"Ascertaining Uncertainty for Efficient Exact Cache Analysis" [1] [TMMR17]

Valentin Touzeau [1] Claire Maïza [1] David Monniaux [1] Jan Reineke [2]

[1]Univ. Grenoble Alpes, VERIMAG, F-38000 Grenoble, France
CNRS, VERIMAG, F-38000 Grenoble, France

[2]Saarland University, Saarland Informatics Campus
Saarbrücken, Germany

July 4, 2017

[1]This work was partially supported by the European Research Council under the European Union's Seventh Framework Programme (FP/2007-2013) / ERC Grant Agreement nr. 306595 "STATOR".

Motivations

Real-time systems

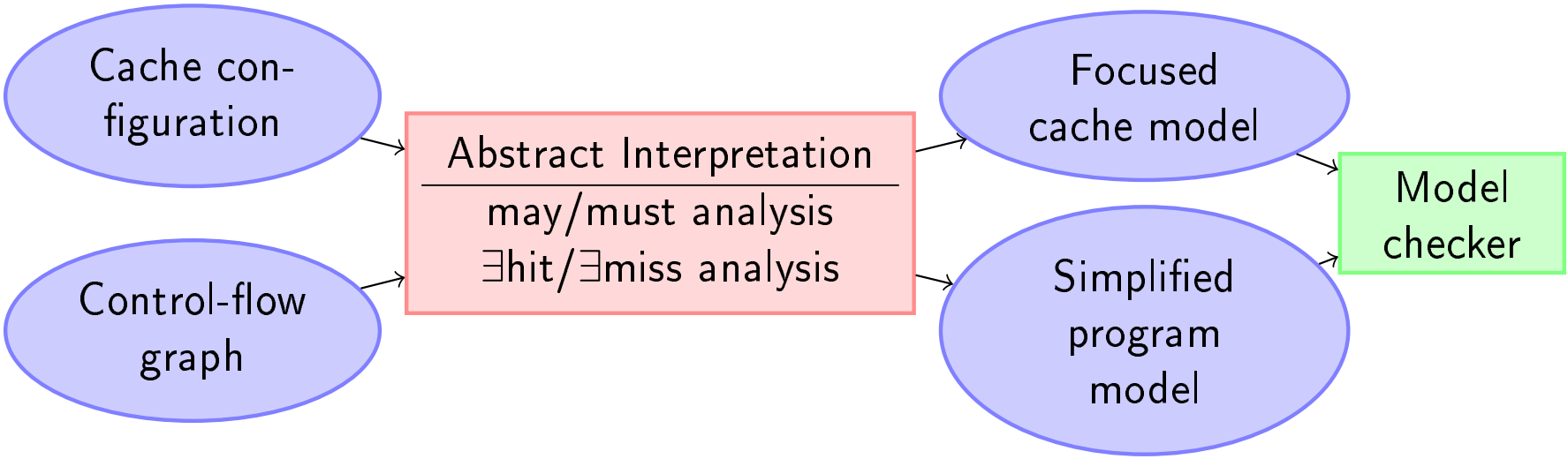
Caches have the biggest influence on Worst Case Execution Time [HP11]

- Cache hit : ~ 2 cycles,
- Cache miss : ~ 100 cycles

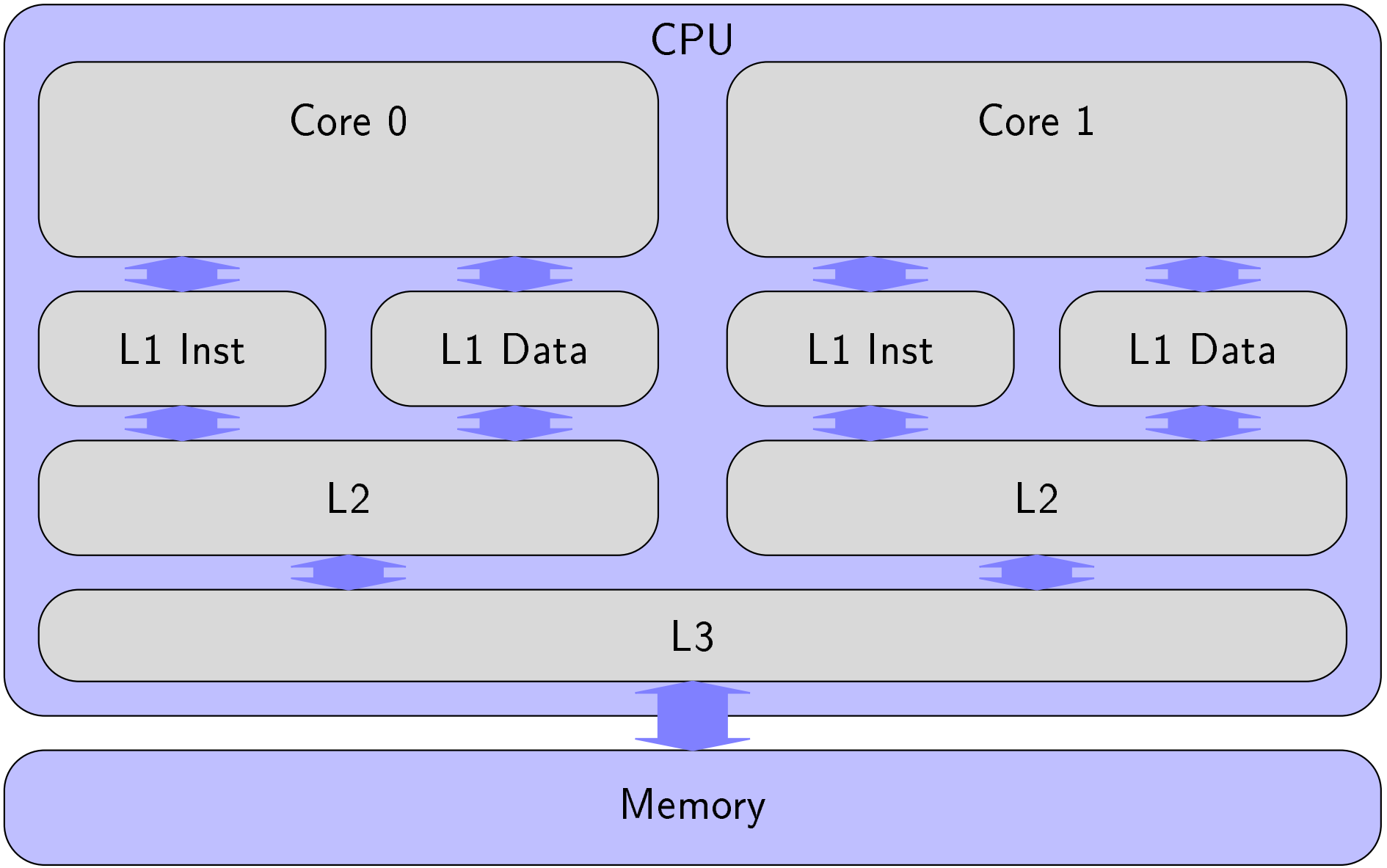
Security

Cache may leak secret information to other process running on the same machine [Ber05] [CLS06]

Framework


Cache configuration
Control-flow graph
Abstract Interpretation
may/must analysis
∃hit/∃miss analysis
Focused cache model
Simplified program model
Model checker

Caches



Caches

Blocks

Insts/Data are transferred by blocks of fixed size

Cache sets

Caches are split into independent sets. A block is assigned to a unique set.

Ways

Each set can contain k blocks (k is called associativity of the cache).

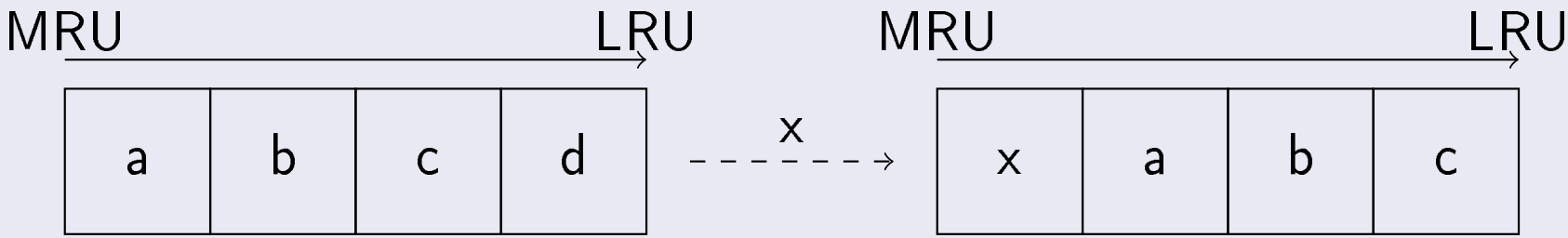
Replacement policy

Selection of the block to evict on a miss when the associated set is full.

Least Recently Used Policy

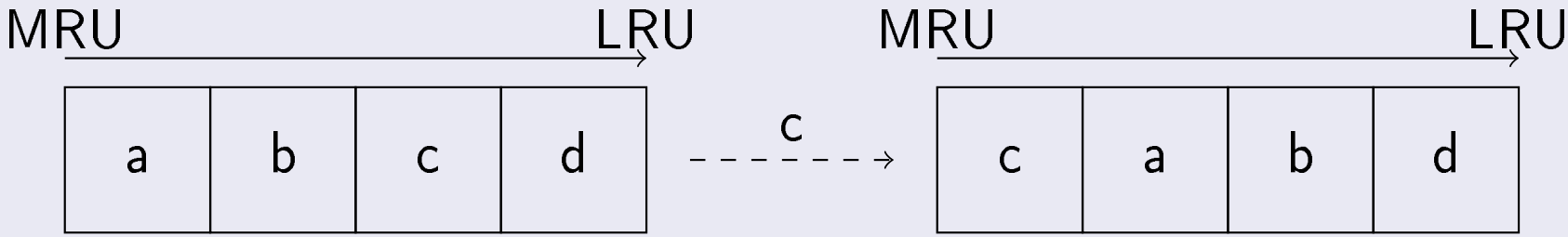
Miss

All blocks are shifted, and LRU is evicted.



Hit

Only younger blocks are shifted.



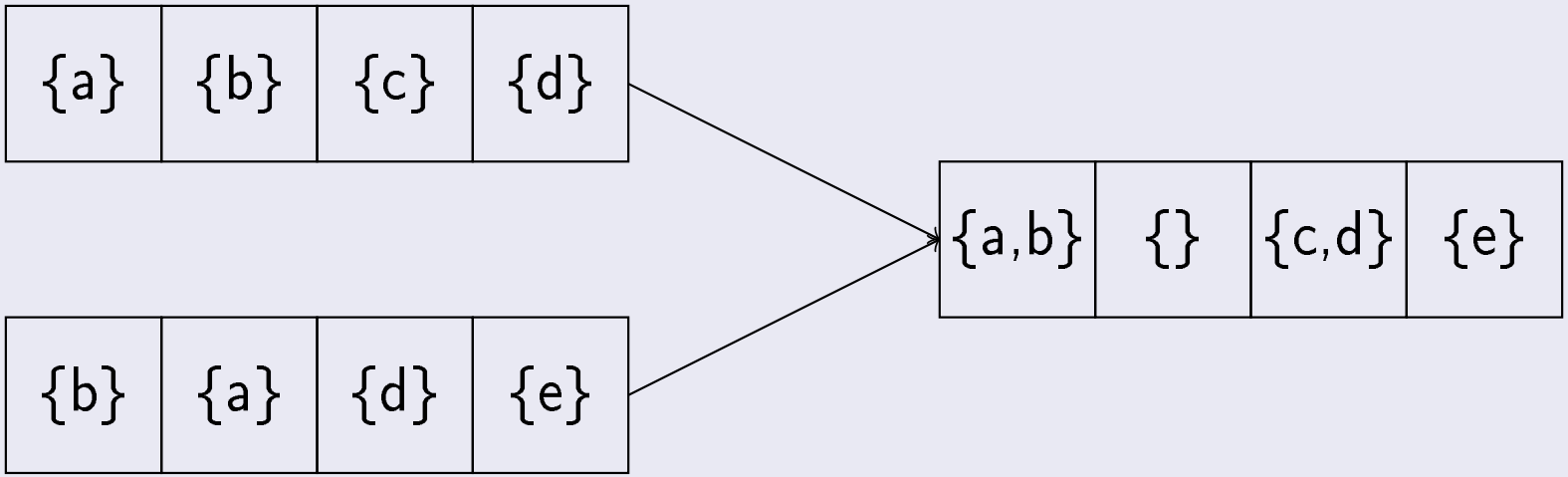
May/Must Analysis [Fer97]

Abstraction of cache states

May/Must overapproximates block ages :

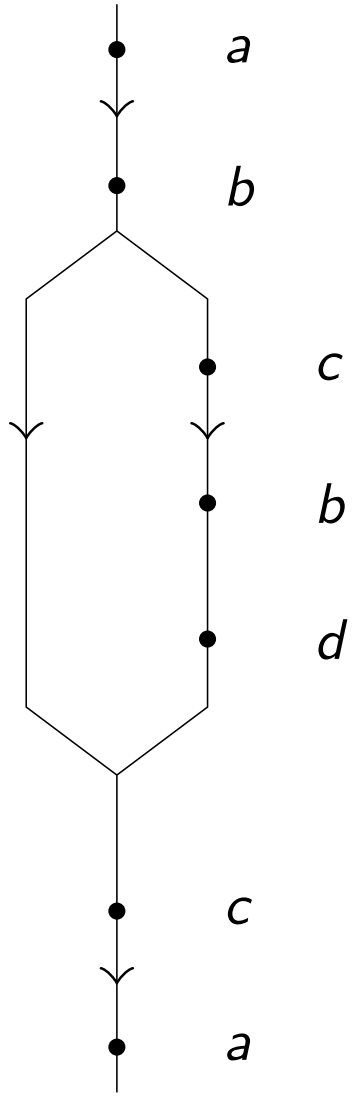
- May : safe lower bound
- Must : safe upper bound

Example : May join



May/Must Analysis

Control Flow Graph



May/Must Analysis

May Analysis

$[a, \bot, \bot, \bot]$

$L = [b, a, \bot, \bot]$

$[c, b, a, \bot]$

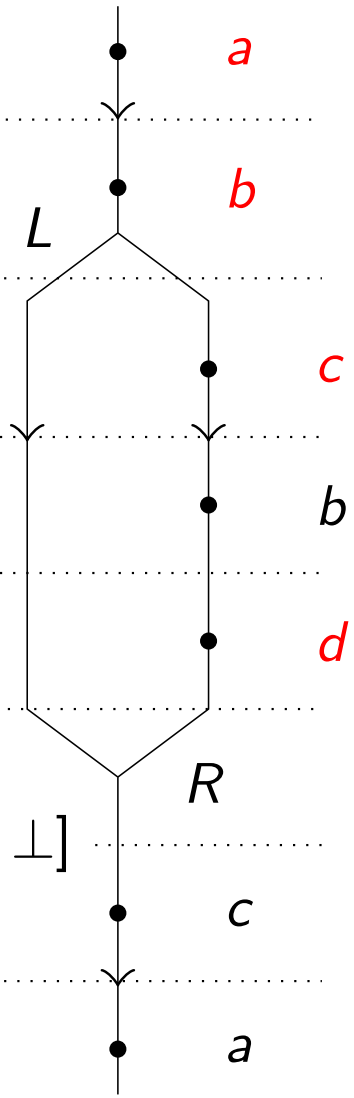
$[b, c, a, \bot]$

$R = [d, b, c, a]$

$L \sqcup_{May} R = [\{b, d\}, a, c, \bot]$

$[c, \{b, d\}, a, \bot]$

Control Flow Graph



May/Must Analysis

May Analysis	Control Flow Graph	Must Analysis
$[a, \bot, \bot, \bot]$	a	$[a, \bot, \bot, \bot]$
$L = [b, a, \bot, \bot]$	b (L)	$L = [b, a, \bot, \bot]$
$[c, b, a, \bot]$	c	$[c, b, a, \bot]$
$[b, c, a, \bot]$	b	$[b, c, a, \bot]$
$R = [d, b, c, a]$	d	$R = [d, b, c, a]$
$L \sqcup_{May} R = [\{b, d\}, a, c, \bot]$	R	$L \sqcup_{Must} R = [\bot, b, \bot, a]$
$[c, \{b, d\}, a, \bot]$	c	$[c, \bot, b, \bot]$
	a	

Definitely Unknown

Main idea

Ensure the existence of one path leading to hit/miss.
Maintain a safe upper/lower bound valid on one path (at least)

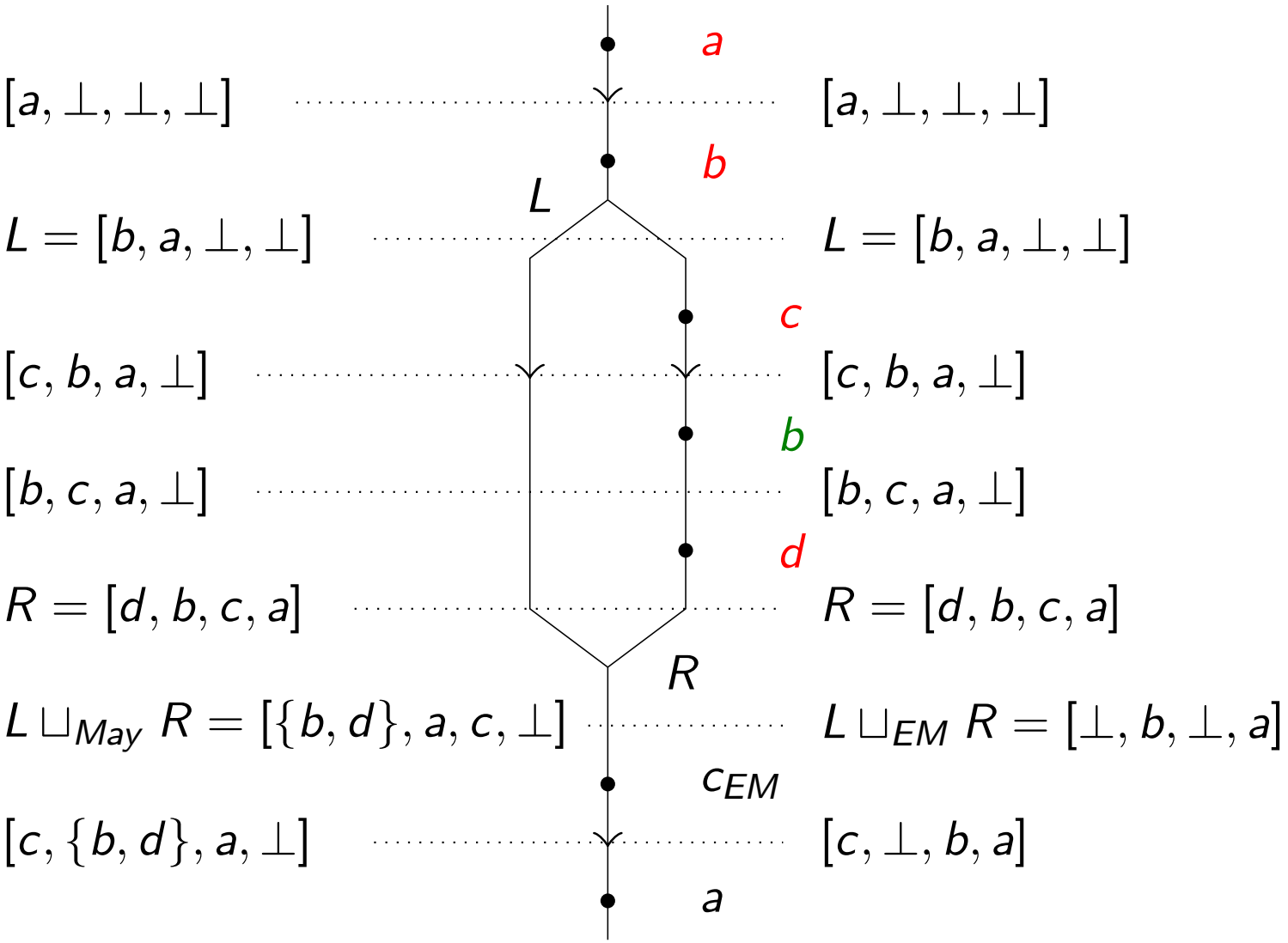
Exist-Miss

Compute a lower bound on block's age.

- Join: Keep the maximum (which is a better lower bound)
- Update: Incrementing this bound is safe if the accessed block is older

Rely on May Analysis

May Analysis | Control Flow Graph | Exist-Miss Analysis

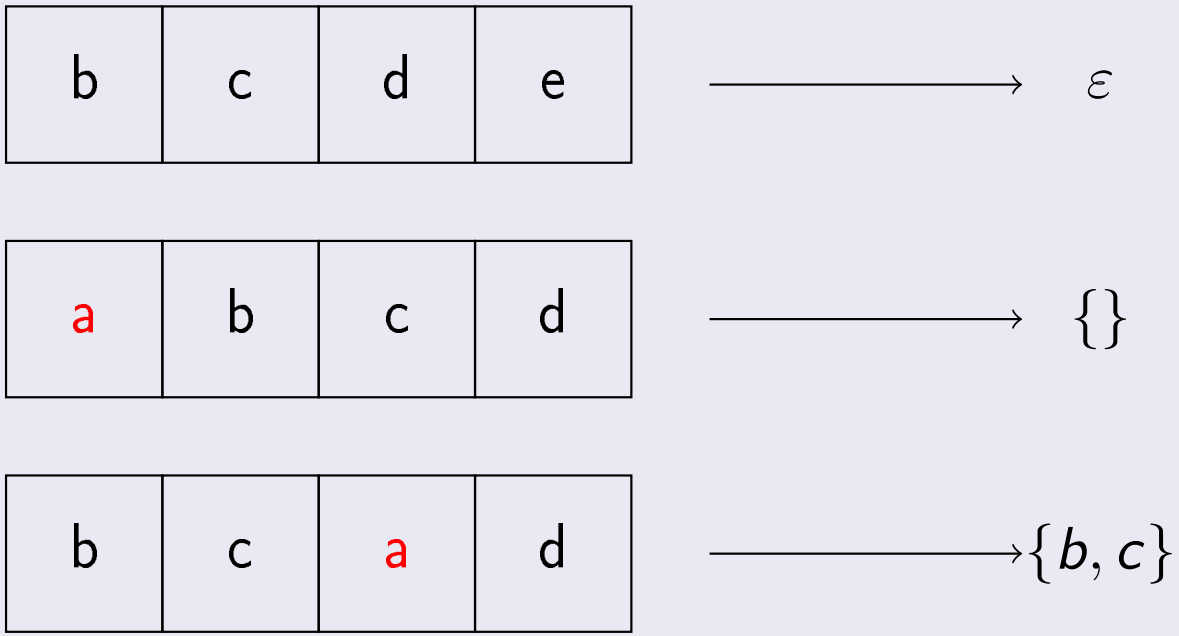


Focus cache model

Main Idea

Given a block a, a's age is incremented on access to b iff b is older than a.
Abstract cache state relatively to a requires tracking younger blocks only.

Abstraction



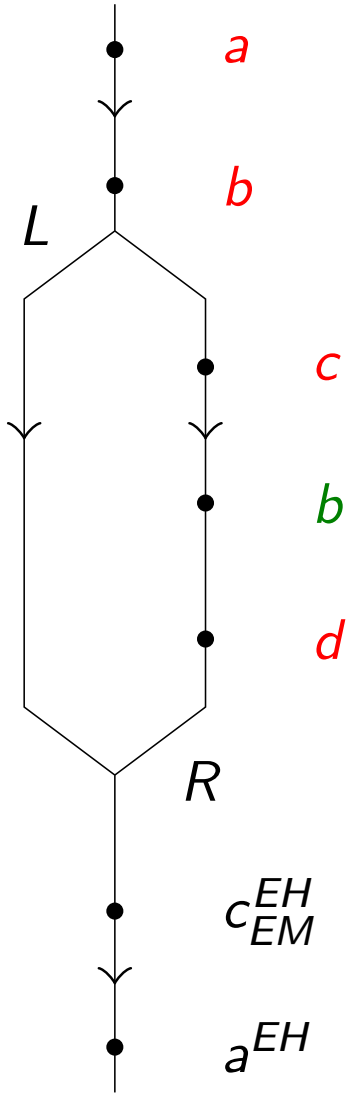
Focus cache model

Update

- Do not track any block until a is accessed:
  $\varepsilon \overset{b}{\mapsto} \varepsilon$
- Start tracking blocks when a is accessed:
  $\varepsilon \overset{a}{\mapsto} \{\}$
- Track younger blocks until a is evicted:
  $$S \overset{b}{\mapsto} \begin{cases} S \cup \{b\} & \text{if } \mid S \cup \{b\} \mid \leq k-1 \\ \varepsilon & \text{otherwise} \end{cases}$$
- Reset the younger blocks set when a is accessed:
  $S \overset{a}{\mapsto} \{\}$

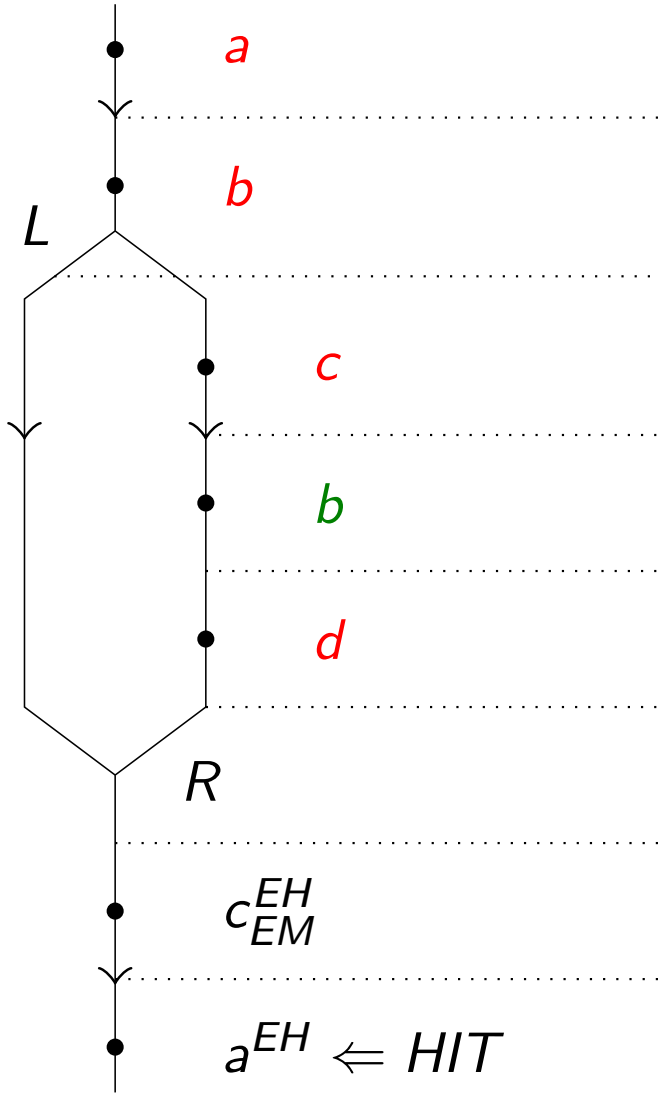
Model Checking Phase

Control Flow Graph



Model Checking Phase

Control Flow Graph



Cache Model Focused on block a

$\{\{\}\}$

$L = \{\{b\}\}$

$\{\{b, c\}\}$

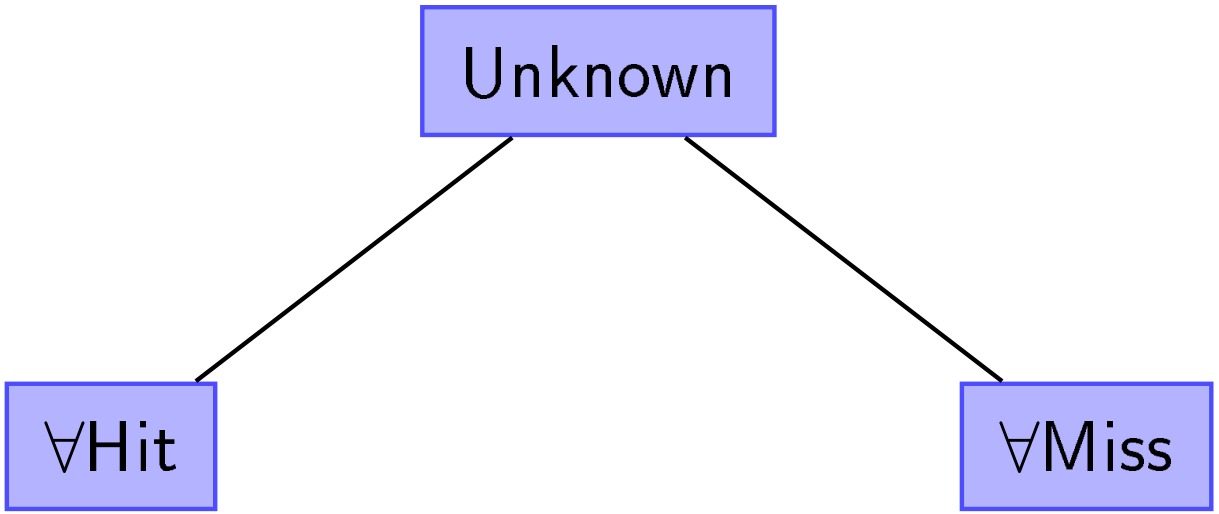
$\{\{b, c\}\}$

$R = \{\{b, c, d\}\}$

$L \cup R = \{\{b\}, \{b, c, d\}\}$

$\{\{b, c\}, \{b, c, d\}\}$

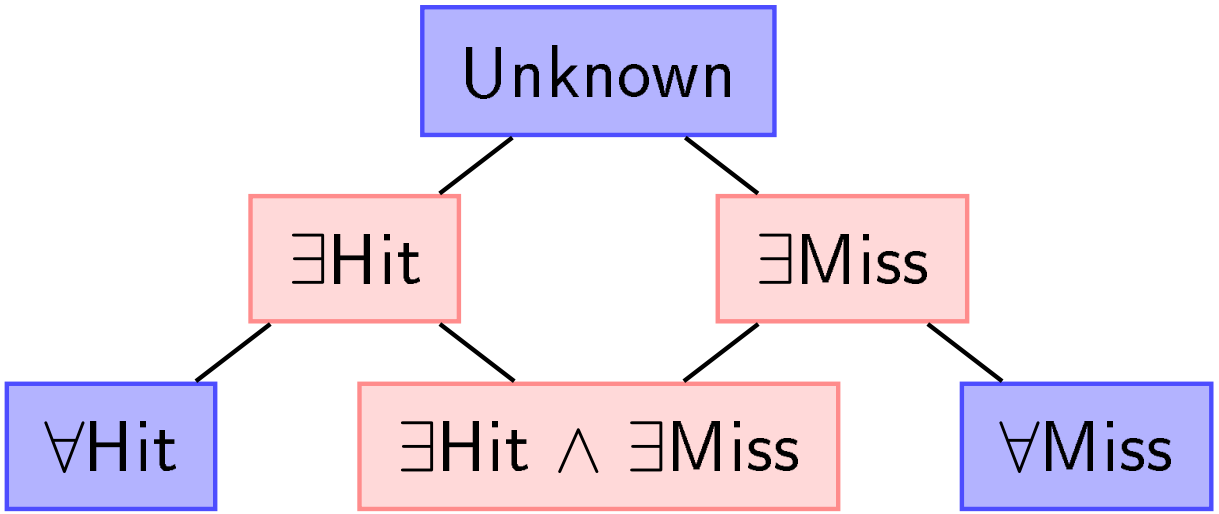
Classification



Legend:

Classical AI

Classification

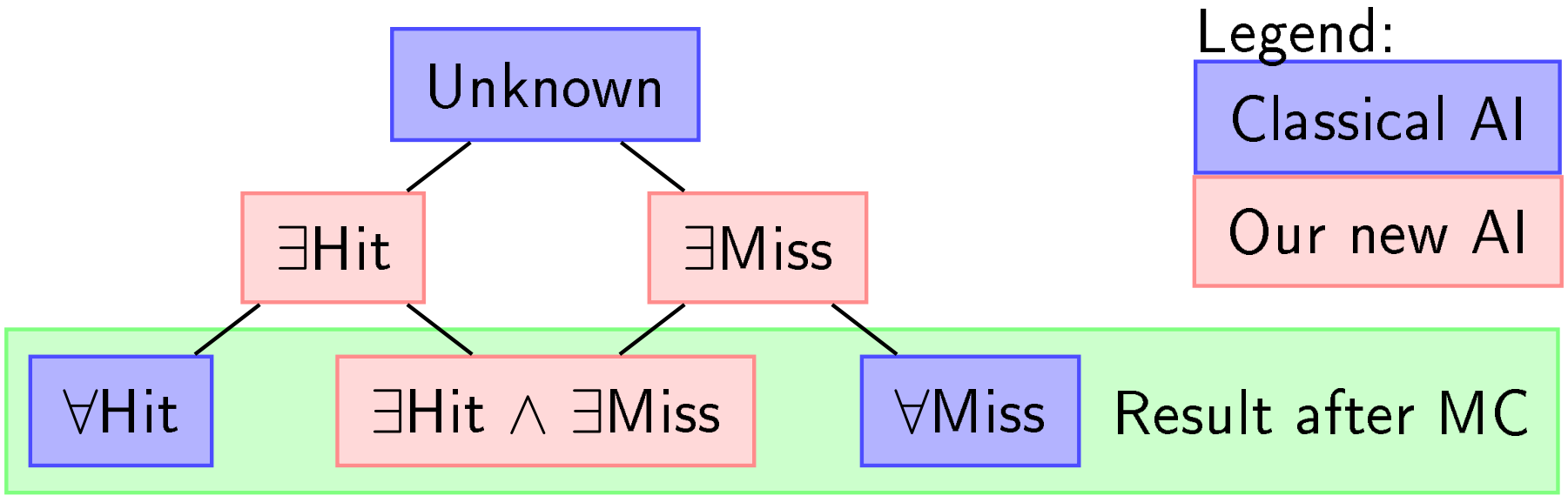


Legend:

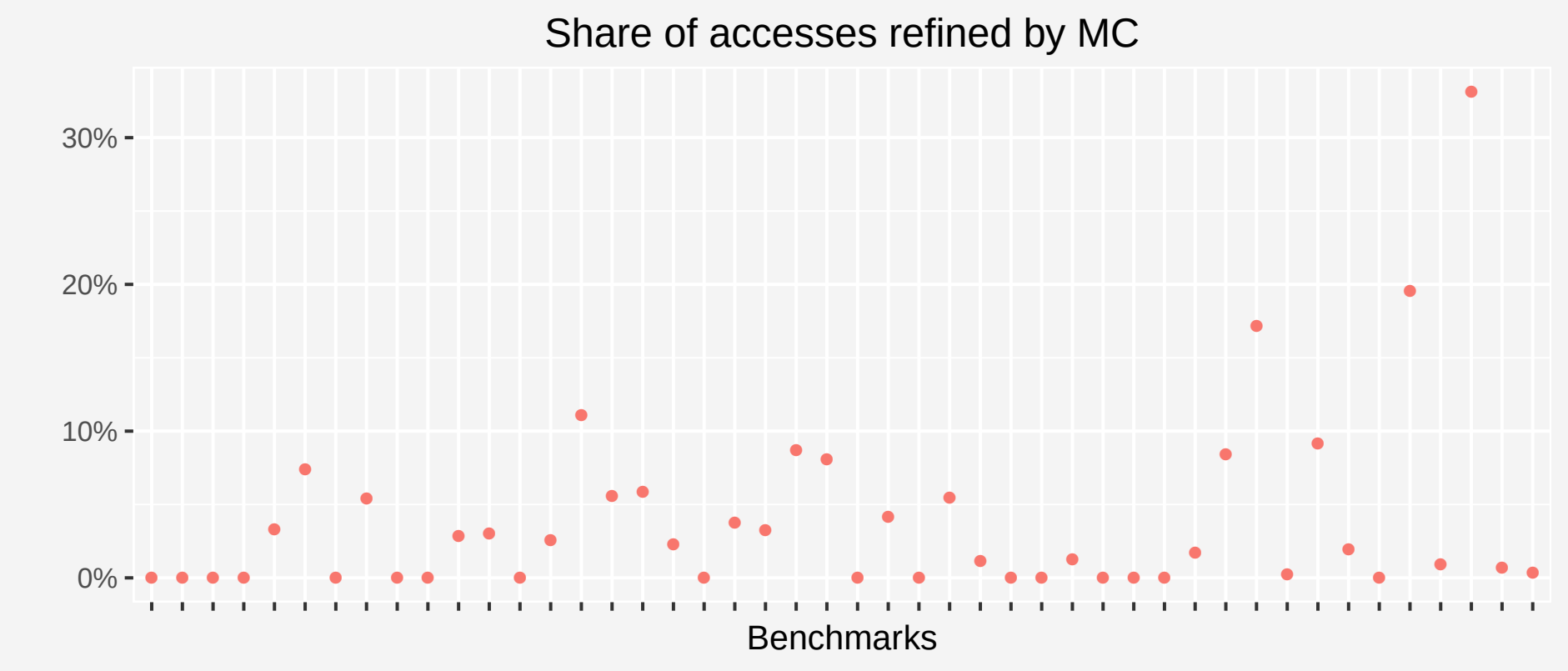
Classical AI

Our new AI

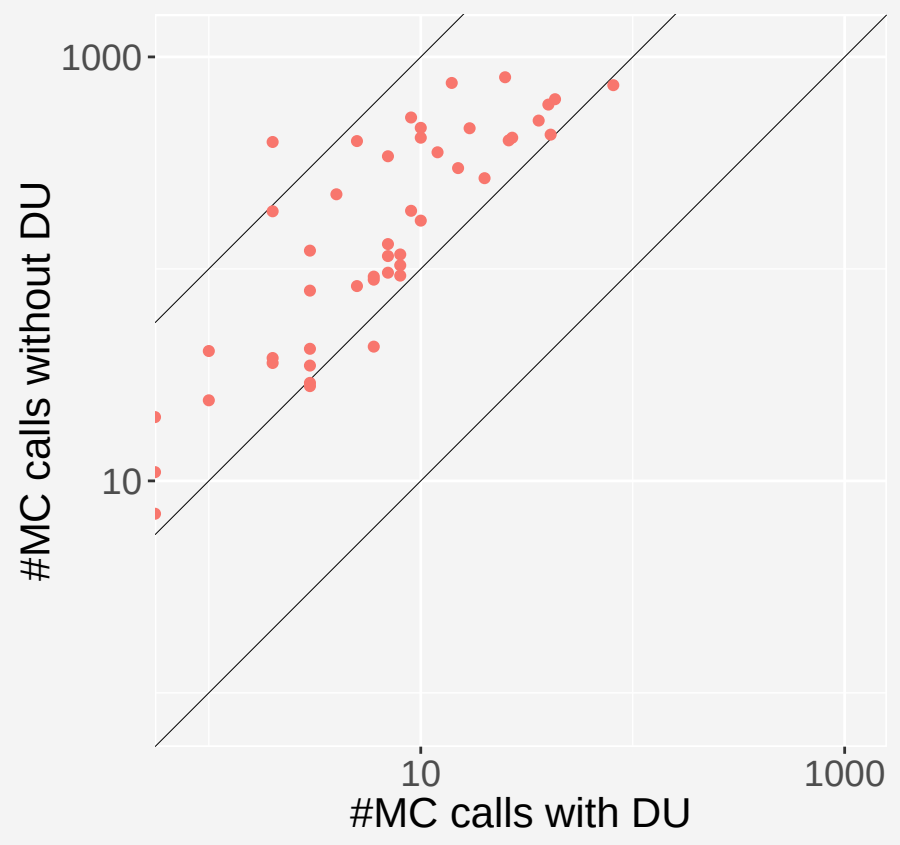
Classification

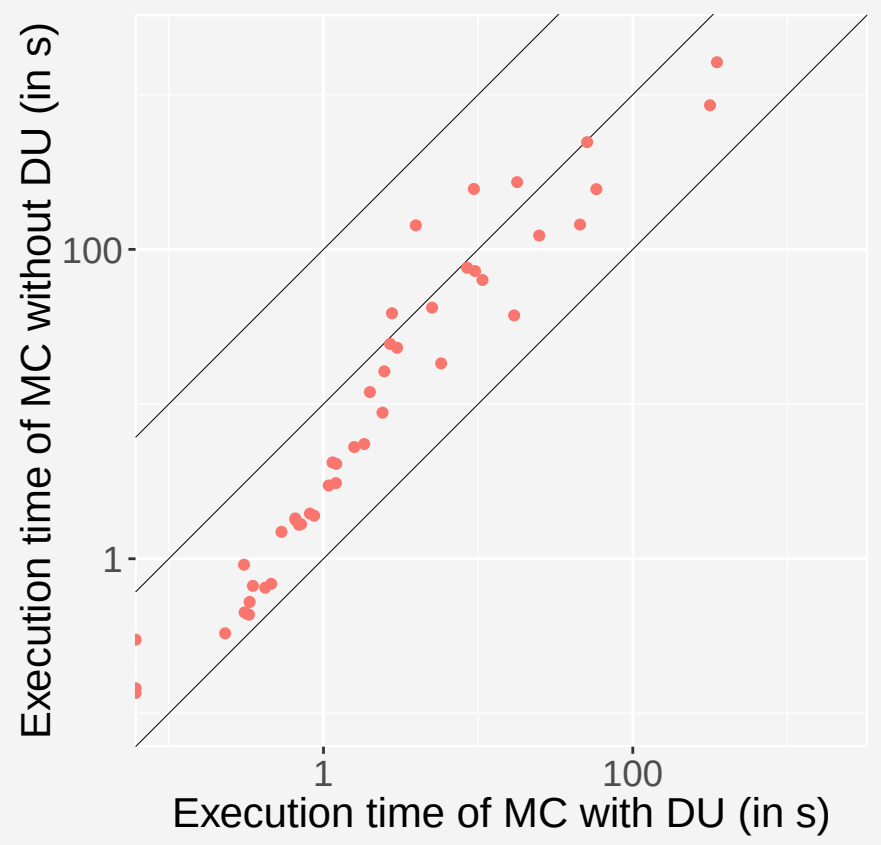


Results



Results


1000
10
#MC calls without DU
10
1000
#MC calls with DU


100
1
Execution time of MC without DU (in s)
1
100
Execution time of MC with DU (in s)

Conclusion

Contributions

- An efficient cache model to derives an exact classification by MC.
- A new analysis to reduce the cost of calling the model checker.

Future work

- Benefits in term of WCET estimation / leakage bound.
- Extension to other replacement policy.
- Adds functional semantics to the model.

Questions ?

References I


Daniel J. Bernstein, *Cache-timing attacks on AES*, 2005.


Anne Canteaut, Cédric Lauradoux, and André Seznec, *Understanding cache attacks*, Tech. Report 5881, INRIA, April 2006.


Christian Ferdinand, *Cache behavior prediction for real-time systems*, Pirrot, 1997.


John L. Hennessy and David A. Patterson, *Computer architecture, fifth edition: A quantitative approach*, 5th ed., Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2011.


Valentin Touzeau, Claire Maiza, David Monniaux, and Jan Reineke, *Ascertaining Uncertainty for Efficient Exact Cache Analysis*, CAV, 2017.