Design and Analysis of Real-Time Systems
Caches in WCET Analysis

Jan Reineke

Department of Computer Science
Saarland University
Saarbrücken, Germany

Advanced Lecture, Summer 2013

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Outline

1 Caches

2 Cache Analysis for Least-Recently-Used

3 Beyond Least-Recently-Used
- Predictability Metrics
- Relative Competitiveness
- Sensitivity – Caches and Measurement-Based Timing Analysis

4 Summary

Outline

1 Caches

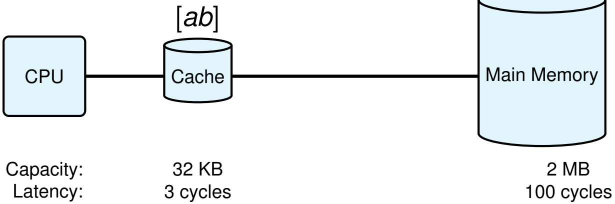
2 Cache Analysis for Least-Recently-Used

3 Beyond Least-Recently-Used
- Predictability Metrics
- Relative Competitiveness
- Sensitivity – Caches and Measurement-Based Timing Analysis

4 Summary

Caches

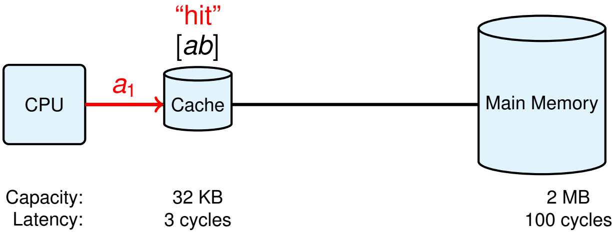
- How they work:
  - dynamically
  - managed by replacement policy



- Why they work: *principle of locality*
  - spatial
  - temporal

Caches

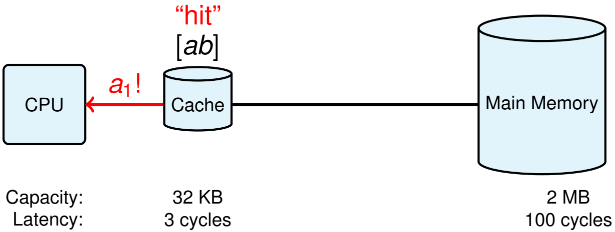
- How they work:
  - dynamically
  - managed by replacement policy



- Why they work: *principle of locality*
  - spatial
  - temporal

Caches

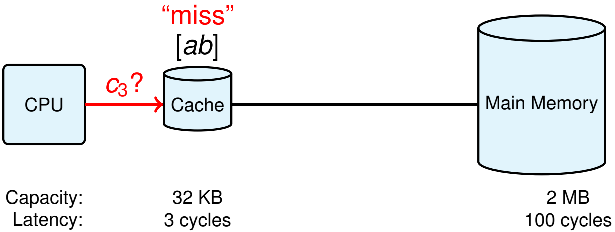
- How they work:
  - dynamically
  - managed by replacement policy



- Why they work: *principle of locality*
  - spatial
  - temporal

Caches

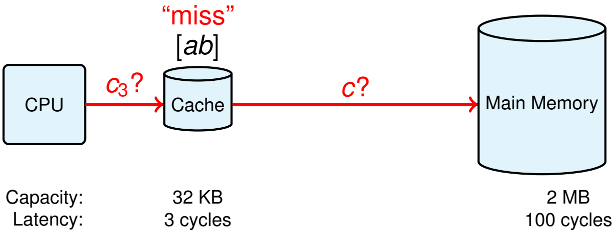
- How they work:
  - dynamically
  - managed by replacement policy



- Why they work: *principle of locality*
  - spatial
  - temporal

Caches

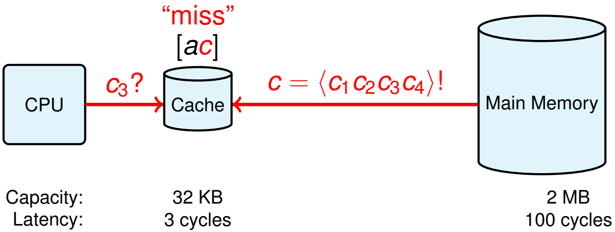
- How they work:
  - dynamically
  - managed by replacement policy



- Why they work: *principle of locality*
  - spatial
  - temporal

Caches

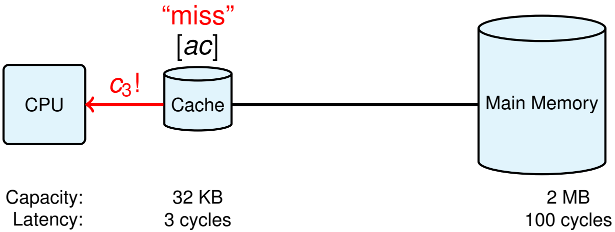
- How they work:
  - dynamically
  - managed by replacement policy



- Why they work: *principle of locality*
  - spatial
  - temporal

Caches

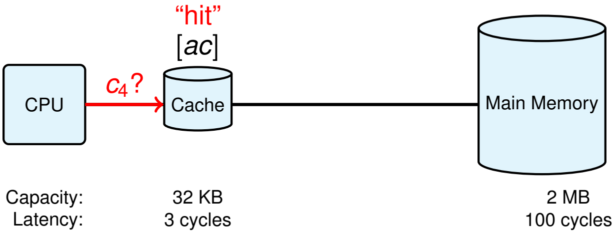
- How they work:
  - dynamically
  - managed by replacement policy



- Why they work: *principle of locality*
  - spatial
  - temporal

Caches

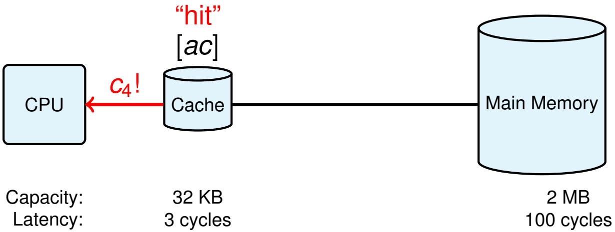
- How they work:
  - dynamically
  - managed by replacement policy



- Why they work: *principle of locality*
  - spatial
  - temporal

Caches

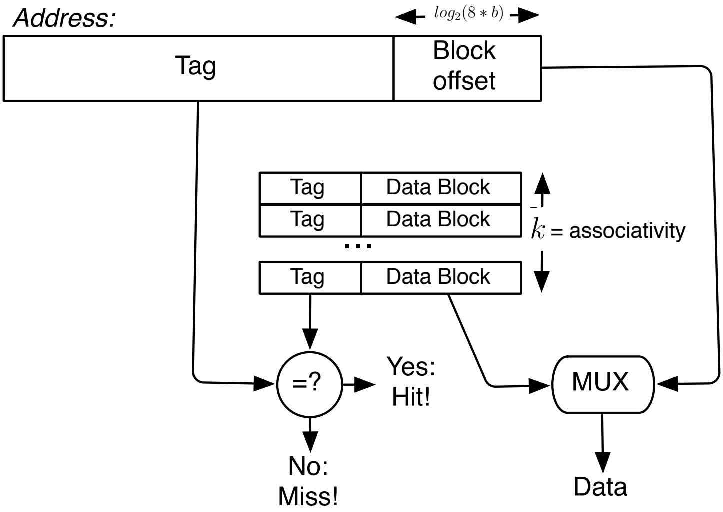
- How they work:
  - dynamically
  - managed by replacement policy



- Why they work: *principle of locality*
  - spatial
  - temporal

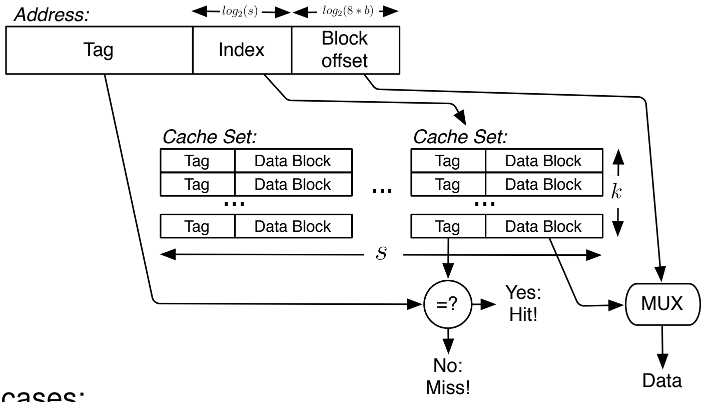
Fully-Associative Caches

SAARLAND UNIVERSITY
COMPUTER SCIENCE



Set-Associative Caches

SAARLAND UNIVERSITY
COMPUTER SCIENCE



Special cases:

- direct-mapped cache: only one line per cache set
- fully-associative cache: only one cache set

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 and POWERPC 75X
- Most Recently Used (MRU) as described in literature

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

Outline

SAARLAND UNIVERSITY
COMPUTER SCIENCE

1 Caches

2 Cache Analysis for Least-Recently-Used

3 Beyond Least-Recently-Used
- Predictability Metrics
- Relative Competitiveness
- Sensitivity – Caches and Measurement-Based Timing Analysis

4 Summary

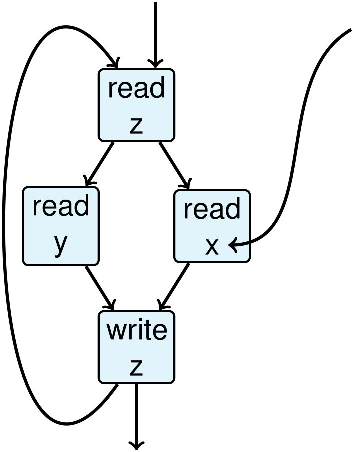
Cache Analysis

Two types of cache analyses:

1. Local guarantees: classification of individual accesses
   - May-Analysis $\longrightarrow$ Overapproximates cache contents
   - Must-Analysis $\longrightarrow$ Underapproximates cache contents
2. Global guarantees: bounds on cache hits/misses

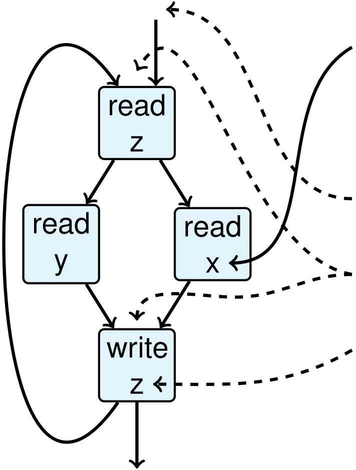
- Cache analyses almost exclusively for LRU
- In practice: FIFO, PLRU, ...

Challenges for Cache Analysis



Always a cache hit/always a miss?

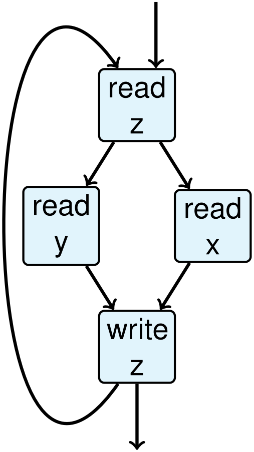
Challenges for Cache Analysis



Always a cache hit/always a miss?

1. Initial cache contents unknown.
2. Different paths lead to these points.
3. Cannot resolve address of *z*.

Deriving Invariants about Cache States using Abstract Interpretation

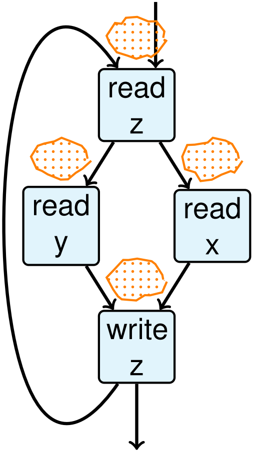


Collecting Semantics =
set of states at each program point that any execution may encounter there

Two approximations:

	Collecting Semantics	uncomputable
$\subseteq$	Cache Semantics	computable
$\subseteq$	γ(Abstract Cache Sem.)	efficiently computable

Deriving Invariants about Cache States using Abstract Interpretation

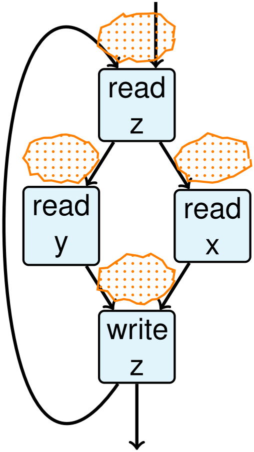


Collecting Semantics =
set of states at each program point that any execution may encounter there

Two approximations:

	Collecting Semantics	uncomputable
$\subseteq$	Cache Semantics	computable
$\subseteq$	γ(Abstract Cache Sem.)	efficiently computable

Deriving Invariants about Cache States using Abstract Interpretation

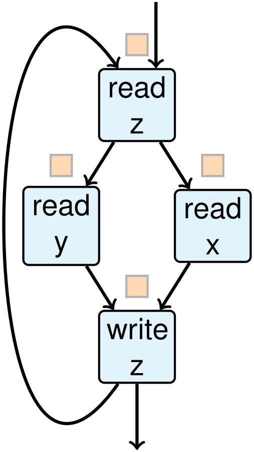


Collecting Semantics =
set of states at each program point that any execution may encounter there

Two approximations:

	Collecting Semantics	uncomputable
$\subseteq$	Cache Semantics	computable
$\subseteq$	γ(Abstract Cache Sem.)	efficiently computable

Deriving Invariants about Cache States using Abstract Interpretation

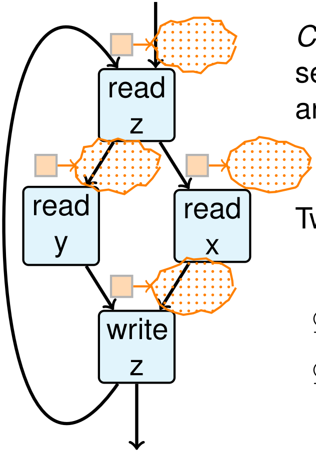


Collecting Semantics =
set of states at each program point that any execution may encounter there

Two approximations:

	Collecting Semantics	uncomputable
$\subseteq$	Cache Semantics	computable
$\subseteq$	γ(Abstract Cache Sem.)	efficiently computable

Deriving Invariants about Cache States using Abstract Interpretation



Collecting Semantics =
set of states at each program point that any execution may encounter there

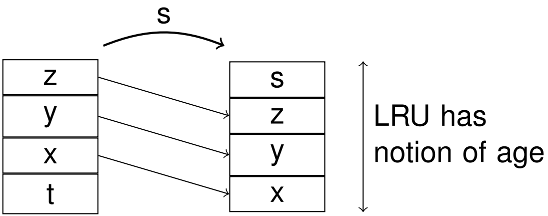
Two approximations:

	Collecting Semantics	uncomputable
$\subseteq$	Cache Semantics	computable
$\subseteq$	γ(Abstract Cache Sem.)	efficiently computable

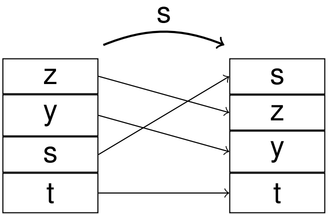
Least-Recently-Used (LRU): Concrete Behavior

SAARLAND UNIVERSITY
COMPUTER SCIENCE

“Cache Miss”:



“Cache Hit”:



LRU: Must-Analysis: Abstract Domain

- Used to predict *cache hits*.
- Maintains *upper bounds on ages* of memory blocks.
- Upper bound $\leq$ associativity $\longrightarrow$ memory block definitely cached.

Example

Abstract state:

{x}	age 0
{}	
{s,t}	
{}	age 3

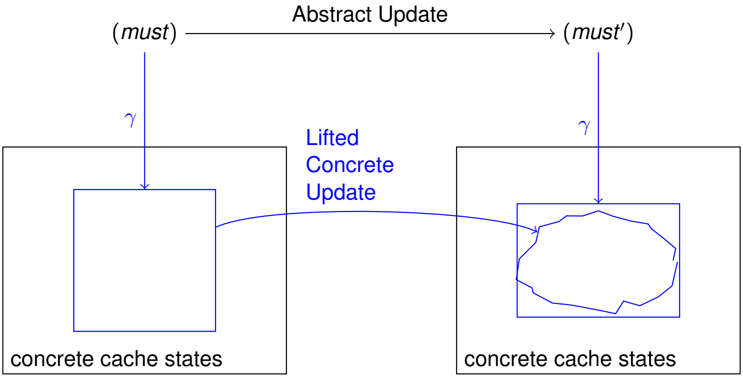
...and its interpretation:

Describes the set of all concrete cache states in which x, s, and t occur,

- x with an age of 0,
- s and t with an age not older than 2.

$$\gamma([\{x\}, \{\}, \{s, t\}, \{\}]) = \{[x, s, t, a], [x, t, s, a], [x, s, t, b], \ldots\}$$

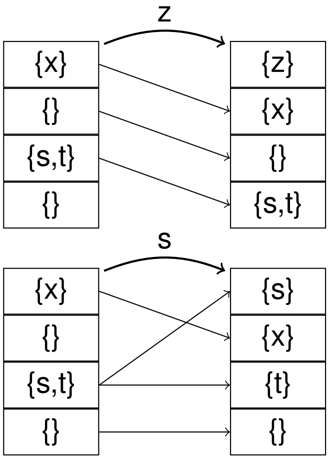
Sound Update – Local Consistency



LRU: Must-Analysis: Update

“Potential Cache Miss”:

“Definite Cache Hit”:



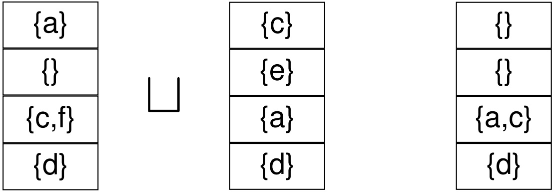
Why does *t* not age in the second case?

LRU: Must-Analysis: Join

Need to combine information where control-flow merges.

Join should be conservative (ensures γ is monotone):

- $\gamma(A) \subseteq \gamma(A \sqcup B)$
- $\gamma(B) \subseteq \gamma(A \sqcup B)$



"Intersection + Maximal Age"

LRU: Must-Analysis: Join

Need to combine information where control-flow merges.

Join should be conservative (ensures γ is monotone):

- $\gamma(A) \subseteq \gamma(A \sqcup B)$
- $\gamma(B) \subseteq \gamma(A \sqcup B)$

{a}		{c}	{}
{}	$\sqcup$	{e}	{}
{c,f}		{a}	{a,c}
{d}		{d}	{d}

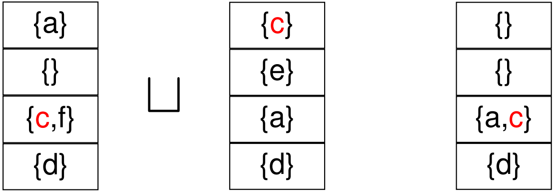
“Intersection + Maximal Age”

LRU: Must-Analysis: Join

Need to combine information where control-flow merges.

Join should be conservative (ensures γ is monotone):

- $\gamma(A) \subseteq \gamma(A \sqcup B)$
- $\gamma(B) \subseteq \gamma(A \sqcup B)$



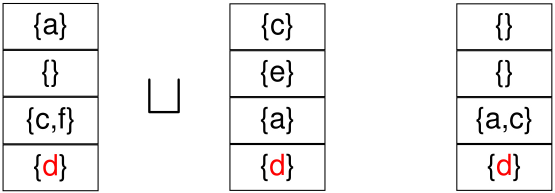
“Intersection + Maximal Age”

LRU: Must-Analysis: Join

Need to combine information where control-flow merges.

Join should be conservative (ensures γ is monotone):

- $\gamma(A) \subseteq \gamma(A \sqcup B)$
- $\gamma(B) \subseteq \gamma(A \sqcup B)$



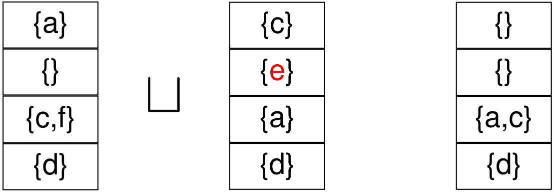
“Intersection + Maximal Age”

LRU: Must-Analysis: Join

Need to combine information where control-flow merges.

Join should be conservative (ensures γ is monotone):

- $\gamma(A) \subseteq \gamma(A \sqcup B)$
- $\gamma(B) \subseteq \gamma(A \sqcup B)$



"Intersection + Maximal Age"

LRU: Must-Analysis: Join

Need to combine information where control-flow merges.

Join should be conservative (ensures γ is monotone):

- $\gamma(A) \subseteq \gamma(A \sqcup B)$
- $\gamma(B) \subseteq \gamma(A \sqcup B)$

{a}		{c}		{}
{}	$\sqcup$	{e}		{}
{c,f}		{a}		{a,c}
{d}		{d}		{d}

"Intersection + Maximal Age"

LRU: Must-Analysis: Join

Need to combine information where control-flow merges.

Join should be conservative (ensures γ is monotone):

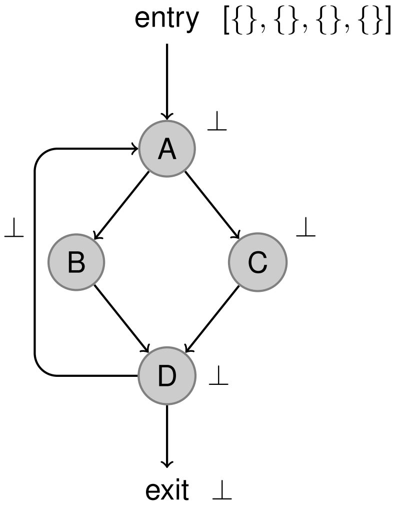
- $\gamma(A) \subseteq \gamma(A \sqcup B)$
- $\gamma(B) \subseteq \gamma(A \sqcup B)$

{a}		{c}	{}
{}	$\sqcup$	{e}	{}
{c,f}		{a}	{a,c}
{d}		{d}	{d}

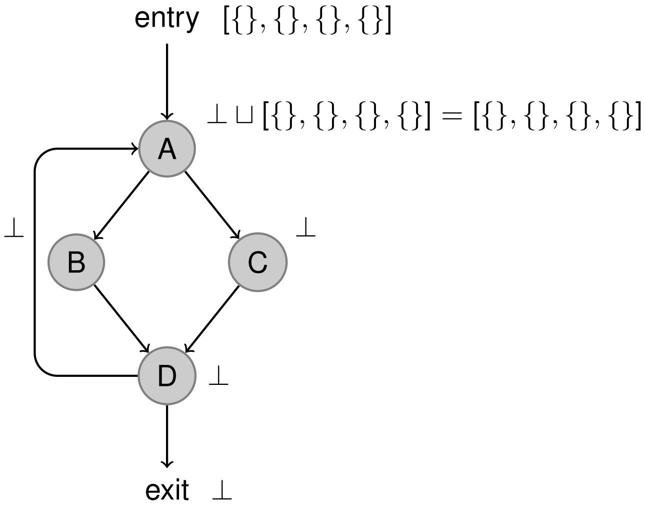
"Intersection + Maximal Age"

How many memory blocks can be in the must-cache?

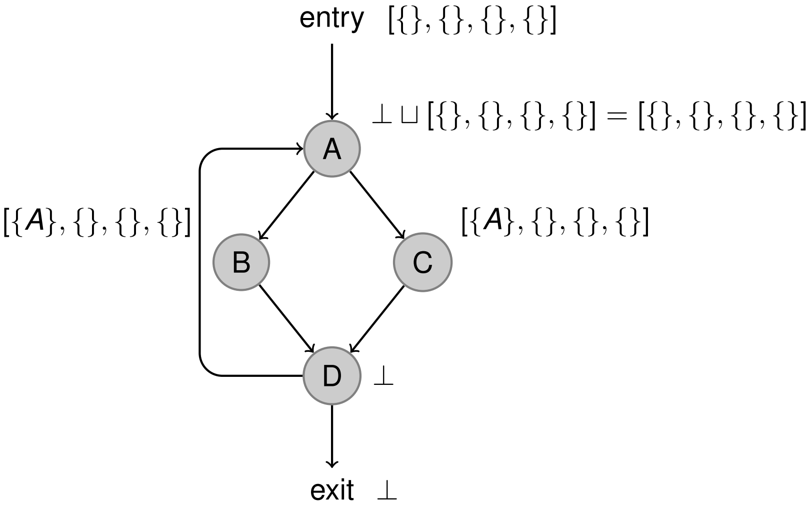
Example: Must-Analysis



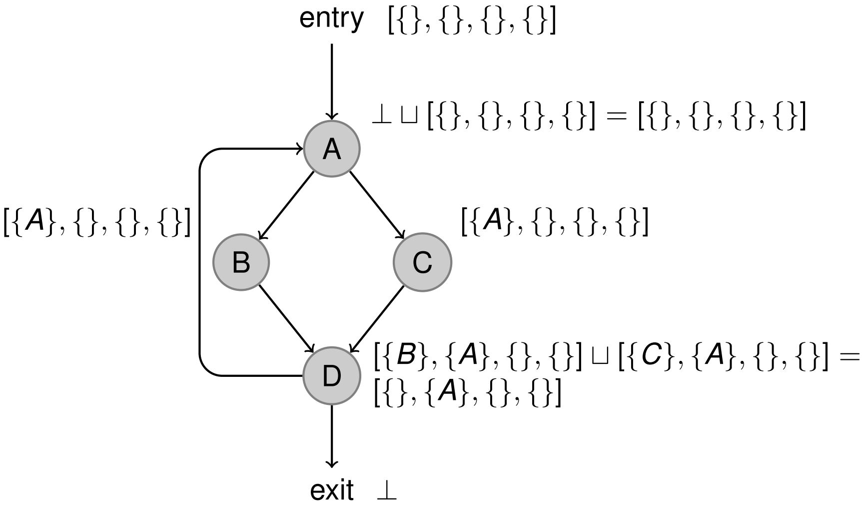
Example: Must-Analysis



Example: Must-Analysis

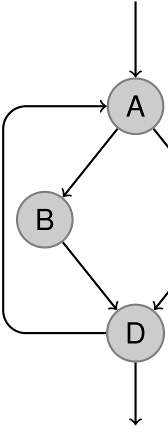


Example: Must-Analysis



Example: Must-Analysis

entry $[\{\}, \{\}, \{\}, \{\}]$



$$[\{D\}, \{\}, \{A\}, \{\}] \sqcup [\{\}, \{\}, \{\}, \{\}] = [\{\}, \{\}, \{\}, \{\}]$$

$[\{A\}, \{\}, \{\}, \{\}]$

$[\{A\}, \{\}, \{\}, \{\}]$

$$[\{B\}, \{A\}, \{\}, \{\}] \sqcup [\{C\}, \{A\}, \{\}, \{\}] = [\{\}, \{A\}, \{\}, \{\}]$$

exit $[\{D\}, \{\}, \{A\}, \{\}]$

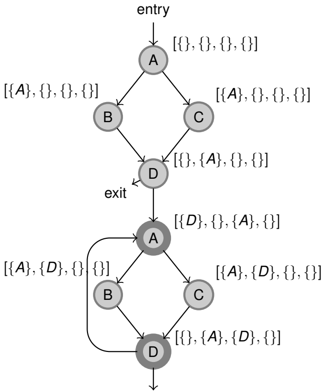
No cache hits can be predicted :-(

Context-Sensitive Analysis/Virtual Loop-Unrolling

- Problem:
  - The first iteration of a loop will always result in cache misses.
  - Similarly for the first execution of a function.
- Solution:
  - Virtually Unroll Loops: Distinguish the first iteration from others
  - Distinguish function calls by calling context.

Virtually unrolling the loop once:

- Accesses to *A* and *D* are provably hits after the first iteration
- Accesses to *B* and *C* can still not be classified. Within each execution of the loop, they may only miss once.
  $\longrightarrow$ Persistence Analysis



LRU: May-Analysis: Abstract Domain

- Used to predict *cache misses*.
- Maintains *lower bounds on ages* of memory blocks.
- Lower bound $\geq$ associativity
  $\longrightarrow$ memory block definitely *not* cached.

Example

Abstract state:

{x,y}	age 0
{}	
{s,t}	
{u}	age 3

. . . and its interpretation:

Describes the set of all concrete cache states in which no memory blocks except *x*, *y*, *s*, *t*, and *u* occur,

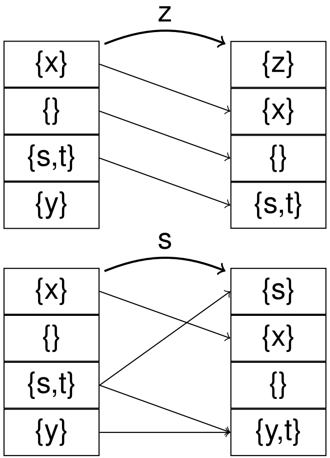
- *x* and *y* with an age of at least 0,
- *s* and *t* with an age of at least 2,
- *u* with an age of at least 3.

$$\gamma([\{x,y\},\{\},\{s,t\},\{u\}]) = \{[x,y,s,t],[y,x,s,t],[x,y,s,u],\ldots\}$$

LRU: May-Analysis: Update

"Definite Cache Miss":

"Potential Cache Hit":



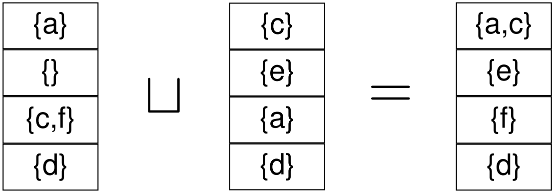
Why does *t* age in the second case?

LRU: May-Analysis: Join

Need to combine information where control-flow merges.

Join should be conservative (ensures γ is monotone):

- $\gamma(A) \subseteq \gamma(A \sqcup B)$
- $\gamma(B) \subseteq \gamma(A \sqcup B)$



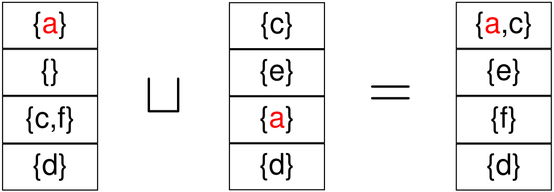
"Union + Minimal Age"

LRU: May-Analysis: Join

Need to combine information where control-flow merges.

Join should be conservative (ensures γ is monotone):

- $\gamma(A) \subseteq \gamma(A \sqcup B)$
- $\gamma(B) \subseteq \gamma(A \sqcup B)$



“Union + Minimal Age”

LRU: May-Analysis: Join

SAARLAND UNIVERSITY
COMPUTER SCIENCE

Need to combine information where control-flow merges.

Join should be conservative (ensures γ is monotone):

- $\gamma(A) \subseteq \gamma(A \sqcup B)$
- $\gamma(B) \subseteq \gamma(A \sqcup B)$

{a}		{c}		{a,c}
{}	$\sqcup$	{e}	$=$	{e}
{c,f}		{a}		{f}
{d}		{d}		{d}

"Union + Minimal Age"

LRU: May-Analysis: Join

Need to combine information where control-flow merges.

Join should be conservative (ensures γ is monotone):

- $\gamma(A) \subseteq \gamma(A \sqcup B)$
- $\gamma(B) \subseteq \gamma(A \sqcup B)$

{a}		{c}		{a,c}
{}	$\sqcup$	{e}	$=$	{e}
{c,f}		{a}		{f}
{d}		{d}		{d}

"Union + Minimal Age"

LRU: May-Analysis: Join

Need to combine information where control-flow merges.

Join should be conservative (ensures γ is monotone):

- $\gamma(A) \subseteq \gamma(A \sqcup B)$
- $\gamma(B) \subseteq \gamma(A \sqcup B)$

{a}		{c}		{a,c}
{}	$\sqcup$	{e}	$=$	{e}
{c,f}		{a}		{f}
{d}		{d}		{d}

"Union + Minimal Age"

LRU: May-Analysis: Join

Need to combine information where control-flow merges.

Join should be conservative (ensures γ is monotone):

- $\gamma(A) \subseteq \gamma(A \sqcup B)$
- $\gamma(B) \subseteq \gamma(A \sqcup B)$

A		B		$A \sqcup B$
{a}		{c}		{a,c}
{}		{e}		{e}
{c,f}	$\sqcup$	{a}	$=$	{f}
{d}		{d}		{d}

"Union + Minimal Age"