Static Probabilistic Timing Analysis for Multi-path Programs

Benjamin Lesage, David Griffin, Sebastian Altmeyer, Robert I. Davis

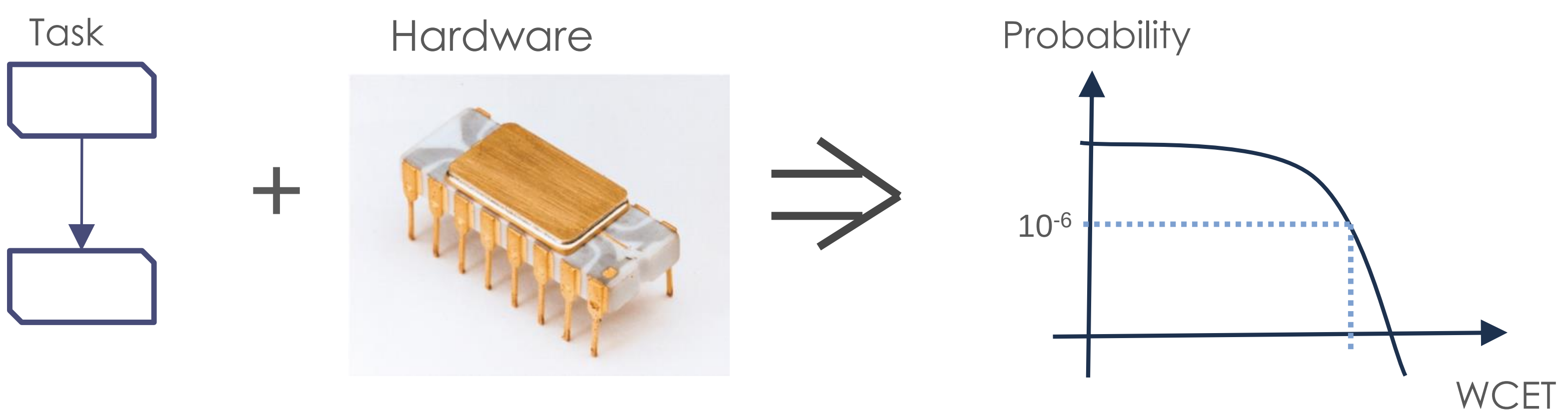
RTSS 2015 – Dec 4th

UNIVERSITÉ DU LUXEMBOURG

UNIVERSITY of York

PROXIMA

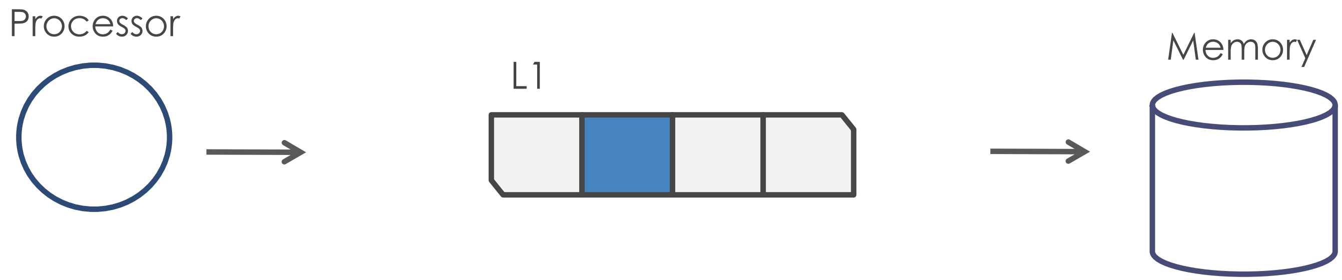
Context
pWCET estimation



- **pWCET:** WCET with attached exceedance probability
  - Bound the occurrence of events in the system
  - Match industry standard
  - Less pessimistic than absolute bounds

Context

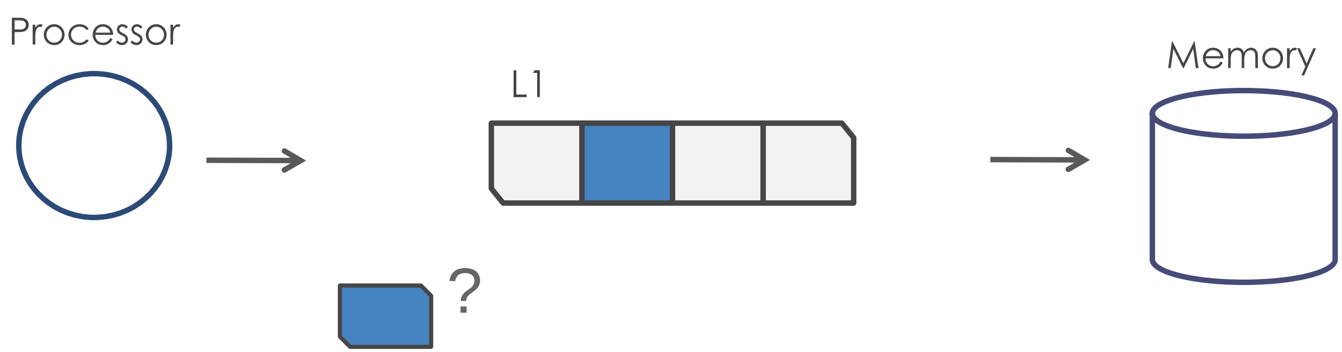
Randomised caches



- Caches bridge the gap between the processor and the memory.
  - Memory requests are served by the cache on hits.

Context

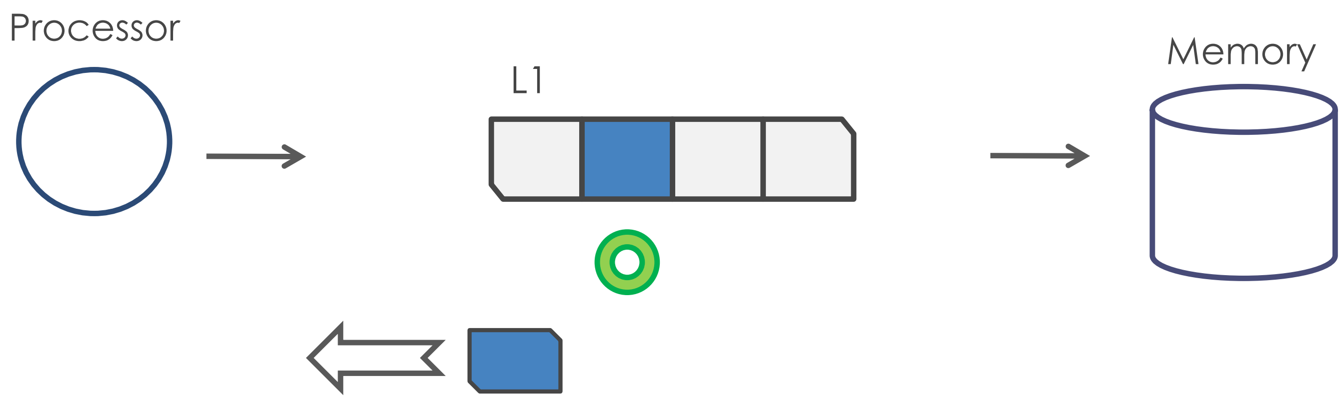
Randomised caches



- Caches bridge the gap between the processor and the memory.
  - Memory requests are served by the cache on hits.

Context

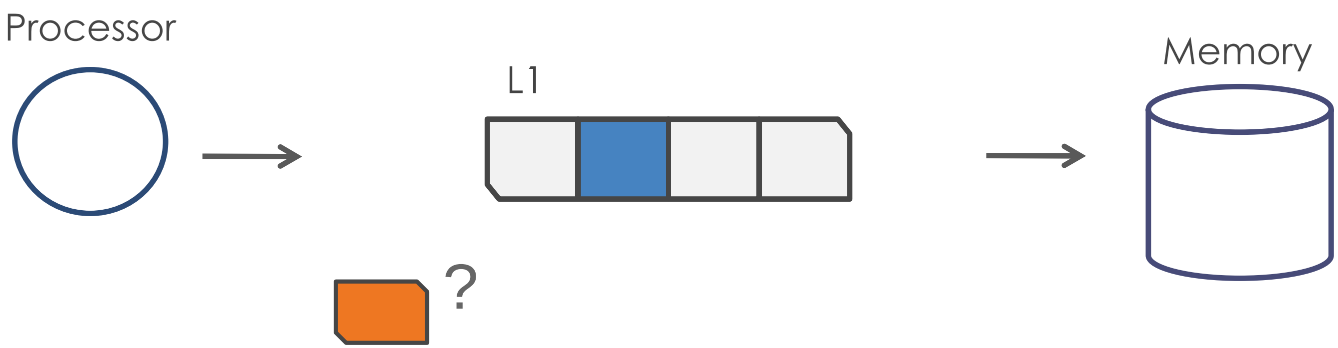
Randomised caches



- Caches bridge the gap between the processor and the memory.
  - Memory requests are served by the cache on hits.

Context

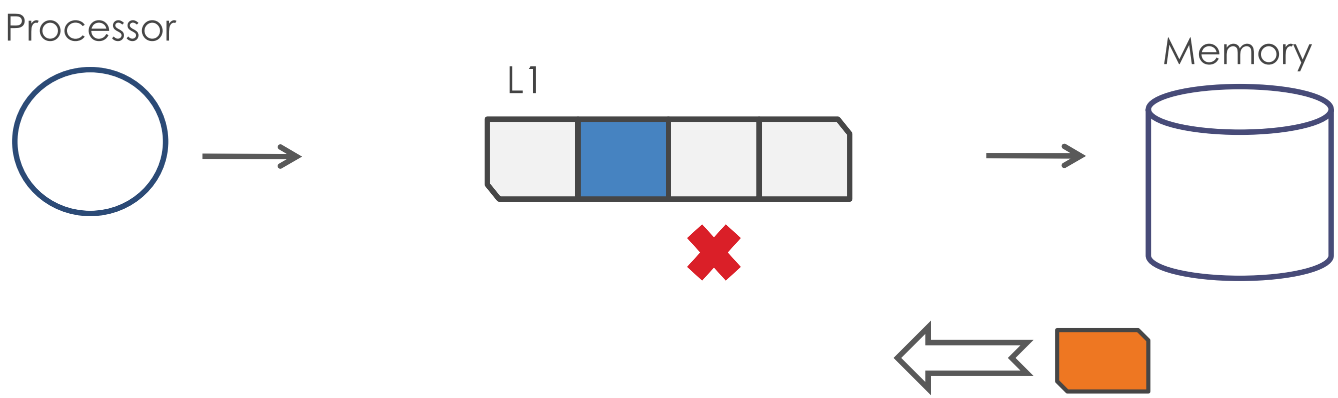
Randomised caches



- Caches bridge the gap between the processor and the memory.
  - Memory requests are served by the cache on hits.
- On a miss, the requested data is inserted in the cache.
  - The data is expected to be reused (locality property).
  - The eviction policy makes room in the cache.

Context

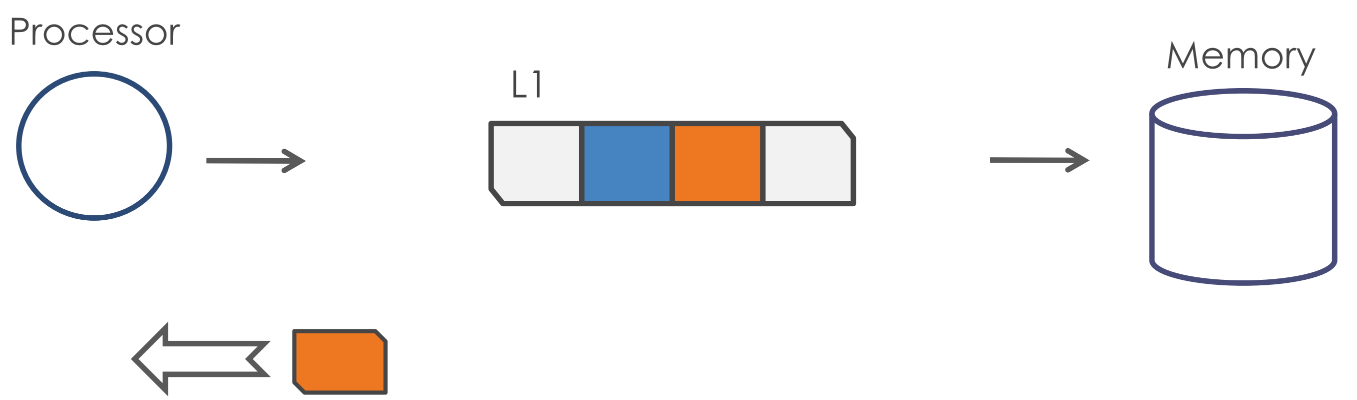
Randomised caches



- Caches bridge the gap between the processor and the memory.
  - Memory requests are served by the cache on hits.
- On a miss, the requested data is inserted in the cache.
  - The data is expected to be reused (locality property).
  - The eviction policy makes room in the cache.

Context

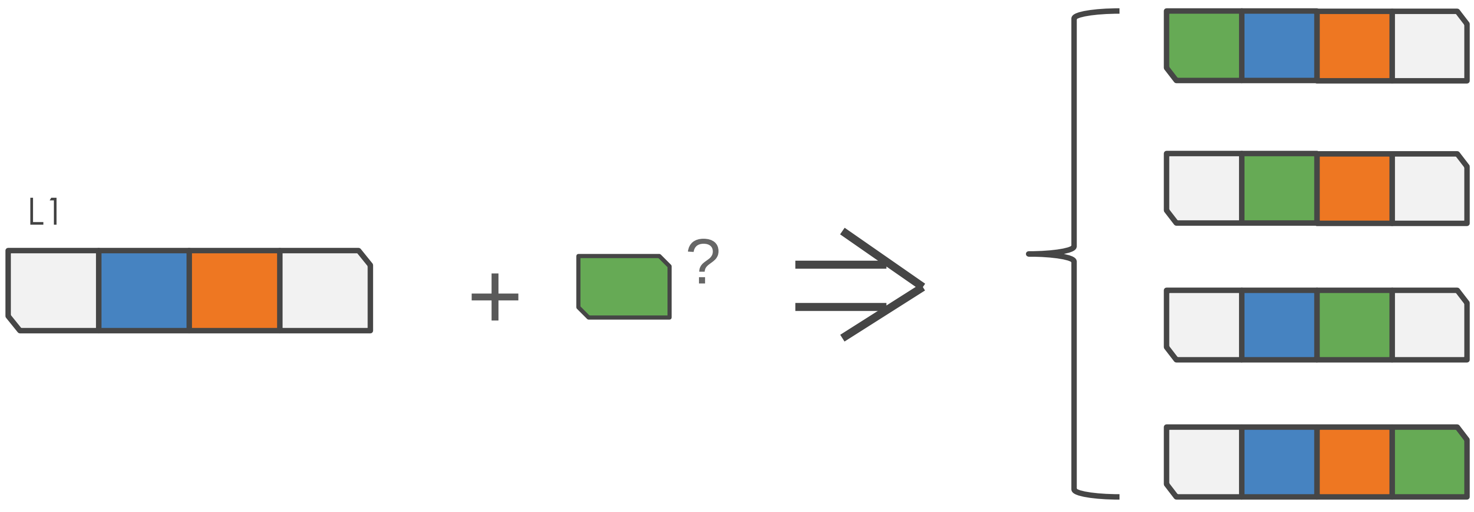
Randomised caches



- Caches bridge the gap between the processor and the memory.
  - Memory requests are served by the cache on hits.
- On a miss, the requested data is inserted in the cache.
  - The data is expected to be reused (locality property).
  - The eviction policy makes room in the cache.

Context

Evict-on-miss replacement policy



- On a miss, evict one of the N cache lines at random.
  - Provide a model suited to pWCET computation.
  - On a miss, each line has the same probability to be kept: $1-\left(\frac{1}{N}\right)=\left(\frac{N-1}{N}\right)$
    - After K misses: $\left(\frac{N-1}{N}\right)^K$

Context

Static probabilistic timing analysis (SPTA)

- **Contention approach:** lower-bound hit probability $P(H^{L1})$ per access.
  - Derive a Probability Mass Function (PMF) for access latency.
  - Convolve the PMF of all accesses.
    - Requires the independence of the bound from actual hit/miss events.

$$P(H^{L1}) = \left(\frac{N-1}{N}\right)^{K} \longrightarrow PMF = \begin{pmatrix} L_{L1} & L_{Mem} \\ P(H^{L1}) & 1 - P(H^{L1}) \end{pmatrix}$$

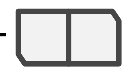
- $\boldsymbol{K}$: Reuse distance, misses from the last insertion in cache.
- $\boldsymbol{N}$: Associativity, number of cache ways.
- $\boldsymbol{L_{L1}}$: Access latency to L1 cache.

Context

Static probabilistic timing analysis (SPTA)

- **Contention approach:** lower-bound hit probability $P(H^{L1})$ per access.
  - Derive a Probability Mass Function (PMF) for access latency.
  - Convolve the PMF of all accesses.
    - Requires the independence of the bound from actual hit/miss events.

$$P(H^{L1}) = \left(\frac{N-1}{N}\right)^{K} \longrightarrow PMF = \begin{pmatrix} L_{L1} & L_{Mem} \\ P(H^{L1}) & 1 - P(H^{L1}) \end{pmatrix}$$

a,b,c,a,b,c +  = 3 predicted hits

- $\boldsymbol{K}$: Reuse distance, misses from the last insertion in cache.
- $\boldsymbol{N}$: Associativity, number of cache ways.
- $\boldsymbol{L_{L1}}$: Access latency to L1 cache.

Context

Static probabilistic timing analysis (SPTA)

- **Contention approach:** lower-bound hit probability $P(H^{L1})$ per access.
  - Derive a Probability Mass Function (PMF) for access latency.
  - Convolve the PMF of all accesses.
    - Requires the independence of the bound from actual hit/miss events.

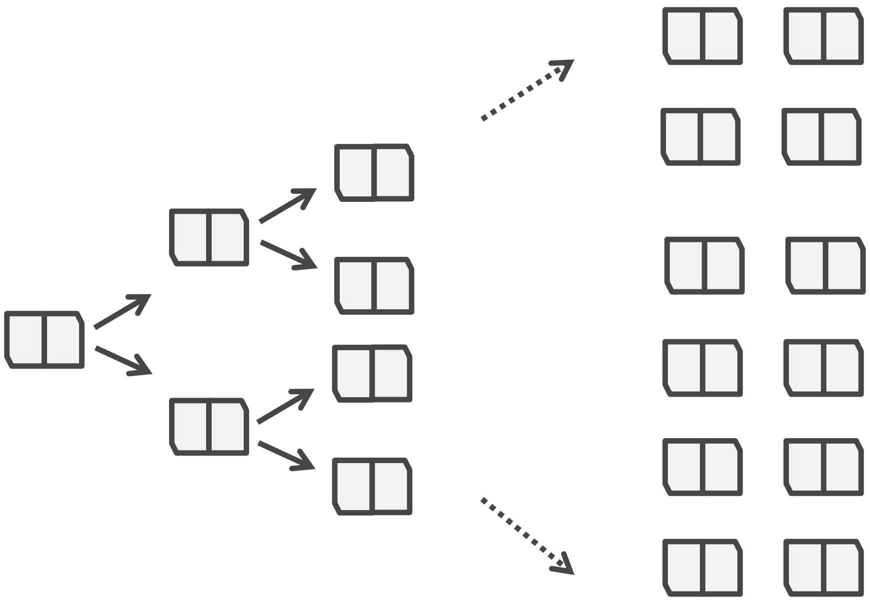
$$P(H^{L1}) = \begin{cases} 0 & Co > N \\ \left(\frac{N-1}{N}\right)^{K} & K \leq N \end{cases} \longrightarrow PMF = \begin{pmatrix} L_{L1} & L_{Mem} \\ P(H^{L1}) & 1 - P(H^{L1}) \end{pmatrix}$$

- $\boldsymbol{K}$: Reuse distance, misses from the last insertion in cache.
- $\boldsymbol{Co}$: Contention, potential hits from the last insertion in cache.
- $\boldsymbol{N}$: Associativity, number of cache ways.
- $\boldsymbol{L_{L1}}$: Access latency to L1 cache.

Context

Static probabilistic timing analysis (SPTA)

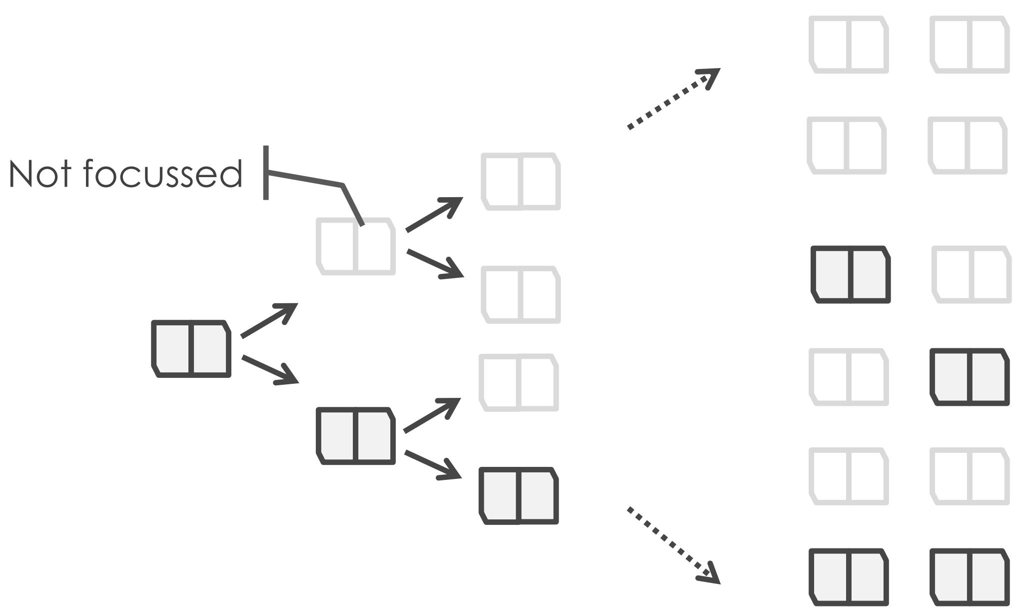
- **Collection approach:** approximate the set of possible cache states
  - With the execution time and occurrence probability.
  - A miss creates a new state per cache line.



Context

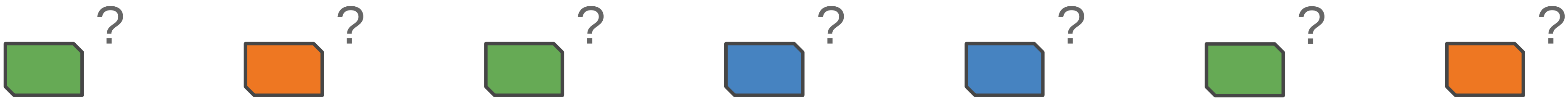
Static probabilistic timing analysis (SPTA)

- **Collection approach:** approximate the set of possible cache states
  - With the execution time and occurrence probability.
  - A miss creates a new state per cache line.
  - Focus on a subset of blocks to reduce complexity.



Context

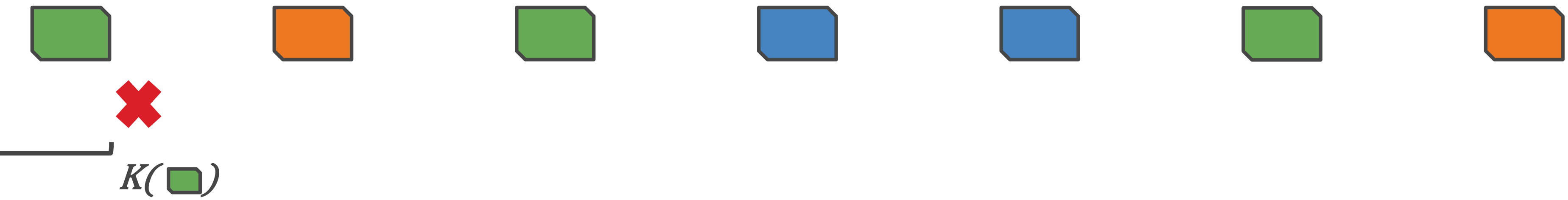
SPTA on Access traces



- SPTA has been defined for traces of accesses
  - Select focused blocks
  - Perform contention and collection analysis
  - Combine computed distributions

Context

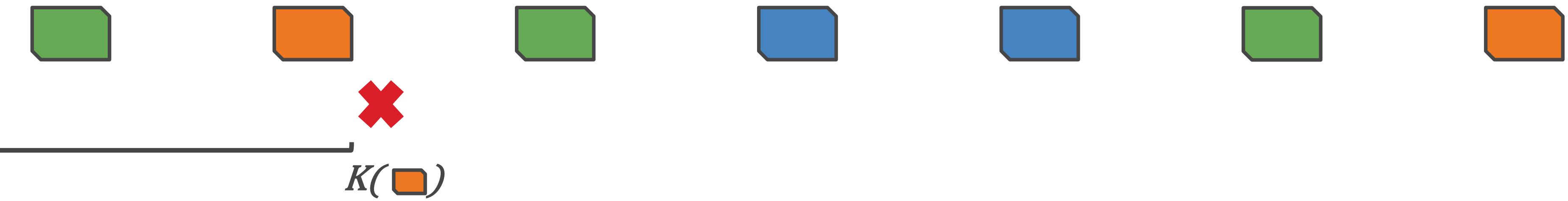
SPTA on Access traces



- SPTA has been defined for traces of accesses
  - Select focused blocks
  - Perform contention and collection analysis
  - Combine computed distributions

Context

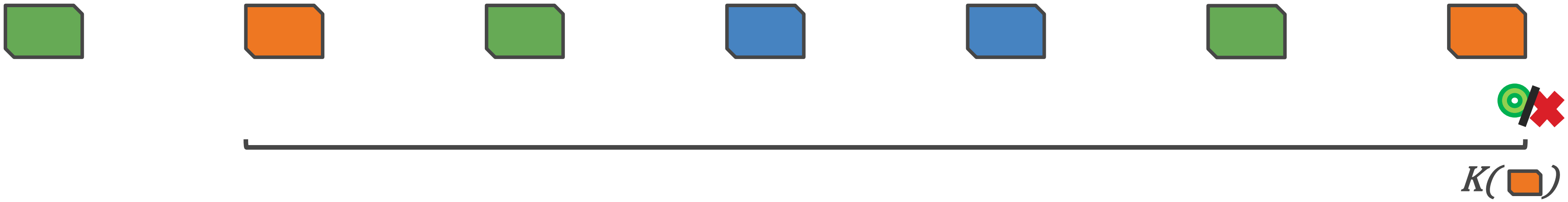
SPTA on Access traces



- SPTA has been defined for traces of accesses
  - Select focused blocks
  - Perform contention and collection analysis
  - Combine computed distributions

Context

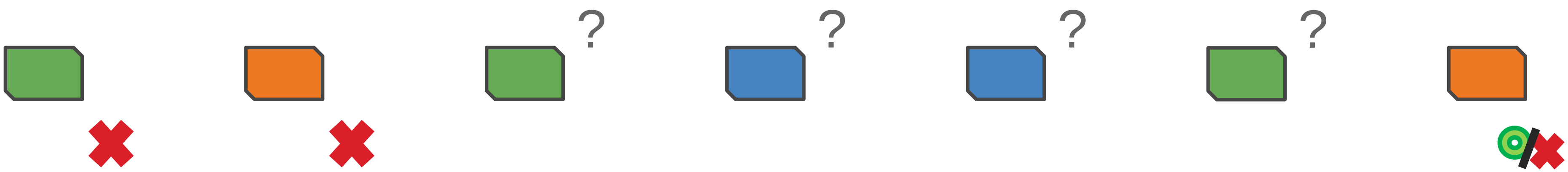
SPTA on Access traces



- SPTA has been defined for traces of accesses
  - Select focused blocks
  - Perform contention and collection analysis
  - Combine computed distributions

Context

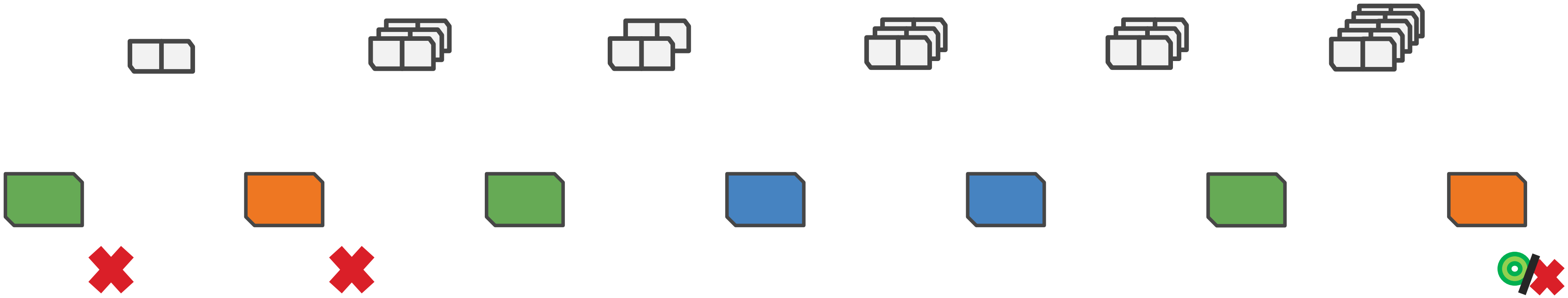
SPTA on Access traces



- SPTA has been defined for traces of accesses
  - Select focused blocks
  - Perform contention and collection analysis
  - Combine computed distributions

Context

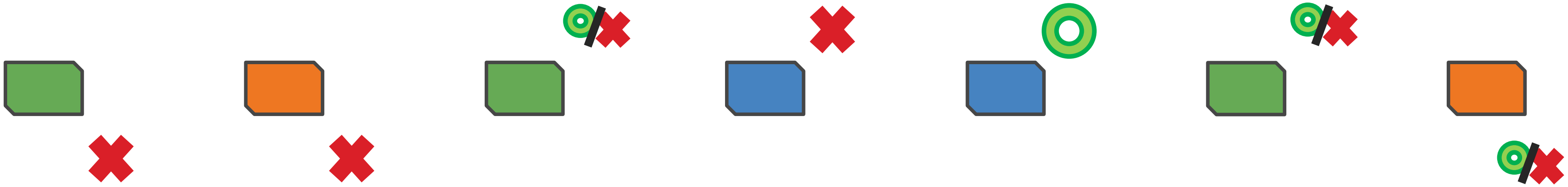
SPTA on Access traces



- SPTA has been defined for traces of accesses
  - Select focused blocks
  - Perform contention and collection analysis
  - Combine computed distributions

Context

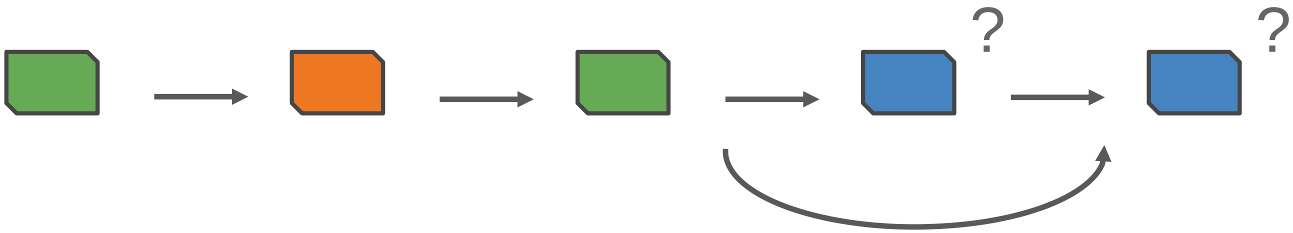
SPTA on Access traces



- SPTA has been defined for traces of accesses
  - Select focused blocks
  - Perform contention and collection analysis
  - Combine computed distributions

Context

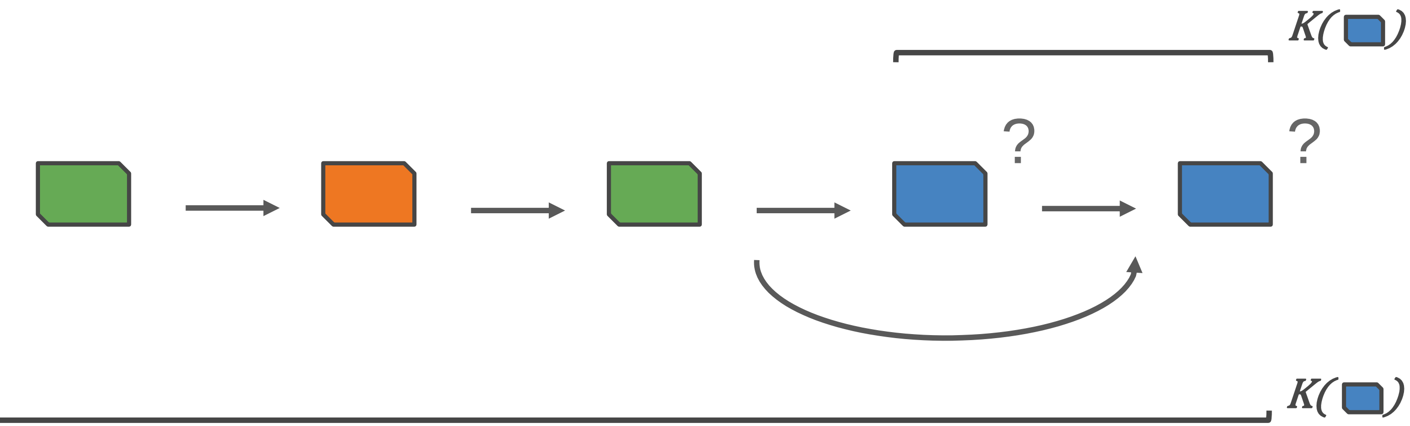
SPTA on Control flow graphs



How to extend existing approaches to control flow graphs ?

Context

SPTA on Control flow graphs



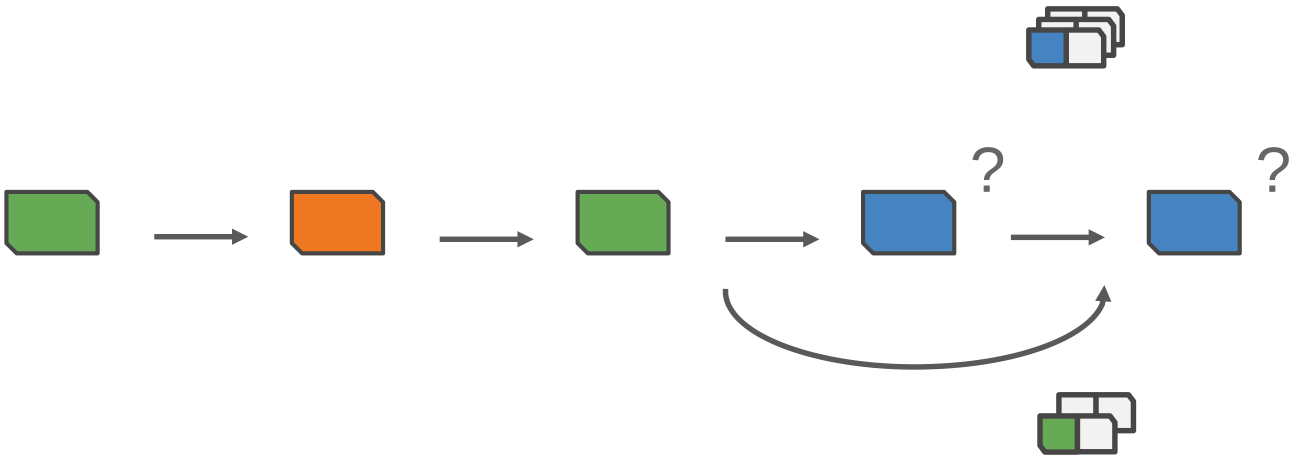
How to extend existing approaches to control flow graphs ?

- Contention analysis
- Focused blocks selection

Collection analysis

Context

SPTA on Control flow graphs



How to extend existing approaches to control flow graphs ?

- Contention analysis
- Focused blocks selection
- Collection analysis

Outline

- Context
- Multipath SPTA
  - Contention analysis
  - Selecting focussed blocks
  - Collection analysis
- WCEP Expansion
  - Definition of Including paths
  - Transformations
- Evaluation
- Conclusions and perspectives

Multipath analysis

Contention analysis

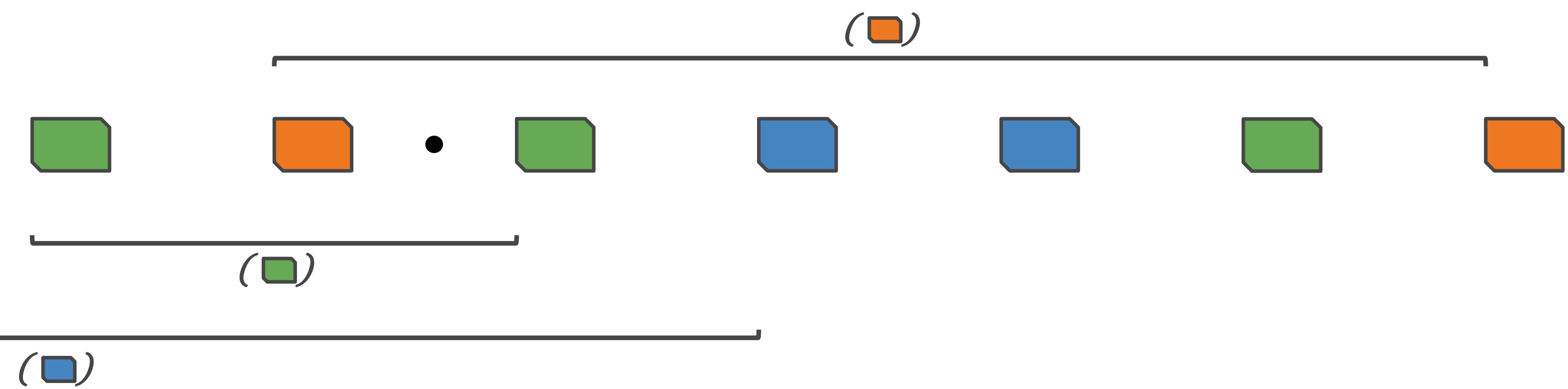
$$P(H^{L1}) = \begin{cases} 0 & Co > N \\ \left(\frac{N-1}{N}\right)^{K} & K \leq N \end{cases}$$

- $\boldsymbol{K}$: Reuse distance, maximum misses from the last insertion in cache.
  - Maximised across all paths leading to access
  - Computed through forward dataflow analysis
- $\boldsymbol{Co}$: Contention, maximum potential hits from the last insertion in cache.
- $\boldsymbol{N}$: Associativity, number of cache ways.
- $\boldsymbol{L_{L1}}$: Access latency to L1 cache.

Multipath SPTA

Selecting focussed blocks

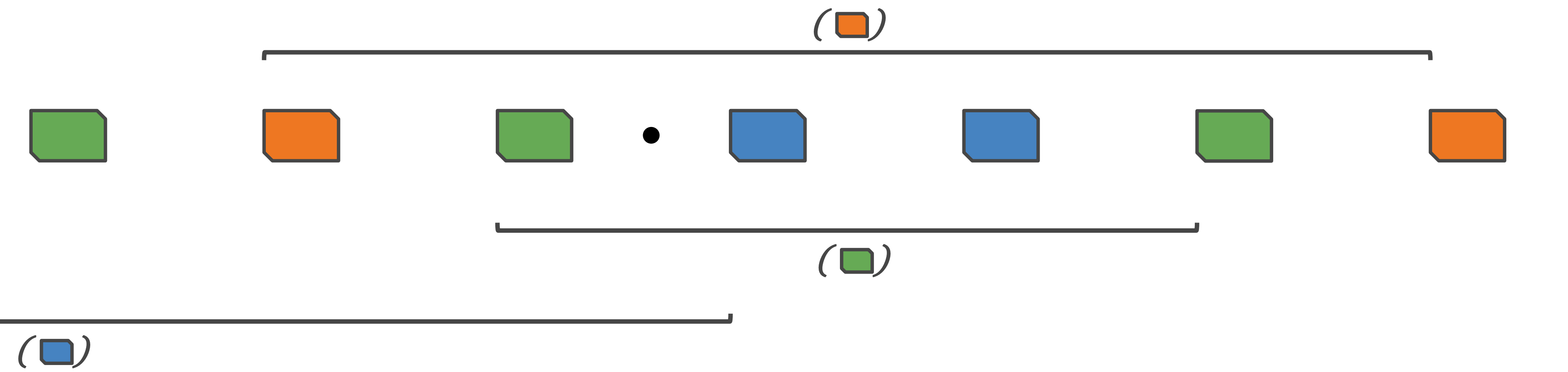
- Enumerate cache states only for $\boldsymbol{R}$ focussed blocks
  - $\boldsymbol{R}$ must change according to path
  - $\boldsymbol{R}$ may change at different points in task
- Focus on blocks with smallest lifespan
  - Most likely to be kept in cache
  - Relies on a lower bound
  - Combines forward and backward reuse distance



Multipath SPTA

Selecting focussed blocks

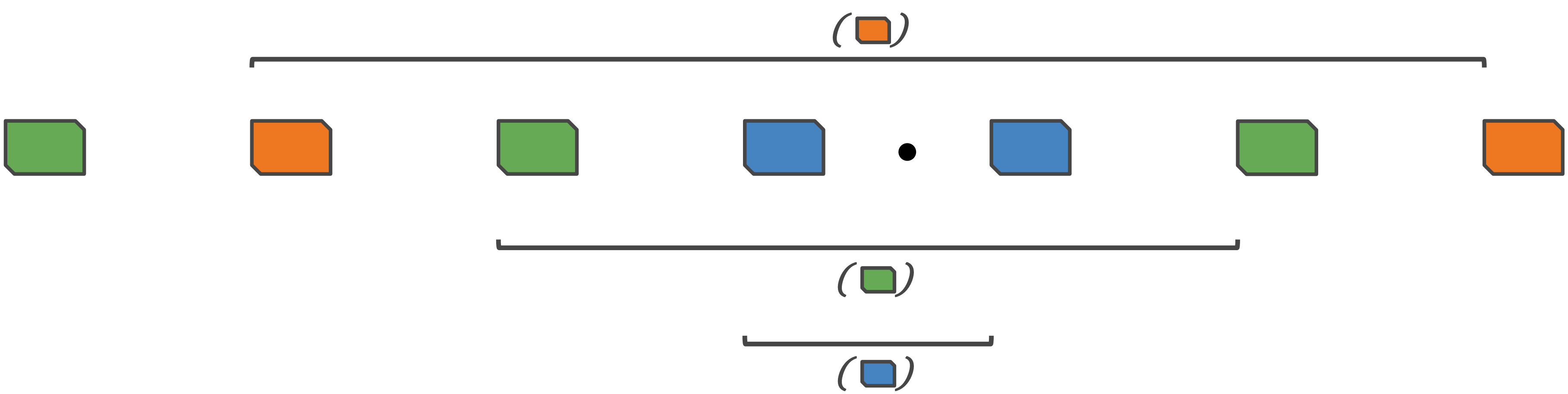
- Enumerate cache states only for $\boldsymbol{R}$ focussed blocks
  - $\boldsymbol{R}$ must change according to path
  - $\boldsymbol{R}$ may change at different points in task
- Focus on blocks with smallest lifespan
  - Most likely to be kept in cache
  - Relies on a lower bound
  - Combines forward and backward reuse distance



Multipath SPTA

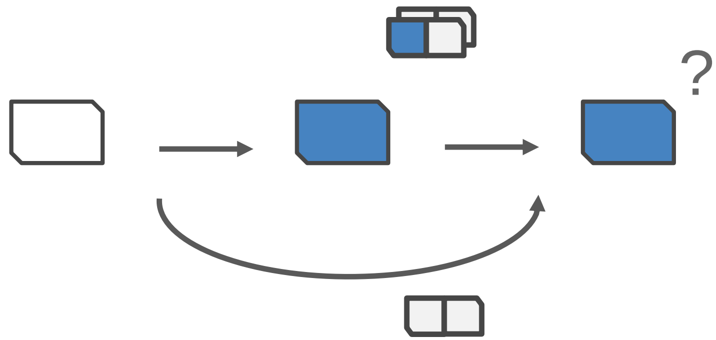
Selecting focussed blocks

- Enumerate cache states only for $\boldsymbol{R}$ focussed blocks
  - $\boldsymbol{R}$ must change according to path
  - $\boldsymbol{R}$ may change at different points in task
- Focus on blocks with smallest lifespan
  - Most likely to be kept in cache
  - Relies on a lower bound
  - Combines forward and backward reuse distance



Multipath analysis

Collection – Control flow convergence



- Analysis state holds a set of :
  - Cache contents
  - Occurrence probability
  - Maximum execution time distribution
- Gather information from all incoming paths
  - Only keep guaranteed information
  - Upper-bound incoming states

Multipath analysis

Collection – Comparison between cache states

- $\boldsymbol{S_a} \sqsubseteq \boldsymbol{S_b}$, S_a results in less pessimistic estimates
  - S_a holds more precise information than S_b
  - $\sqsubseteq$: Partial ordering between set of cache states

-  $\sqsubseteq$ 
  - Loss of information related to cache contents

- ( ,**1**) $\sqsubseteq$ {( ,**1/3**), ( ,**2/3**)}
  - Information split across contents

- $\boldsymbol{L} \leq \boldsymbol{U} \Rightarrow$ ( ,1,$\boldsymbol{L}$) $\sqsubseteq$ ( ,1,$\boldsymbol{U}$)
  - The contribution of U to pWCET is greater than L

Multipath analysis

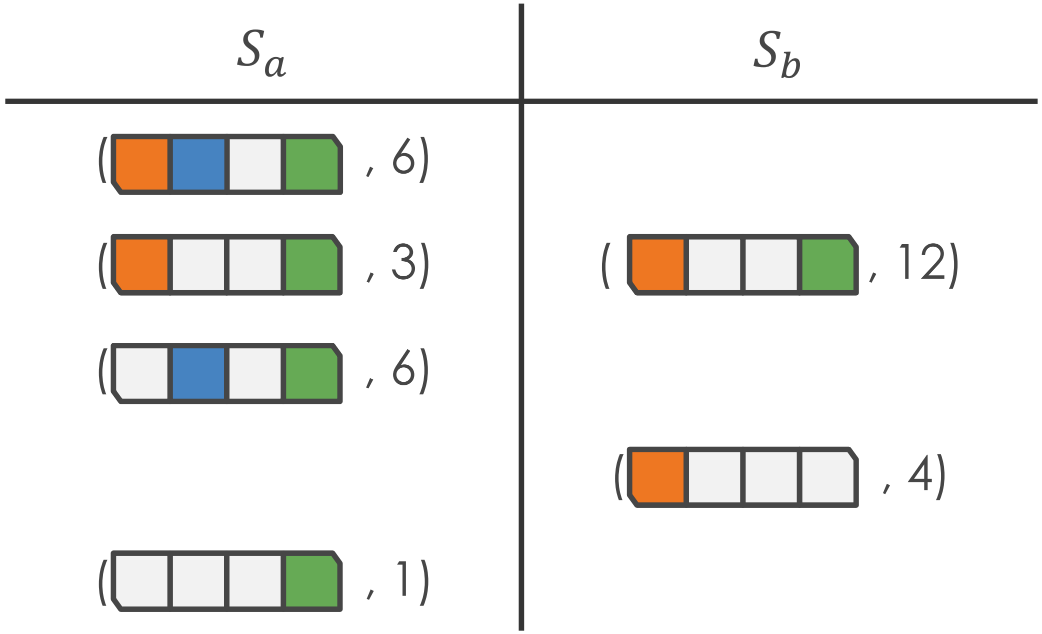
Collection – Comparison between cache states

- $\boldsymbol{S_a} \sqsubseteq \boldsymbol{S_b}$, S_a results in less pessimistic estimates
  - S_a holds more precise information than S_b
  - $\sqsubseteq$: Partial ordering between set of cache states

- $\sqcup$: compute an upper-bound on input states
  - $S_a \sqsubseteq (S_a \sqcup S_b)$ and $S_b \sqsubseteq (S_a \sqcup S_b)$
  - $\sqcup$ is a valid join function

Multipath analysis

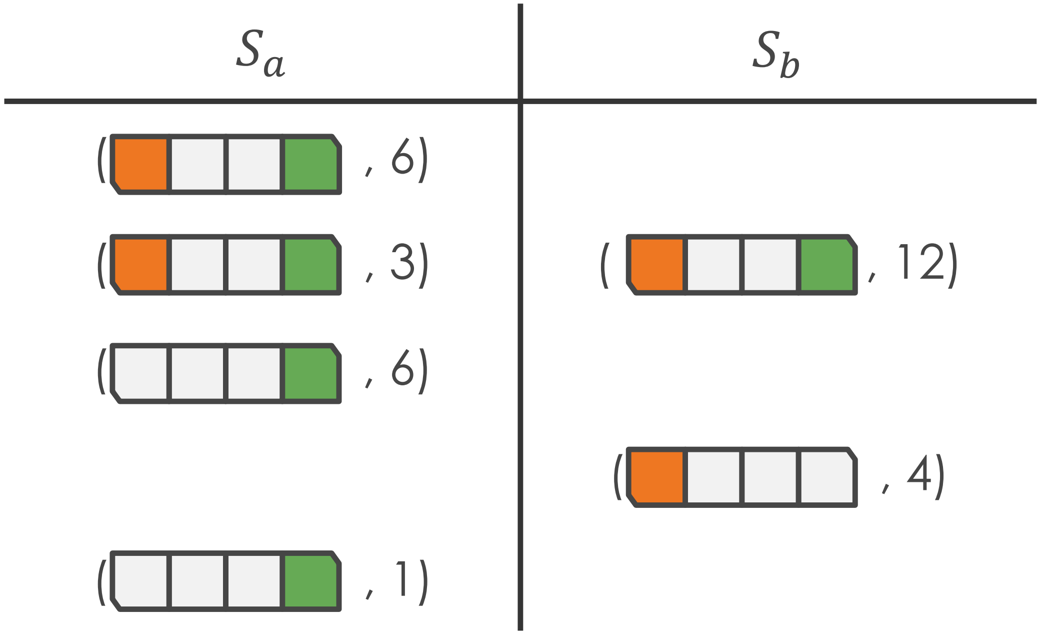
Collection – Defining a join function



- Keep only common blocks and contents
  - Occurrence bounded by lowest denominator
  - Maximise merged distributions (Omitted)
  - Merge unmatched states into empty state

Multipath analysis

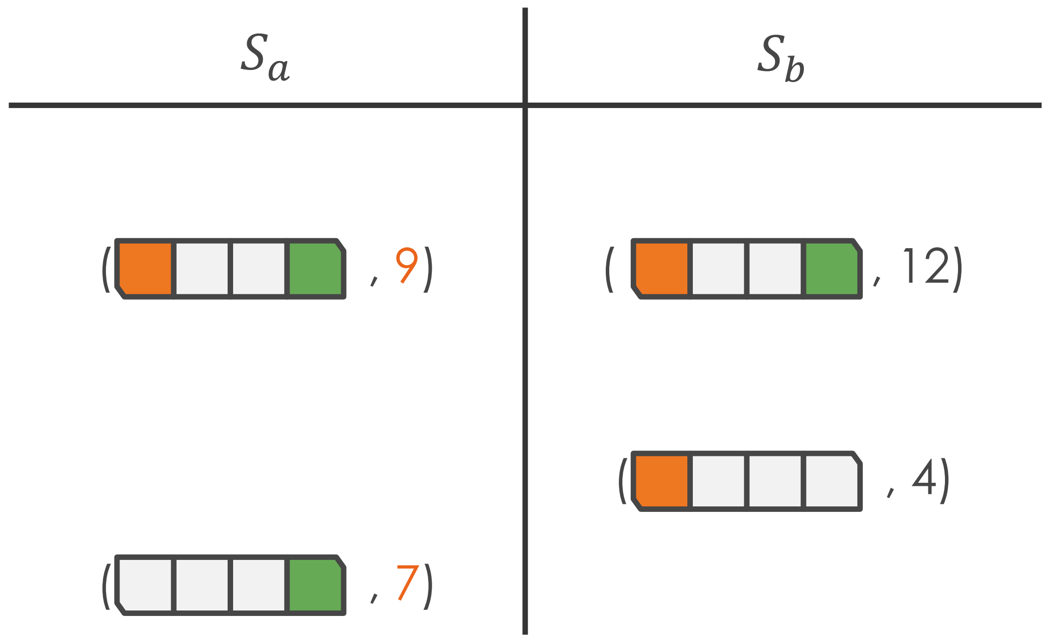
Collection – Defining a join function



- Keep only common blocks and contents
  - Occurrence bounded by lowest denominator
  - Maximise merged distributions (Omitted)
  - Merge unmatched states into empty state

Multipath analysis

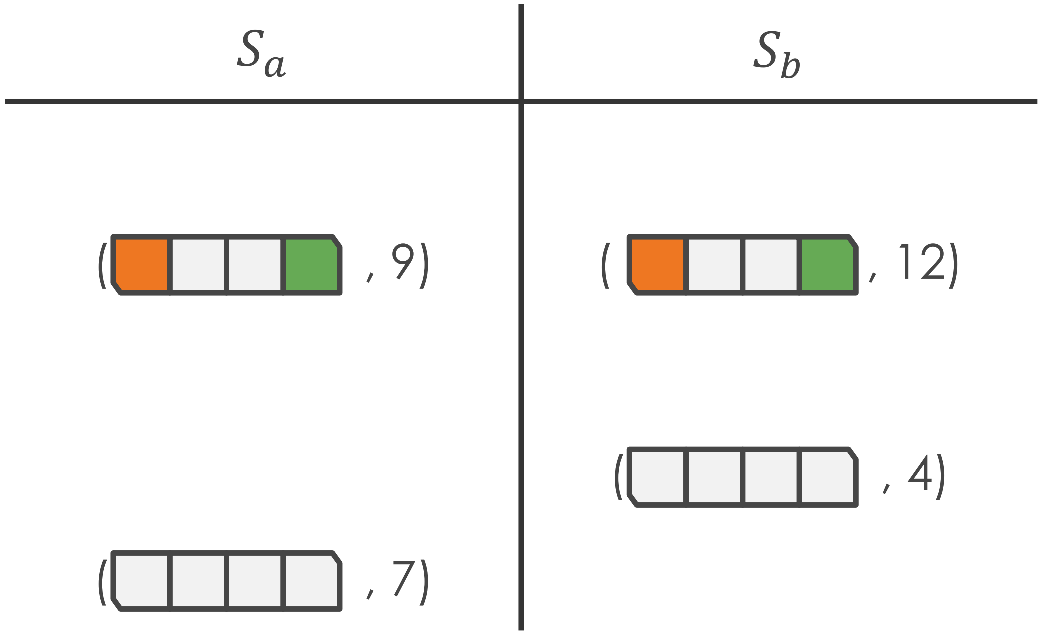
Collection – Defining a join function



- Keep only common blocks and contents
  - Occurrence bounded by lowest denominator
  - Maximise merged distributions (Omitted)
  - Merge unmatched states into empty state

Multipath analysis

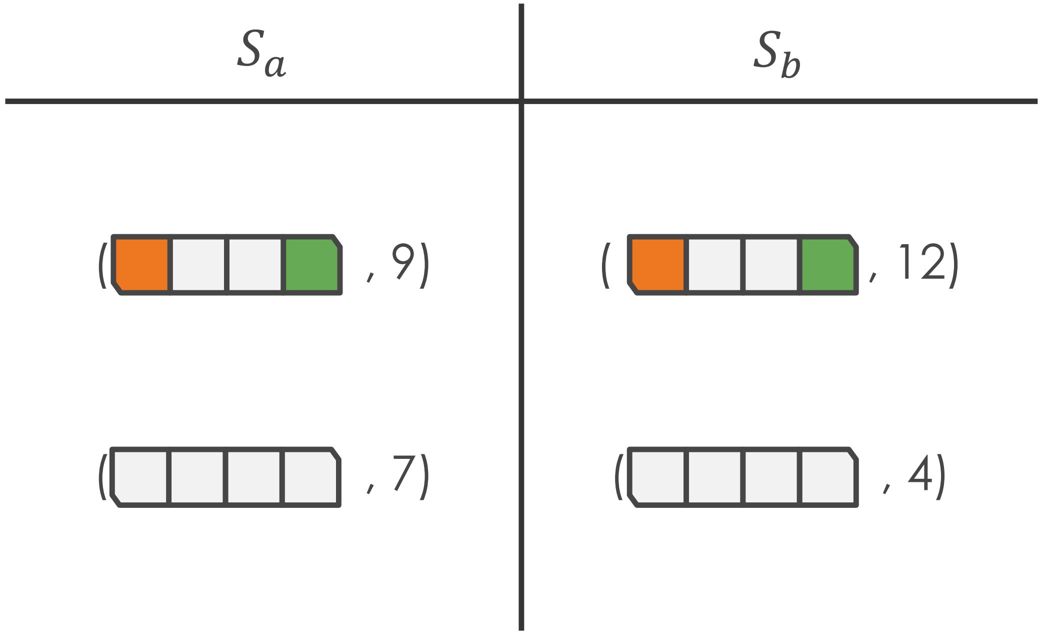
Collection – Defining a join function



- Keep only common blocks and contents
  - Occurrence bounded by lowest denominator
  - Maximise merged distributions (Omitted)
  - Merge unmatched states into empty state

Multipath analysis

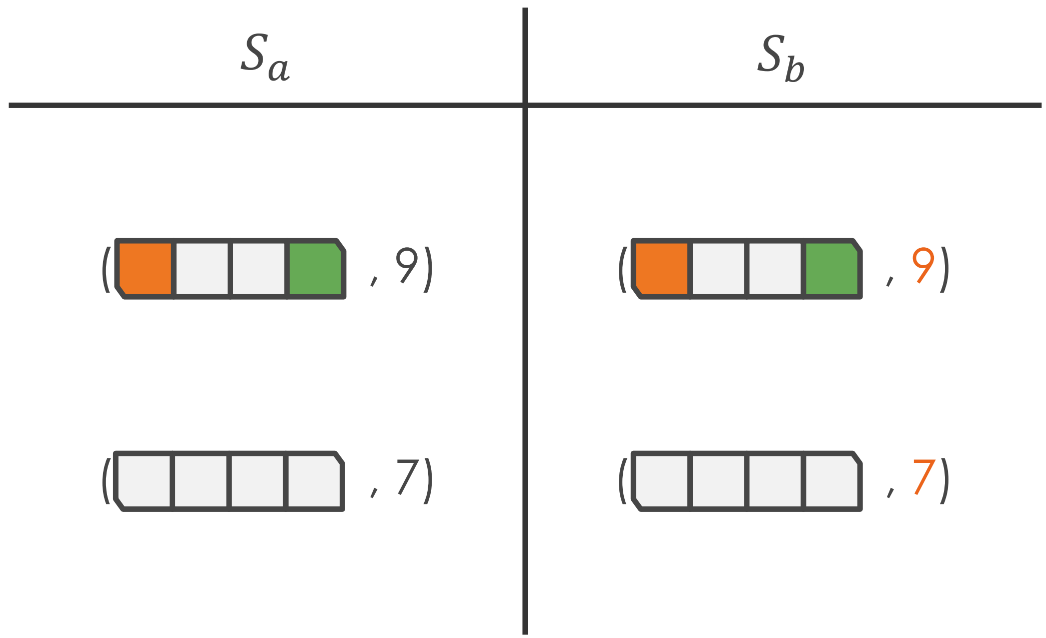
Collection – Defining a join function



- Keep only common blocks and contents
  - Occurrence bounded by lowest denominator
  - Maximise merged distributions (Omitted)
  - Merge unmatched states into empty state

Multipath analysis

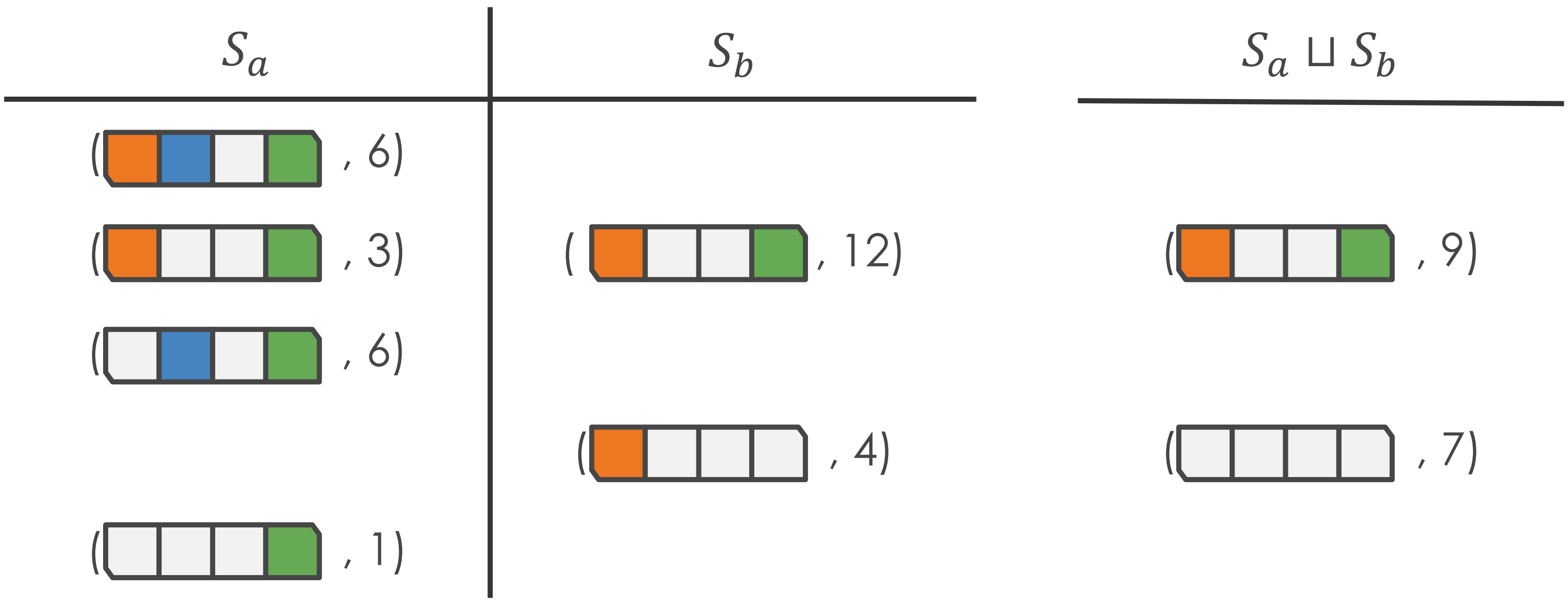
Collection – Defining a join function



- Keep only common blocks and contents
  - Occurrence bounded by lowest denominator
  - Maximise merged distributions (Omitted)
  - Merge unmatched states into empty state

Multipath analysis

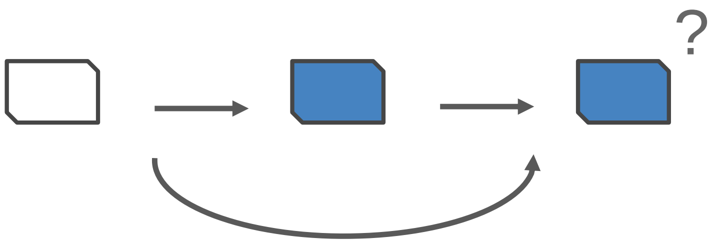
Collection – Defining a join function



- Keep only common blocks and contents
  - Occurrence bounded by lowest denominator
  - Maximise merged distributions (Omitted)
  - Merge unmatched states into empty state

WCEP Expansion

Path redundancy



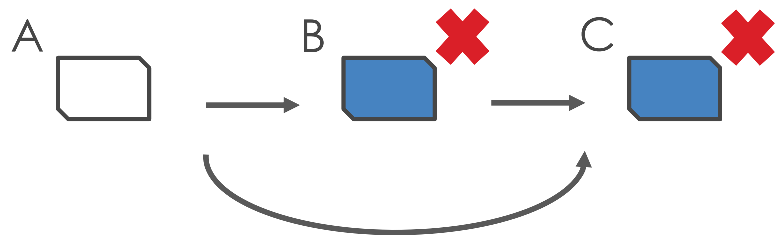
- **Redundant path:** path P_1 is redundant with path P_2 if:
  - $pWCET(P_1) \leq pWCET(P_2)$

Redundant paths can be ignored by the analysis

- **Inclusion** is a sub-case of redundancy
  - An **including** path holds at least the same sequence of accesses
  - Proof in the paper
  - Exploited in MBPTA, [PUB: Path Upper-Bounding, ECRTS'14]

WCEP Expansion

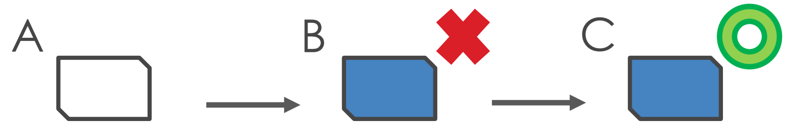
Transformation - Empty conditional removals



- **Empty branches** generate including paths
  - An edge from A to C, C also reached through B from A
  - Empty branches Captured through dominators in CFG

WCEP Expansion

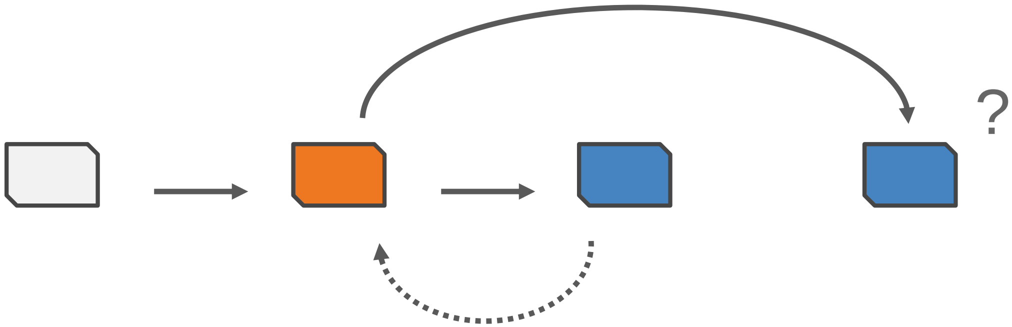
Transformation - Empty conditional removals



- **Empty branches** generate including paths
  - An edge from A to C, C also reached through B from A
  - Empty branches Captured through dominators in CFG

WCEP Expansion

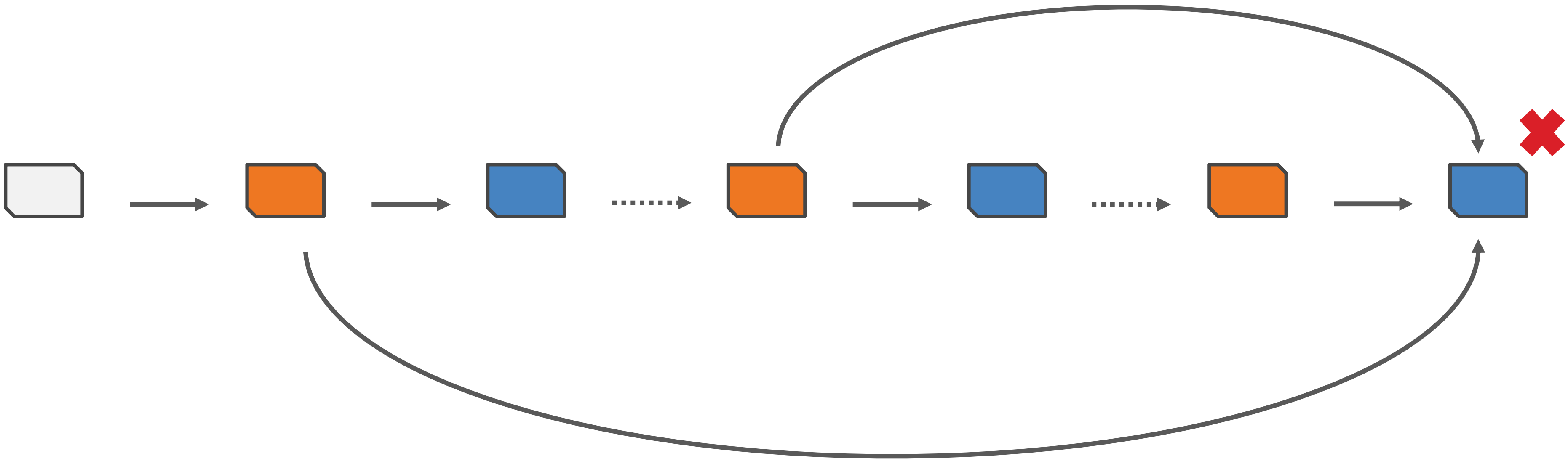
Transformation - Loop unrolling



- **Loop unrolling** generates including paths
  - Virtual unrolling used in the absence of fixed-point computation
  - Enforce maximum loop iterations

WCEP Expansion

Transformation - Loop unrolling



- **Loop unrolling** generates including paths
  - Virtual unrolling used in the absence of fixed-point computation
  - Enforce maximum loop iterations

WCEP Expansion

Transformation - Loop unrolling

- **Loop unrolling** generates including paths
  - Virtual unrolling used in the absence of fixed-point computation
  - Enforce maximum loop iterations

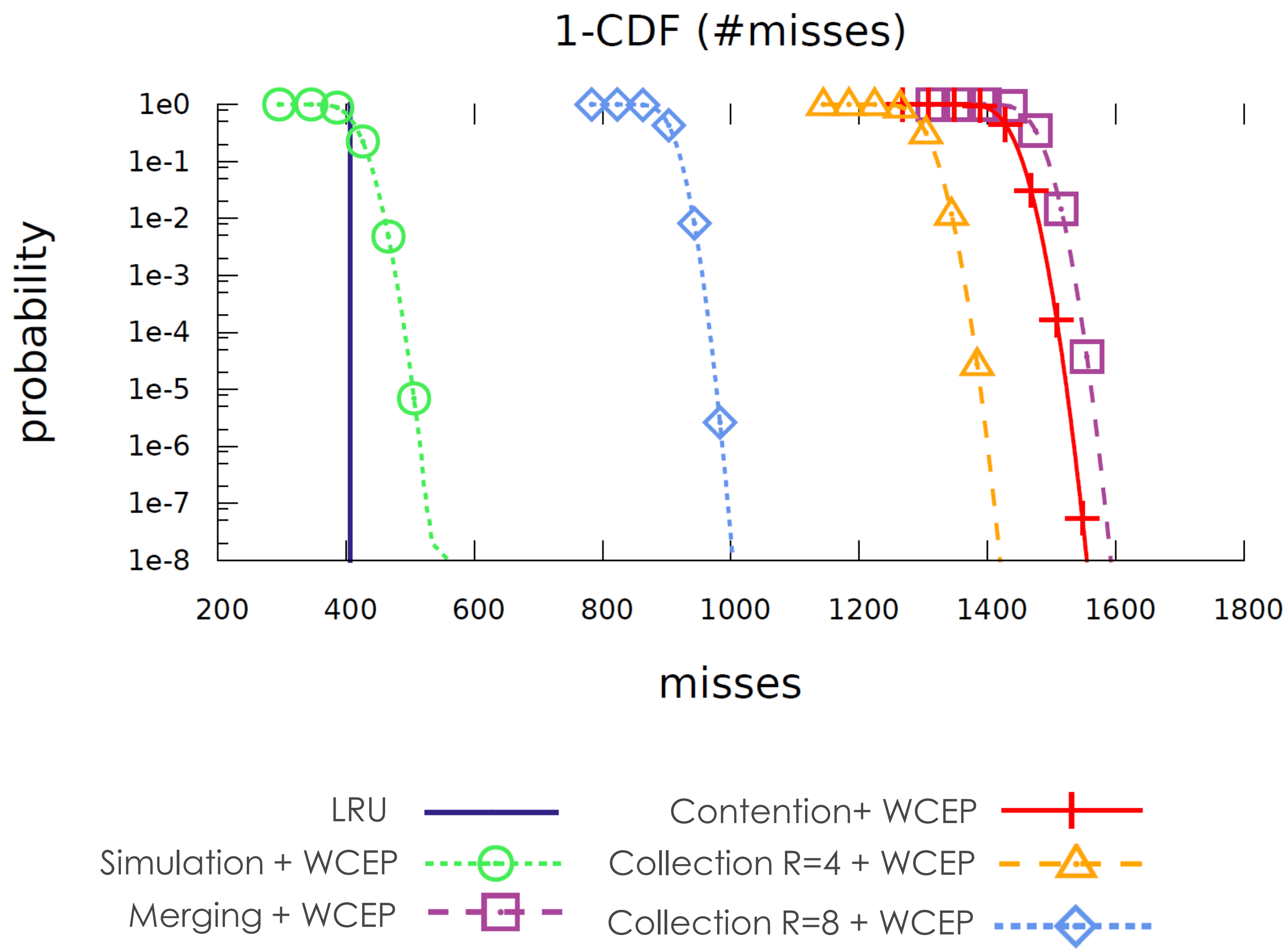
Evaluation

Experimental conditions

- Analysis of misses in instruction cache
  - 16-way, fully associative
  - 32B lines
- Excerpt of the TACLeBench suite
  - Focus on interesting results
- Compared methods:
  - Simulation: distribution over 10^8 runs
  - Merging: synthetic path upper-bound based on reuse-distance
  - Contention
  - Collection: collection approach with R focussed blocks
  - LRU: deterministic LRU cache analysis

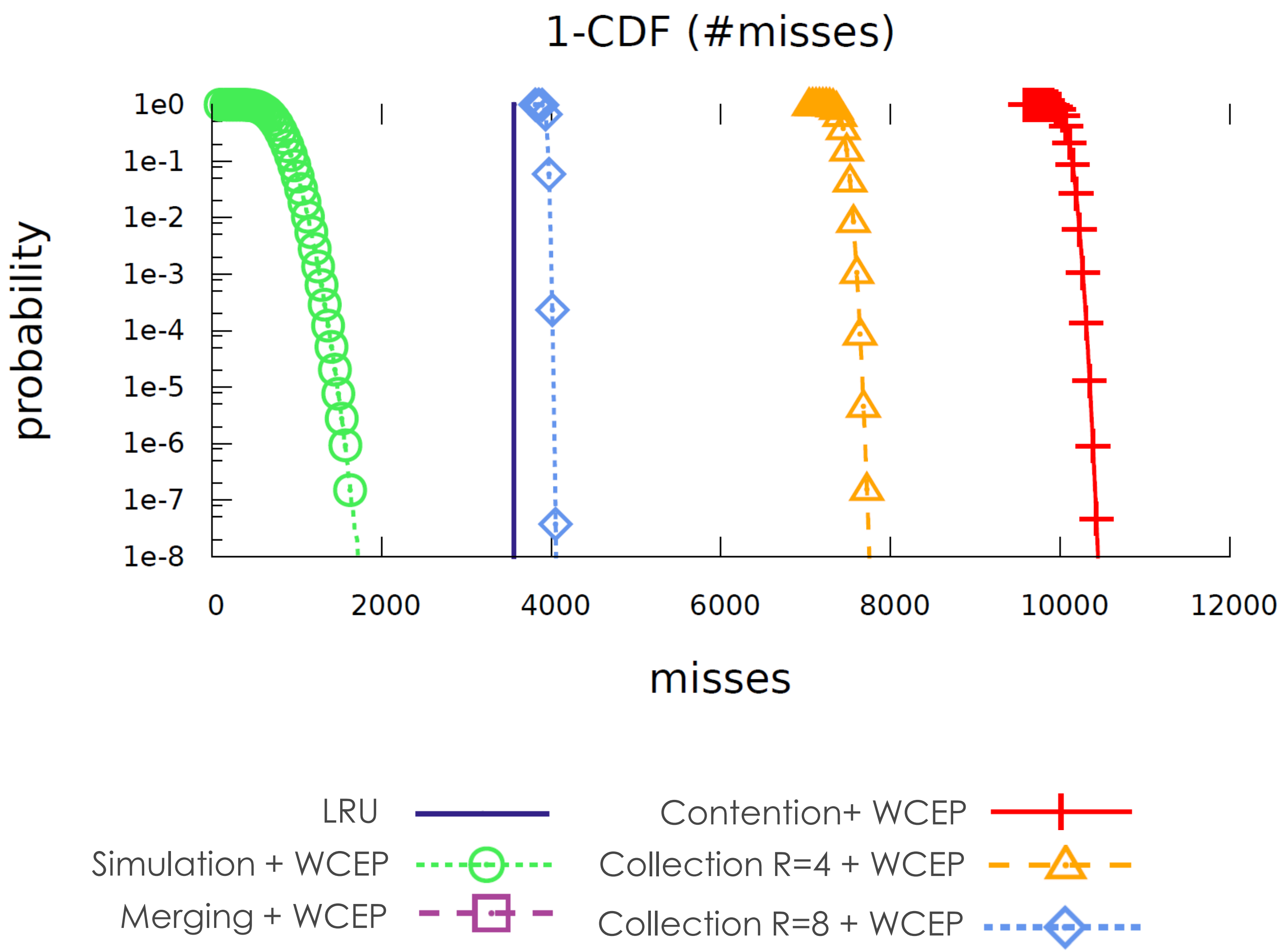
Evaluation

Results – ud, 3K accesses



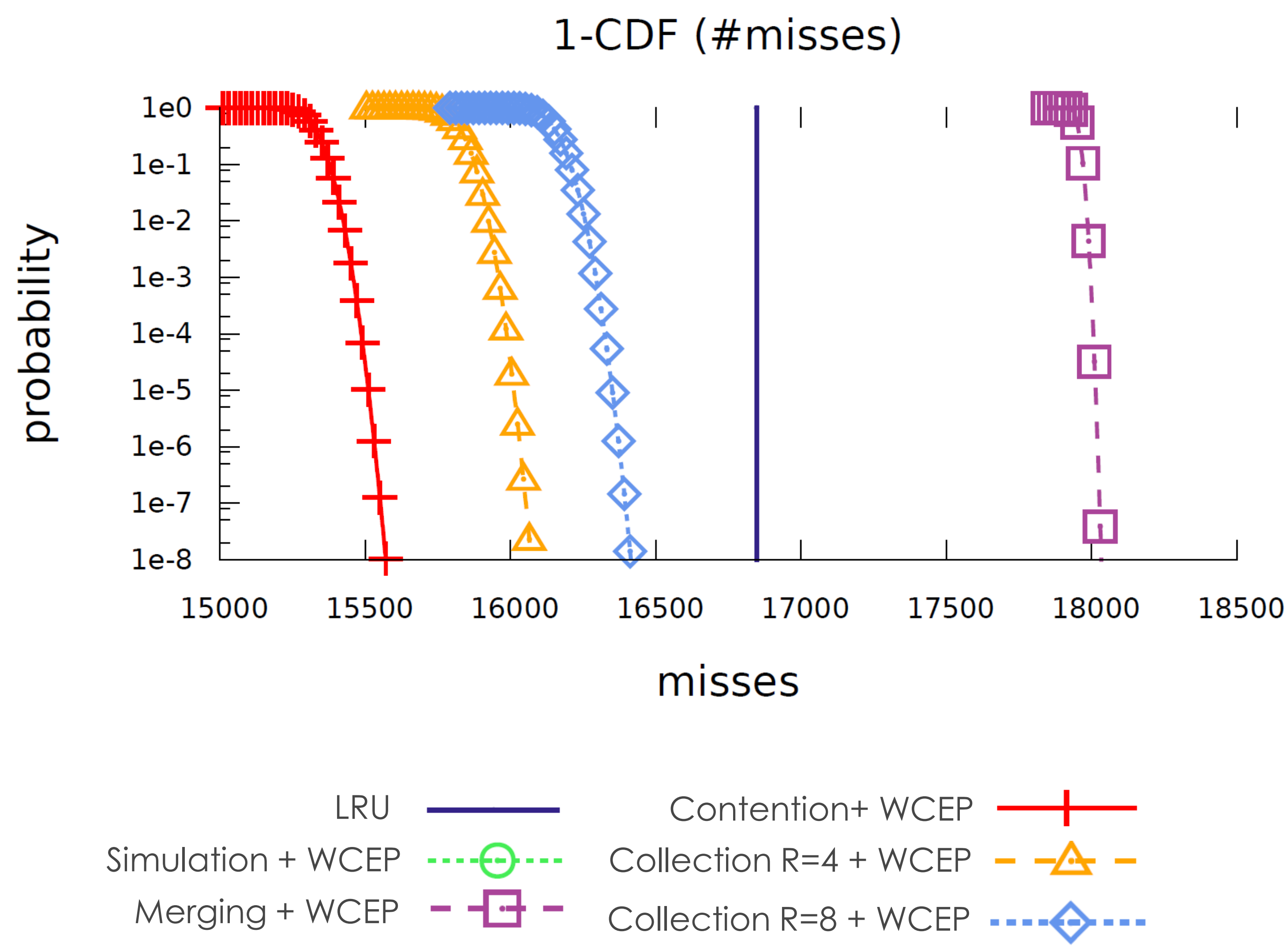
Evaluation

Results – compress, 31K accesses



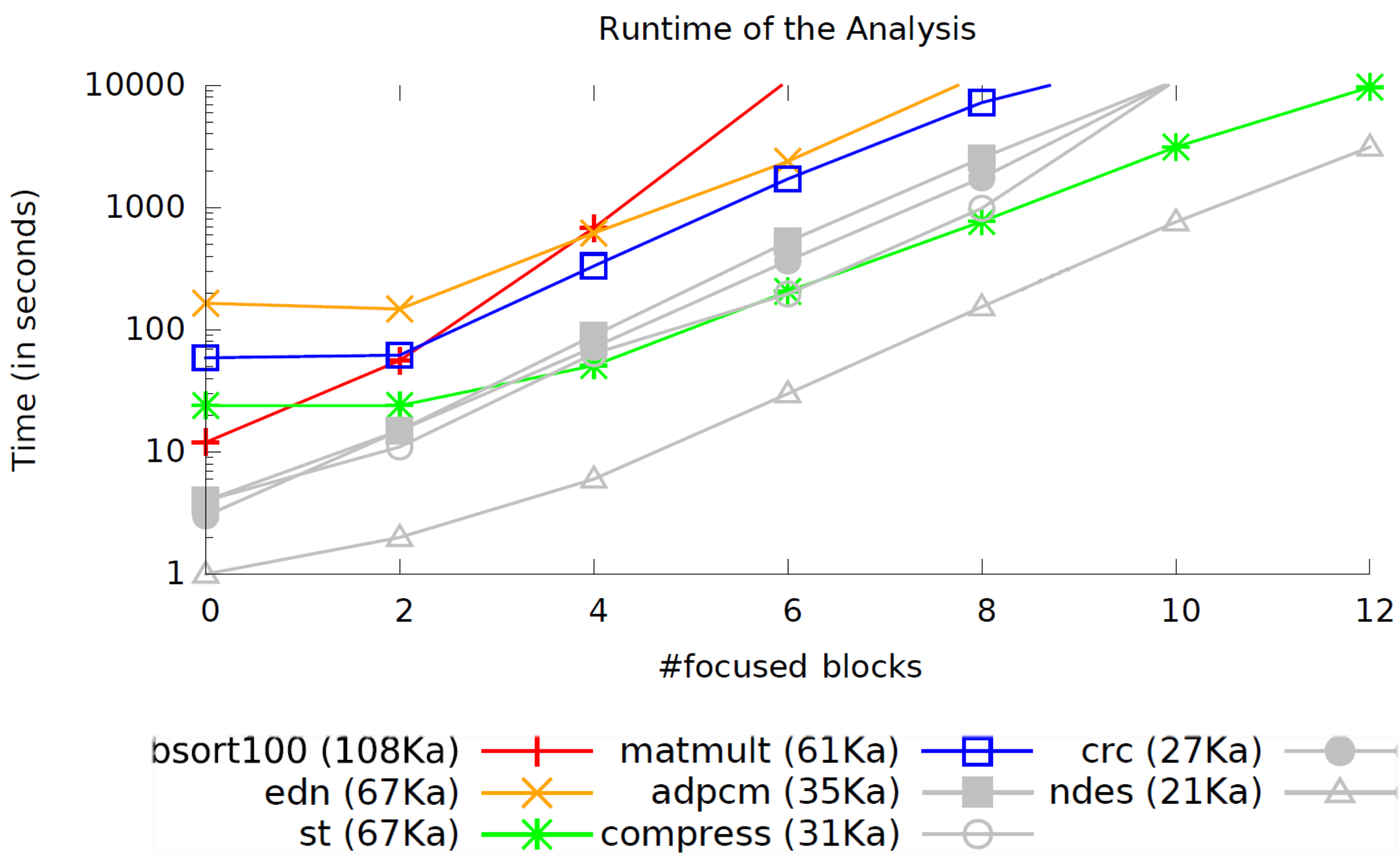
Evaluation

Results – fft 18K accesses



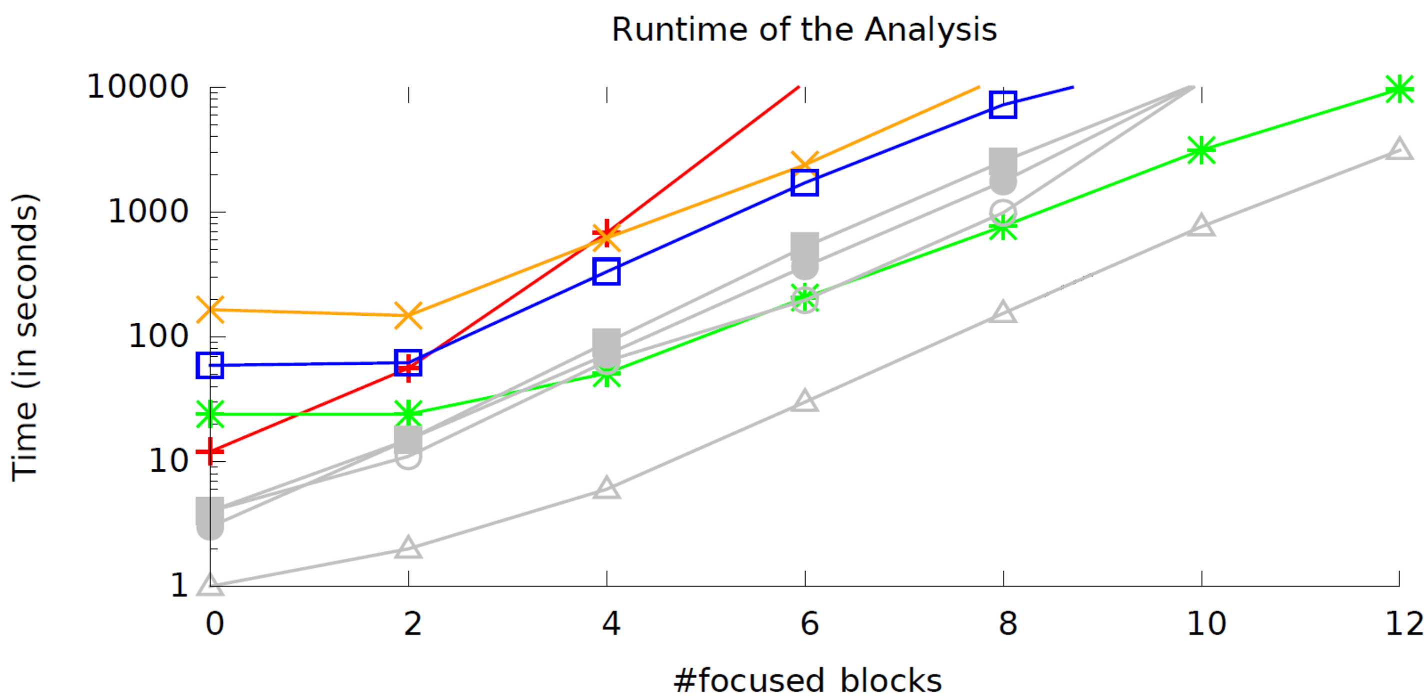
Evaluation

Complexity



Evaluation

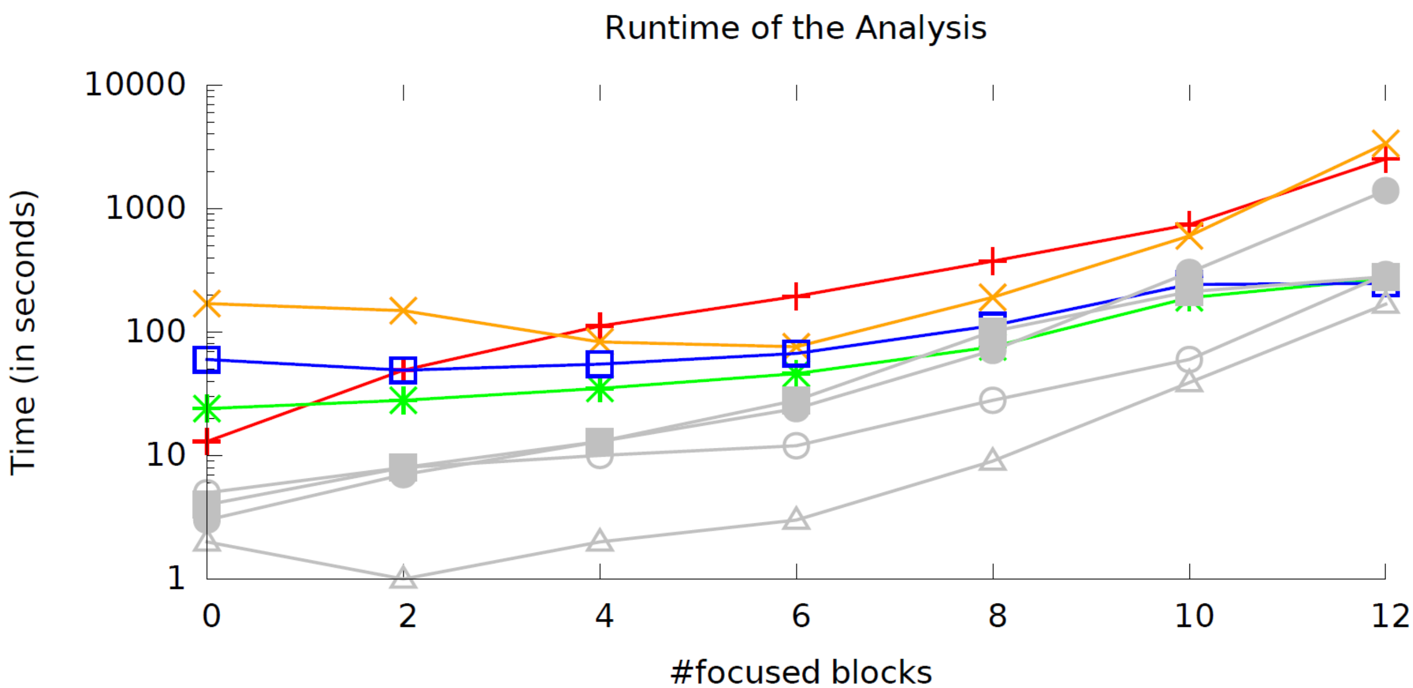
Complexity



- **Complexity**: $O(|S| \times m \times \log(m))$
  - $\boldsymbol{m}$: number of accesses in the program
  - $|\boldsymbol{S}|$: number of possible cache states, $R \leq associativity \Rightarrow 2^R = |S|$
  - **R**: number of focused blocks

Evaluation

Complexity – Control flow partitioning



- Reduce complexity through control-flow partitioning
  - Split the CFG in independent chunks of 1000 Misses
  - B. Pasdeloup, "Static probabilistic timing analysis of worst-case execution time for random replacement caches," INRIA, Tech. Rep., 2014

Conclusions and perspectives

Definition of a multipath approach to SPTA:
- Extend collection approaches
- Extend contention approaches
- Orthogonal to SPTA optimisation approaches

Identification and removal of non pWCET-relevant paths:
- Based on simple heuristics
- Reduced complexity and pessimism

Improve conservation of information on join

Identify additional cases for path redundancy

Backup

- Memory hierarchies: the quick version RTSOPS'14
- Mostly inclusive Memory hierarchies: the anomalies and beyond
- Exclusive Memory hierarchies: Pushing things around

- Improving on the join function: Salvaging capacity
- Benefits of WCEP: TODO
- Impact of CFG-partitioning on precision: TODO

Memory hierarchies

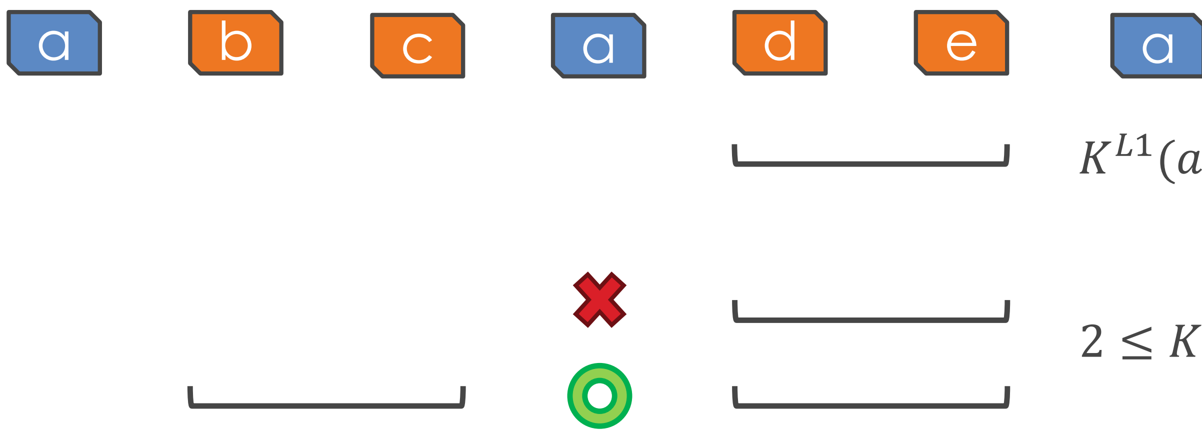
Impact on SPTA

- Hierarchies induce additional dependencies on different levels.
  - Hinder the definition of sound hit probabilities.
  - Hinder the sound combination of hit probabilities.
- Assumptions for contention on single caches do not hold.
  - No model of the different hierarchy policies.
- Increases the complexity of the collection approaches.
  - Evictions on multiple levels multiply the number of states.

Memory hierarchies

Mostly-inclusive policy - SPTA

- Compute the reuse distance from the guaranteed insertion in cache.
  - No guarantee on misses with randomised caches.
  - The requested block is in the L1.



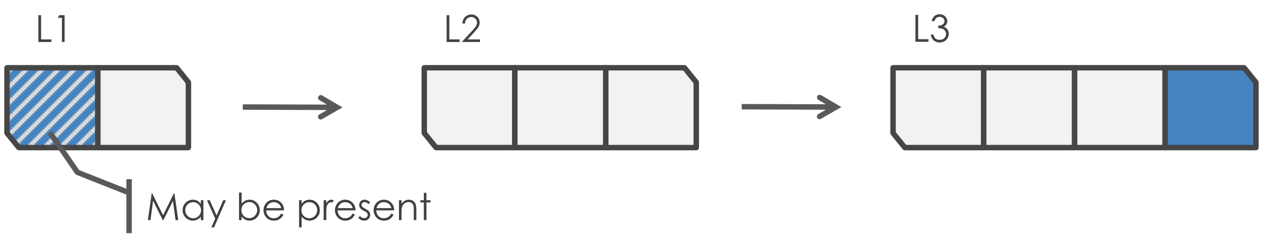
$K^{L1}(a) = 2$

$2 \leq K^{L2}(a) \leq 4$

Memory hierarchies

Mostly-inclusive policy - SPTA

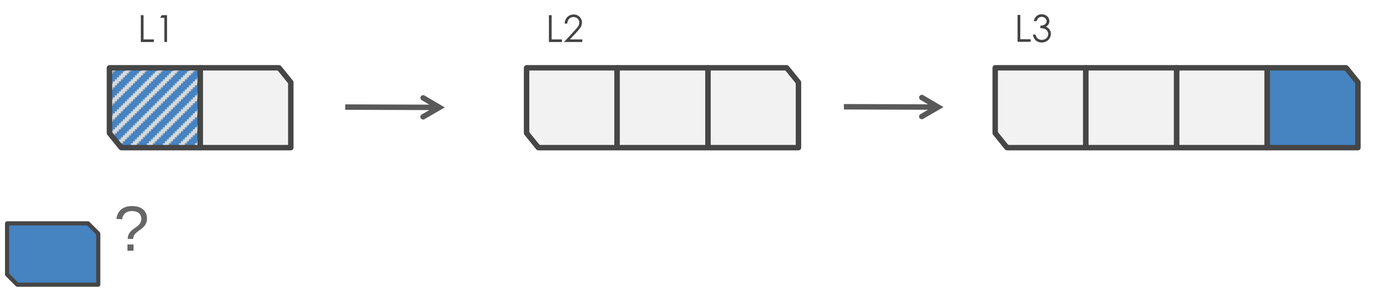
- A miss is not the worst-case contents-wise.
  - Assuming an insertion occurs might result in lower latencies later.
  - Discrepancy with temporal worst-case.



Memory hierarchies

Mostly-inclusive policy - SPTA

- A miss is not the worst-case contents-wise.
  - Assuming an insertion occurs might result in lower latencies later.
  - Discrepancy with temporal worst-case.

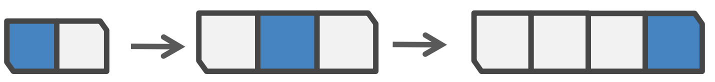


Memory hierarchies

Mostly-inclusive policy - SPTA

- A miss is not the worst-case contents-wise.
  - Assuming an insertion occurs might result in lower latencies later.
  - Discrepancy with temporal worst-case.

L1 Miss

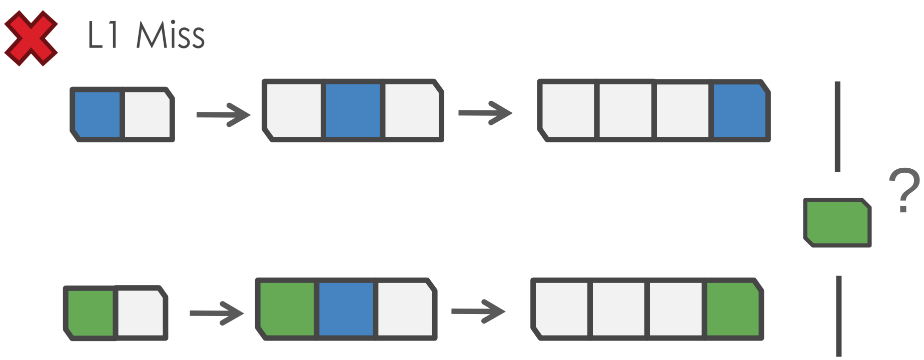


Memory hierarchies

Mostly-inclusive policy - SPTA

RTSOPS'14

- A miss is not the worst-case contents-wise.
  - Assuming an insertion occurs might result in lower latencies later.
  - Discrepancy with temporal worst-case.

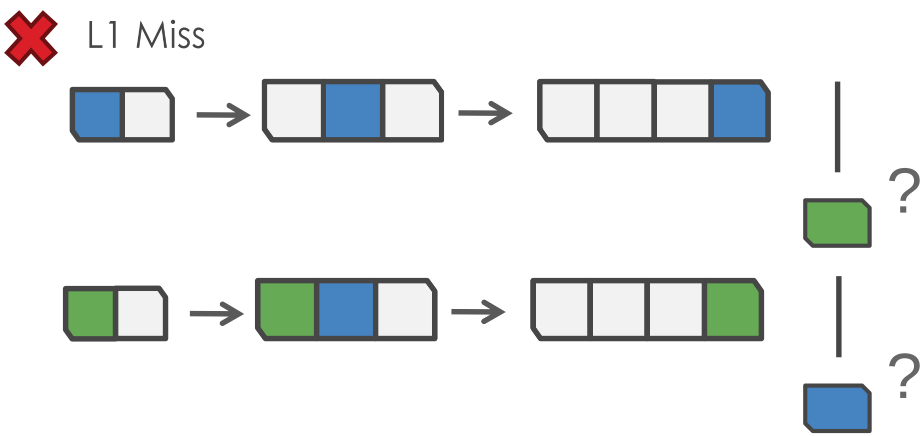


Memory hierarchies

Mostly-inclusive policy - SPTA

RTSOPS'14

- A miss is not the worst-case contents-wise.
  - Assuming an insertion occurs might result in lower latencies later.
  - Discrepancy with temporal worst-case.

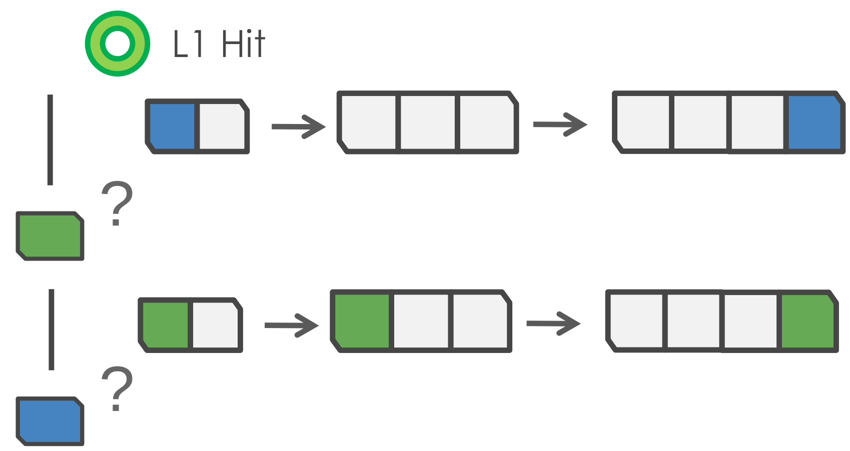


$L_{L3} + L_{Mem} + L_{L2}$

Memory hierarchies

Mostly-inclusive policy - SPTA

- A miss is not the worst-case contents-wise.
  - Assuming an insertion occurs might result in lower latencies later.
  - Discrepancy with temporal worst-case.



$$L_{L1} + L_{Mem} + L_{Mem}$$

Memory hierarchies

Mostly-inclusive policy - SPTA

RTSOPS'14

- A miss is not the worst-case contents-wise.
  - Assuming an insertion occurs might result in lower latencies later.
  - Discrepancy with temporal worst-case.

L1 Miss	$\geq$	L1 Hit
$L_{L3} + L_{Mem} + L_{L2}$	$\leq$	$L_{L1} + L_{Mem} + L_{Mem}$

Memory hierarchies

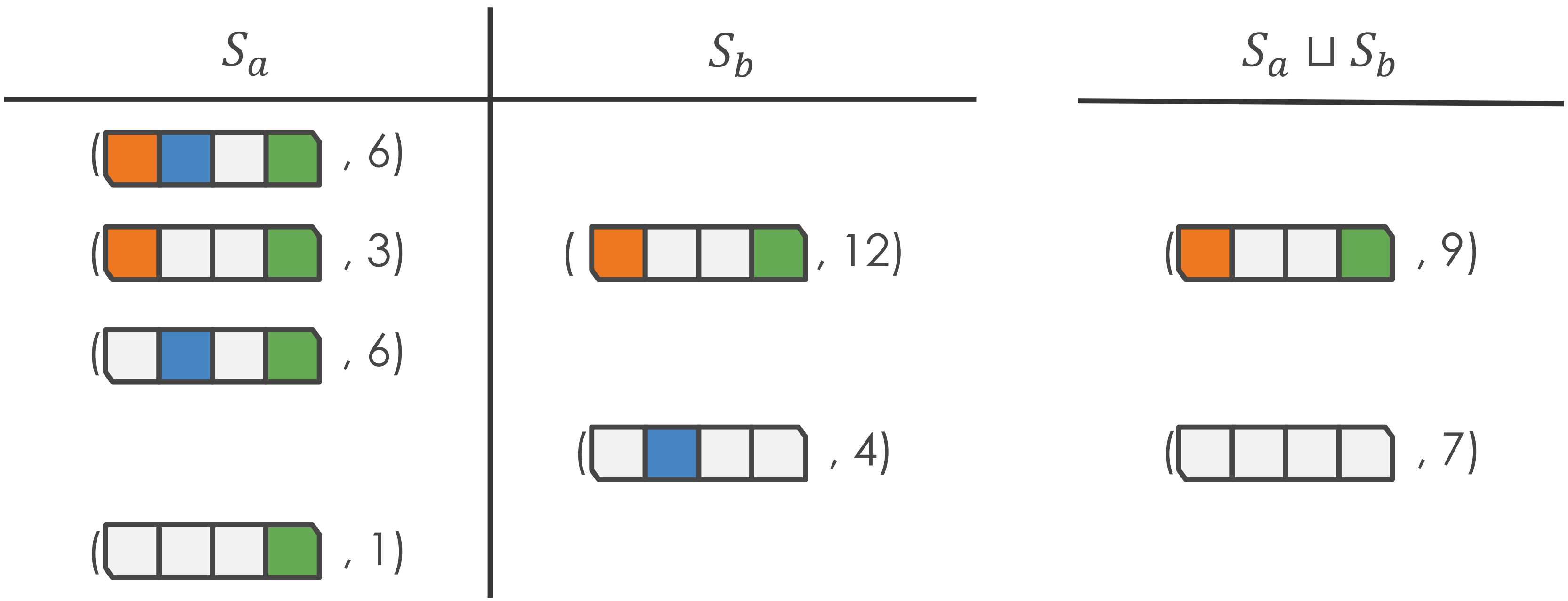
Exclusive policy - Properties

RTSOPS'14

- Miss on the L1 contribute to the reuse distance of all levels.
  - Hits beyond the L1 trigger invalidations.
- Insertion on L occur on eviction from L-1.
  - Insertions on L do not match the sequence of accesses.
  - No guarantee on evictions from L-1 with randomised caches.

Multipath analysis

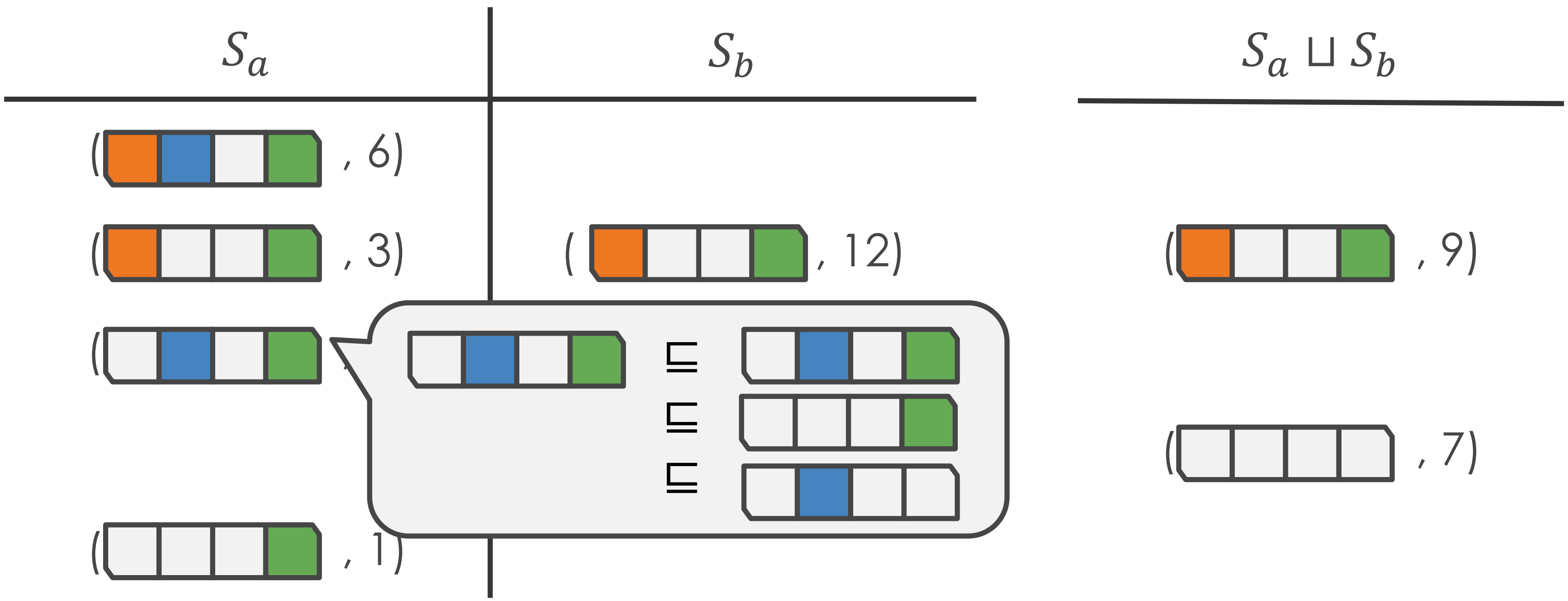
Improving the join function



- Keep only common blocks and contents
  - Occurrence bounded by lowest denominator
  - Maximise merged distributions (Omitted)
  - Merge unmatched states into included states

Multipath analysis

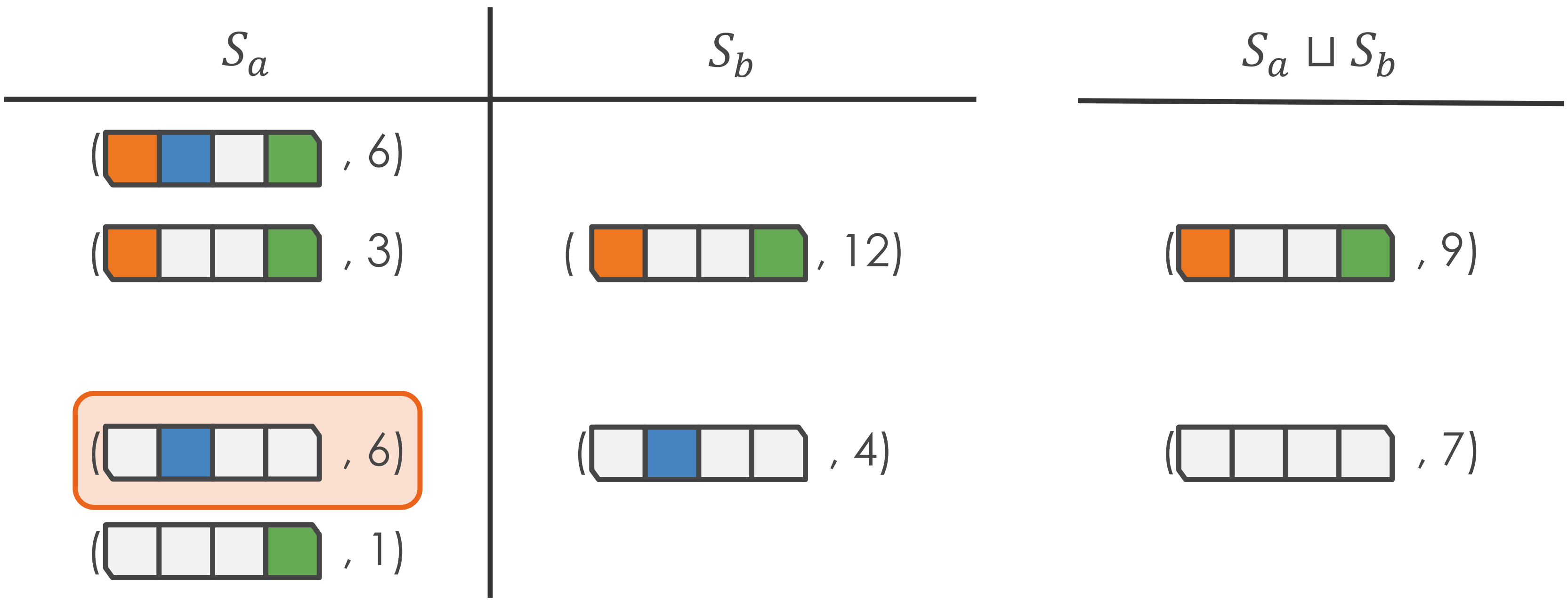
Improving the join function



- Keep only common blocks and contents
  - Occurrence bounded by lowest denominator
  - Maximise merged distributions (Omitted)
  - Merge unmatched states into included states

Multipath analysis

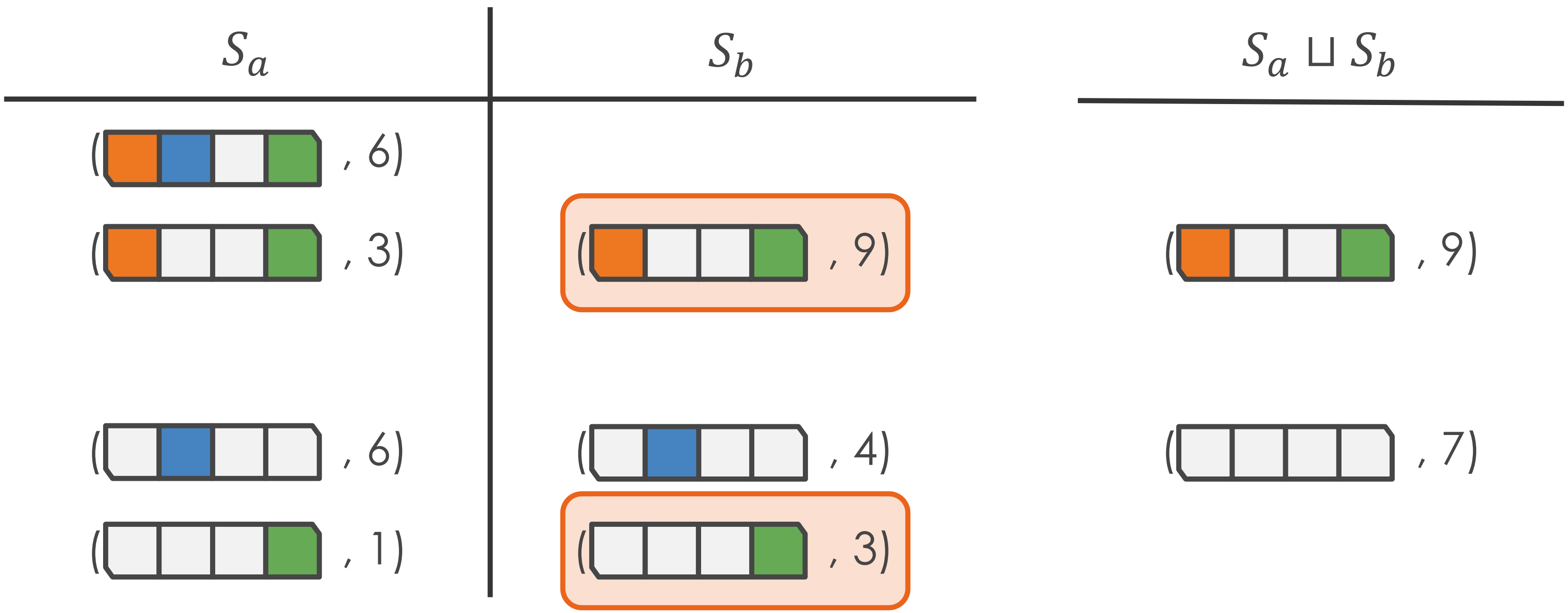
Improving the join function



- Keep only common blocks and contents
  - Occurrence bounded by lowest denominator
  - Maximise merged distributions (Omitted)
  - Merge unmatched states into included states

Multipath analysis

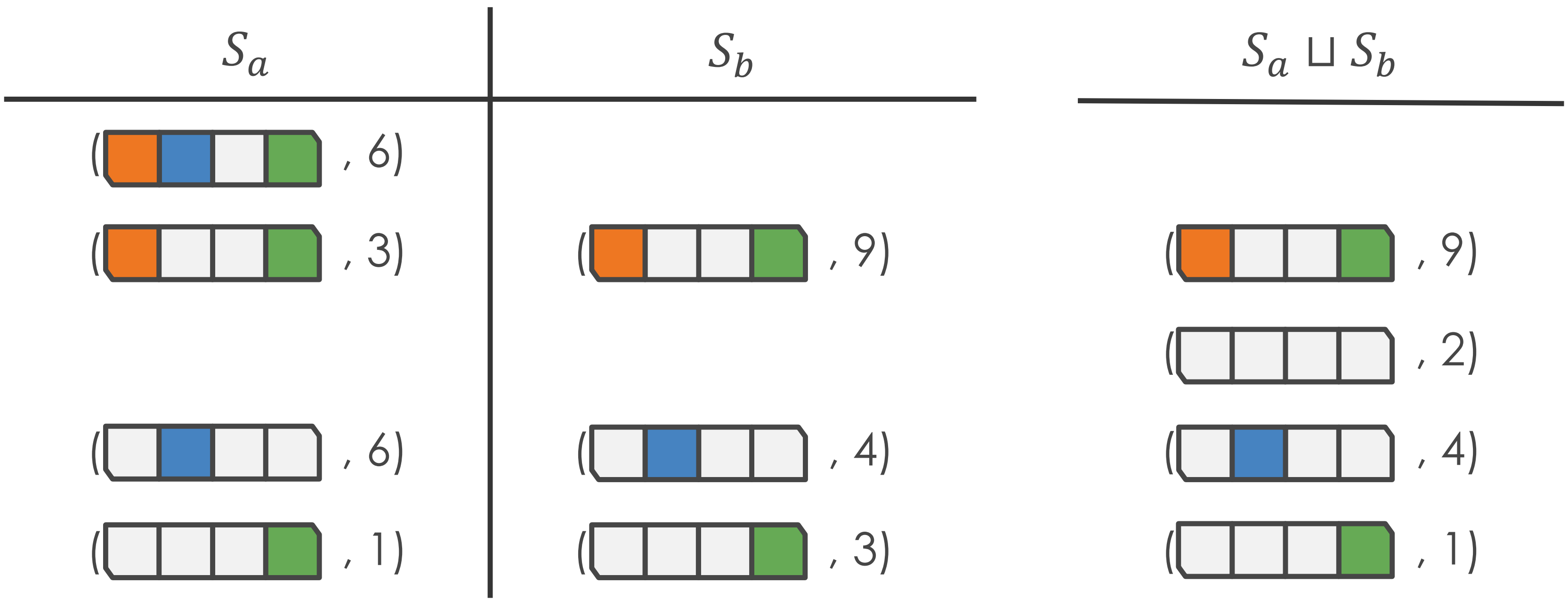
Improving the join function



- Keep only common blocks and contents
  - Occurrence bounded by lowest denominator
  - Maximise merged distributions (Omitted)
  - Merge unmatched states into included states

Multipath analysis

Improving the join function



- Keep only common blocks and contents
  - Occurrence bounded by lowest denominator
  - Maximise merged distributions (Omitted)
  - Merge unmatched states into included states