Verification of Real-Time Systems
Caches in WCET Analysis: Beyond LRU

Jan Reineke

Department of Computer Science
Saarland University
Saarbrücken, Germany

Advanced Lecture, Summer 2015

Outline

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Outline

1. Beyond Least-Recently-Used
   - Popular Replacement Policies
   - Why Are They Hard to Analyze?
   - Generic Analysis Approach: Relative Competitiveness
   - Specialized Analyses for FIFO

2. Summary

Outline

SAARLAND UNIVERSITY
COMPUTER SCIENCE

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, ATOM, CORE 2, ... and POWERPC 75X
- Most Recently Used (MRU) used in
  INTEL NEHALEM

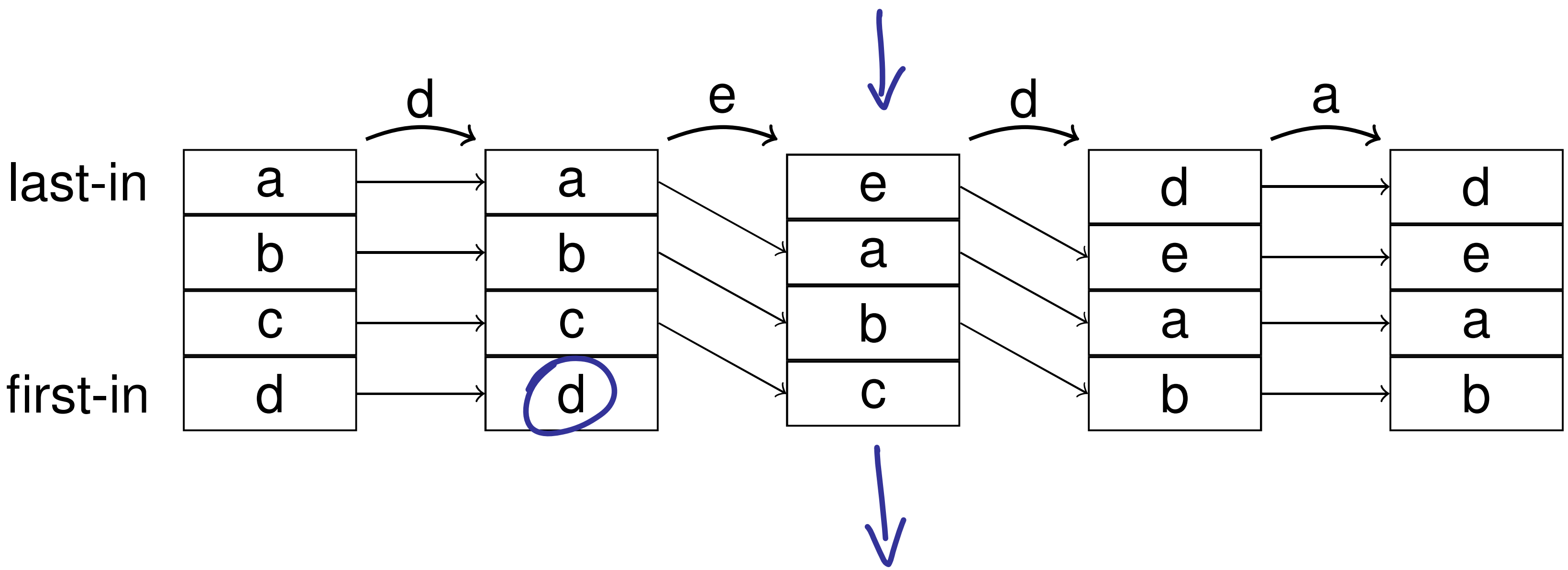
Each cache set is treated independently:
$\longrightarrow$ Set-associative caches are compositions of fully-associative caches.

First-In First-Out (also known as Round-Robin)

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Replace block that has been in the cache for the longest time.

- Cheap to implement:
  - Only $\log_2 k$ status bits, versus $\log_2(k!) \approx k \cdot \log k$ status bits for LRU.
  - No change to status bits upon hits.
- "Logical abstraction": order blocks from last- to first-in.
- Example:

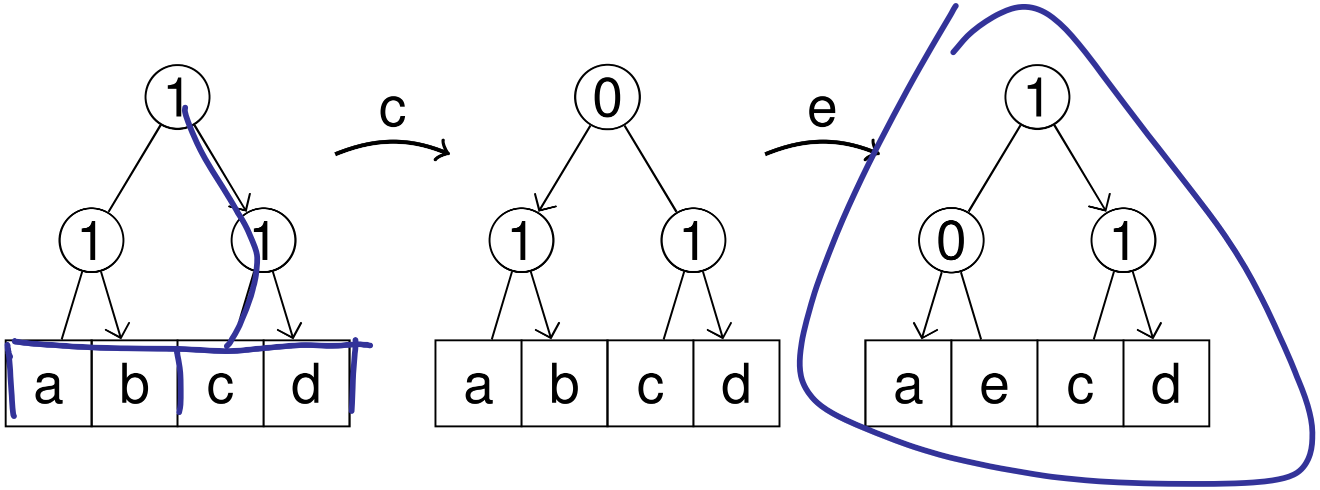


Pseudo-LRU

PLRU

Maintains tree of $k - 1$ "tree-bits" that point to cache line to replace.

- Similar in performance to LRU.
- Cheaper to implement:
  - Only $k - 1$ status bits, versus $\log_2 k!$ status bits for LRU.
  - At most $\log_2 k$ status bits change upon any access + new values independent of previous values.
- "Logical abstraction": order cache lines by order in which misses would replace them
- Example:

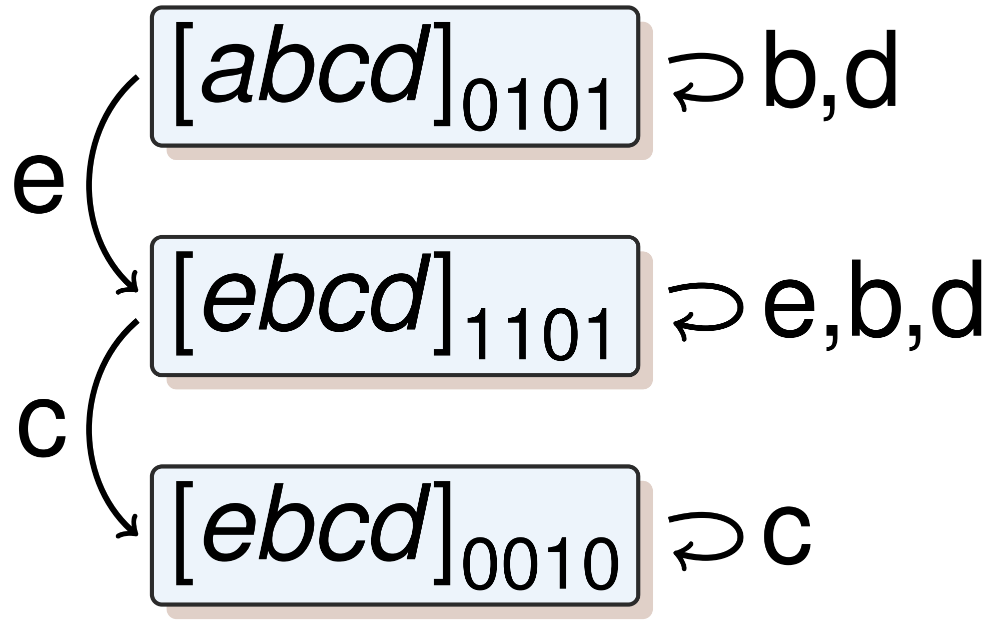


ecad

Most-Recently-Used (also known as Not-Most-Recently-Used ;-))

$PLRU_M$

- Associate one “recently-used” bit with each cache line.
- Replace block in first cache lines whose bit is not set.
- Upon access set bit to 1. Flip all others bits back to 0 when last 0 bit is flipped.
- Example:

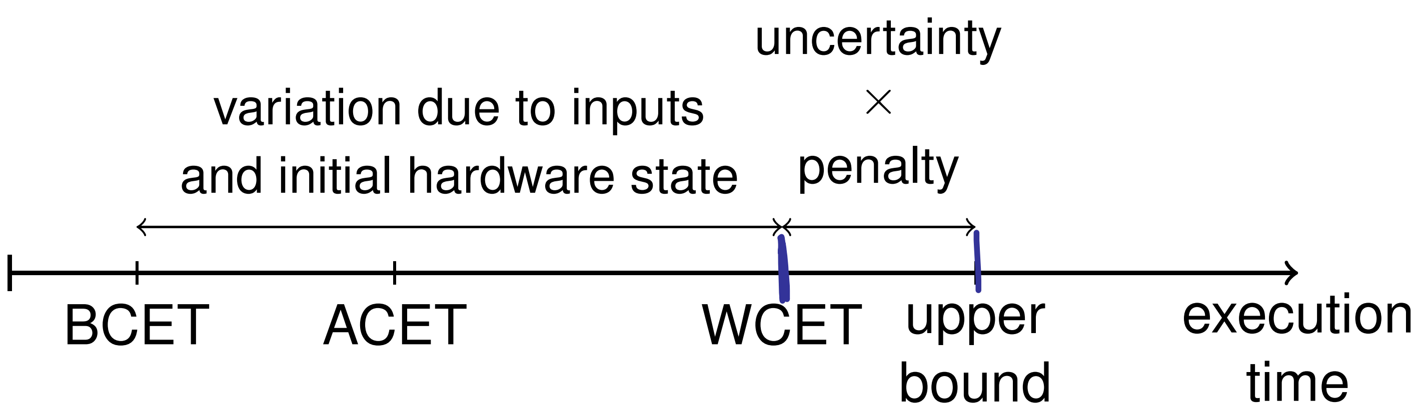


Outline

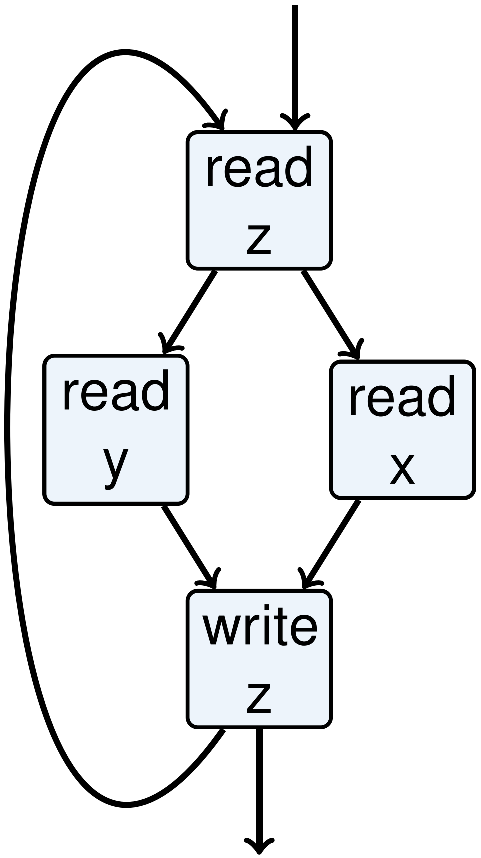
Uncertainty in WCET Analysis

SAARLAND UNIVERSITY
COMPUTER SCIENCE

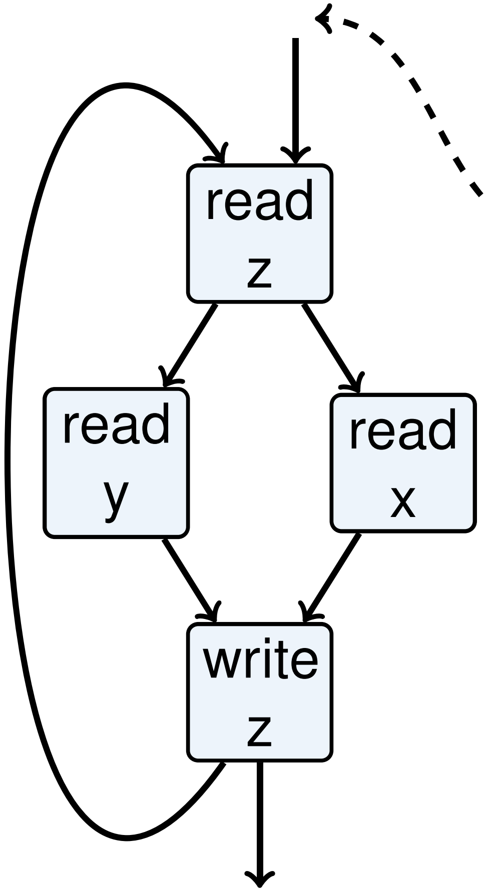
- Amount of uncertainty determines precision of WCET analysis
- Uncertainty in cache analysis depends on replacement policy



Uncertainty in Cache Analysis

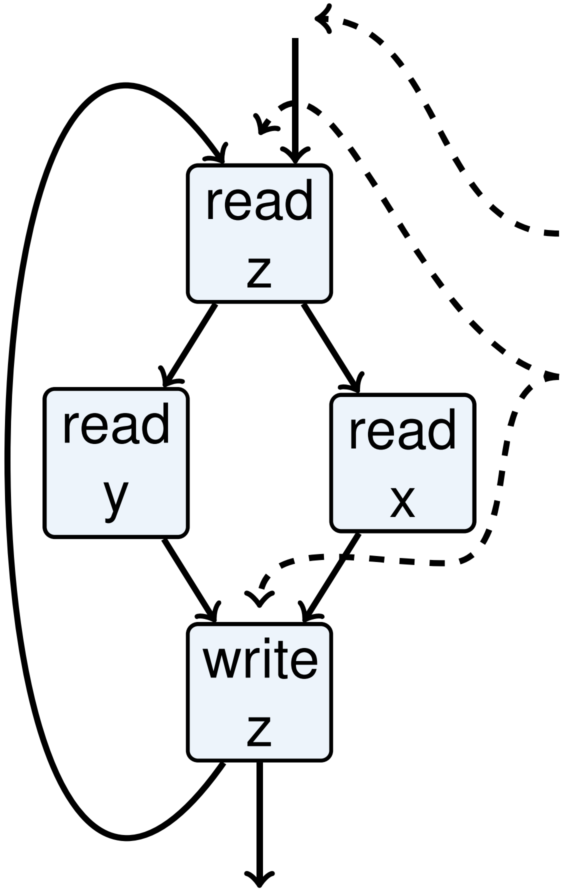


Uncertainty in Cache Analysis



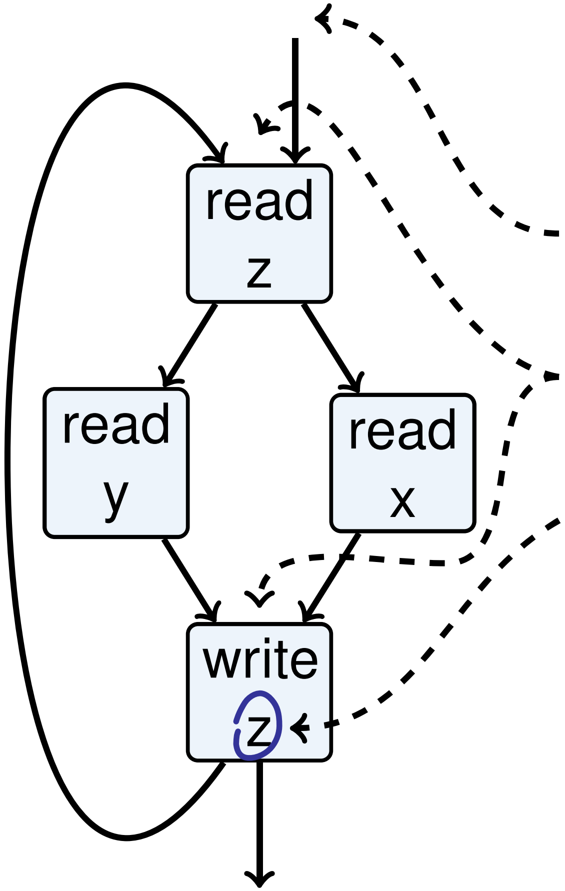
1. Initial cache contents unknown.

Uncertainty in Cache Analysis



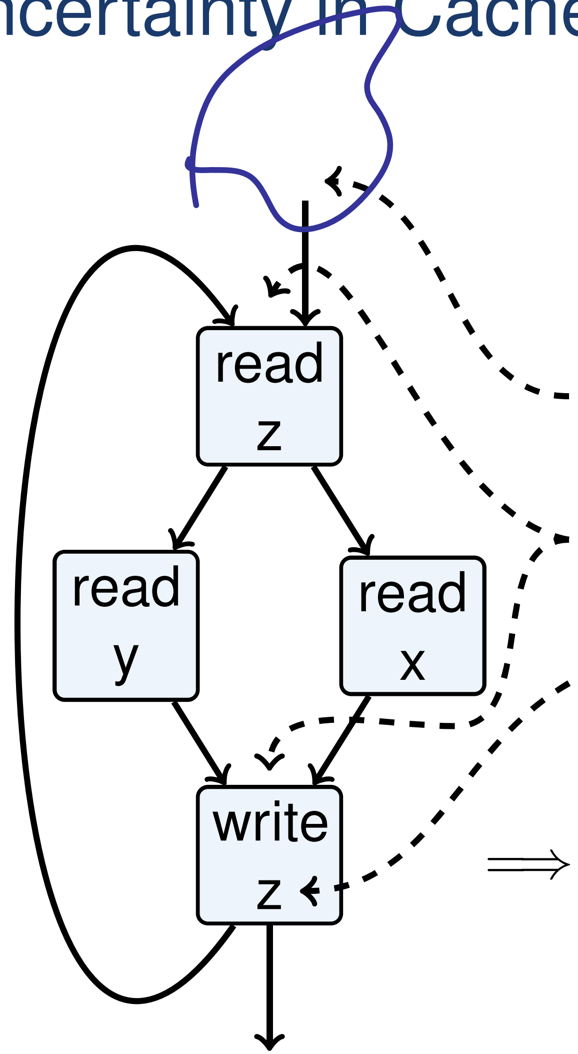
1. Initial cache contents unknown.
2. Need to combine information.

Uncertainty in Cache Analysis



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

Uncertainty in Cache Analysis

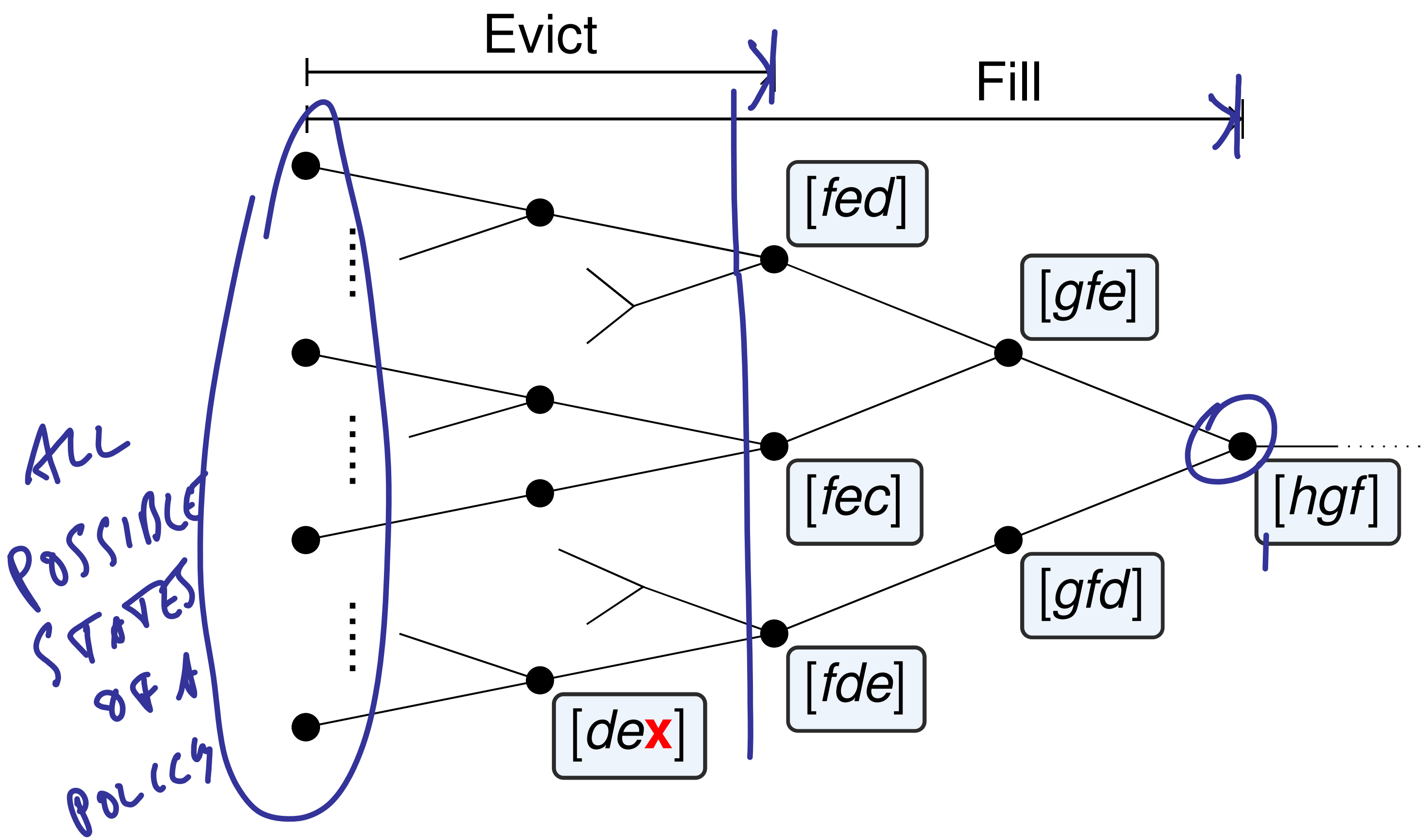


1. Initial cache contents unknown.
2. Need to combine information.
3. Cannot resolve address of *z*.

$\Longrightarrow$ Amount of uncertainty depends on ability to recover information

Predictability Metrics

SAARLAND UNIVERSITY
COMPUTER SCIENCE



Sequence: $\langle a, \ldots, e, \quad f, \quad g, \quad h \rangle$

Meaning of Metrics

SAARLAND UNIVERSITY
COMPUTER SCIENCE

- 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?

$\longrightarrow$ Evict and Fill bound the precision of *any* static cache analysis.

Can thus serve as a benchmark for analyses.

Formalization: May- and Must-Information

SAARLAND UNIVERSITY
COMPUTER SCIENCE

P = POLICY
C_P = SET OF STATES OF P

$$May^{\mathbf{P}}(s) \quad := \quad \bigcup_{p \in C^{\mathbf{P}}} CC_{\mathbf{P}}(\widehat{\text{update}}_{\mathbf{P}}(p, s))$$

"CACHE CONTENTS"

$$Must^{\mathbf{P}}(s) \quad := \quad \bigcap_{p \in C^{\mathbf{P}}} CC_{\mathbf{P}}(\widehat{\text{update}}_{\mathbf{P}}(p, s))$$

a b c

e d g

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

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

$S^{\neq}$: set of finite access sequences with pairwise different accesses

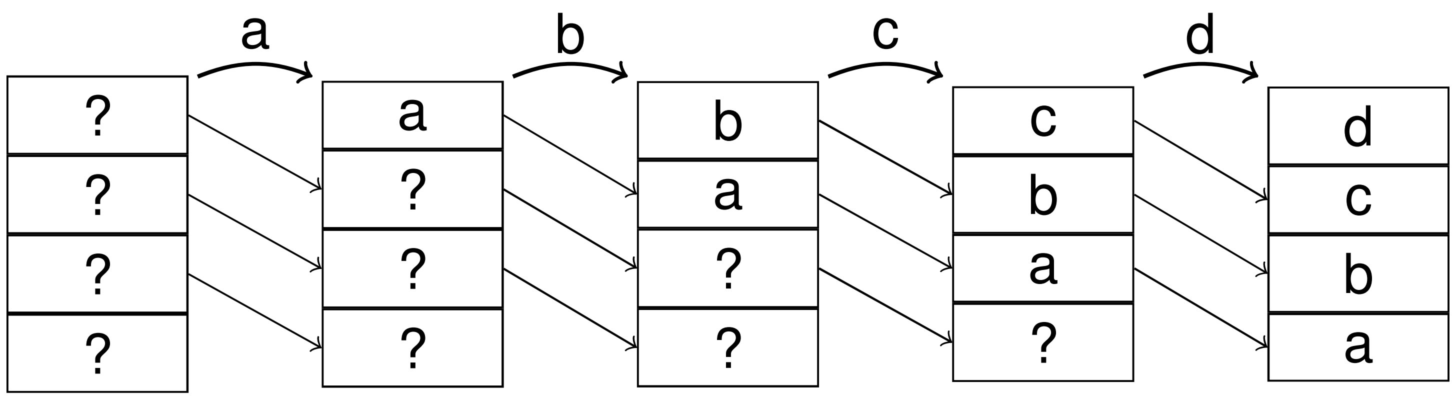
Definitions of Metrics

$$
\begin{aligned}
\mathrm{Evict}^{\mathbf{P}} &:= \min\left\{ n \mid may^{\mathbf{P}}(n) \leq n \right\}, \\
\mathrm{Fill}^{\mathbf{P}} &:= \min\left\{ n \mid must^{\mathbf{P}}(n) = k \right\},
\end{aligned}
$$

where k is **P**'s associativity.

Evaluation of Least-Recently-Used

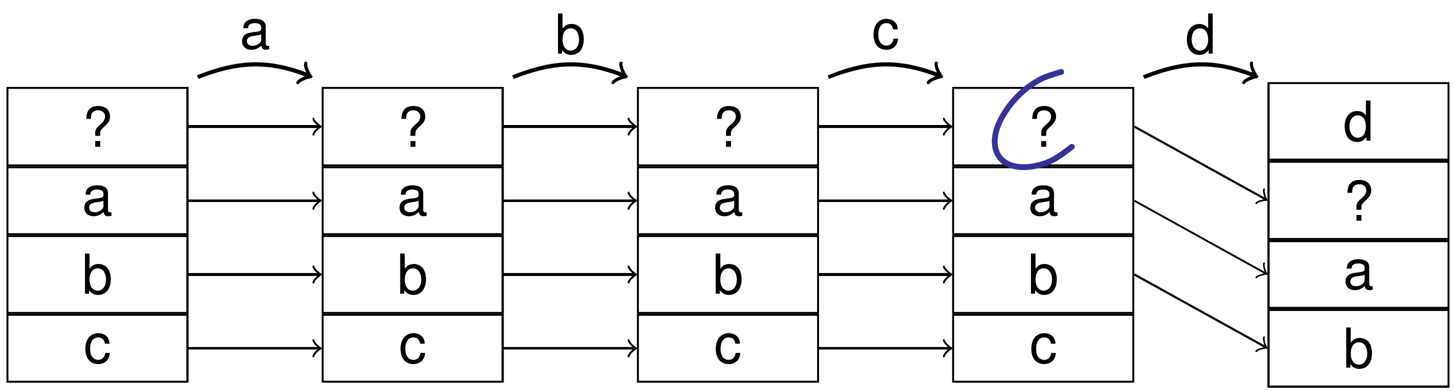
- LRU "forgets" about past quickly:
  - cares about most-recent access to each block only
  - order of previous accesses irrelevant



- 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)

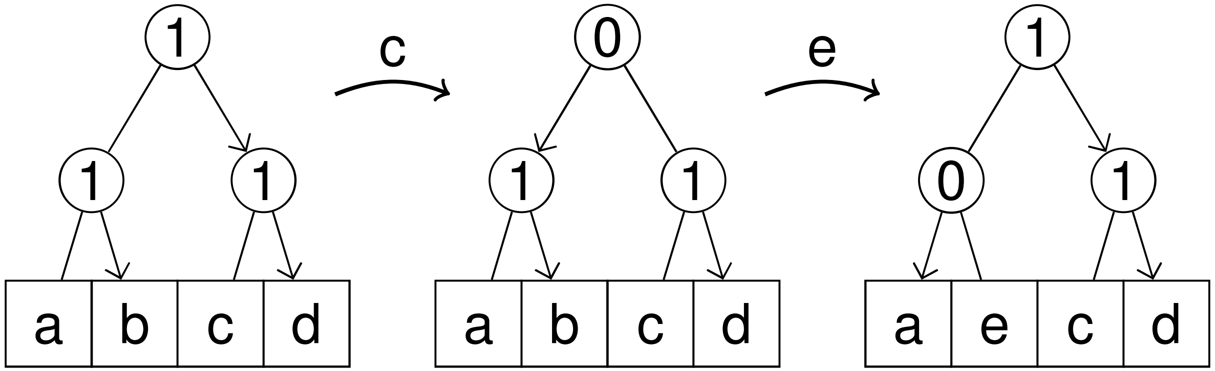
- Like LRU in the miss-case
- But: "Ignores" hits



- In the worst-case $k - 1$ hits and k misses: (k = associativity)
  $\longrightarrow$ Evict(k) $= 2k - 1$
- Another k accesses to obtain complete knowledge:
  $\longrightarrow$ 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:
  - $\text{Evict}(k) = \frac{k}{2}\log_2 k + 1$
  - $\text{Fill}(k) = \frac{k}{2}\log_2 k + k - 1$

Evaluation of MRU (sketch)

Worst-case for Evict:

$$[123\ldots(k-1)k]_{000\ldots01}$$

$$\xrightarrow{a_1 a_2 a_3 \ldots a_{k-1}}$$

$$[a_1 a_2 a_3 \ldots a_{k-2} a_{k-1} k]_{000\ldots010}$$

$$\xrightarrow{b_1 b_2 b_3 \ldots b_{k-2}}$$

$$[b_1 b_2 b_3 \ldots b_{k-2} a_{k-1} k]_{111\ldots110}$$

$$\xrightarrow{b_{k-1}}$$

$$[b_1 b_2 b_3 \ldots b_{k-2} a_{k-1} b_{k-1}]_{000\ldots001}$$

Evaluation of MRU (sketch)

Worst-case for Evict:

$$[123\ldots(k-1){\color{red}k}]_{000\ldots01}$$

$$\xrightarrow{a_1 a_2 a_3 \ldots a_{k-1}} [a_1 a_2 a_3 \ldots a_{k-2} a_{k-1} {\color{red}k}]_{000\ldots010}$$

$$\xrightarrow{b_1 b_2 b_3 \ldots b_{k-2}} [b_1 b_2 b_3 \ldots b_{k-2} a_{k-1} {\color{red}k}]_{111\ldots110}$$

$$\xrightarrow{b_{k-1}} [b_1 b_2 b_3 \ldots b_{k-2} a_{k-1} b_{k-1}]_{000\ldots001}$$

$$\mathit{Evict}(k) = (k-1) + (k-2) + 1 = 2k-2$$

Evaluation of MRU (sketch)

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Worst-case for Evict:

$$[123\ldots(k-1)k]_{000\ldots01}$$

$$\xrightarrow{a_1 a_2 a_3 \ldots a_{k-1}} [a_1 a_2 a_3 \ldots a_{k-2} a_{k-1} k]_{000\ldots010}$$

$$\xrightarrow{b_1 b_2 b_3 \ldots b_{k-2}} [b_1 b_2 b_3 \ldots b_{k-2} a_{k-1} k]_{111\ldots110}$$

$$\xrightarrow{b_{k-1}} [b_1 b_2 b_3 \ldots b_{k-2} a_{k-1} b_{k-1}]_{000\ldots001}$$

$$Evict(k) = (k-1) + (k-2) + 1 = 2k-2$$

What about Fill(k)? ∞

Evaluation of Policies: Summary

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Policy	Evict(k)	Fill(k)	Evict(8)	Fill(8)
LRU	k	k	8	8
FIFO	$2k-1$	$3k-1$	15	23
MRU	$2k-2$	$\infty/3k-4$	14	$\infty/20$
PLRU	$\frac{k}{2}\log_2 k + 1$	$\frac{k}{2}\log_2 k + k - 1$	13	19

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

$\longrightarrow$ Use LRU if predictability is a concern.

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

Outline

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

$$m_{\mathbf{P}}(p, s) \quad = \quad \textit{number of misses that policy } \mathbf{P} \textit{ incurs on access sequence } s \in M^* \textit{ starting in state } p \in C^{\mathbf{P}}$$

Definition – Relative Miss-Competitiveness

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Notation

$$m_{\mathbf{P}}(p, s) \quad = \quad \textit{number of misses that policy } \mathbf{P} \textit{ incurs on access sequence } s \in M^* \textit{ starting in state } p \in C^{\mathbf{P}}$$

Definition (Relative miss competitiveness)

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

$$m_{\mathbf{P}}(p, s) \leq k \cdot m_{\mathbf{Q}}(q, s) + c$$

for all access sequences $s \in M^*$ and cache-set states $p \in C^{\mathbf{P}}$, $q \in C^{\mathbf{Q}}$ that are compatible $p \sim q$.

Definition – Relative Miss-Competitiveness

Notation

$$m_{\mathbf{P}}(p, s) \quad = \quad \textit{number of misses that policy } \mathbf{P} \textit{ incurs on access sequence } s \in M^* \textit{ starting in state } p \in C^{\mathbf{P}}$$

Definition (Relative miss competitiveness)

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

$$m_{\mathbf{P}}(p, s) \leq k \cdot m_{\mathbf{Q}}(q, s) + c$$

for all access sequences $s \in M^*$ and cache-set states $p \in C^{\mathbf{P}}, q \in C^{\mathbf{Q}}$ 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

SAARLAND UNIVERSITY
COMPUTER SCIENCE

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

Example – Relative Miss-Competitiveness

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

Best: **P** is $(1, 0)$-miss-competitive relative to **Q**.

Example – Relative Miss-Competitiveness

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

Best: **P** is $(1, 0)$-miss-competitive relative to **Q**.

Worst: **P** is not-miss-competitive (or ∞-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

SAARLAND UNIVERSITY
COMPUTER SCIENCE

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 $(\frac{2}{3}, 3)$-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.

Worst: **P** is $(0, 0)$-hit-competitive relative to **Q**.
Analogue to ∞-miss-competitiveness.

Local Guarantees: $(1, 0)$-Competitiveness

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

$$m_{\mathbf{P}}(p, s) \leq 1 \cdot m_{\mathbf{Q}}(q, s) + 0$$

$$\Leftrightarrow m_{\mathbf{P}}(p, s) \leq m_{\mathbf{Q}}(q, s)$$

Local Guarantees: $(1, 0)$-Competitiveness

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

$$m_{\mathbf{P}}(p, s) \leq 1 \cdot m_{\mathbf{Q}}(q, s) + 0$$
$$\Leftrightarrow m_{\mathbf{P}}(p, s) \leq m_{\mathbf{Q}}(q, s)$$

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**:

$$m_{\mathbf{P}}(p, s) \leq 1 \cdot m_{\mathbf{Q}}(q, s) + 0$$
$$\Leftrightarrow m_{\mathbf{P}}(p, s) \leq m_{\mathbf{Q}}(q, s)$$

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

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Given: Global guarantees for policy **Q**.
Wanted: Global guarantees for policy **P**.

1. Determine competitiveness of policy **P** relative to policy **Q**.

$$\boxed{\mathbf{m_P \leq k \cdot m_Q + c}}$$

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**.

$$\mathbf{m_P \leq k \cdot m_Q + c}$$

2. Compute global guarantee for task T under policy **Q**.

$$\mathbf{m_Q(T)}$$

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**.

$$\mathbf{m_P \leq k \cdot m_Q + c}$$

2. Compute global guarantee for task T under policy **Q**.

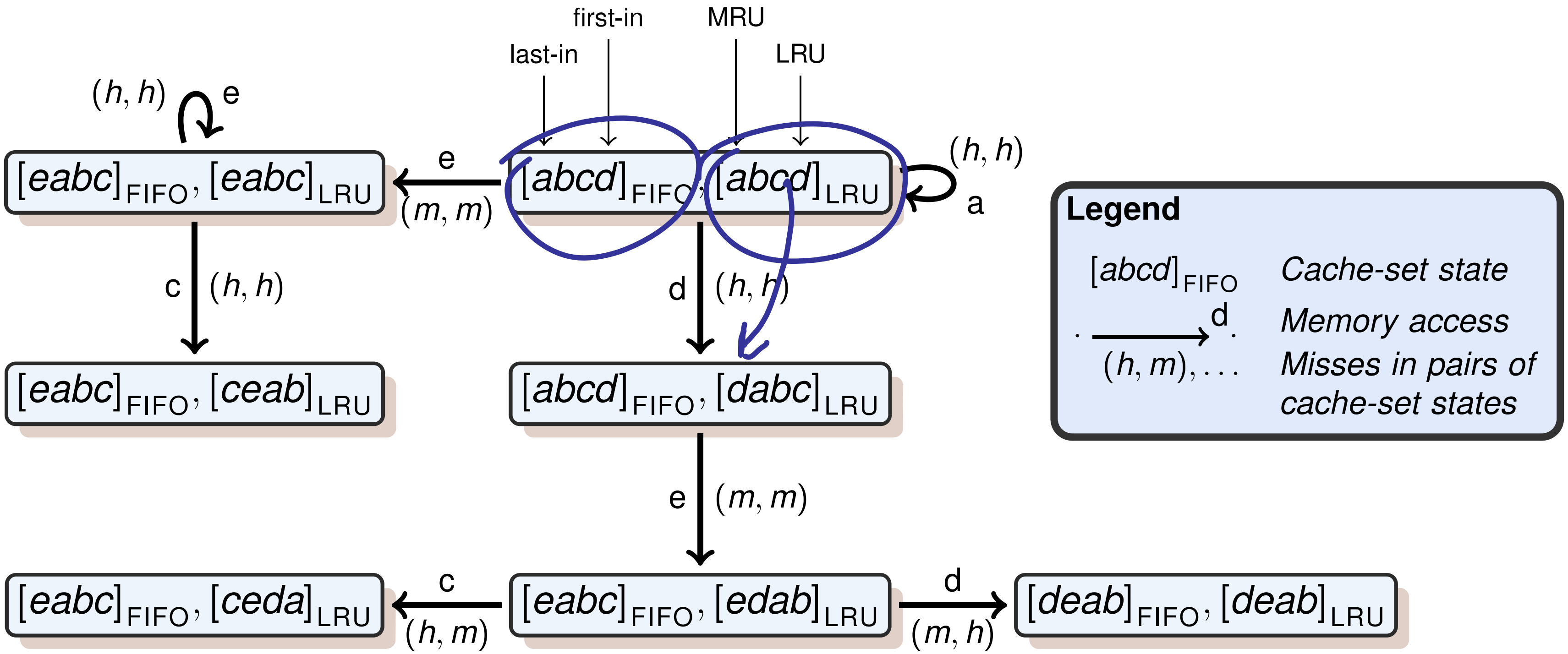
$$\mathbf{m_Q(T)}$$

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**.

$$\mathbf{m_P \leq k \cdot m_Q + c} \quad \mathbf{m_Q(T)} \quad = \quad \mathbf{m_P(T)}$$

Relative Competitiveness: Automatic Computation

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



Competitive miss ratio = maximum ratio of misses in policy **P** to misses in policy **Q** in transition system

Transition System is ∞ Large

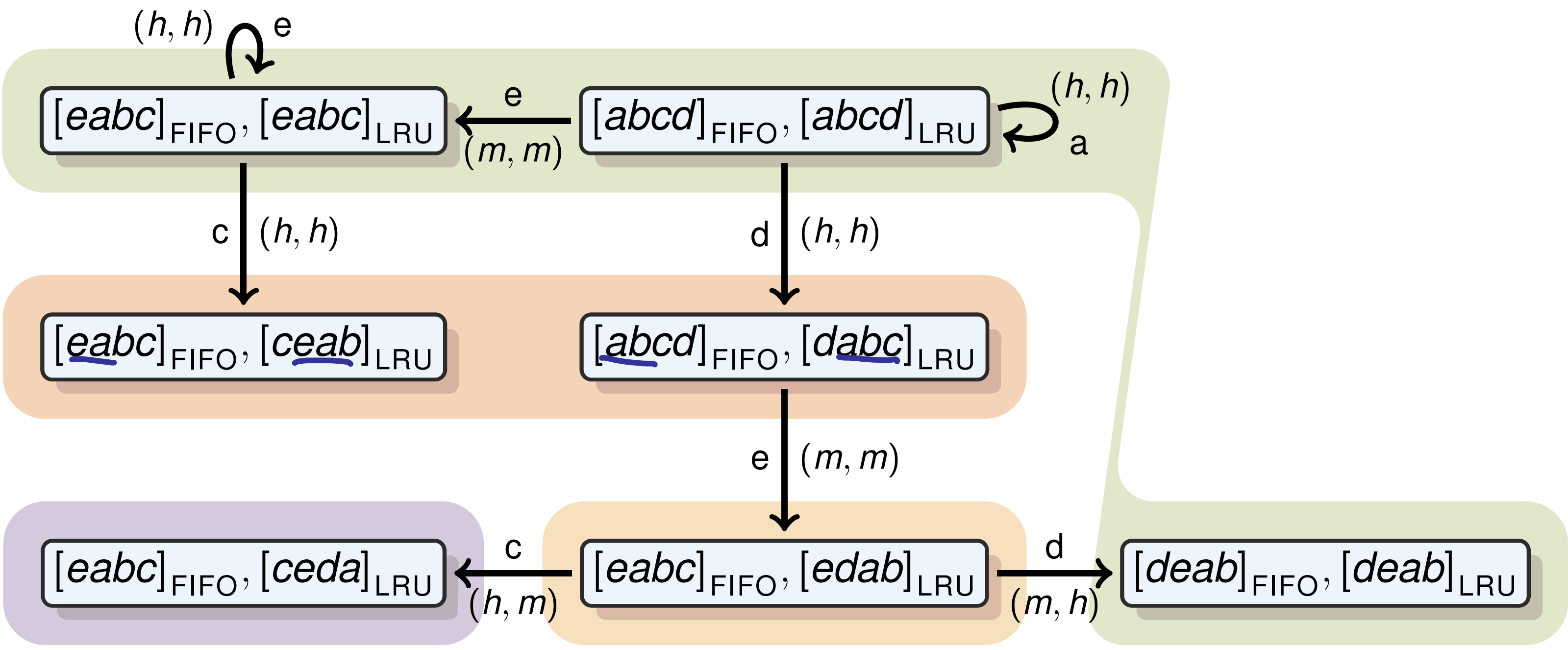
Problem: The induced transition system is ∞ large.

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

$$\begin{tikzcd} {[abc]_{\mathrm{LRU}}, [bde]_{\mathrm{FIFO}}} \arrow[d, "c"', "(h, m)"] & \approx & {[fgl]_{\mathrm{LRU}}, [ghm]_{\mathrm{FIFO}}} \arrow[d, "l"', "(h, m)"] \\ {[cab]_{\mathrm{LRU}}, [cbd]_{\mathrm{FIFO}}} & \approx & {[lfg]_{\mathrm{LRU}}, [lgh]_{\mathrm{FIFO}}} \end{tikzcd}$$

Solution: Construct *finite* quotient transition system.

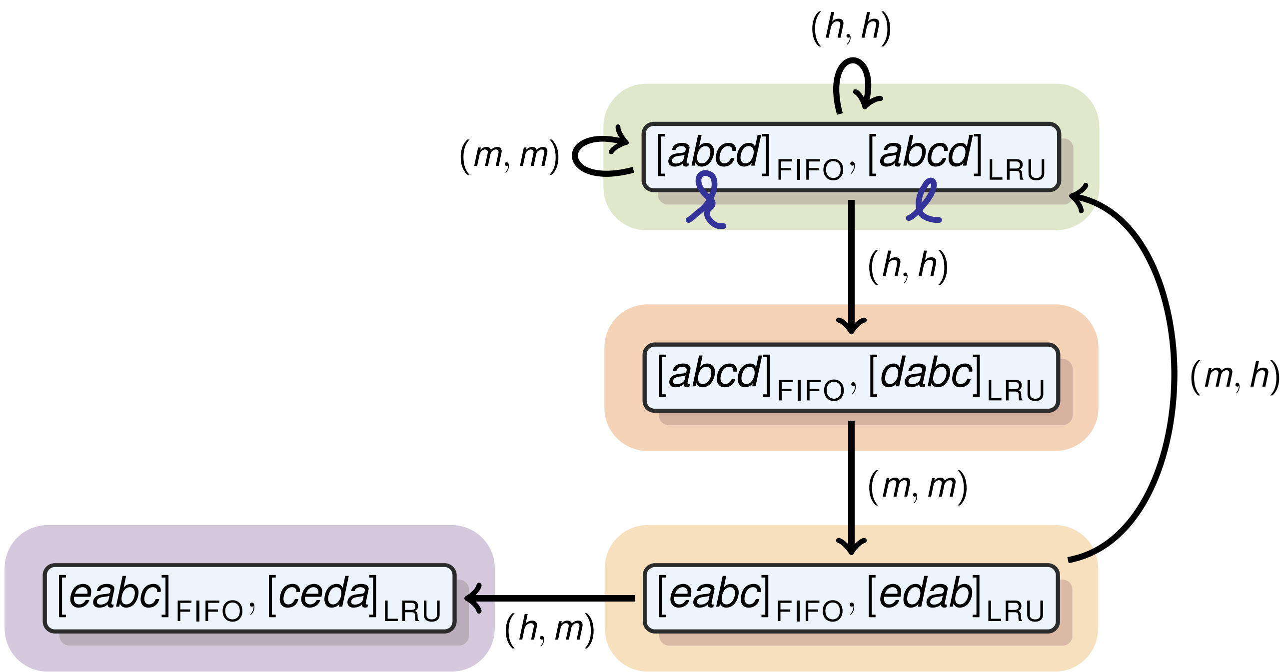
≈-Equivalent States in Running Example



Finite Quotient Transition System

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Merging ≈-equivalent states yields a finite quotient transition system:

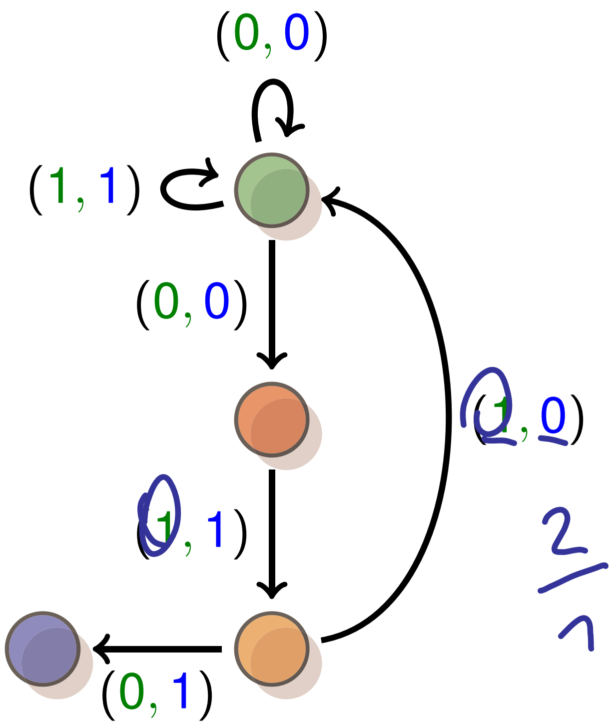


ℓ + ℓ

Competitive Ratio = Maximum Cycle Ratio

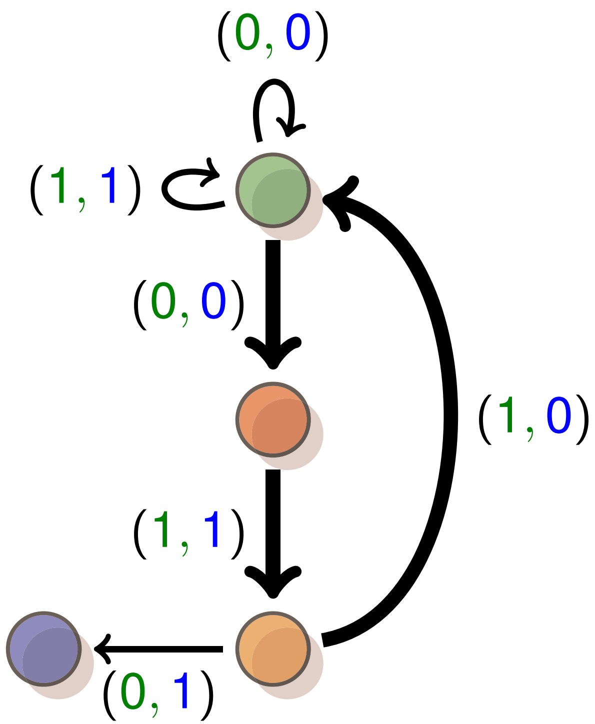
Competitive miss ratio =

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.

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($1 + log_2 k$),
$\longrightarrow$ LRU-*must*-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($1 + log_2 k$),
$\longrightarrow$ LRU-*must*-analysis can be used for PLRU

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

Generalizations

SAARLAND UNIVERSITY
COMPUTER SCIENCE

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

Previously unknown facts:

$$\text{PLRU}(k) \text{ is } (1,0) \text{ comp. rel. to } \text{LRU}(1 + \mathit{log}_2 k),$$

$\longrightarrow$ LRU-*must*-analysis can be used for PLRU

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

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

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

$$\text{LRU}(2k-2) \text{ is } (1,0) \text{ comp. rel. to } \text{MRU}(k).$$

$\longrightarrow$ LRU-*may*-analysis can be used for FIFO and MRU
$\longrightarrow$ optimal with respect to predictability metric Evict

Outline

SAARLAND UNIVERSITY
COMPUTER SCIENCE

More Precise May- and Must Analyses for FIFO

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Relative competitiveness provides a may-analysis for FIFO, but no must-analysis.

Three ideas:

- Construct must-analysis based on may-analysis results:
  After a miss, a block is guaranteed to stay in the cache for longer.
- Predict misses more quickly by taking into account prior miss predictions.
  $\longrightarrow$ Grund/Reineke, Static Analysis Symposium (SAS) 2009.
- Predict hits more quickly by observing "phase behavior".
  $\longrightarrow$ Grund/Reineke, Euromicro Conference on Real-Time Systems (ECRTS) 2010.

Outline

Summary

Predictability Metrics

...quantify the predictability of replacement policies.

$\longrightarrow$ LRU is the most predictable policy.

Summary

Predictability Metrics

...quantify the predictability of replacement policies.

$\longrightarrow$ LRU is the most predictable policy.

Relative Competitiveness

...allows to derive guarantees on cache performance,

...yields *may*- and *must*-analyses for FIFO, PLRU and MRU.

Summary

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Predictability Metrics

. . . quantify the predictability of replacement policies.

$\longrightarrow$ LRU is the most predictable policy.

Relative Competitiveness

. . . allows to derive guarantees on cache performance,

. . . yields *may*- and *must*-analyses for FIFO, PLRU and MRU.

Specialized Analyses for FIFO

. . . improve precision over pure relative competitiveness approach.

. . . but may become very complex.

Relation: Pred. Metrics $\leftrightarrow$ Rel. Competitiveness

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Let $P(k)$ be $(1, 0)$-miss-competitive relative to policy $Q(l)$, then

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

(ii) $mls^P(k) \geq mls^Q(l)$.

Alternative Pred. Metrics $\leftrightarrow$ Rel. Competitiveness

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Let l be the smallest associativity, such that $\mathrm{LRU}(l)$ 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 $\mathrm{LRU}(l)$. Then

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

Size of Transition System

$$\underbrace{2^{l+l'}}_{\substack{\text{status bits}\\ \text{of } \mathbf{P} \text{ and } \mathbf{Q}}} \cdot \underbrace{\sum_{i=0}^{k} \binom{k}{i}}_{\text{non-empty lines in } \mathbf{P}} \cdot \underbrace{\sum_{i'=0}^{k'} \binom{k'}{i'}}_{\text{non-empty lines in } \mathbf{Q}} \cdot \underbrace{\sum_{j=0}^{\min\{i,i'\}} \binom{i}{j}\binom{i'}{j} j!}_{\substack{\text{number of overlappings}\\ \text{in non-empty lines}}}$$

$$\begin{aligned}
\sum_{j=0}^{\min\{k,k'\}} \binom{k}{j}\binom{k'}{j} j! \quad &\leq \quad k! \cdot k'! \sum_{j=0}^{\min\{k,k'\}} \frac{1}{(k-j)!j!(k'-j)!} \\
&\leq \quad k! \cdot k'! \sum_{j=0}^{\infty} \frac{1}{j!} = e \cdot k! \cdot k'!
\end{aligned}$$

This can be bounded by

$$2^{l+l'+k+k'} \leq |(C_k^l \times C_{k'}^{l'})/\approx| \leq 2^{l+l'+k+k'} \cdot \underbrace{e \cdot k! \cdot k'!}_{\text{bound on number of overlappings}}$$

Compatible States

$$\begin{tikzcd}
\boxed{i^{\mathbf{P}} = [\perp\perp\perp\perp]_{\mathbf{P}}} \arrow[r, phantom, "\approx"] \arrow[d, squiggly, "\widehat{\mathrm{update}}_{\mathbf{P}}(i^{\mathbf{P}}, s)"] & \boxed{i^{\mathbf{Q}} = [\perp\perp\perp\perp]_{\mathbf{Q}}} \arrow[d, squiggly, "\widehat{\mathrm{update}}_{\mathbf{Q}}(i^{\mathbf{Q}}, s)"] \\
\boxed{p} \arrow[r, phantom, "\approx"] & \boxed{q}
\end{tikzcd}$$

$(1, 0)$-Competitiveness and May/Must-Analyses

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

$$
\begin{tikzcd}
\boxed{p} \arrow[dd, "{m_{\mathbf{P}}(p, \langle x \rangle) = 1}"'] & \approx & \boxed{q} \arrow[dd, "{m_{\mathbf{Q}}(q, \langle x \rangle) = 1}"] \\
 & \Longrightarrow & \\
\boxed{p'} & \approx & \boxed{q'}
\end{tikzcd}
$$

$(1,0)$-Competitiveness and May/Must-Analyses

SAARLAND UNIVERSITY
COMPUTER SCIENCE

$$
\begin{tikzcd}
C^{\mathbf{P}} \arrow[d, "S"'] \arrow[r, phantom, "\approx"] & C^{\mathbf{Q}} \arrow[d, "S"] \\
P \arrow[d] \arrow[r, phantom, "\approx"] & Q \arrow[d] \\
P' \arrow[r, phantom, "\approx"] & Q'
\end{tikzcd}
$$

$$\forall p \in P : m_{\mathbf{P}}(p, \langle x \rangle) = 1 \quad \Longrightarrow \quad \forall q \in Q : m_{\mathbf{Q}}(q, \langle x \rangle) = 1$$