Abstract Interpretation of FIFO Replacement

Daniel Grund   Jan Reineke

Saarland University, Saarbrücken, Germany

Static Analysis Symposium 2009

SAARLAND UNIVERSITY

COMPUTER SCIENCE

Outline

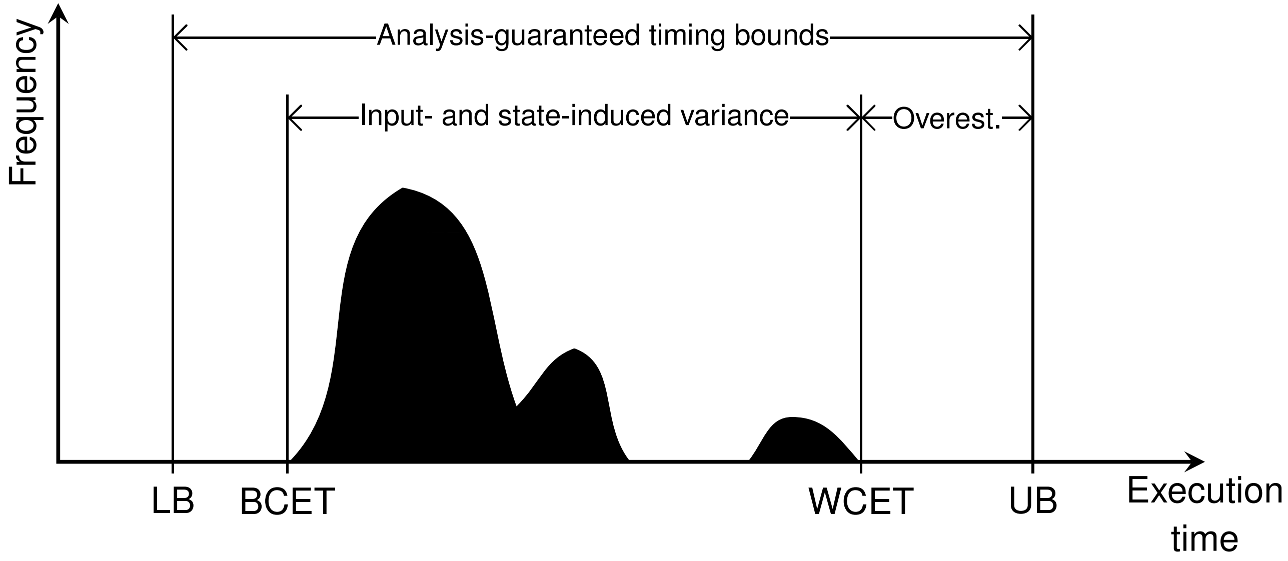
SAARLAND UNIVERSITY
COMPUTER SCIENCE

Outline

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Notions in Timing Analysis

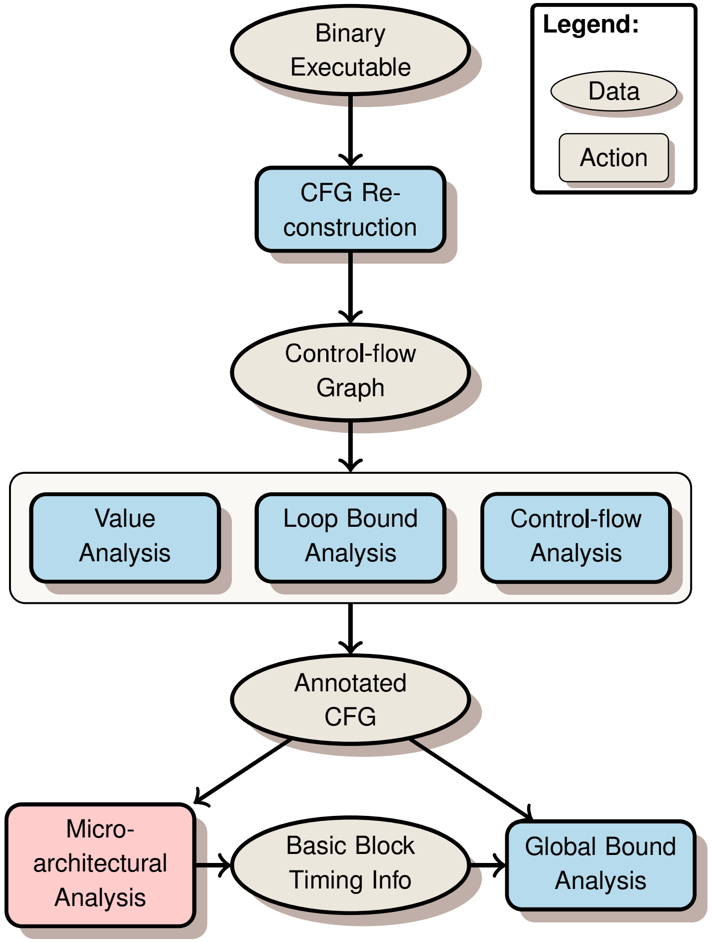
SAARLAND UNIVERSITY
COMPUTER SCIENCE



- Execution time depends on
  - program input
  - initial hardware state
- Bounds required for schedulability analysis of real-time systems

Static Timing-Analysis Framework

SAARLAND UNIVERSITY
COMPUTER SCIENCE

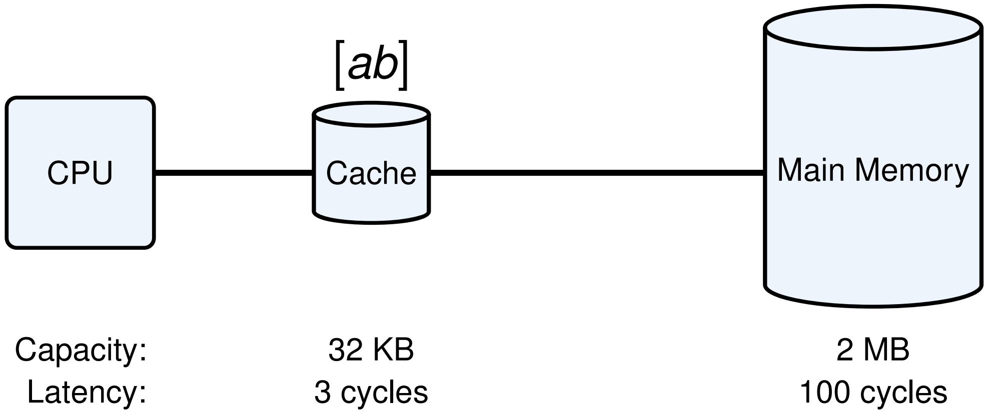


Framework implemented by *aiT* of AbsInt

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

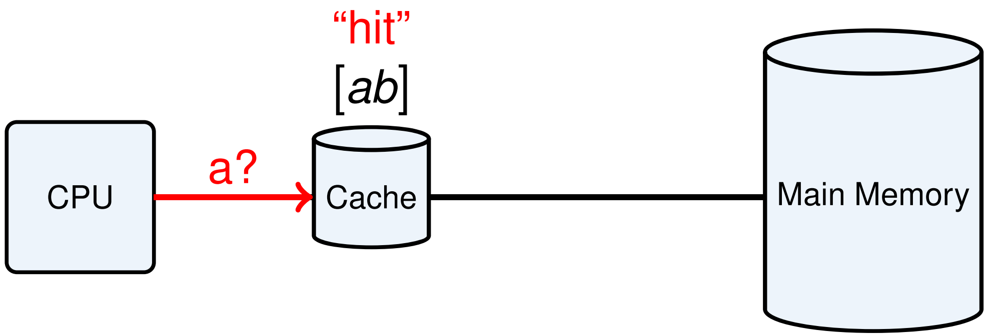
Caches and Replacement Policies



- Caches transparently buffer memory blocks

Caches and Replacement Policies

SAARLAND UNIVERSITY
COMPUTER SCIENCE

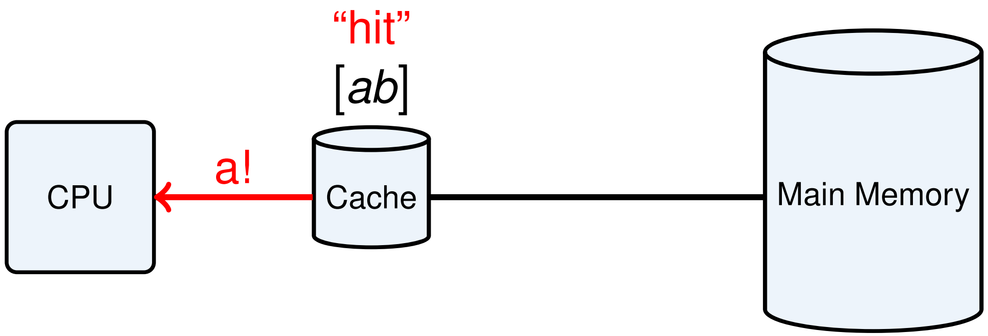


Capacity:	32 KB	2 MB
Latency:	3 cycles	100 cycles

- Caches transparently buffer memory blocks

Caches and Replacement Policies

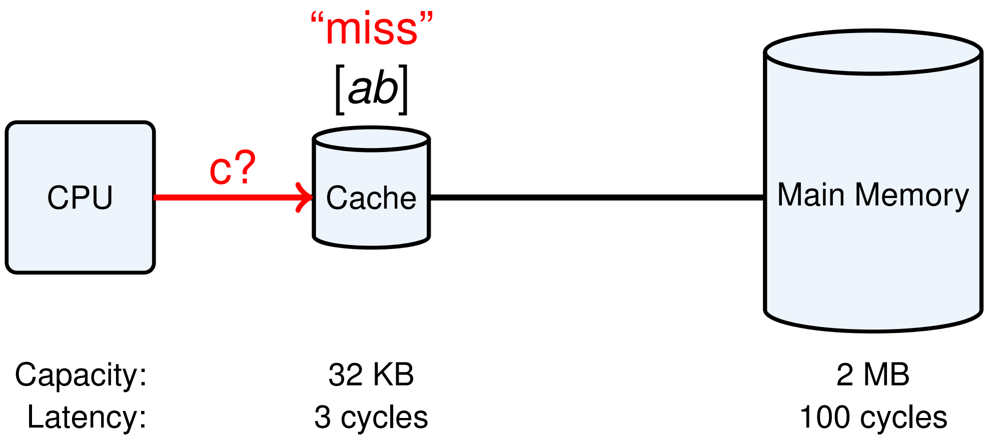
SAARLAND UNIVERSITY
COMPUTER SCIENCE



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

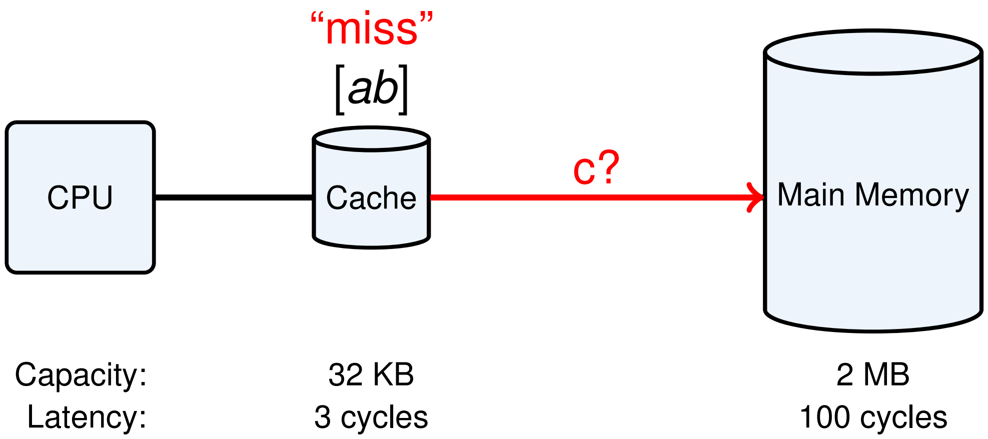
- Caches transparently buffer memory blocks

Caches and Replacement Policies



- Caches transparently buffer memory blocks

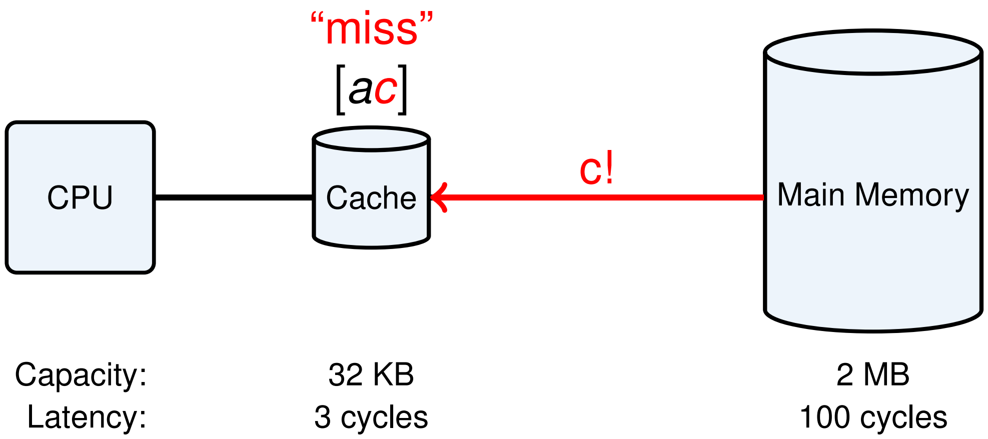
Caches and Replacement Policies



- Caches transparently buffer memory blocks

Caches and Replacement Policies

SAARLAND UNIVERSITY
COMPUTER SCIENCE

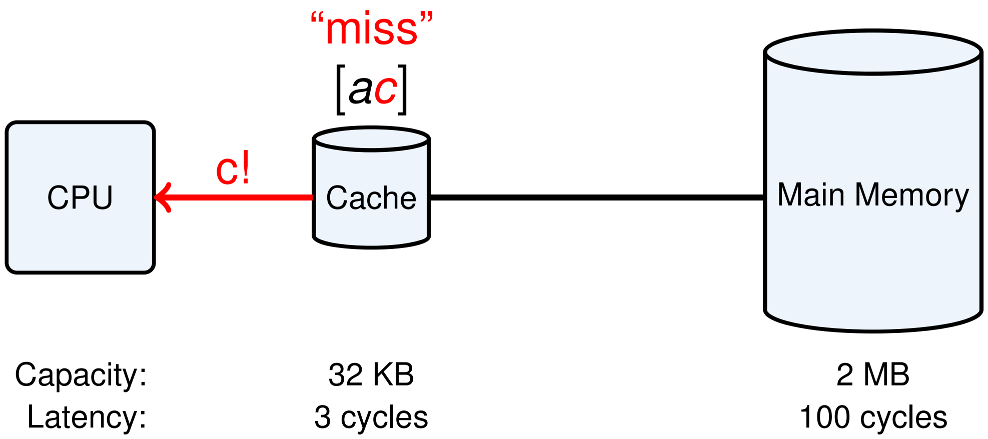


- Caches transparently buffer memory blocks
- Replacement policy *dynamically* decides which element to replace

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

- Have great influence on abstraction and (obtainable) analysis precision

Caches and Replacement Policies



- 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

- Have great influence on abstraction and (obtainable) analysis precision

Cache Analysis: Motivation & Application

- Cache performance has great influence on overall performance
- Need tight bounds on cache performance
- Otherwise derived timing bounds may be useless:
  - tasks are deemed not schedulable
  - waste of hardware resources

- Application: Buffers with transparent replacement
  - Instruction- and data-caches
  - Branch target buffers (BTB, BTIC)
  - Translation lookaside buffers (TLB)

Static Cache Analysis

- derives 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

Outline

1. Introduction & Motivation
   - Timing Analysis
   - Cache Analysis
2. Abstract Interpretation of FIFO
   - Challenge FIFO Replacement
   - Domain Cooperation
   - Must Analysis
   - May Analysis
3. Evaluation
   - Related Work
   - Analysis Precision
4. Summary

Concrete Semantics: What is FIFO?

SAARLAND UNIVERSITY
COMPUTER SCIENCE

- State of FIFO cache of size k: $s \in \mathcal{S} := \mathcal{T}^k$

$$\overset{\text{last-in}}{\downarrow}\qquad\overset{\text{first-in}}{\downarrow}$$
$$[t_0, \ldots, t_{k-1}]$$

- Examples:

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

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

- Update: $\mathcal{U}_\mathcal{S} : \mathcal{S} \times \mathcal{T} \rightarrow \mathcal{S}$

$$\mathcal{U}_\mathcal{S}([t_0, \ldots, t_{k-1}], t) := \begin{cases} [t_0, \ldots, t_{k-1}] & : \exists i : t = t_i \quad \text{“cache hit”} \\ [t, t_0, \ldots, t_{k-2}] & : \text{otherwise} \quad \text{“cache miss”} \end{cases}$$

Challenge: How to Predict Hits?

- Consider a FIFO cache with unknown contents

$$[?, ?, ?, ?] \xrightarrow[hit]{a} [?, a, ?, ?] \xrightarrow[hit]{b} [?, a, ?, b]$$

$$[?, ?, ?, ?] \xrightarrow[hit]{a} [?, ?, ?, a] \xrightarrow[miss]{b} [b, ?, ?, ?]$$

- If *a may* be a hit, then *b may* evict *a*

$\Rightarrow$ Can only predict hits for most recently accessed element

- Can one do better?

$$[?, ?, ?, ?] \xrightarrow[miss]{a} [a, ?, ?, ?] \xrightarrow[miss]{b} [b, a, ?, ?]$$

- If *a is* a miss, then *b cannot* evict *a*

$\Rightarrow$ Can predict hits for *a* until *k* further misses might have happened

Challenge: How to Predict Hits?

- Consider a FIFO cache with unknown contents

$$[?, ?, ?, ?] \xrightarrow[hit]{a} [?, a, ?, ?] \xrightarrow[hit]{b} [?, a, ?, b]$$

$$[?, ?, ?, ?] \xrightarrow[hit]{a} [?, ?, ?, a] \xrightarrow[miss]{b} [b, ?, ?, ?]$$

- If *a may* be a hit, then *b may* evict *a*

⇒ Can only predict hits for most recently accessed element

- Can one do better?

$$[?, ?, ?, ?] \xrightarrow[miss]{a} [a, ?, ?, ?] \xrightarrow[miss]{b} [b, a, ?, ?]$$

- If *a is* a miss, then *b cannot* evict *a*

⇒ Can predict hits for *a* until *k* further misses might have happened

⇒ Need may-information to obtain precise must-information

A Solution. Well, our Contributions

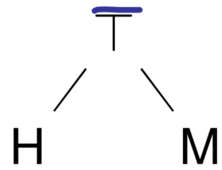
- Framework for static cache analysis
  - policy independent
  - couples must- and may-analyses
  - analyses cooperate via “update reduction”
- FIFO must-analysis
  - can profit from may-information
  - hence, also better must-information
- FIFO may-analysis
  - utilizes order of hits and misses
  - more precise than prior analyses

Framework and Classification

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Domain: $\mathit{Fifo} := \mathit{Must} \times \mathit{May}$

Classification: $\mathit{Class} := \{\mathrm{H}, \mathrm{M}\}^{\top}$

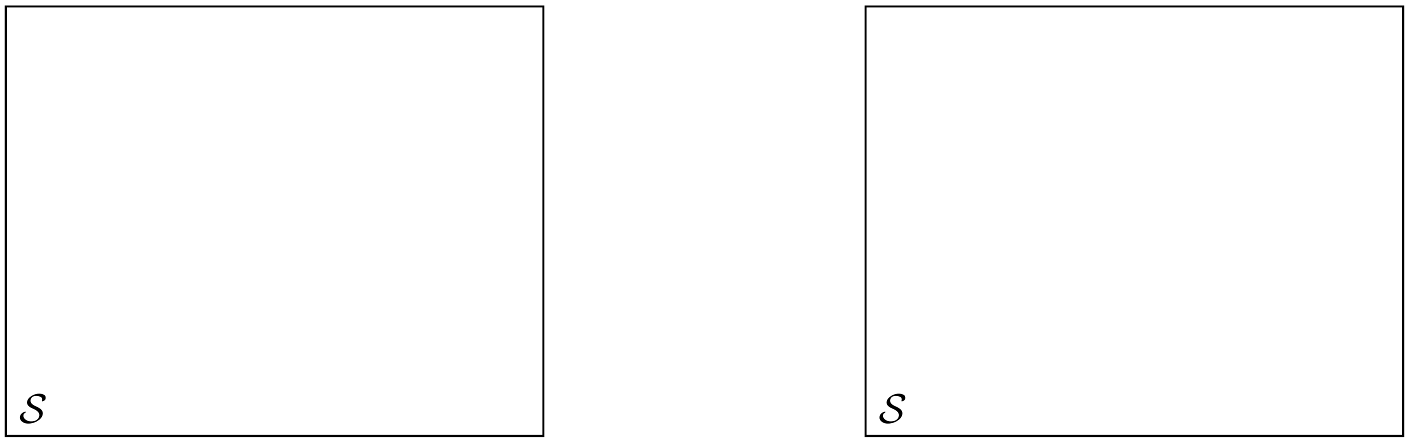


H : cache hit
M : cache miss
$\top$: unclassified

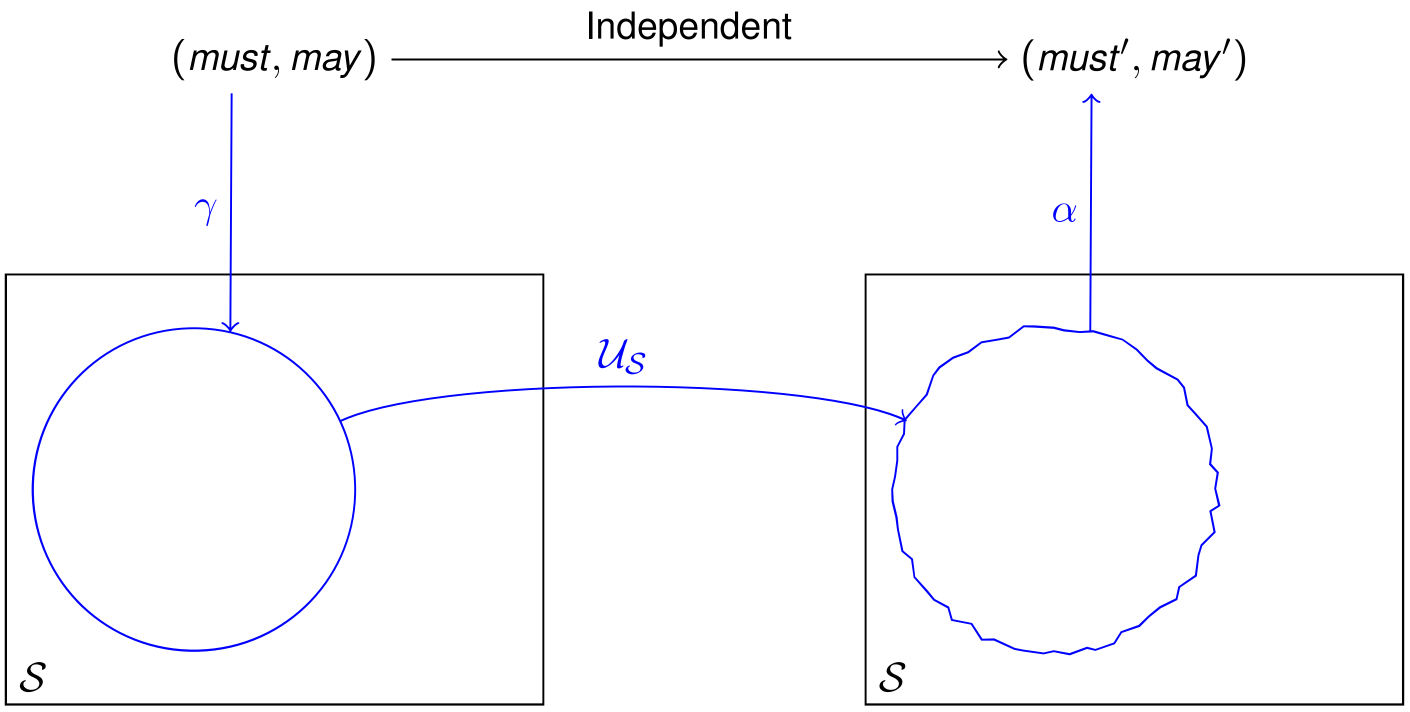
$$\mathcal{C}_{\mathit{Fifo}} : \mathit{Fifo} \times \mathcal{T} \longrightarrow \mathit{Class}$$
$$\mathcal{C}_{\mathit{Fifo}}((\mathit{must}, \mathit{may}), t) := \mathcal{C}_{\mathit{Must}}(\mathit{must}, t) \sqcap \mathcal{C}_{\mathit{May}}(\mathit{may}, t)$$

Domain Cooperation via Update Reduction

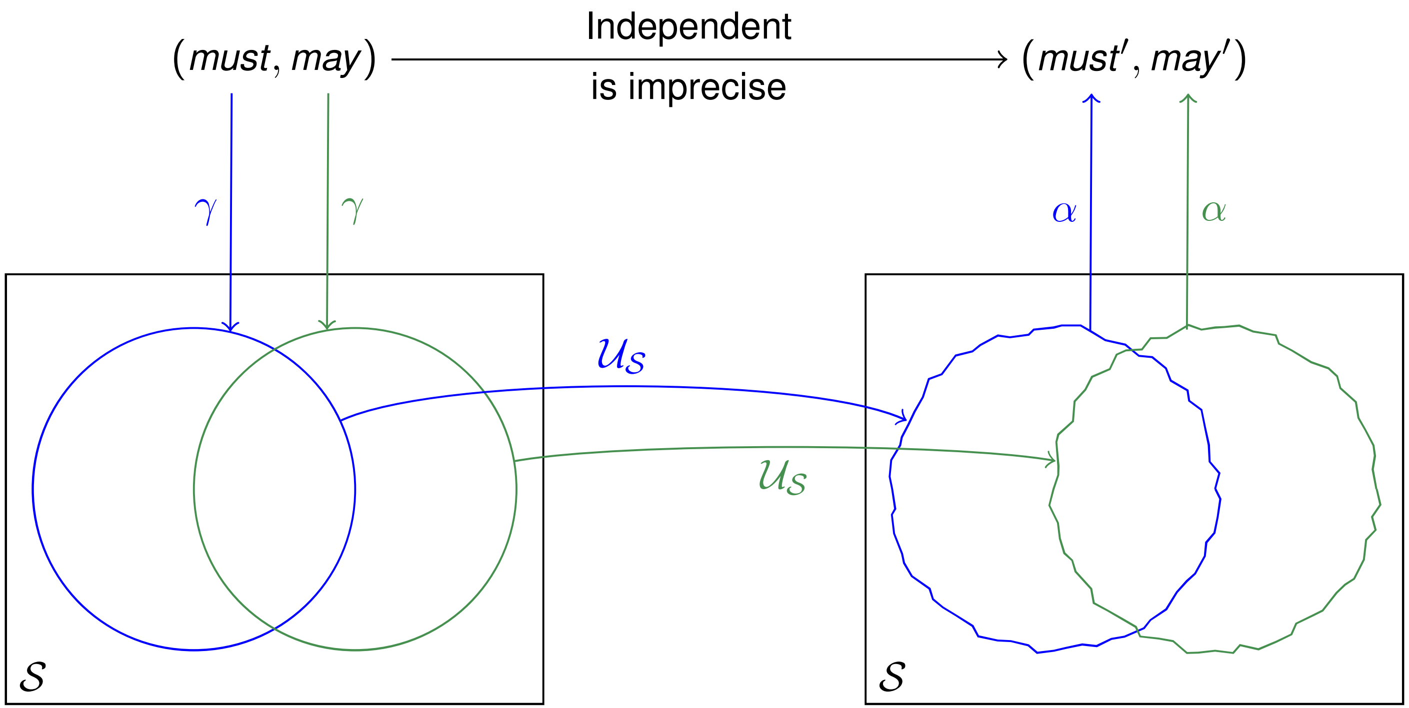
$$(\mathit{must}, \mathit{may}) \xrightarrow{\text{Independent}} (\mathit{must}', \mathit{may}')$$



Domain Cooperation via Update Reduction

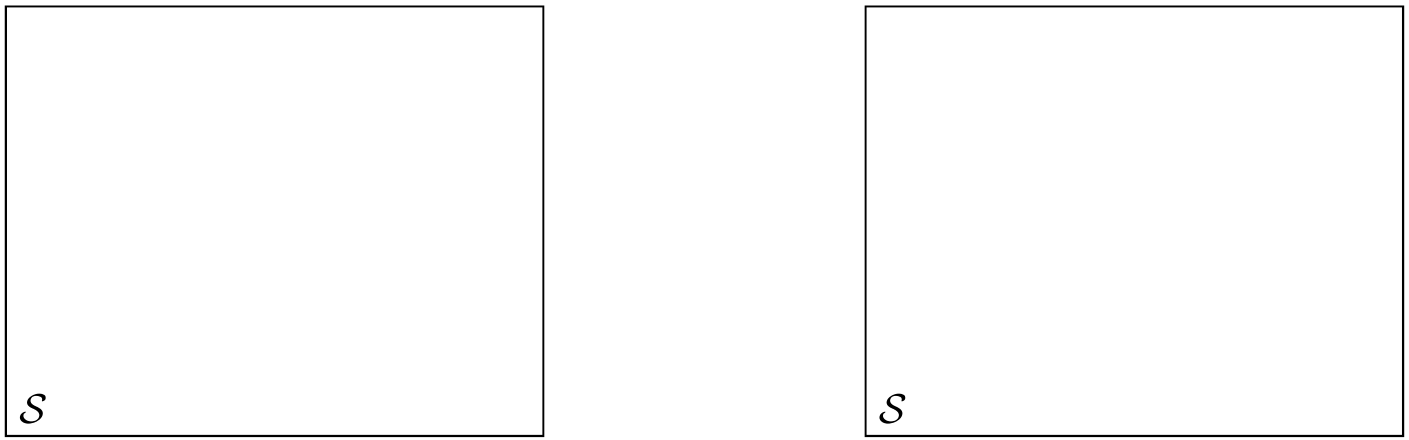


Domain Cooperation via Update Reduction

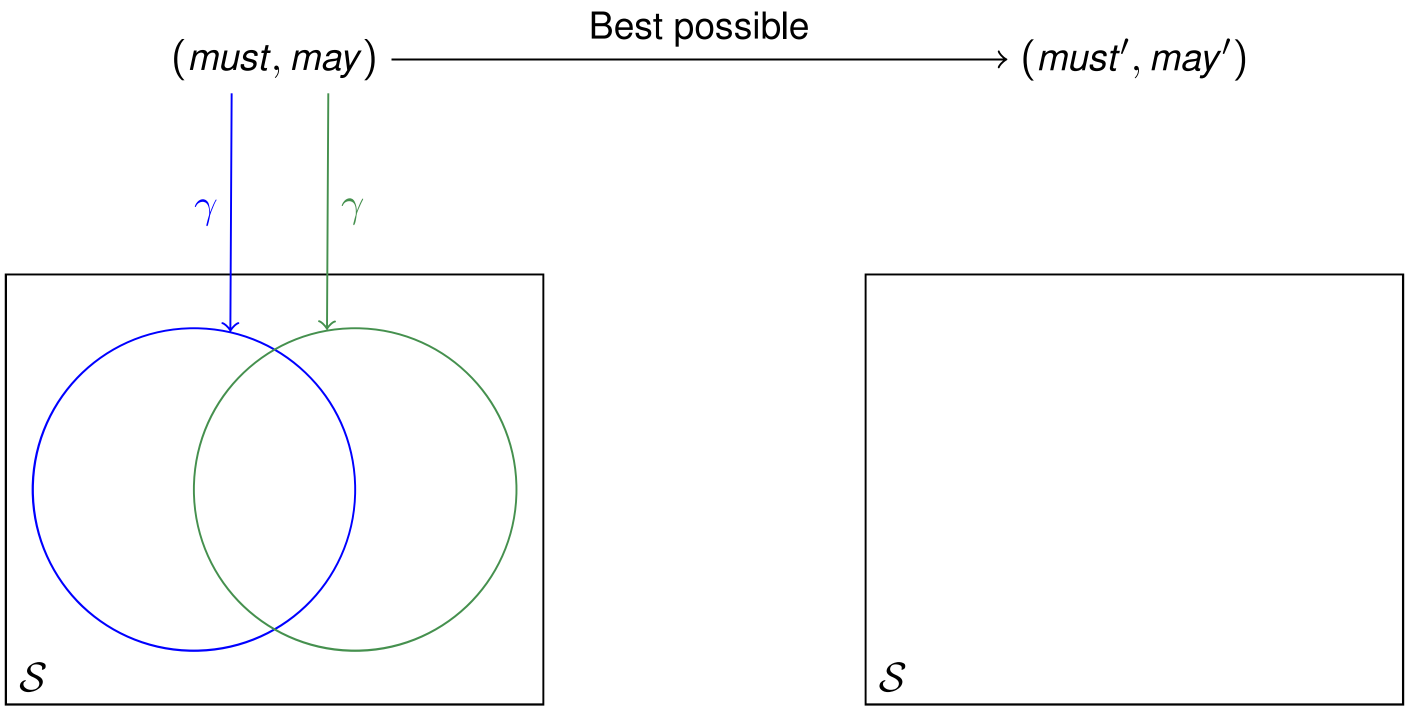


Domain Cooperation via Update Reduction

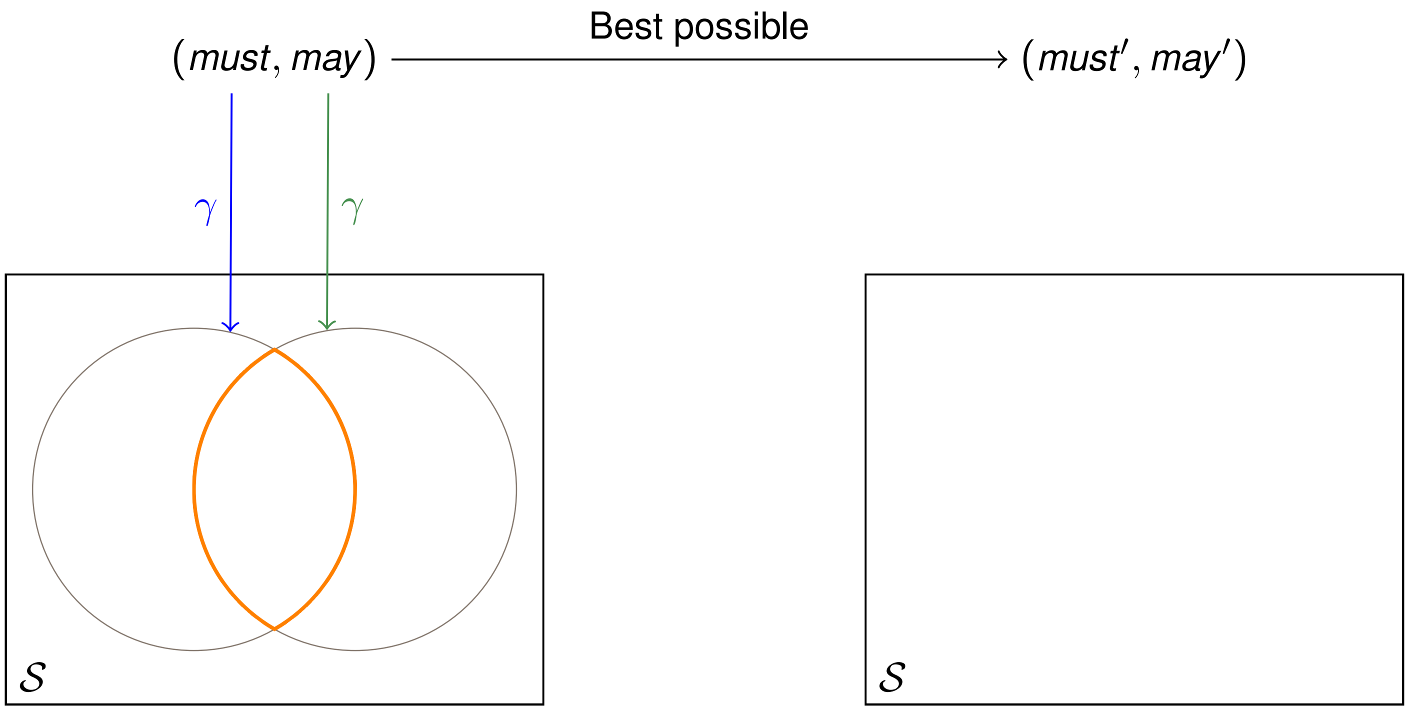
$$(must, may) \xrightarrow{\text{Best possible}} (must', may')$$



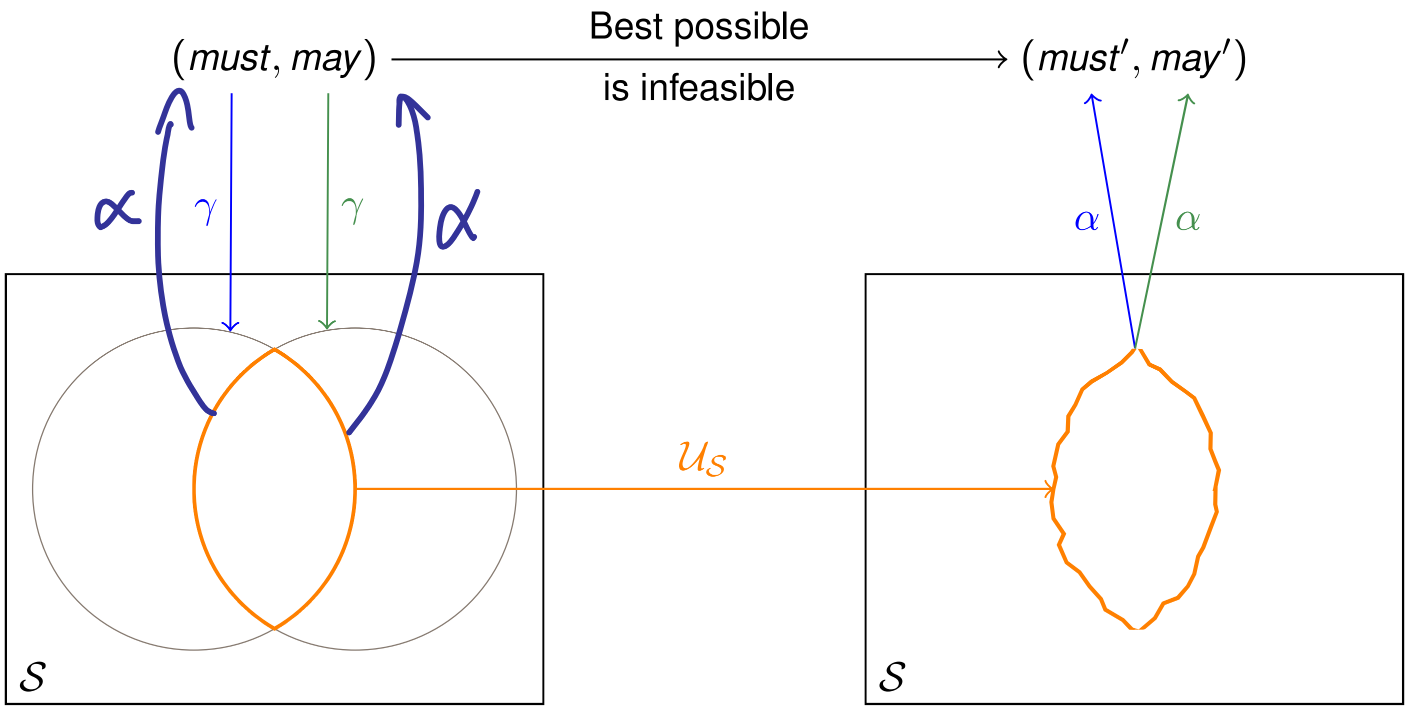
Domain Cooperation via Update Reduction



Domain Cooperation via Update Reduction



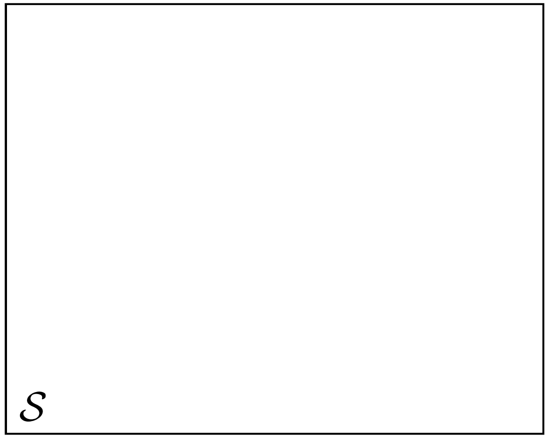
Domain Cooperation via Update Reduction

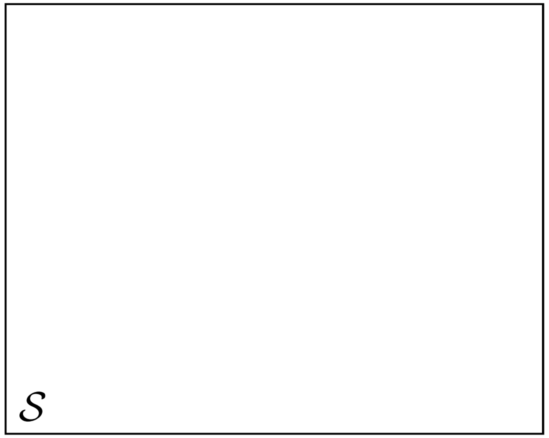


Domain Cooperation via Update Reduction

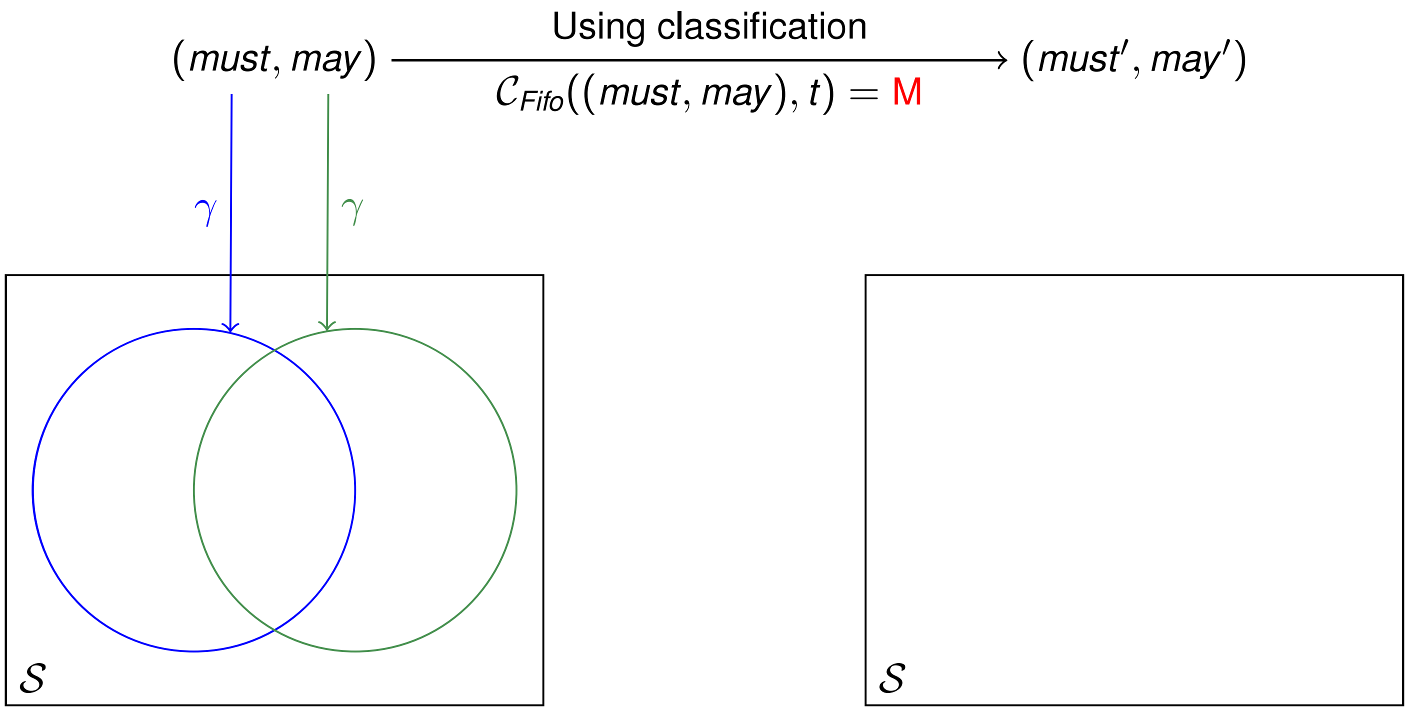
SAARLAND UNIVERSITY
COMPUTER SCIENCE

$$(\mathit{must}, \mathit{may}) \xrightarrow[\mathcal{C}_{\mathit{Fifo}}((\mathit{must}, \mathit{may}), t) = \textcolor{red}{\mathrm{M}}]{\text{Using classification}} (\mathit{must}', \mathit{may}')$$

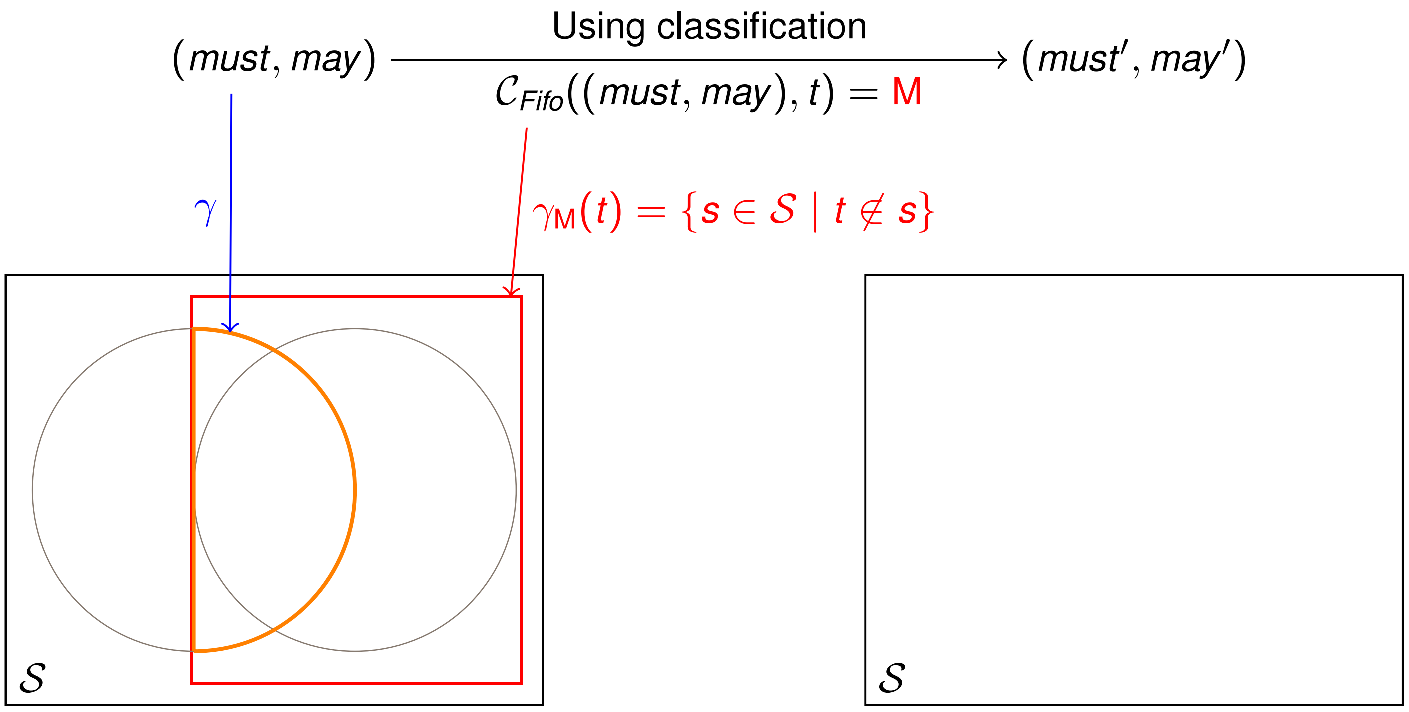




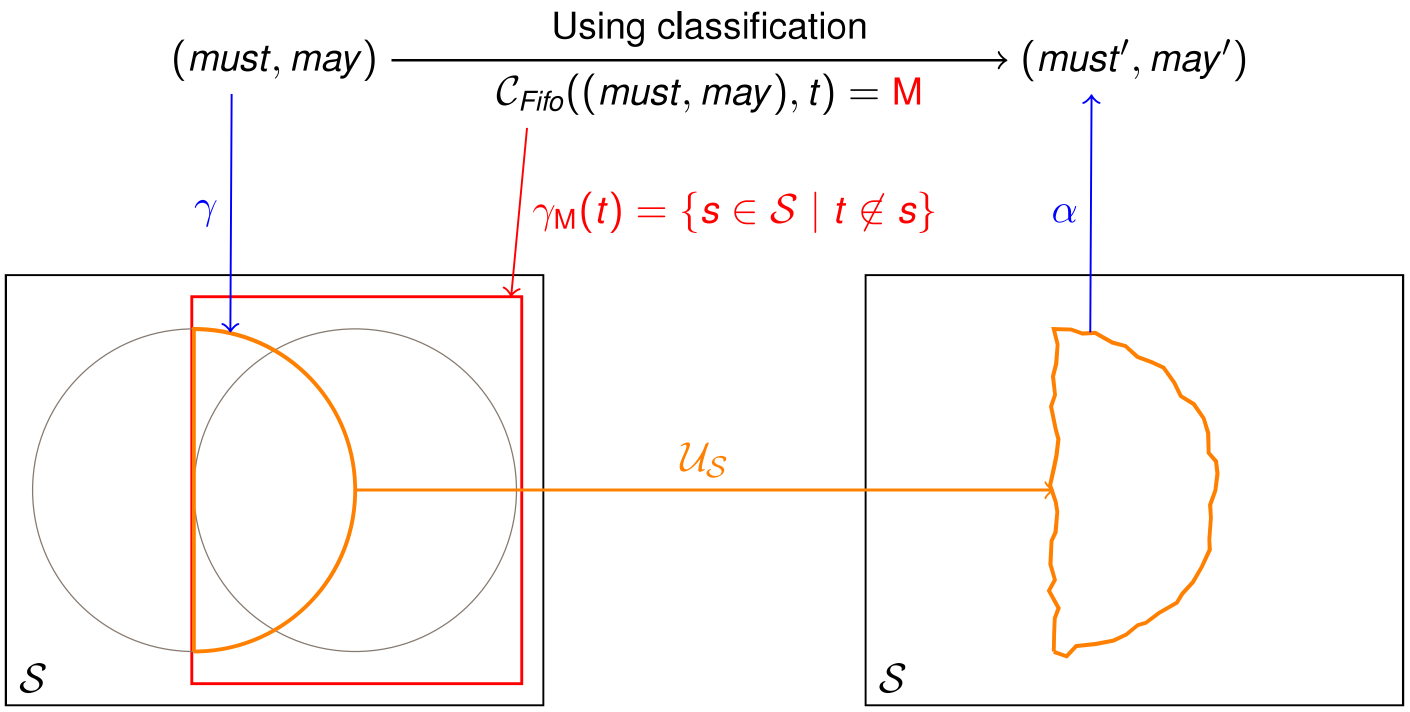
Domain Cooperation via Update Reduction



Domain Cooperation via Update Reduction



Domain Cooperation via Update Reduction



Outline

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Must-Analysis: Potential Misses

SAARLAND UNIVERSITY
COMPUTER SCIENCE

- For FIFO, a newly inserted element is evicted after k misses

$\Rightarrow$ Maintain upper bound on number of misses: Potential misses

- Abstract must-domain closely resembles the concrete domain

$$Must_{Fifo_k} := [T_0, \ldots, T_{k-1}],$$

where $T_i \in \mathcal{P}(\mathcal{T})$, $T_i \cap T_j = \emptyset$, and $\sum |T_i| \leq k$.

- $t \in T_i \Rightarrow$ at most i misses since insertion of t
- Concretization example

$$\gamma([\{f\}, \emptyset, \{a, c\}, \{b\}]) = \{[f, c, a, b], [f, a, c, b]\}$$

Must-Analysis: Update

SAARLAND UNIVERSITY
COMPUTER SCIENCE

$$\mathcal{U}_{Must} : Must \times \mathcal{T} \times Class \rightarrow Must$$

$$\mathcal{U}_{Must}([T_0, \ldots, T_{k-1}], t, cl) := \begin{cases} [\emptyset, T_0, \ldots, T_{k-2} \cup \{t\}] & : cl = \top \\ [T_0, \ldots, T_{k-1}] & : cl = \mathrm{H} \\ [\{t\}, T_0, \ldots, T_{k-2}] & : cl = \mathrm{M} \end{cases}$$

- Misses classified by may-analysis
- Last case only possible due to domain cooperation

Outline

SAARLAND UNIVERSITY
COMPUTER SCIENCE

May-Analysis: Definite Misses

SAARLAND UNIVERSITY
COMPUTER SCIENCE

- How to predict misses?

$\Rightarrow$ Maintain lower bound on number of misses: Definite misses

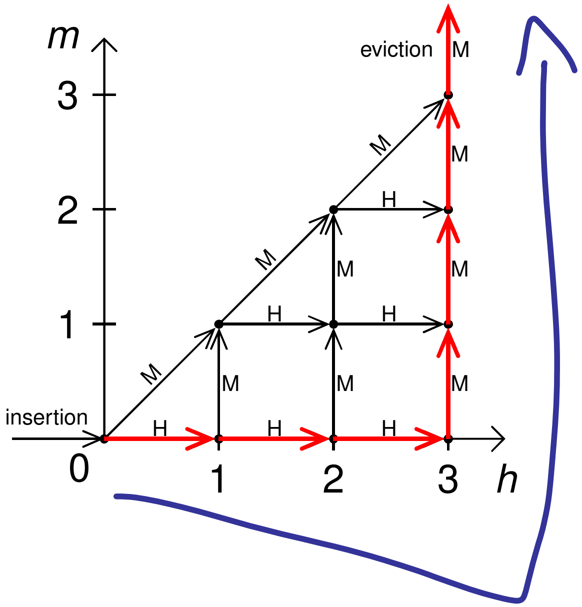
- Initially, anything might be cached
- To classify a miss for an *individual* access, one needs to predict k other misses first

Lemma

A newly inserted element is evicted after accesses to at most $2k - 1$ pairwise different elements.

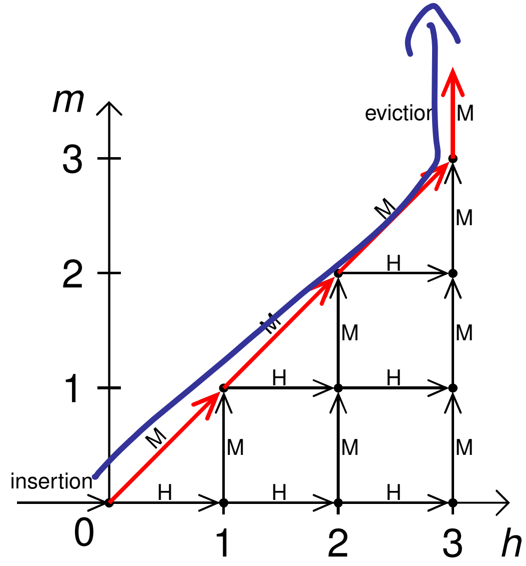
May-Analysis: Idea “Early Misses”

- Initial state: $[x, c, b, a]$
- Sequences of different length:
  - $\langle a, b, c, e, f, g, h \rangle$
- Common final state: $[h, g, f, e]$



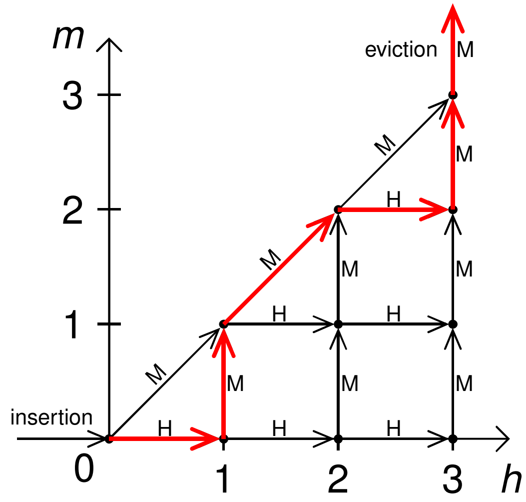
May-Analysis: Idea "Early Misses"

- Initial state: $[x, c, b, a]$
- Sequences of different length:
  - $\langle a, b, c, e, f, g, h \rangle$
  - $\langle e, f, g, h \rangle$
- Common final state: $[h, g, f, e]$



May-Analysis: Idea “Early Misses”

- Initial state: $[x, c, b, a]$
- Sequences of different length:
  - $\langle a, b, c, e, f, g, h \rangle$
  - $\langle e, f, g, h \rangle$
  - $\langle a, e, f, c, g, h \rangle$
- Common final state: $[h, g, f, e]$

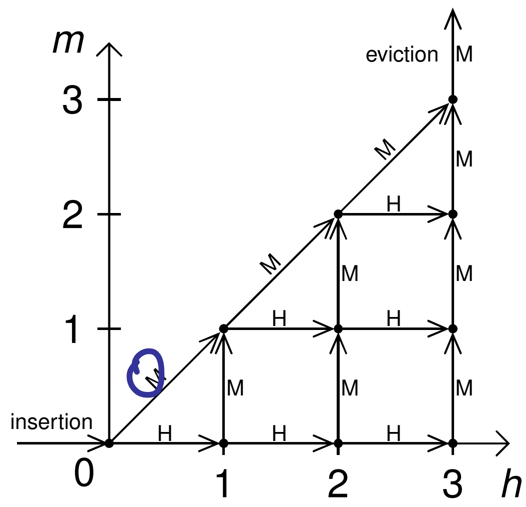


May-Analysis: Idea “Early Misses”

SAARLAND UNIVERSITY
COMPUTER SCIENCE

- Initial state: $[x, c, b, a]$
- Sequences of different length:
  - $\langle a, b, c, e, f, g, h \rangle$
  - $\langle e, f, g, h \rangle$
  - $\langle a, e, f, c, g, h \rangle$
- Common final state: $[h, g, f, e]$

$\Rightarrow$ Sequences differ in number of hits

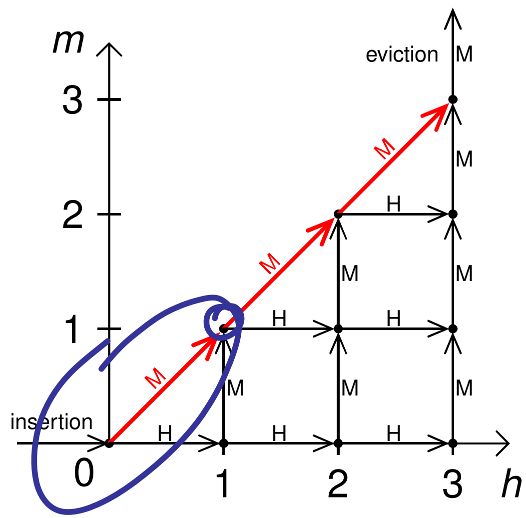


May-Analysis: Idea "Early Misses"

SAARLAND UNIVERSITY
COMPUTER SCIENCE

- Initial state: $[x, c, b, a]$
- Sequences of different length:
  - $\langle a, b, c, e, f, g, h \rangle$
  - $\langle e, f, g, h \rangle$
  - $\langle a, e, f, c, g, h \rangle$
- Common final state: $[h, g, f, e]$

$\Rightarrow$ Sequences differ in number of hits



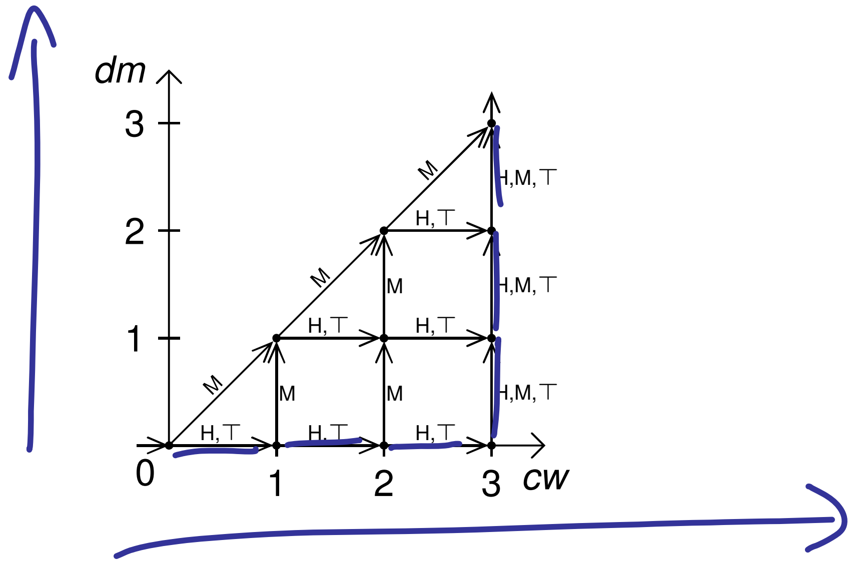
- "Early misses"
  - preclude hits to thereby evicted elements
  - reduce number of possible accesses between insertion and eviction

$\Rightarrow$ Order of hits and misses is important

May-Analysis: Domain

SAARLAND UNIVERSITY
COMPUTER SCIENCE

- May analysis approximates position in triangle
- Unclassified access → "take the longer way"



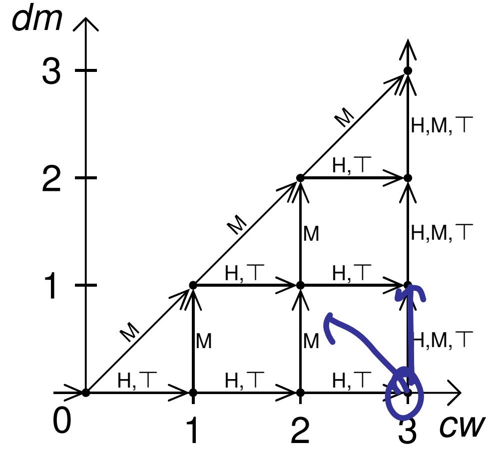
- For each element t, the analysis maintains:

A Set of *Potentially Accessed Elements*

dm Number of *Definite Misses*

cw Number of *Covered Ways* (Covered Cache Positions)

May-Analysis: Example



- Assume sequence $\langle x, a, b, c \rangle$ and all accesses are unclassified
- Then for x one has:

$A = \{a, b, c\}$	a, b and c might have been accessed since the last insertion of x
$dm = 0$	0 misses have definitely happened since the last insertion of x
$cw = 3$	Assuming that all unclassified accesses were hits, then 3 elements of A must be cached

- Consider next access to d

Outline

Relative Competitiveness

- How many misses would FIFO have if LRU has m_{LRU} misses?

Definition: Relative Competitiveness

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

$$m_P(p, s) \leq f \cdot m_Q(q, s) + c$$

for all access sequences s and compatible cache states p, q.

- E.g. LRU$(2k - 1)$ is $(1, 0)$ miss-competitive vs. FIFO(k)

$\Rightarrow$ LRU$(2k - 1)$ may-analysis can be used for FIFO(k) may analysis

Evaluation Setup

SAARLAND UNIVERSITY
COMPUTER SCIENCE

- Must-analysis:

  CM Canonical must-analysis (this paper)

- May-analyses:

  No None

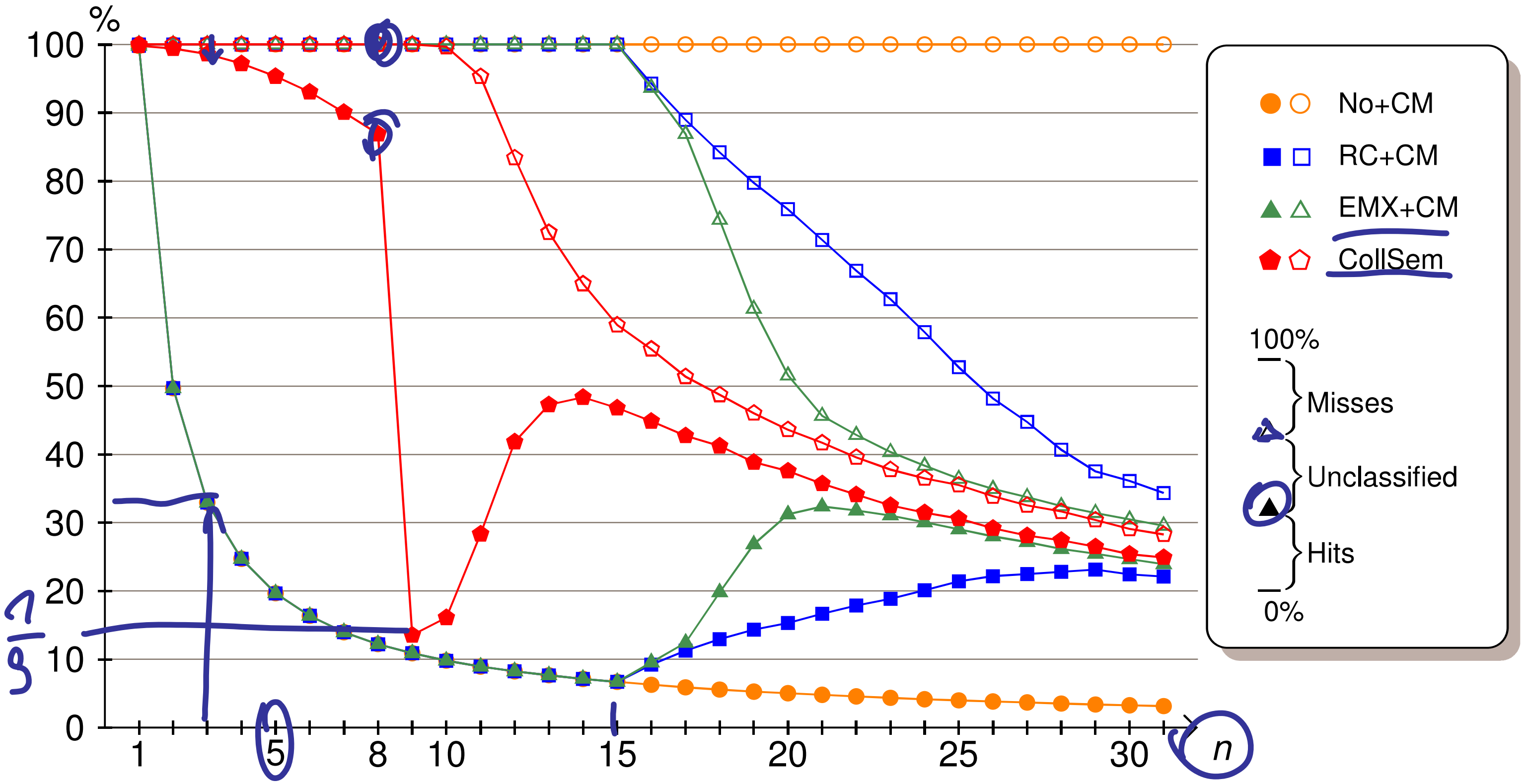
  RC Based on relative competitiveness

  EMX Early Miss eXploitation (this paper)

- Instantiations of cache analysis framework:
  - No+CM
  - RC+CM
  - EMX+CM
- Synthetic benchmarks:
  - Random access sequences and program fragments

Evaluation Results

SAARLAND UNIVERSITY
COMPUTER SCIENCE

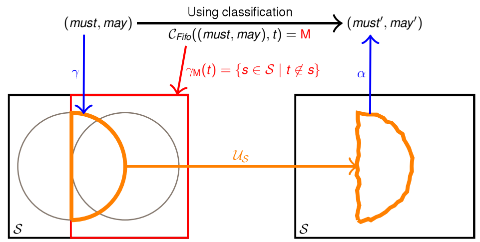


- Average guaranteed hit- and miss-rates for a cache of size 8
- *n* is number of pairwise different elements that are accessed

Outline

Summary

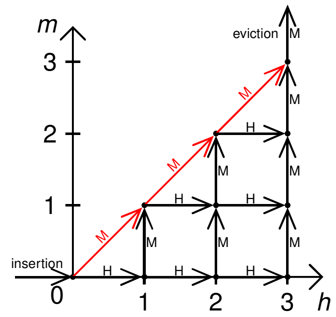
- Cache analysis framework
  - Couple several analyses
  - Cooperation via classifications



- Canonical FIFO must-analysis
  - Potential misses

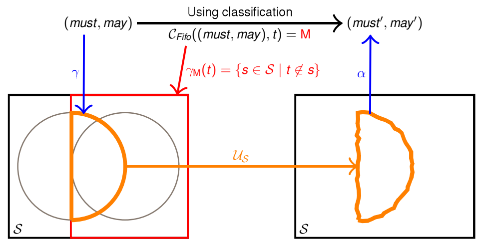
$$\mathcal{U}_{Must}([T_0, \ldots, T_{k-1}], t, cl) := \begin{cases} [\emptyset, T_0, \ldots, T_{k-2} \cup \{t\}] & : cl = \top \\ [T_0, \ldots, T_{k-1}] & : cl = \mathrm{H} \\ [\{t\}, T_0, \ldots, T_{k-2}] & : cl = \mathrm{M} \end{cases}$$

- EMX FIFO may-analysis
  - Definite misses
  - Early Miss eXploitation



Summary

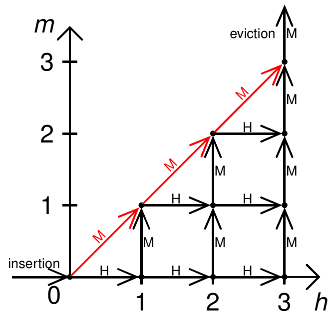
- Cache analysis framework
  - Couple several analyses
  - Cooperation via classifications



- Canonical FIFO must-analysis
  - Potential misses

$$\mathcal{U}_{Must}([T_0, \ldots, T_{k-1}], t, cl) := \begin{cases} [\emptyset, T_0, \ldots, T_{k-2} \cup \{t\}] & : cl = \top \\ [T_0, \ldots, T_{k-1}] & : cl = \mathrm{H} \\ [\{t\}, T_0, \ldots, T_{k-2}] & : cl = \mathrm{M} \end{cases}$$

- EMX FIFO may-analysis
  - Definite misses
  - Early Miss eXploitation



Thank you for listening. Questions?

Further Reading

- R. Wilhelm et al.
  The worst-case execution time problem—
  overview of methods and survey of tools
  Transactions on Embedded Computing Systems, 7(3), 2008
- J. Reineke and D. Grund
  Relative competitive analysis of cache replacement policies
  LCTES 2008

Related Work: LRU Analyses

- Bounds on number of cache misses:

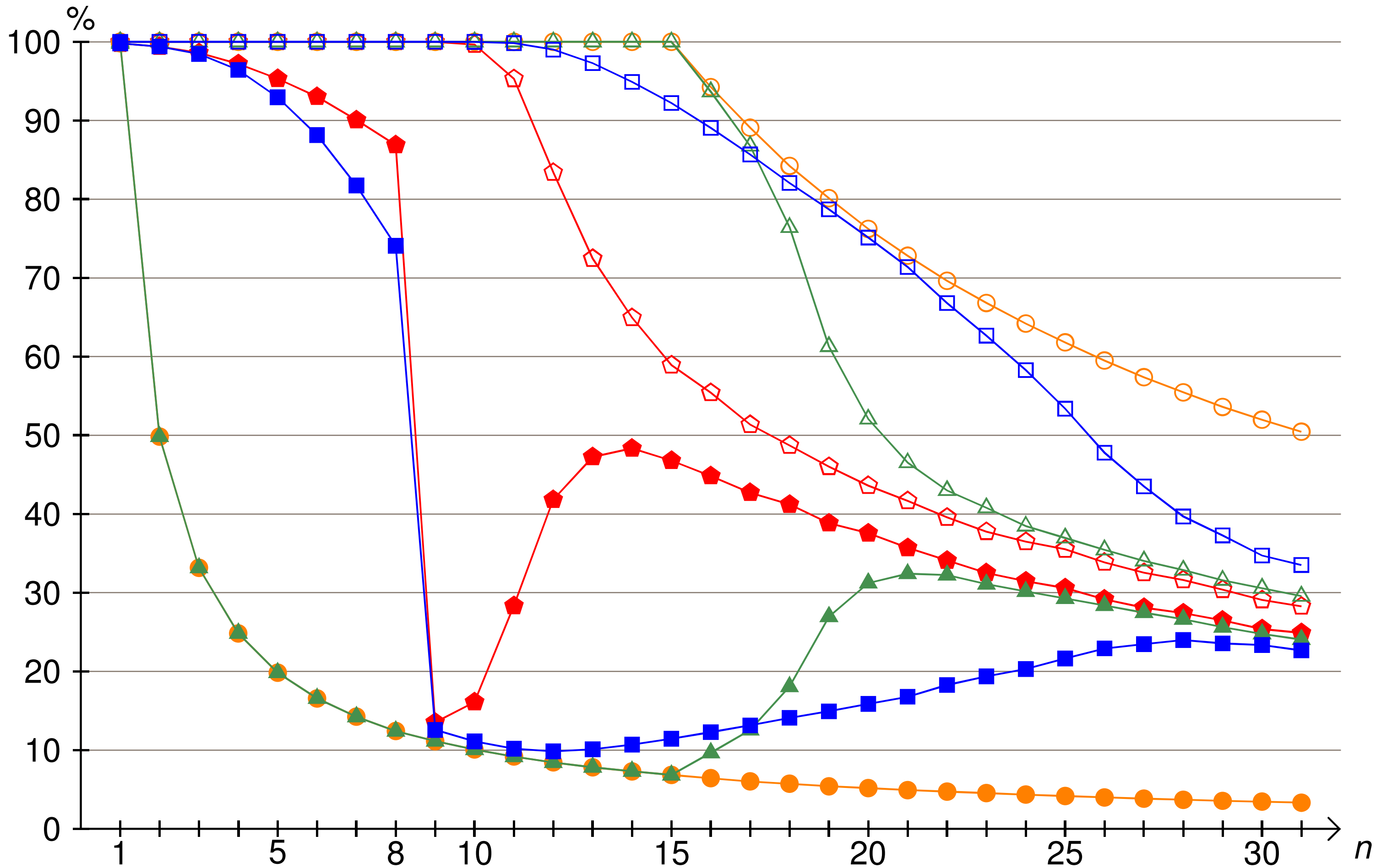
Ghosh	Cache Miss Equations, loop nests
Chatterjee	Exact model of cache behavior for loop nests

- Classification of individual accesses:

Mueller	By "static cache simulation"
Li	By integer linear programming
Ferdinand	By abstract interpretation

- Only for LRU caches

What About The Gap for $n \leq k$?



Does the Initial Cache State Make a Difference?

SAARLAND UNIVERSITY
COMPUTER SCIENCE

- Yes, a FIFO of size k, is (k,k) sensitive

Definition: Sensitivity

Policy P is (f, c) miss-sensitive if

$$m_P(p, s) \leq f \cdot m_P(p', s) + c$$

for all access sequences s and all cache states p, p'.

$\Rightarrow$ For FIFO of size 4, execution time may differ by a factor of 3

R. Wilhelm et al.
Memory Hierarchies, Pipelines, and Buses for
Future Architectures in Time-critical Embedded Systems
IEEE Transactions on CAD of Integrated Circuits and Systems 2009