# Precise and Efficient FIFO-Replacement Analysis Based on Static Phase Detection

Daniel Grund[1] Jan Reineke[2]

[1]Saarland University, Saarbrücken, Germany

[2]University of California, Berkeley, USA

Euromicro Conference on Real-Time Systems 2010

SAARLAND UNIVERSITY

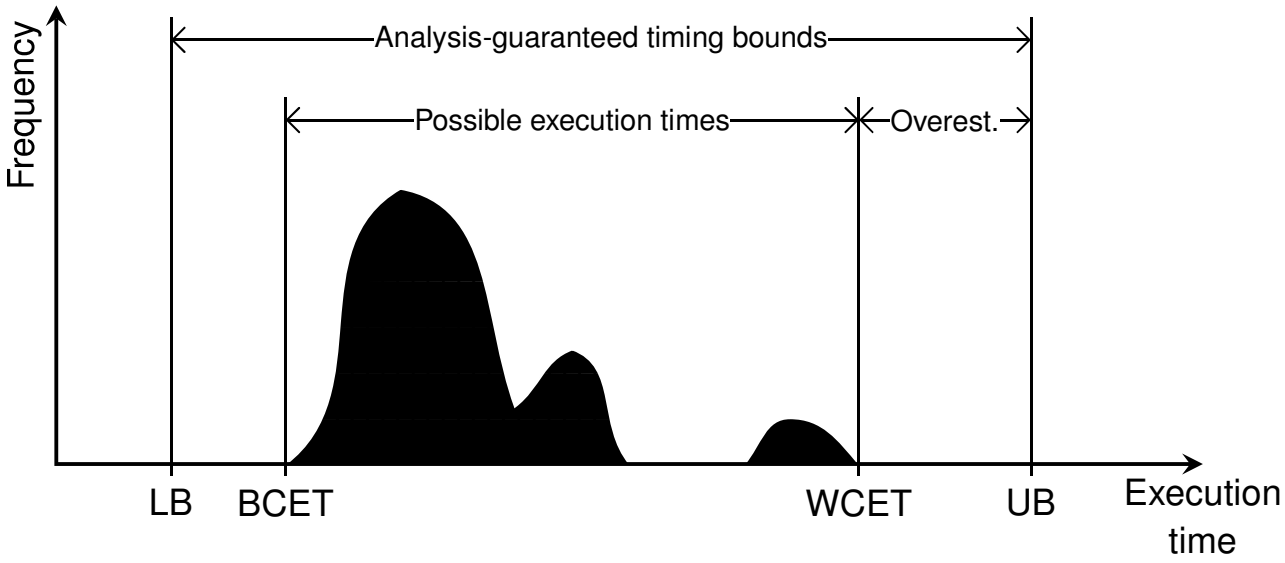
COMPUTER SCIENCE

# Outline

SAARLAND UNIVERSITY
COMPUTER SCIENCE

# Timing Analysis for Real-Time Systems

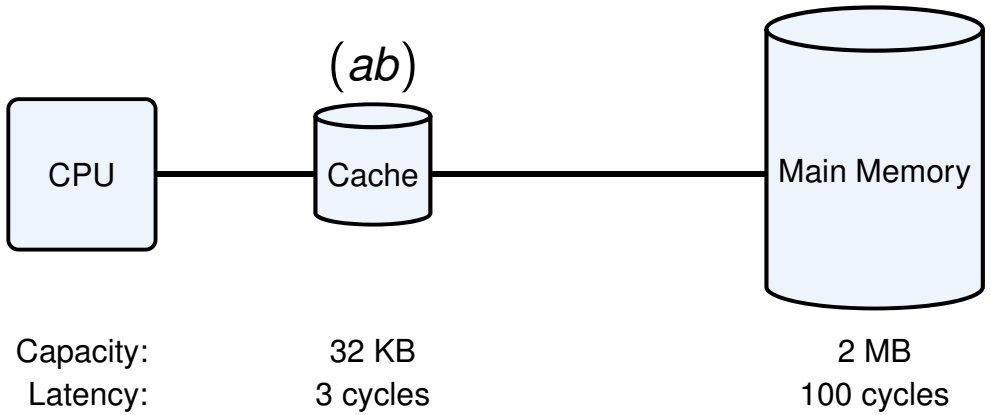
SAARLAND UNIVERSITY
COMPUTER SCIENCE




- Need to bound execution time of programs
- Execution time influenced by architectural features
  - pipelines, caches, branch prediction, ...
- Need to analyze behavior of architectural components

# Caches and Replacement Policies

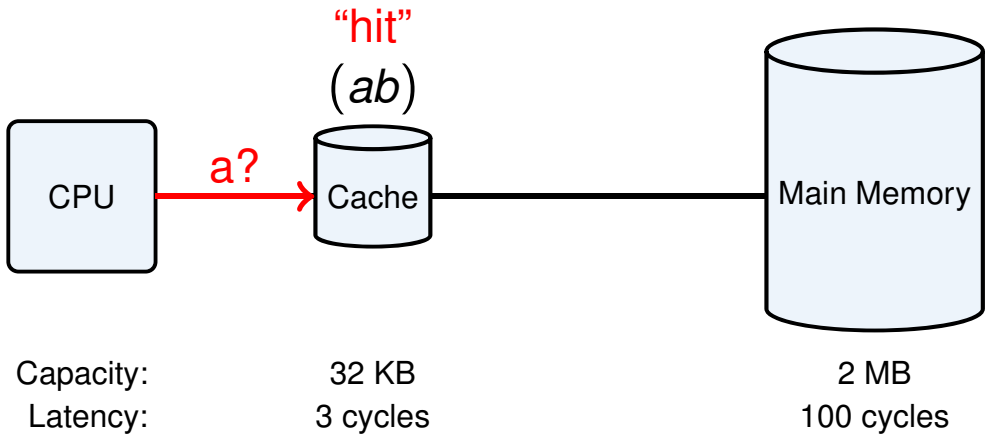





- Caches transparently buffer memory blocks

# Caches and Replacement Policies

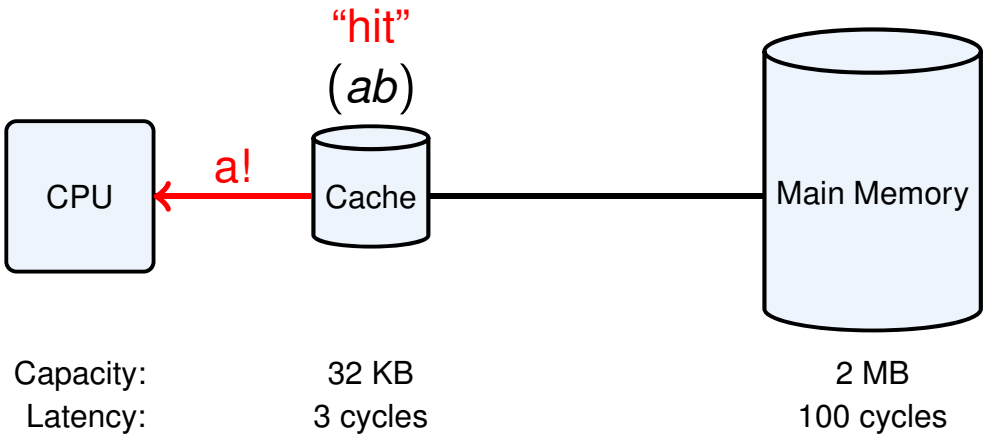
SAARLAND UNIVERSITY
COMPUTER SCIENCE




- Caches transparently buffer memory blocks

# Caches and Replacement Policies

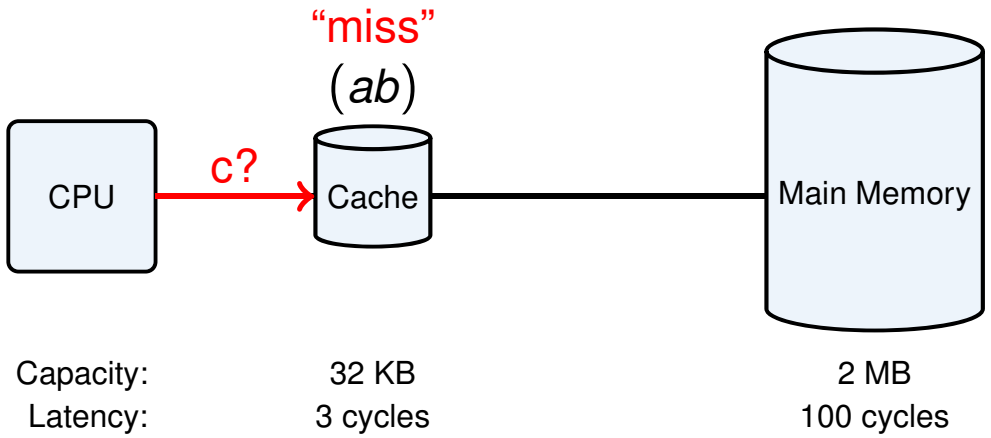
SAARLAND UNIVERSITY
COMPUTER SCIENCE




- Caches transparently buffer memory blocks

# Caches and Replacement Policies

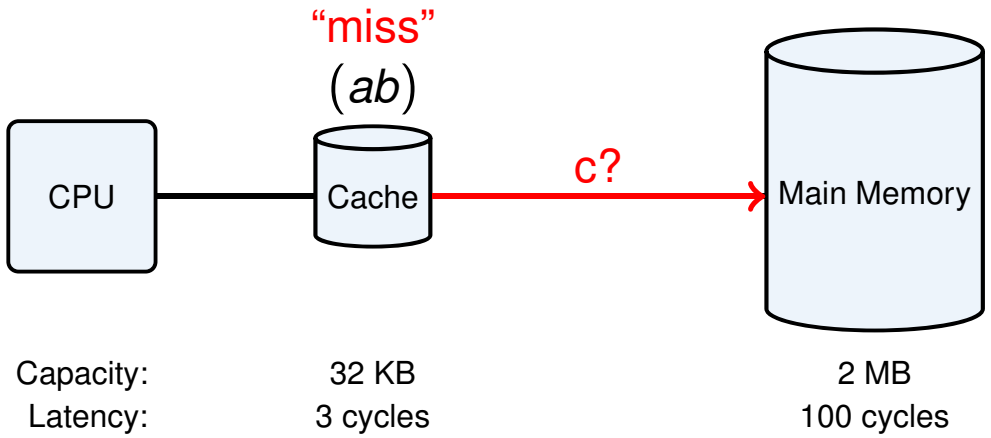
SAARLAND UNIVERSITY
COMPUTER SCIENCE




- Caches transparently buffer memory blocks

# Caches and Replacement Policies

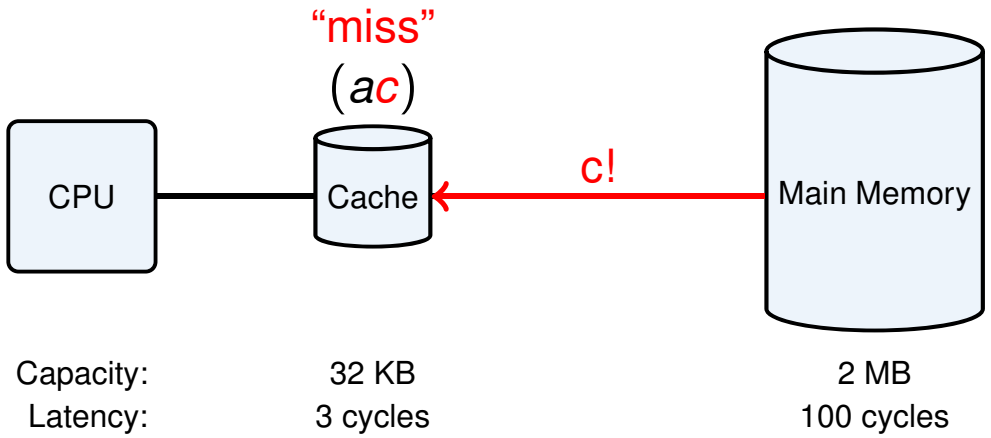
SAARLAND UNIVERSITY
COMPUTER SCIENCE




- Caches transparently buffer memory blocks

# Caches and Replacement Policies

SAARLAND UNIVERSITY
COMPUTER SCIENCE

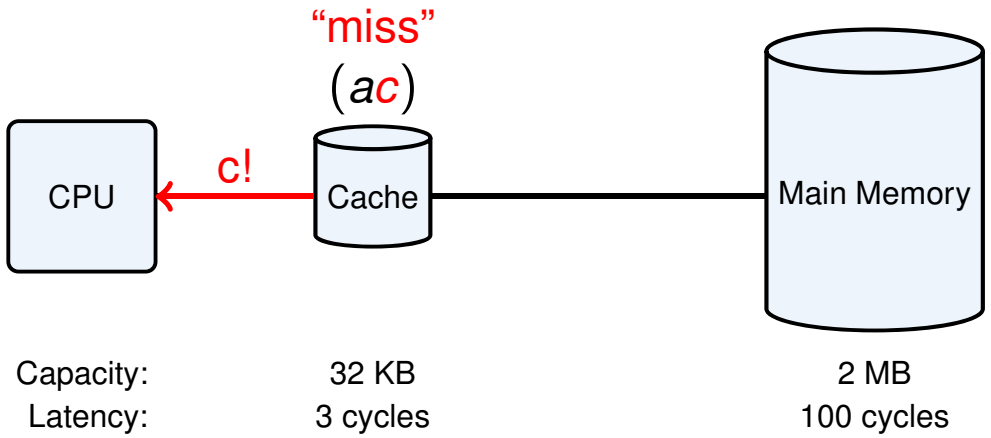



- Caches transparently buffer memory blocks
- Replacement policy *dynamically* decides which element to replace

| | |
|---|---|
| LRU | least recently used |
| PLRU | pseudo LRU |
| FIFO | first-in first-out |

# Caches and Replacement Policies

SAARLAND UNIVERSITY
COMPUTER SCIENCE




- Caches transparently buffer memory blocks
- Replacement policy *dynamically* decides which element to replace

  LRU least recently used
  PLRU pseudo LRU
  FIFO first-in first-out

# Static Cache Analysis

## Goals & Notions

SAARLAND UNIVERSITY
COMPUTER SCIENCE

- Derive approximations to cache contents at each program point
- in order to classify memory accesses as cache hits or cache misses

### Must-information

- Underapproximation of cache contents
- Used to soundly classify cache hits

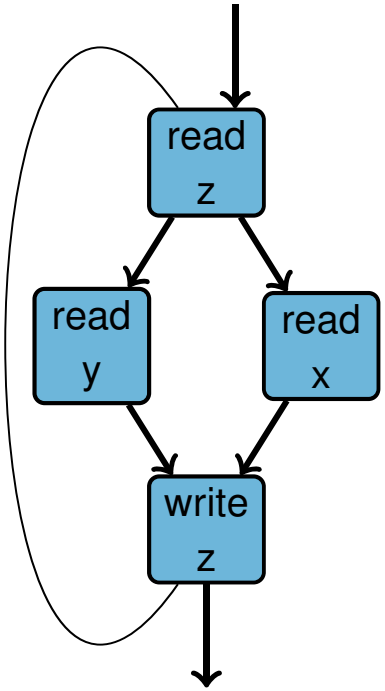
### May-information

- Overapproximation of cache contents
- Used to soundly classify cache misses

# Static Cache Analysis

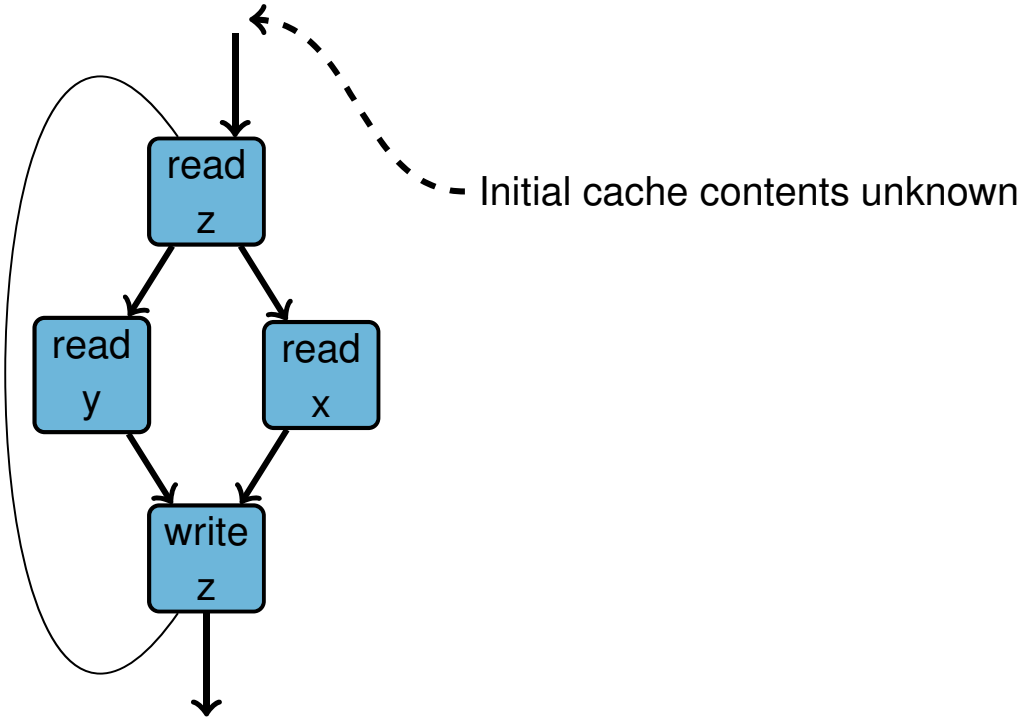
## Challenges

SAARLAND UNIVERSITY
COMPUTER SCIENCE



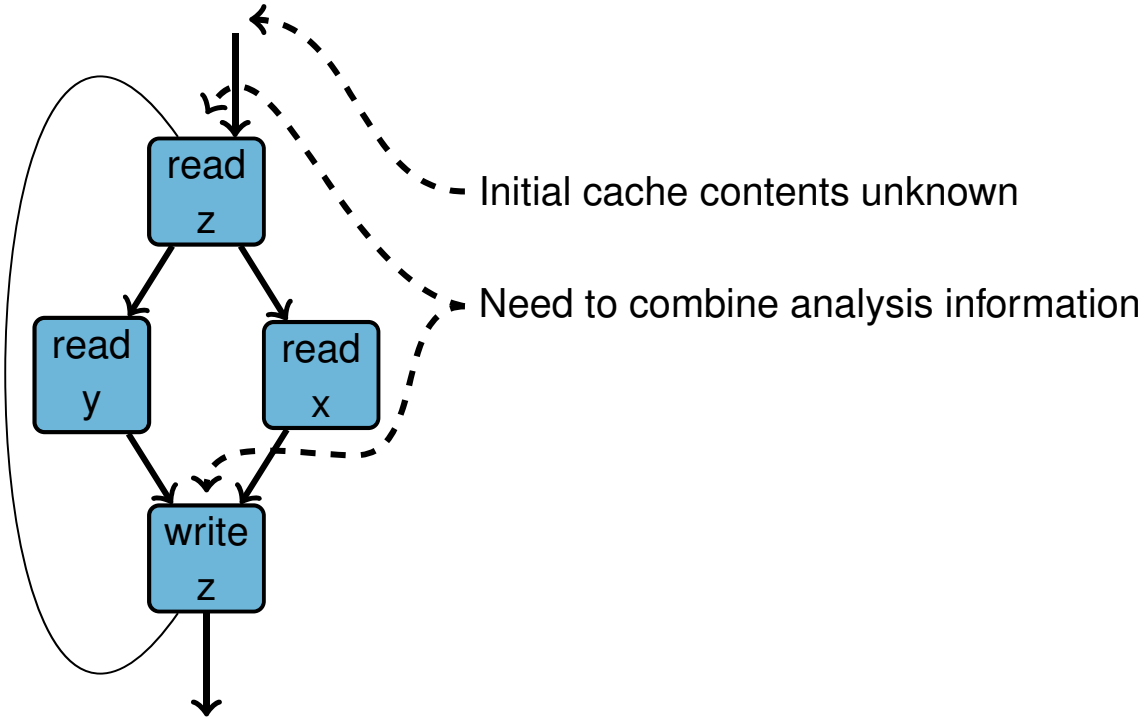
# Static Cache Analysis

## Challenges



# Static Cache Analysis

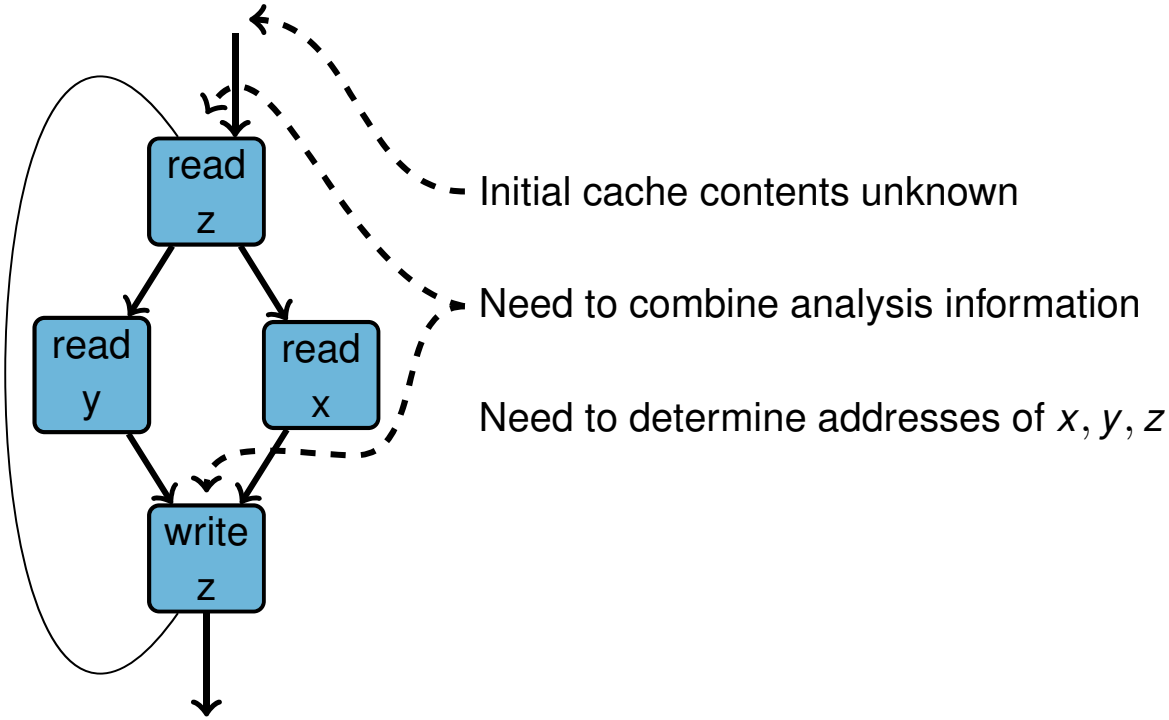
## Challenges



# Static Cache Analysis

## Challenges

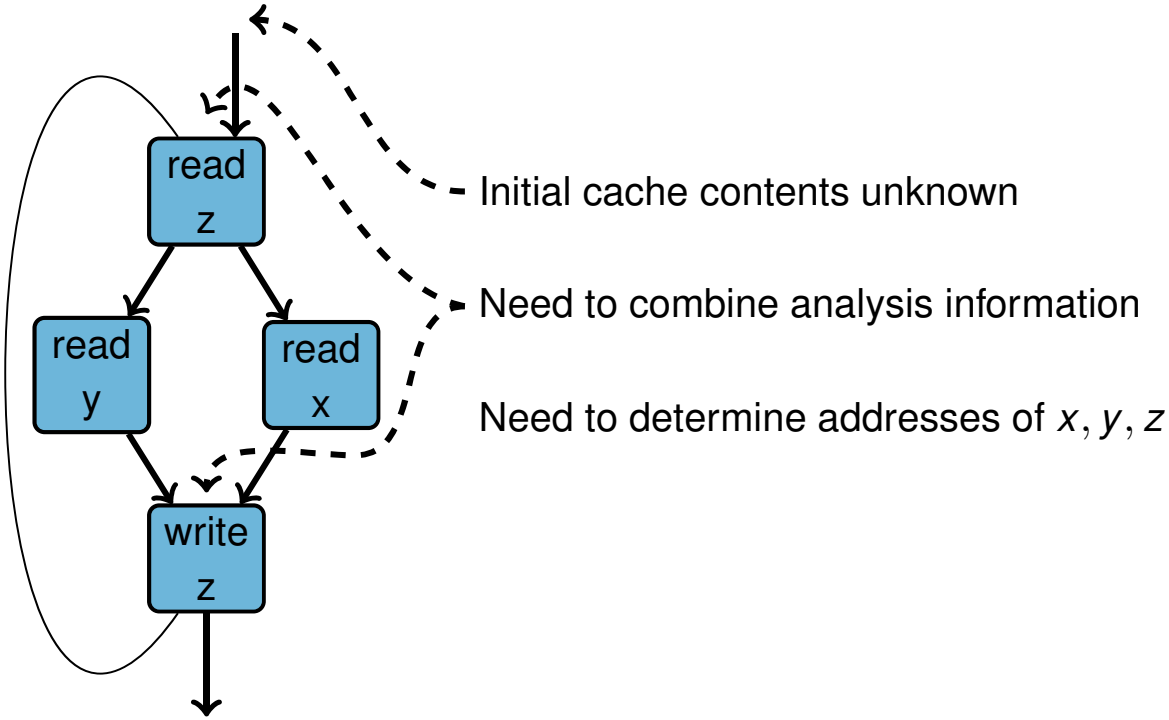
SAARLAND UNIVERSITY
COMPUTER SCIENCE



# Static Cache Analysis

## Challenges

SAARLAND UNIVERSITY
COMPUTER SCIENCE



# FIFO Replacement

SAARLAND UNIVERSITY
COMPUTER SCIENCE

- FIFO cache of size $k$:

$$\underset{\text{last-in}}{\downarrow} \quad \underset{\text{first-in}}{\downarrow}$$
$$[b_1, \ldots, b_k] \in \mathcal{Q}_k := \mathcal{B}_\perp^k$$

- Example updates:

$$[d, c, b, a] \xrightarrow[hit]{c} [d, c, b, a]$$

$$[d, c, b, a] \xrightarrow[miss]{e} [e, d, c, b]$$

# FIFO Replacement Analysis

## Why Predicting Hits is Difficult

SAARLAND UNIVERSITY
COMPUTER SCIENCE

- Take a set of blocks $B$ that does fit into a cache $q$
- For example, $B = \{a, b, e\}$ and $k = 4$. $|B| \leq k$.
- Access all blocks in $B$:

$$q \xrightarrow{\langle a,b,e \rangle} q'$$

- Must all accessed blocks be cached? $\forall q : B \subseteq q'$?

# FIFO Replacement Analysis

## Why Predicting Hits is Difficult

SAARLAND UNIVERSITY
COMPUTER SCIENCE

- Take a set of blocks $B$ that does fit into a cache $q$
- For example, $B = \{a, b, e\}$ and $k = 4$. $|B| \leq k$.
- Access all blocks in $B$:

$$q \xrightarrow{\langle a,b,e \rangle} q'$$

- Must all accessed blocks be cached? $\forall q : B \subseteq q'$? No.

$$[d, c, b, a] \xrightarrow[hit]{a} [d, c, b, a] \xrightarrow[hit]{b} [d, c, b, a] \xrightarrow[miss]{e} [e, d, c, b] \not\ni a$$

### Observation

After accessing a set of "fitting" blocks,
not all of them must be cached.

# FIFO Replacement Analysis

## Why Predicting Misses is Difficult

SAARLAND UNIVERSITY
COMPUTER SCIENCE

- Take a set of blocks $B$ that does not fit into a cache $q$
- For example, $B = \{a, b, c, d, e, f\}$ and $k = 4$. $|B| \geq k$.
- Access all blocks in $B$:

$$q \xrightarrow{\langle a,b,c,d,e,f \rangle} q'$$

- Must all non-accessed blocks be evicted? $\forall q : q' \subseteq B$?

# FIFO Replacement Analysis

## Why Predicting Misses is Difficult

SAARLAND UNIVERSITY
COMPUTER SCIENCE

- Take a set of blocks $B$ that does not fit into a cache $q$
- For example, $B = \{a, b, c, d, e, f\}$ and $k = 4$. $|B| \geq k$.
- Access all blocks in $B$:

$$q \xrightarrow{\langle a,b,c,d,e,f \rangle} q'$$

- Must all non-accessed blocks be evicted? $\forall q : q' \subseteq B$? No.

$$[x, c, b, a] \xrightarrow[hits]{\langle a,b,c \rangle} [x, c, b, a] \xrightarrow[misses]{\langle d,e,f \rangle} [f, e, d, x] \ni x$$

### Observation

After accessing a set of "non-fitting" blocks, other blocks may still be cached.

# Outline

SAARLAND UNIVERSITY
COMPUTER SCIENCE

# To the point: Anticipation & Idea

SAARLAND UNIVERSITY
COMPUTER SCIENCE

- Considering repeated accesses to “fitting” blocks $B$ helps:
  $B = \{a, b, c\}$
  $s = \langle a, b, b, c, b, b, a, c, c, a, b, \ldots \rangle$
  Eventually, all blocks in $B$ must be cached.

- Need to detect repetitions
- Partition access sequence $s$ into phases

**Definition (Phase)**

A *B*-phase is an access sequence $s$ such that the *set* of accessed blocks $A(s) = B$.

$$\langle \underbrace{a\ ,\ b\ ,\ b\ ,\ c}_{\{a,b,c\}\text{-phase}}, \underbrace{b\ ,\ b\ ,\ a\ ,\ c}_{\{a,b,c\}\text{-phase}}, \ldots \rangle$$

# Predicting Hits by Detecting Phases

SAARLAND UNIVERSITY
COMPUTER SCIENCE

## Lemma (Single Phase)

Let $s$ be a $B$-phase and $|B| \leq k$.

$$\forall q \in \mathcal{Q}_k, q \xrightarrow{s} q' : \qquad \vee$$

# Predicting Hits by Detecting Phases

SAARLAND UNIVERSITY
COMPUTER SCIENCE

## Lemma (Single Phase)

Let $s$ be a $B$-phase and $|B| \leq k$.

$$\forall q \in \mathcal{Q}_k, q \xrightarrow{s} q' : \quad B \subseteq q' \vee$$

1. Either all blocks already cached:
   - $B \subseteq q \Rightarrow$ only hits in $s \Rightarrow B \subseteq q'$

# Predicting Hits by Detecting Phases

SAARLAND UNIVERSITY
COMPUTER SCIENCE

## Lemma (Single Phase)

Let $s$ be a $B$-phase and $|B| \leq k$.

$$\forall q \in \mathcal{Q}_k, q \xrightarrow{s} q' : \quad B \subseteq q' \vee C_1(q') \subseteq B$$

1. Either all blocks already cached:
   - $B \subseteq q \Rightarrow$ only hits in $s \Rightarrow B \subseteq q'$
2. Or not:
   - $B \not\subseteq q \Rightarrow$ at least one miss $s \Rightarrow C_1(q') \subseteq B$
   - $[d, c, b, a] \xrightarrow{\langle a,b,e \rangle} [\underbrace{e}_{C_1(q')=\{e\}\subseteq B}, d, c, b]$

# Predicting Hits by Detecting Phases

SAARLAND UNIVERSITY
COMPUTER SCIENCE

## Theorem (Multiple Phases)

Let $s_i$ be $B$-phases and $|B| \leq k$ and $s = s_1 \circ \ldots \circ s_j$

$$\forall q \in \mathcal{Q}_k, q \xrightarrow{s} q' : \quad B \subseteq q' \vee C_j(q') \subseteq B$$

1. For each individual phase the lemma applies
2. Misses, if any, accumulate in last-in positions $C_j(q')$

$$[d, c, b, a] \xrightarrow{\langle a,b,e \rangle} [\underbrace{e}_{C_1 \subseteq B}, d, c, b] \xrightarrow{\langle b,a,e \rangle} [\underbrace{a, e}_{C_2 \subseteq B}, d, c] \xrightarrow{\langle a,b,e \rangle} [\underbrace{b, a, e}_{C_3 \subseteq B}, d]$$

# Predicting Hits by Detecting Phases

SAARLAND UNIVERSITY
COMPUTER SCIENCE

## Theorem (Multiple Phases)

Let $s_i$ be $B$-phases and $|B| \leq k$ and $s = s_1 \circ \ldots \circ s_j$

$$\forall q \in \mathcal{Q}_k, q \xrightarrow{s} q' : \quad B \subseteq q' \vee C_j(q') \subseteq B$$

1. For each individual phase the lemma applies
2. Misses, if any, accumulate in last-in positions $C_j(q')$

$$[d,c,b,a] \xrightarrow{\langle a,b,e\rangle} [\underbrace{e}_{C_1 \subseteq B}, d, c, b] \xrightarrow{\langle b,a,e\rangle} [\underbrace{a,e}_{C_2 \subseteq B}, d, c] \xrightarrow{\langle a,b,e\rangle} [\underbrace{b,a,e}_{C_3 \subseteq B}, d]$$

- Corollary: After $|B|$ $B$-phases, all blocks in $B$ must be cached

# The Must Analysis

## How to Count Phases

SAARLAND UNIVERSITY
COMPUTER SCIENCE

- For phase blocks $B$, the analysis maintains:
  - $P$ phase progress, blocks already accessed in current phase
  - $pc$ phase counter, number of detected $B$-phases
- Predicts hits for blocks in $B$ if $pc = |B|$

**Example for $B = \{a, b\}$**

| $s$ | | $a$ | $b$ | | $b$ | $b$ | $a$ | | $b$ |
|---|---|---|---|---|---|---|---|---|---|
| $P$ | $\emptyset$ | $\{a\}$ | $\{a,b\}$ | $\emptyset$ | $\{b\}$ | $\{b\}$ | $\{a,b\}$ | $\emptyset$ | $\{b\}$ |
| $pc$ | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 2 | 2 |
| Hit | | | | | | | | | Hit |

# The Must Analysis

## Dependency on Future Accesses

SAARLAND UNIVERSITY
COMPUTER SCIENCE

- Need $|B|$ $B$-phases to predict hits for blocks in $B$
- How to choose $B$?
- After observing $\langle a, b, c \rangle$ it makes sense trying to detect
  - 2 further $\{a, b, c\}$-phases
  - 1 further $\{b, c\}$-phase
  - 0 further $\{c\}$-phases
- Optimal $B$ depends on future accesses
  - $\langle a, b, c,$ $a, b, c,$ $a, b, c,$ $a, b, c \rangle$
  - $\langle a, b, c,$ $b, c,$ $b, c,$ $b, c,$ $b, c \rangle$

# The Must Analysis

## Resolving the Dependency

SAARLAND UNIVERSITY
COMPUTER SCIENCE

- Perform multiple analyses for different $B$ sets
- For which?

- $|B|$ already determines sensible contents of $B$
- For $|B| = 2$, after $\langle a, b, c \rangle$
  - already detected 1 $\{b, c\}$-phase
  - no advantage in trying to detect 2 $\{x, y\}$-phases

$\Rightarrow$ Perform $k$ analyses for different $B$ sets
  - for each phase size $n = 1 \ldots k$
  - $B_n$ consists of the $n$ most-recently-used blocks

# The Must Analysis

## Subanalyses for $n = 1 \ldots 3$

SAARLAND UNIVERSITY
COMPUTER SCIENCE

$$a \quad b \quad c \quad c \quad b \quad c \quad a \quad a \quad c \quad a \quad b \quad a$$

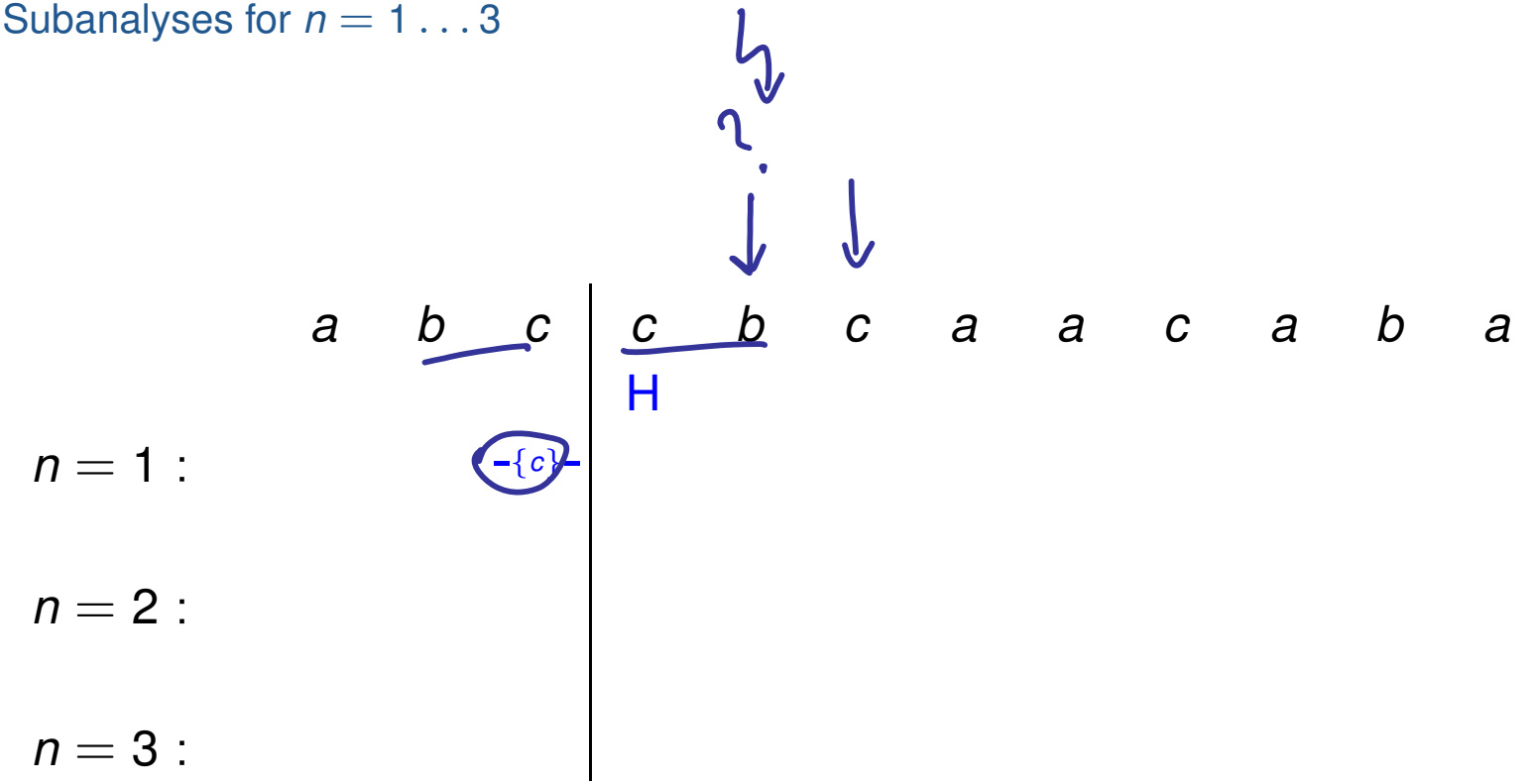
$n = 1$ :

$n = 2$ :

$n = 3$ :

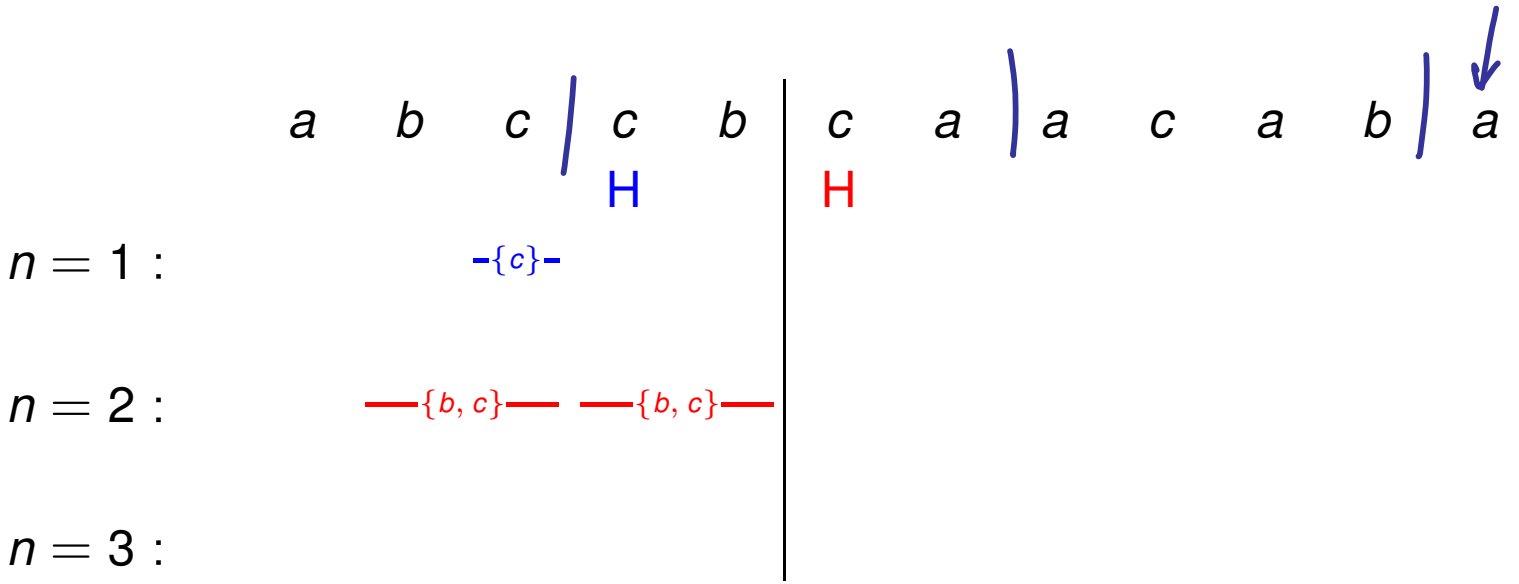
# The Must Analysis

## Subanalyses for $n = 1 \ldots 3$



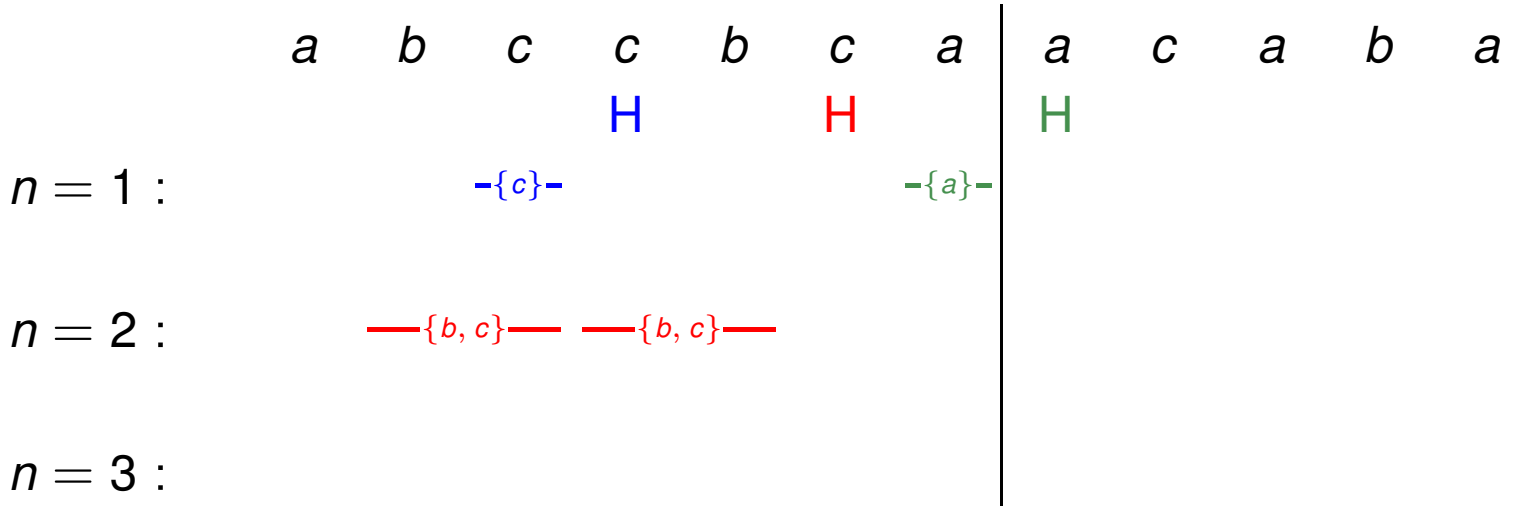
# The Must Analysis

## Subanalyses for $n = 1 \ldots 3$



# The Must Analysis

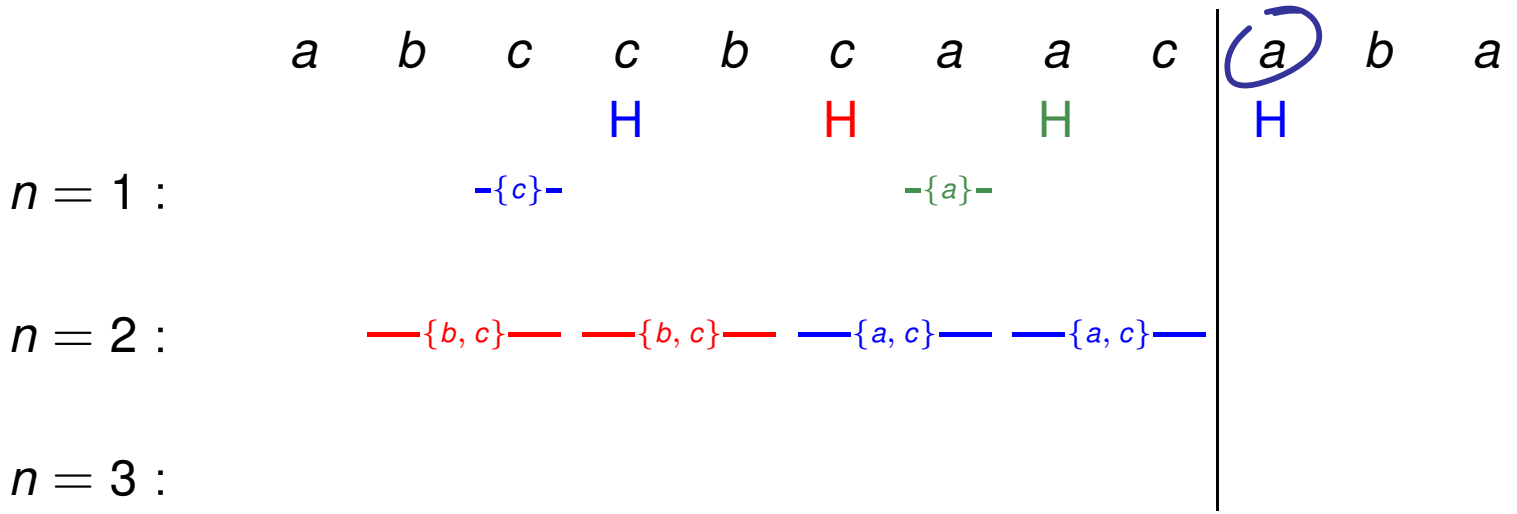
## Subanalyses for $n = 1 \ldots 3$



# The Must Analysis

## Subanalyses for $n = 1 \ldots 3$

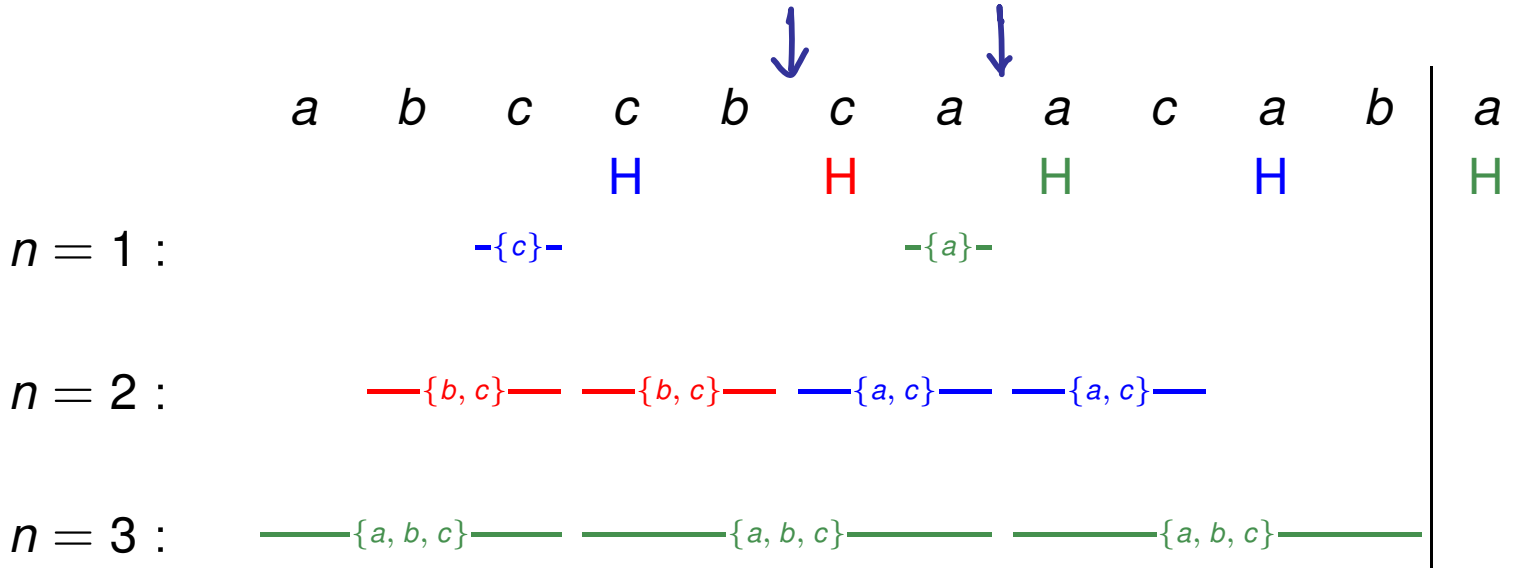
SAARLAND UNIVERSITY
COMPUTER SCIENCE



# The Must Analysis

## Subanalyses for $n = 1 \ldots 3$

SAARLAND UNIVERSITY
COMPUTER SCIENCE



# Efficient Implementation

## Observation

SAARLAND UNIVERSITY
COMPUTER SCIENCE

- For $n = 1 \ldots k$, analysis needs to maintain:
  - phase blocks $B_n \in 2^{\mathcal{B}}$
  - phase progress $P_n \in 2^{\mathcal{B}}$
  - phase counter $pc_n \in \mathbb{N}$

- Phase blocks $B_n$ are the $n$ most-recently-used blocks

$\Rightarrow$ For $i < j : B_i \subseteq B_j$

$\Rightarrow$ Encode all $B_n$ in a single LRU-stack

- For all $i : P_i \subseteq B_i$

$\Rightarrow$ Encode all $P_n$ as “pointers” into the stack

# Efficient Implementation

## Encoding

SAARLAND UNIVERSITY
COMPUTER SCIENCE

- For phase blocks $B_n$:
  - $pc_n$ complete $B_n$-phases were detected
  - current phase progress is $B_{pp_n}$

| | |
|---|---|
| $B_1$ | $pc_1, pp_1$ |
| $B_2 \setminus B_1$ | $pc_2, pp_2$ |
| $B_3 \setminus B_2$ | $pc_3, pp_3$ |
| $B_4 \setminus B_3$ | $pc_4, pp_4$ |

| | |
|---|---|
| $B_1$ | $\{b\}$ |
| $P_1$ | $\emptyset$ |
| $pc_1$ | 1 |
| $B_2$ | $\{b, c\}$ |
| $P_2$ | $\{b\}$ |
| $pc_2$ | 2 |
| $B_3$ | $\{a, b, c\}$ |
| $P_3$ | $\{b, c\}$ |
| $pc_3$ | 1 |

=

| | pc pp |
|---|---|
| $\{b\}$ | 1, 0 |
| $\{c\}$ | 2, 1 |
| $\{a\}$ | 1, 2 |
| $\emptyset$ | 0, 3 |

# Outline

SAARLAND UNIVERSITY
COMPUTER SCIENCE

# Contents of the Paper

SAARLAND UNIVERSITY
COMPUTER SCIENCE

- So far, we have seen parts of the must-analysis
- The paper contains, for must- and may-analysis,
  - basic theorem
  - generalization to arbitrary control-flow
  - formalization as abstract interpretation

# Outline

SAARLAND UNIVERSITY
COMPUTER SCIENCE

# Brief History of Replacement Analysis

SAARLAND UNIVERSITY
COMPUTER SCIENCE

| | |
|---|---|
| Before ’97 | LRU analyses |
| LCTRTS’97 | Precise and efficient must- and may-analysis for LRU [1] |
| LCTES’08 | Generic analyses for FIFO and PLRU [2] |
| SAS’09 | Cache analysis framework and FIFO analysis [3] |
| WCET’10 | Toward precise analysis for PLRU [4] |
| ECRTS’10 | Precise and efficient must- and may-analysis for FIFO |

# Evaluation Setup

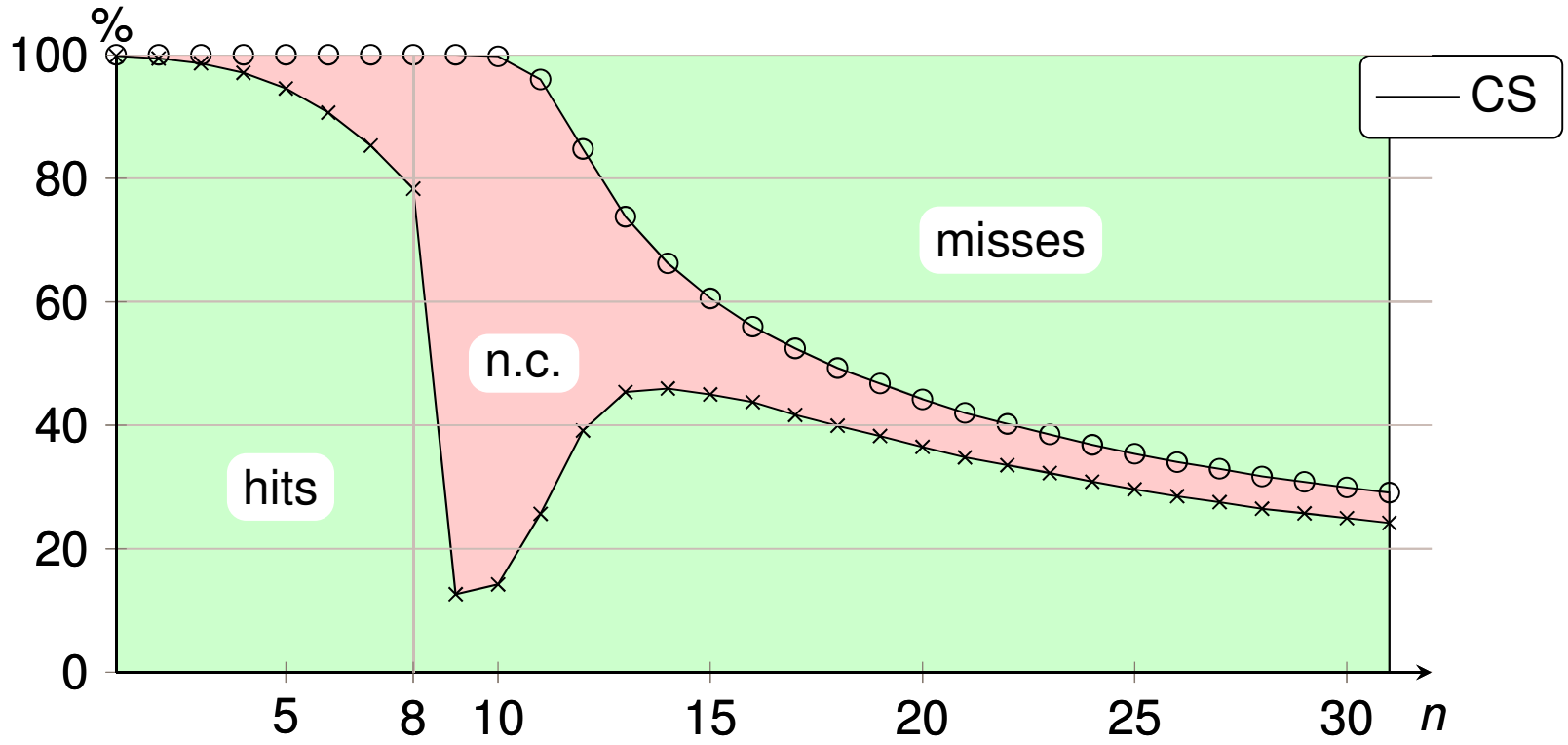
SAARLAND UNIVERSITY
COMPUTER SCIENCE

- Analyses:
  - HAM Must-analysis of SAS'09
  - RC Generic analyses of LCTES'08
  - PD Phase detecting analyses
- Collecting semantics:
  - CS Limit for any static analysis
- Spectrum of synthetic benchmarks:
  - Random access sequences and program fragments
  - With varying locality

# Evaluation Results

*k*=8, random sequences

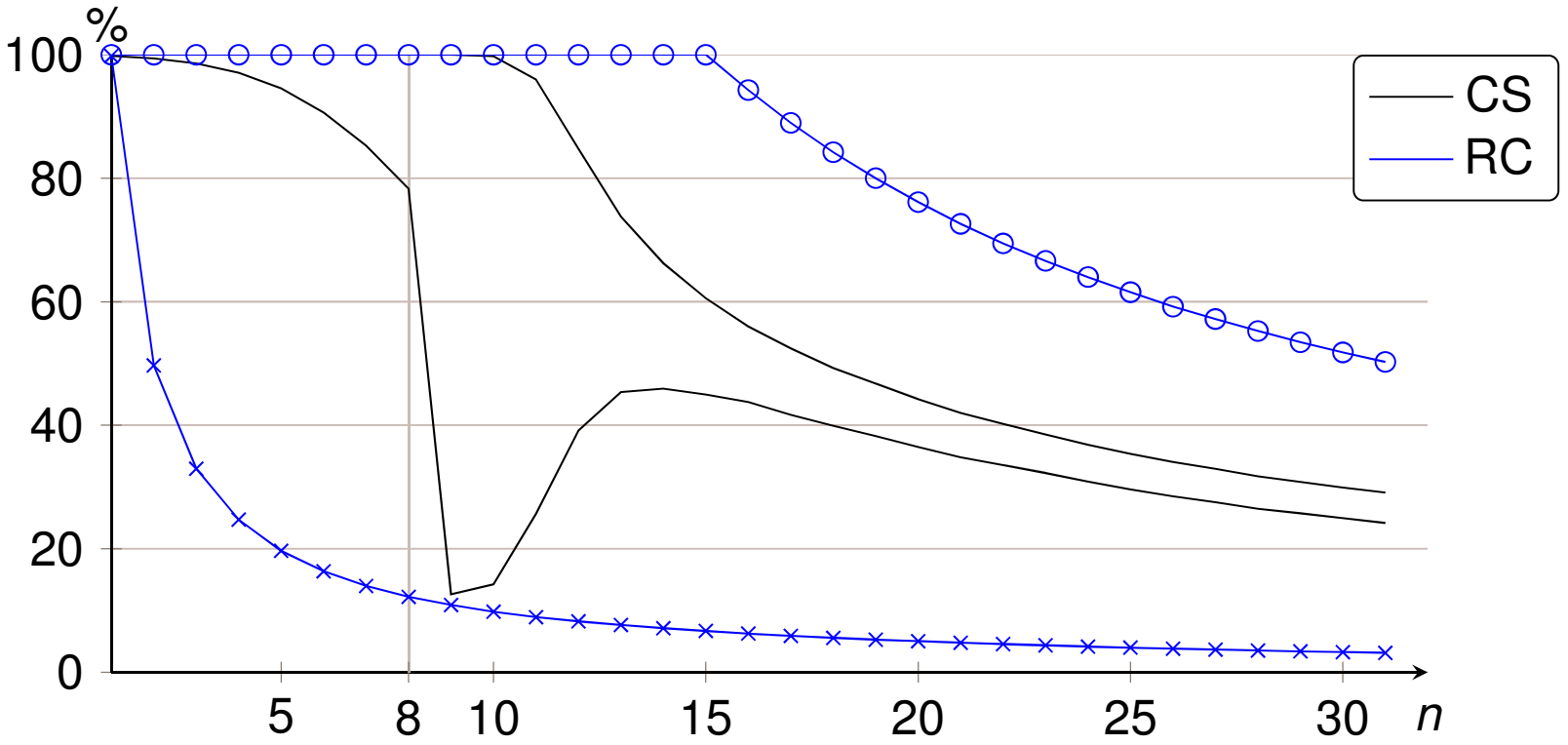
SAARLAND UNIVERSITY
COMPUTER SCIENCE




- *n* is number of distinct elements that get accessed
- Average guaranteed hit- and miss-rates

# Evaluation Results

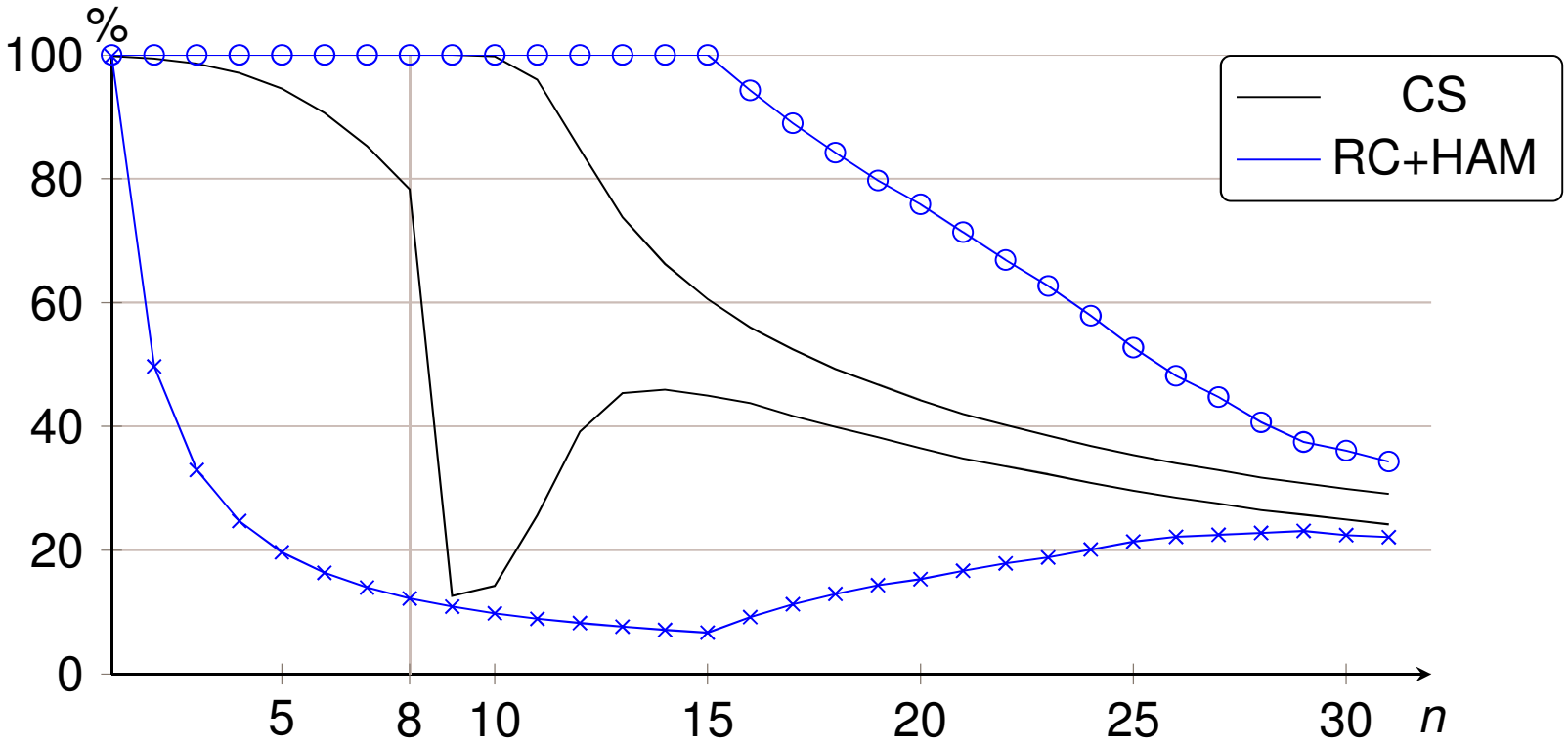
*k*=8, random sequences




- *n* is number of distinct elements that get accessed
- Average guaranteed hit- and miss-rates

# Evaluation Results

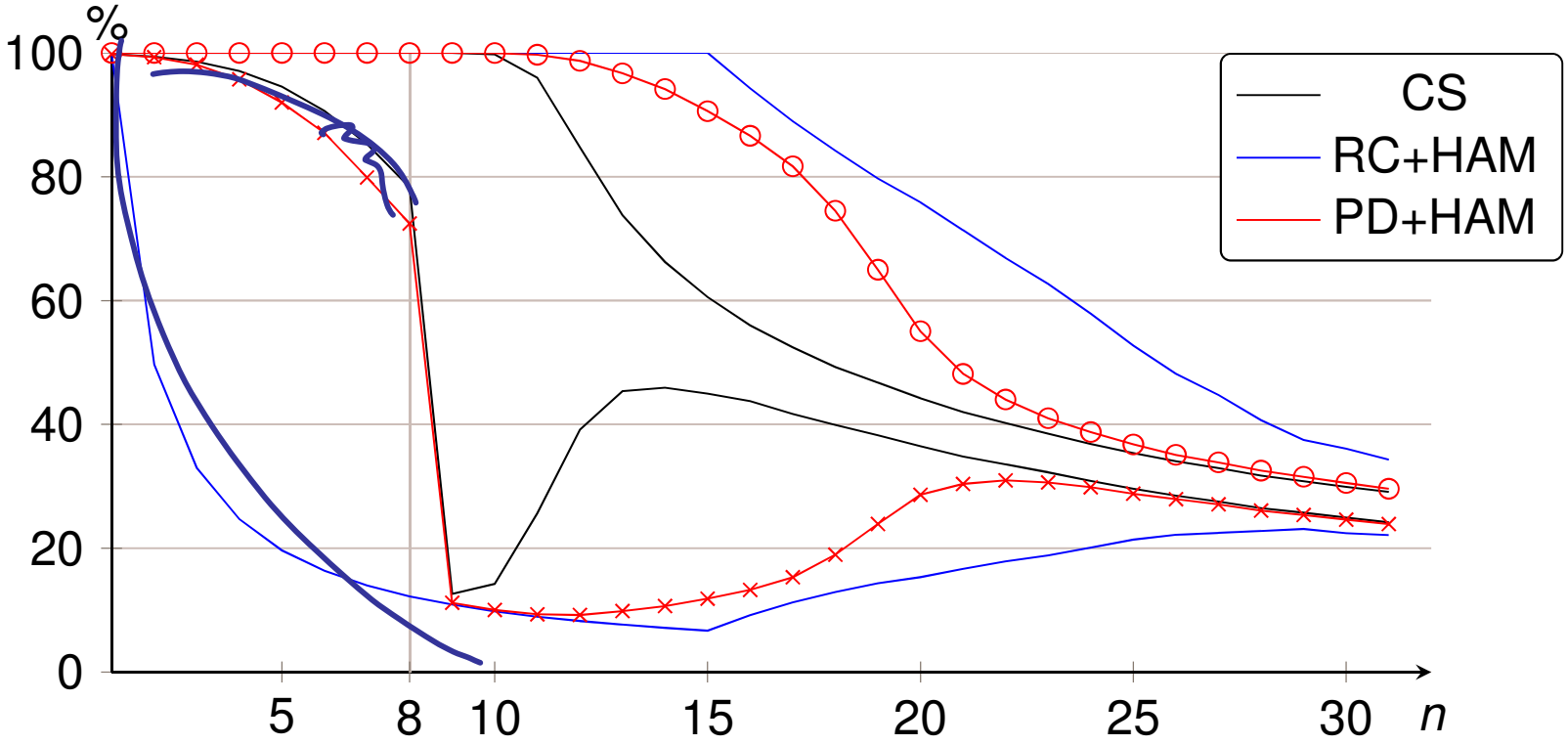
*k*=8, random sequences




- *n* is number of distinct elements that get accessed
- Average guaranteed hit- and miss-rates

# Evaluation Results

*k*=8, random sequences




- *n* is number of distinct elements that get accessed
- Average guaranteed hit- and miss-rates

# Outline

SAARLAND UNIVERSITY
COMPUTER SCIENCE

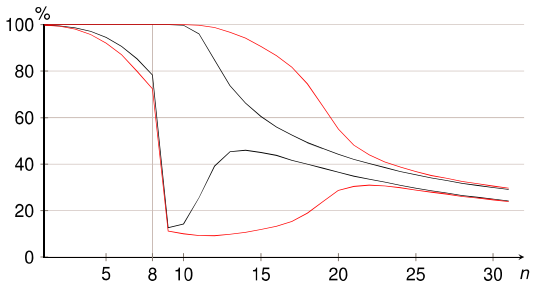
# Summary

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Precise and Efficient
FIFO-Replacement Analysis based on Static Phase Detection

# Summary

SAARLAND UNIVERSITY
COMPUTER SCIENCE




| | |
|---|---|
| $B_1$ | $\{b\}$ |
| $P_1$ | $\emptyset$ |
| $pc_1$ | 1 |
| $B_2$ | $\{b, c\}$ |
| $P_2$ | $\{b\}$ |
| $pc_2$ | 2 |
| $B_3$ | $\{a, b, c\}$ |
| $P_3$ | $\{b, c\}$ |
| $pc_3$ | 1 |

=

| | |
|---|---|
| $\{b\}$ | 1, 0 |
| $\{c\}$ | 2, 1 |
| $\{a\}$ | 1, 2 |
| $\emptyset$ | 0, 3 |

## Precise and Efficient FIFO-Replacement Analysis based on Static Phase Detection

$$\langle \overbrace{a\ ,\ \underbrace{b\ ,\ c}_{\{b,c\}\text{-phase}}}^{\{a,b,c\}\text{-phase}}\ ,\ \overbrace{\underbrace{c\ ,\ b}_{\{b,c\}\text{-phase}}\ ,\ b\ ,\ a}^{\{a,b,c\}\text{-phase}}\ ,\ \ldots\rangle$$

- Two theorems on FIFO-contents
  - bound on number of phases
  - must be cached / evicted
- Must- and may-analysis
  - static phase detection
  - multiple sub-analyses

# Further Reading

SAARLAND UNIVERSITY
COMPUTER SCIENCE


C. Ferdinand
Cache Behaviour Prediction for Real-Time Systems
*PhD Thesis, Saarland University, 1997*

J. Reineke and D. Grund
Relative competitive analysis of cache replacement policies
*LCTES 2008*

D. Grund and J. Reineke
Abstract Interpretation of FIFO Replacement
*SAS 2009*

D. Grund and J. Reineke
Toward Precise PLRU Cache Analysis
*WCET 2010*

# Related Work: LRU Analyses

SAARLAND UNIVERSITY
COMPUTER SCIENCE

- Analyses directed at worst-case execution-time analysis

  | | |
  |---|---|
  | Mueller | By “static cache simulation” |
  | Li | By integer linear programming |
  | Ferdinand | By abstract interpretation |

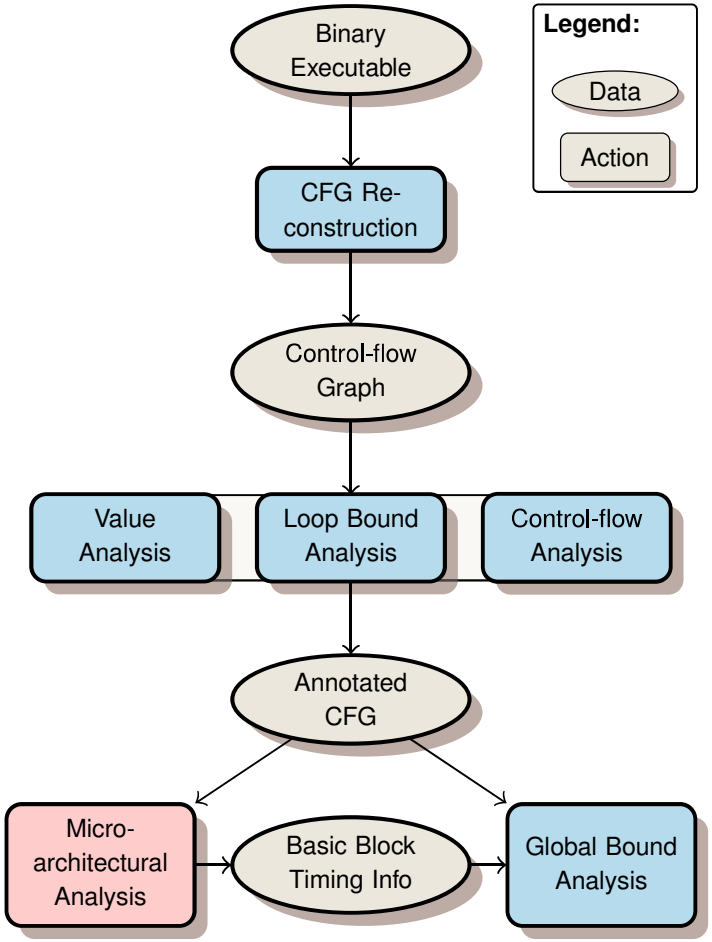
- Other than that

  | | |
  |---|---|
  | Ghosh | Cache Miss Equations, loop nests |
  | Chatterjee | Exact model of cache behavior for loop nests |

- All for LRU caches only

# Static Timing-Analysis Framework

SAARLAND UNIVERSITY
COMPUTER SCIENCE




Micro-architectural analysis

- models pipeline, caches, buses, etc.
- derives bounds on BB exec. times
- is an abstract interpretation with a huge domain
- is the computationally most expensive module

# Applicability

SAARLAND UNIVERSITY
COMPUTER SCIENCE

- Any buffer with transparent FIFO replacement:
  - Individual cache sets of instruction of data caches (I$, D$)
  - Branch target buffers (BTB, BTIC)
  - Translation lookaside buffers (TLB)
- Instances:

| | |
|---|---|
| I$ D$ | ARM 1136, 1156, 1176, 920T, 922T, 926EJ-S ($k \in \{2, 4, 64\}$) |
| I$ D$ | Marvell (Intel) XScale(s) ($k = 32$) |
| BTB | Freescale (Motorola) MPC 56x, 7450-Family ($k \in \{4, 8\}$) |
| | ... |

# Must Analysis

## Full Example for $k = 3$

SAARLAND UNIVERSITY
COMPUTER SCIENCE

- For $1 \leq n \leq k$ maintain $B_n, P_n, pc_n$

### Example

| $s$ | | $a$ | $b$ | $c$ | $c$ | $b$ | $c$ | $a$ |
|---|---|---|---|---|---|---|---|---|
| $B_1$ | $\emptyset$ | $\{a\}$ | $\{b\}$ | $\{c\}$ | $\{c\}$ | $\{b\}$ | $\{c\}$ | $\{a\}$ |
| $P_1$ | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ | $\emptyset$ |
| $pc_1$ | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| $B_2$ | $\emptyset$ | $\{a\}$ | $\{a, b\}$ | $\{b, c\}$ | $\{b, c\}$ | $\{b, c\}$ | $\{b, c\}$ | $\{a, c\}$ |
| $P_2$ | $\emptyset$ | $\{a\}$ | $\emptyset$ | $\emptyset$ | $\{c\}$ | $\emptyset$ | $\{c\}$ | $\emptyset$ |
| $pc_2$ | 0 | 0 | 1 | 1 | 1 | 2 | 2 | 1 |
| $B_3$ | $\emptyset$ | $\{a\}$ | $\{a, b\}$ | $\{a, b, c\}$ | $\{a, b, c\}$ | $\{a, b, c\}$ | $\{a, b, c\}$ | $\{a, b, c\}$ |
| $P_3$ | $\emptyset$ | $\{a\}$ | $\{a, b\}$ | $\emptyset$ | $\{c\}$ | $\{b, c\}$ | $\{b, c\}$ | $\emptyset$ |
| $pc_3$ | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 2 |
| Hit | | | | | Hit | | Hit | |

# Must Analysis

## Abstraction and Join

SAARLAND UNIVERSITY
COMPUTER SCIENCE

- Analysis domain is $Lru_k^{\leq} \times PC_k \times PP_k \subset (2^{\mathcal{B}})^k \times \mathbb{N}^k \times \mathbb{N}^k$

  $Lru_k^{\leq}$ LRU must-analysis, under-approximates accessed blocks

  $PC_k$ lower bounds on number of phases

  $PP_k$ lower bounds on phase progress

- Reuse abstract transformer and join of $Lru_k^{\leq}$
- Define appropriately for $PC_k$ and $PP_k$

# May-Analysis

SAARLAND UNIVERSITY
COMPUTER SCIENCE

- Similar to must-analysis
- Difference: Phases may be of different lengths and contents

**Theorem (Multiple Phases)**

$s = s_1 \circ \ldots \circ s_j, \forall i : |A(s_i)| = n_i \geq k$:

$$\forall q \in \mathcal{Q}_k, q \xrightarrow{s} q' : \quad C_{\sum_{i=1}^{j}(n_i-k+1)}(q') \subseteq A(s) = \bigcup_i A(s_i)$$

- More simultaneous sub-analyses
- Similar implementation employing LRU may-analysis