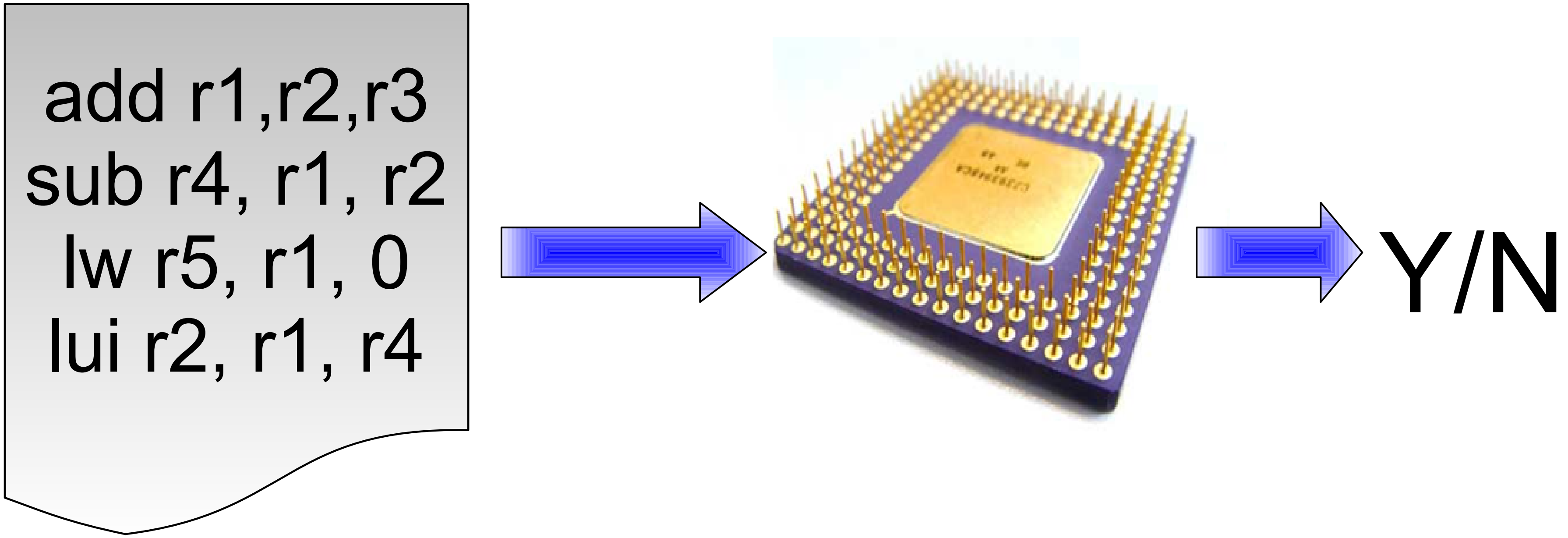
SMT-based Test Program Generation for Cache Memory

Evgeny Kornikhin
Moscow State University

Testing by Test Programs

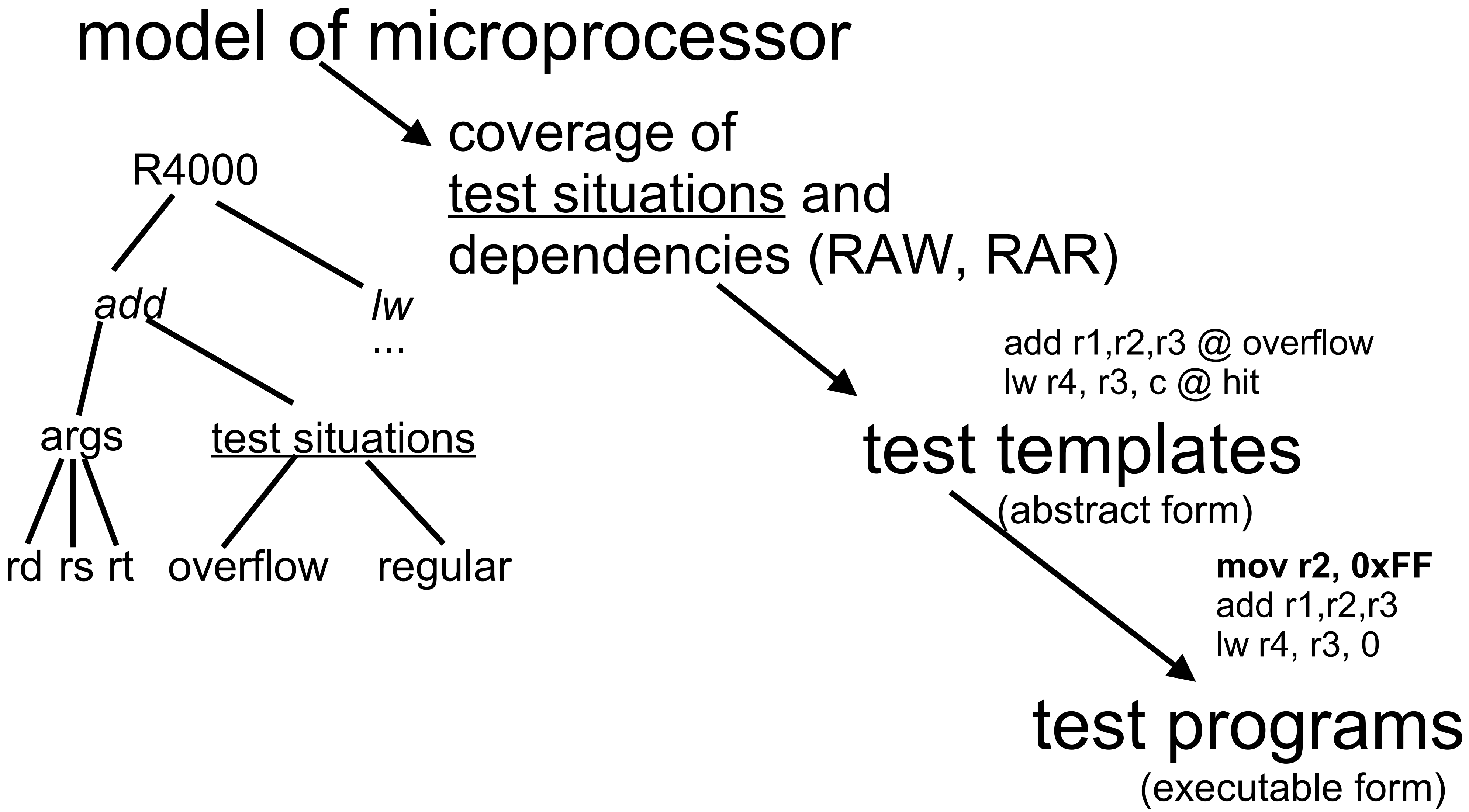


```
add r1,r2,r3
sub r4, r1, r2
lw r5, r1, 0
lui r2, r1, r4
```

assembler program
(test program)

microprocessor

Test Programs Generation



model of microprocessor

coverage of
<u>test situations</u> and
dependencies (RAW, RAR)

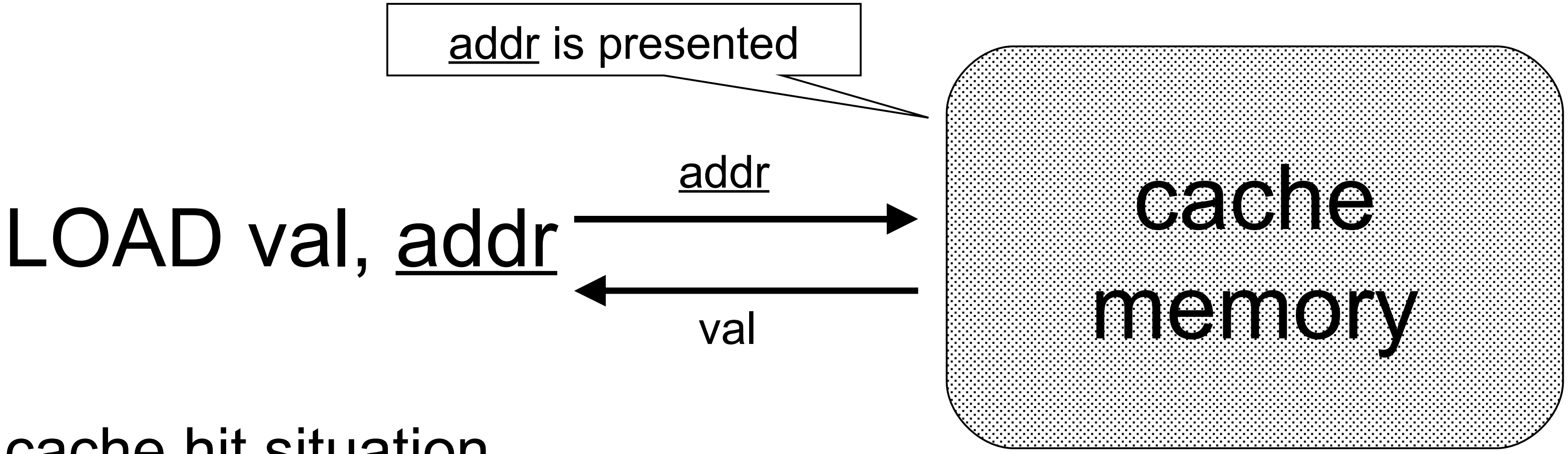
add r1,r2,r3 @ overflow
lw r4, r3, c @ hit

test templates
(abstract form)

mov r2, 0xFF
add r1,r2,r3
lw r4, r3, 0

test programs
(executable form)

Cache memory behavior



addr is presented

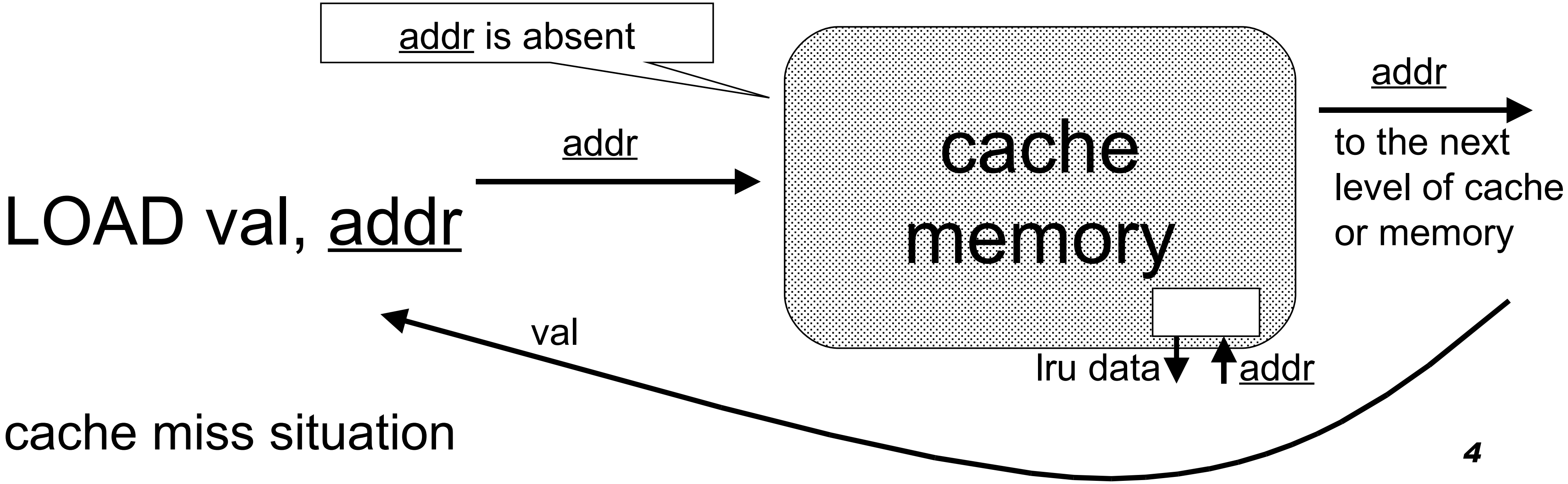
addr

LOAD val, addr

val

cache memory

cache hit situation



addr is absent

addr

LOAD val, addr

cache memory

addr

to the next level of cache or memory

val

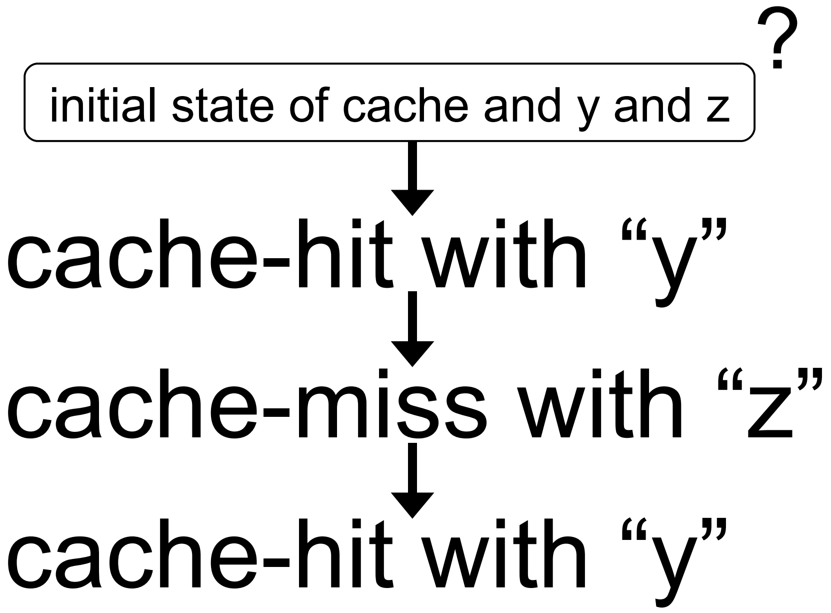
lru data

addr

cache miss situation

Problem again

LOAD x, y @ hit
STORE u, z @ miss
LOAD z, y @ hit



?

initial state of cache and y and z

cache-hit with “y”

cache-miss with “z”

cache-hit with “y”

initial state of cache

(addresses of cached data)

and registers = ?

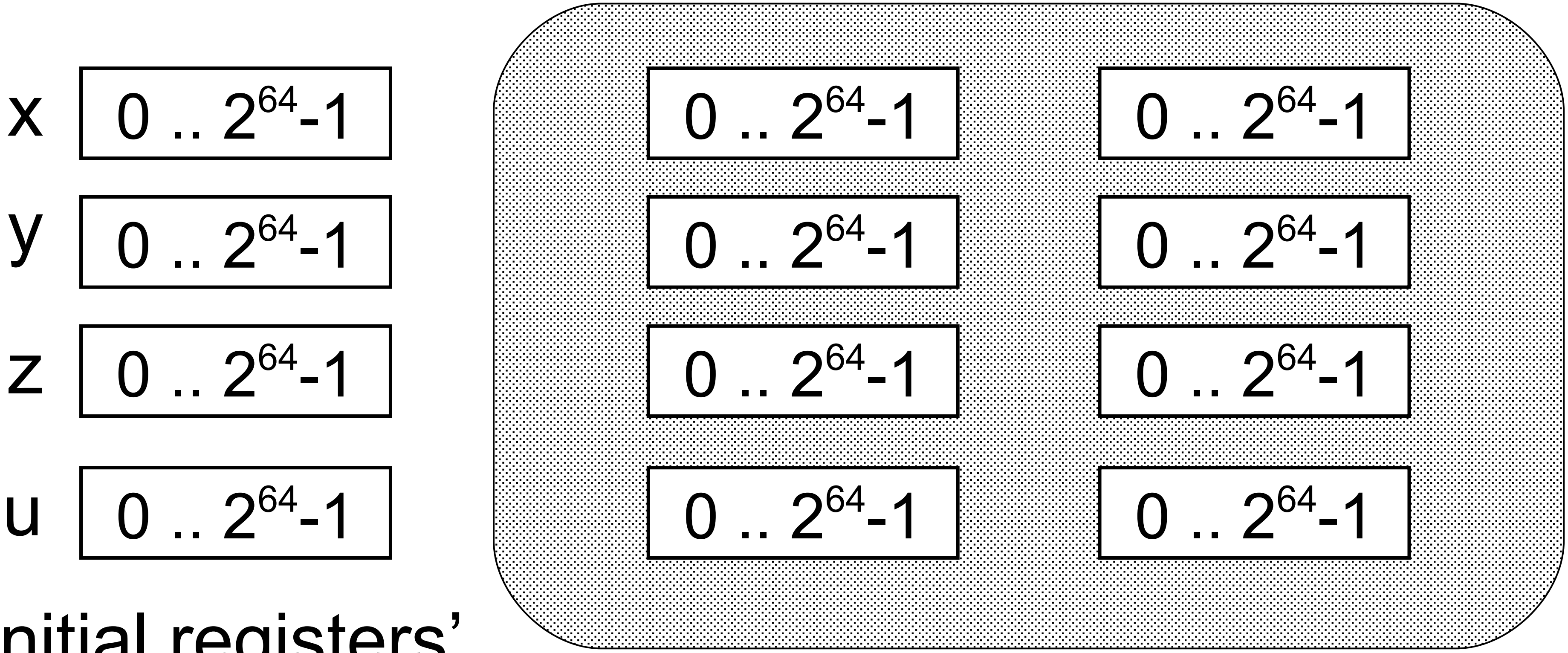
Huge exhaustive search

```
LOAD x, y @ hit
STORE u, z @ miss
LOAD z, y @ hit
```

select values

↓ ↑

run template



x $0 .. 2^{64}-1$

y $0 .. 2^{64}-1$

z $0 .. 2^{64}-1$

u $0 .. 2^{64}-1$

$0 .. 2^{64}-1$ $0 .. 2^{64}-1$

$0 .. 2^{64}-1$ $0 .. 2^{64}-1$

$0 .. 2^{64}-1$ $0 .. 2^{64}-1$

$0 .. 2^{64}-1$ $0 .. 2^{64}-1$

initial registers' values

initial cache state

Hard behavior constraints

LOAD x, y @ hit
(cache-hit with “y”)

index(L,y) > 0
L’ == inc_counter(L,y)

STORE u,z @ miss→z0
(cache-miss with “z”)

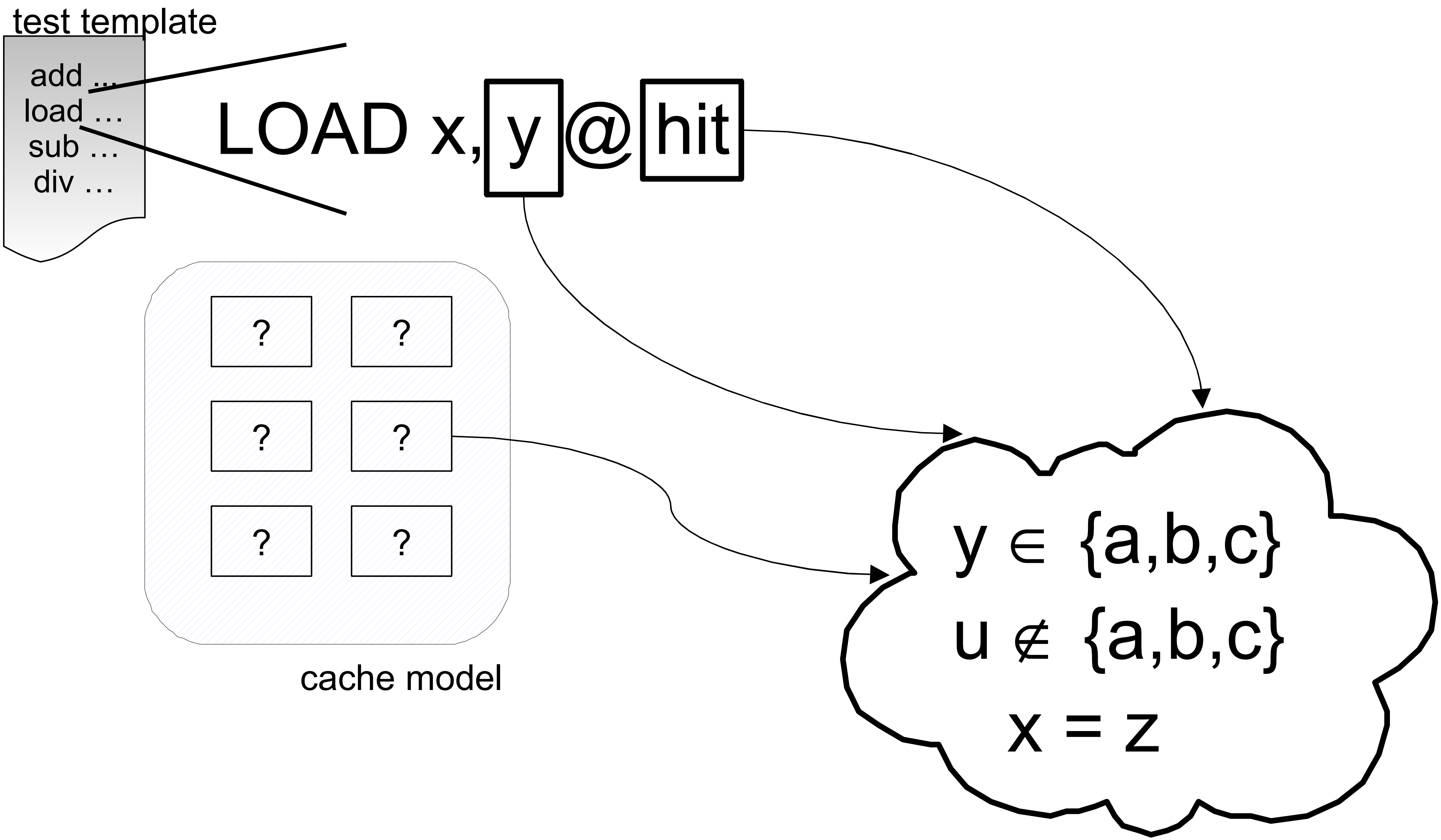
index(L’,z) == 0
counter(L’,z0) → min
L” == remove(L’,z0)
L”’ == add(L”,z)

LOAD z, y @ hit
(cache-hit with “y”)

index(L”’,y) > 0

x, y, z, u, L = ?

Key Idea



test template

add ...
load …
sub …
div …

LOAD x, y @ hit

? ?
? ?
? ?

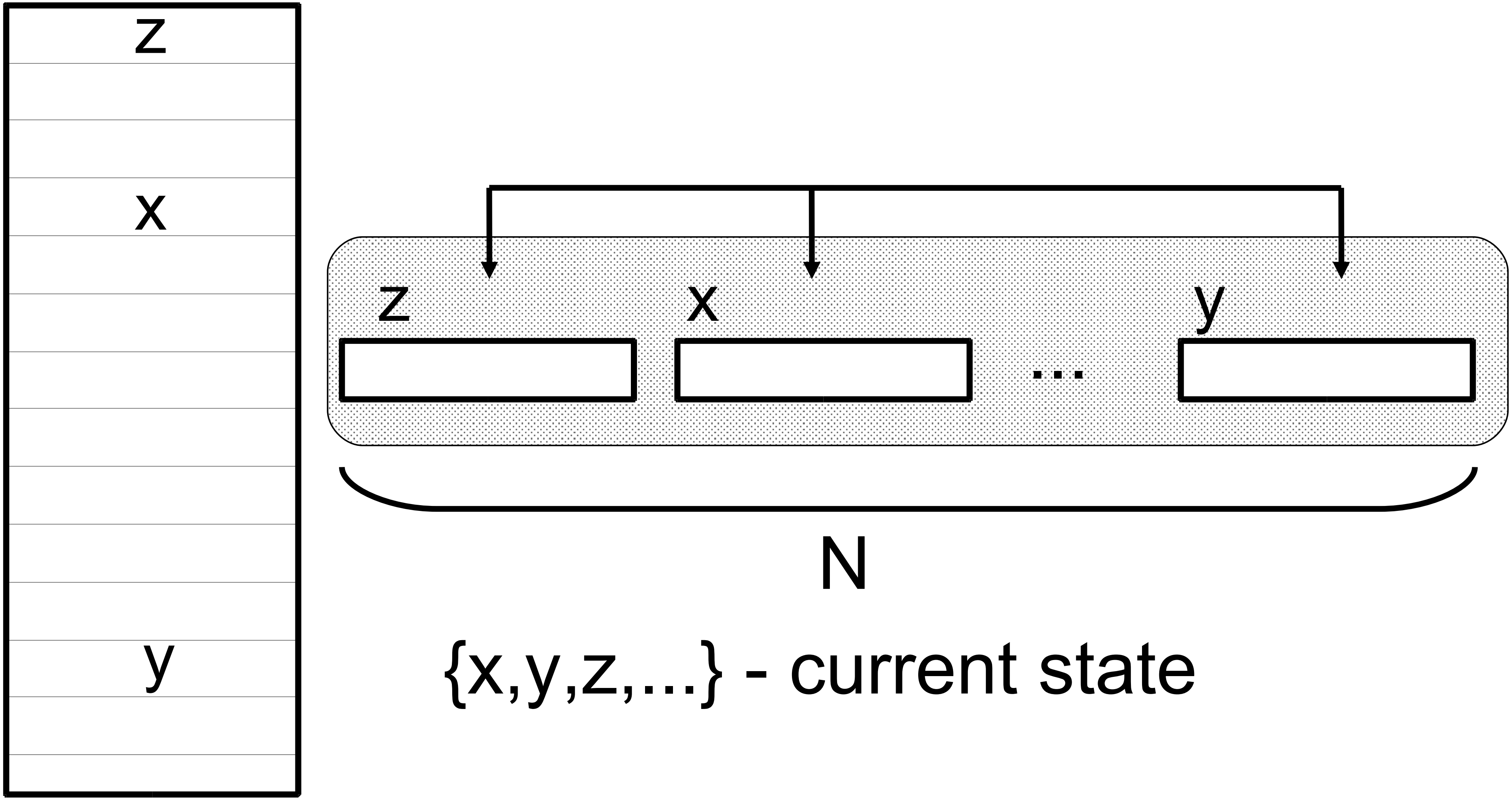
cache model

$y \in \{a,b,c\}$

$u \notin \{a,b,c\}$

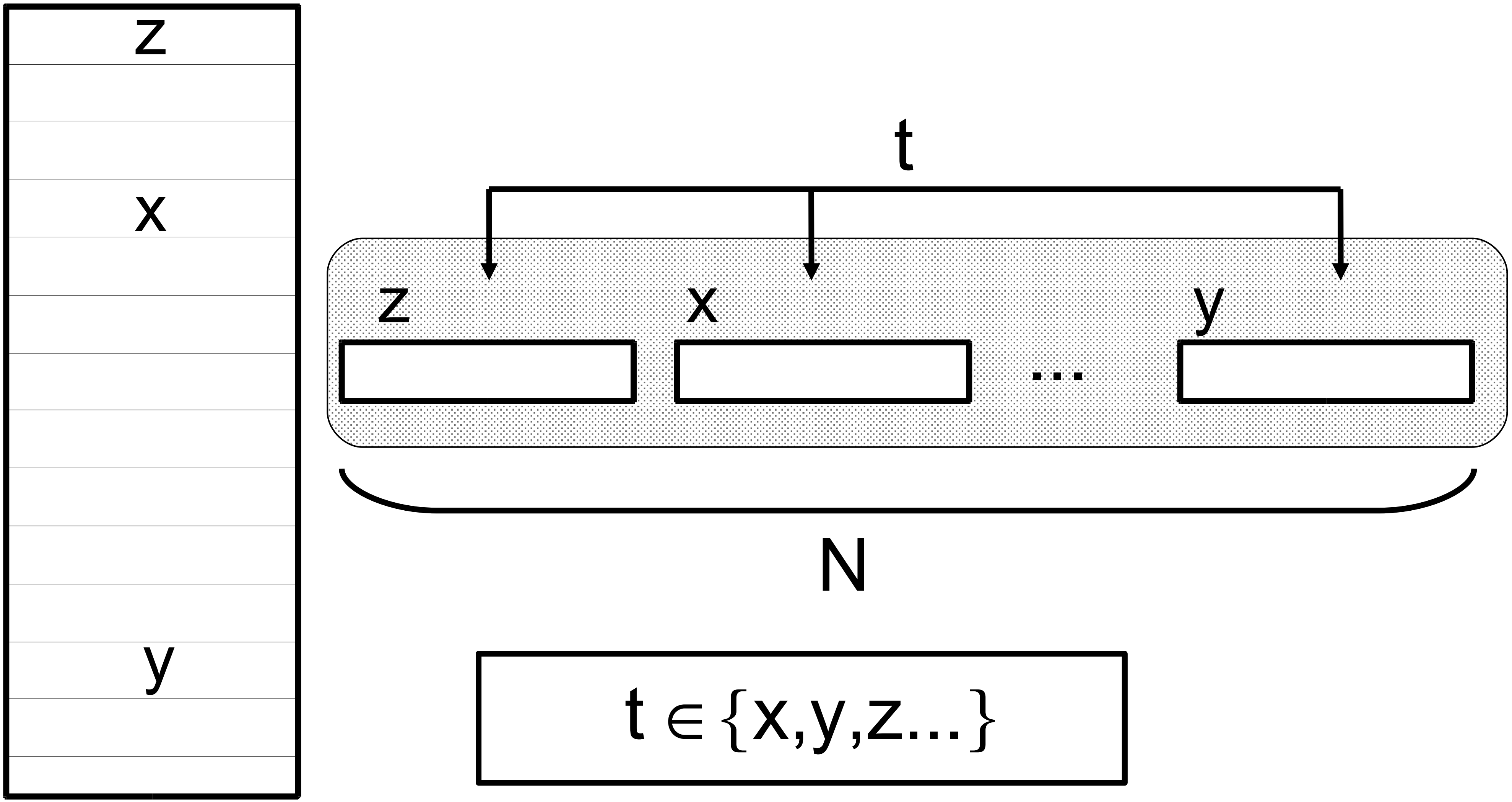
$x = z$

Fully Associative Cache



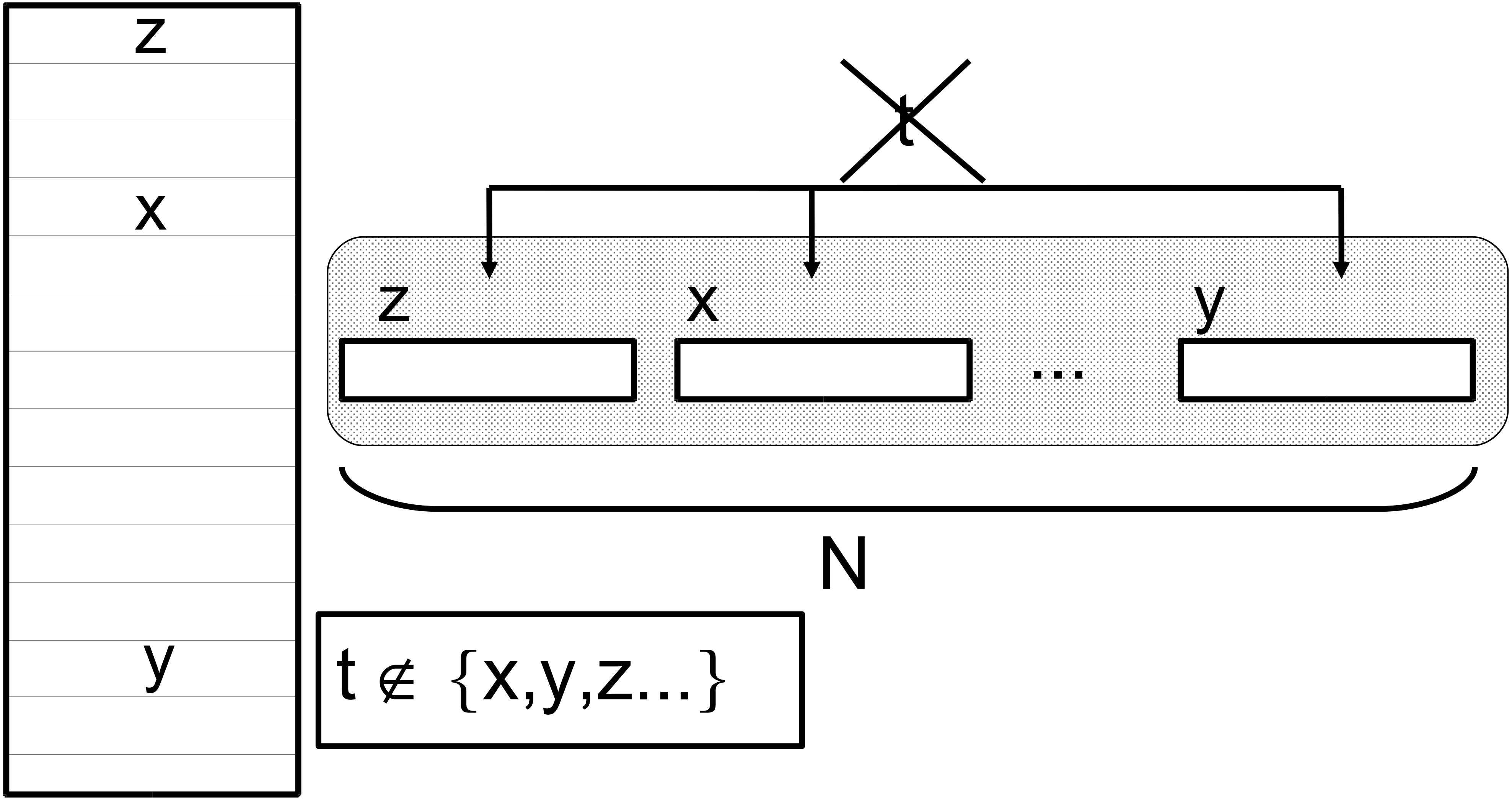
{x,y,z,...} - current state

Cache-hit hit(t)



$t \in \{x,y,z...\}$

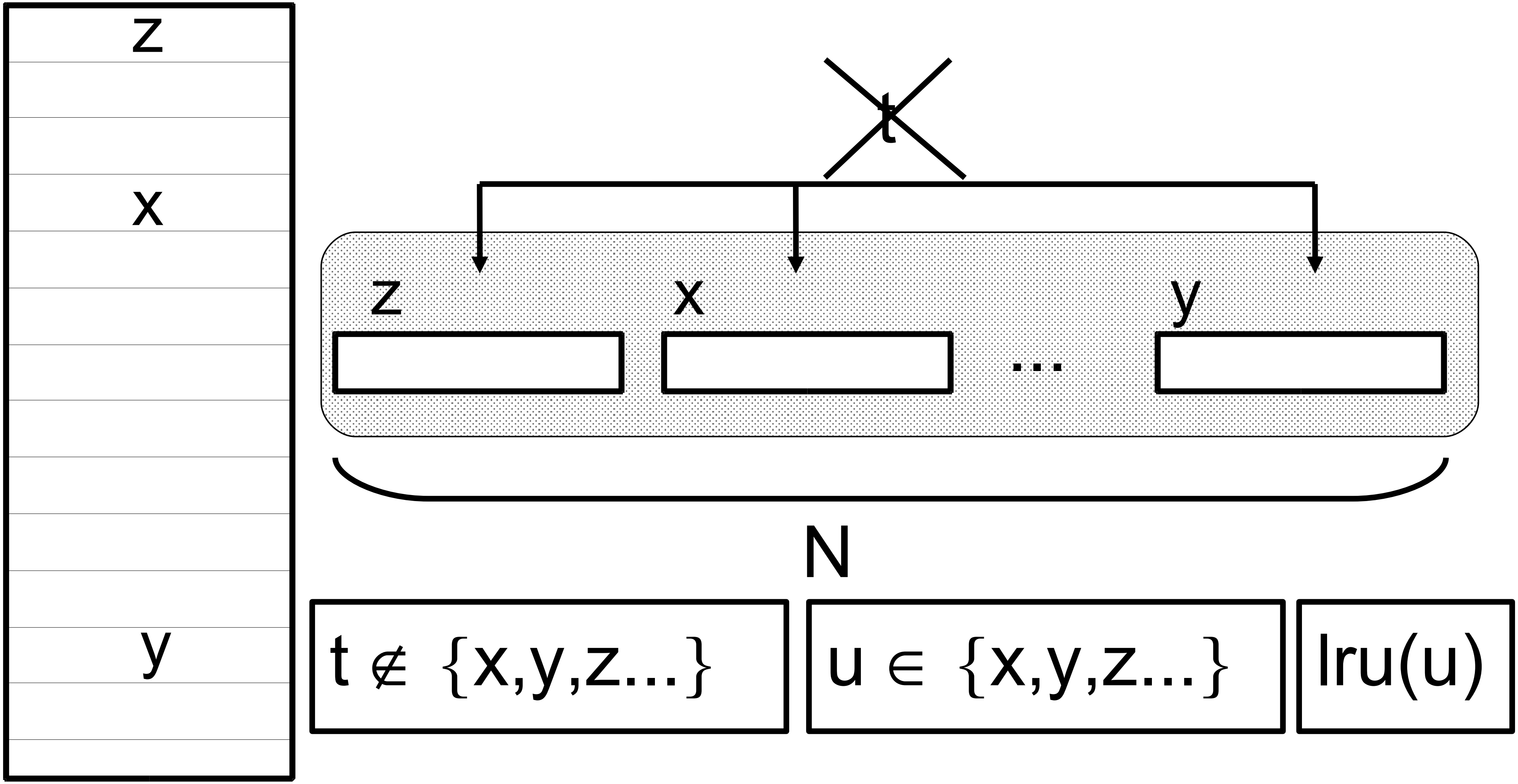
Cache-miss miss(t)



t ∉ {x,y,z...}

new cache {x,y,z...}∪{t}\{?}

Cache-miss miss(t)→u



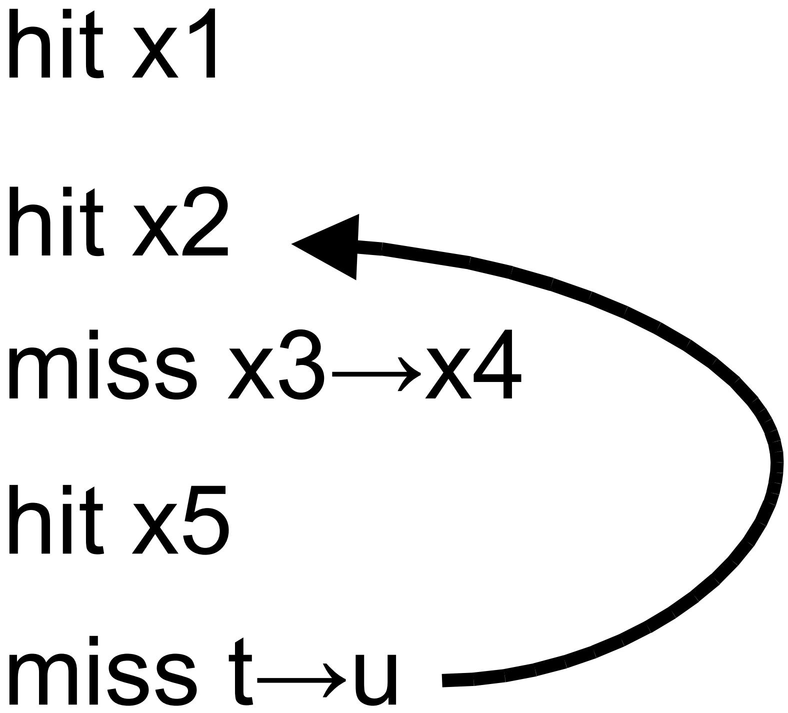
$t \notin \{x,y,z...\}$

$u \in \{x,y,z...\}$

lru(u)

new cache $\{x,y,z...\} \cup \{t\} \setminus \{u\}$

lru(u)



hit x1

hit x2

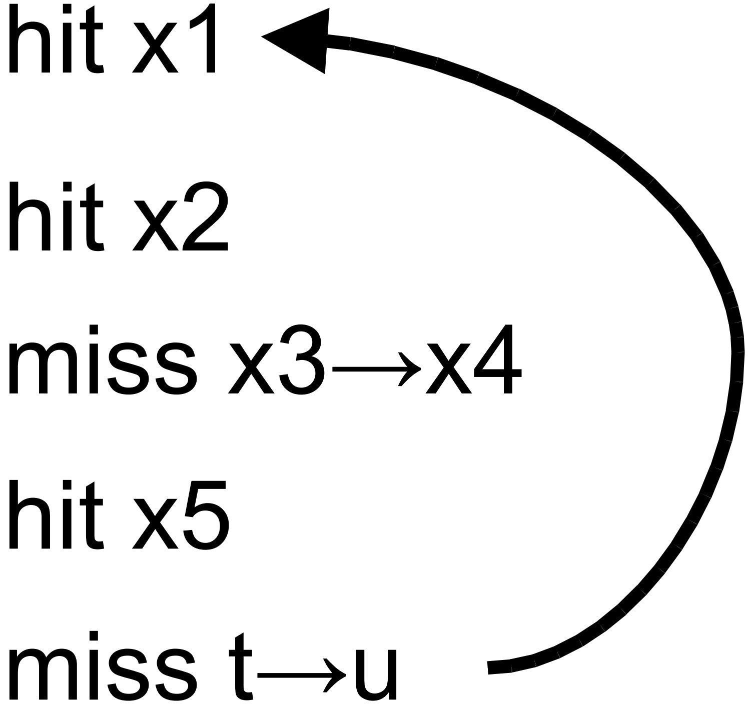
miss x3→x4

hit x5

miss t→u

$$\begin{cases} u = x2 \\ \{x3, x5\} = L\setminus\{u\} \end{cases}$$

lru(u)



hit x1

hit x2

miss x3→x4

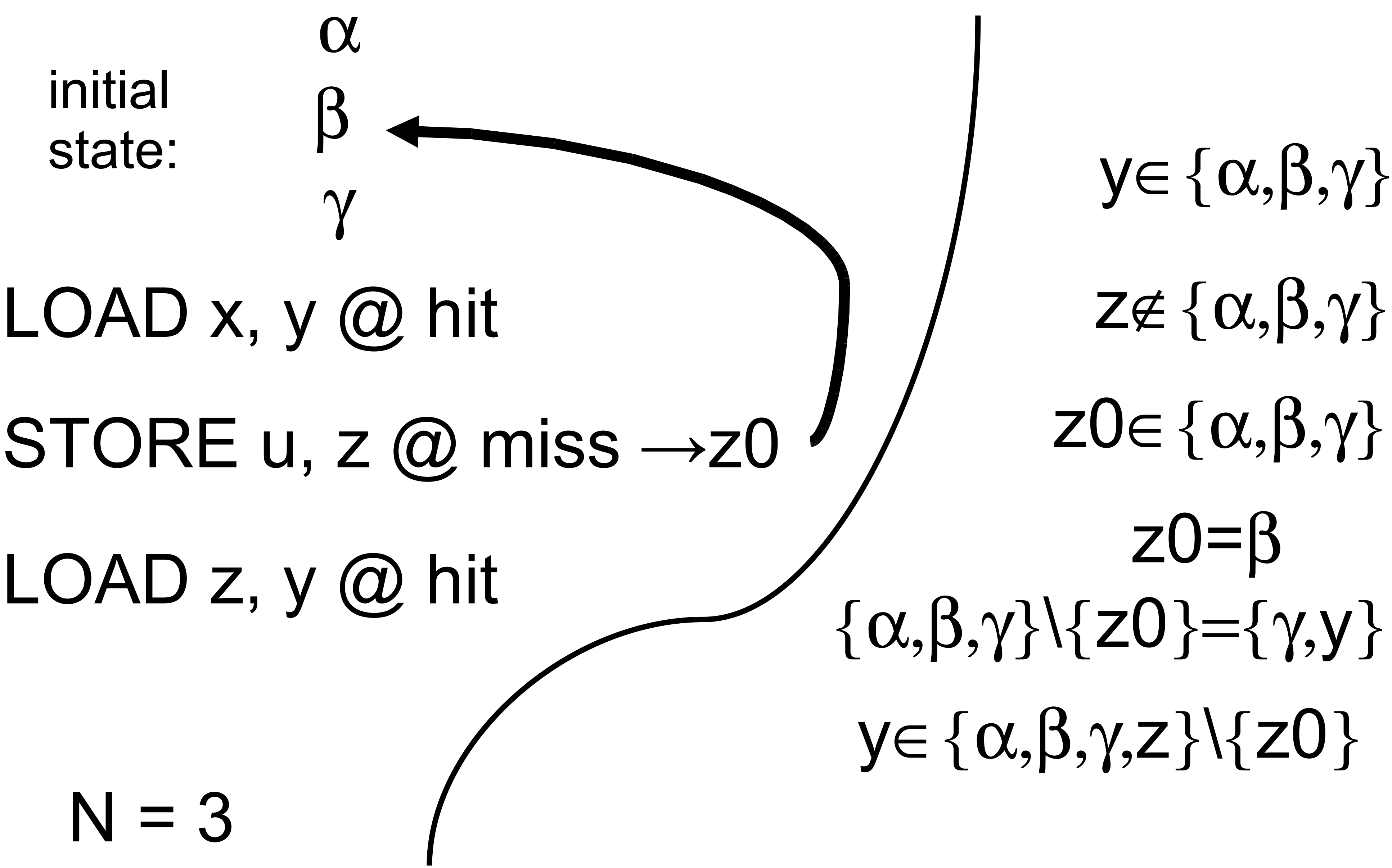
hit x5

miss t→u

$$\begin{cases} u = x1 \\ \{x2, x3, x5\} = L\setminus\{u\} \end{cases}$$

there are another cases..

Example



initial state:

α

β

γ

LOAD x, y @ hit

STORE u, z @ miss →z0

LOAD z, y @ hit

N = 3

$y \in \{\alpha,\beta,\gamma\}$

$z \notin \{\alpha,\beta,\gamma\}$

$z0 \in \{\alpha,\beta,\gamma\}$

$z0=\beta$

$\{\alpha,\beta,\gamma\}\setminus\{z0\}=\{\gamma,y\}$

$y \in \{\alpha,\beta,\gamma,z\}\setminus\{z0\}$

Example

$y = \alpha$

$z \notin \{\alpha,\beta,\gamma\}$

$y=\alpha=0$

$\beta=1$

$\gamma=2$

$z=3$

$y \in \{\alpha,\beta,\gamma\}$

$z \notin \{\alpha,\beta,\gamma\}$

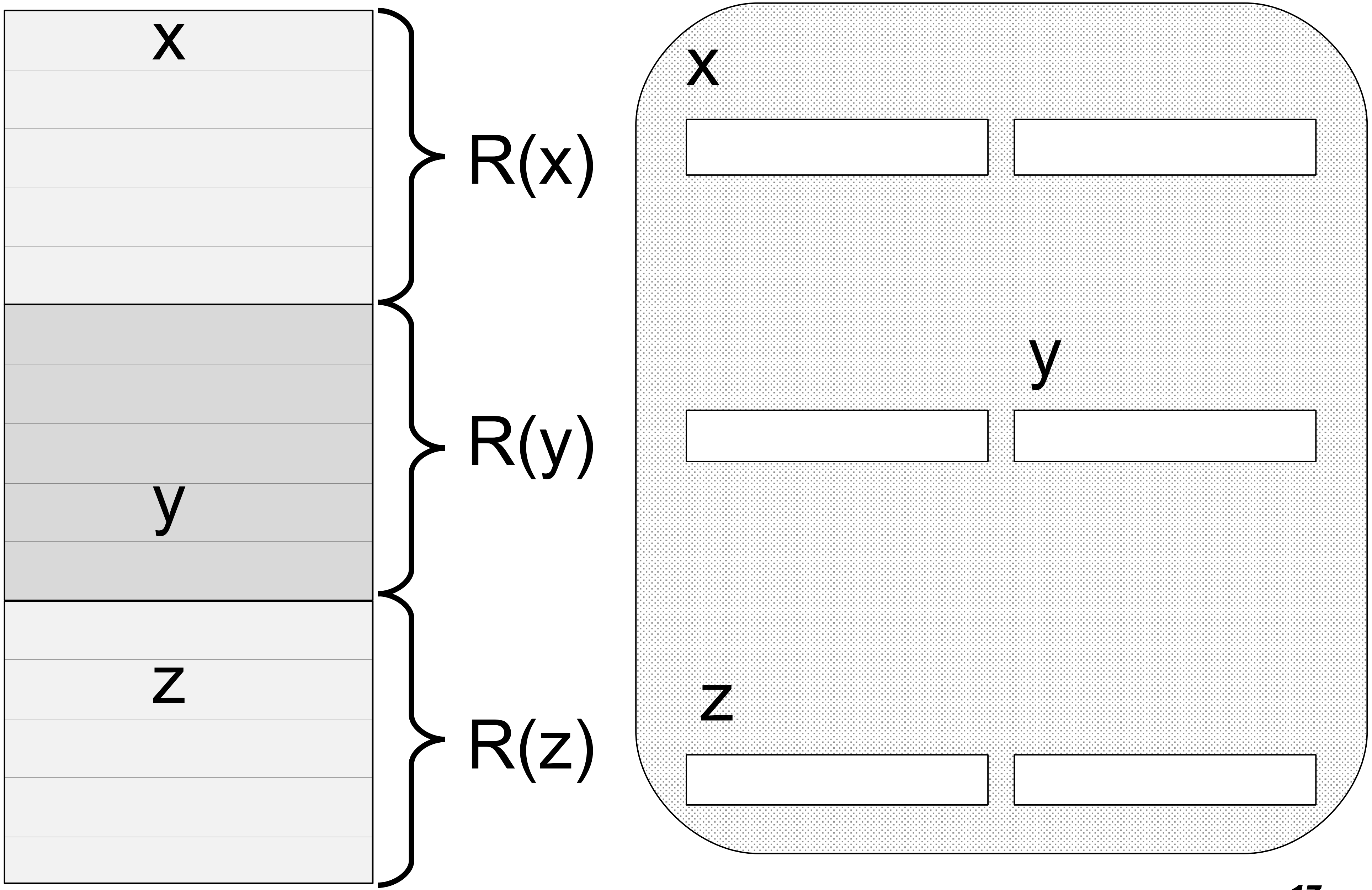
$z0 \in \{\alpha,\beta,\gamma\}$

$z0=\beta$

$\{\alpha,\beta,\gamma\}\backslash\{z0\}=\{\gamma,y\}$

$y \in \{\alpha,\beta,\gamma,z\}\backslash\{z0\}$

Common Cache



Common Cache

hit(t) $\boxed{t \in L}$

miss(t)→u

$u \in L$

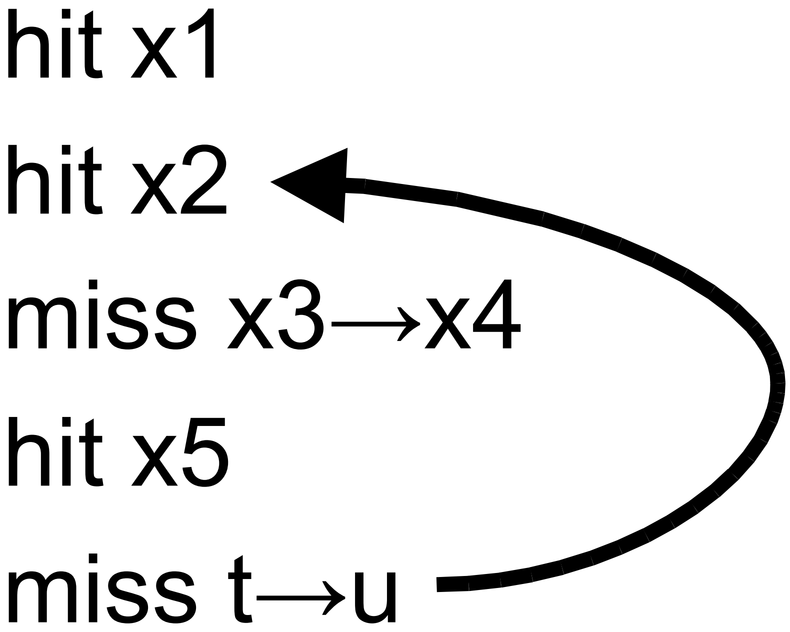
$t \notin L$

$L \cup \{t\} \setminus \{u\}$ is the next cache

$R(t) = R(u)$

lru(u)

lru(u)



hit x1

hit x2

miss x3→x4

hit x5

miss t→u

$$\begin{cases} u = x2 \\ \{x3, x5\} \cap R(u) \\ \quad = (L \setminus \{y\}) \cap R(u) \end{cases}$$

Example

$x1,x2 \in \{a1,a2,b1,b2,c1,c2\}$

$x3 \notin \{a1,a2,b1,b2,c1,c2\}$

$R(x3) = R(y3)$

$x4 \in \{a1,...,c2,x3\}\backslash\{y3\}$

$x5 \notin \{a1,...,c2,x3\}\backslash\{y3\}$

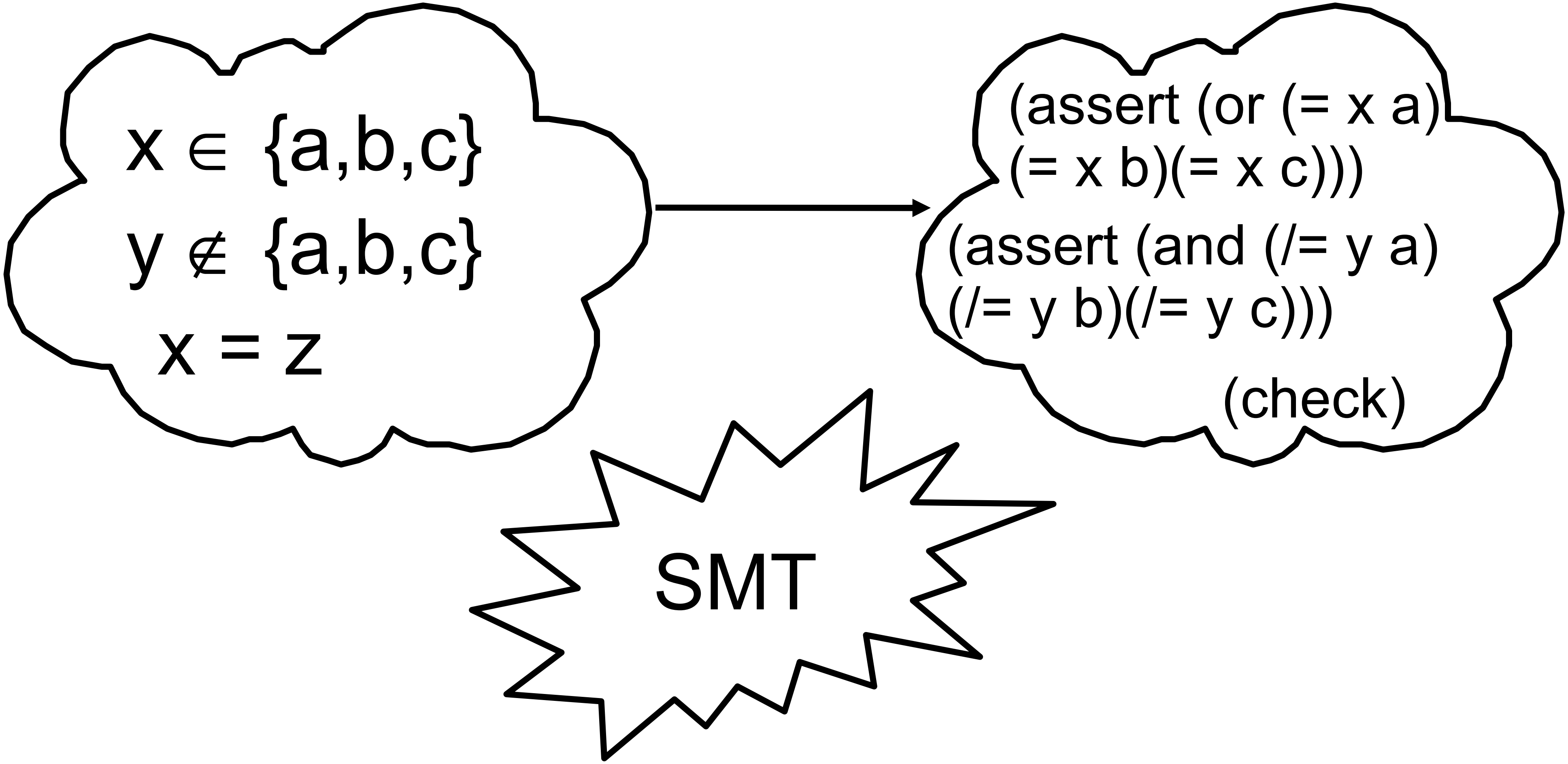
$y3 = c2$

$\{y3\} = (\{a1,...,c2\}\backslash\{x1,x2, y3\})\cap R(y3)$

$y5 = x2$

$\{y5\} = (\{a1...c2,x3\}\backslash\{y3,y5, x3,x4\})\cap R(y5)$

Solver



$x \in \{a,b,c\}$

$y \notin \{a,b,c\}$

$x = z$

```
(assert (or (= x a)
(= x b)(= x c)))
(assert (and (/= y a)
(/= y b)(/= y c)))
(check)
```

SMT

SAT modulo theories

Yices

Conclusions

- Cache behavior can be expressed using equations on addresses’ set
- Equations can be solved by SMT-solver

Contacts

http://tesla-project.googlecode.com

http://hardware.ispras.ru

kornevgen@gmail.com

Thank you for attention!