Cache analysis

What do we want to compute?

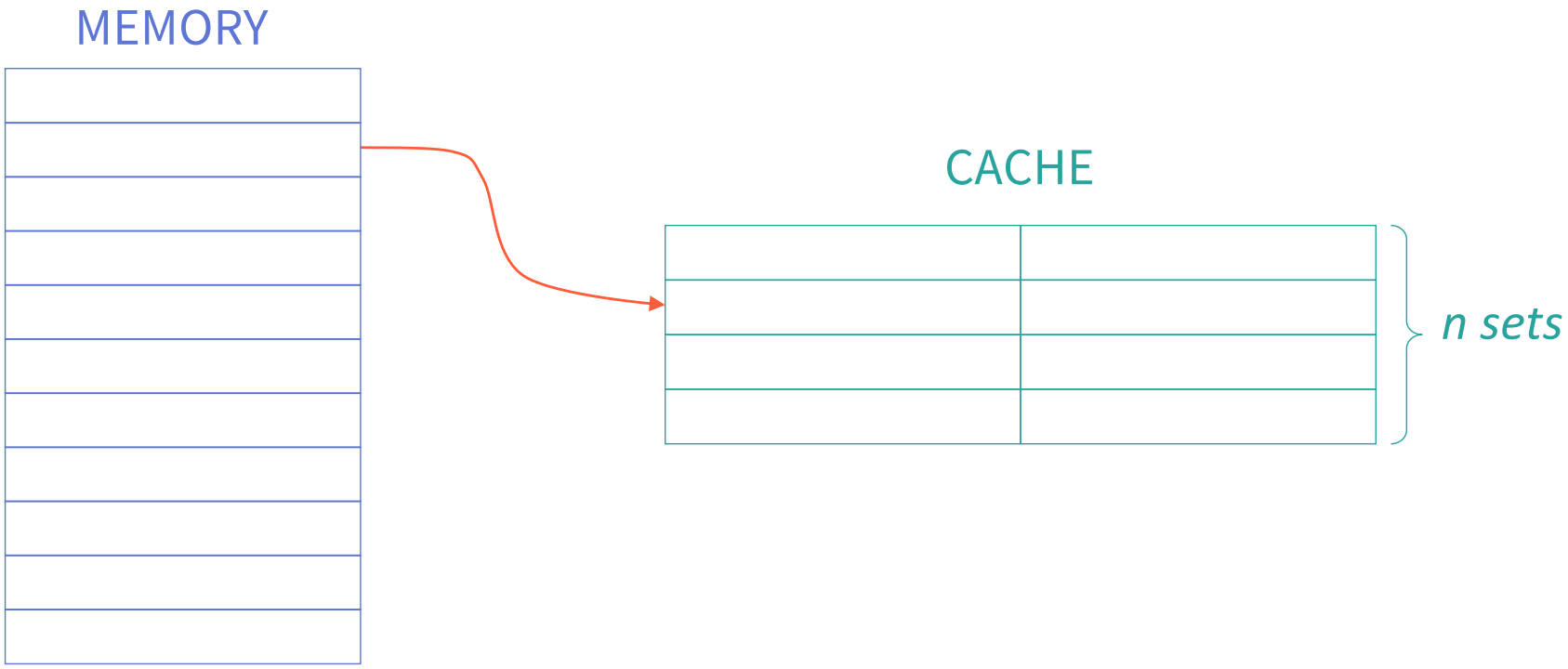
Classify accesses to cache memories

- AlwaysHit, Always Miss → latency is known
- FirstMiss → latency is known
- ? (NotClassified) → latency is unknown

Cache analysis by Abstract Interpretation

Assumptions

- set-associative instruction cache
- with an LRU replacement policy



Cache analysis by Abstract Interpretation

(Concrete) State of a cache set

sequence of accesses: a-b-c-d

contents of the set

a	c	d	b

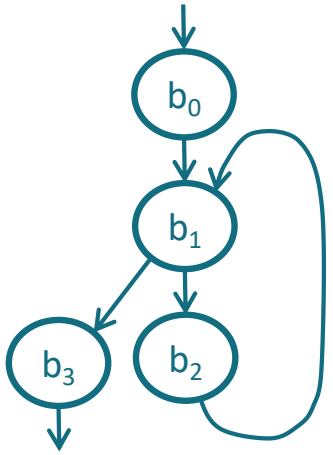
age (LRU)

3	1	0	2

d	0
c	1
b	2
a	3

Abstract interpretation-based analysis

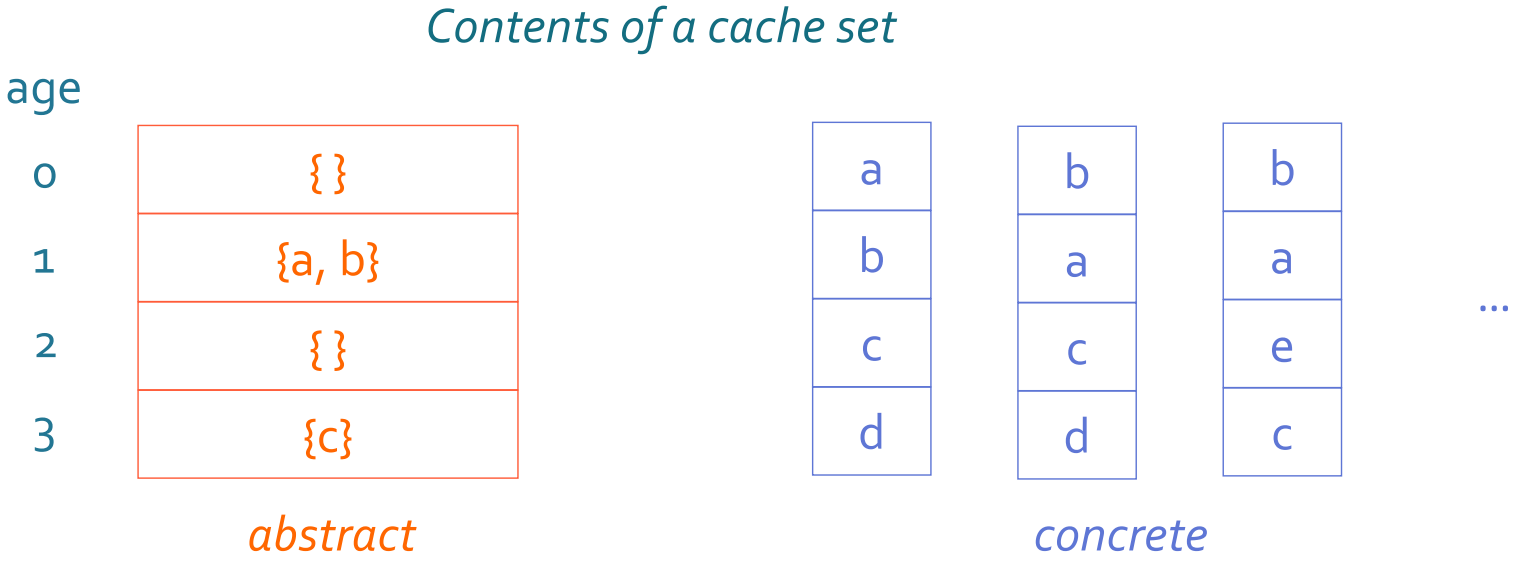
- compute possible cache states at each point in the program
- two analyses:
  - MAY: which memory blocks *may* be in the cache?
  - MUST: … *must* …. ?
- for each analysis:
  - abstract cache states (ACS)
  - update and join functions



Cache analysis by Abstract Interpretation

MUST analysis

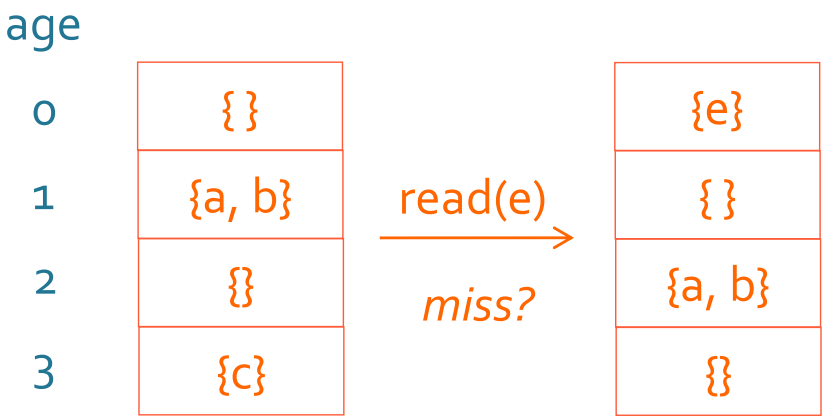
- abstract cache state = upper bounds on block ages

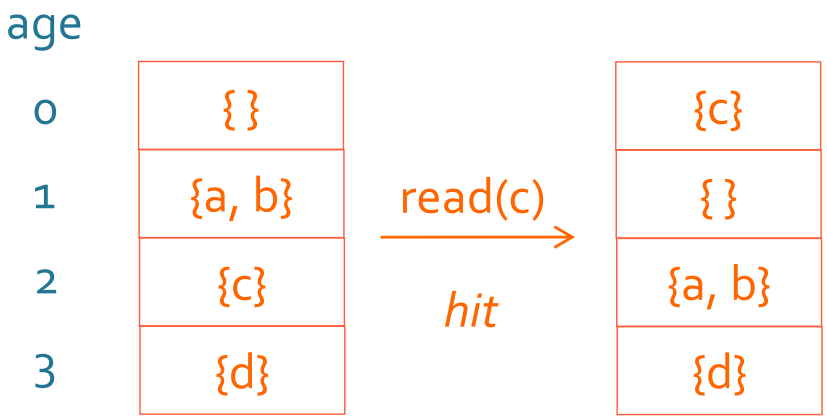


Cache analysis by Abstract Interpretation

MUST analysis

- update() function

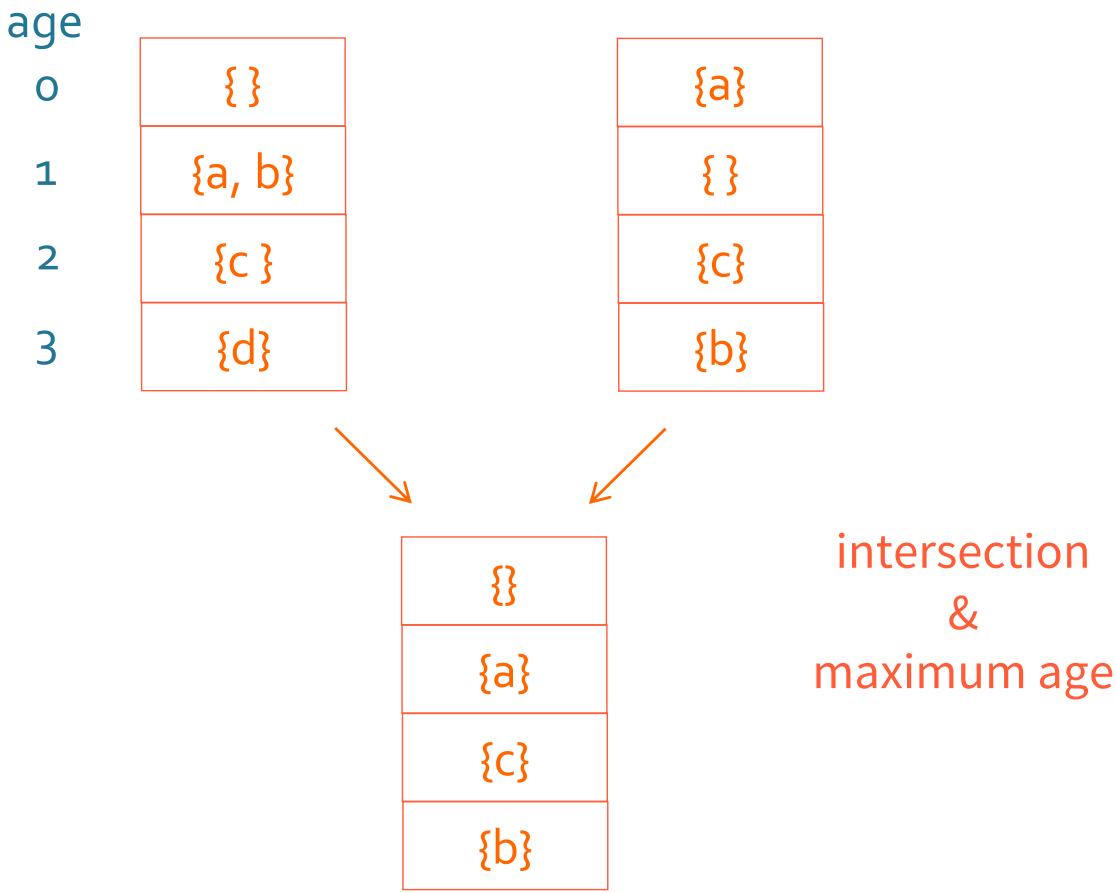




Cache analysis by Abstract Interpretation

MUST analysis

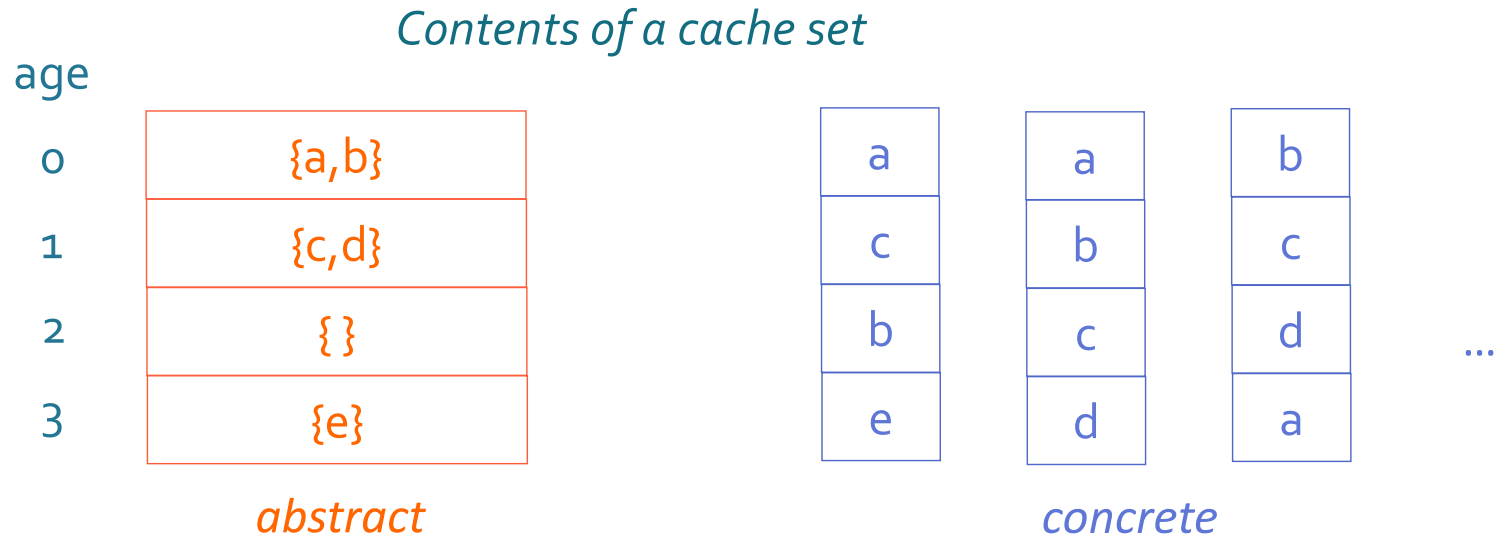
- join() function



Cache analysis by Abstract Interpretation

MAY analysis

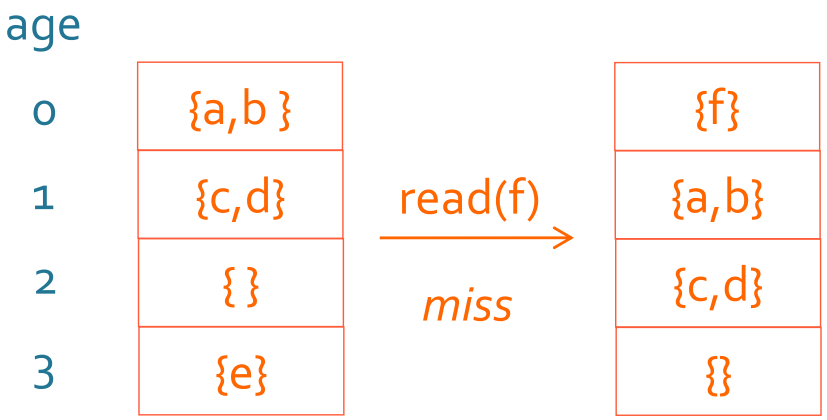
- abstract cache state = lower bounds on block ages

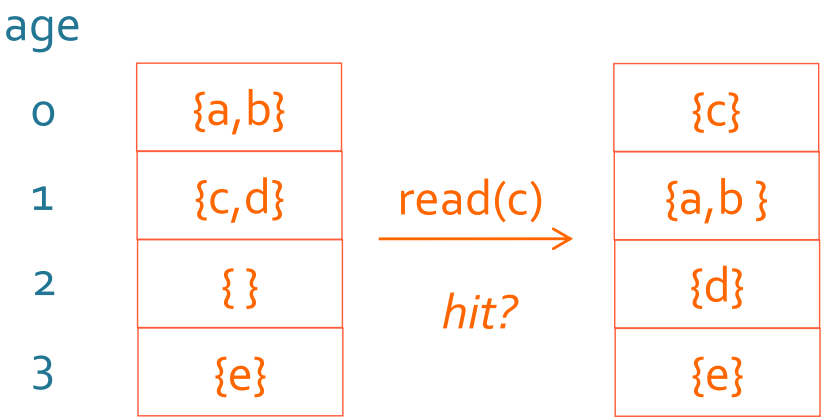


Cache analysis by Abstract Interpretation

MAY analysis

- update() function

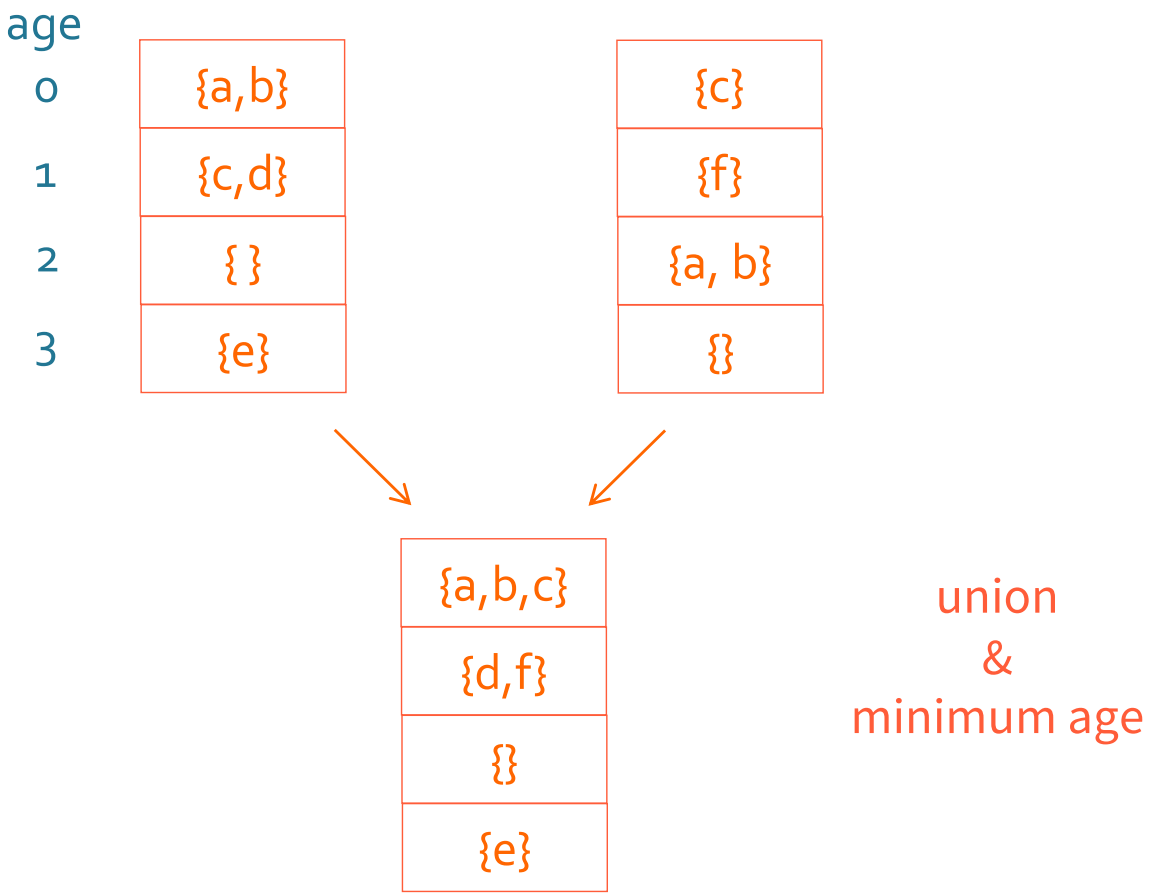




Cache analysis by Abstract Interpretation

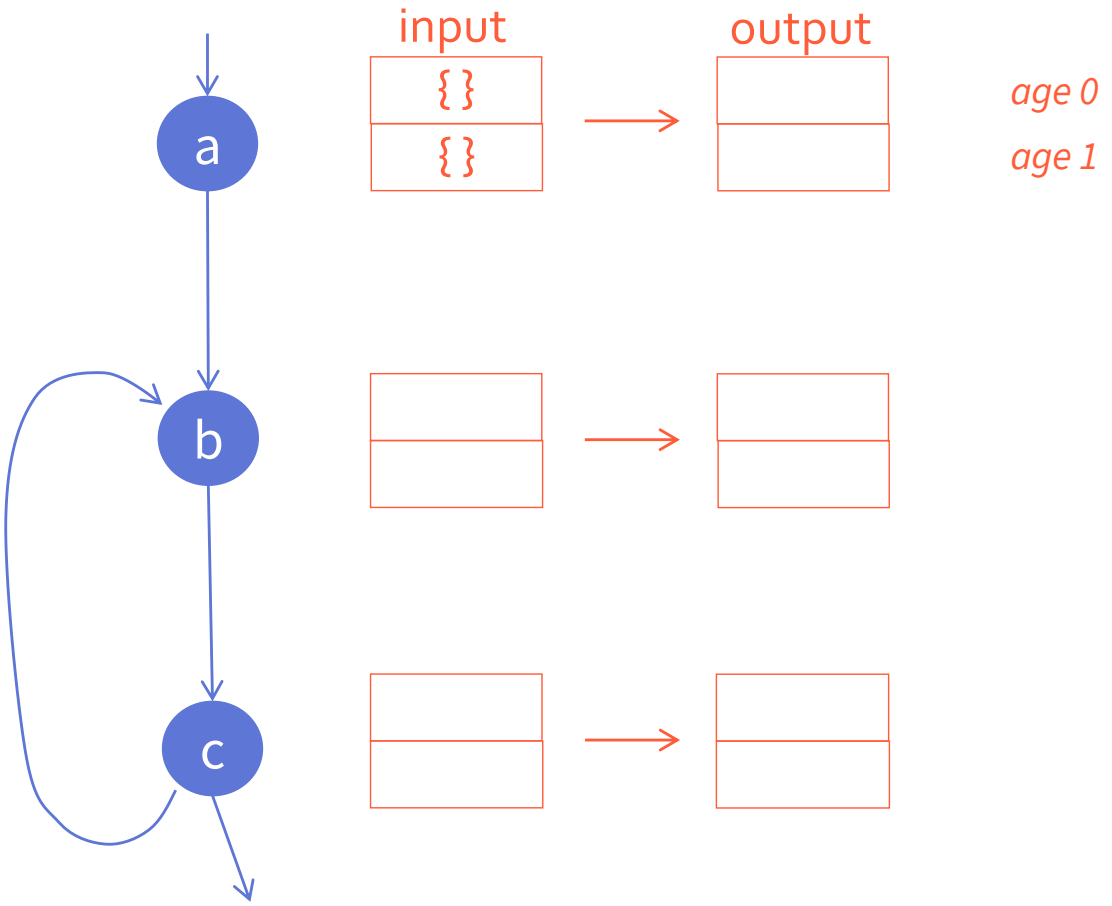
MAY analysis

- join() function



Cache analysis by Abstract Interpretation

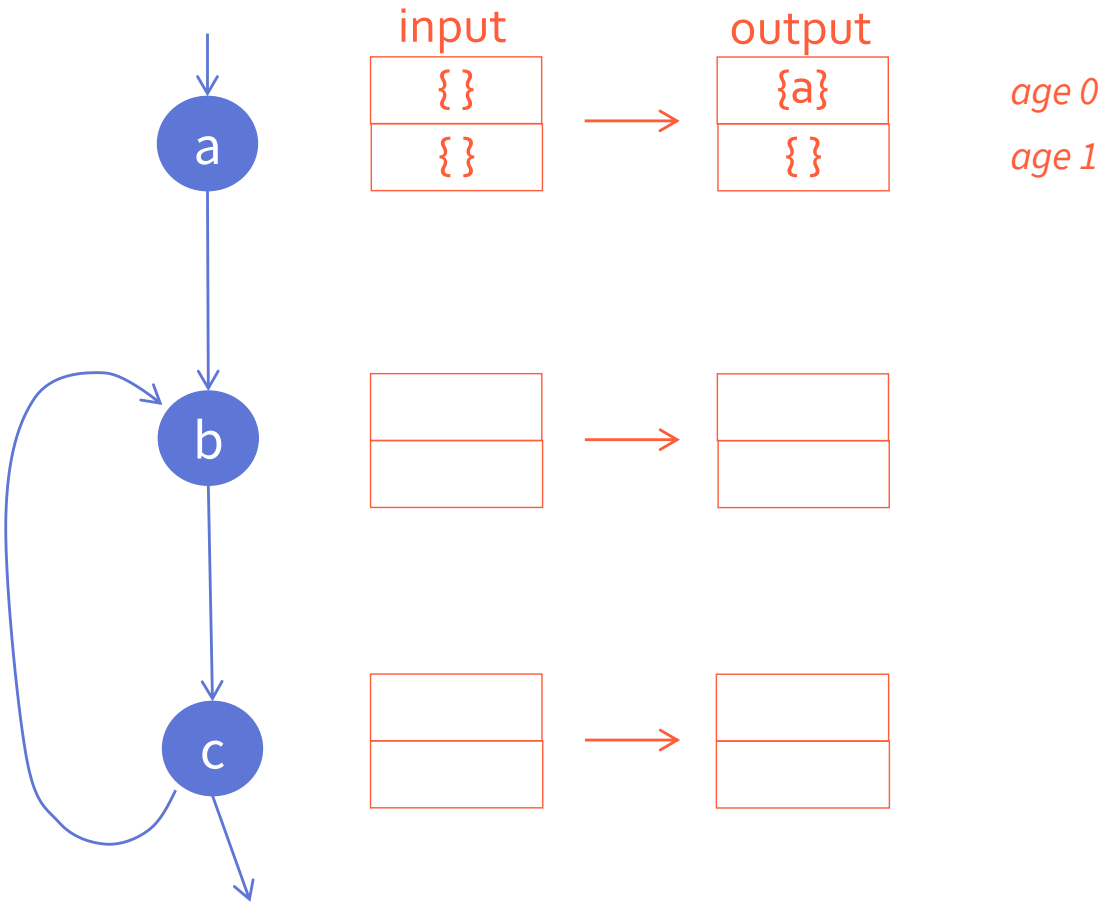
MAY analysis



join = union & minimum age

Cache analysis by Abstract Interpretation

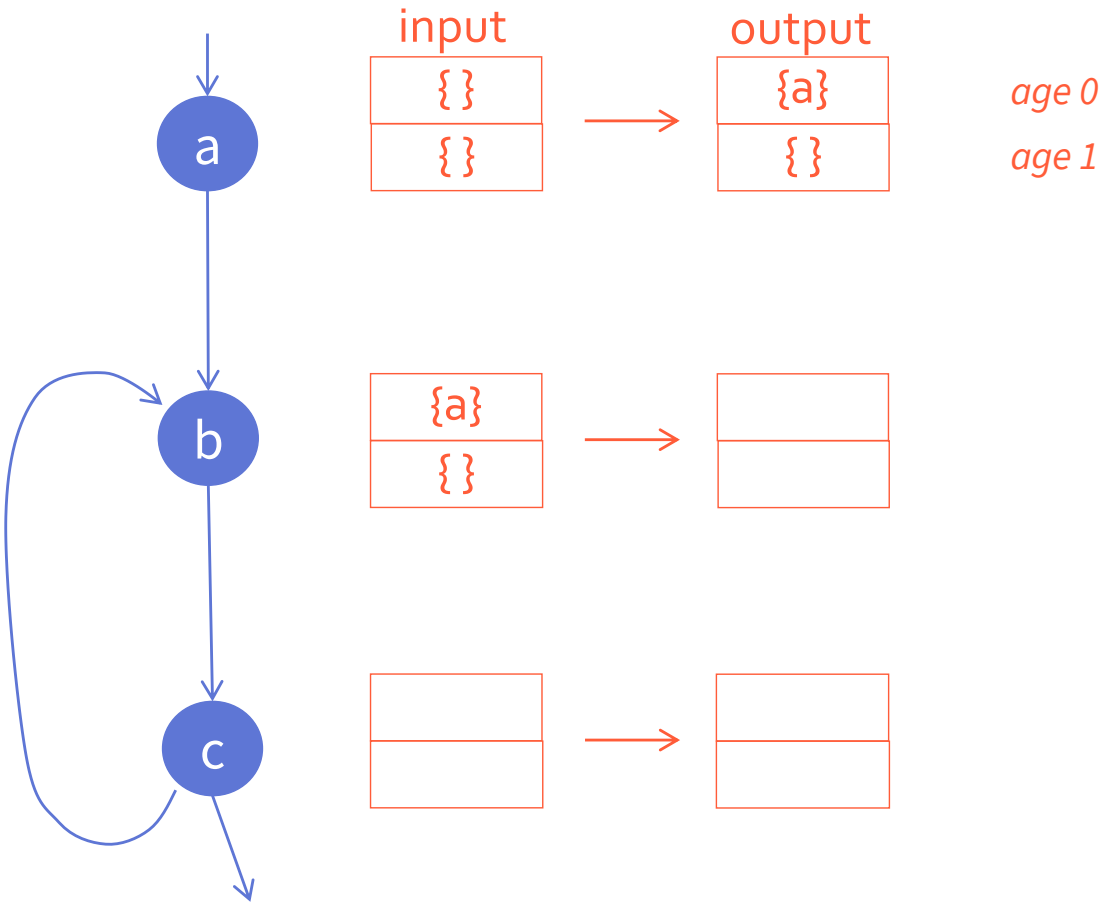
MAY analysis



join = union & minimum age

Cache analysis by Abstract Interpretation

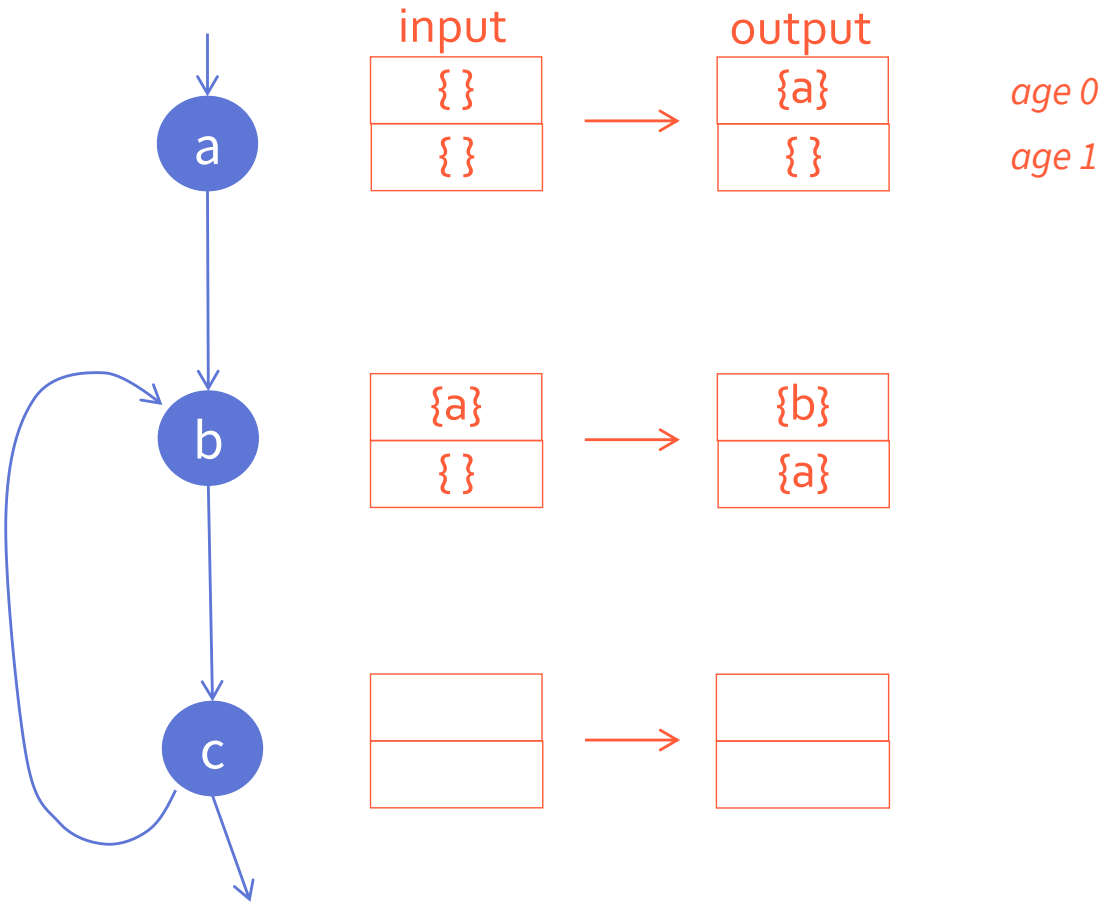
MAY analysis



join = union & minimum age

Cache analysis by Abstract Interpretation

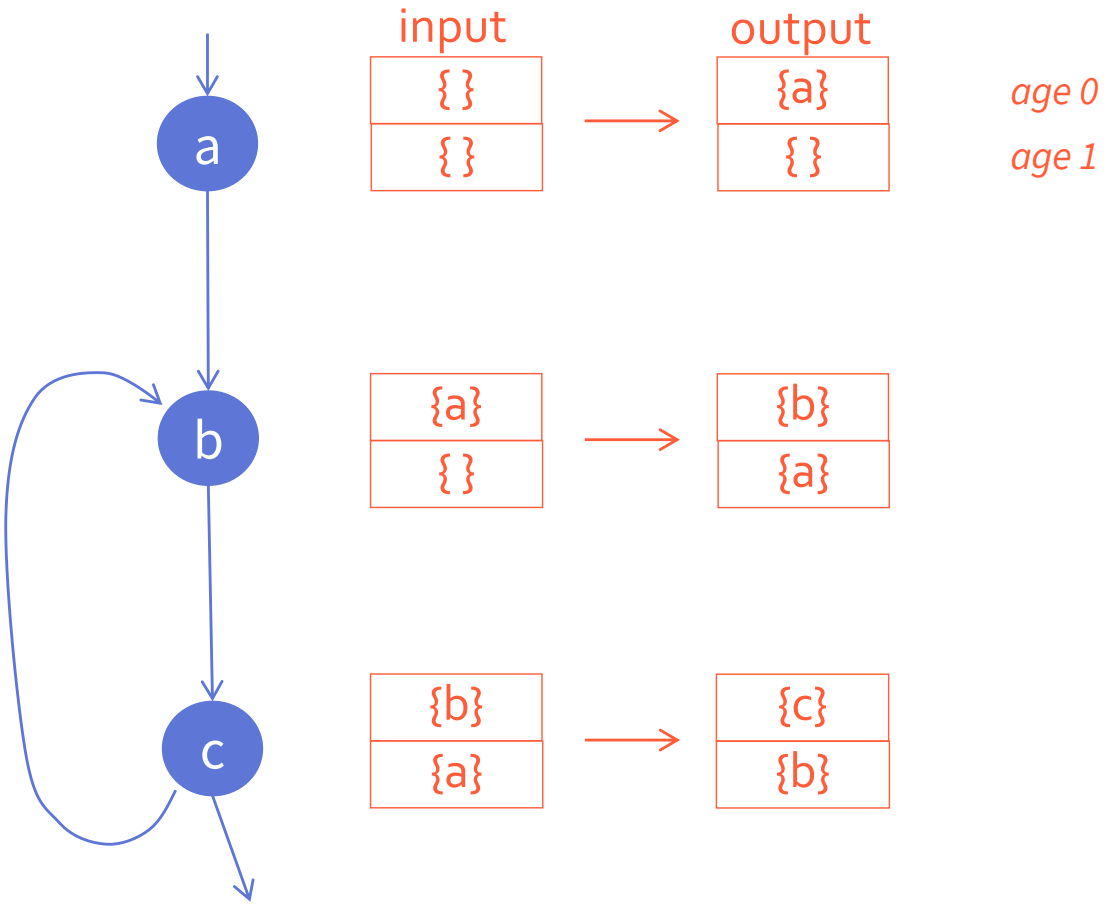
MAY analysis



join = union & minimum age

Cache analysis by Abstract Interpretation

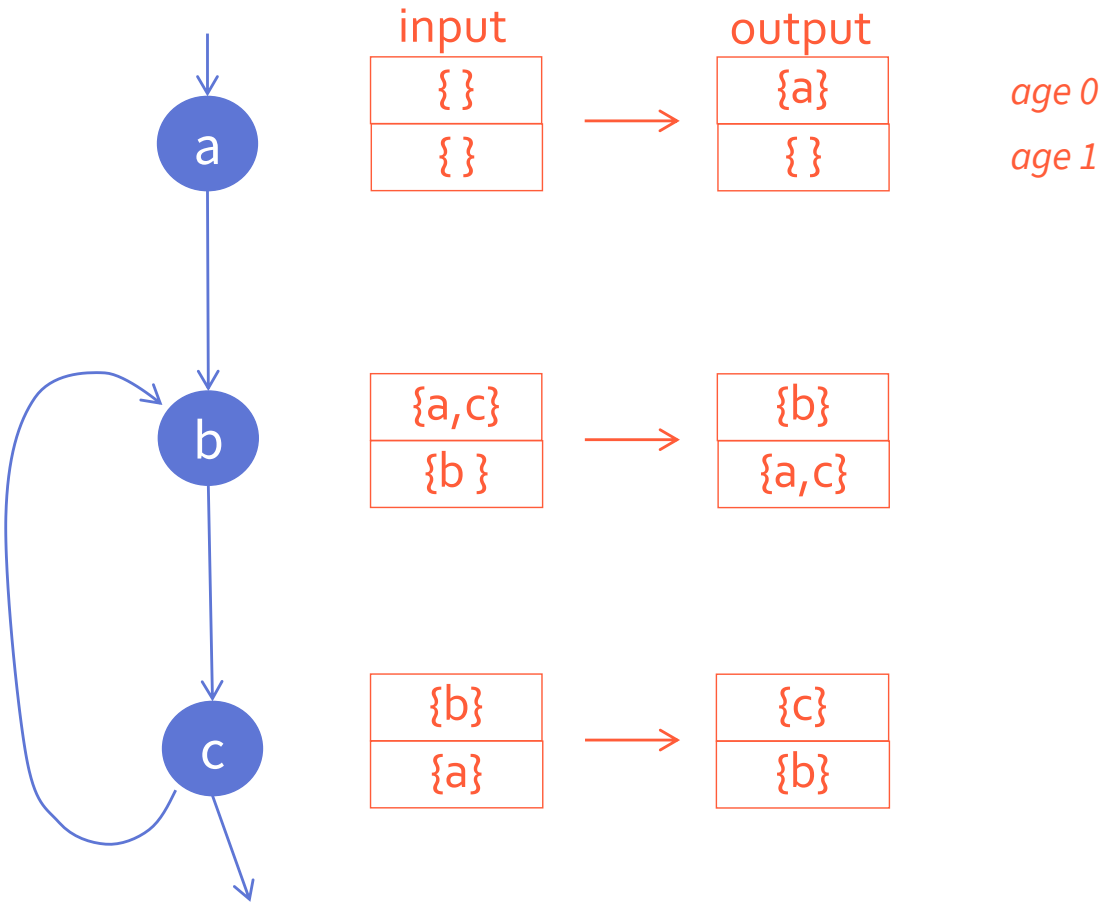
MAY analysis



join = union & minimum age

Cache analysis by Abstract Interpretation

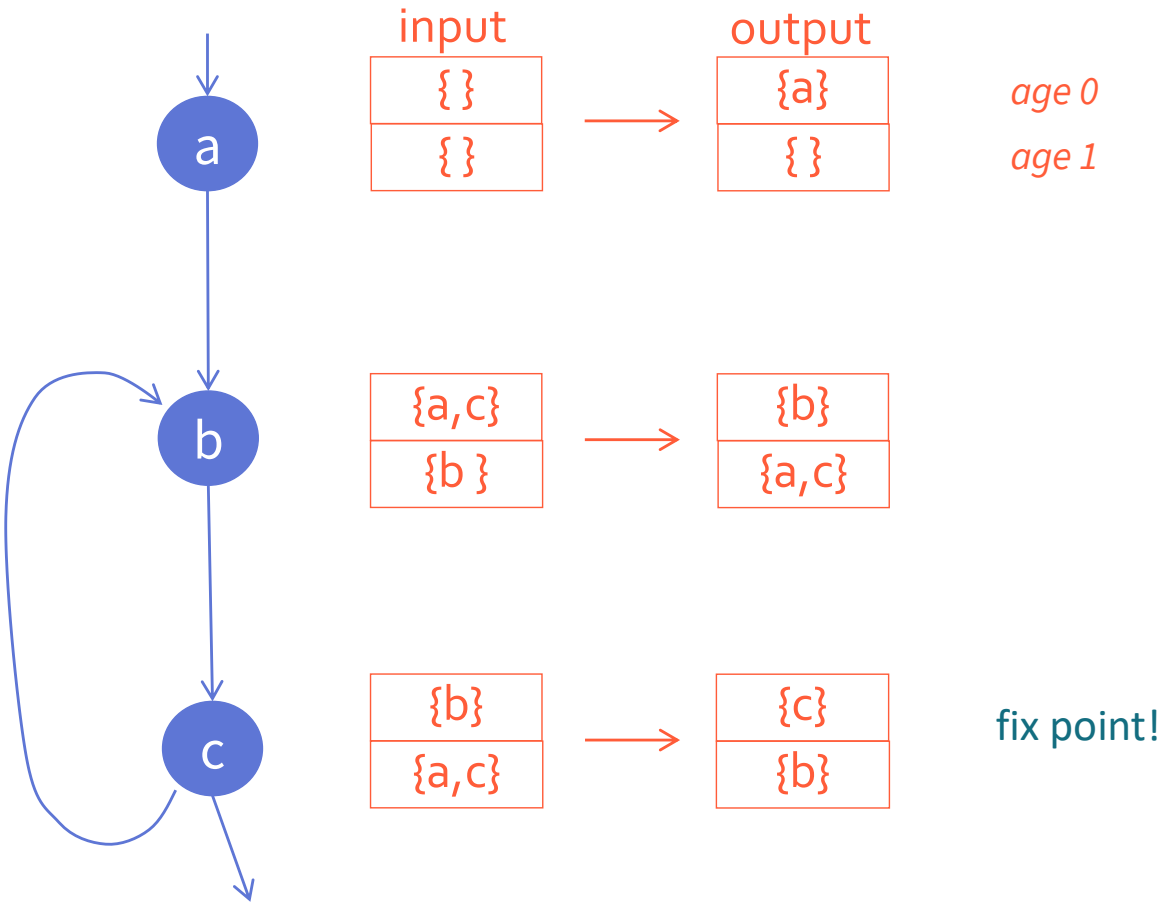
MAY analysis



join = union & minimum age

Cache analysis by Abstract Interpretation

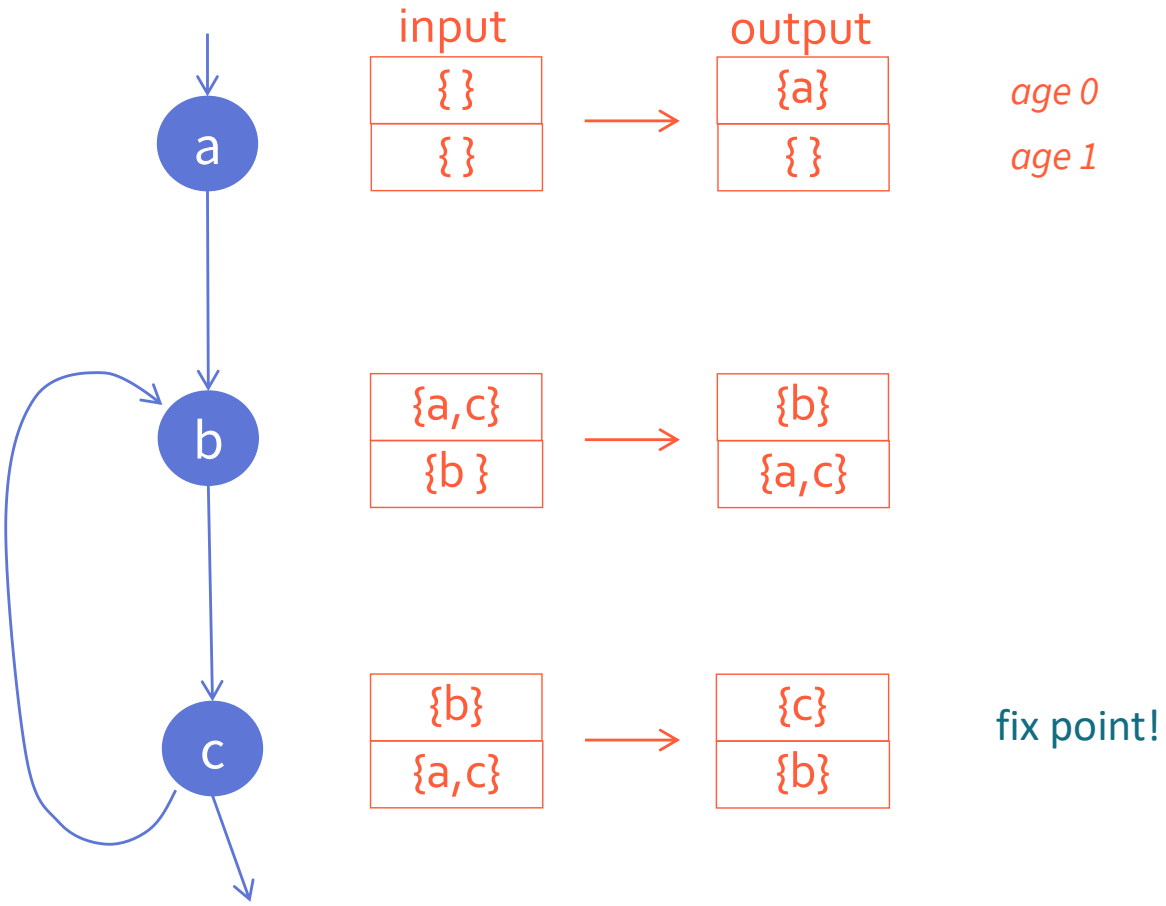
MAY analysis



join = union & minimum age

Cache analysis by Abstract Interpretation

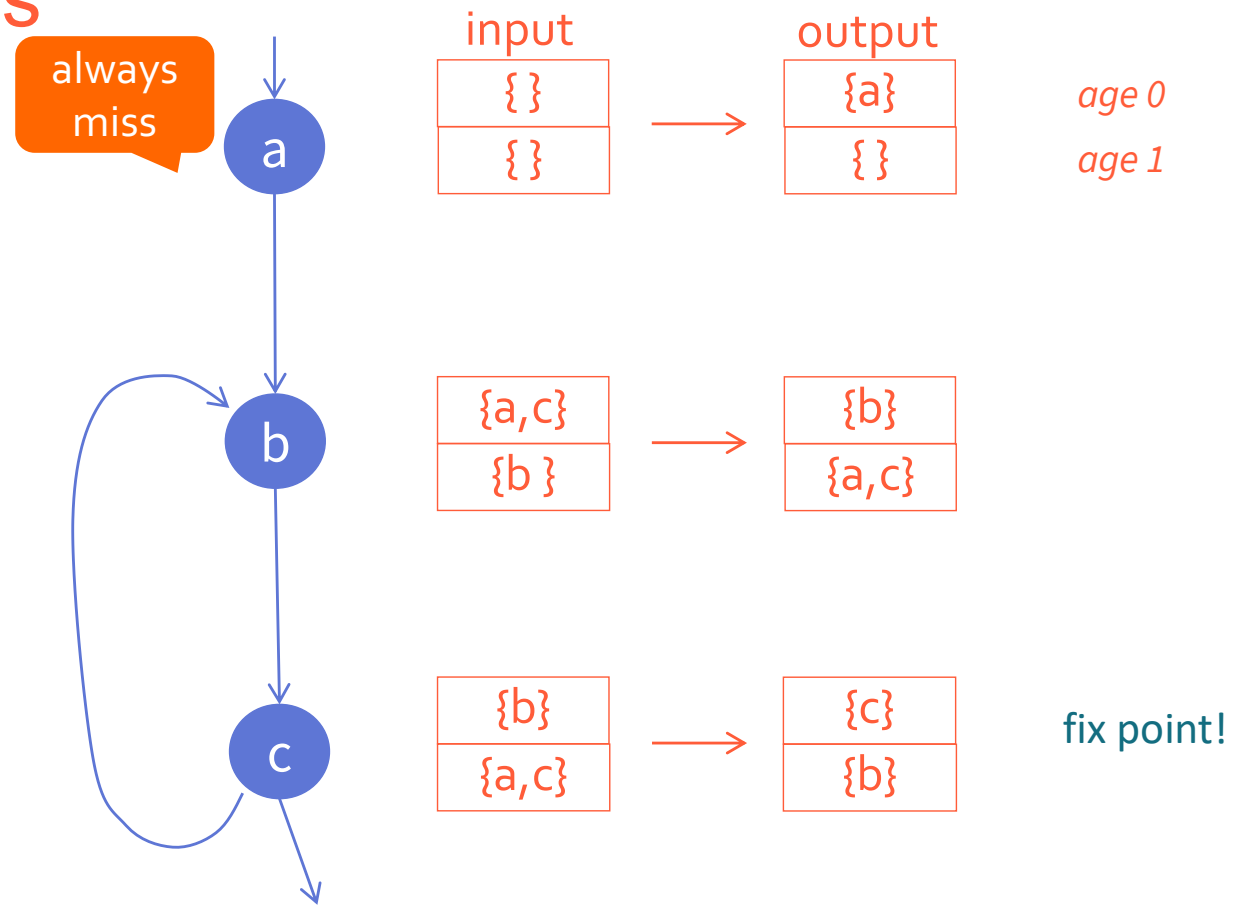
MAY analysis



join = union & minimum age

Cache analysis by Abstract Interpretation

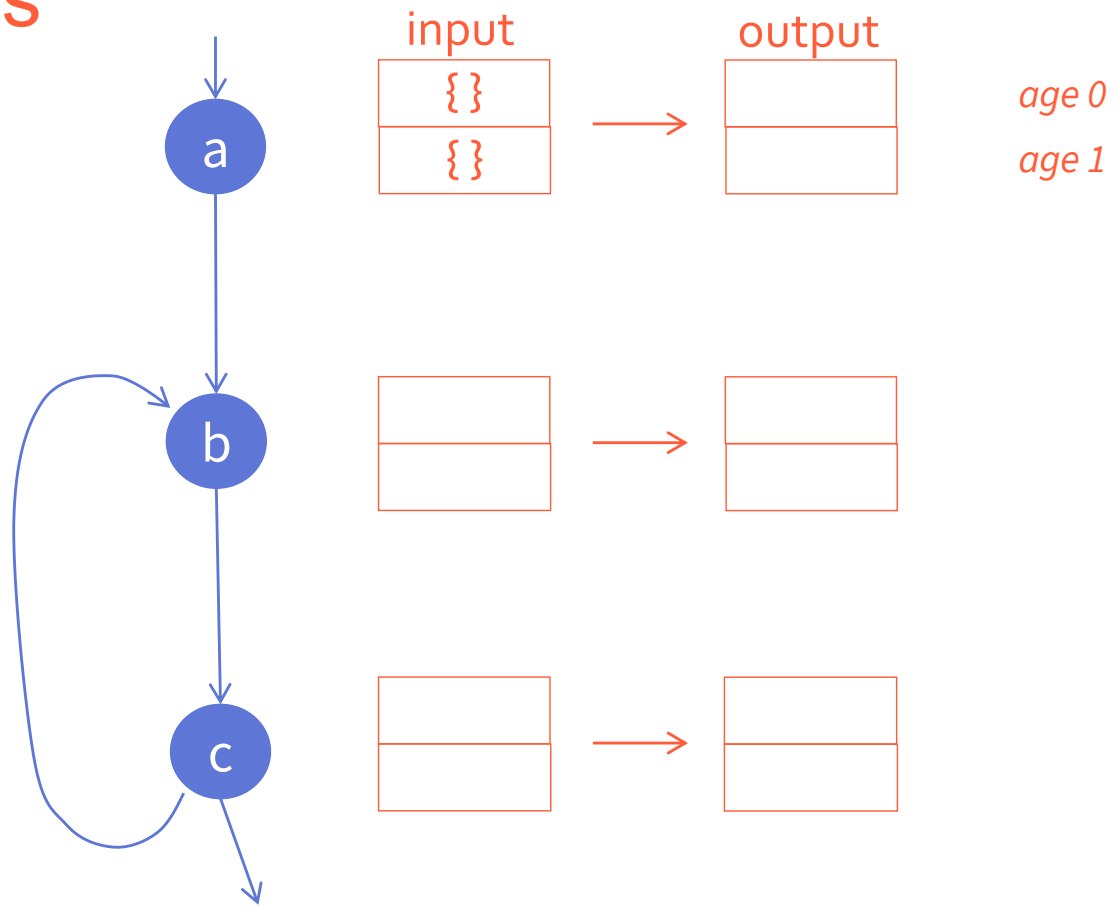
MAY analysis



join = union & minimum age

Cache analysis by Abstract Interpretation

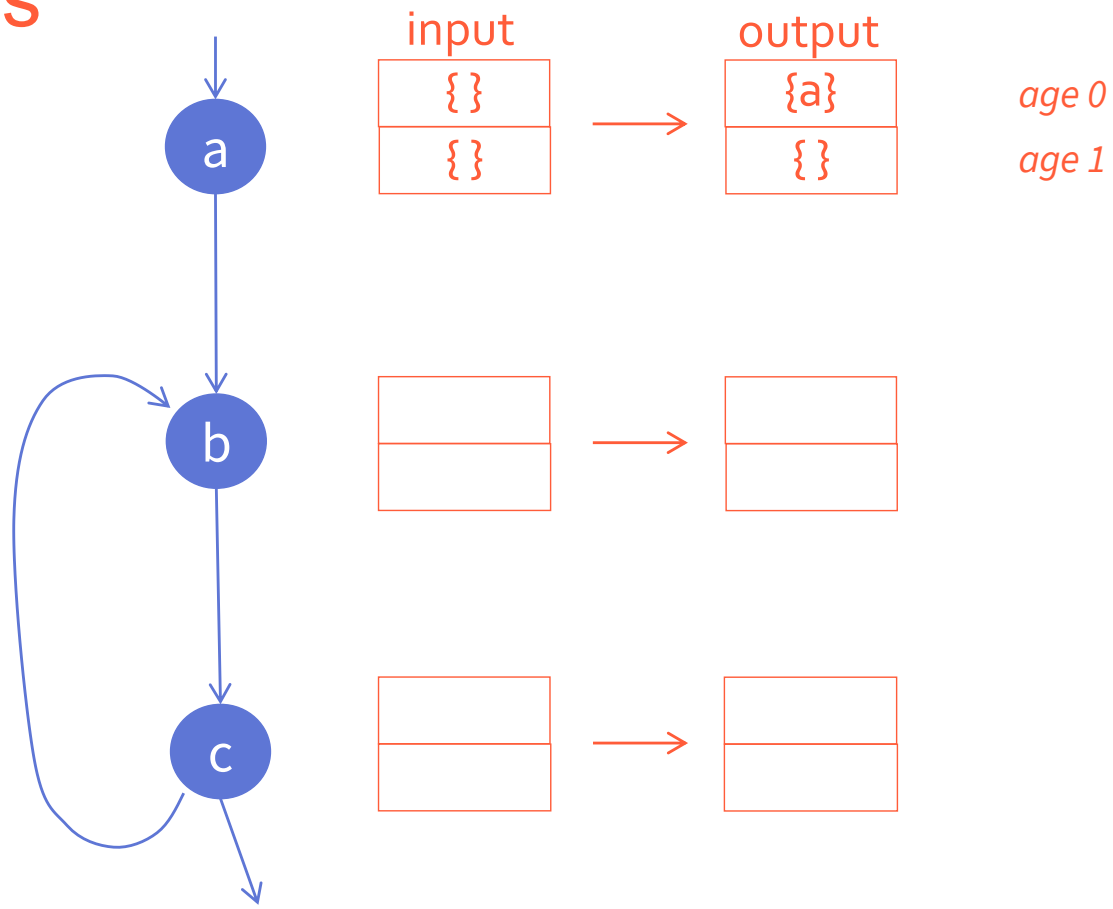
MUST analysis



join = intersection & maximum age

Cache analysis by Abstract Interpretation

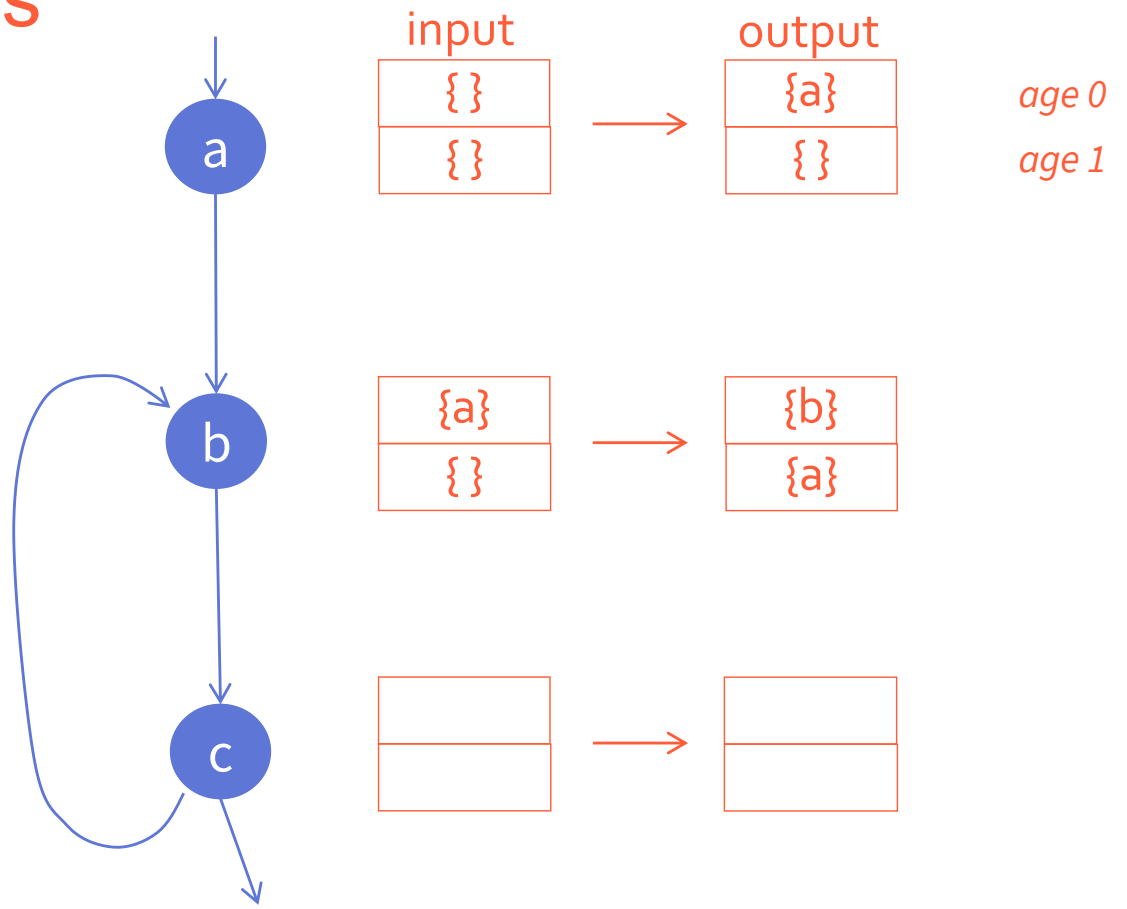
MUST analysis



join = intersection & maximum age

Cache analysis by Abstract Interpretation

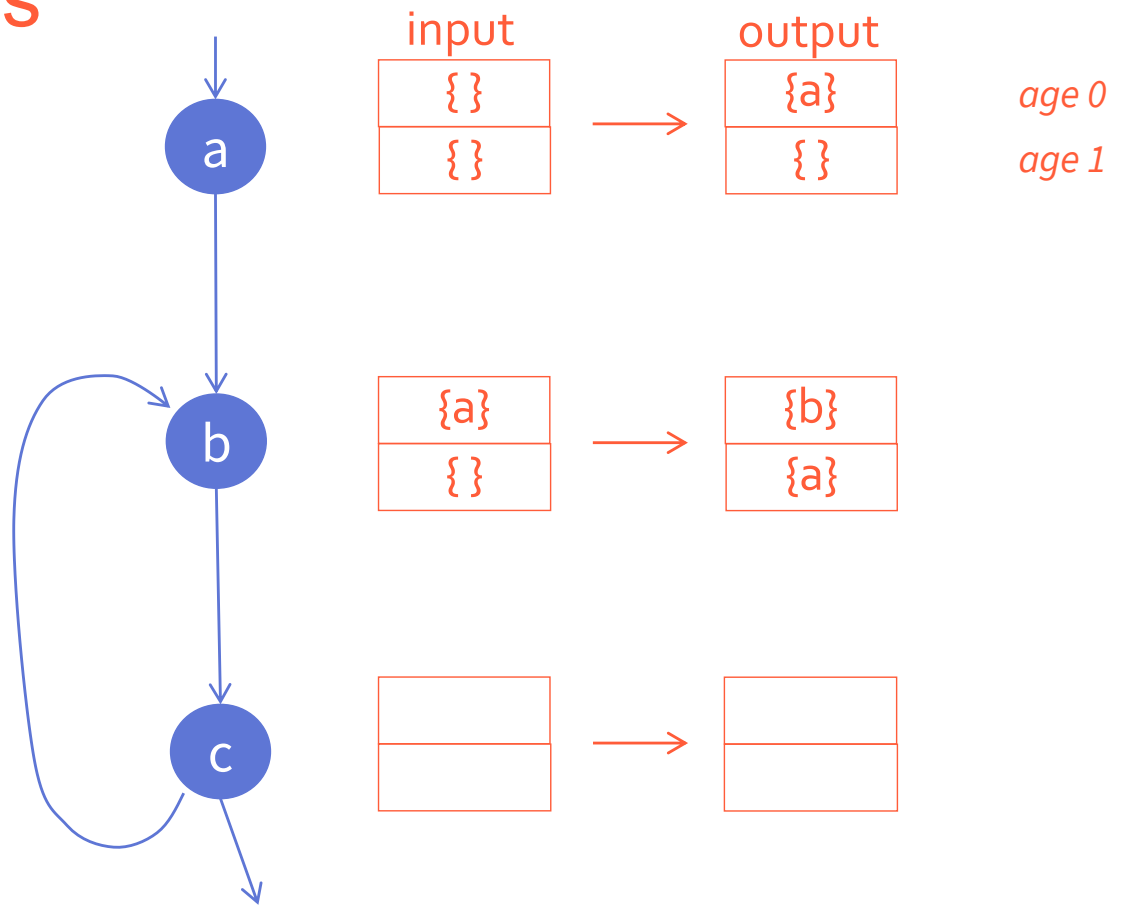
MUST analysis



join = intersection & maximum age

Cache analysis by Abstract Interpretation

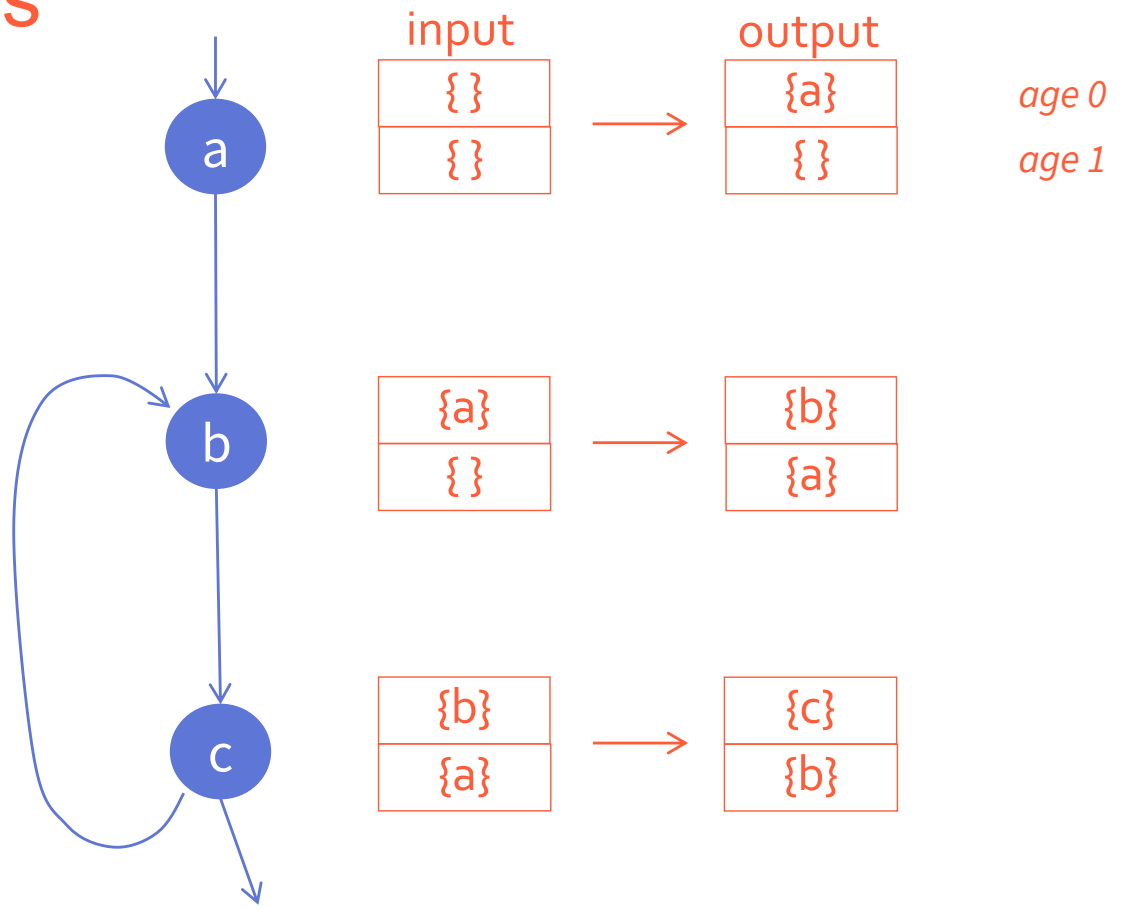
MUST analysis



join = intersection & maximum age

Cache analysis by Abstract Interpretation

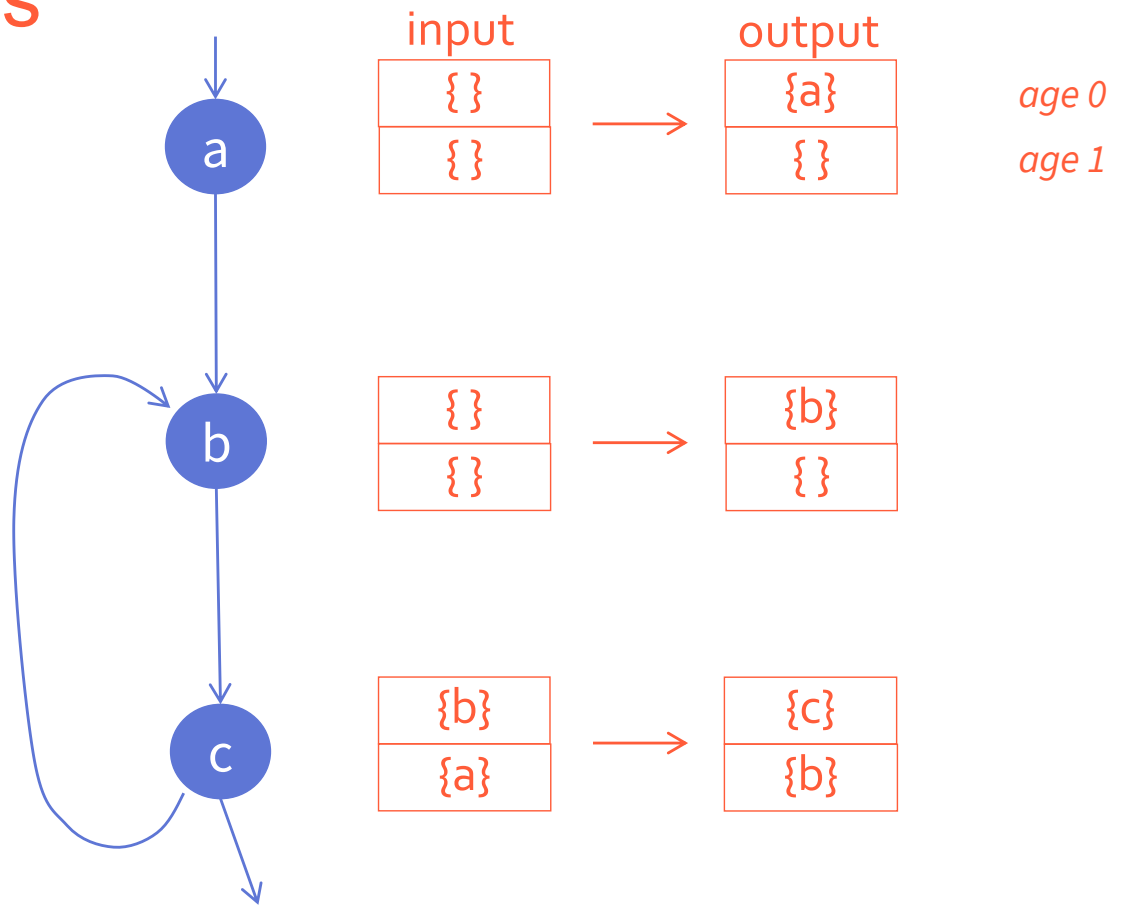
MUST analysis



join = intersection & maximum age

Cache analysis by Abstract Interpretation

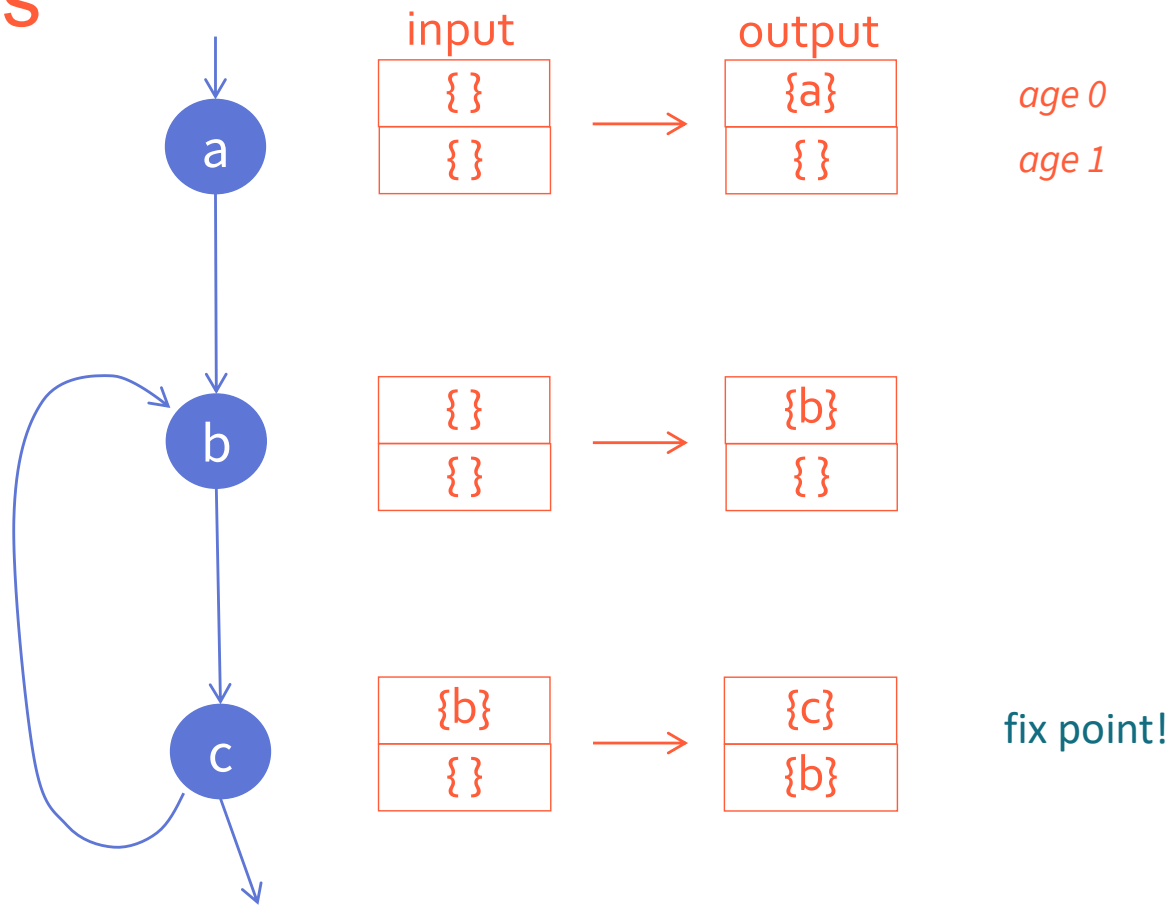
MUST analysis



join = intersection & maximum age

Cache analysis by Abstract Interpretation

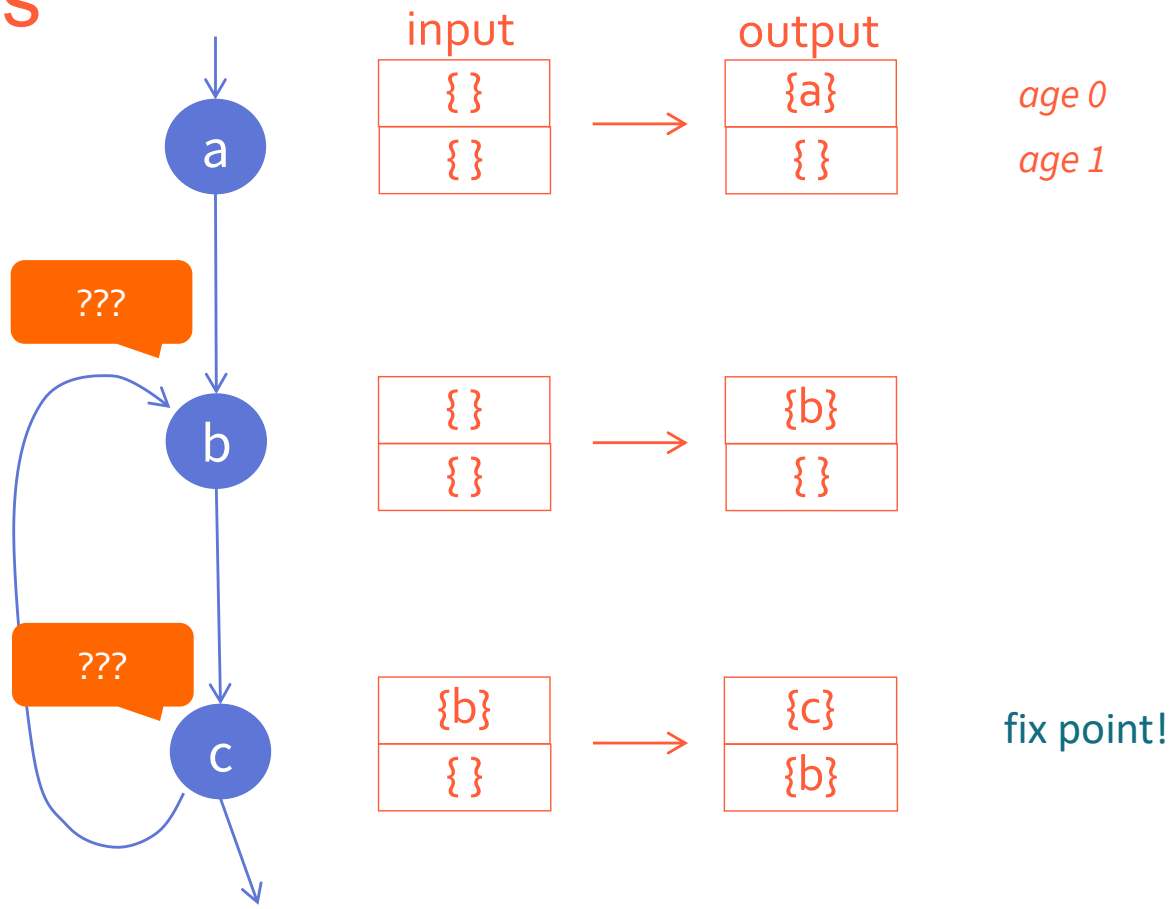
MUST analysis



join = intersection & maximum age

Cache analysis by Abstract Interpretation

MUST analysis



join = intersection & maximum age

Cache analysis by Abstract Interpretation

PERSISTENCE analysis

- abstract cache state = maximum ages + specific age for evicted blocks

age	
0	{ }
1	{a, b}
2	{ }
3	{c}
evicted	{e,f}

Cache analysis by Abstract Interpretation

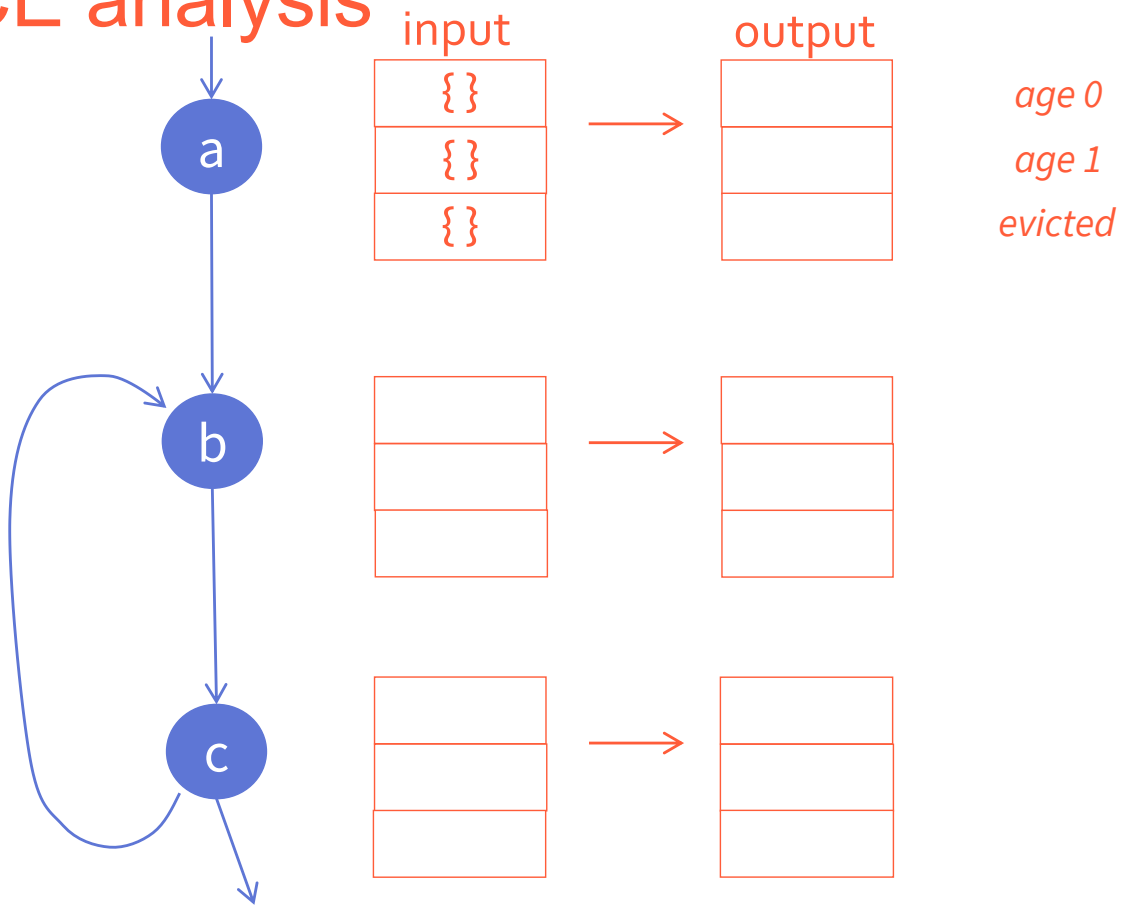
PERSISTENCE analysis

- update()
  - as for the MUST analysis
- join()
  - union & maximum age

- determines which blocks cannot be evicted once loaded in the cache

Cache analysis by Abstract Interpretation

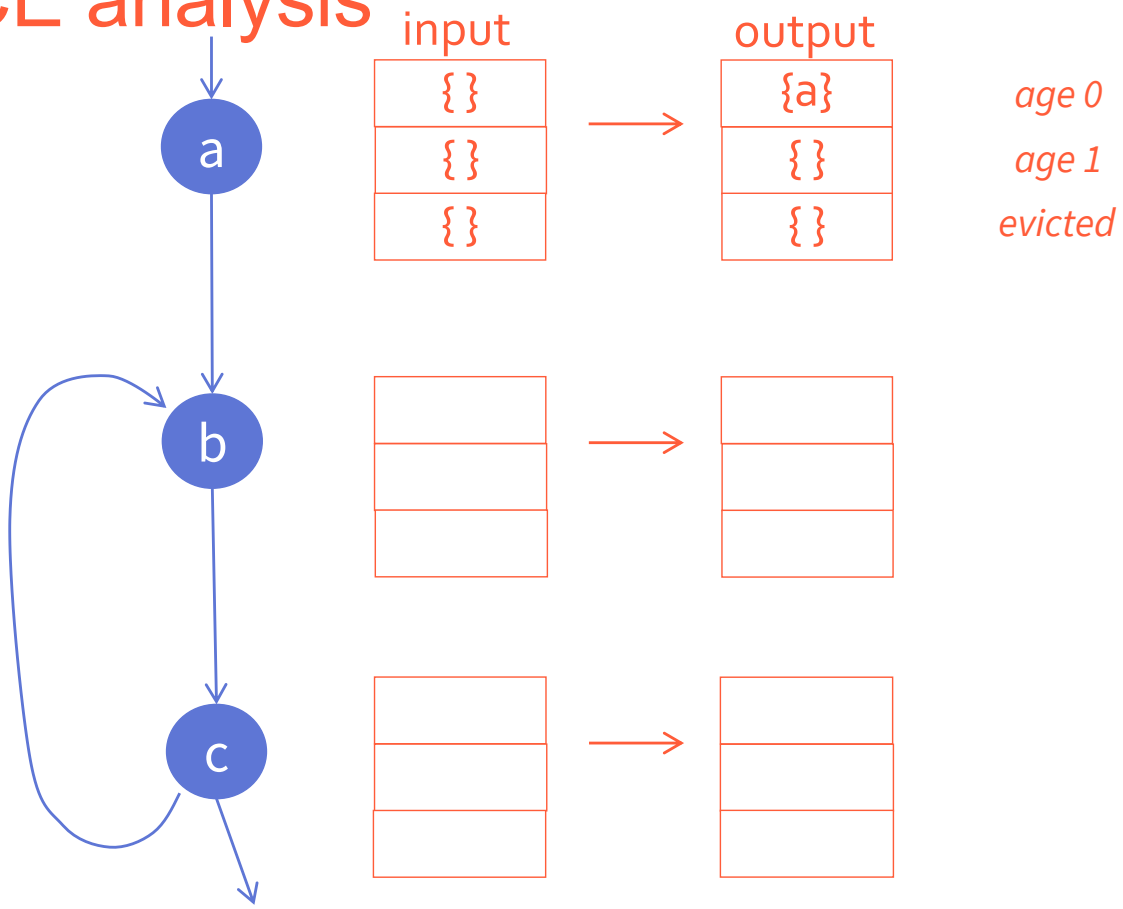
PERSISTENCE analysis



join = union & maximum age

Cache analysis by Abstract Interpretation

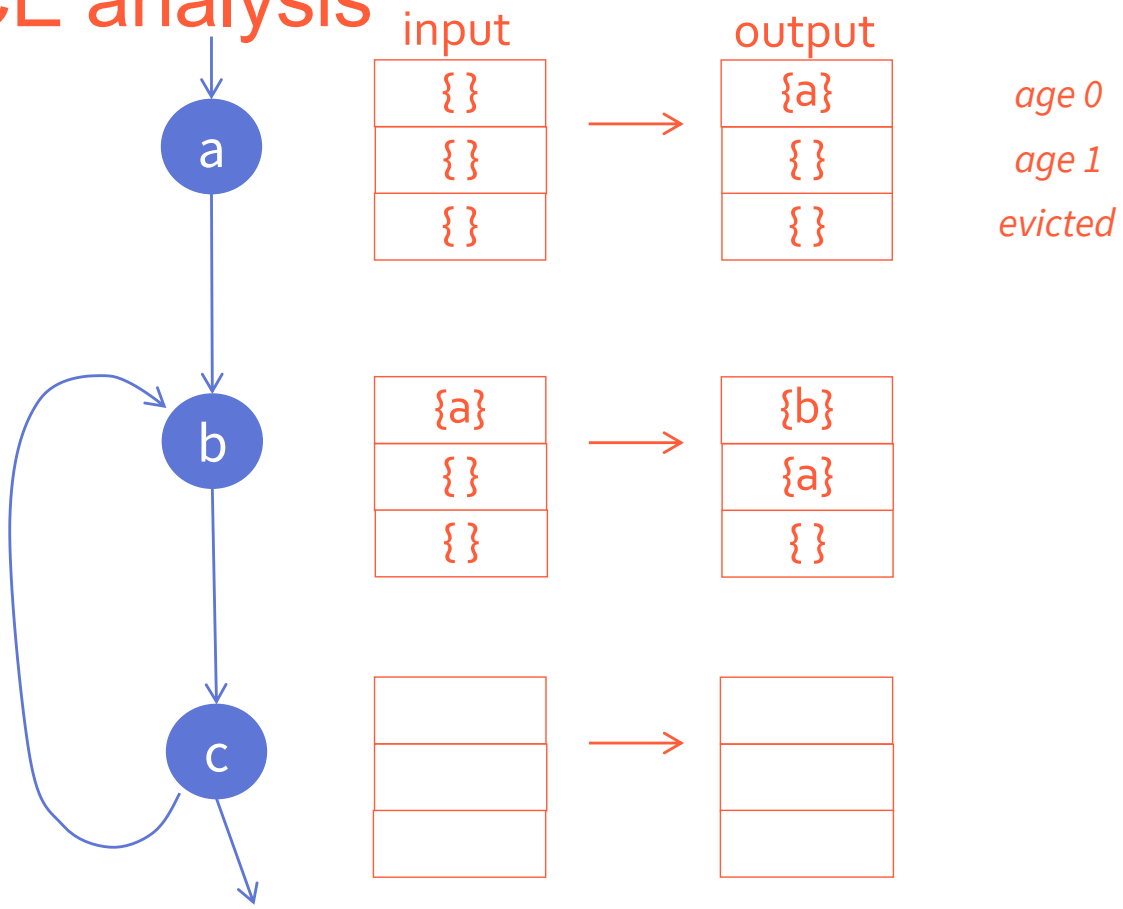
PERSISTENCE analysis



join = union & maximum age

Cache analysis by Abstract Interpretation

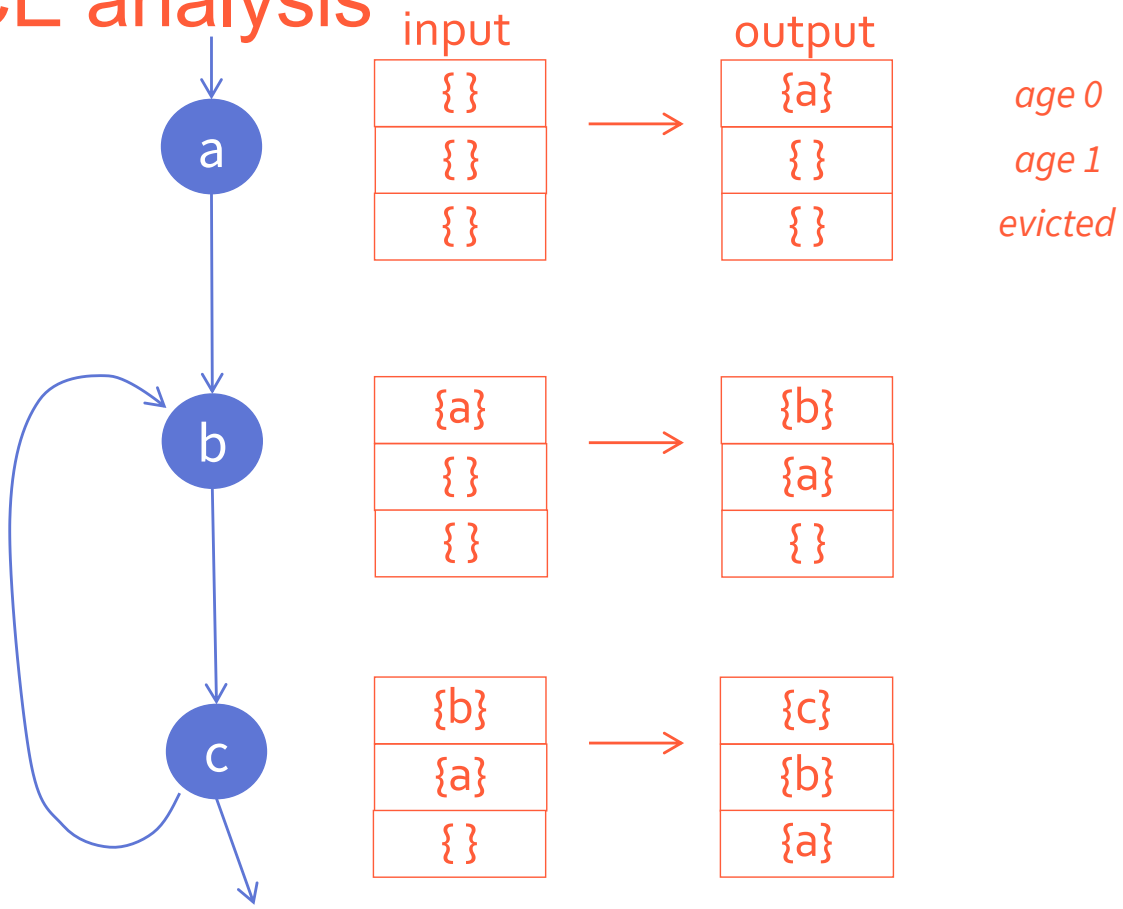
PERSISTENCE analysis



join = union & maximum age

Cache analysis by Abstract Interpretation

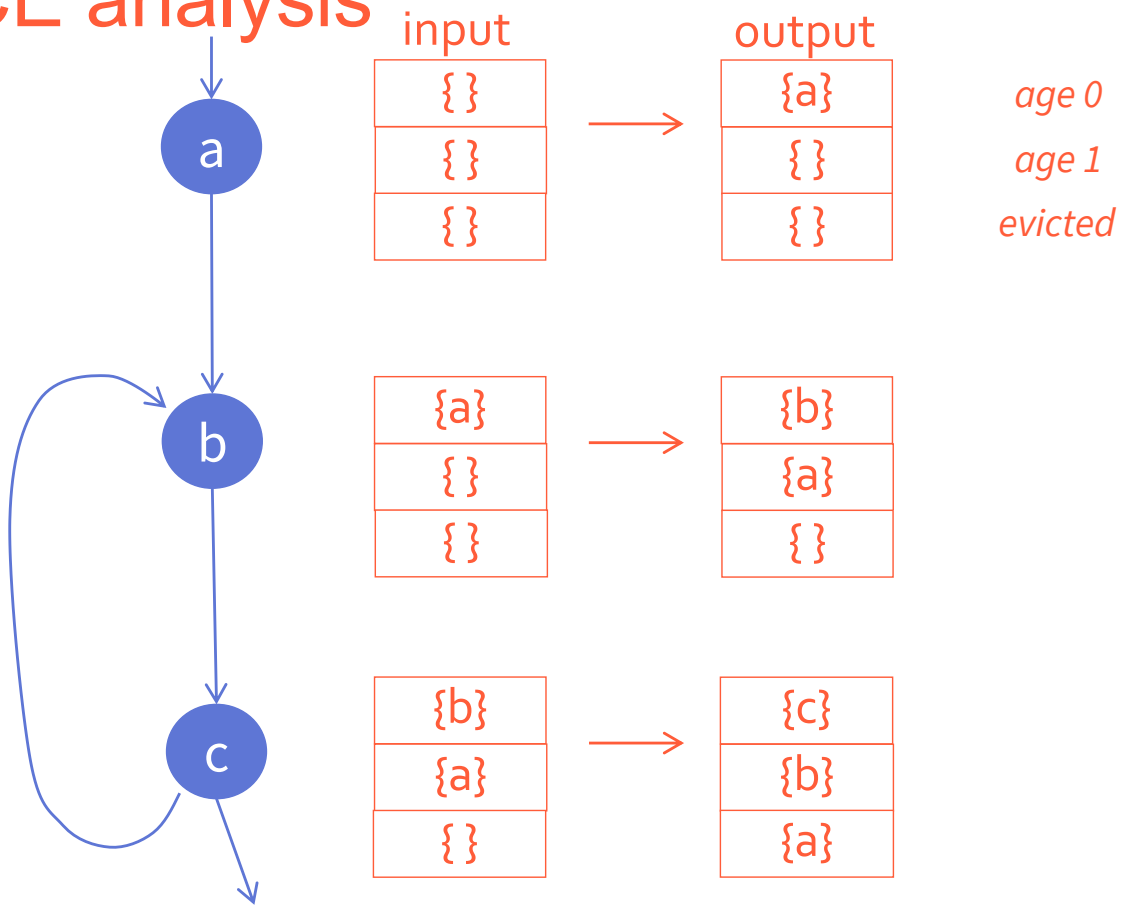
PERSISTENCE analysis



join = union & maximum age

Cache analysis by Abstract Interpretation

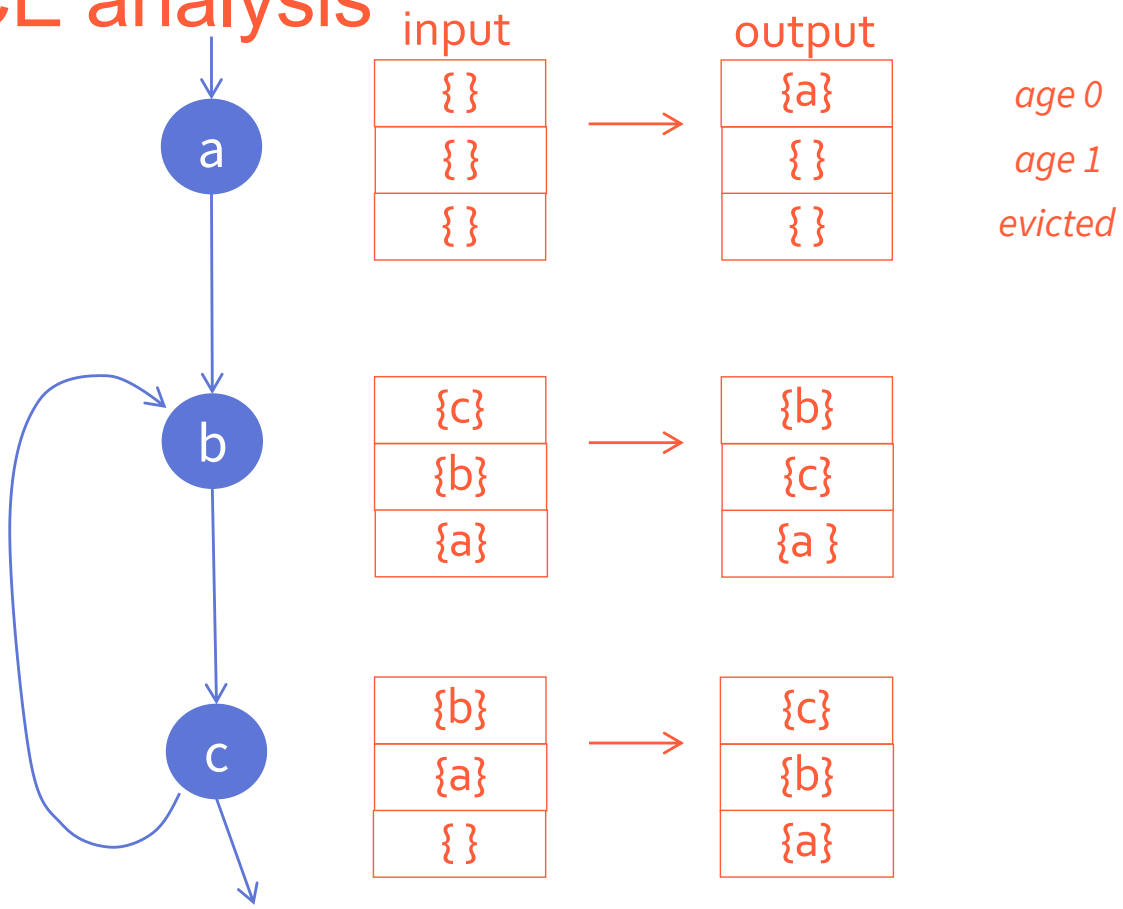
PERSISTENCE analysis



join = union & maximum age

Cache analysis by Abstract Interpretation

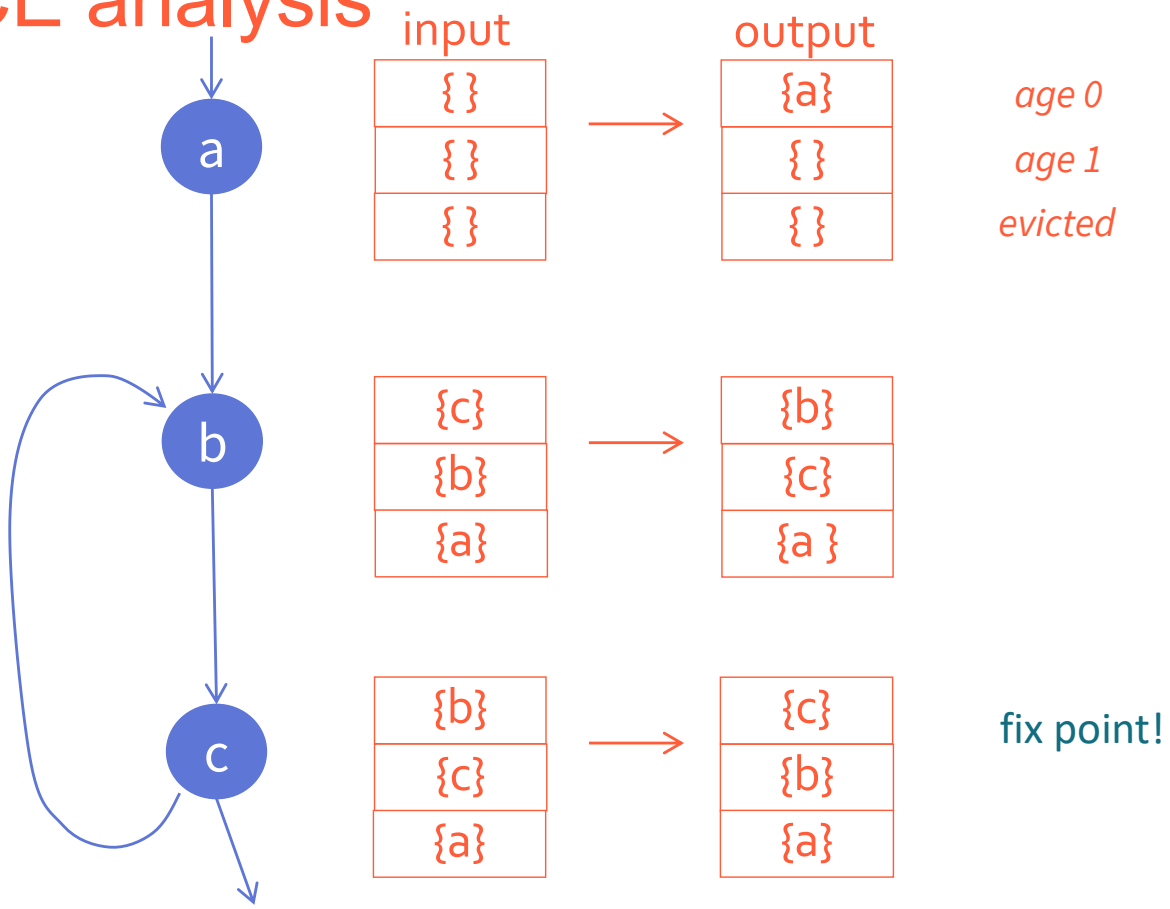
PERSISTENCE analysis



join = union & maximum age

Cache analysis by Abstract Interpretation

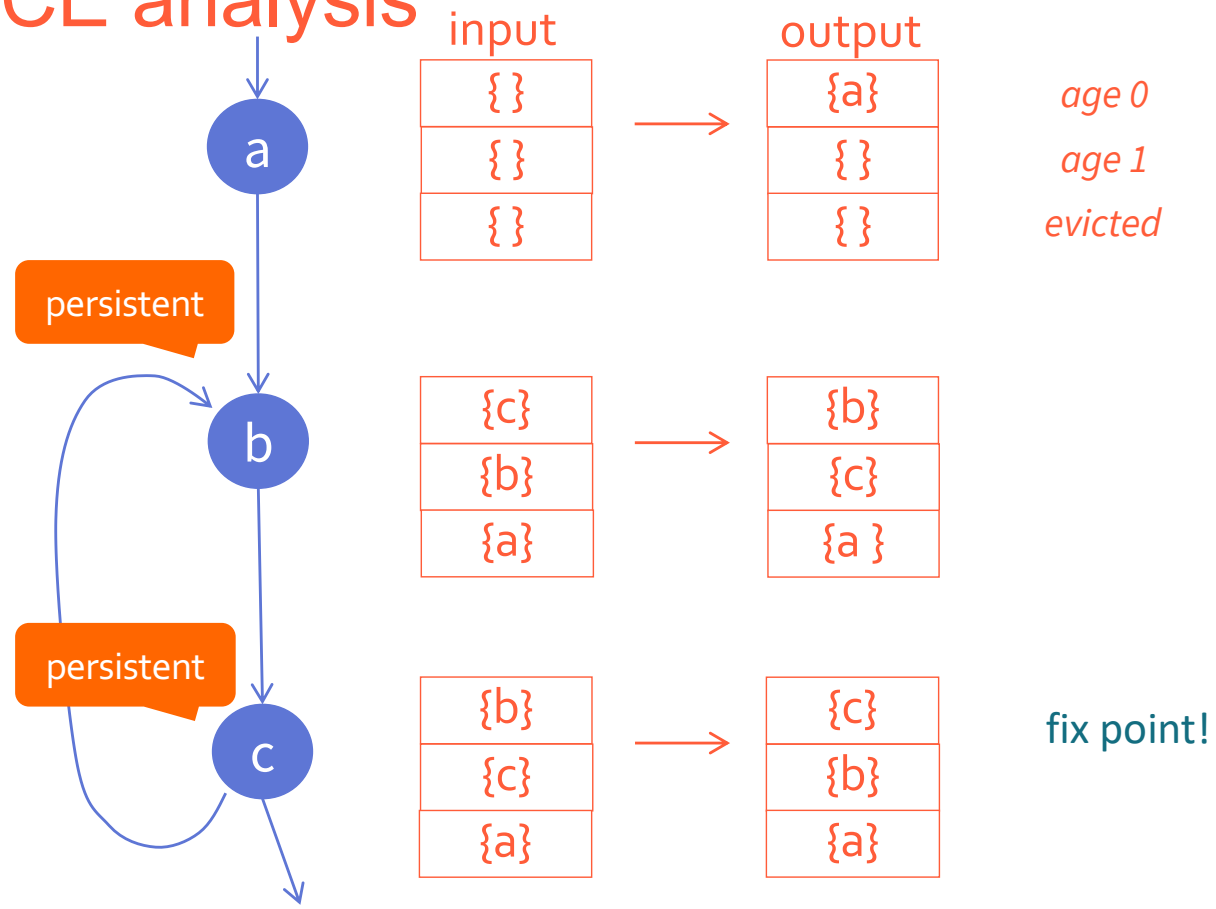
PERSISTENCE analysis



join = union & maximum age

Cache analysis by Abstract Interpretation

PERSISTENCE analysis



join = union & maximum age

Analysis of data caches

Main challenge:

- unknown and multiple memory addresses

```
char a[MAX];

void isort() {
   int i,p,j,x;

   for (i=1 ; i<MAX ; i++) {
      p = 0;
      while(a[p] < a[i])
         p++;
      x = a[i];
      for (j=i-1 ; j>=p ; j--)
         a[j+1] = a[j];
      a[p] = x;
   }
}
```

```
isort:   stmfd sp!,{r0-6}
         mov  r0,#1
         adr  r10,a
for1:    cmp  r0,#MAX
         bcs  end
         mov  r1,#0
while:   ldrb r2,[r10,r1]
         ldrb r3,[r10,r0]
         cmp  r2,r3
         bcs  next1
         add  r1,r1,#1
         b    while
next1:   sub  r4,r0,#1
for2:    cmp  r4,r1
         blt  next2
         ldrb r5,[r10,r4]
         add  r6,r4,#1
         strb r5,[r10,r6]
         sub  r4,r4,#1
         b for2
next2:   strb r3,[r10,r1]
         add  r0,r0,#1
         b    for1
end:     ldmfd sp!,{r0-6}
         mov pc,lr
```

Analysis of a hierarchy of caches