Quel temps d'exécution pour mon programme dans le pire des cas ?

Système temps-réel

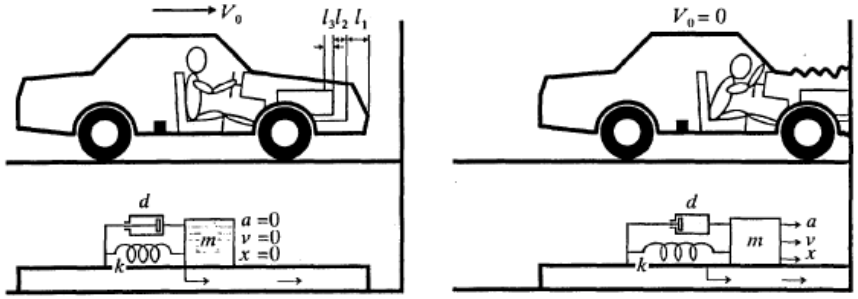
Un système soumis à des contraintes temporelles

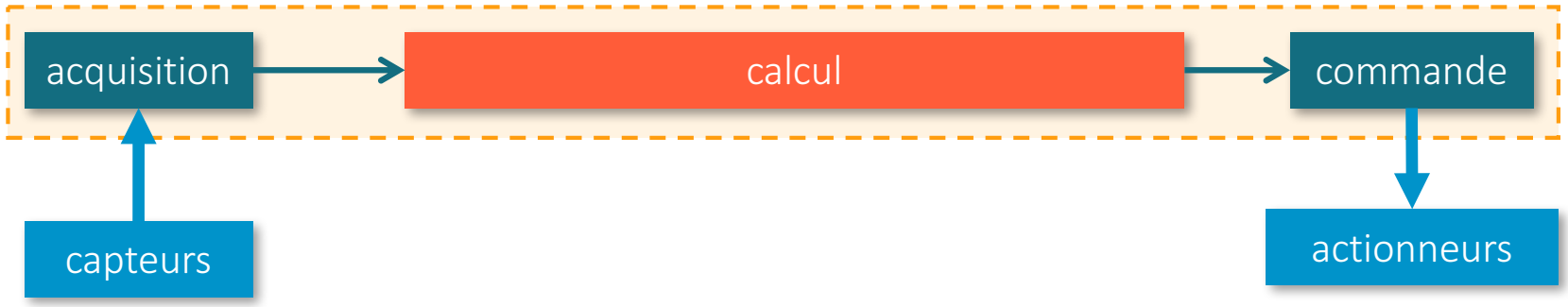
- par exemple, sur le temps qui sépare un événement de la réponse du système
- la validité du système repose autant sur le respect des contraintes de temps (*deadlines*) que sur la correction du résultat des calculs

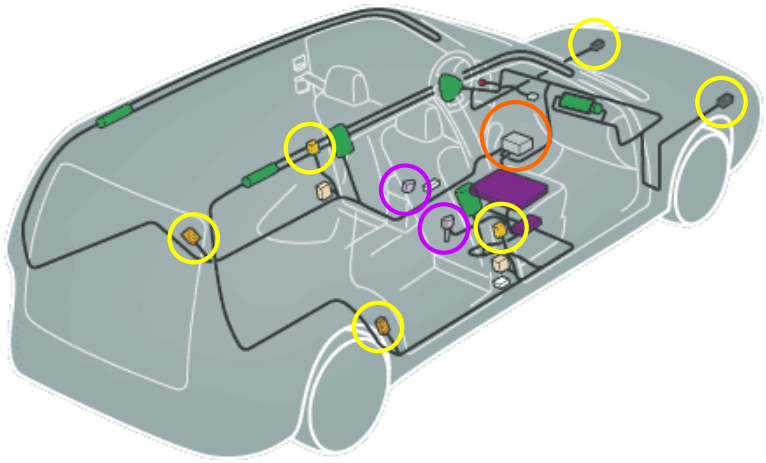
Temps-réel ≠ rapide

- ce qui importe, c'est d'être capable d'estimer (borner) le temps de réponse

Exemple : contrôle d'un airbag

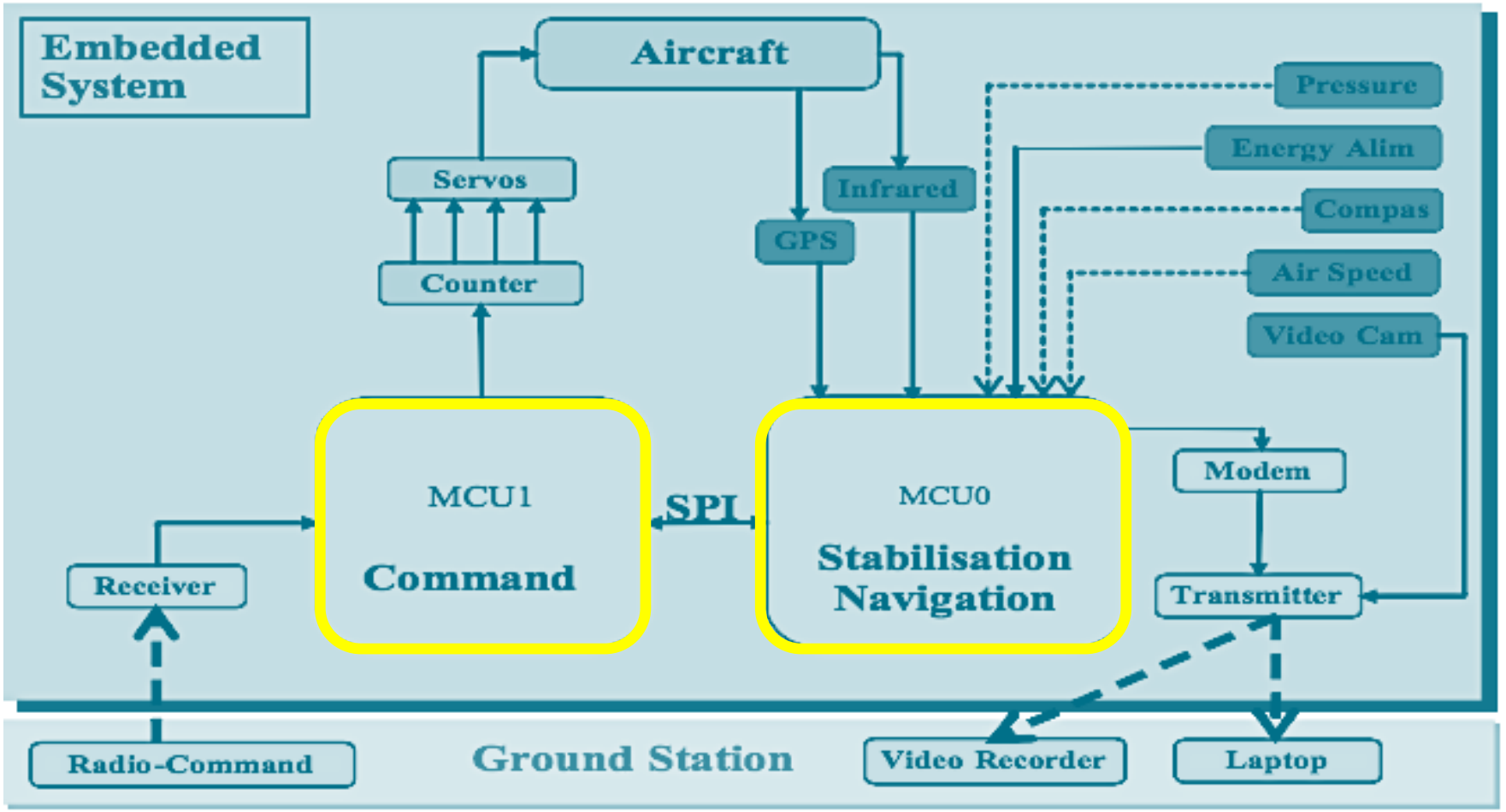






Autre exemple : contrôle d'un drone

PapaBench (inspiré du projet Paparazzi de l'ENAC)



Autre exemple : contrôle d'un drone

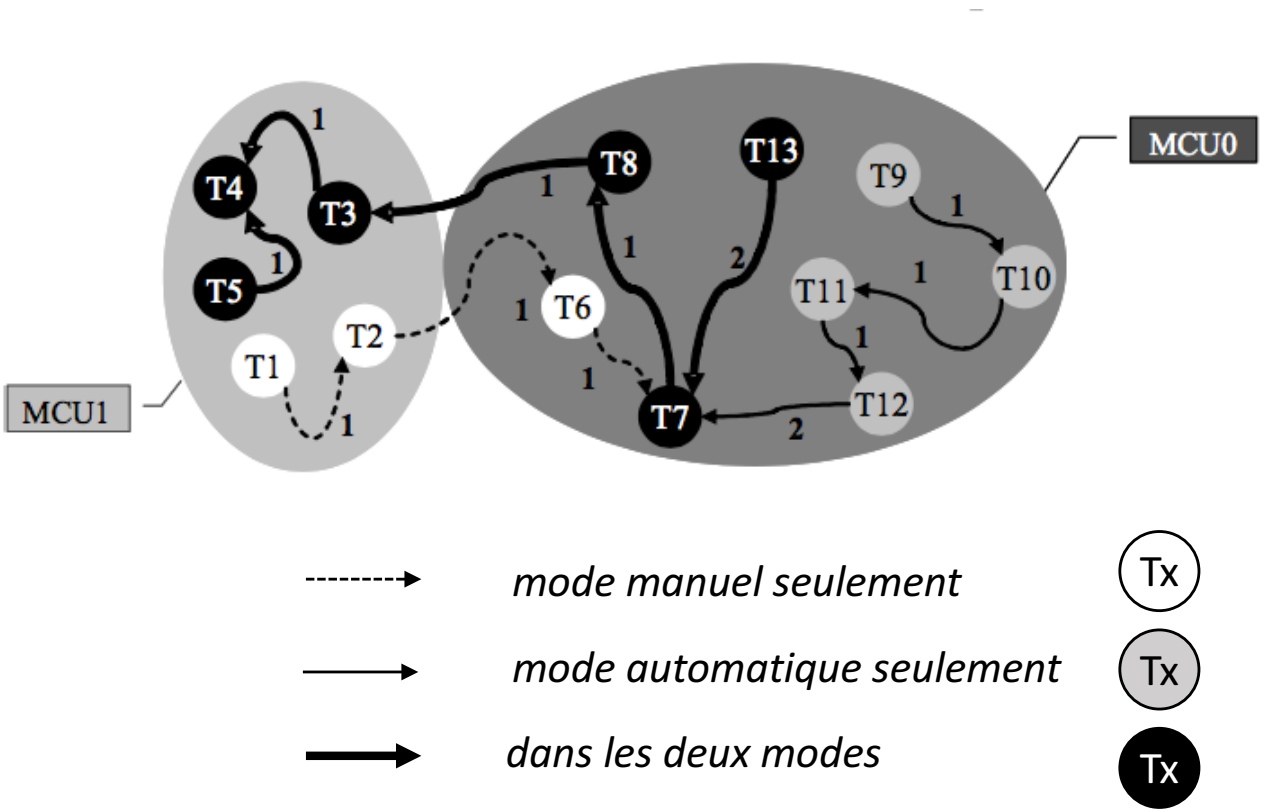
PapaBench (inspiré du projet Paparazzi de l'ENAC)

ID	*Description*	*Frequency*
T1	Receive Radio-Command orders	40Hz
T2	Send Data to MCU0	40Hz
T3	Receive MCU0 values	20Hz
T4	Transmit Servos	20Hz
T5	Check Failsafe	20Hz
I1	Transmission Servos interrupt	-
I2	SPI interrupt of MCU1	-
I3	Radio interrupt	-

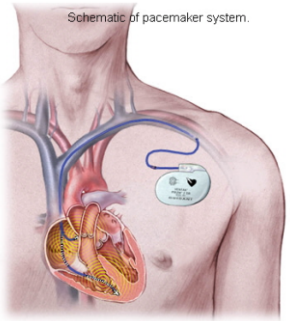
Table 1: *MCU1 tasks and interrupts*

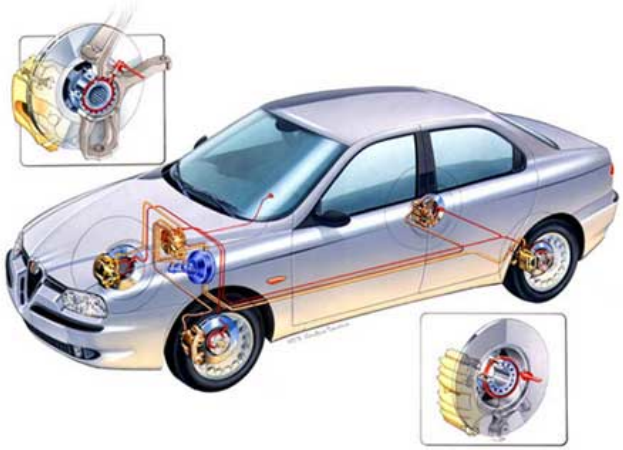
ID	*Description*	*Frequency*
T6	Managing Radio orders	40Hz
T7	Stabilization	20Hz
T8	Send Data to MCU1	20Hz
T9	Receive GPS Data	4Hz
T10	Navigation	4Hz
T11	Altitude Control	4Hz
T12	Climb Control	4Hz
T13	Reporting Task	10Hz
I4	SPI interrupt of MCU0	-
I5	Modem interrupt	-
I6	GPS interrupt	-

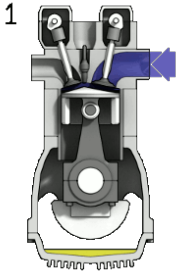
Table 2: *MCU0 tasks and interrupts*

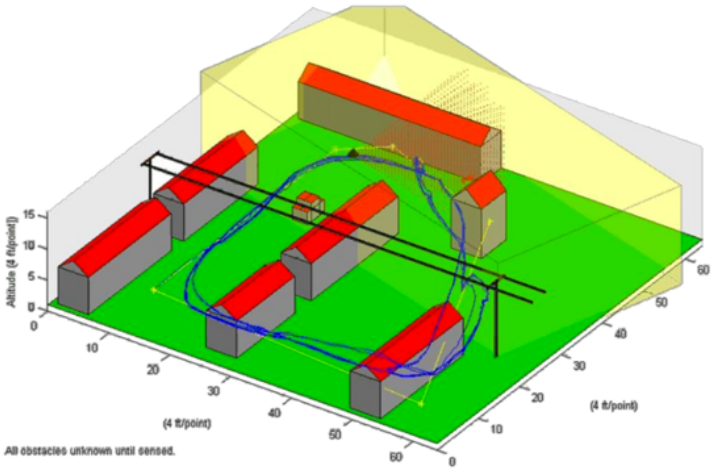


Autres exemples





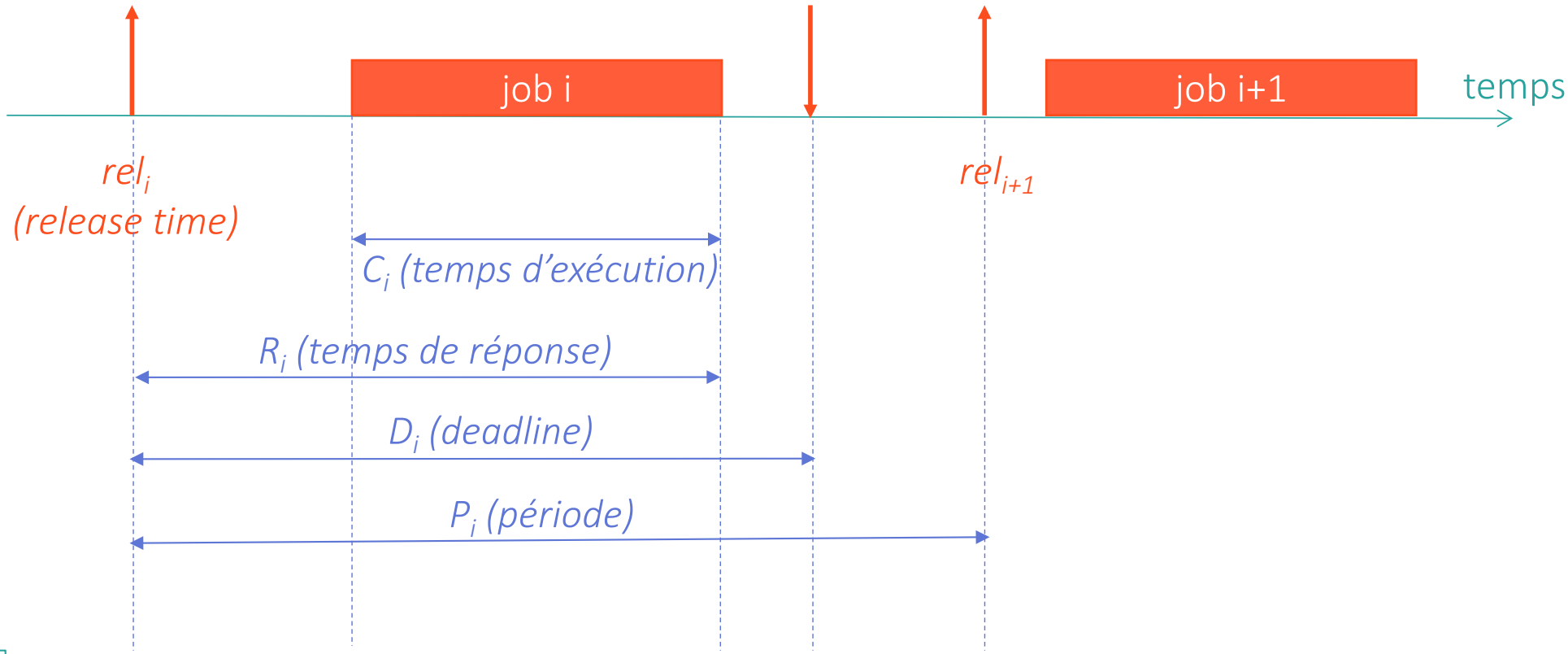




Ordonnancement temps-réel

Modèle de tâche

- périodique, sporadique, apériodique



Ordonnancement temps-réel

Objectif

- allouer des ressources partagées aux tâches de sorte qu'elles puissent toutes respecter leurs échéances

Approches

- hors-ligne ou en-ligne
- priorités fixes ou dynamiques
  - Rate Monotonic, Deadline Monotonic, Earliest Deadline First
- préemptif ou pas
- sur une plateforme multicoeurs, global ou partitionné

Chetto et al.
Ordonnancement temps-réel, ISTE Wiley, 2014

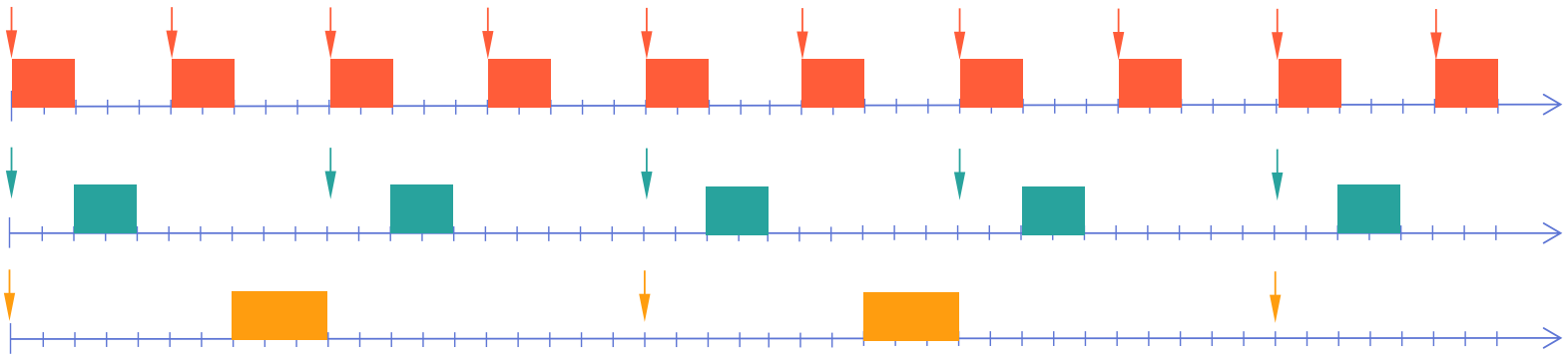
Test d'ordonnançabilité

Exemple : EDF (Earliest Deadline First) non préemptif

$$\sum_{i=1}^{n} \frac{C_i}{P_i} \leq 1$$

$$C_i + \sum_{j=1}^{i-1} \left\lceil \frac{L-1}{P_j} \right\rceil \times C_j \leq L \quad avec \quad 1 < i \leq n \quad et \quad P_1 < L \leq P_i$$

	C_i	$P_i=D_i$
τ_1	2	5
τ_2	2	10
τ_3	3	20



De quoi dépend le temps d'exécution d'une tâche ?

- des données en entrée

```
index = 0;
while (data_in[index] != 0){
     /* processing */
}
```

combien d'itérations ?

```
if (temperature > T_MAX){
     /* error processing */
}
else{
     /* normal processing */
}
```

quelle branche ?

- de l'état initial du processeur

```
sum = 0;
for (i=0 ; i<100 ; i++)
     sum += a[i];
```

que contient le cache ?

- des interférences avec d'autres tâches

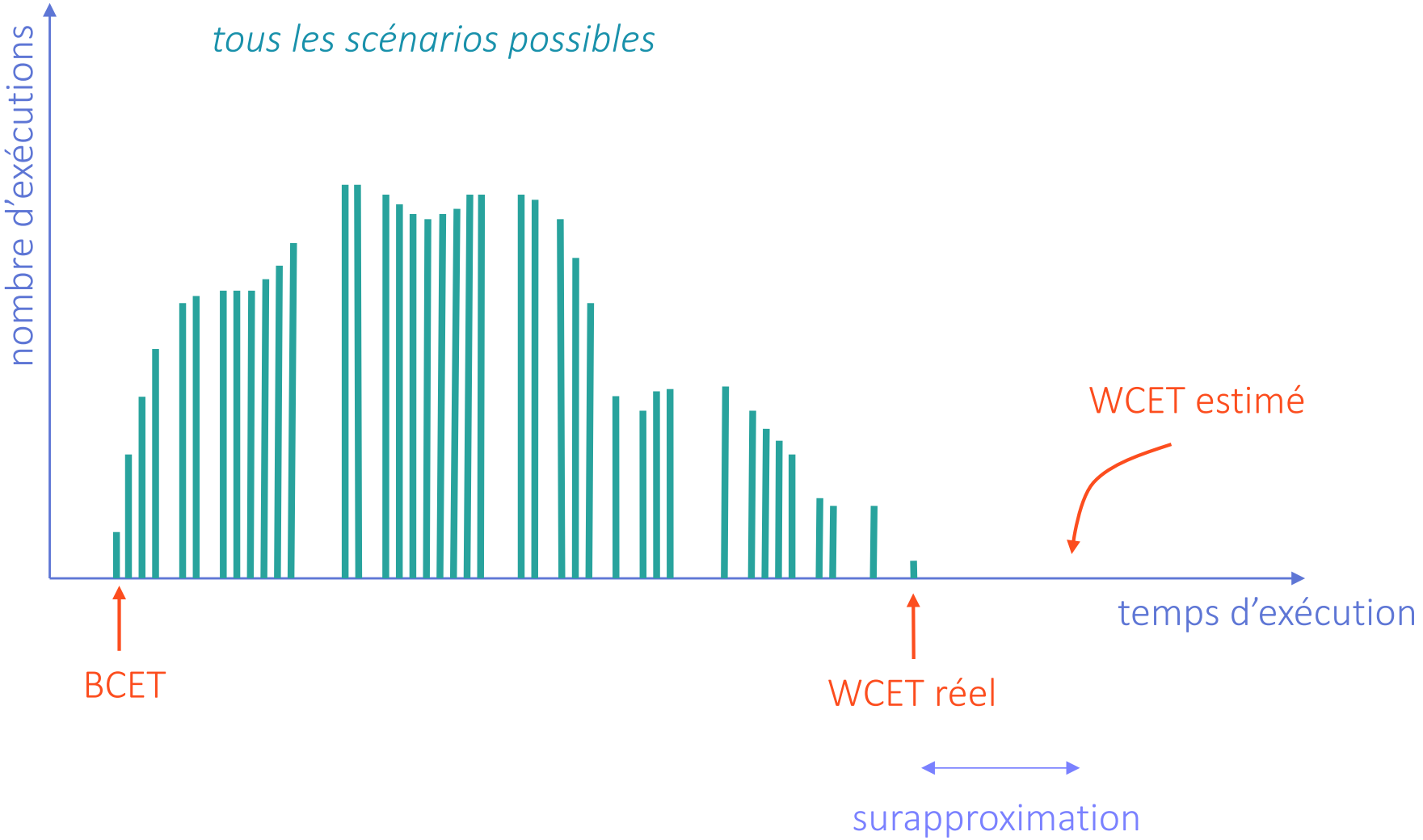
sur un défaut de cache, quelle latence d'accès à la memoire ?

Temps d'exécution pire cas

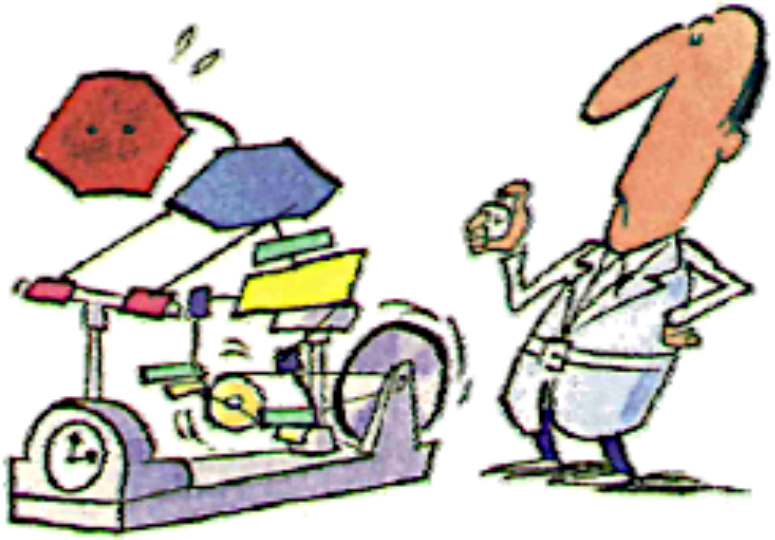
Worst-Case Execution Time (WCET)

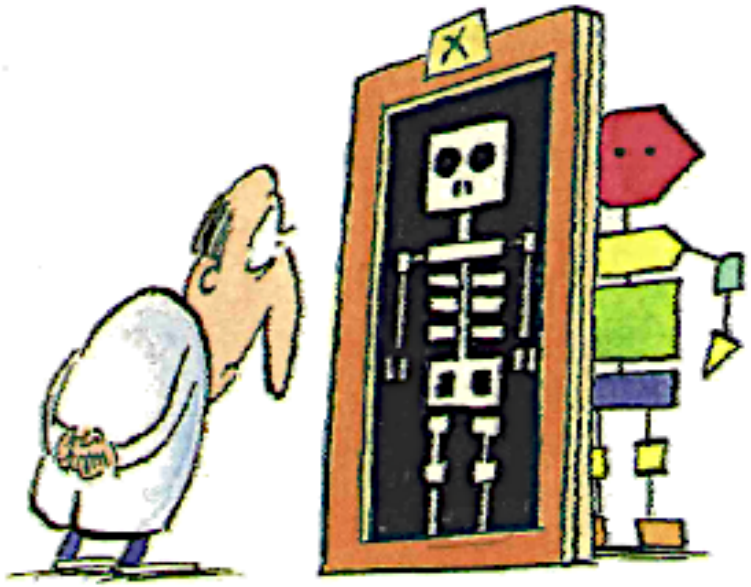
- valeur maximale que peut prendre le temps d'exécution
  - quelles que soient les données en entrée
  - quel que soit l'état initial du matériel
  - quelles que soient les interférences avec d'autres tâches

Temps d’exécution pire cas

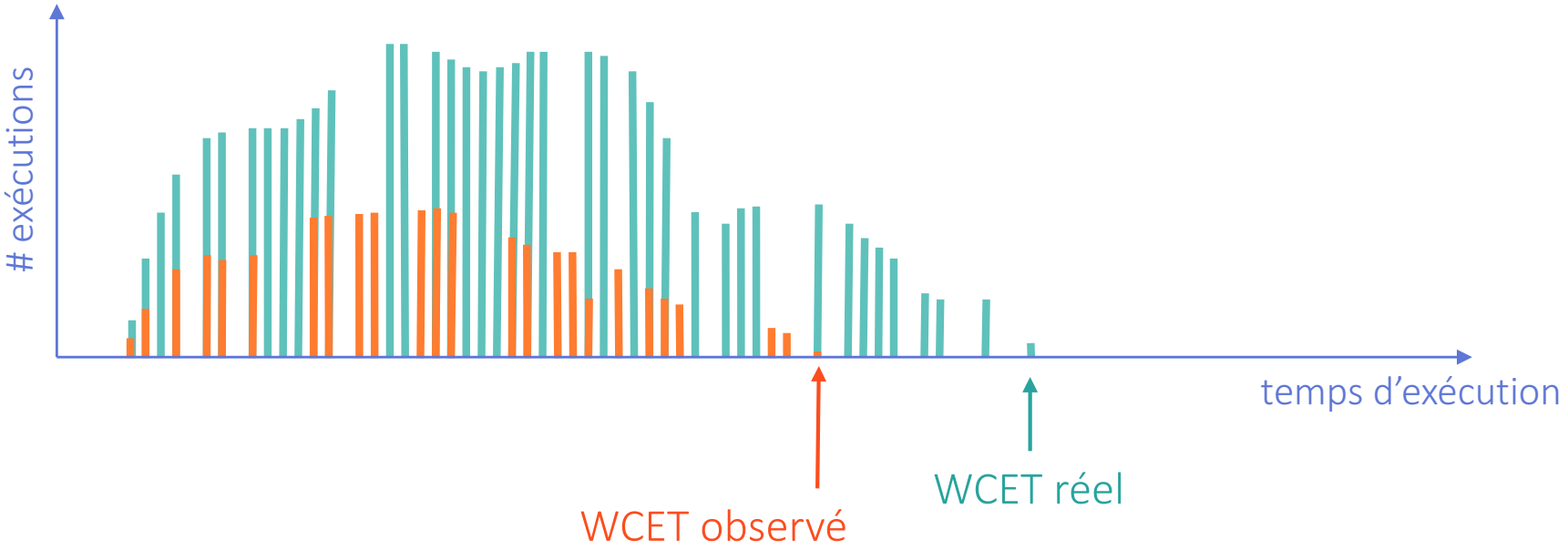


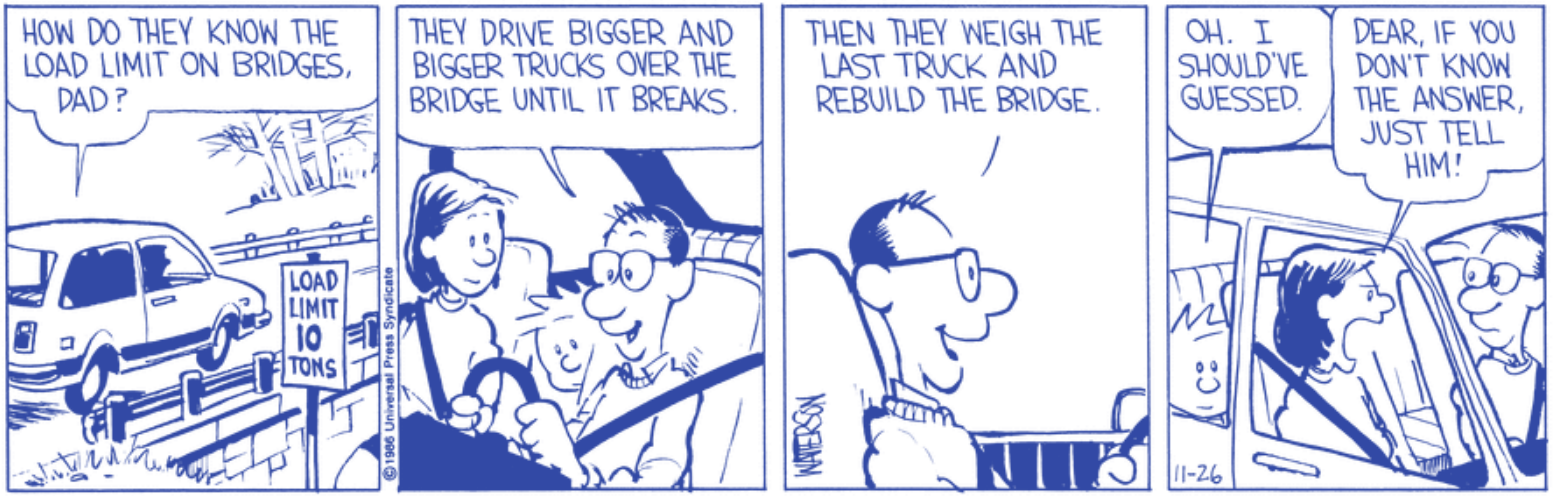
Comment déterminer le WCET ?





Mesures

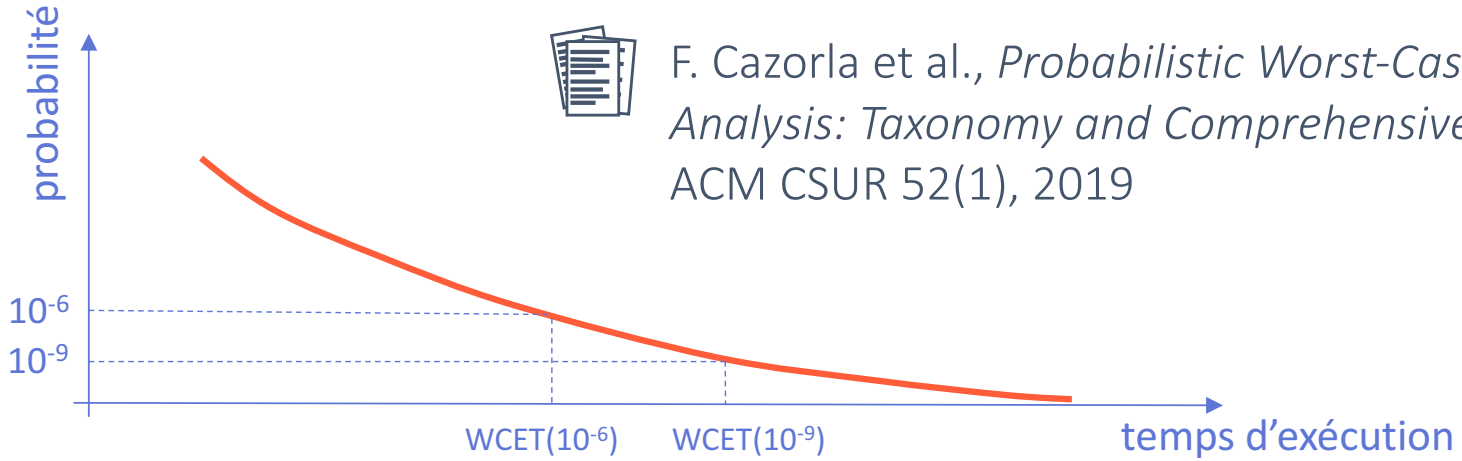

exécutions
temps d'exécution
WCET observé
WCET réel


HOW DO THEY KNOW THE LOAD LIMIT ON BRIDGES, DAD?
LOAD LIMIT 10 TONS
THEY DRIVE BIGGER AND BIGGER TRUCKS OVER THE BRIDGE UNTIL IT BREAKS.
© 1986 Universal Press Syndicate
THEN THEY WEIGH THE LAST TRUCK AND REBUILD THE BRIDGE.
WATTERSON
OH. I SHOULD'VE GUESSED.
DEAR, IF YOU DON'T KNOW THE ANSWER, JUST TELL HIM!
11-26

Analyse probabiliste

Idée de base

- calculer la probabilité d'une défaillance temporelle (le WCET dépasse une certaine valeur)



F. Cazorla et al., *Probabilistic Worst-Case Timing Analysis: Taxonomy and Comprehensive Survey*, ACM CSUR 52(1), 2019

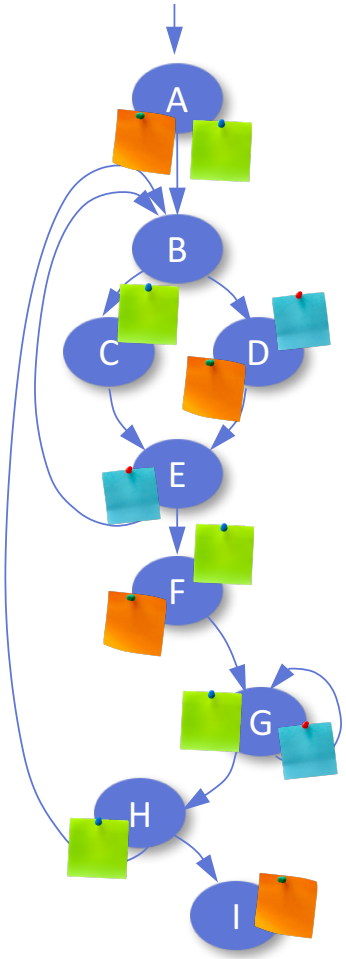
Techniques

- matériel « randomisé »
- analyses statistiques, théorie des valeurs extrêmes
- statique ou dynamique

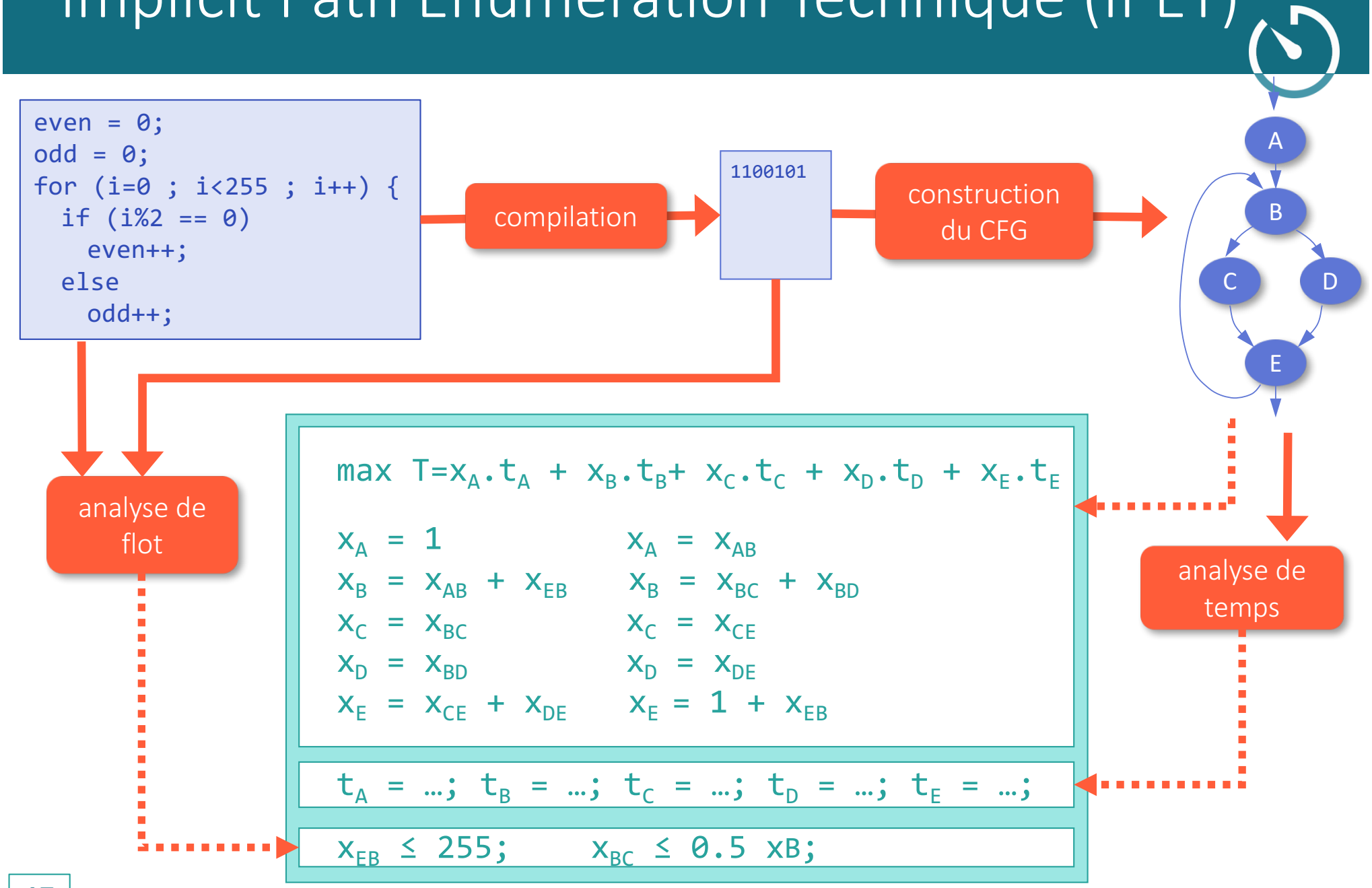
Analyse statique

Principe général

- graphe de flot de contrôle (CFG)
- « décoré » par une série d'analyses
- calcul des WCETs des blocs de base
- combinaison de ces WCETs pour déterminer le WCET global du programme



Implicit Path Enumeration Technique (IPET)



Implicit Path Enumeration Technique (IPET)

$$\max T = x_A.t_A + x_B.t_B + x_C.t_C + x_D.t_D + x_E.t_E$$

$$x_A = 1 \qquad x_A = x_{AB}$$
$$x_B = x_{AB} + x_{EB} \qquad x_B = x_{BC} + x_{BD}$$
$$x_C = x_{BC} \qquad x_C = x_{CE}$$
$$x_D = x_{BD} \qquad x_D = x_{DE}$$
$$x_E = x_{CE} + x_{DE} \qquad x_E = 1 + x_{EB}$$

$$t_A = \ldots; \; t_B = \ldots; \; t_C = \ldots; \; t_D = \ldots; \; t_E = \ldots;$$

$$x_{EB} \leq 255; \qquad x_{BC} \leq 0.5 \; xB;$$

solveur ILP

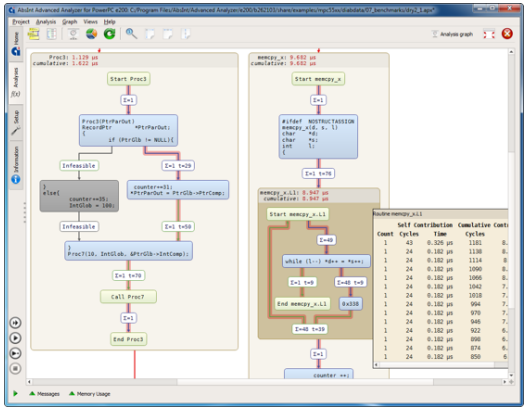
$$\max T = \ldots$$
$$x_A = 1 \qquad x_{AB} = \ldots$$
$$x_B = \ldots \qquad x_{BC} = \ldots \qquad x_{BD} = \ldots$$
$$x_C = \ldots \qquad x_{CE} = \ldots$$
$$x_D = \ldots \qquad x_{DE} = \ldots$$
$$x_E = \ldots \qquad x_{EB} = \ldots$$

Li and Malik, *Performance Analysis of Embedded Software using Implicit Path Enumeration, 1995*

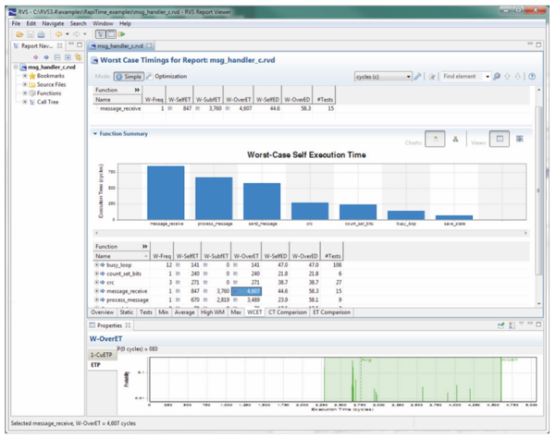
Outils pour l'analyse de WCET

Outils commercialisés :

- aiT AbsInt

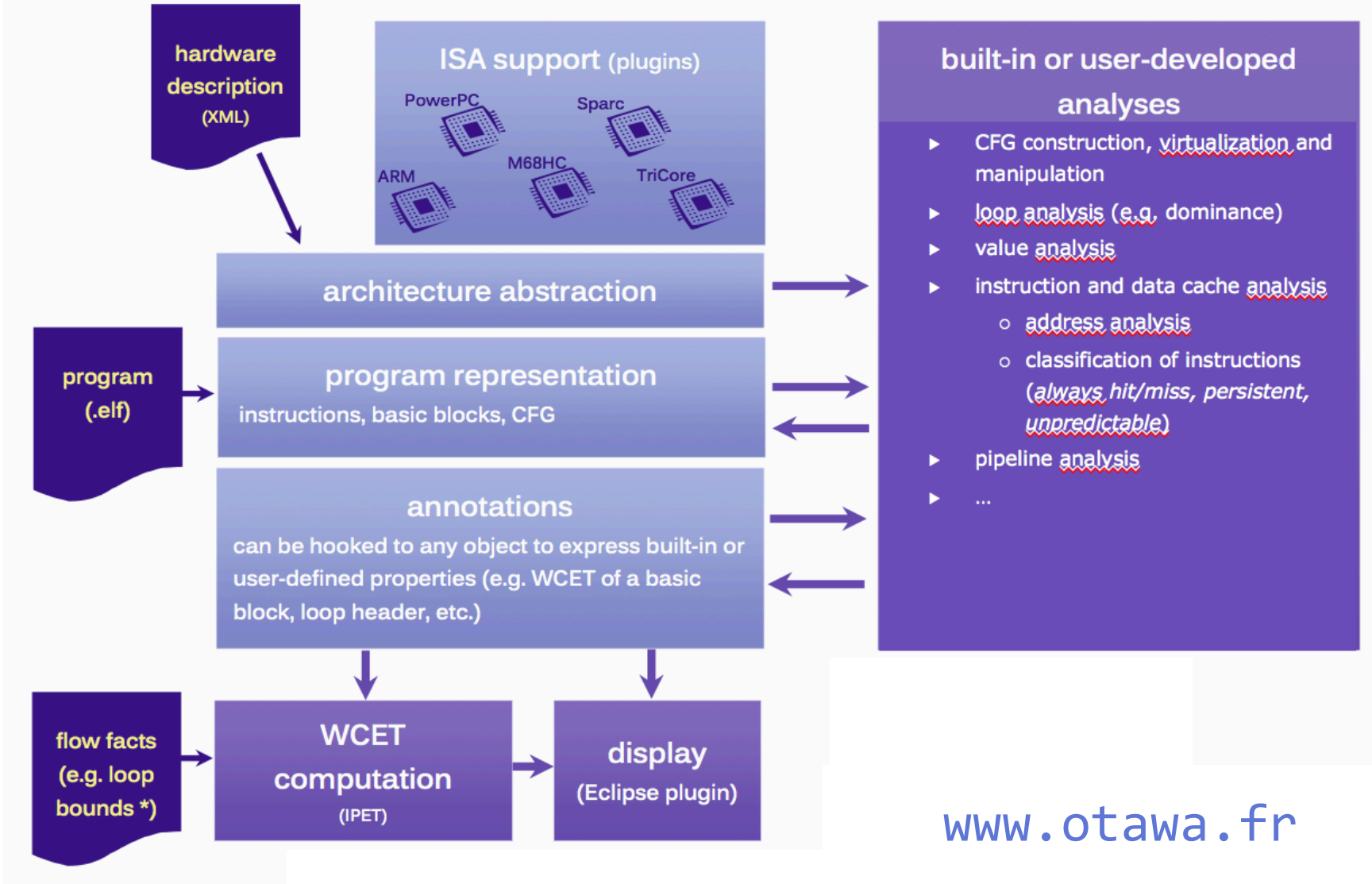


- RapiTime RAPITA Systems Ltd

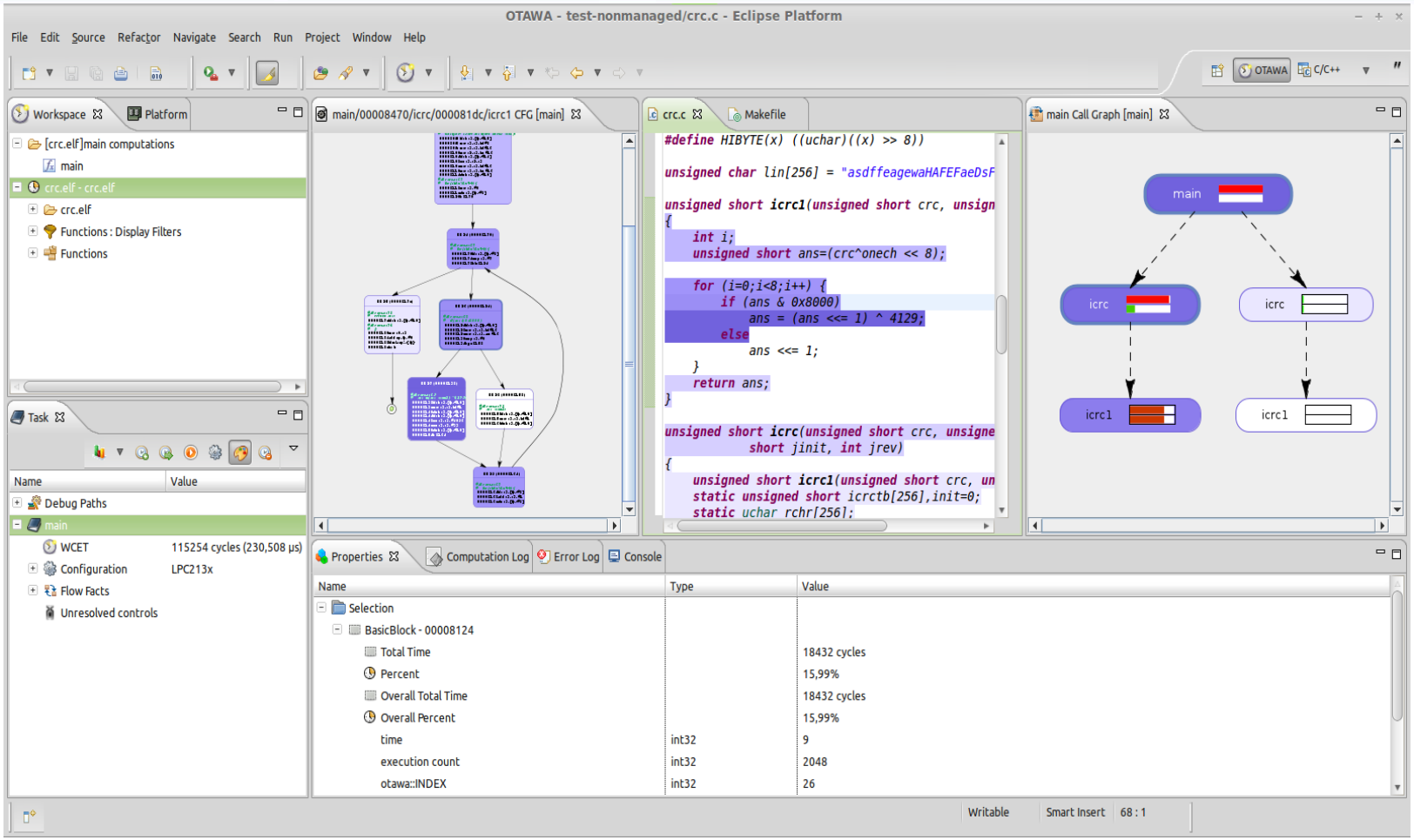


Outils académiques :

- Chronos (Singapour)
- Heptane (Rennes)
- OTAWA (Toulouse)
- SWEET (Mälardalen)


hardware description (XML)
ISA support (plugins)
PowerPC
Sparc
ARM
M68HC
TriCore
architecture abstraction
program (.elf)
program representation
instructions, basic blocks, CFG
annotations
can be hooked to any object to express built-in or user-defined properties (e.g. WCET of a basic block, loop header, etc.)
flow facts (e.g. loop bounds *)
WCET computation
(IPET)
display
(Eclipse plugin)
built-in or user-developed analyses
CFG construction, virtualization and manipulation
loop analysis (e.g. dominance)
value analysis
instruction and data cache analysis
address analysis
classification of instructions (always hit/miss, persistent, unpredictable)
pipeline analysis
...

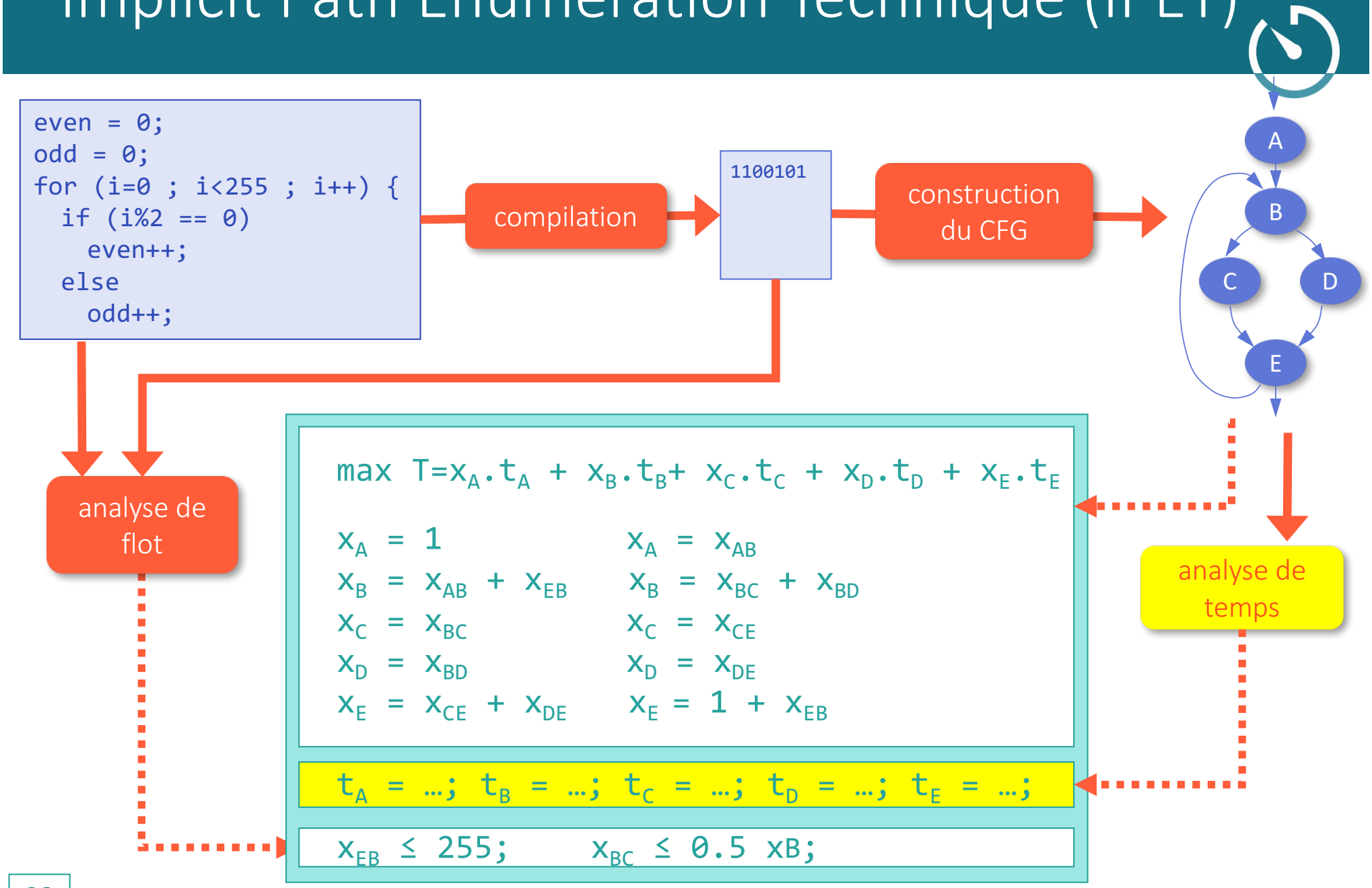
www.otawa.fr



Ballabriga et al. *OTAWA: An Open Toolbox for Adaptive WCET Analysis*, SEUS 2010

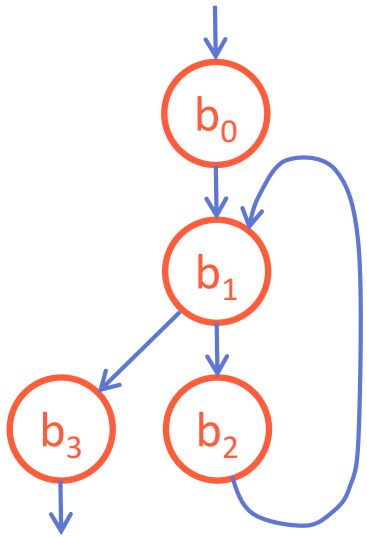
Analyse temporelle d’une tâche en isolation

Implicit Path Enumeration Technique (IPET)



Exemple

```
int product = 1;
int init_val;
for (int i=1; i<N ; i++){
    product *= i;
    a[i] = init_val;
}
```



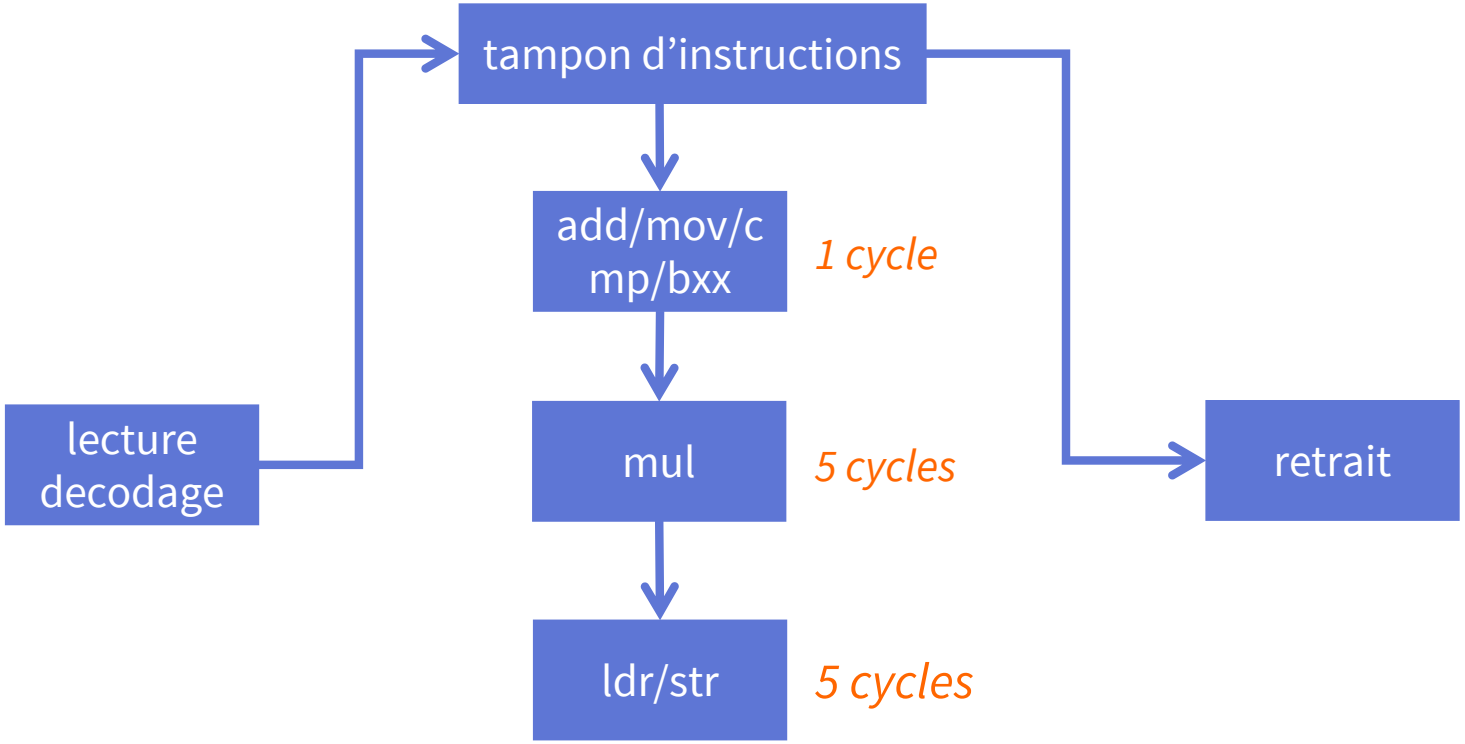

```
__start:    mov r0,#1    @ product
            mov r1,#1    @ i
            adr r2,a
            adr r3,init_val
            ldr r3,[r3]
```
b_0

```
while:      mul r0,r1,r0
            cmp r1,#N
            bhs end
```
b_1

```
            str r3,[r2],#4
            add r1,r1,#1
            b while
```
b_2

```
end:        adr r2,product
            str r0,[r2]
```
b_3

Exemple



une instruction lue/décodée à chaque cycle

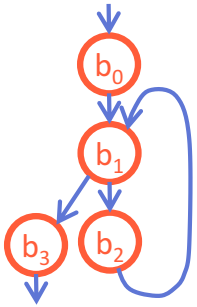
une instruction lancée à chaque cycle

une instruction retirée à chaque cycle

(dans l'ordre du programme)

les branchements sont prédits non pris

Comment calculer le temps d'exécution d'un bloc de base ?




```
0a   __start:  mov r0,#1
0b             mov r1,#0
0c             adr r2,a
0d             adr r3,init_val
0e             ldr r3,[r3]
1a   while:    mul r0,r1,r0
1b             cmp r1,#N
1c             bhs end
2a             str r3,[r2],#4
2b             add r1,r1,#1
2c             b while
3a   end:      adr r2,product
3b             str r0,[r2]
```

b_1 après b_0 — t_0 = 11 cycles — t_{0-1} = 3 cycles

F	0a	0b	0c	0d	0e	1a	1b	1c										
A		0a	0b	0c	0d			1b	1c									
Mu							1a	1a	1a	1a	1a							
Me						0e	0e	0e	0e	0e								
C			0a	0b	0c	0d					0e	1a	1b	1c				

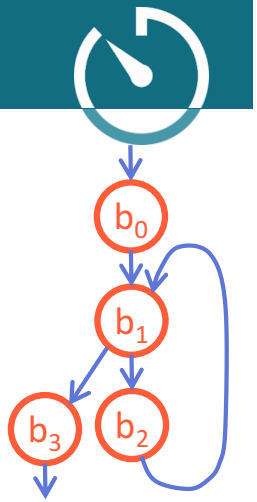
b_1 après b_2 — t_{2-1} = 4 cycles

F	2a	2b	2c	x	1a	1b	1c											
A			2b	2c			1b	1c										
Mu						1a	1a	1a	1a	1a								
Me		2a	2a	2a	2a	2a												
C							2a	2b	2c		1a	1b	1c					

$$\max T = x_0.t_0 + x_1.t_1 + x_2.t_2 + x_3.t_3$$

$$\max T = x_0.t_0 + x_{0-1}.t_{0-1} + x_{2-1}.t_{2-1} + x_{1-2}.t_{1-2} + x_{1-3}.t_{2-3}$$

Limites



- pipelines complexes :
  superscalaires, ooo, unités fonctionnelles multiples, etc.
- état initial / historique : effets longs

```
0a  __start:  mov r0,#1
0b            mov r1,#0
0c            adr r2,a
0d            adr r3,init_val
0e            ldr r3,[r3]
1a  while:    mul r0,r1,r0
1b            cmp r1,#N
1c            bhs end
2a            str r3,[r2],#4
2b            add r1,r1,#1
2c            b while
3a  end:      adr r2,product
3b            str r0,[r2]
```

b_1 après b_0 — t_0 = 11 cycles — $t_{0\text{-}1}$ = 3 cycles

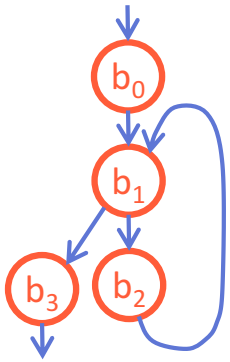
F	0a	0b	0c	0d	0e	1a	1b	1c										
A		0a	0b	0c	0d			1b	1c									
Mu							1a	1a	1a	1a	1a							
Me						0e	0e	0e	0e	0e								
C			0a	0b	0c	0d					0e	1a	1b	1c				

b_2 après b_1 — $t_{1\text{-}2}$ = 3 cycles

F	1a	1b	1c	2a	2b	2c												
A			1b	1c		2b	2c											
Mu		1a	1a	1a	1a	1a												
Me					2a	2a	2a	2a	2a									
C							1a	1b	1c	2a	2b	2c						

$$\max T = x_0.t_0 + x_{0\text{-}1}.t_{0\text{-}1} + x_{2\text{-}1}.t_{2\text{-}1} + x_{1\text{-}2}.t_{1\text{-}2} + x_{1\text{-}3}.t_{2\text{-}3}$$

Effets temporels longs




```
0a  __start:  mov r0,#1
0b            mov r1,#0
0c            adr r2,a
0d            adr r3,init_val
0e            ldr r3,[r3]
1a  while:    mul r0,r1,r0
1b            cmp r1,#N
1c            bhs end
2a            str r3,[r2],#4
2b            add r1,r1,#1
2c            b while
3a  end:      adr r2,product
3b            str r0,[r2]
```

t_0 = 11 cycles $t_{0\text{-}1}$ = 3 cycles $t_{1\text{-}2}$ = 3 cycles

$t_{0\text{-}1\text{-}2} = t_0 + t_{0\text{-}1} + t_{1\text{-}2}$ = 17 cycles

b_2 après b_1 après b_0

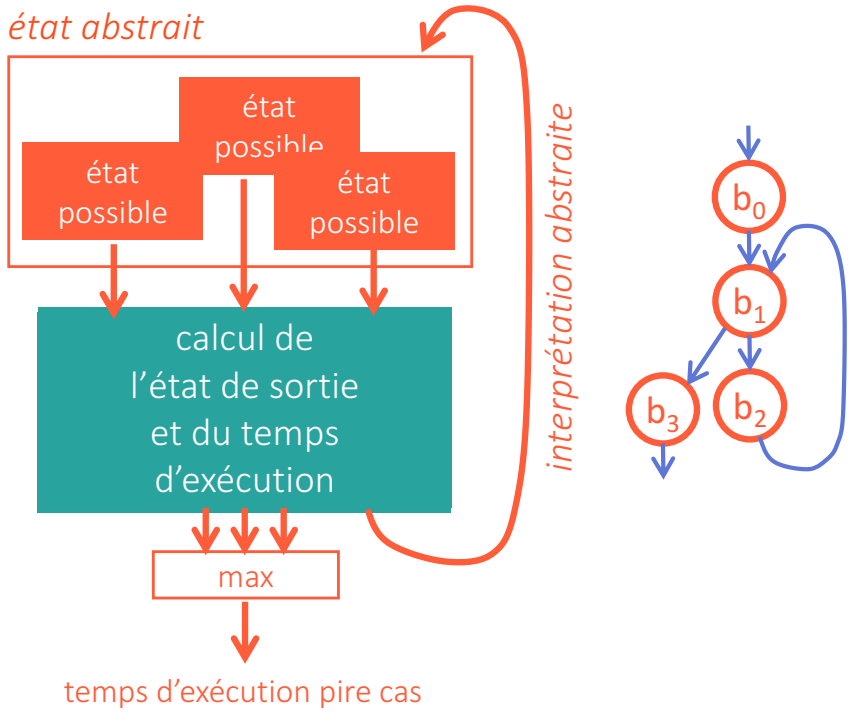
F	0a	0b	0c	0d	0e	1a	1b	1c	2a	2b	2c						
A		0a	0b	0c	0d			1b	1c		2b	2c					
Mu							1a	1a	1a	1a	1a						
Me						0e	0e	0e	0e	0e	2a	2a	2a	2a	2a		
C			0a	0b	0c	0d					0e	1a	1b	1c		2a	2b	2c

$t_{0\text{-}1\text{-}2}$ = 18 cycles 😟

Interprétation abstraite

Une théorie d'approximation de la sémantique de programmes

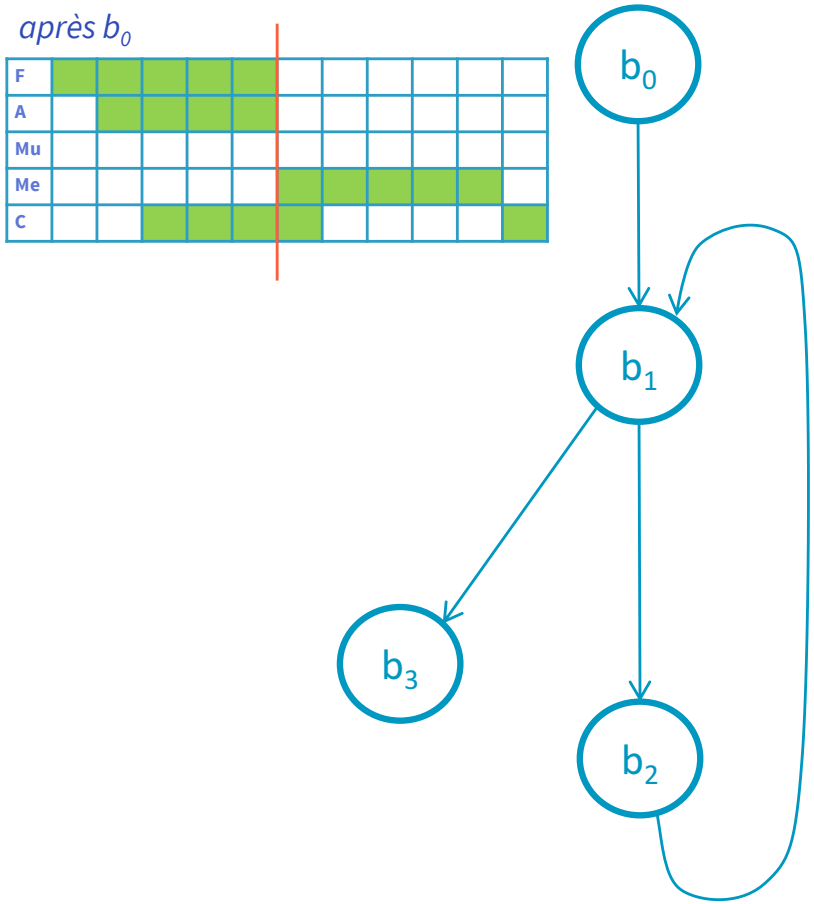
- exécution partielle d'un programme pour obtenir des informations qui peuvent être utilisées pour répondre à une question donnée



P. Cousot, R. Cousot, *Abstract interpretation: a unified lattice model for static analysis of programs by construction or approximation of fixpoints,* POPL, 1977

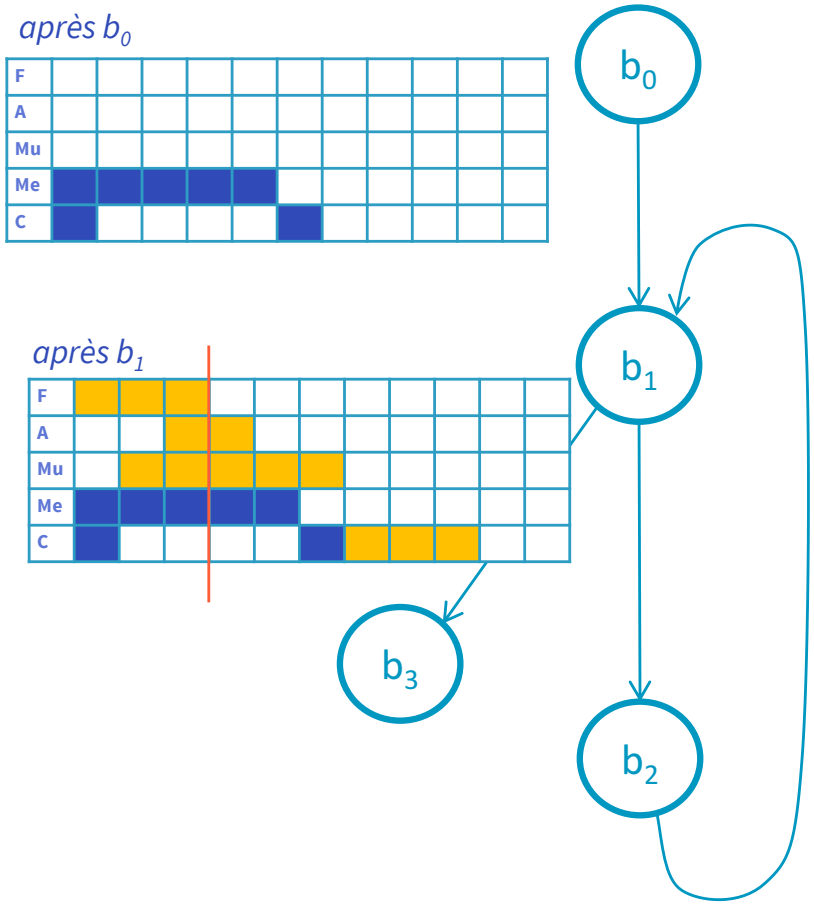
Comment pourrait-on analyser l'effet du pipeline par interprétation abstraite ?

```
0a  __start:  mov r0,#1
0b            mov r1,#0
0c            adr r2,a
0d            adr r3,init_val
0e            ldr r3,[r3]
1a  while:    mul r0,r1,r0
1b            cmp r1,#N
1c            bhs end
2a            str r3,[r2],#4
2b            add r1,r1,#1
2c            b while
3a  end:      adr r2,product
3b            str r0,[r2]
```



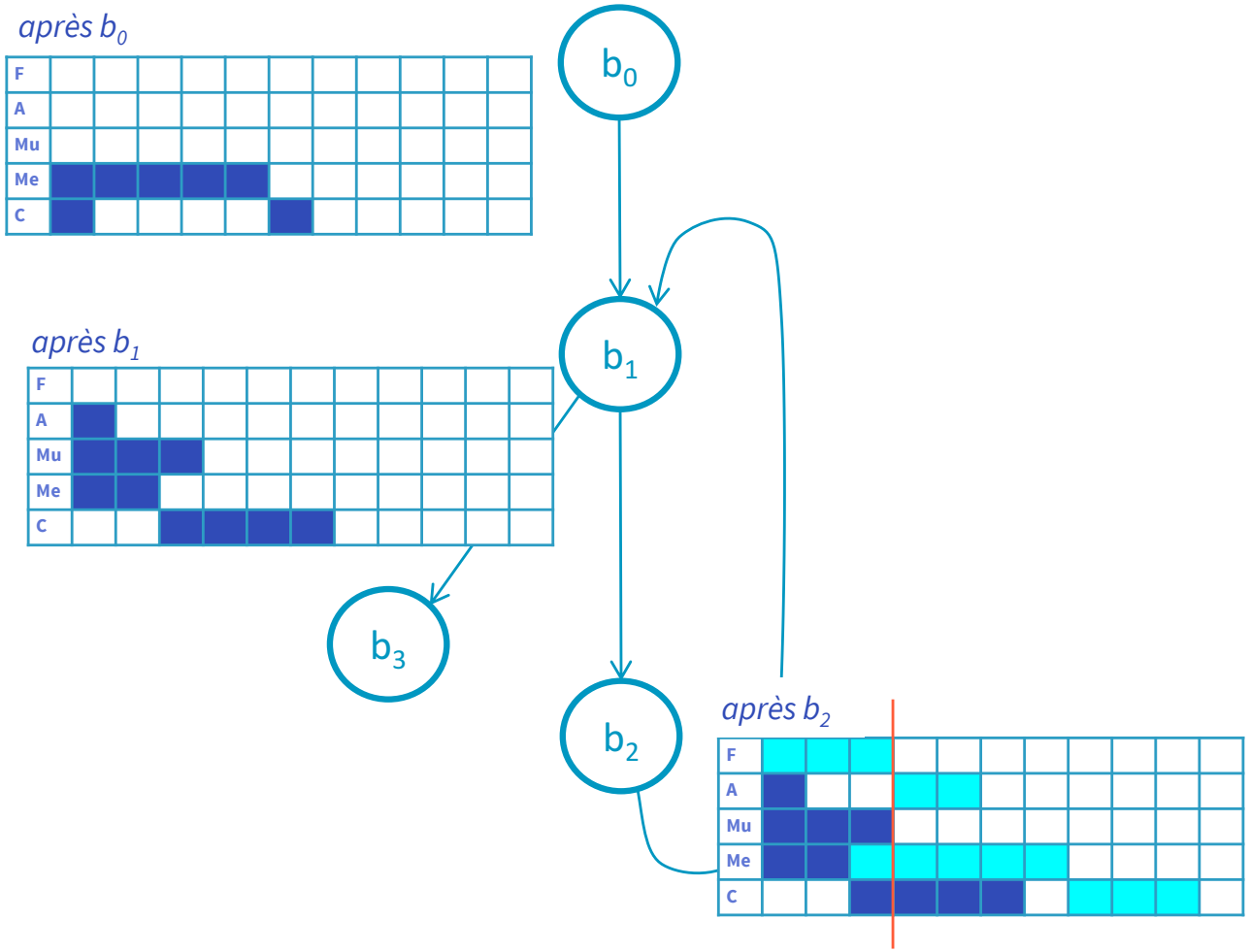
Comment pourrait-on analyser l'effet du pipeline par interprétation abstraite ?

```
0a  __start:  mov r0,#1
0b            mov r1,#0
0c            adr r2,a
0d            adr r3,init_val
0e            ldr r3,[r3]
1a  while:    mul r0,r1,r0
1b            cmp r1,#N
1c            bhs end
2a            str r3,[r2],#4
2b            add r1,r1,#1
2c            b while
3a  end:      adr r2,product
3b            str r0,[r2]
```



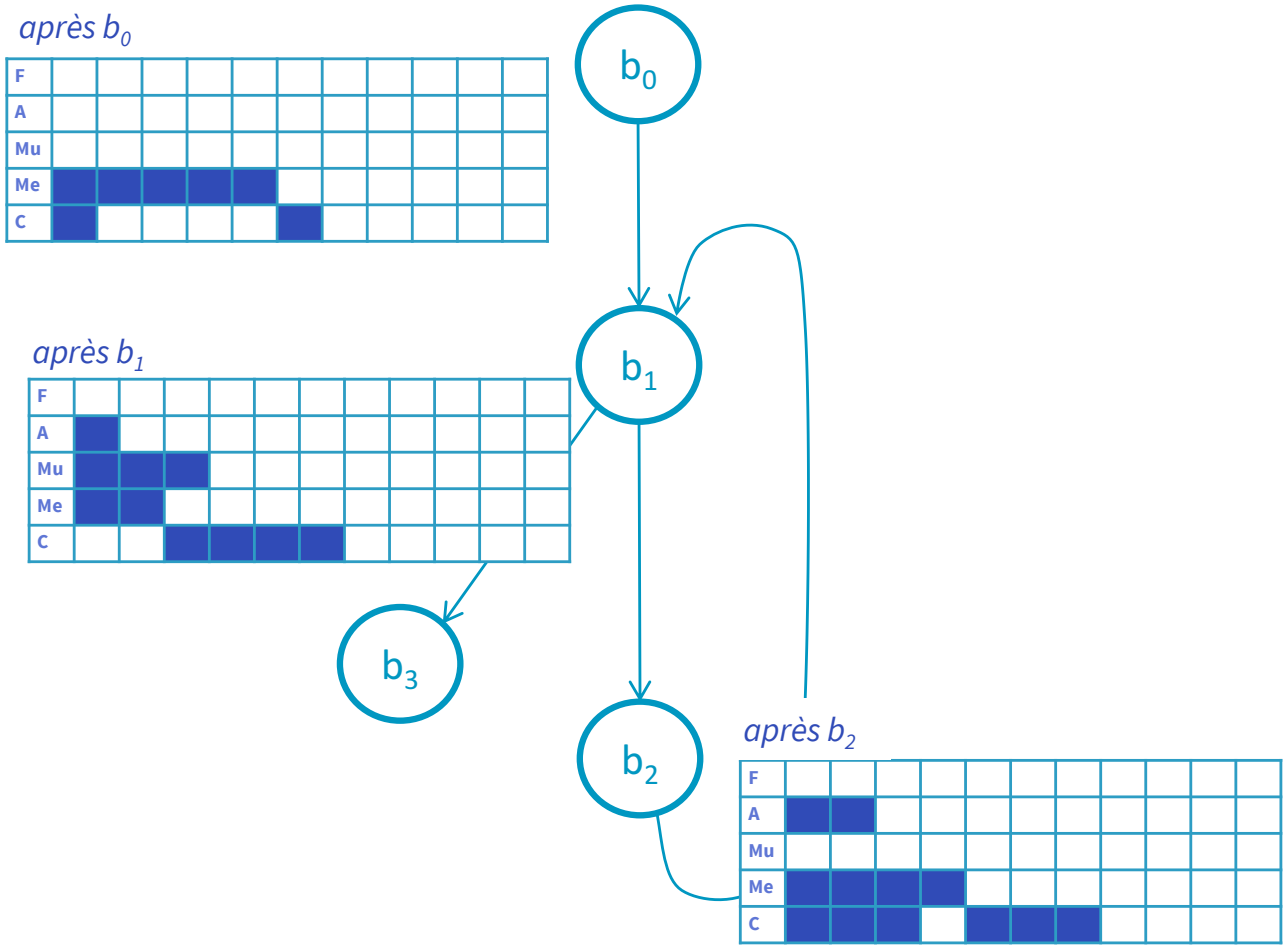
Comment pourrait-on analyser l'effet du pipeline par interprétation abstraite ?

```
0a  __start:  mov r0,#1
0b            mov r1,#0
0c            adr r2,a
0d            adr r3,init_val
0e            ldr r3,[r3]
1a  while:    mul r0,r1,r0
1b            cmp r1,#N
1c            bhs end
2a            str r3,[r2],#4
2b            add r1,r1,#1
2c            b while
3a  end:      adr r2,product
3b            str r0,[r2]
```



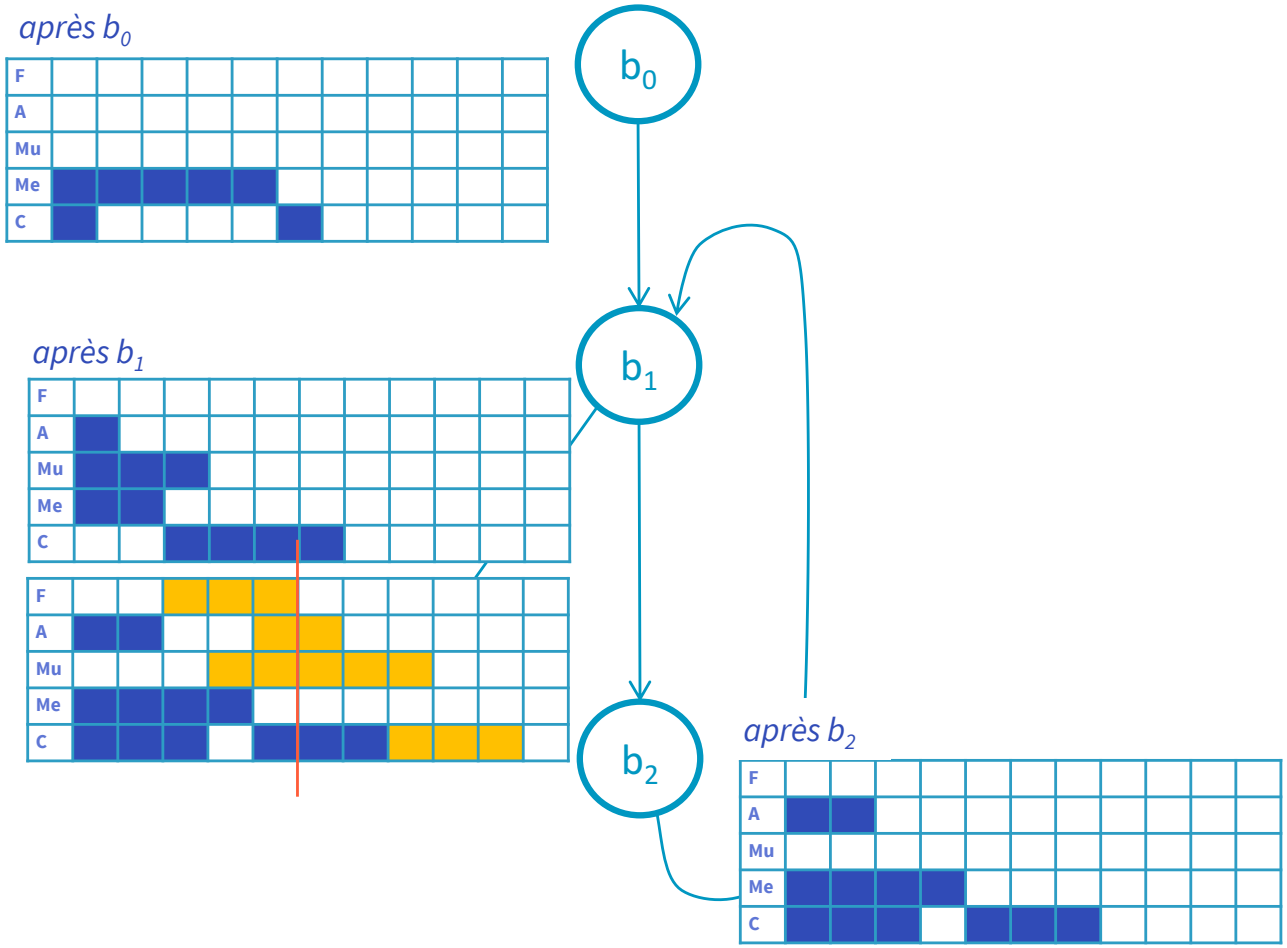
Comment pourrait-on analyser l'effet du pipeline par interprétation abstraite ?

```
0a  __start:  mov r0,#1
0b            mov r1,#0
0c            adr r2,a
0d            adr r3,init_val
0e            ldr r3,[r3]
1a  while:    mul r0,r1,r0
1b            cmp r1,#N
1c            bhs end
2a            str r3,[r2],#4
2b            add r1,r1,#1
2c            b while
3a  end:      adr r2,product
3b            str r0,[r2]
```



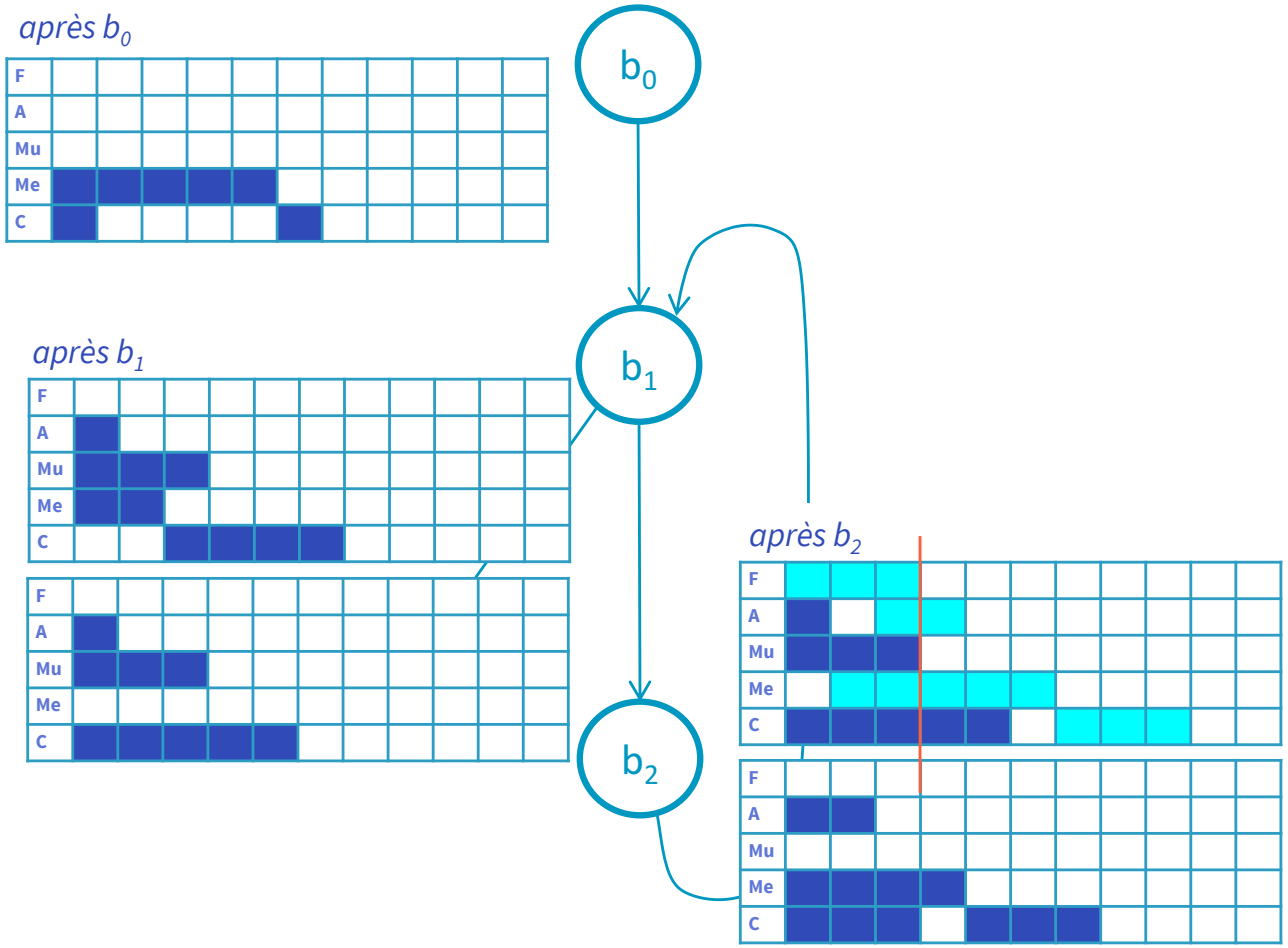
Comment pourrait-on analyser l'effet du pipeline par interprétation abstraite ?

```
0a  __start:  mov r0,#1
0b            mov r1,#0
0c            adr r2,a
0d            adr r3,init_val
0e            ldr r3,[r3]
1a  while:    mul r0,r1,r0
1b            cmp r1,#N
1c            bhs end
2a            str r3,[r2],#4
2b            add r1,r1,#1
2c            b while
3a  end:      adr r2,product
3b            str r0,[r2]
```



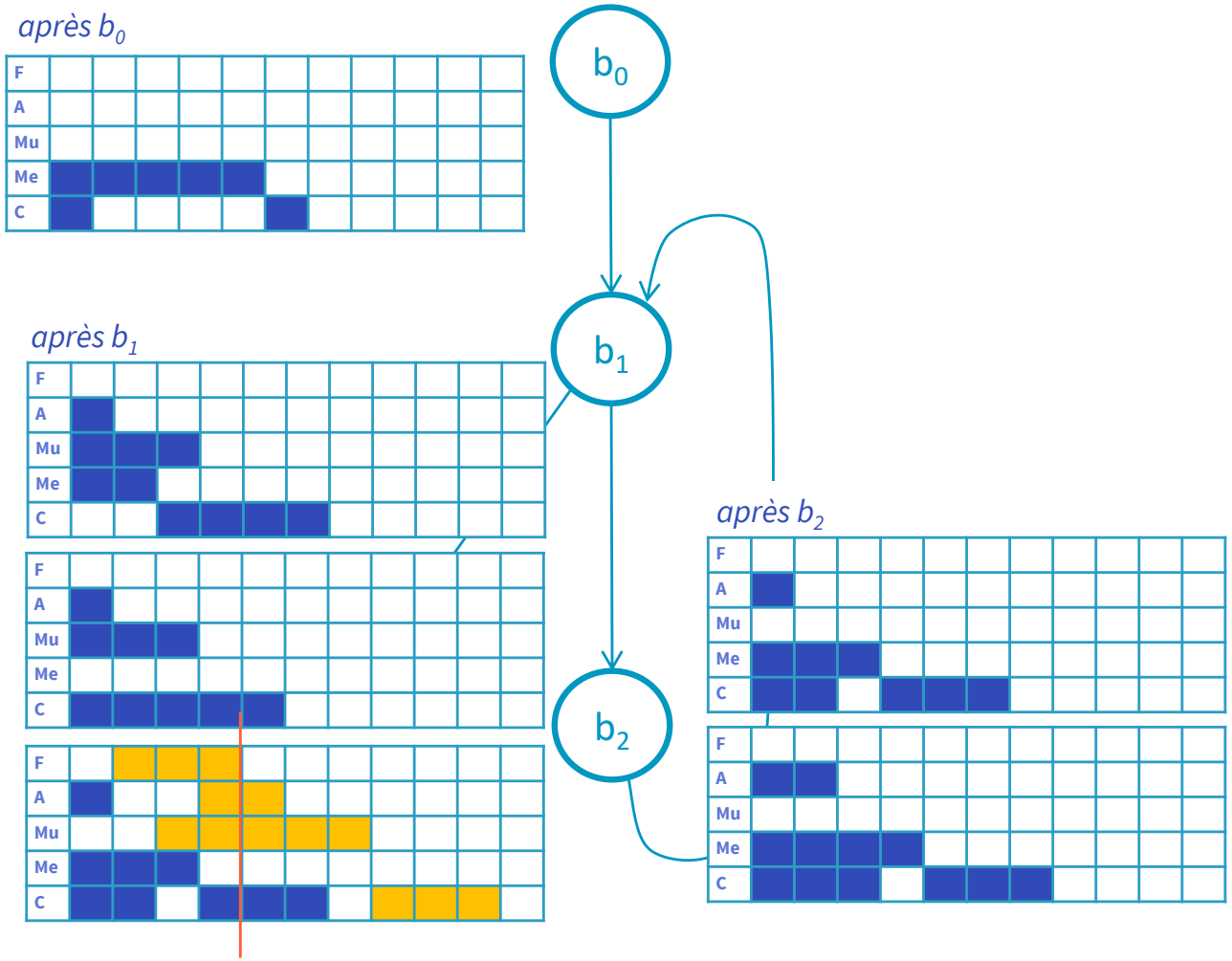
Comment pourrait-on analyser l'effet du pipeline par interprétation abstraite ?

```
0a  __start:  mov r0,#1
0b            mov r1,#0
0c            adr r2,a
0d            adr r3,init_val
0e            ldr r3,[r3]
1a  while:    mul r0,r1,r0
1b            cmp r1,#N
1c            bhs end
2a            str r3,[r2],#4
2b            add r1,r1,#1
2c            b while
3a  end:      adr r2,product
3b            str r0,[r2]
```



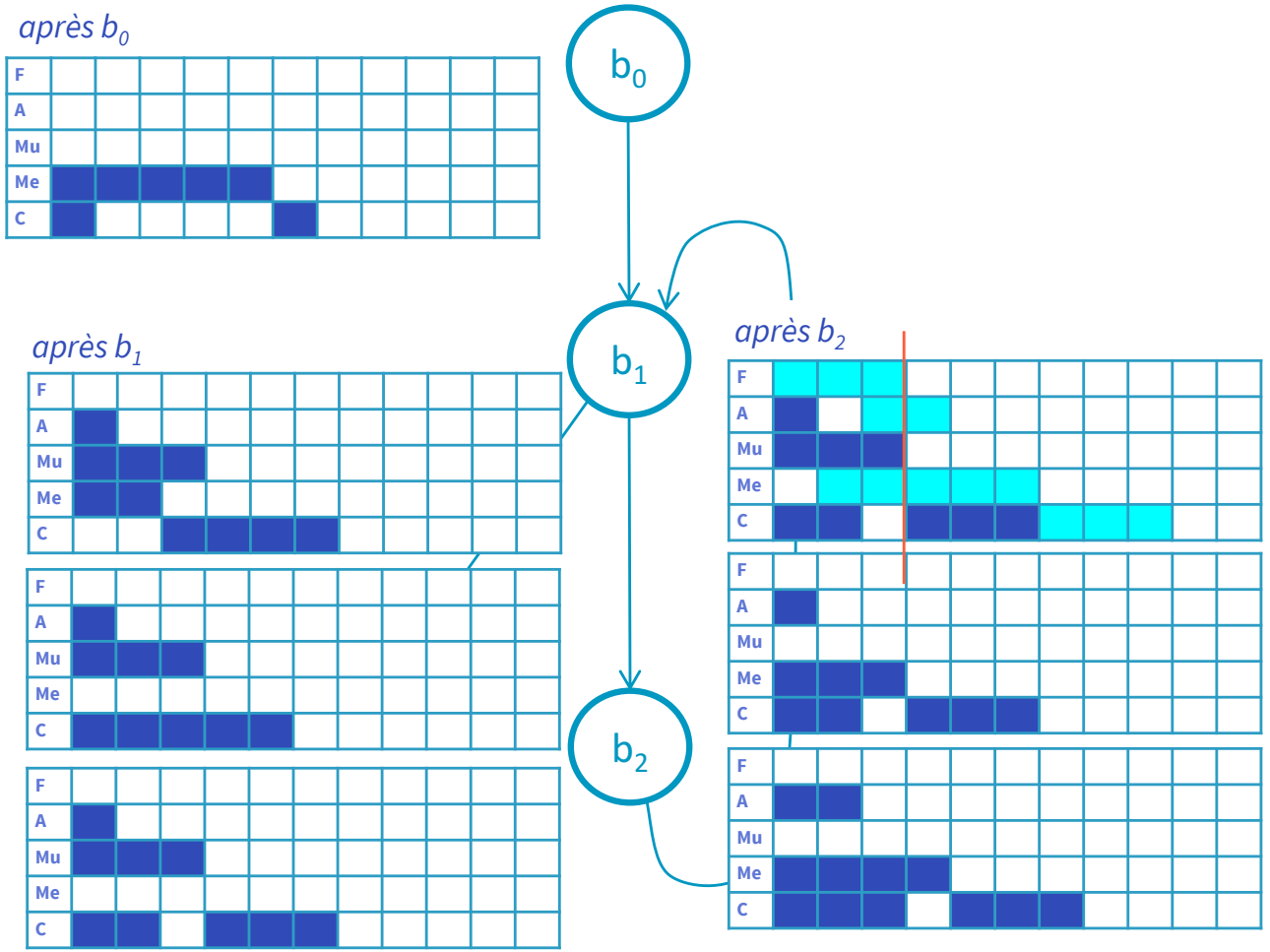
Comment pourrait-on analyser l'effet du pipeline par interprétation abstraite ?

```
0a  __start:  mov r0,#1
0b            mov r1,#0
0c            adr r2,a
0d            adr r3,init_val
0e            ldr r3,[r3]
1a  while:    mul r0,r1,r0
1b            cmp r1,#N
1c            bhs end
2a            str r3,[r2],#4
2b            add r1,r1,#1
2c            b while
3a  end:      adr r2,product
3b            str r0,[r2]
```



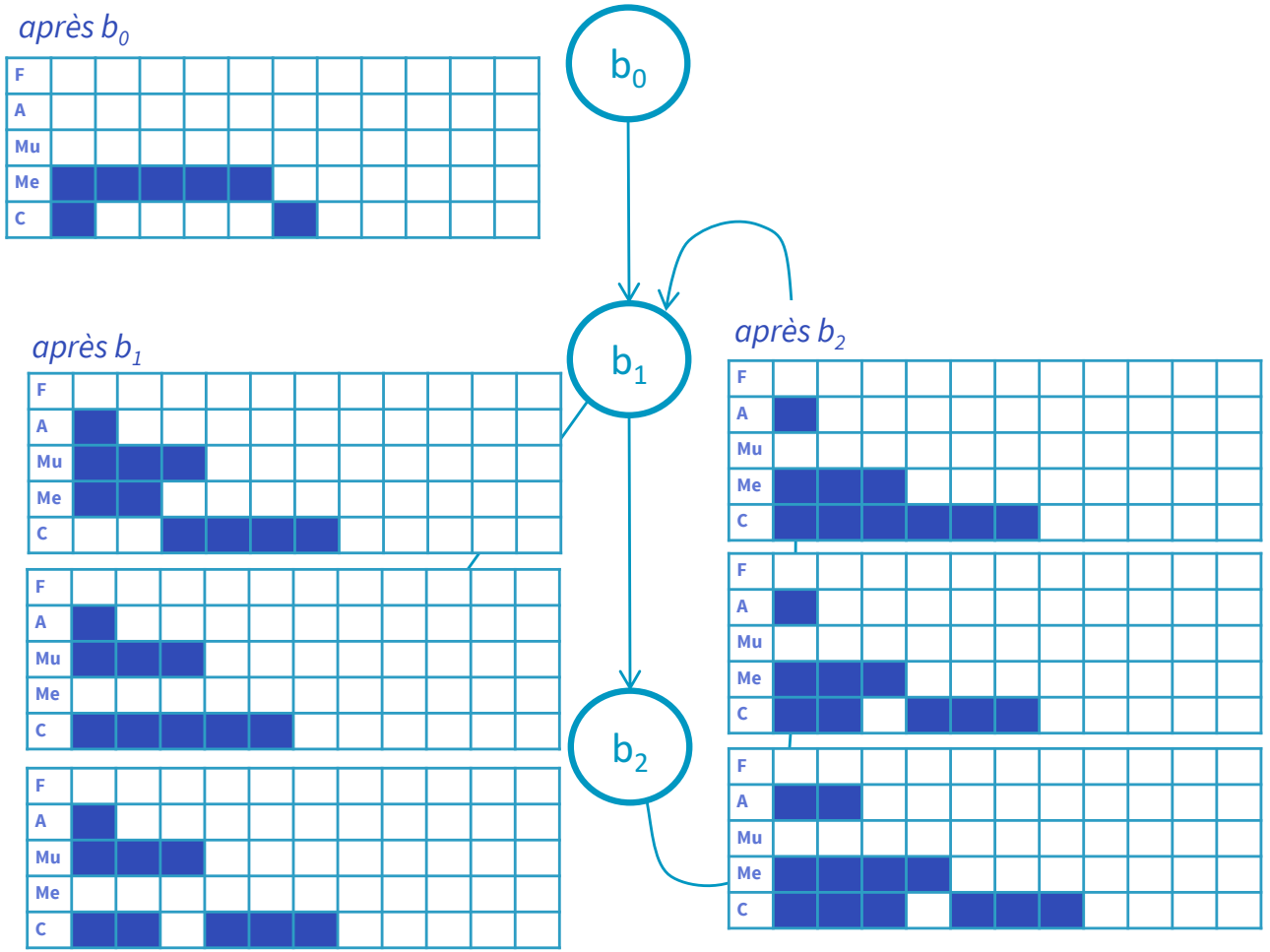
Comment pourrait-on analyser l'effet du pipeline par interprétation abstraite ?

```
0a  __start:  mov r0,#1
0b            mov r1,#0
0c            adr r2,a
0d            adr r3,init_val
0e            ldr r3,[r3]
1a  while:    mul r0,r1,r0
1b            cmp r1,#N
1c            bhs end
2a            str r3,[r2],#4
2b            add r1,r1,#1
2c            b while
3a  end:      adr r2,product
3b            str r0,[r2]
```



Comment pourrait-on analyser l'effet du pipeline par interprétation abstraite ?

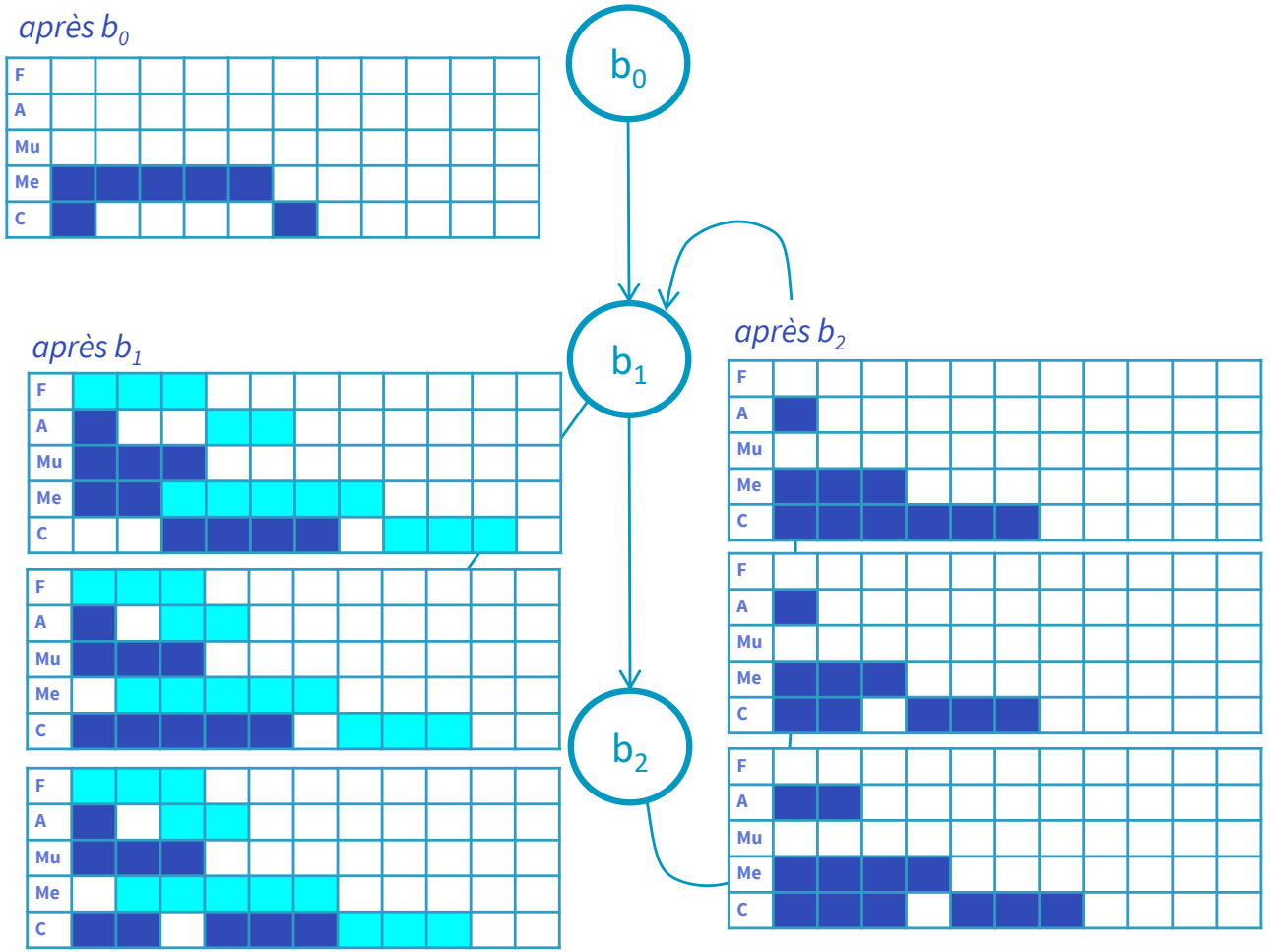
```
0a  __start:  mov r0,#1
0b            mov r1,#0
0c            adr r2,a
0d            adr r3,init_val
0e            ldr r3,[r3]
1a  while:    mul r0,r1,r0
1b            cmp r1,#N
1c            bhs end
2a            str r3,[r2],#4
2b            add r1,r1,#1
2c            b while
3a  end:      adr r2,product
3b            str r0,[r2]
```



on a atteint
un point fixe !

Comment pourrait-on analyser l'effet du pipeline par interprétation abstraite ?

```
0a  __start:  mov r0,#1
0b            mov r1,#0
0c            adr r2,a
0d            adr r3,init_val
0e            ldr r3,[r3]
1a  while:    mul r0,r1,r0
1b            cmp r1,#N
1c            bhs end
2a            str r3,[r2],#4
2b            add r1,r1,#1
2c            b while
3a  end:      adr r2,product
3b            str r0,[r2]
```

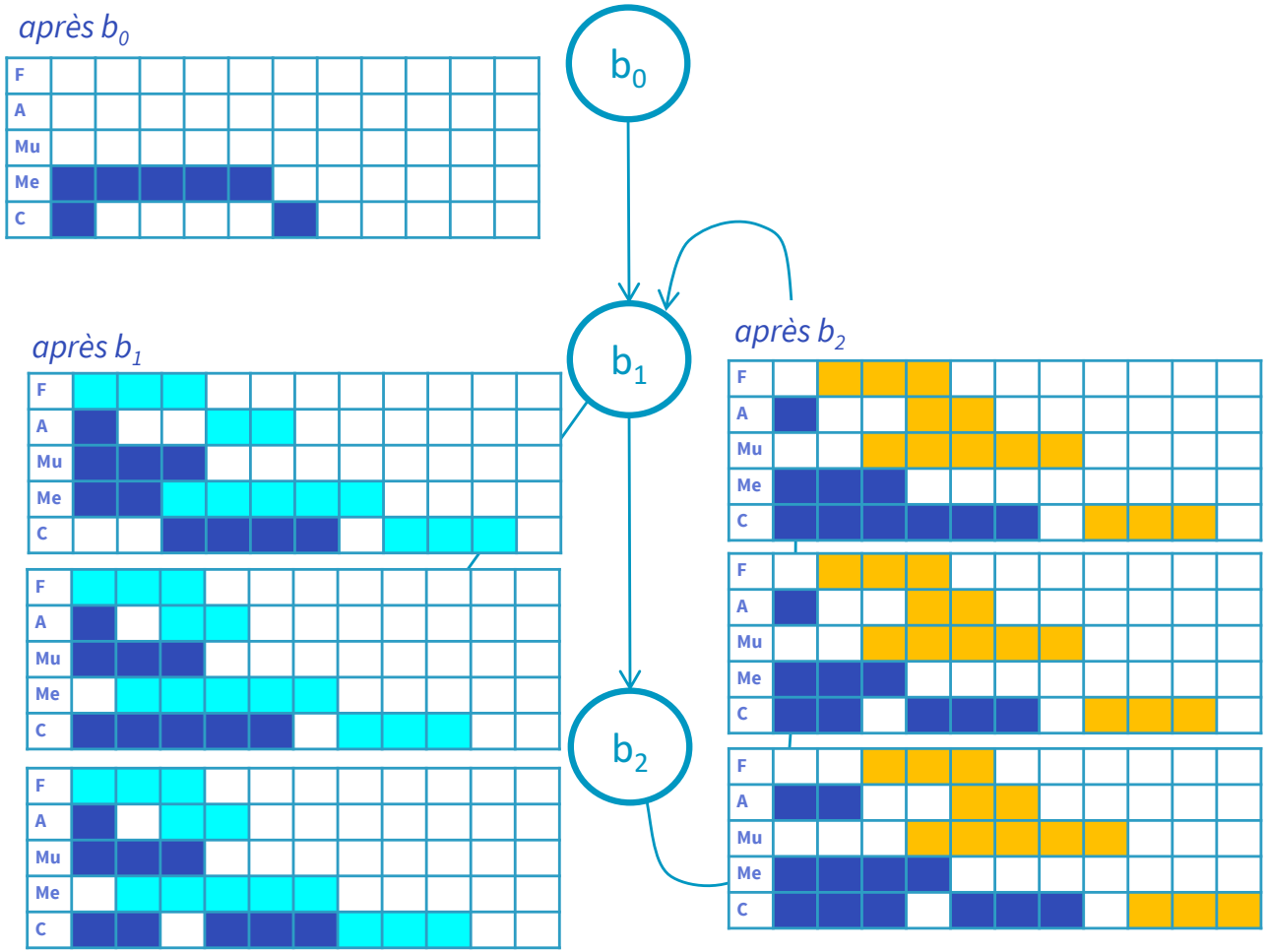


$t_{1\text{-}2} = \max(4,4,3) = 4$

$t_{0\text{-}1\text{-}2} = 11 + 3 + \mathbf{4} = 18$

Comment pourrait-on analyser l'effet du pipeline par interprétation abstraite ?

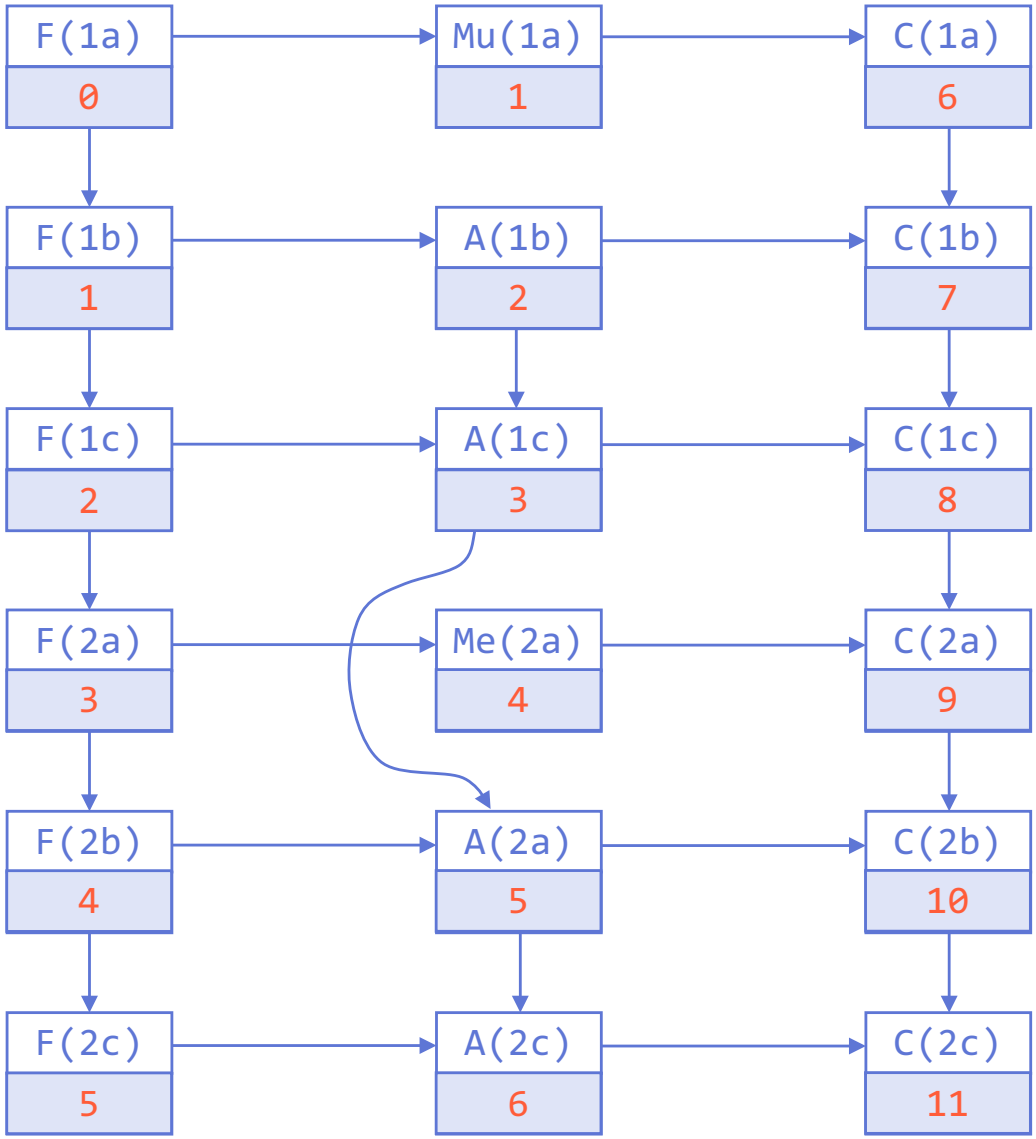
```
0a  __start:  mov r0,#1
0b            mov r1,#0
0c            adr r2,a
0d            adr r3,init_val
0e            ldr r3,[r3]
1a  while:    mul r0,r1,r0
1b            cmp r1,#N
1c            bhs end
2a            str r3,[r2],#4
2b            add r1,r1,#1
2c            b while
3a  end:      adr r2,product
3b            str r0,[r2]
```



$t_{2\text{-}1} = \max(4,4,4) = 4$

Une autre approche (locale) pour l'analyse de pipeline

```
0a  __start:  mov r0,#1
0b            mov r1,#0
0c            adr r2,a
0d            adr r3,init_val
0e            ldr r3,[r3]
1a  while:    mul r0,r1,r0
1b            cmp r1,#N
1c            bhs end
2a            str r3,[r2],#4
2b            add r1,r1,#1
2c            b while
3a  end:      adr r2,product
3b            str r0,[r2]
```



$t_{1\text{-}2}$ = 3 cycles

Une autre approche (locale) pour l'analyse de pipeline

Etat initial « paramétré »

```
0a  __start:  mov r0,#1
0b            mov r1,#0
0c            adr r2,a
0d            adr r3,init_val
0e            ldr r3,[r3]
1a  while:    mul r0,r1,r0
1b            cmp r1,#N
1c            bhs end
2a            str r3,[r2],#4
2b            add r1,r1,#1
2c            b while
3a  end:      adr r2,product
3b            str r0,[r2]
```

chaque ressource a une date de disponibilité

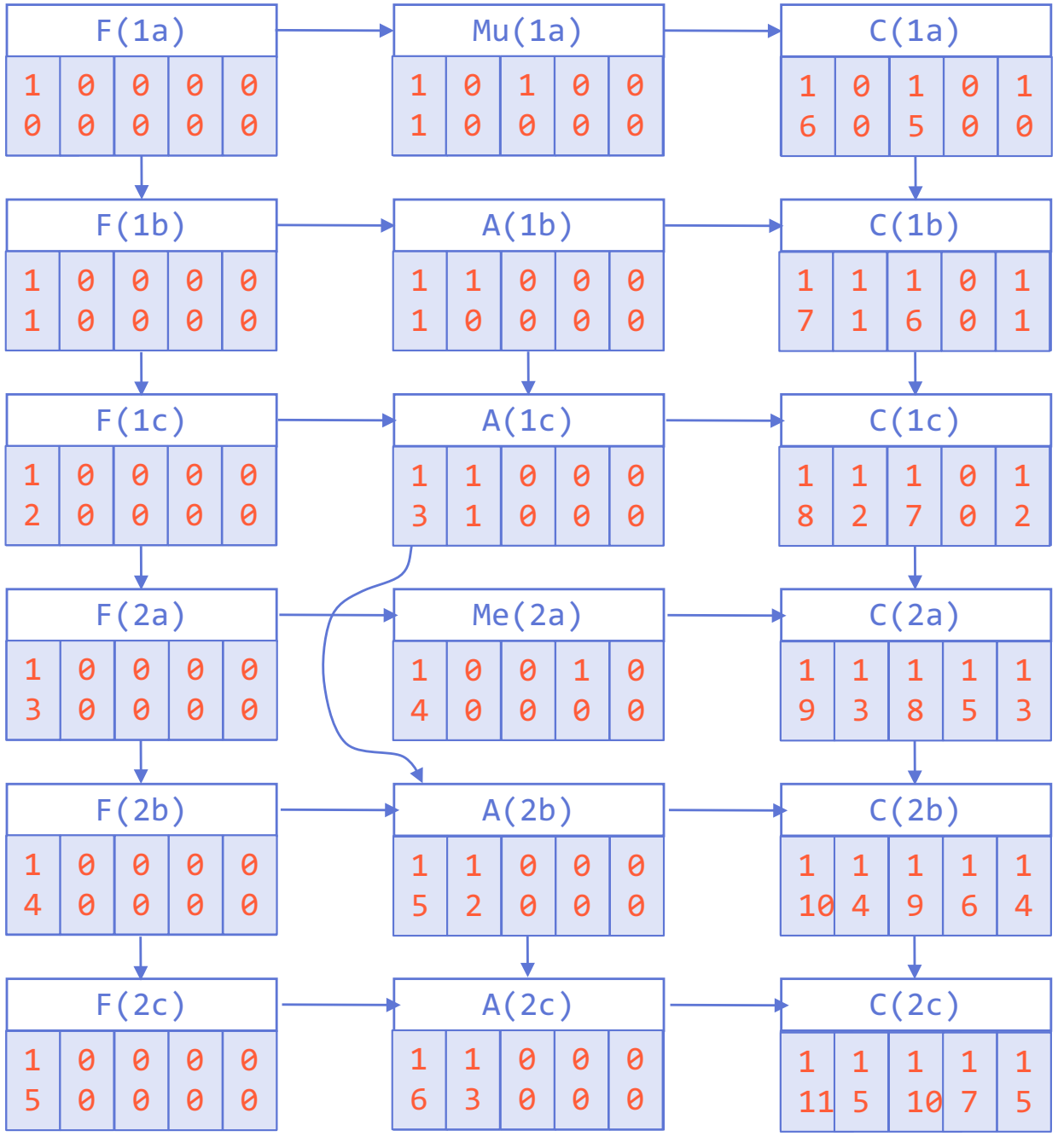
date de démarrage d'un noeud

F	A	Mu	Me	C	r0	r1	r2	r3

e_i = dépend de la ressource ?

d_i = délai après que la ressource soit devenue disponible

Rochange, Sainrat, *A Context-Parameterized Model for Static Analysis of ExecutionTimes,* Transactions on HiPEAC, Springer, 2009




```
0a  __start:  mov r0,#1
0b            mov r1,#0
0c            adr r2,a
0d            adr r3,init_val
0e            ldr r3,[r3]
1a  while:    mul r0,r1,r0
1b            cmp r1,#N
1c            bhs end
2a            str r3,[r2],#4
2b            add r1,r1,#1
2c            b while
3a  end:      adr r2,product
3b            str r0,[r2]
```

Une autre approche (locale) pour l'analyse de pipeline

C(1c)				
1	1	1	0	1
8	2	7	0	2

C(2c)				
1	1	1	1	1
11	5	10	7	5

F	A	Mu	Me	C

$t_{C(1c)} = \max(t_F+8 , t_A+2 , t_{Mu}+7 , , t_C+2)$

$t_{Me} \leq t_C - 1$

$t_{C(2c)} = \max(t_F+11 , t_A+5 , t_{Mu}+10 , t_{Me}+7 , t_C+5)$

$\leq t_C + 6$

$t_{C(2c)} - t_{C(1c)} = \max(11-8 , 5-2 , 10-7 , 6-2 , 5-2)$

$t_{1\text{-}2} \leq 4$ cycles

Instruction à latence variable

Latence variable ?

- qui dépend de la valeur des opérandes (ex : multiplication)
- qui dépend de l'état de certains composants
  - exemple : cache d'instructions ou de données

Peut-on considérer le pire cas ?

- la valeur maximale de la latence

Risque d'anomalie temporelle…

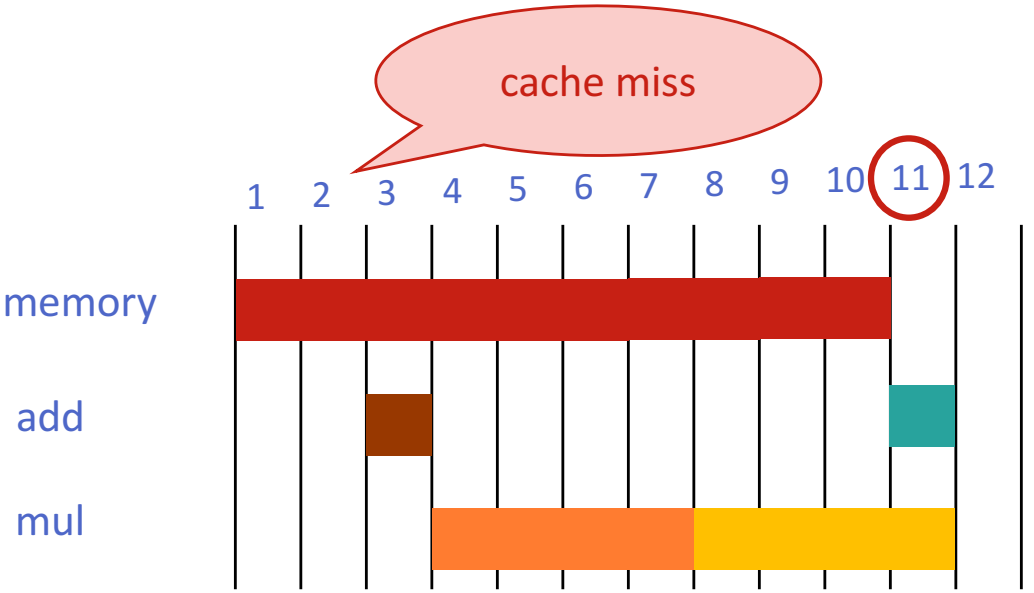
Lundqvist, Stenström, *Timing anomalies in dynamically scheduled microprocessors.* RTSS, 1999

Reineke et al., *A definition and classification of timing anomalies.* Workshop on WCET analysis, 2006

Anomalies temporelles

decode cycle		
1		`ldr r4,[r3]`
2		`add r5,r4,r4`
3		`add r11,r10,r10`
4		`mul r11,r8,r11`
5		`mul r19,r1,r2`

Le pire cas local n'est pas forcément le pire cas global…



Analyse du comportement du cache

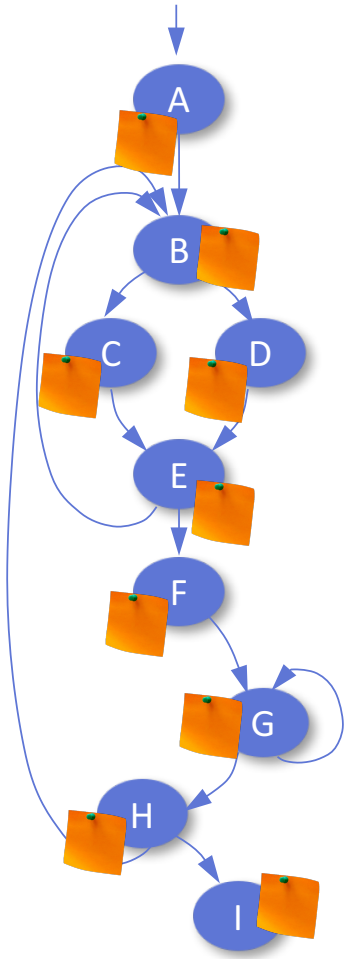
Objectif : classifier les accès au cache

- AlwaysHit, Always Miss
- FirstMiss

→ la latence est connue

- ? (NotClassified)

→ la latence n'est pas connue



Alt et al.,
Cache behaviour prediction by abstract interpretation, SAS 1996

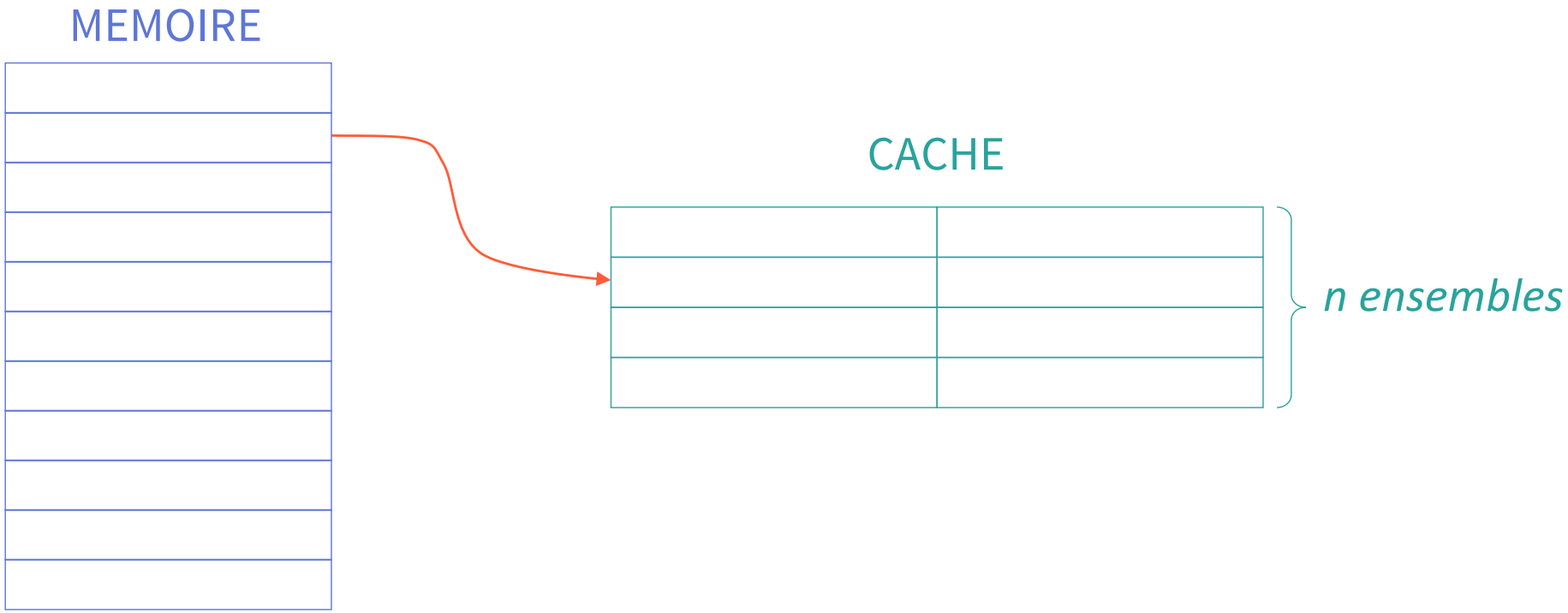
Li, Malik, Wolfe,
Cache modeling for real-time software: beyond direct-mapped instruction caches, RTSS 1996

Lv et al.,
*A **Survey** on Static Cache Analysis for Real-Time Systems,* LITES 3(1), 2016

Analyse du comportement du cache

Hypothèses (pour la présentation)

- cache d'instructions associatif par ensembles
- avec une politique de remplacement LRU



Analyse du comportement du cache

Etat (concret) d'un ensemble du cache

séquence d'accès : a-b-c-d

contenu de l'ensemble

a	c	d	b

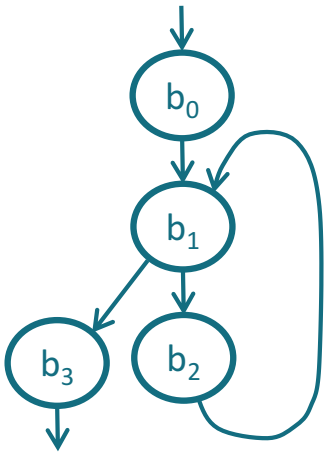
âge (LRU)

3	1	0	2

d	0
c	1
b	2
a	3

Analyse par interprétation abstraite

- calcule les états possibles de chaque ensemble en chaque point du programme
- deux analyses :
  - MAY: quels blocs mémoire *peuvent* être dans le cache?
  - MUST: … *doivent* …. ?
- pour chaque analyse, on doit définir :
  - ce qu'est l'état abstrait d'un ensemble (ACS)
  - des fonctions de mise à jour (`update()`) et de jonction (`join()`)



Analyse du comportement du cache

Analyse MUST

- état abstrait d'un ensemble = bornes supérieure sur les âges des blocs mémoire

Contenu d'un ensemble

âge	
0	{ }
1	{a, b}
2	{ }
3	{c}

abstrait

a	b	b
b	a	a
c	c	e
d	d	c

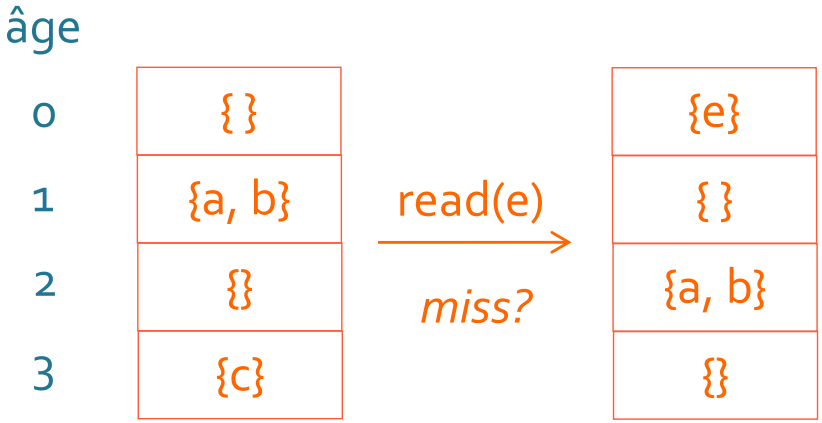
...

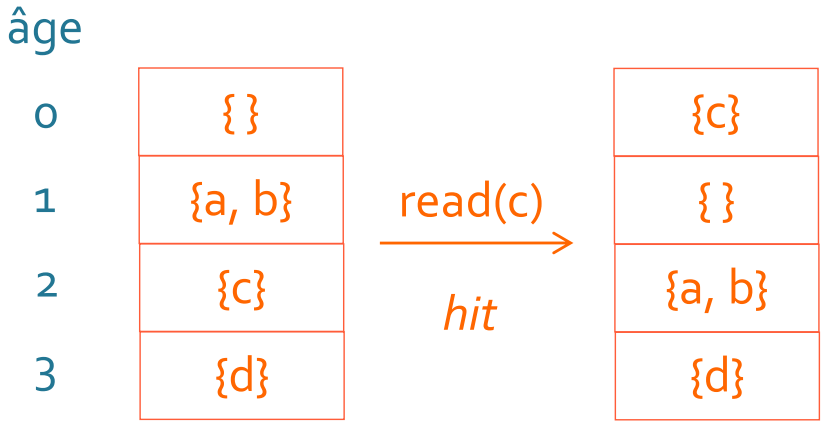
concret

Analyse du comportement du cache

Analyse MUST

- fonction `update()`

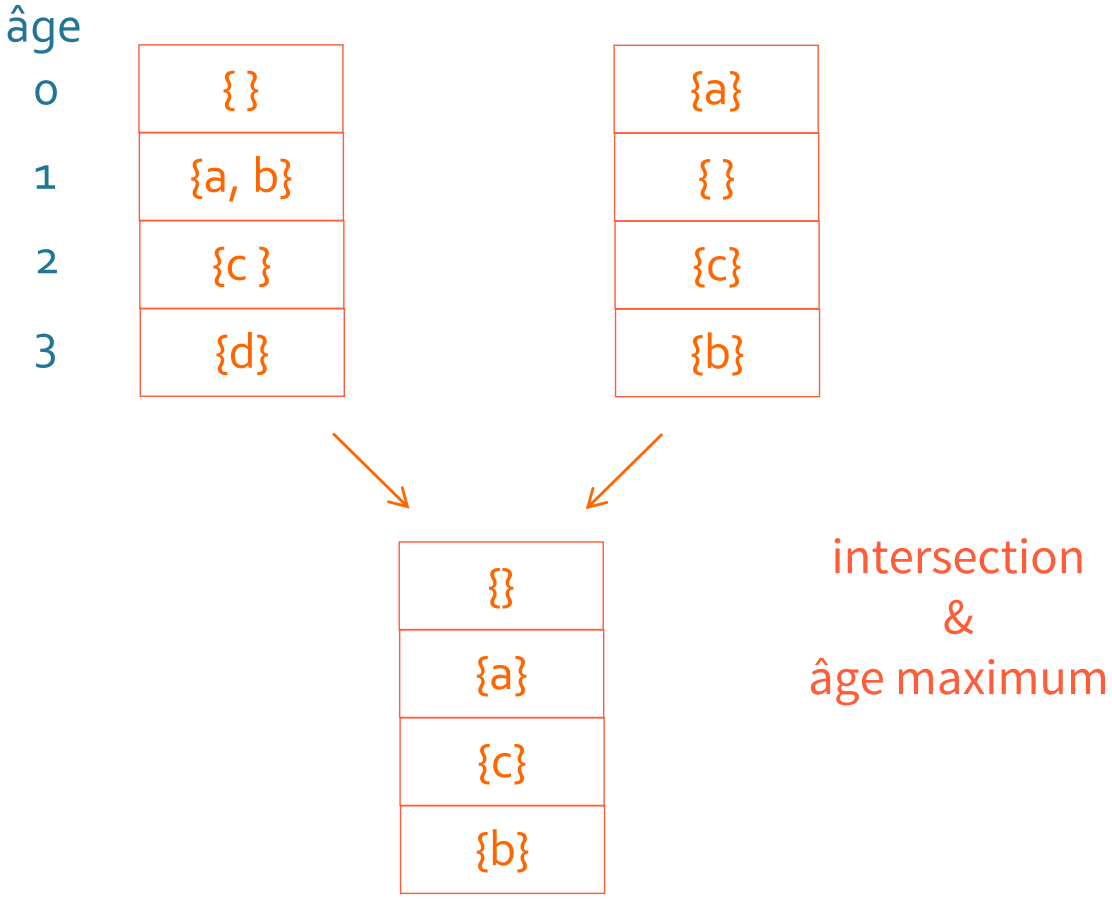




Analyse du comportement du cache

Analyse MUST

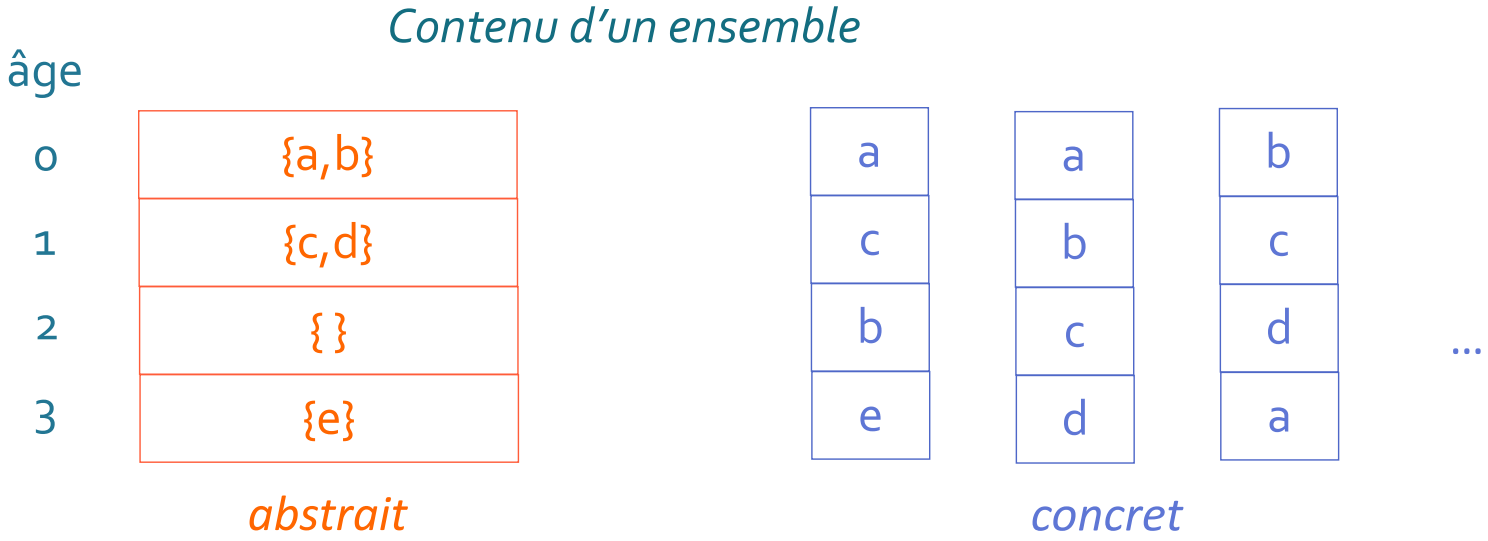
- fonction `join()`



Analyse du comportement du cache

Analyse MAY

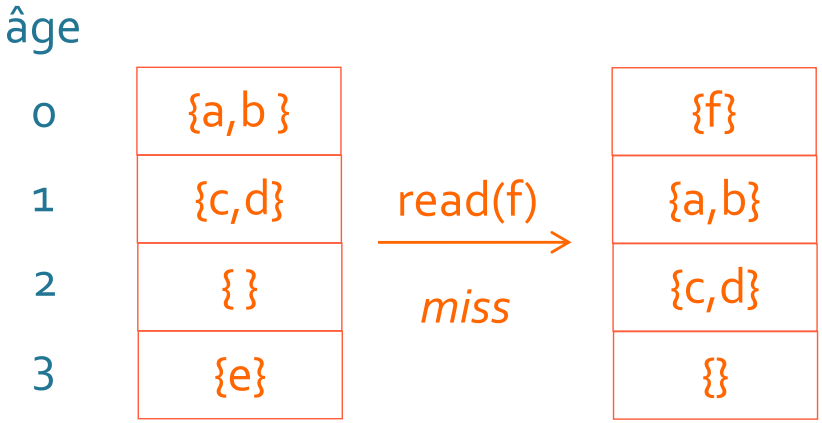
- état abstrait d'un ensemble = bornes inférieures sur les âges des blocs mémoire

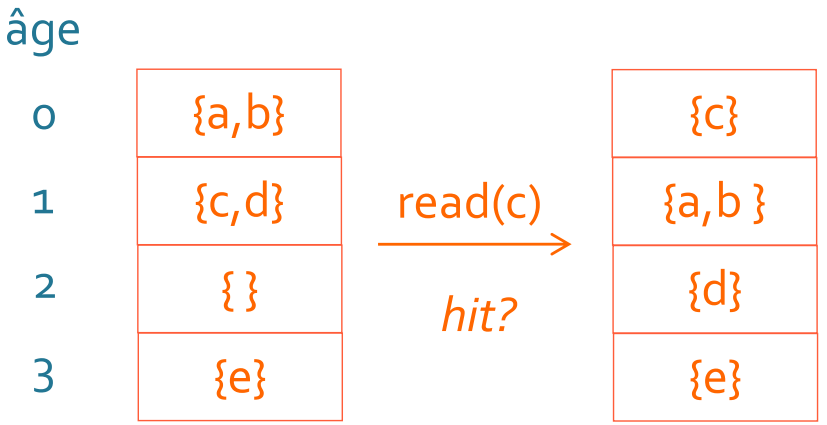


Analyse du comportement du cache

Analyse MAY

- fonction `update()`

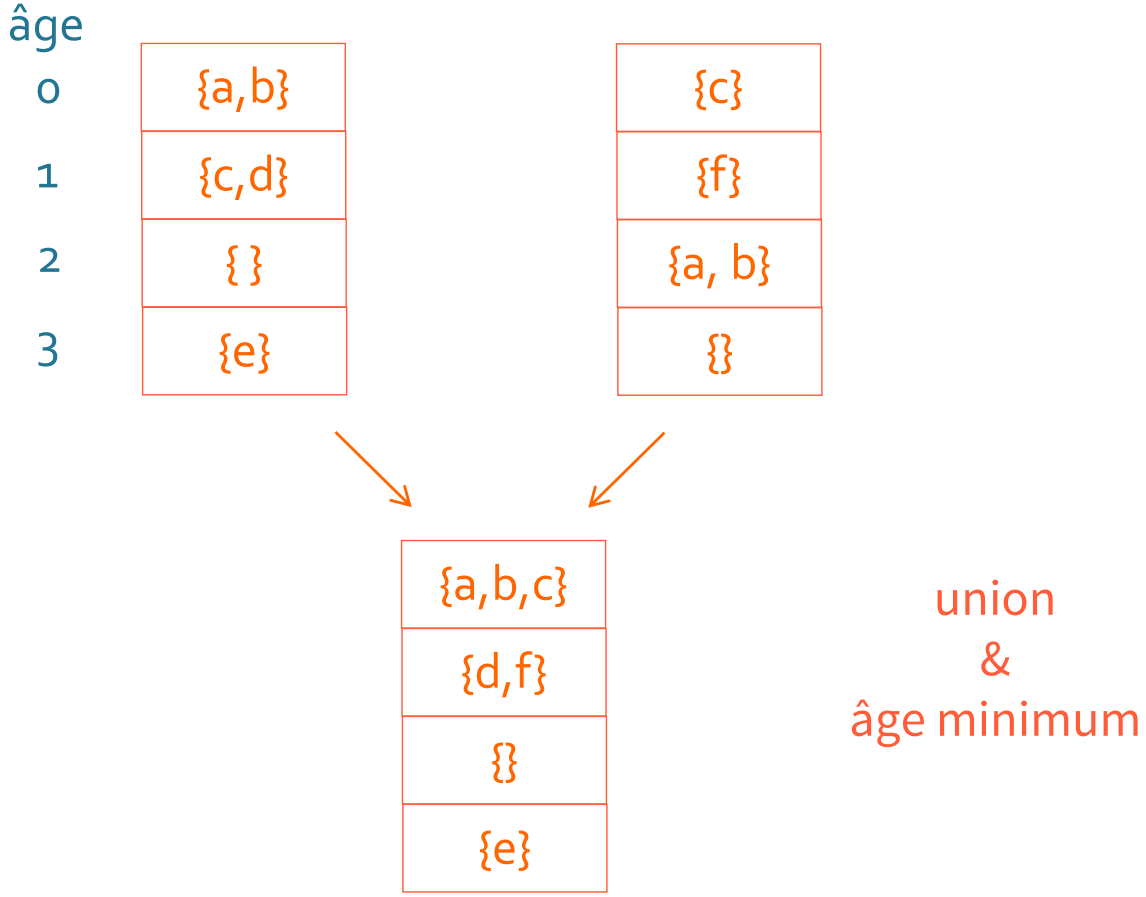




Analyse du comportement du cache

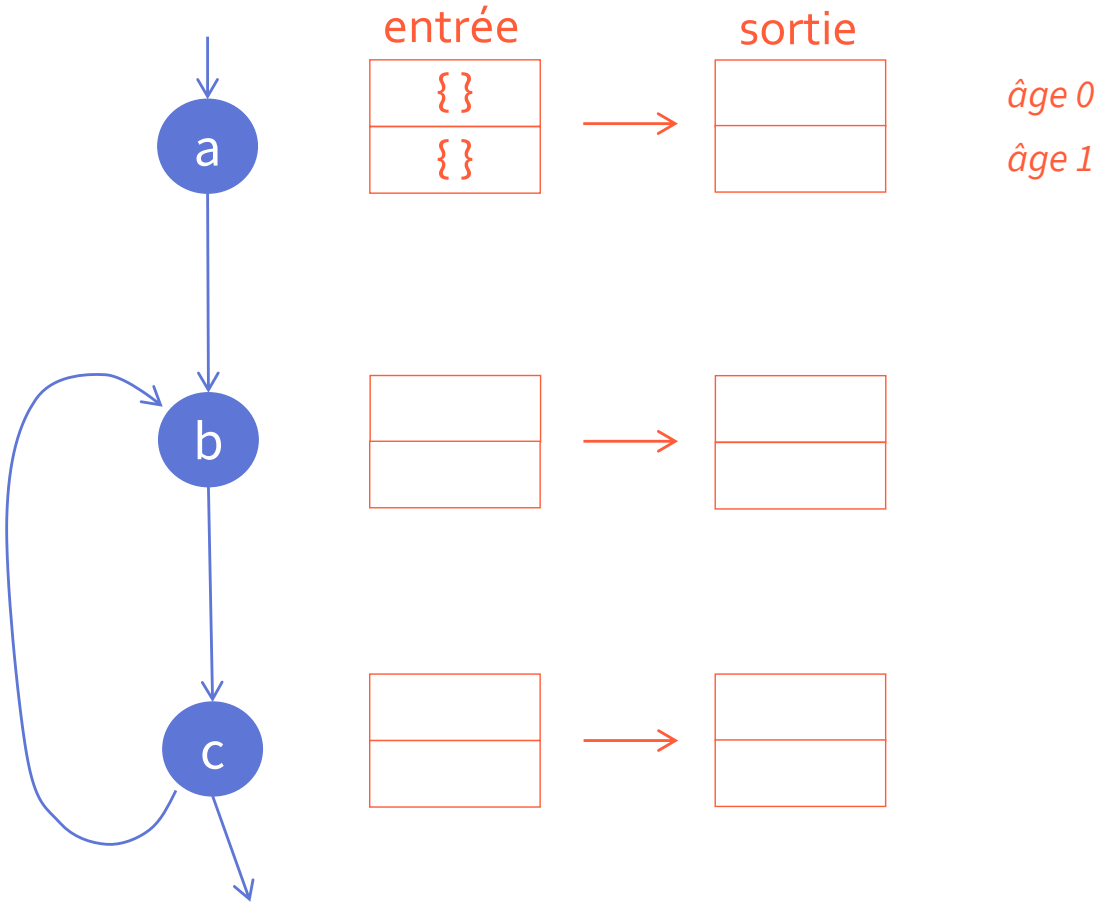
Analyse MAY

- fonction `join()`



Analyse du comportement du cache

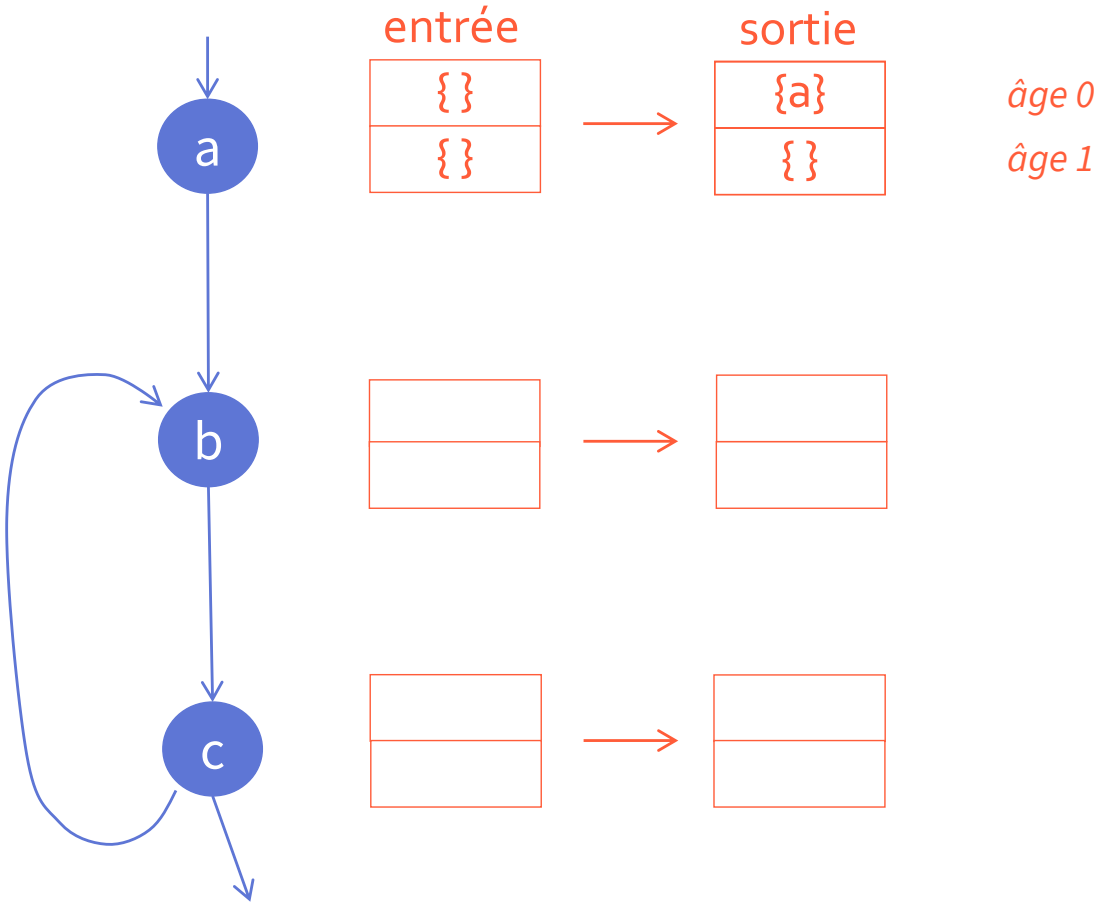
Analyse MAY



join = union & âge minimum

Analyse du comportement du cache

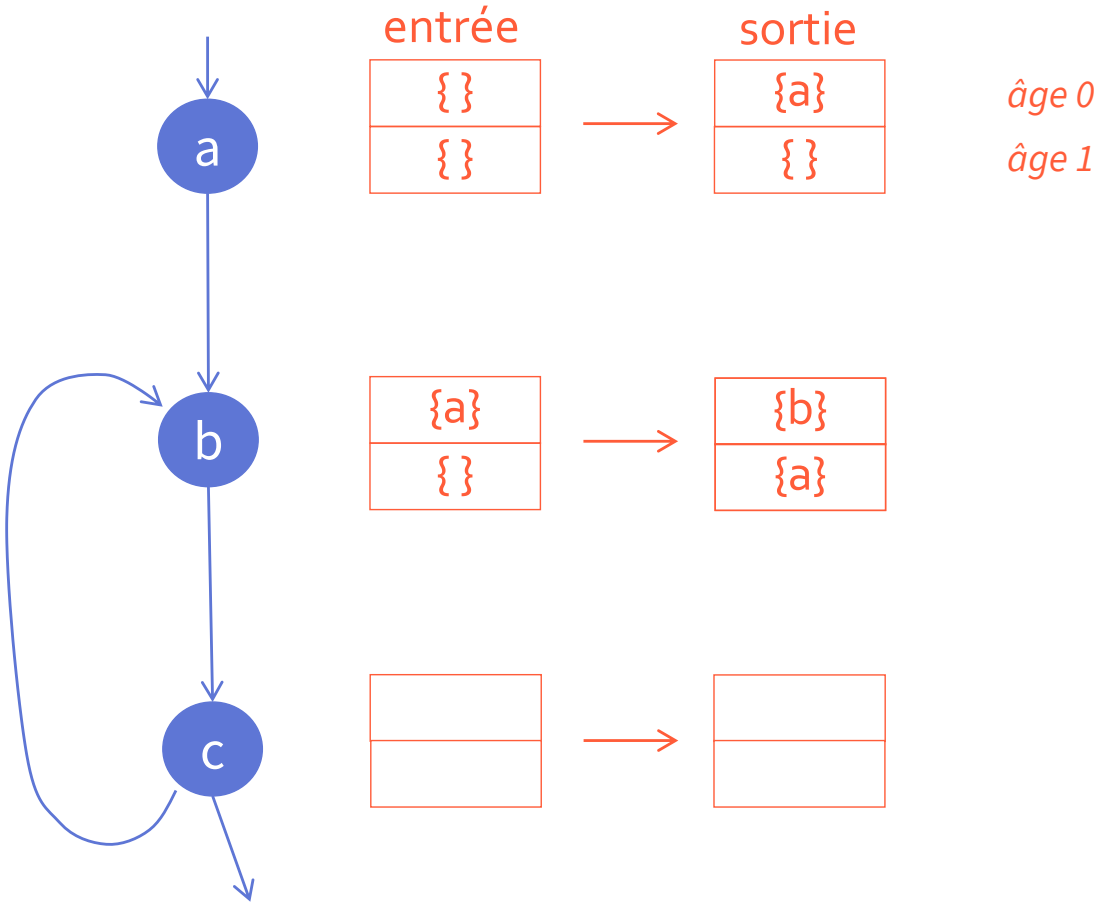
Analyse MAY



join = union & âge minimum

Analyse du comportement du cache

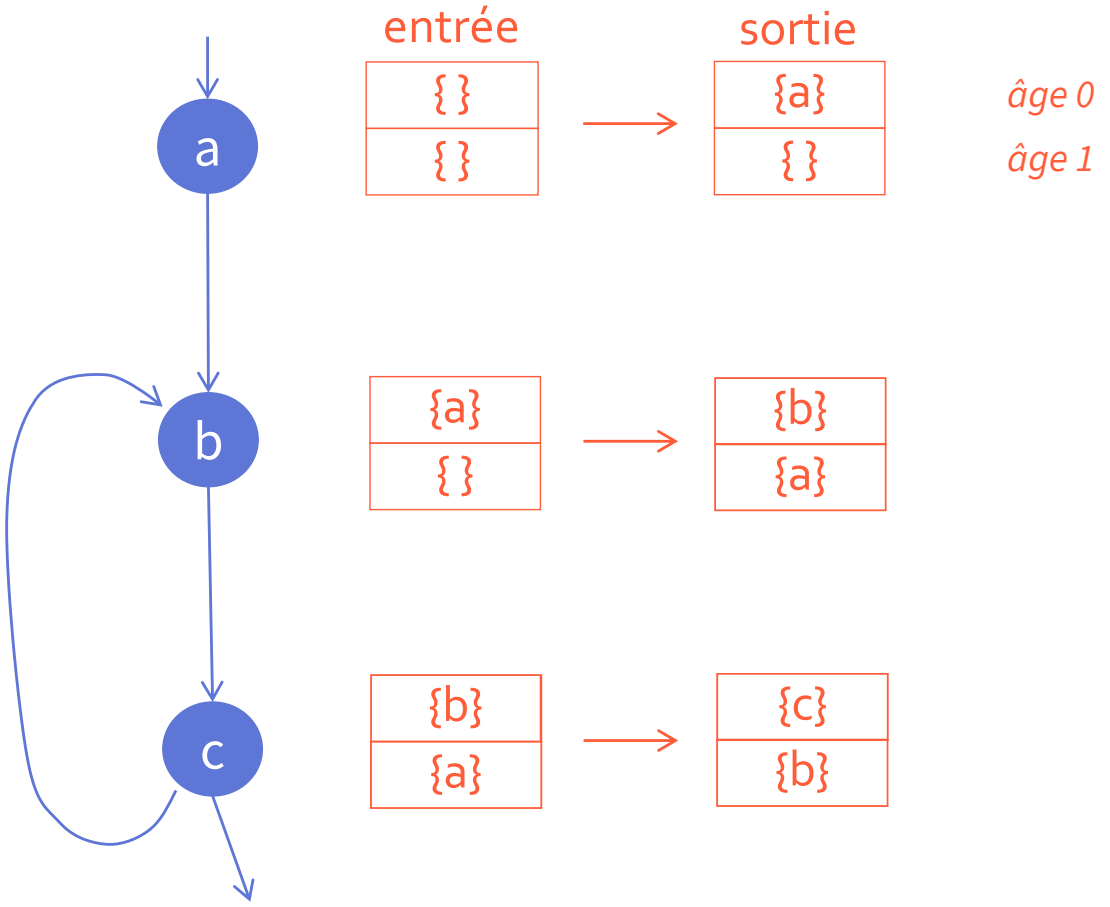
Analyse MAY



join = union & âge minimum

Analyse du comportement du cache

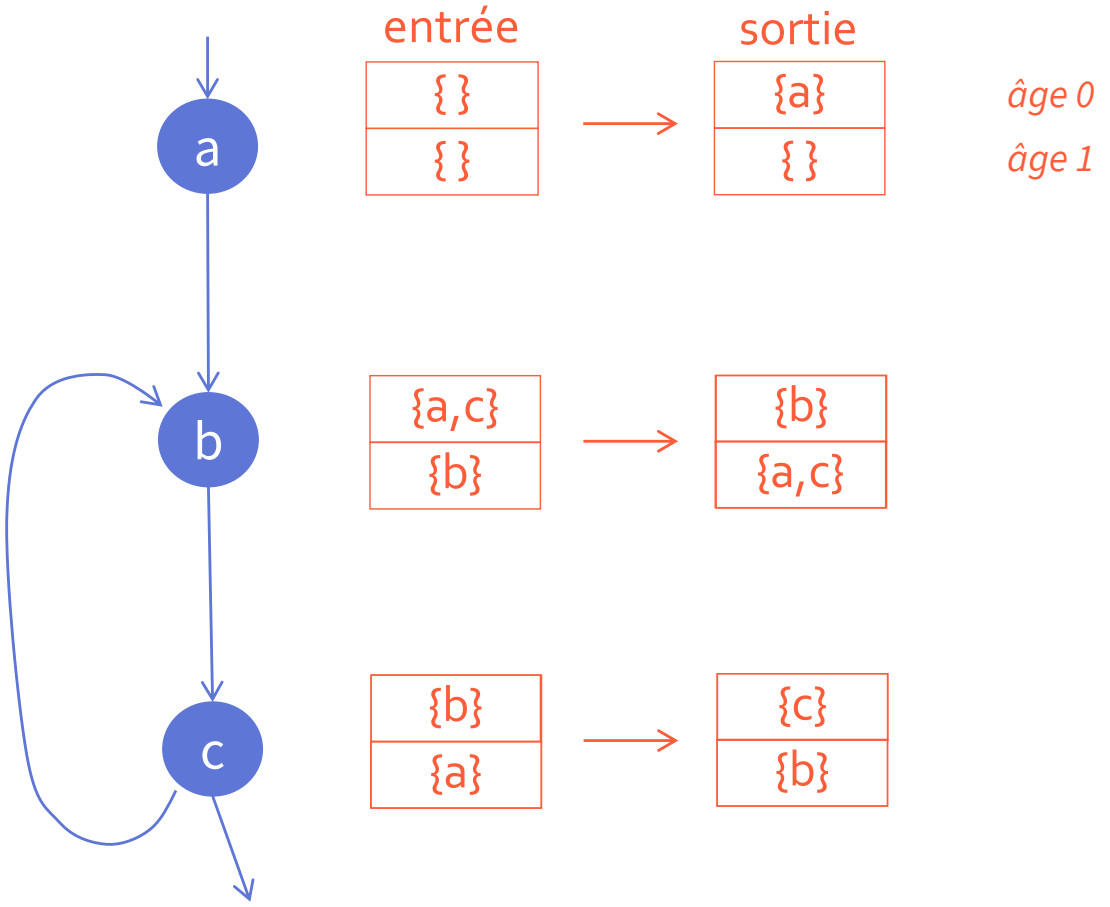
Analyse MAY



join = union & âge minimum

Analyse du comportement du cache

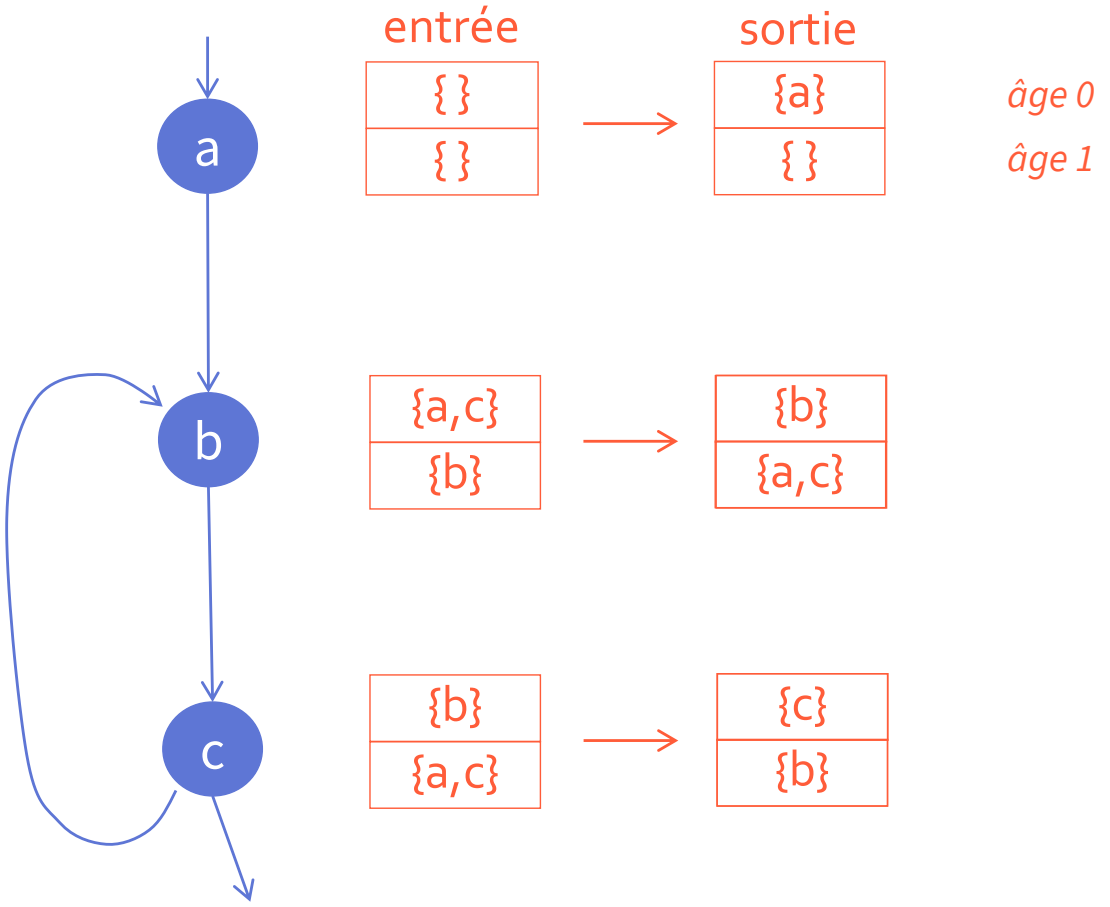
Analyse MAY



join = union & âge minimum

Analyse du comportement du cache

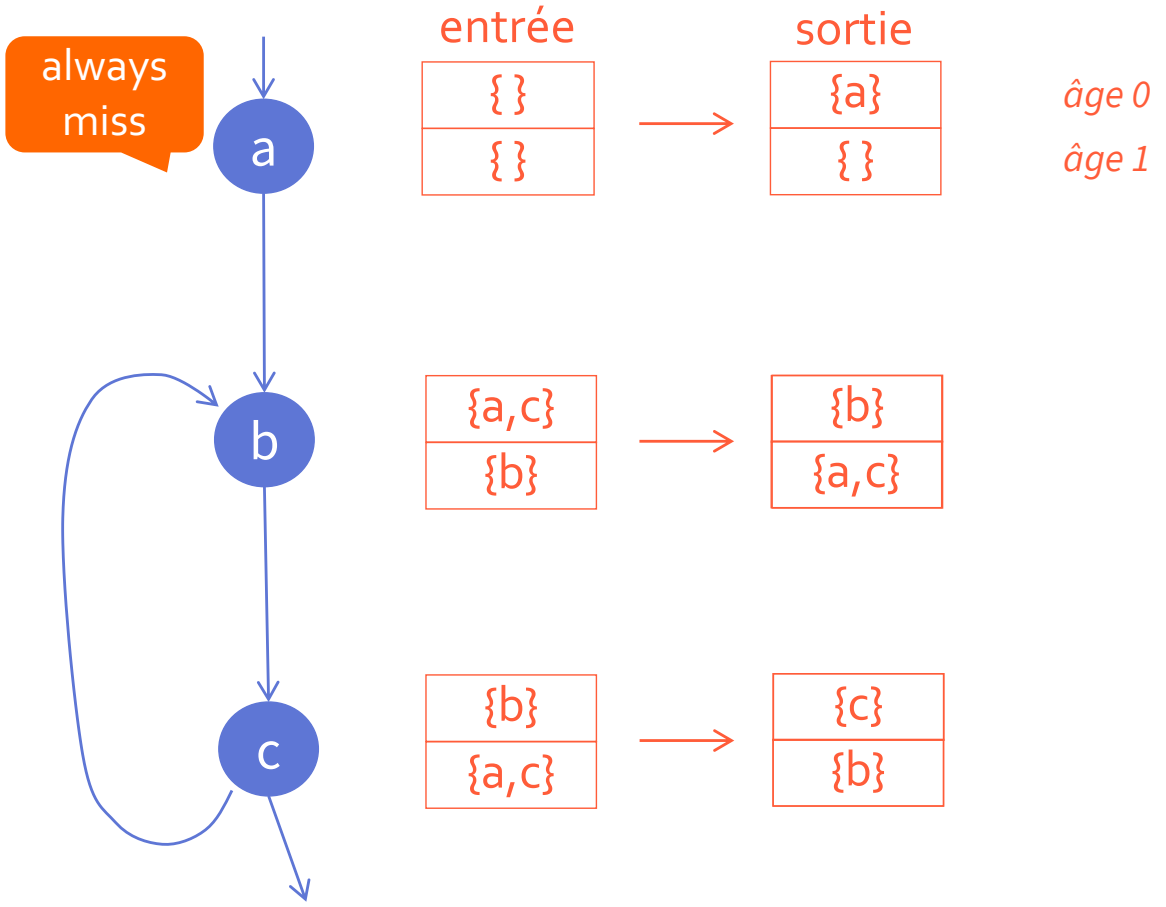
Analyse MAY



join = union & âge minimum

Analyse du comportement du cache

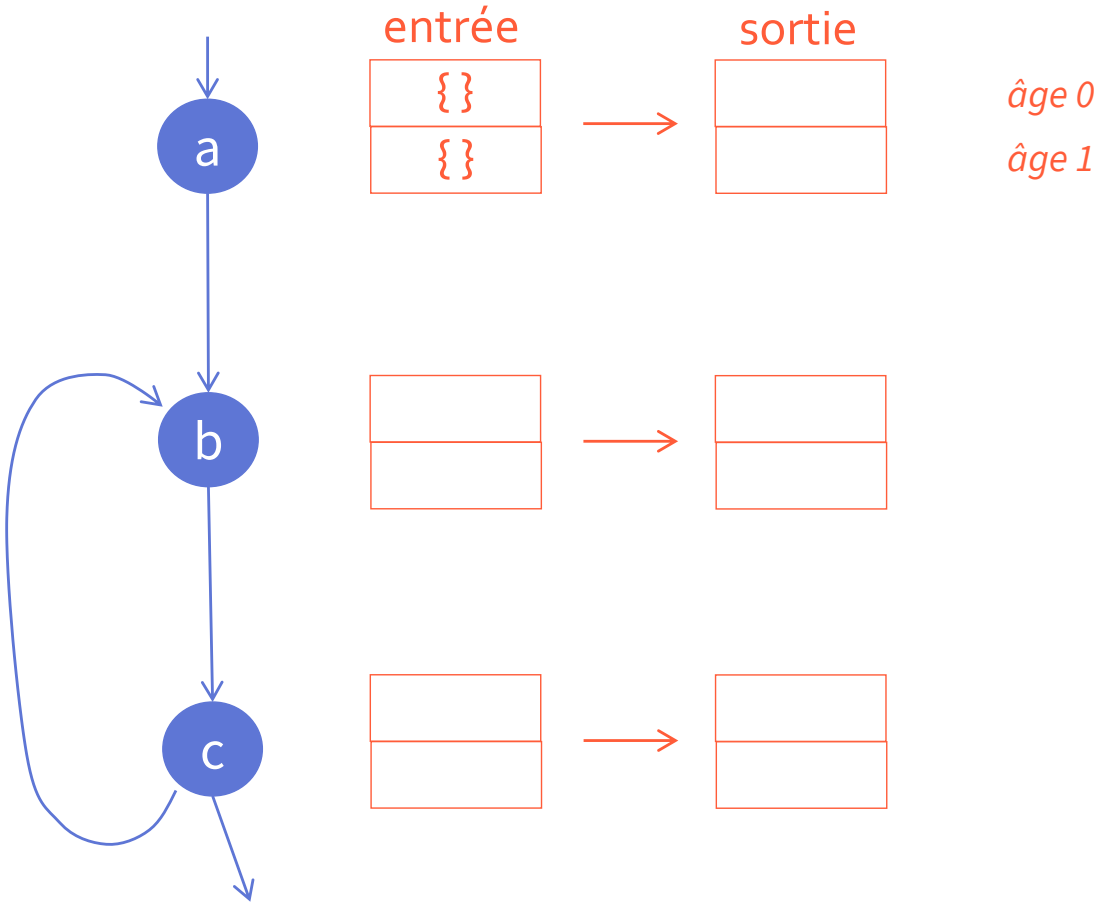
Analyse MAY



join = union & âge minimum

Analyse du comportement du cache

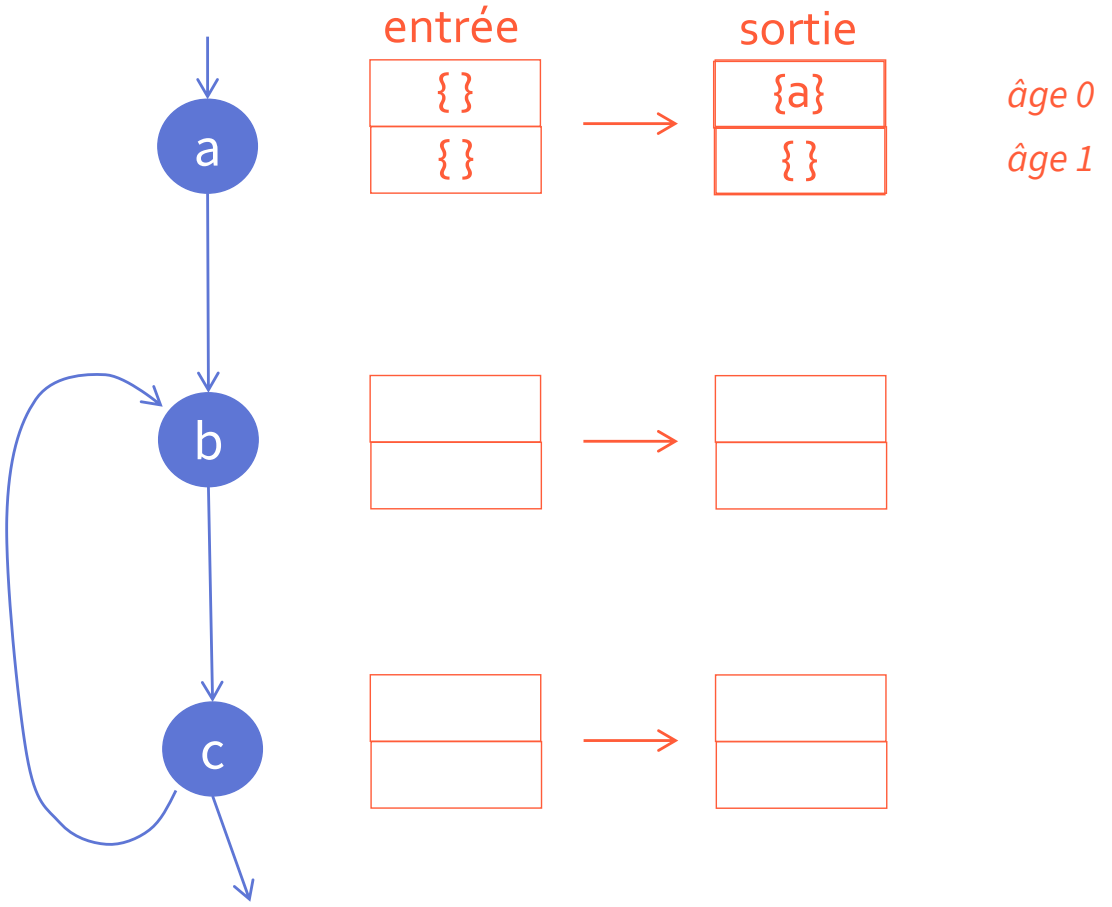
Analyse MUST



join = intersection & âge maximum

Analyse du comportement du cache

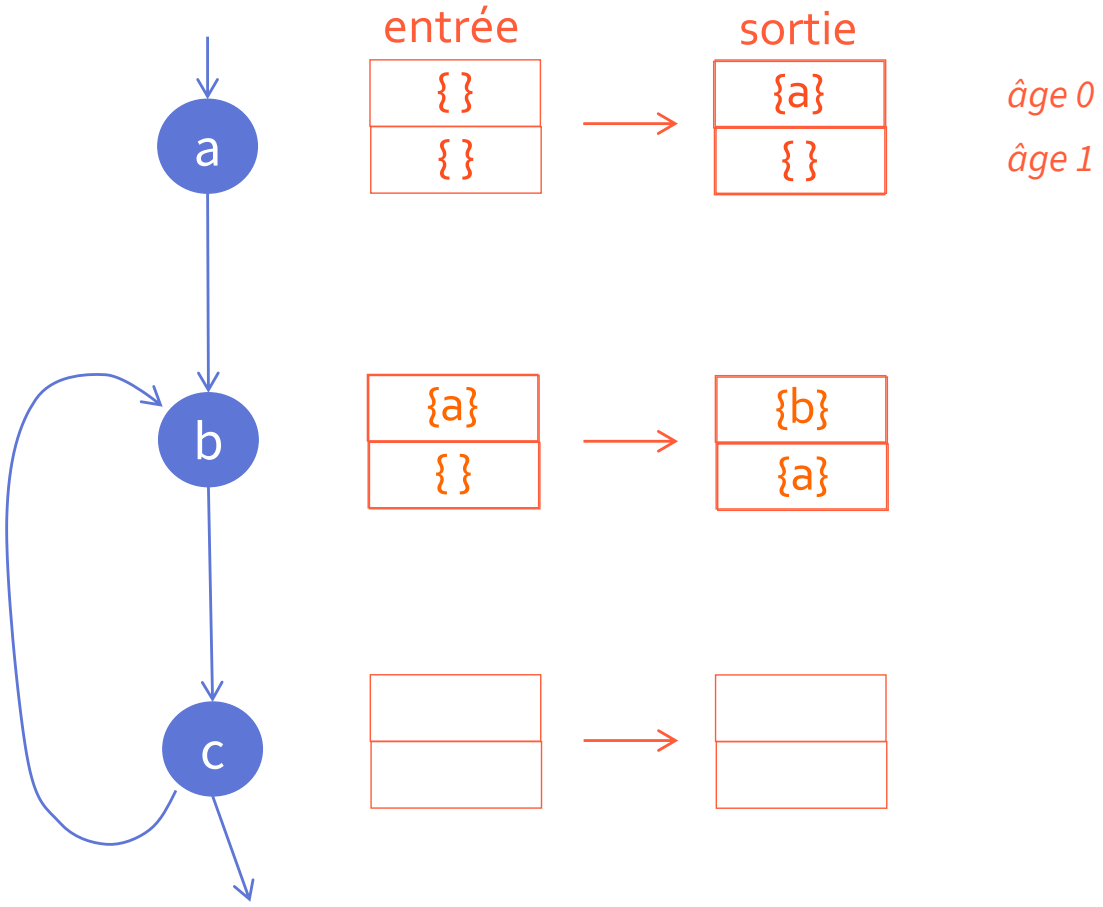
Analyse MUST



join = intersection & âge maximum

Analyse du comportement du cache

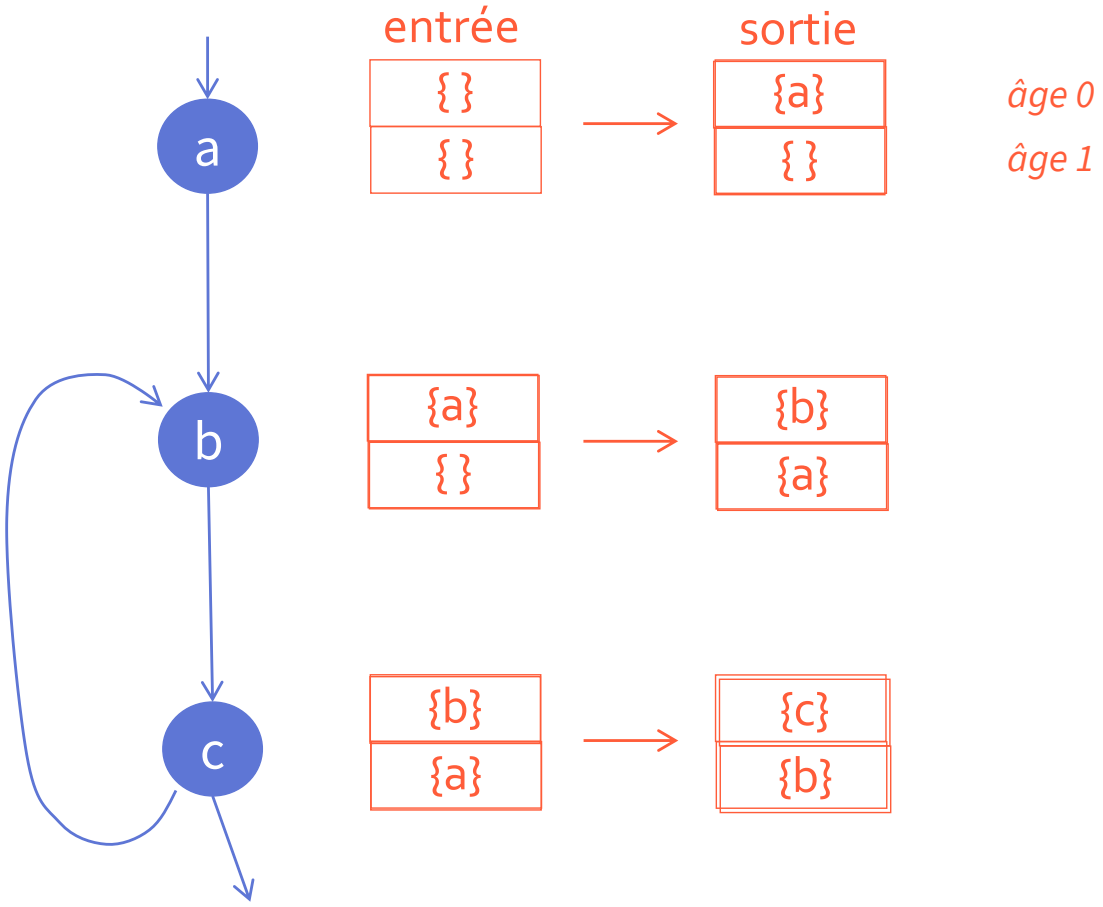
Analyse MUST



join = intersection & âge maximum

Analyse du comportement du cache

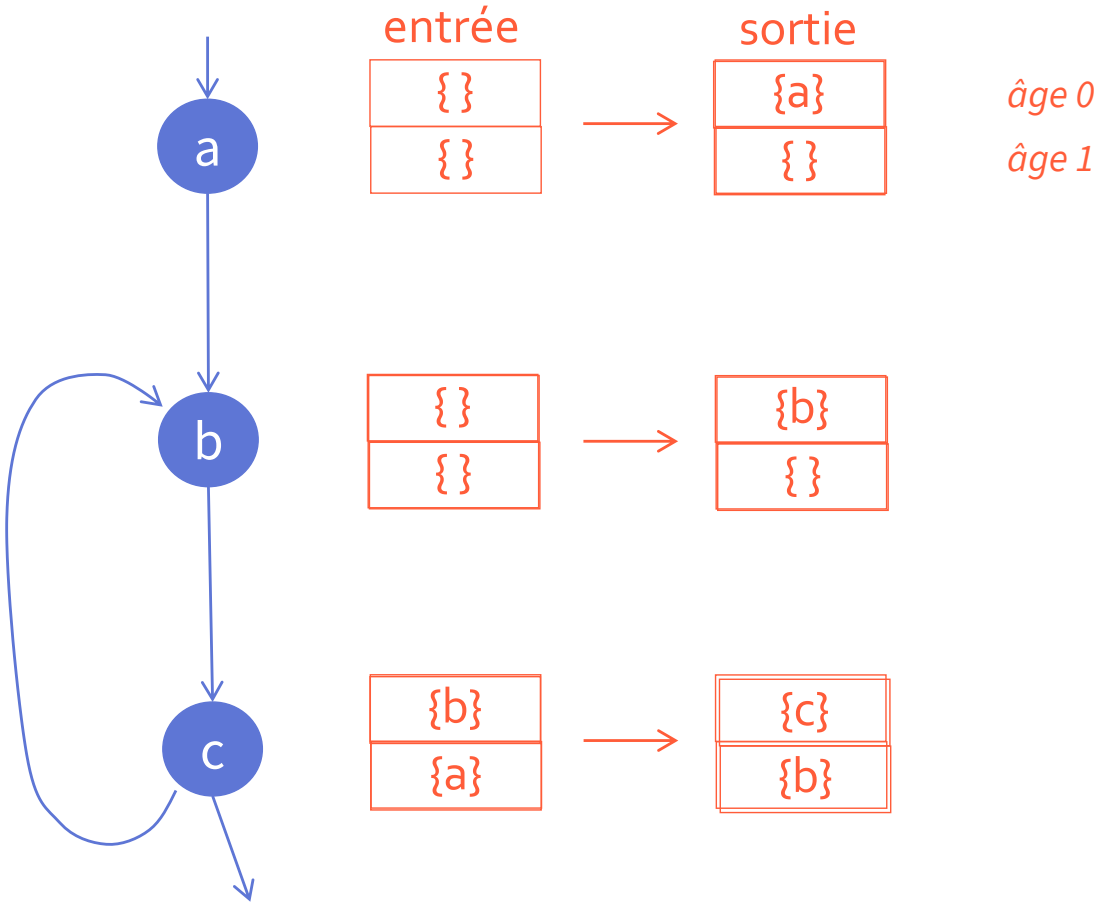
Analyse MUST



join = intersection & âge maximum

Analyse du comportement du cache

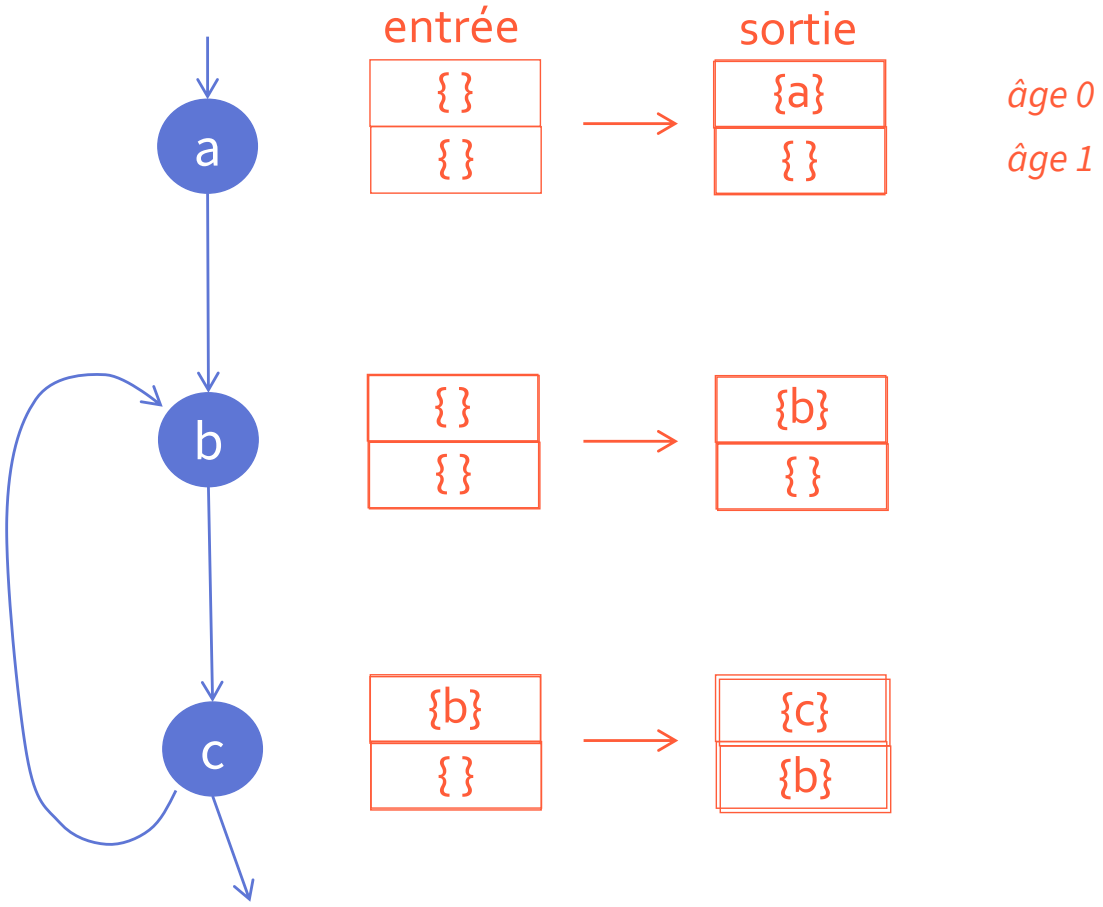
Analyse MUST



join = intersection & âge maximum

Analyse du comportement du cache

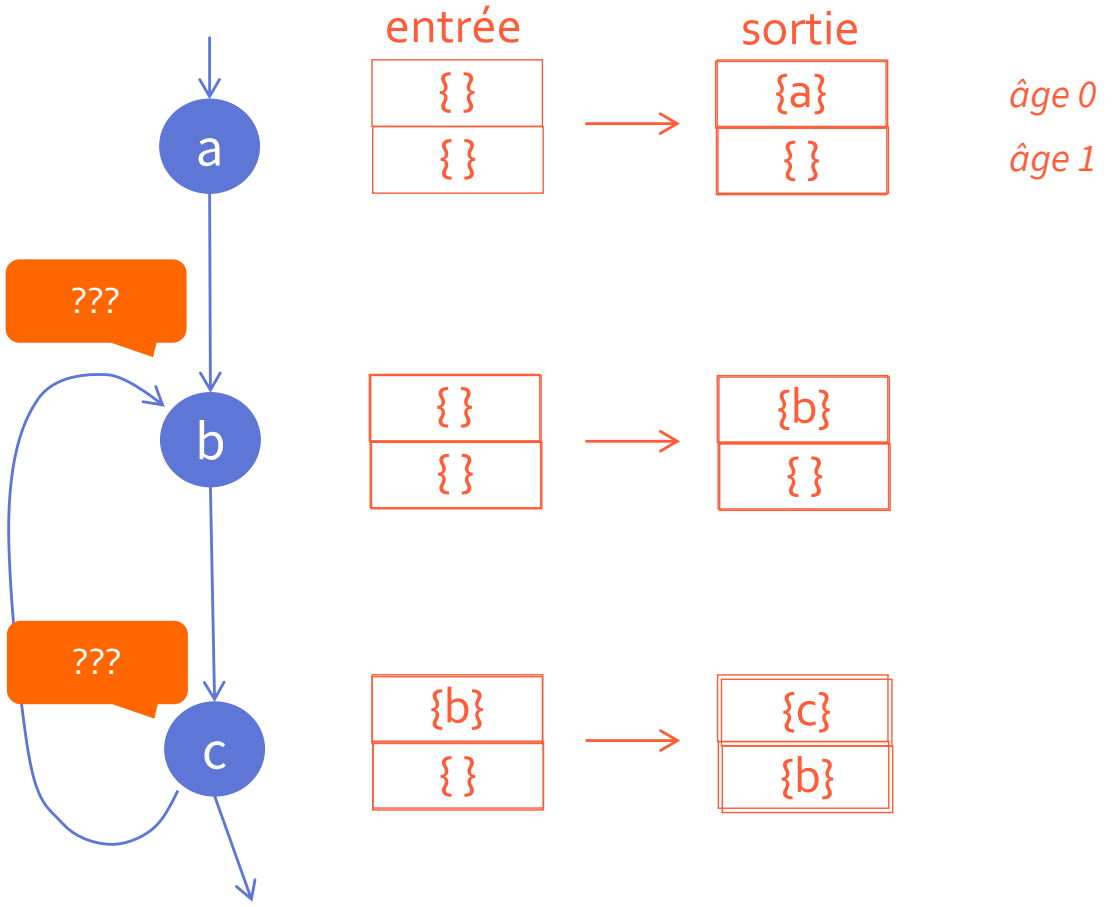
Analyse MUST



join = intersection & âge maximum

Analyse du comportement du cache

Analyse MUST



Analyse du comportement du cache

Analyse de PERSISTENCE

- état abstrait d'un ensemble = âges maximum + âge spécifique pour les blocs remplacés

âge	
0	{ }
1	{a, b}
2	{ }
3	{c}
remplacé	{e,f}

Analyse du comportement du cache

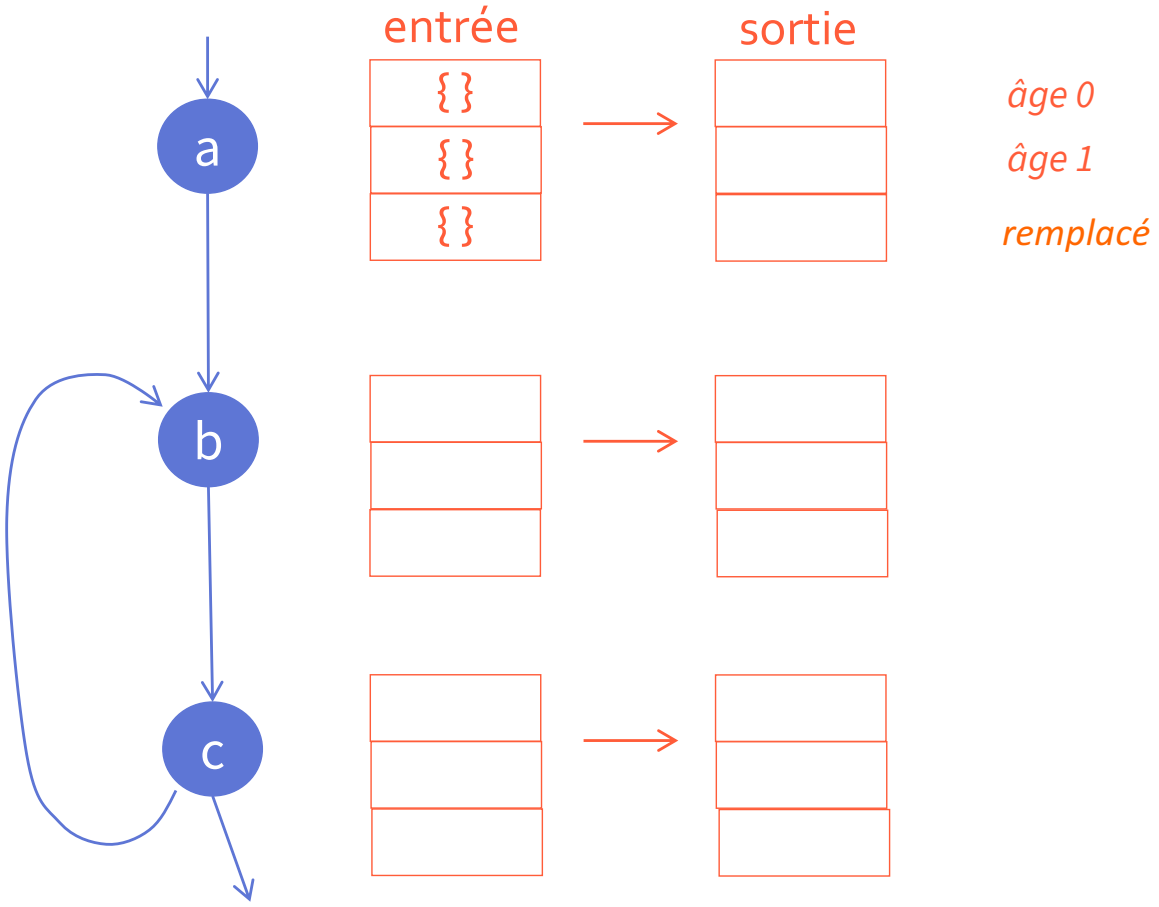
Analyse de PERSISTENCE

- `update()`
  - comme pour l'analyse MUST
- `join()`
  - union & âge maximum

- détermine les blocs qui ne peuvent pas être éjectés (remplacés) une fois qu'ils ont été chargés dans le cache

Analyse du comportement du cache

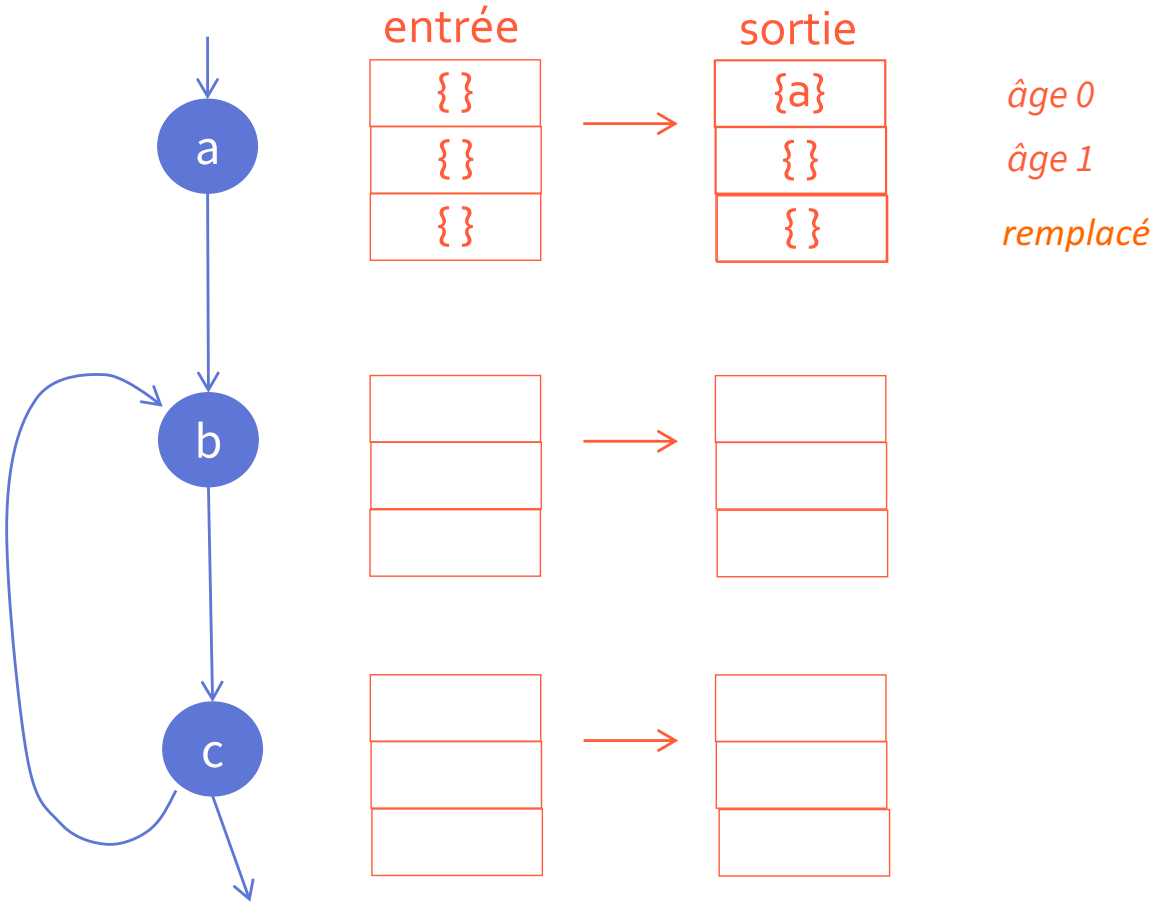
PERSISTENCE



join = union & âge maximum

Analyse du comportement du cache

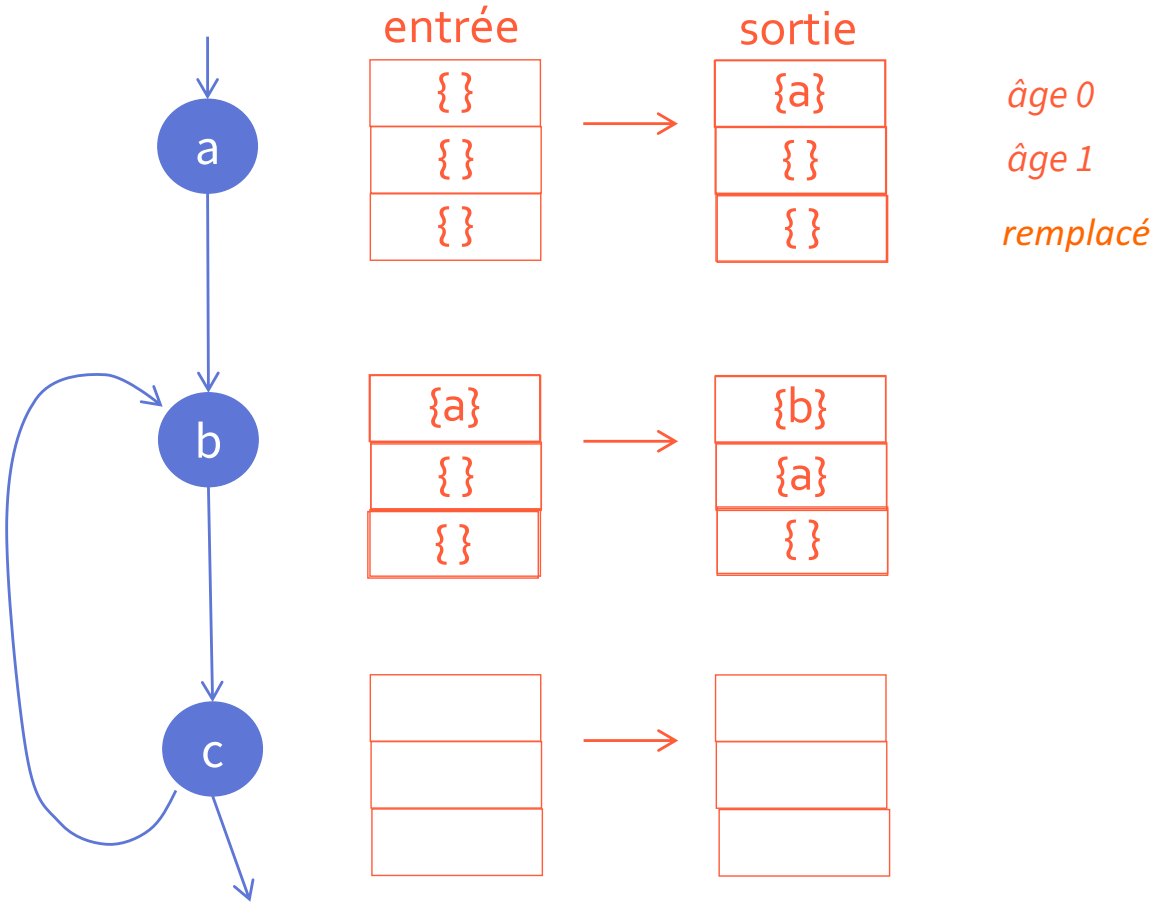
PERSISTENCE



join = union & âge maximum

Analyse du comportement du cache

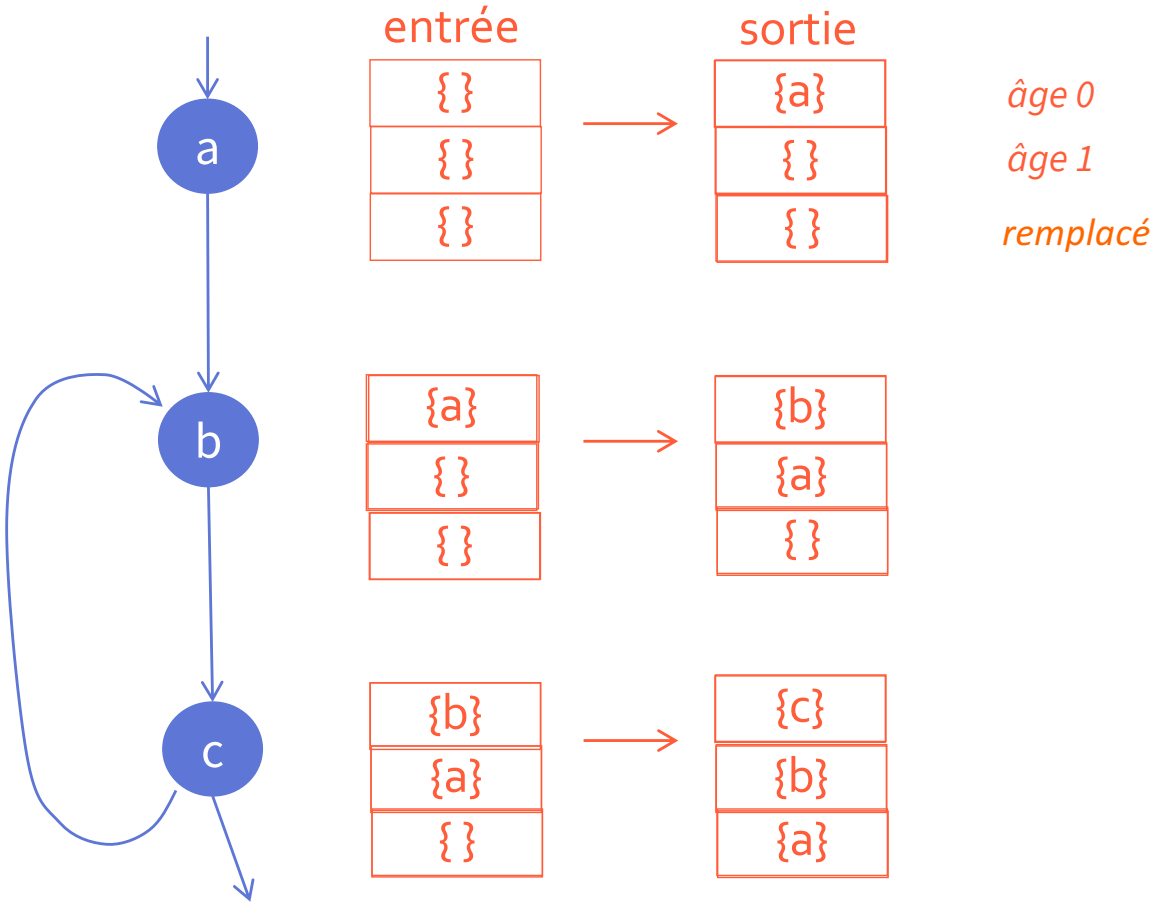
PERSISTENCE



join = union & âge maximum

Analyse du comportement du cache

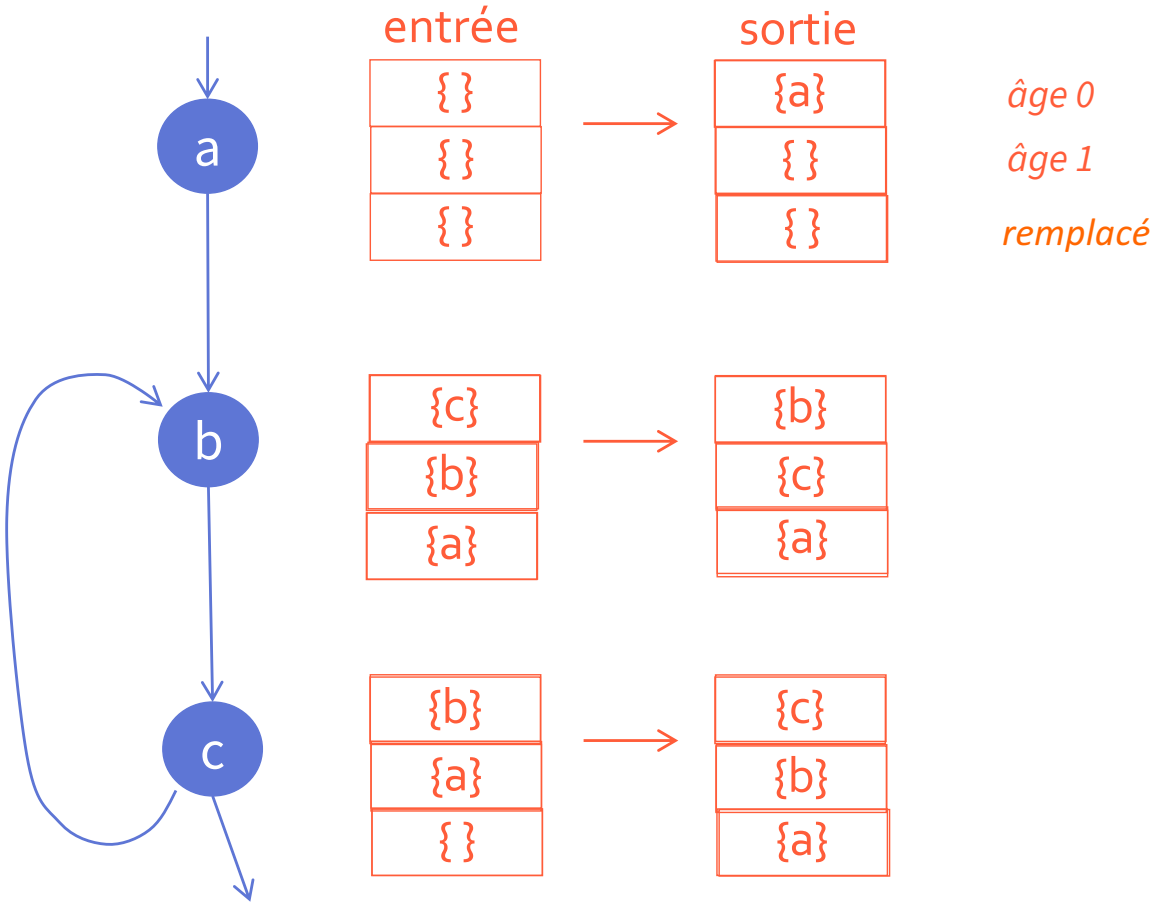
PERSISTENCE



join = union & âge maximum

Analyse du comportement du cache

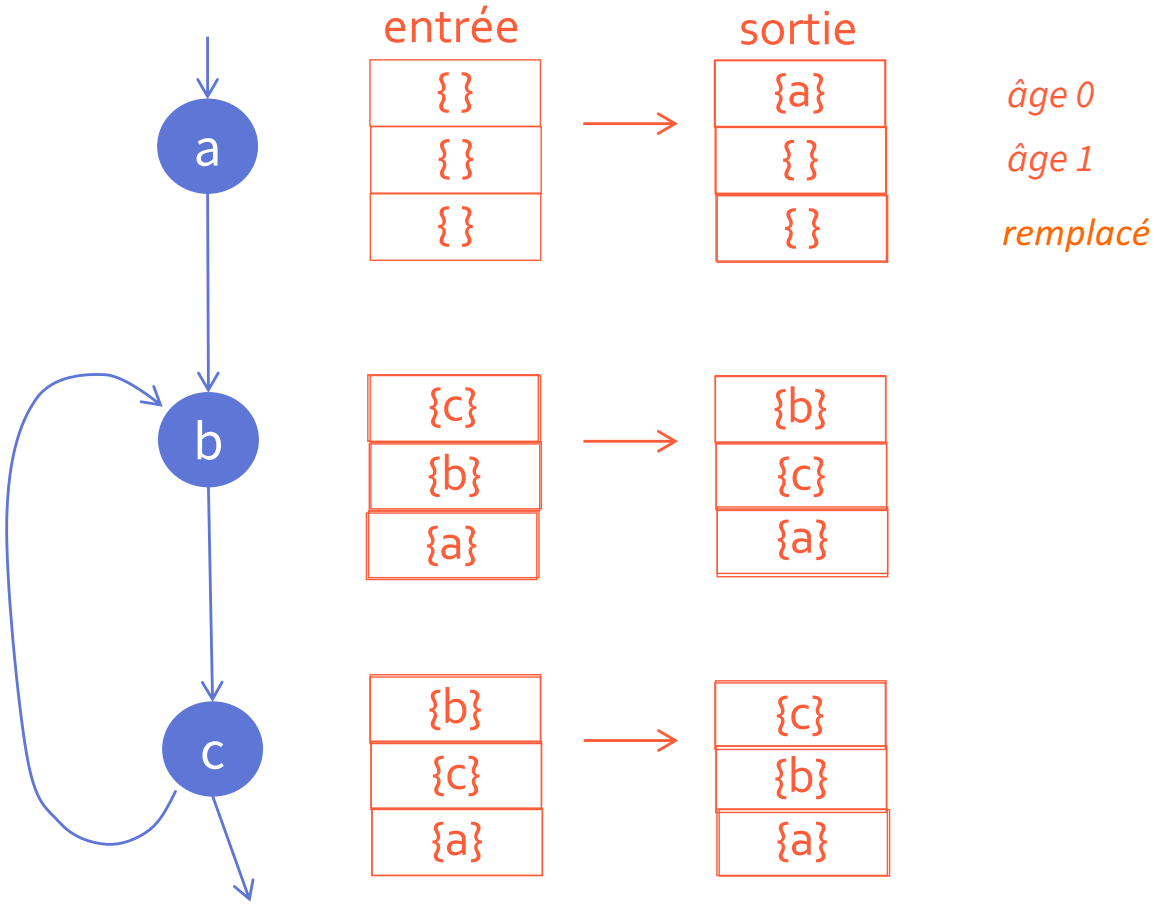
PERSISTENCE



join = union & âge maximum

Analyse du comportement du cache

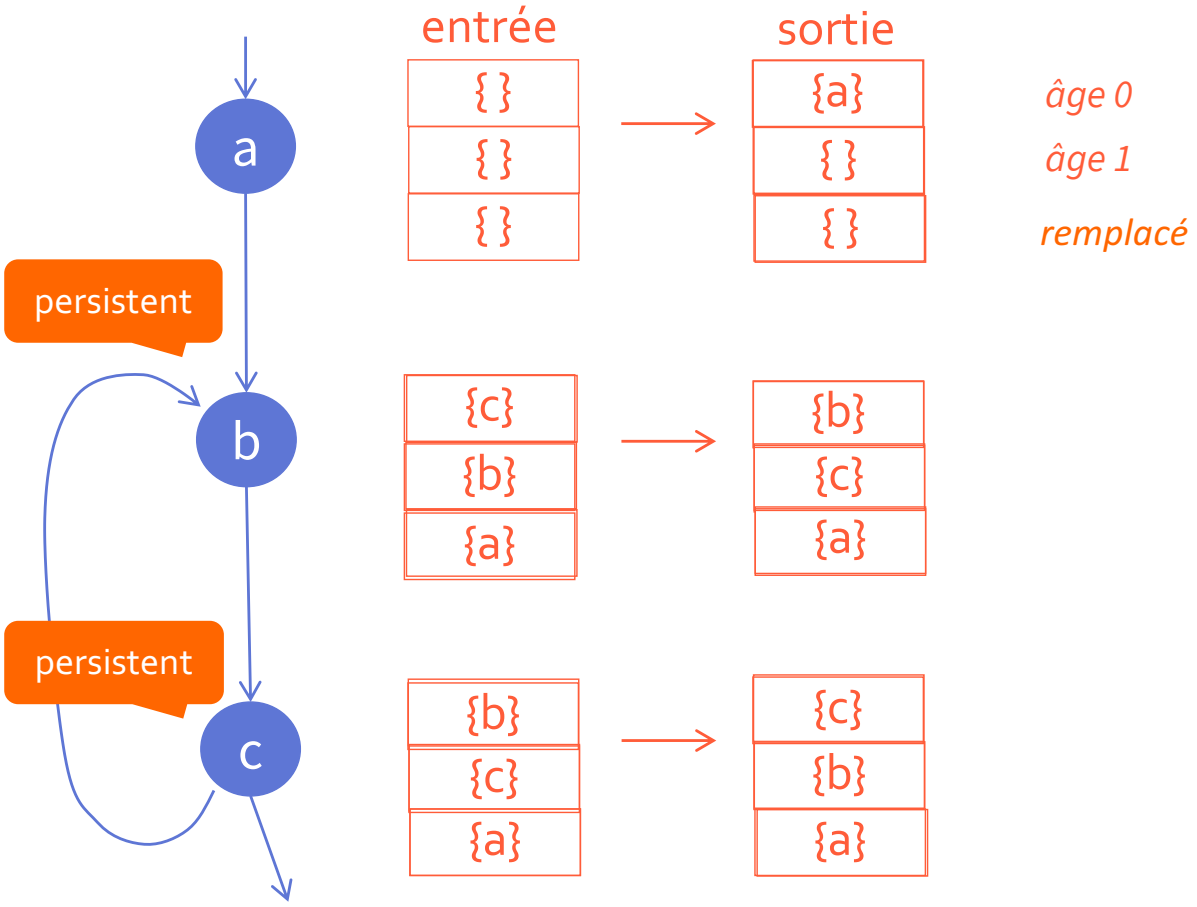
PERSISTENCE



join = union & âge maximum

Analyse du comportement du cache

PERSISTENCE



join = union & âge maximum

Analyse du cache de données

Principal défi :

- adresses cibles inconnues et/ou multiples

```
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
```

Analyse du cache de données

Analyse des adresses

Ramaprasad, Mueller, *Bounding Worst-Case Data Cache Behaviour by Analytically Deriving Cache Reference Patterns,* RTAS 2005

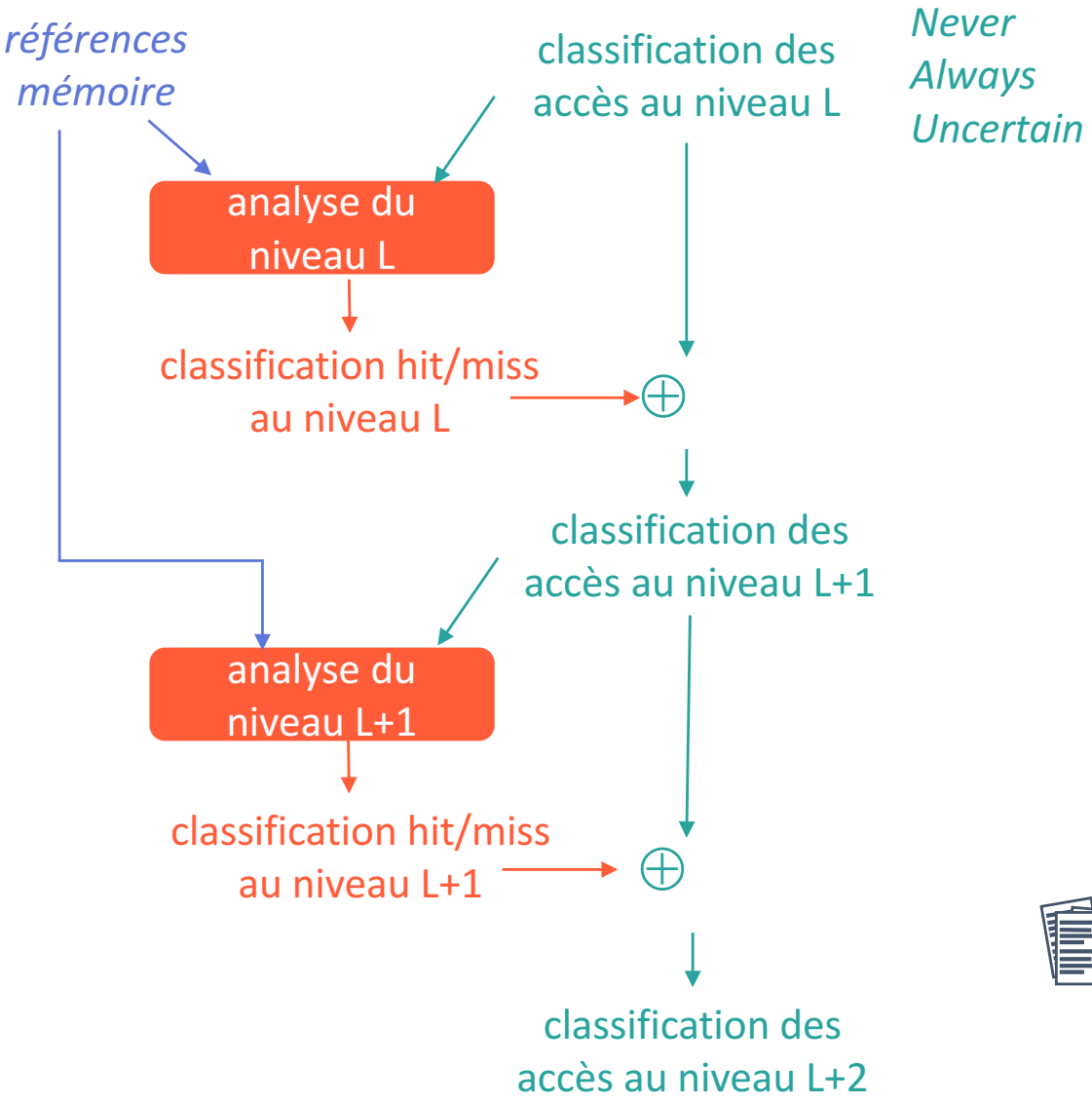
Huynh, Ju, Roychoudhury, *Scope-aware Data Cache Analysis for WCET Estimation,* RTAS 2011

Hahn, Grund, *Relational Cache Analysis for Static Timing Analysis,* ECRTS 2012

Impact des écritures

Blaß, Hahn, Reineke, *Write-Back Caches in WCET Analysis,* ECRTS 2017

Analyse d'une hiérarchie de caches



Hardy, Puaut, *WCET Analysis of Multi-Level Non-Inclusive Set-Associative Instruction Caches*, RTSS 2008

Verrouillage de cache

Objectif :

- rendre la latence des accès mémoire plus prévisible

Défis :

- mise en oeuvre
  - sélectionner le contenu à verrouiller
  - insérer des instructions de chargement et de verrouillage
- surcoût temporel
  - routines de verrouillage, défauts additionnels

Mittal, *A Survey of Techniques for Cache Locking*,
ACM Trans. on Design Automation of Electronic Systems 21(3), 2016

Verrouillage de cache

Différentes approches

- verrouillage statique ou dynamique
- complet ou partiel
- verrouillage de ligne ou de voie

- stratégies pour sélectionner le contenu :
  - algorithmes génétiques
  - programmation linéaire
  - algorithmes gloutons

Verrouillage de cache

Exemple

Puaut, *WCET-centric Software-controlled Instruction Caches for Hard Real-Time Systems*, ECRTS, 2006

- Points de rechargement : entrées de fonctions et de boucles

- Algorithme glouton
  - sélection du contenu fondé sur la fréquence des instructions sur le chemin le plus long (WCEP)
  - algorithme itératif qui met à jour régulièrement le WCEP et sélectionne les blocs qui tirent le plus bénéfice du verrouillage de leurs instructions

- Algorithme génétique
  - chromosome = paire (point de chargement, contenu à charger)
  - fonction fitness = WCET
  - crossover = échanger un chromosome aléatoire entre les parents
  - mutations = suppression ou ajout d'un point de chargement, modification du contenu du cache (aléatoirement, un nombre fixe de bloc smémoire)
  - population initiale = nombre fixe de points de chargement choisis aléatoiremet et contenu de cache aléatoire

Verrouillage de cache

Références

Campoy et al., *Cache Contents Selection for Statically-locked Instruction Caches: an Algorithm Comparison*, ECRTS 2005

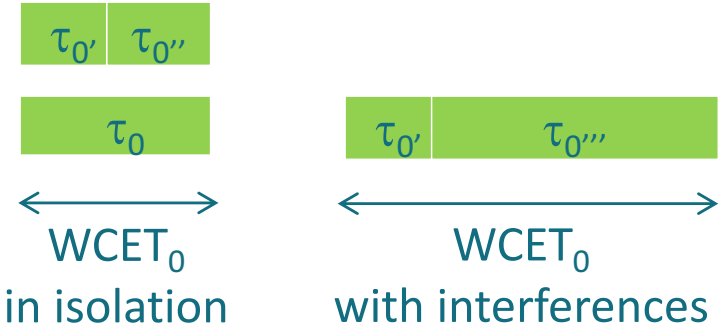
Ding, Liang, Mitra, *WCET-centric Partial Instruction Cache Locking*, DAC 2012

Ding, Liang, Mitra, *WCET-centric Dynamic Instruction Cache Locking*, DAC 2014

Zheng, Wu, *WCET-aware Dynamic D-cache Locking for a Single Task*, LCTES 2015

Interférences avec d’autres tâches

Impact de la préemption



Calcul du coût de la préemption vis-à-vis du cache

- CRPD : *Cache-Related Preemption Delays*
- quels sont les blocs qui vont (peut-être) être éjectés par la tâche préemptrice et devront être rechargés dans le cache ?

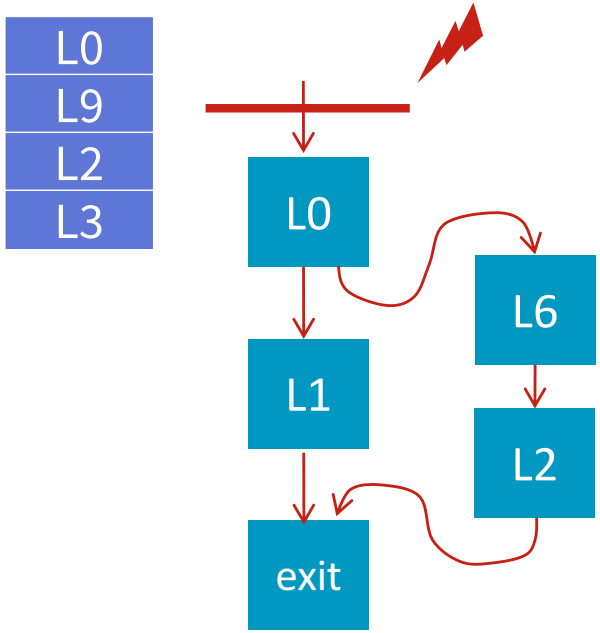
Lee et al. *Analysis of Cache-Related Preemption Delays in Fixed-priority Preemptive Scheduling,* IEEE Trans. on Computers 47(6)), 1998

Analyse des blocs utiles

Analyse des blocs utiles (*Useful Cache Blocks*)

- susceptibles d'être dans le cache au point P
- peuvent être réutilisés au point Q, qui peut être atteint depuis P, sans être remplacés sur ce chemin (hormis par préemption)

contenu du cache (à correspondance directe) avant le point de préemption



UCBs ?

- L0 est dans le cache et sera réutilisé
- L1 n'est pas dans le cache
- L2 est dans le cache mais ne sera pas réutilisé

UCBs = {L0}

Calcul des CRPDs

Useful Cache Blocks (UCBs)

- leur nombre est une borne supérieure du nombre de défauts de cache supplémentaires dus à la préemption
- pour une préemption au point P:

$$CRPD_{UCB}(P) = \sum_{cache\ sets} \min(|UCB(P)|, a) \times t$$

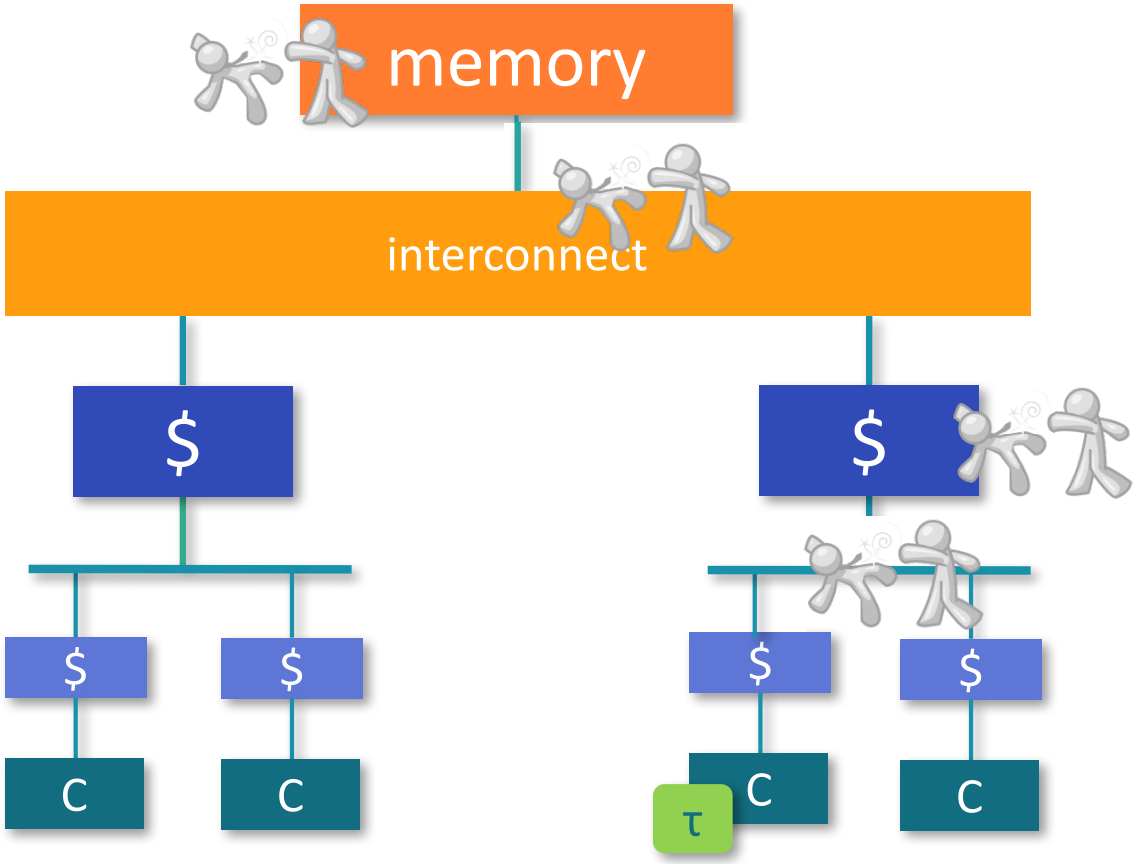
- pour une préemption en n'importe quel point :

$$CRPD_{UCB} = \max_{P} CRPD_{UCB}(P)$$

Evicting Cache Blocks (ECBs)

- seuls les blocs référencés par une tâche préemptrice peuvent déteriorer le contenu du cache…

Interférences dans une architecture multi-cœurs



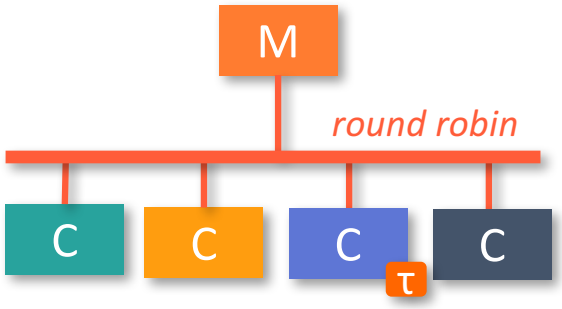
Estimation des interférences

Analyse intégrée

- considère l'entrelacement précis des accès à la ressource partagée
- très (trop) complexe

Analyse conservatrice

- pessimiste



Analyse du coût des interférences

- suppose la *compositionnalité*
  - $WCET = WCET_{isolation} + coût\_interférences$
- vise à considérer le nombre « réel » d'interférences pour limiter le pessimisme

Approches faisant l'hypothèse de compositionnalité

Pellizzoni et al., *Worst-case Delay Analysis for Memory Interference in Multicore Systems,* DATE 2010

Schliecker, Negrean, Ernst, *Bounding the Shared Resource Load for the Performance Analysis of Multiprocessor Systems,* DAC 2010

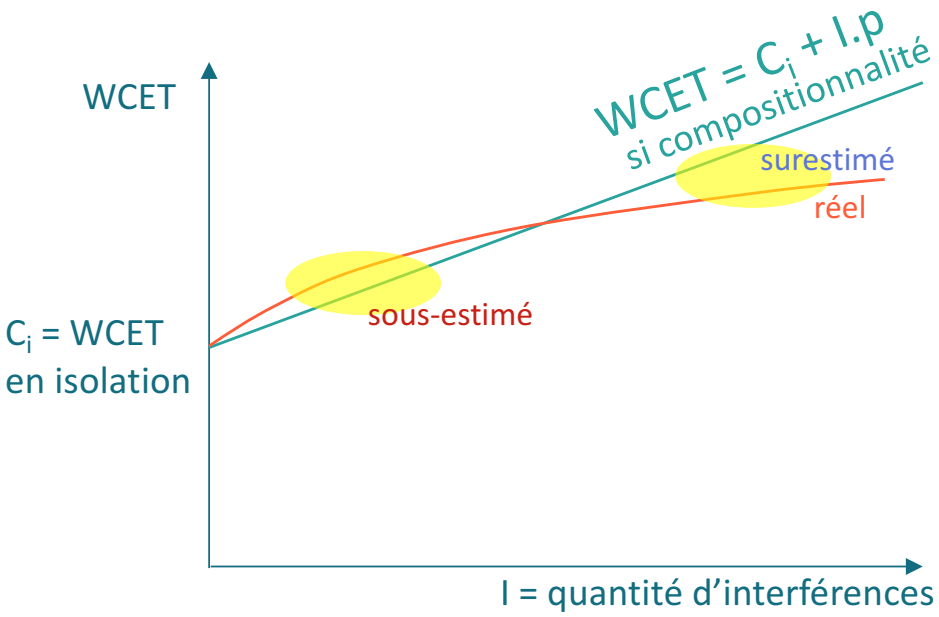
Kim et al., *Bounding Memory Interference Delay in COTS-based Multicore Systems,* RTAS 2014

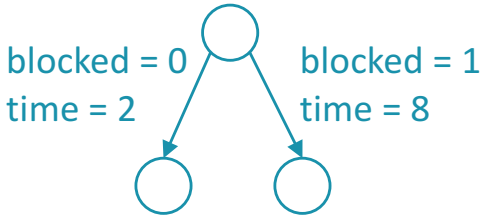
Altmeyer et al., *A Generic and Compositional Framework for Multicore Response Time Analysis,* RTNS 2015

Dasari, Nelis, Akesson, *A Framework for Memory Contention in Multi-core Platforms,* Journal of Real-Time Systems 52(3), 2016

Analyse compositionnelle

Hahn, Jacobs, Reineke
Enabling Compositionality for Multicore Timing Analysis, RTNS 2016



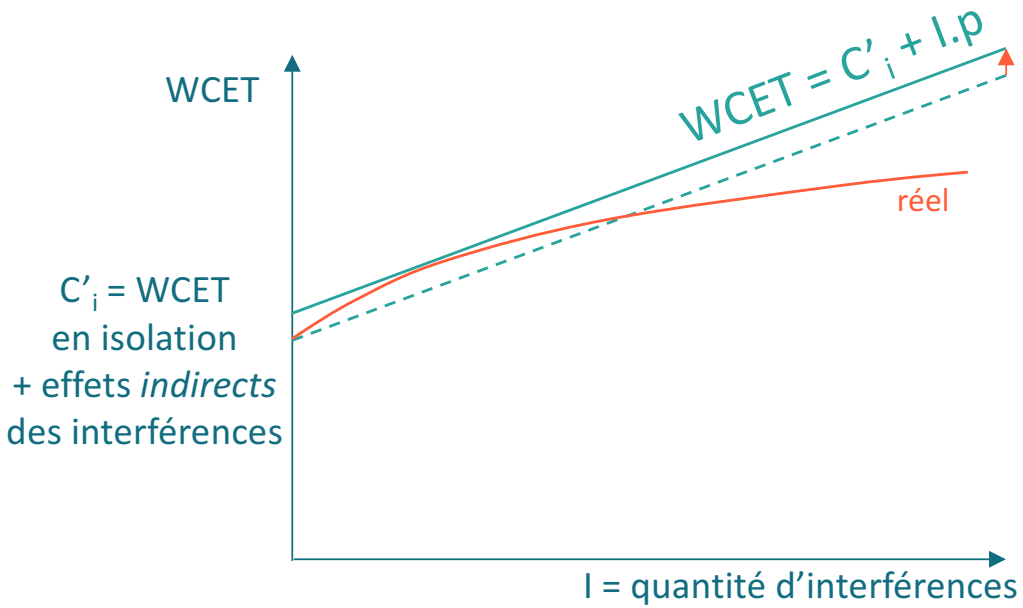


$$\max \sum_e time_e . x_e$$

$$\sum_e blocked . x_e \leq I$$

Analyse compositionnelle

Idée : calculer un WCET de base « compositionnel »

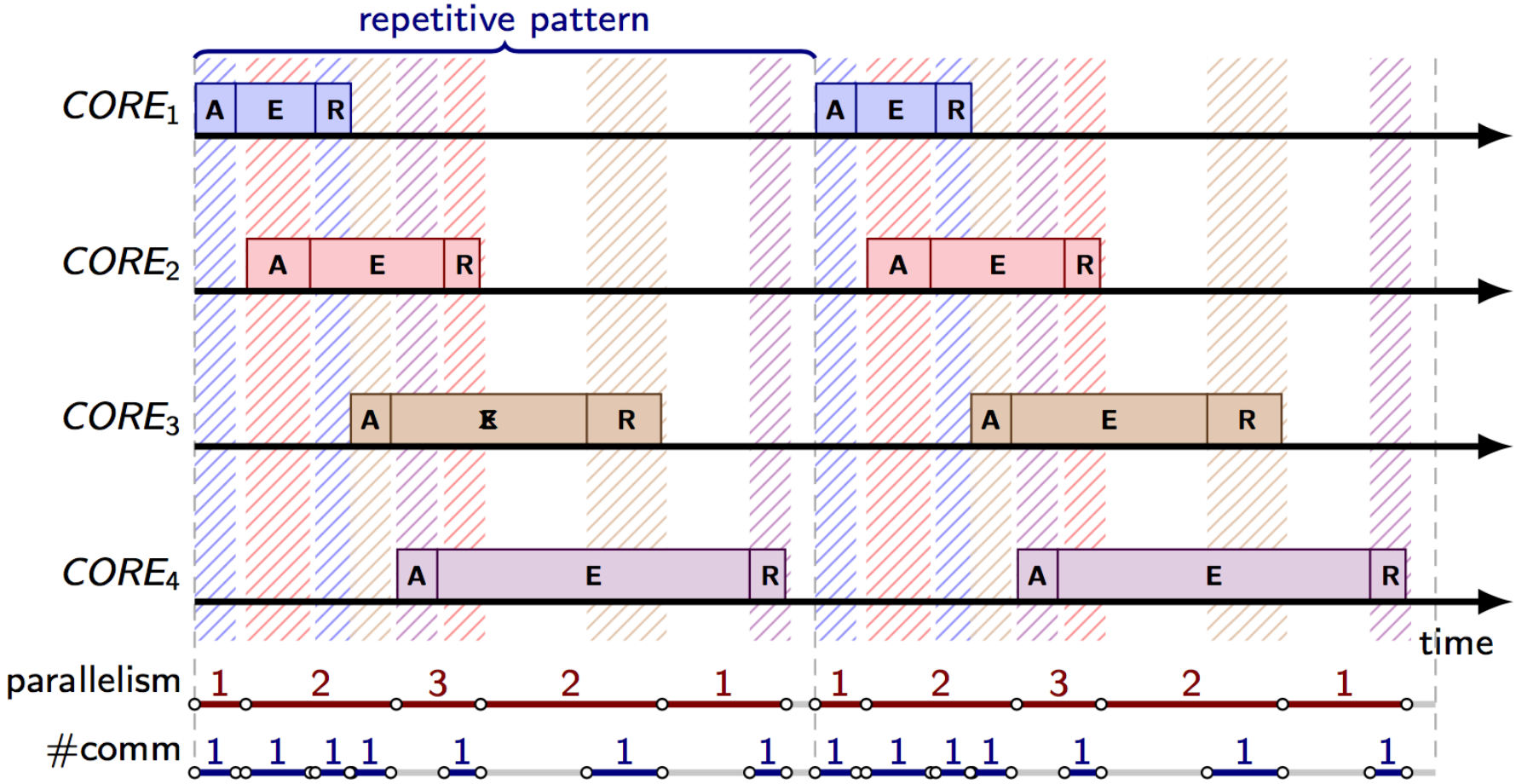


$$\sum_{e} blocked.x_e \leq i$$

$$\max\left(\sum_{e} time_e.x_e\right) - p.i$$

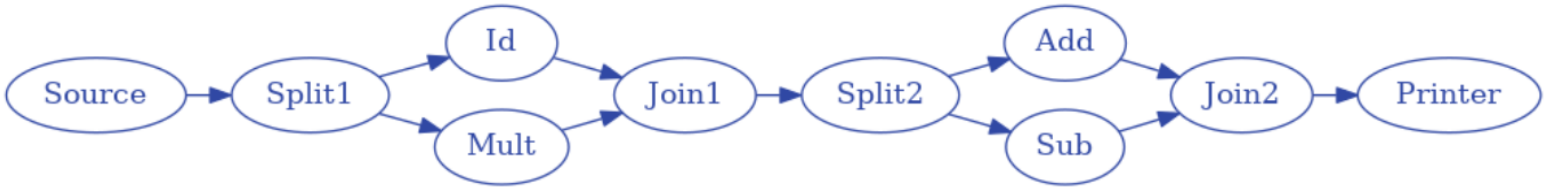
Modèle de calcul prévisible

Durrieu et al., *Predictable Flight Management System Implementation on a Multicore Processor*, ERTS 2014



A = Acquisition / E = Exécution / R = Restitution

Modèle de calcul prévisible



graphe de tâches dépendantes

Objectif

- minimiser le temps de réponse global en contrôlant les interférences via le *mapping* et l'ordonnancement des tâches sur les cœurs

Deux approches

- formulation ILP
- heuristique de type « ordonnancement par liste »

Rouxel, Derrien, Puaut, *Tightening contention delays while scheduling parallel applications on multi-core architectures*, EMSOFT 2017

Perspectives


TIME FOR QUESTIONS
12
1
2
3
4