq \gamma(A \sqcup B)$

```
\begin{array}{c|c|c}
\{a\} & \{c\} & \{a,c\} \\
\{\} & \{e\} & \{e\} \\
\{c,f\} & \{a\} & \{f\} \\
\{d\} & \{d\} & \{d\}
\end{array}
```

“Union + Minimal Age”
LRU: May-Analysis: Join

Need to combine information where control-flow merges.

Join should be conservative:

- \( \gamma(A) \subseteq \gamma(A \sqcup B) \)
- \( \gamma(B) \subseteq \gamma(A \sqcup B) \)

\[
\begin{align*}
\{a\} & \sqcup \{c\} = \{a,c\} \\
\{\} & \sqcup \{e\} = \{e\} \\
\{c,f\} & \sqcup \{a\} = \{a,c\} \\
\{d\} & \sqcup \{d\} = \{d\}
\end{align*}
\]

“Union + Minimal Age”
LRU: May-Analysis: Join

Need to combine information where control-flow merges.

Join should be conservative:

\[ \gamma(A) \subseteq \gamma(A \sqcup B) \]
\[ \gamma(B) \subseteq \gamma(A \sqcup B) \]

\[
\begin{array}{c|c}
\{a\} & \{c\} \\
\{\} & \{e\} \\
\{c,f\} & \{a\} \\
\{d\} & \{d\} \\
\end{array}
\sqcup
\begin{array}{c|c}
\{c\} & \{a,c\} \\
\{e\} & \{e\} \\
\{d\} & \{f\} \\
\{d\} & \{d\} \\
\end{array}
= \\
\begin{array}{c|c}
\{a,c\} & \\
\{e\} & \\
\{f\} & \\
\{d\} & \\
\end{array}
\]

“Union + Minimal Age”
LRU: May-Analysis: Join

Need to combine information where control-flow merges.

Join should be conservative:

- $\gamma(A) \subseteq \gamma(A \cup B)$
- $\gamma(B) \subseteq \gamma(A \cup B)$

```
\{a\}  \{c\}  \{a,c\}
{\}    {e}    {e}
{c,f}  {a}    {f}
{d}    {d}    {d}
```

“Union + Minimal Age”
LRU: May-Analysis: Join

Need to combine information where control-flow merges.

Join should be conservative:

1. \( \gamma(A) \subseteq \gamma(A \sqcup B) \)
2. \( \gamma(B) \subseteq \gamma(A \sqcup B) \)

\[
\begin{array}{c|c|c}
\{a\} & \{c\} & \{a,c\} \\
\{\} & \{e\} & \{e\} \\
\{c,f\} & \{a\} & \{f\} \\
\{d\} & \{d\} & \{d\} \\
\end{array}
\]

“Union + Minimal Age”
LRU: May-Analysis: Join

Need to combine information where control-flow merges.

Join should be conservative:

- $\gamma(A) \subseteq \gamma(A \sqcup B)$
- $\gamma(B) \subseteq \gamma(A \sqcup B)$

```
\begin{array}{c}
\{a\} \\
\{\} \\
\{c,f\} \\
\{d\}
\end{array}
\sqcup
\begin{array}{c}
\{c\} \\
\{e\} \\
\{a\} \\
\{d\}
\end{array}
= \begin{array}{c}
\{a,c\} \\
\{e\} \\
\{f\} \\
\{d\}
\end{array}
```

“Union + Minimal Age”
Outline

1. Caches

2. Cache Analysis for Least-Recently-Used

3. Beyond Least-Recently-Used
   - Predictability Metrics
   - Relative Competitiveness
   - Sensitivity – Caches and Measurement-Based Timing Analysis

4. Summary
Outline

1. Caches

2. Cache Analysis for Least-Recently-Used

3. Beyond Least-Recently-Used
   - Predictability Metrics
   - Relative Competitiveness
   - Sensitivity – Caches and Measurement-Based Timing Analysis

4. Summary
- Amount of uncertainty determines precision of WCET analysis
- Uncertainty in cache analysis depends on replacement policy

\[ \text{uncertainty} \times \text{penalty} \]

Variation due to inputs and initial hardware state

BCET | ACET | WCET upper bound | execution time
Uncertainty in Cache Analysis

1. Initial cache contents unknown.
2. Need to combine information.
3. Cannot resolve address of $z$.

Amount of uncertainty determined by ability to recover information.

Jan Reineke
Uncertainty in Cache Analysis

1. Initial cache contents unknown.
Uncertainty in Cache Analysis

1. Initial cache contents unknown.
2. Need to combine information.

- read z
- read y
- read x
- write z

Amount of uncertainty determined by ability to recover information.
Uncertainty in Cache Analysis

1. Initial cache contents unknown.
2. Need to combine information.
3. Cannot resolve address of $z$. 

Jan Reineke
Caches in WCET Analysis
ARTIST Summer School 2009 25 / 51
Uncertainty in Cache Analysis

1. Initial cache contents unknown.
2. Need to combine information.
3. Cannot resolve address of $z$.

$\Rightarrow$ Amount of uncertainty determined by ability to recover information.
Predictability Metrics

Sequence: \( \langle a, \ldots, e, f, g, h \rangle \)
Meaning of Metrics

- Evict
  - Number of accesses to obtain *any* may-information.
  - I.e. when can an analysis predict any cache misses?
- Fill
  - Number of accesses to complete *may*- and must-information.
  - I.e. when can an analysis predict each access?

→ Evict and Fill bound the precision of *any* static cache analysis. Can thus serve as a benchmark for analyses.
Evaluation of Least-Recently-Used

- LRU “forgets” about past quickly:
  - cares about most-recent access to each block only
  - order of previous accesses irrelevant

In the example: Evict = Fill = 4

In general: Evict(k) = Fill(k) = k, where k is the associativity of the cache
Evaluation of First-In First-Out (sketch)

- Like LRU in the miss-case
- But: “Ignores” hits

In the worst-case $k - 1$ hits and $k$ misses: \( (k = \text{associativity}) \)

\[ \text{Evict}(k) = 2k - 1 \]

Another $k$ accesses to obtain complete knowledge:

\[ \text{Fill}(k) = 3k - 1 \]
Evaluation of Pseudo-LRU (sketch)

- Tree-bits point to block to be replaced

Accesses “rejuvenate” neighborhood
  - Active blocks keep their (inactive) neighborhood in the cache

Analysis yields:
  - Evict\( (k) = \frac{k}{2} \log_2 k + 1 \)
  - Fill\( (k) = \frac{k}{2} \log_2 k + k - 1 \)
### Evaluation of Policies

<table>
<thead>
<tr>
<th>Policy</th>
<th>Evict((k))</th>
<th>Fill((k))</th>
<th>Evict((8))</th>
<th>Fill((8))</th>
</tr>
</thead>
<tbody>
<tr>
<td>LRU</td>
<td>(k)</td>
<td>(k)</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>FIFO</td>
<td>(2k - 1)</td>
<td>(3k - 1)</td>
<td>15</td>
<td>23</td>
</tr>
<tr>
<td>MRU</td>
<td>(2k - 2)</td>
<td>(\infty / 3k - 4)</td>
<td>14</td>
<td>(\infty / 20)</td>
</tr>
<tr>
<td>PLRU</td>
<td>(k/2 \log_2 k + 1)</td>
<td>(k/2 \log_2 k + k - 1)</td>
<td>13</td>
<td>19</td>
</tr>
</tbody>
</table>

- LRU is optimal w.r.t. metrics.
- Other policies are much less predictable.

→ Use LRU if predictability is a concern.

- How to obtain *may*- and *must*-information within the given limits for other policies?
Outline

1. Caches

2. Cache Analysis for Least-Recently-Used

3. Beyond Least-Recently-Used
   - Predictability Metrics
   - Relative Competitiveness
   - Sensitivity – Caches and Measurement-Based Timing Analysis

4. Summary
Relative Competitiveness

- **Competitiveness** (Sleator and Tarjan, 1985): worst-case performance of an online policy *relative to the optimal offline policy*
  - used to evaluate online policies

- **Relative competitiveness** (Reineke and Grund, 2008): worst-case performance of an online policy *relative to another online policy*
  - used to derive local and global cache analyses
### Definition – Relative Miss-Competitiveness

**Notation**

\[ m_P(p, s) = \text{number of misses that policy } P \text{ incurs on access sequence } s \in M^* \text{ starting in state } p \in C^P \]
### Definition – Relative Miss-Competitiveness

**Notation**

\[
m_P(p, s) = \text{number of misses that policy } P \text{ incurs on access sequence } s \in M^* \text{ starting in state } p \in C^P
\]

**Definition (Relative miss competitiveness)**

Policy \( P \) is \((k, c)\)-miss-competitive relative to policy \( Q \) if

\[
m_P(p, s) \leq k \cdot m_Q(q, s) + c
\]

for all access sequences \( s \in M^* \) and cache-set states \( p \in C^P, q \in C^Q \) that are compatible \( p \sim q \).
Definition – Relative Miss-Competitiveness

Notation

\[ m_P(p, s) = \text{number of misses that policy } P \text{ incurs on access sequence } s \in M^* \text{ starting in state } p \in C^p \]

Definition (Relative miss competitiveness)

Policy \( P \) is \((k, c)\)-miss-competitive relative to policy \( Q \) if

\[ m_P(p, s) \leq k \cdot m_Q(q, s) + c \]

for all access sequences \( s \in M^* \) and cache-set states \( p \in C^p, q \in C^Q \) that are compatible \( p \sim q \).

Definition (Competitive miss ratio of \( P \) relative to \( Q \))

The smallest \( k \), s.t. \( P \) is \((k, c)\)-miss-competitive rel. to \( Q \) for some \( c \).
Example – Relative Miss-Competitiveness

\[ P \text{ is (3, 4)-miss-competitive relative to } Q. \]
If \( Q \) incurs \( x \) misses, then \( P \) incurs at most \( 3 \cdot x + 4 \) misses.
Example – Relative Miss-Competitiveness

\( \mathbf{P} \) is \((3, 4)\)-miss-competitive relative to \( \mathbf{Q} \).
If \( \mathbf{Q} \) incurs \( x \) misses, then \( \mathbf{P} \) incurs at most \( 3 \cdot x + 4 \) misses.

**Best:** \( \mathbf{P} \) is \((1, 0)\)-miss-competitive relative to \( \mathbf{Q} \).
Example – Relative Miss-Competitiveness

\[ P \text{ is } (3, 4)\text{-miss-competitive relative to } Q. \]
If \( Q \) incurs \( x \) misses, then \( P \) incurs at most \( 3 \cdot x + 4 \) misses.

**Best:** \( P \) is \((1, 0)\text{-miss-competitive relative to } Q.\)

**Worst:** \( P \) is not-miss-competitive (or \( \infty \text{-miss-competitive} \)) relative to \( Q. \)
Example – Relative Hit-Competitiveness

\( P \) is \( (\frac{2}{3}, 3) \)-hit-competitive relative to \( Q \).
If \( Q \) has \( x \) hits, then \( P \) has at least \( \frac{2}{3} \cdot x - 3 \) hits.
Example – Relative Hit-Competitiveness

\( P \) is \( (\frac{2}{3}, 3) \)-hit-competitive relative to \( Q \).
If \( Q \) has \( x \) hits, then \( P \) has at least \( \frac{2}{3} \cdot x - 3 \) hits.

**Best:** \( P \) is \((1, 0)\)-hit-competitive relative to \( Q \).
Equivalent to \((1, 0)\)-miss-competitiveness.
Example – Relative Hit-Competitiveness

P is \(\left(\frac{2}{3}, 3\right)\)-hit-competitive relative to Q.
If Q has \(x\) hits, then P has at least \(\frac{2}{3} \cdot x - 3\) hits.

**Best:** P is \((1, 0)\)-hit-competitive relative to Q.
Equivalent to \((1, 0)\)-miss-competitiveness.

**Worst:** P is \((0, 0)\)-hit-competitive relative to Q.
Analogue to \(\infty\)-miss-competitiveness.
Local Guarantees: (1, 0)-Competitiveness

Let $P$ be (1, 0)-competitive relative to $Q$:

$$m_P(p, s) \leq 1 \cdot m_Q(q, s) + 0$$

$$\iff m_P(p, s) \leq m_Q(q, s)$$

If $Q$ "hits", so does $P$, and if $P$ "misses", so does $Q$.

As a consequence, $1$-analysis for $Q$ is also a $1$-analysis for $P$, and $2$-analysis for $P$ is also a $2$-analysis for $Q$. 
Local Guarantees: (1, 0)-Competitiveness

Let $P$ be $(1, 0)$-competitive relative to $Q$:

$$m_P(p, s) \leq 1 \cdot m_Q(q, s) + 0$$

$$\Leftrightarrow m_P(p, s) \leq m_Q(q, s)$$

1. If $Q$ “hits”, so does $P$, and
2. if $P$ “misses”, so does $Q$. 
Local Guarantees: (1, 0)-Competitiveness

Let $P$ be (1, 0)-competitive relative to $Q$:

$$m_P(p, s) \leq 1 \cdot m_Q(q, s) + 0$$

$$\iff m_P(p, s) \leq m_Q(q, s)$$

1. If $Q$ “hits”, so does $P$, and
2. if $P$ “misses”, so does $Q$.

As a consequence,

1. a must-analysis for $Q$ is also a must-analysis for $P$, and
2. a may-analysis for $P$ is also a may-analysis for $Q$. 
Global Guarantees: \((k, c)\)-Competitiveness

**Given:** Global guarantees for policy \(Q\).

**Wanted:** Global guarantees for policy \(P\).
Global Guarantees: \((k, c)\)-Competitiveness

Given:  Global guarantees for policy \(Q\).
Wanted:  Global guarantees for policy \(P\).

1. Determine competitiveness of policy \(P\) relative to policy \(Q\).

\[ m_P \leq k \cdot m_Q + c \]
Global Guarantees: \((k, c)\)-Competitiveness

Given: Global guarantees for policy \(Q\).

Wanted: Global guarantees for policy \(P\).

1. Determine competitiveness of policy \(P\) relative to policy \(Q\).
   \[ m_P \leq k \cdot m_Q + c \]

2. Compute global guarantee for task \(T\) under policy \(Q\).
   \[ m_Q(T) \]
Global Guarantees: \((k, c)\)-Competitiveness

**Given:** Global guarantees for policy \(Q\).

**Wanted:** Global guarantees for policy \(P\).

1. Determine competitiveness of policy \(P\) relative to policy \(Q\).
   \[
   m_P \leq k \cdot m_Q + c
   \]

2. Compute global guarantee for task \(T\) under policy \(Q\).
   \[
   m_Q(T)
   \]

3. Calculate global guarantee on the number of misses for \(P\) using the global guarantee for \(Q\) and the competitiveness results of \(P\) relative to \(Q\).
   \[
   m_P \leq k \cdot m_Q + c \quad m_Q(T) = m_P(T)
   \]
Relative Competitiveness: Automatic Computation

P and Q (here: FIFO and LRU) induce transition system:

\[
\begin{align*}
&[eabc]_{\text{FIFO}}, [eabc]_{\text{LRU}} \\
\xrightarrow{e} &[abcd]_{\text{FIFO}}, [abcd]_{\text{LRU}} \\
\xleftarrow{c} &[eabc]_{\text{FIFO}}, [ceab]_{\text{LRU}} \\
\xrightarrow{d} &[abcd]_{\text{FIFO}}, [dabc]_{\text{LRU}} \\
\xleftarrow{e} &[eabc]_{\text{FIFO}}, [ceda]_{\text{LRU}} \\
\xrightarrow{d} &[deab]_{\text{FIFO}}, [deab]_{\text{LRU}}
\end{align*}
\]

Legend:
- \([abcd]_{\text{FIFO}}\) Cache-set state
- \([abcd]_{\text{LRU}}\) Memory access
- \((h,m), \ldots\) Misses in pairs of cache-set states

Competitive miss ratio = maximum ratio of misses in policy P to misses in policy Q in transition system
Transition System is $\infty$ Large

**Problem:** The induced transition system is $\infty$ large.

**Observation:** Only the *relative positions* of elements matter:

\[
\begin{align*}
[abc]_{\text{LRU}}, [bde]_{\text{FIFO}} & \approx [fgl]_{\text{LRU}}, [ghm]_{\text{FIFO}} \\
\quad c \quad (h, m) & \quad I \quad (h, m)
\end{align*}
\]

\[
\begin{align*}
[cab]_{\text{LRU}}, [cbd]_{\text{FIFO}} & \approx [lfg]_{\text{LRU}}, [lgh]_{\text{FIFO}} \\
\quad c \quad (h, m) & \quad l \quad (h, m)
\end{align*}
\]

**Solution:** Construct *finite* quotient transition system.
〜-Equivalent States in Running Example

\[
\begin{align*}
 [eabc]_{\text{FIFO}} & \rightarrow [eabc]_{\text{LRU}} \\
 [abcd]_{\text{FIFO}} & \rightarrow [abcd]_{\text{LRU}} \\
 [eabc]_{\text{FIFO}} & \rightarrow [ceab]_{\text{LRU}} \\
 [abcd]_{\text{FIFO}} & \rightarrow [dabc]_{\text{LRU}} \\
 [eabc]_{\text{FIFO}} & \rightarrow [ceda]_{\text{LRU}} \\
 [eabc]_{\text{FIFO}} & \rightarrow [edab]_{\text{LRU}} \\
 [deab]_{\text{FIFO}} & \rightarrow [deab]_{\text{LRU}}
\end{align*}
\]
Merging ≈-equivalent states yields a finite quotient transition system:
Competitive Ratio = Maximum Cycle Ratio

Competitive miss ratio =

maximum ratio of misses in policy $P$ to misses in policy $Q$
Competitive Ratio = Maximum Cycle Ratio

Competitive miss ratio =
maximum ratio of misses in policy \( P \) to misses in policy \( Q \)

Maximum cycle ratio = \[
\frac{0 + 1 + 1}{0 + 1 + 0} = 2
\]
Tool Implementation

- Implemented in Java, called Relacs
- Interface for replacement policies

- Fully automatic
- Provides example sequences for competitive ratio and constant

- Analysis usually practically feasible up to associativity 8
  - limited by memory consumption
  - depends on similarity of replacement policies

Online version:

http://rw4.cs.uni-sb.de/~reineke/relacs
Generalizations

Identified patterns and proved generalizations by hand. Aided by example sequences generated by tool.

Previously unknown facts:

- \( \text{PLRU}(k) \) is \((1,0)\) comp. rel. to \( \text{LRU}(1+\log_2 k) \), \( \Rightarrow \) \( \text{LRU} \)-analysis can be used for \( \text{PLRU} \)

- \( \text{FIFO}(k) \) is \((1/2,k-1/2)\) hit-comp. rel. to \( \text{LRU}(k) \), whereas \( \text{LRU}(k) \) is \((0,0)\) hit-comp. rel. to \( \text{FIFO}(k) \), but \( \text{LRU}(2k-1) \) is \((1,0)\) comp. rel. to \( \text{FIFO}(k) \), and \( \text{LRU}(2k-2) \) is \((1,0)\) comp. rel. to \( \text{MRU}(k) \).

\( \Rightarrow \) \( \text{LRU} \)-analysis may be used for \( \text{FIFO} \) and \( \text{MRU} \) \( \Rightarrow \) optimal with respect to predictability metric \( \text{Evict} \)

\( \text{FIFO} \)-analysis used in the analysis of the branch target buffer of the Motorola PowerPC 56X.
Generalizations

Identified patterns and proved generalizations by hand. Aided by example sequences generated by tool.

Previously unknown facts:

\[
\text{PLRU}(k) \text{ is } (1, 0) \text{ comp. rel. to } \text{LRU}(1 + \log_2 k),
\]

\[\text{→ LRU-} \text{must}\text{-analysis can be used for PLRU}\]
Generalizations

Identified patterns and proved generalizations by hand. Aided by example sequences generated by tool.

Previously unknown facts:

- PLRU($k$) is $(1, 0)$ comp. rel. to LRU($1 + \log_2 k$),
  \[\text{LRU-must-analysis can be used for PLRU}\]

- FIFO($k$) is $\left(\frac{1}{2}, \frac{k-1}{2}\right)$ hit-comp. rel. to LRU($k$), whereas

- LRU($k$) is $(0, 0)$ hit-comp. rel. to FIFO($k$), but
Generalizations

Identified patterns and proved generalizations by hand. Aided by example sequences generated by tool.

Previously unknown facts:

- $\text{PLRU}(k)$ is $(1, 0)$ comp. rel. to $\text{LRU}(1 + \log_2 k)$, $\rightarrow$ LRU-	extit{must}-analysis can be used for PLRU

- $\text{FIFO}(k)$ is $(\frac{1}{2}, \frac{k-1}{2})$ hit-comp. rel. to $\text{LRU}(k)$, whereas $\text{LRU}(k)$ is $(0, 0)$ hit-comp. rel. to $\text{FIFO}(k)$, but

- $\text{LRU}(2k - 1)$ is $(1, 0)$ comp. rel. to $\text{FIFO}(k)$, and $\text{LRU}(2k - 2)$ is $(1, 0)$ comp. rel. to $\text{MRU}(k)$. $\rightarrow$ LRU-	extit{may}-analysis can be used for FIFO and MRU $\rightarrow$ optimal with respect to predictability metric Evict
Generalizations

Identified patterns and proved generalizations by hand. Aided by example sequences generated by tool.

Previously unknown facts:

\[
\begin{align*}
\text{PLRU}(k) & \text{ is } (1, 0) \text{ comp. rel. to } \text{LRU}(1 + \log_2 k), \\
\quad \rightarrow & \text{ LRU-} \textit{must}-\text{analysis can be used for PLRU} \\
\text{FIFO}(k) & \text{ is } \left(\frac{1}{2}, \frac{k - 1}{2}\right) \text{ hit-comp. rel. to } \text{LRU}(k), \text{ whereas} \\
\text{LRU}(k) & \text{ is } (0, 0) \text{ hit-comp. rel. to FIFO}(k), \text{ but} \\
\text{LRU}(2k - 1) & \text{ is } (1, 0) \text{ comp. rel. to FIFO}(k), \text{ and} \\
\text{LRU}(2k - 2) & \text{ is } (1, 0) \text{ comp. rel. to MRU}(k). \\
\quad \rightarrow & \text{ LRU-} \textit{may}-\text{analysis can be used for FIFO and MRU} \\
\quad \rightarrow & \text{ optimal with respect to predictability metric Evict} \\
\end{align*}
\]

FIFO- \textit{may}-analysis used in the analysis of the branch target buffer of the \textsc{Motorola PowerPC 56x}.
Outline

1. Caches

2. Cache Analysis for Least-Recently-Used

3. Beyond Least-Recently-Used
   - Predictability Metrics
   - Relative Competitiveness
   - Sensitivity – Caches and Measurement-Based Timing Analysis

4. Summary
Measurement-Based Timing Analysis

- Run program on a number of inputs and initial states.
- Combine measurements for basic blocks to obtain WCET estimation.
- Sensitivity Analysis demonstrates this approach may be dramatically wrong.
Measurement-Based Timing Analysis

- Run program on a number of inputs and initial states.
- Combine measurements for basic blocks to obtain WCET estimation.
- Sensitivity Analysis demonstrates this approach may be dramatically wrong.
Definition (Miss sensitivity)

Policy $\mathcal{P}$ is $(k, c)$-miss-sensitive if

$$m_\mathcal{P}(q, s) \leq k \cdot m_\mathcal{P}(q', s) + c$$

for all access sequences $s \in M^*$ and cache-set states $q, q' \in C^\mathcal{P}$.
Sensitivity Results

<table>
<thead>
<tr>
<th>Policy</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>LRU</td>
<td>1,2</td>
<td>1,3</td>
<td>1,4</td>
<td>1,5</td>
<td>1,6</td>
<td>1,7</td>
<td>1,8</td>
</tr>
<tr>
<td>FIFO</td>
<td>2,2</td>
<td>3,3</td>
<td>4,4</td>
<td>5,5</td>
<td>6,6</td>
<td>7,7</td>
<td>8,8</td>
</tr>
<tr>
<td>PLRU</td>
<td>1,2</td>
<td>–</td>
<td>∞</td>
<td>–</td>
<td>–</td>
<td>–</td>
<td>∞</td>
</tr>
<tr>
<td>MRU</td>
<td>1,2</td>
<td>3,4</td>
<td>5,6</td>
<td>7,8</td>
<td>MEM</td>
<td>MEM</td>
<td>MEM</td>
</tr>
</tbody>
</table>

- LRU is optimal. Performance varies in the least possible way.
- For FIFO, PLRU, and MRU the number of misses may vary strongly.
- Case study based on simple model of execution time by Hennessy and Patterson (2003): WCET may be 3 times higher than a measured execution time for 4-way FIFO.
Outline

1 Caches

2 Cache Analysis for Least-Recently-Used

3 Beyond Least-Recently-Used
   - Predictability Metrics
   - Relative Competitiveness
   - Sensitivity – Caches and Measurement-Based Timing Analysis

4 Summary
Summary

Cache Analysis for Least-Recently-Used

...efficiently represents sets of cache states by bounding the age of memory blocks from above and below.

...requires context-sensitivity for precision.
Summary

Cache Analysis for Least-Recently-Used

...efficiently represents sets of cache states by bounding the age of memory blocks from above and below.

...requires context-sensitivity for precision.

Predictability Metrics

...quantify the predictability of replacement policies.

→ LRU is the most predictable policy.

Thank you for your attention!
Summary

Cache Analysis for Least-Recently-Used

...efficiently represents sets of cache states by bounding the age of memory blocks from above and below.

...requires context-sensitivity for precision.

Predictability Metrics

...quantify the predictability of replacement policies.

→ LRU is the most predictable policy.

Relative Competitiveness

...allows to derive guarantees on cache performance,

...yields first may-analyses for FIFO and MRU.
Summary

Cache Analysis for Least-Recently-Used

...efficiently represents sets of cache states by bounding the age of memory blocks from above and below.

...requires context-sensitivity for precision.

Predictability Metrics

...quantify the predictability of replacement policies.

→ LRU is the most predictable policy.

Relative Competitiveness

...allows to derive guarantees on cache performance,

...yields first may-analyses for FIFO and MRU.

Sensitivity Analysis

...determines the influence of initial state on cache performance.
Summary

Cache Analysis for Least-Recently-Used

...efficiently represents sets of cache states by bounding the age of memory blocks from above and below.

...requires context-sensitivity for precision.

Predictability Metrics

...quantify the predictability of replacement policies.

→ LRU is the most predictable policy.

Relative Competitiveness

...allows to derive guarantees on cache performance,

...yields first *may*-analyses for FIFO and MRU.

Sensitivity Analysis

...determines the influence of initial state on cache performance.

Thank you for your attention!
Summary

Cache Analysis for Least-Recently-Used

... efficiently represents sets of cache states by bounding the age of memory blocks from above and below.

... requires context-sensitivity for precision.

Predictability Metrics

... quantify the predictability of replacement policies.

→ LRU is the most predictable policy.

Relative Competitiveness

... allows to derive guarantees on cache performance,

... yields first *may*-analyses for FIFO and MRU.

Sensitivity Analysis

... determines the influence of initial state on cache performance.

Thank you for your attention!


Most-Recently-Used – MRU

MRU-bits record whether line was recently used

\[
\begin{align*}
[abcd]_{0101} \rightarrow b, d \\
[ebcd]_{1101} \rightarrow e, b, d \\
[ebcd]_{0010} \rightarrow c
\end{align*}
\]

\[\rightarrow\] Never converges
Pseudo-LRU – PLRU

Initial cache-set state $[a, b, c, d]_{110}$.

After a miss on e. State: $[a, b, e, d]_{011}$.

After a hit on a. State: $[a, b, e, d]_{111}$.

After a miss on f. State: $[a, b, e, f]_{010}$.

Hit on a “rejuvenates” neighborhood; “saves” b from eviction.
May- and Must-Information

\[
\text{May}^P(s) := \bigcup_{p \in C^P} CC_p(update_p(p, s))
\]

\[
\text{Must}^P(s) := \bigcap_{p \in C^P} CC_p(update_p(p, s))
\]

\[
\text{may}^P(n) := \left| \text{May}^P(s) \right|, \text{where } s \in S^\neq \subsetneq M^*, |s| = n
\]

\[
\text{must}^P(n) := \left| \text{Must}^P(s) \right|, \text{where } s \in S^\neq \subsetneq M^*, |s| = n
\]

\(S^\neq\) : set of finite access sequences with pairwise different accesses
Definitions of Metrics

\[ \text{Evict}^P := \min \left\{ n \mid \text{may}^P(n) \leq n \right\}, \]

\[ \text{Fill}^P := \min \left\{ n \mid \text{must}^P(n) = k \right\}, \]

where \( k \) is \( P \)'s associativity.
Let $P(k)$ be $(1, 0)$-miss-competitive relative to policy $Q(l)$, then

(i) $\text{Evict}^P(k) \geq \text{Evict}^Q(l)$,

(ii) $\text{mls}^P(k) \geq \text{mls}^Q(l)$. 
Alternative Pred. Metrics ↔ Rel. Competitiveness

Let $l$ be the smallest associativity, such that $\text{LRU}(l)$ is $(1, 0)$-miss-competitive relative to $P(k)$. Then

$$\text{Alt-Evict}^P(k) = l.$$ 

Let $l$ be the greatest associativity, such that $P(k)$ is $(1, 0)$-miss-competitive relative to $\text{LRU}(l)$. Then

$$\text{Alt-mls}^P(k) = l.$$
Size of Transition System

\[
2^{l+l'} \cdot \sum_{i=0}^{k} \binom{k}{i} \cdot \sum_{i'=0}^{k'} \binom{k'}{i'} \cdot \sum_{j=0}^{|\min\{i,i'\}|} \binom{j}{i} \binom{j}{i'} j!
\]

status bits of P and Q

non-empty lines in P

non-empty lines in Q

number of overlappings in non-empty lines

\[
\min\{k,k'\} \sum_{j=0}^{|\min\{k,k'\}|} \binom{k}{j} \binom{k'}{j} j! \leq k! \cdot k'! \sum_{j=0}^{|\min\{k,k'\}|} \frac{1}{(k-j)!j!(k'-j)!}
\]

\[
\leq k! \cdot k'! \sum_{j=0}^{\infty} \frac{1}{j!} = e \cdot k! \cdot k'!
\]

This can be bounded by

\[
2^{l+l'+k+k'} \leq \left| (C_k^l \times C_{k'}^{l'}) \right| \approx 1 \leq 2^{l+l'+k+k'} \cdot e \cdot k! \cdot k'!
\]

bound on number of overlappings
Compatible States

\[ i^P = [\bot \bot \bot \bot]_P \approx i^Q = [\bot \bot \bot \bot]_Q \]

update\textsubscript{P}(i^P, s) \approx update\textsubscript{Q}(i^Q, s)
Let $P$ be $(1, 0)$-competitive relative to $Q$, then

$$m_P(p, \langle x \rangle) = 1 \implies m_Q(q, \langle x \rangle) = 1$$
(1, 0)-Competitiveness and May/Must-Analyses

\[ \forall p \in P : m_P(p, \langle x \rangle) = 1 \]

\[ \forall q \in Q : m_Q(q, \langle x \rangle) = 1 \]
Case Study: Impact of Sensitivity

- Simple model of execution time from Hennessy & Patterson (2003)
- \( CPI_{hit} = \) Cycles per instruction assuming cache hits only
- \( \frac{Memory \text{ accesses}}{Instruction} \) including instruction and data fetches

\[
\frac{T_{wc}}{T_{meas}} = CPI_{hit} + \frac{Memory \text{ accesses}}{Instruction} \times Miss \text{ rate}_{wc} \times Miss \text{ penalty} \quad CPI_{hit} + \frac{Memory \text{ accesses}}{Instruction} \times Miss \text{ rate}_{meas} \times Miss \text{ penalty}
\]

\[
= \frac{1.5 + 1.2 \times 0.20 \times 50}{1.5 + 1.2 \times 0.05 \times 50} = \frac{13.5}{4.5} = 3
\]