Run-time Control to Increase Task Parallelism in Mixed-Critical Systems

Angeliki Kritikakou[1], Olivier Baldellon[2], Claire Pagetti[1],

Christine Rochange[3] and Matthieu Roy[2]

10/07/2014, ECRTS'14, Madrid, Spain

1 ONERA
Toulouse, France

2 LAAS-CNRS
Toulouse, France

3 IRIT Institut de Recherche en Informatique de Toulouse CNRS INPT UPS UT1
Toulouse, France

Outline

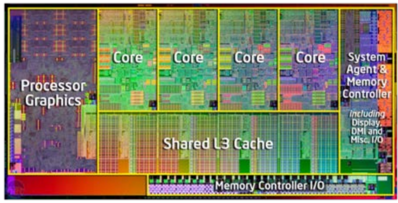
- Introduction & Motivation
- Design time analysis
- Run-time control
- Proofs
- Evaluation results
- Conclusions & Future directions

Target domain

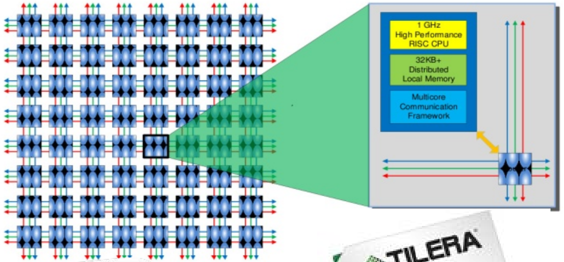
- **Software:** Mixed Critical applications
  - Hard & soft deadlines
  - Different consequences in failure
  - Criticality Levels: DAL, ASIL, …
  - Domains: Aeronautics, Automotive, …

Target domain

- **Software:** Mixed Critical applications
  - Hard & soft deadlines
  - Different consequences in failure
  - Criticality Levels: DAL, ASIL, …
  - Domains: Aeronautics, Automotive, …

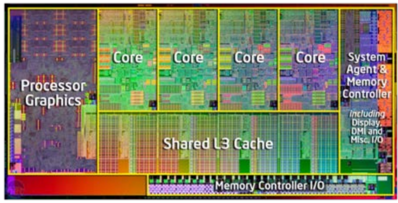


- **Hardware:** Multi/many-cores
  - Cores with components with dynamic behavior
  - Shared resources
  - COTS: Tilera, Kalray, Texas TMS, ...

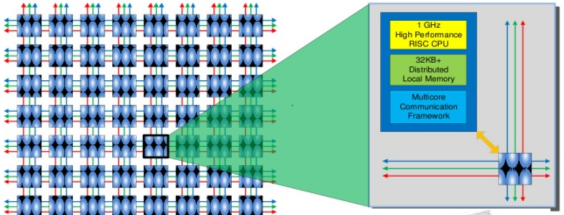


Target domain

- **Software:** Mixed Critical applications
  - Hard & soft deadlines
  - Different consequences in failure
  - Criticality Levels: DAL, ASIL, …
  - Domains: Aeronautics, Automotive, …



- **Hardware:** Multi/many-cores
  - Cores with components with dynamic behavior
  - Shared resources
  - COTS: Tilera, Kalray, Texas TMS, ...

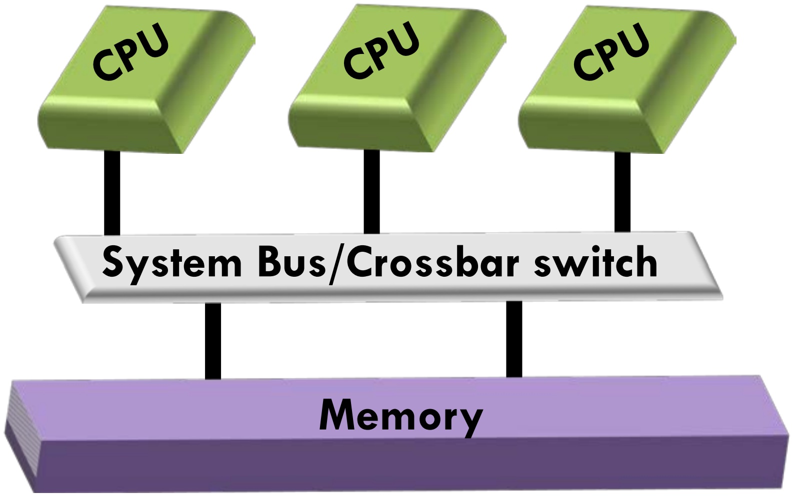


Challenge: Guarantee hard real-time response **&** improve cores utilization

Motivation

- Safe WCET static analysis for critical task

Multi/Many-cores

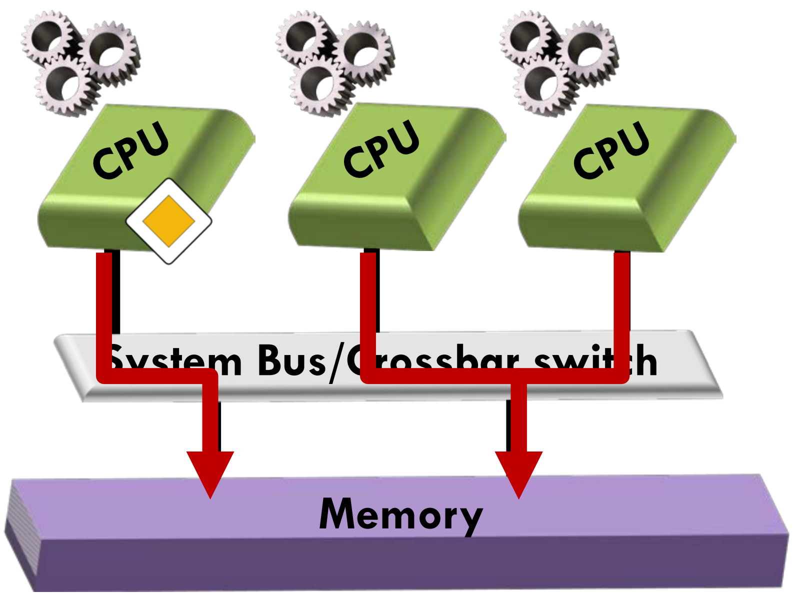


Motivation

- Safe WCET static analysis for critical task

Multi/Many-cores

- Maximum load

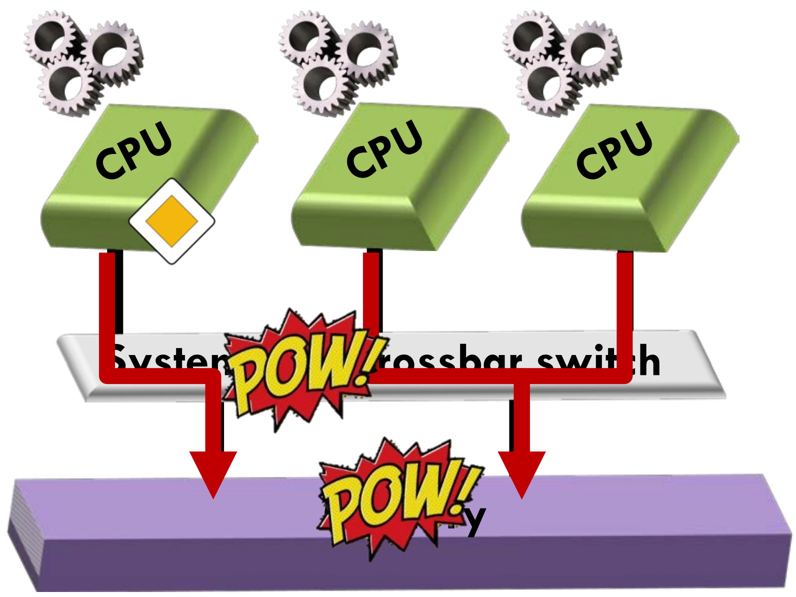


Motivation

- Safe WCET static analysis for critical task

Multi/Many-cores

- Maximum load
- Full congestion

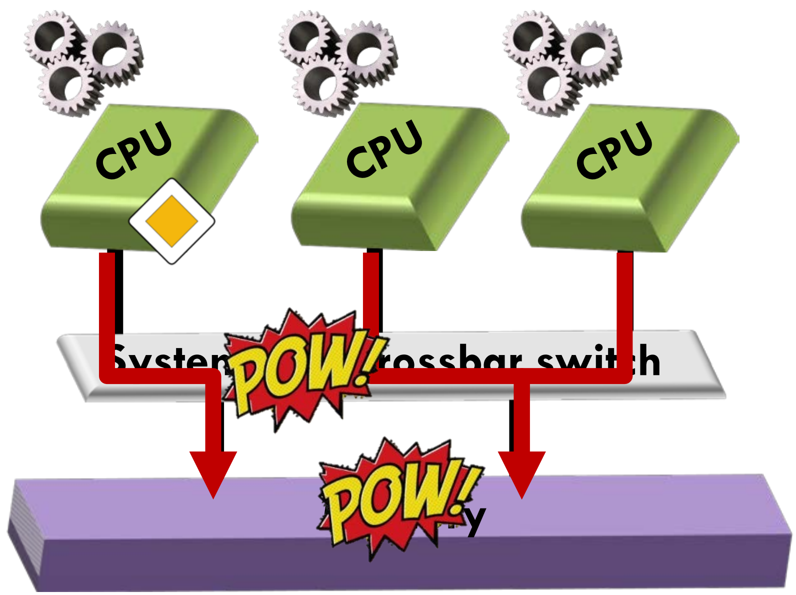


Motivation

- Safe WCET static analysis for critical task

Multi/Many-cores

- Maximum load
- Full congestion



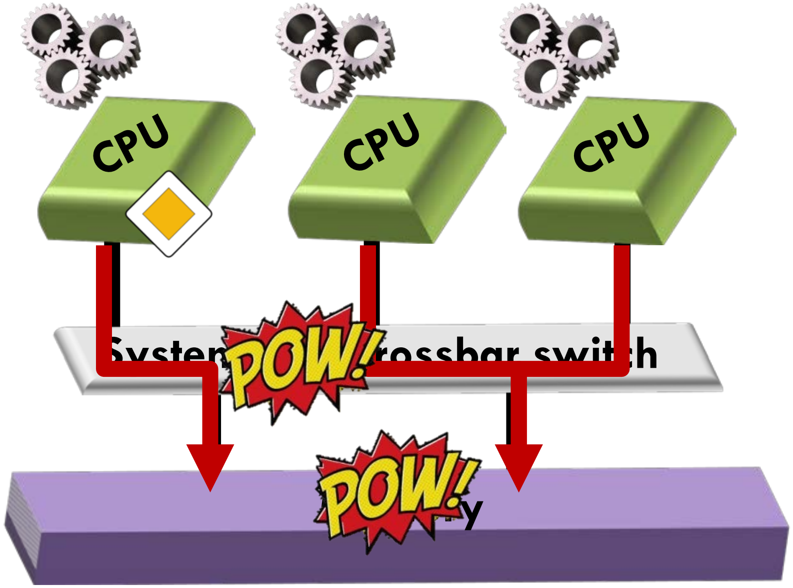
WCET > D_C

Motivation

- Safe WCET static analysis for critical task

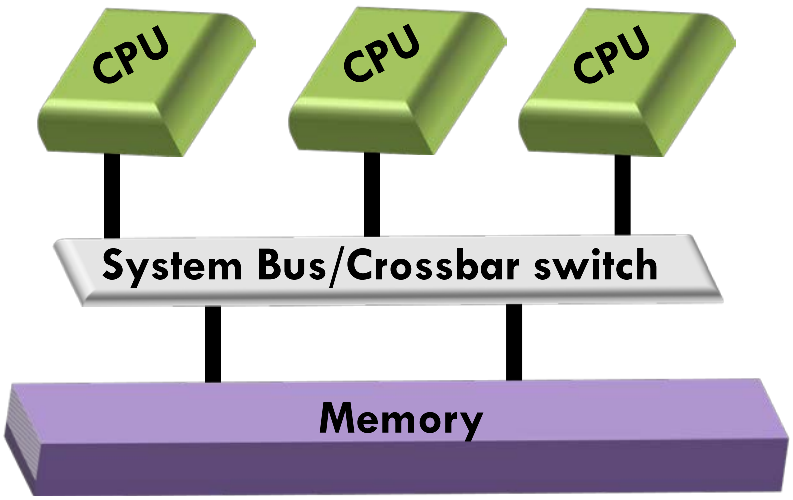
Multi/Many-cores

- Maximum load
- Full congestion



WCET > D_C

Single core/Single application

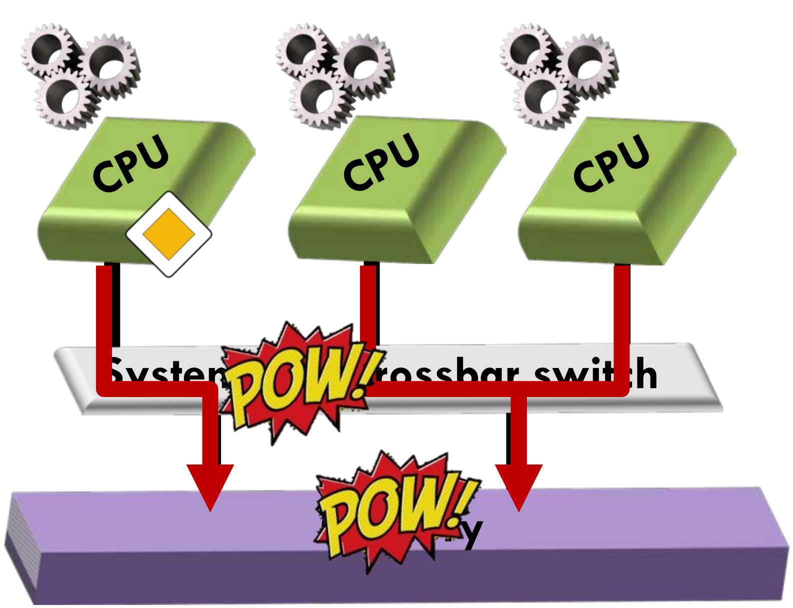


Motivation

- Safe WCET static analysis for critical task

Multi/Many-cores

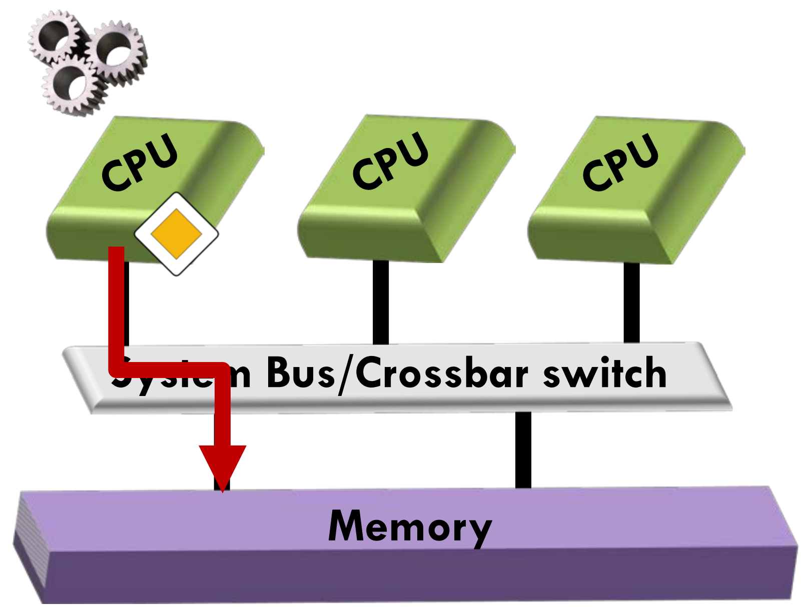
- Maximum load
- Full congestion



WCET > D_C

Single core/Single application

- Isolation
- No congestion



Motivation

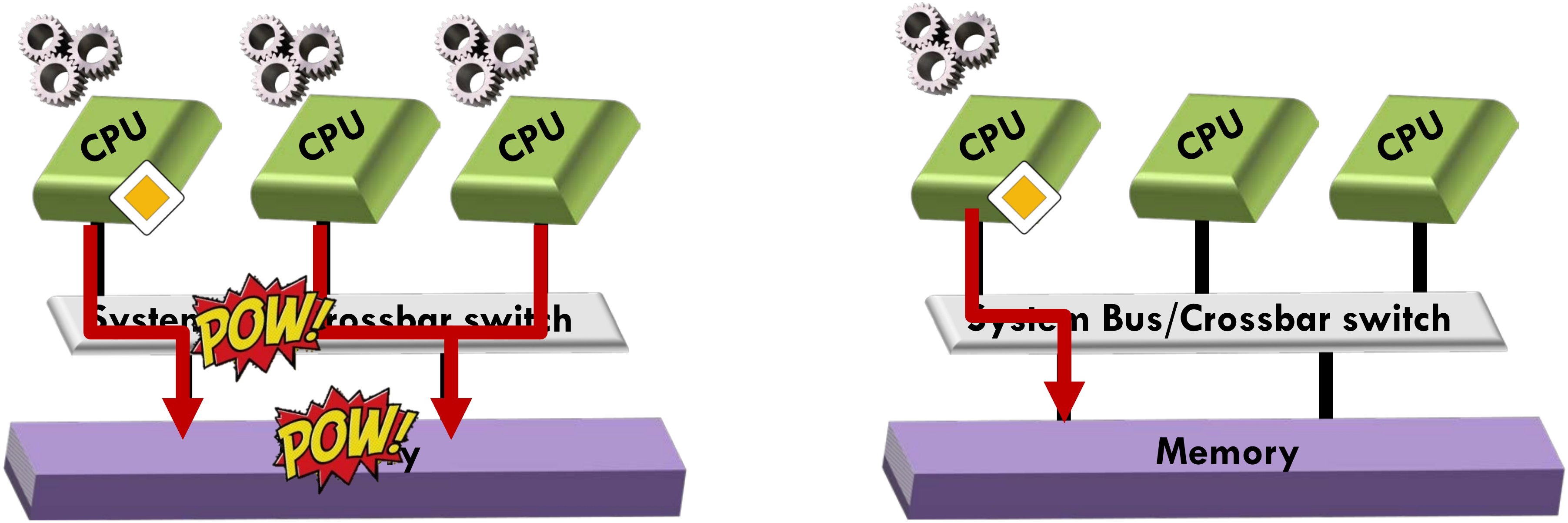
- Safe WCET static analysis for critical task

Multi/Many-cores

- Maximum load
- Full congestion

Single core/Single application

- Isolation
- No congestion



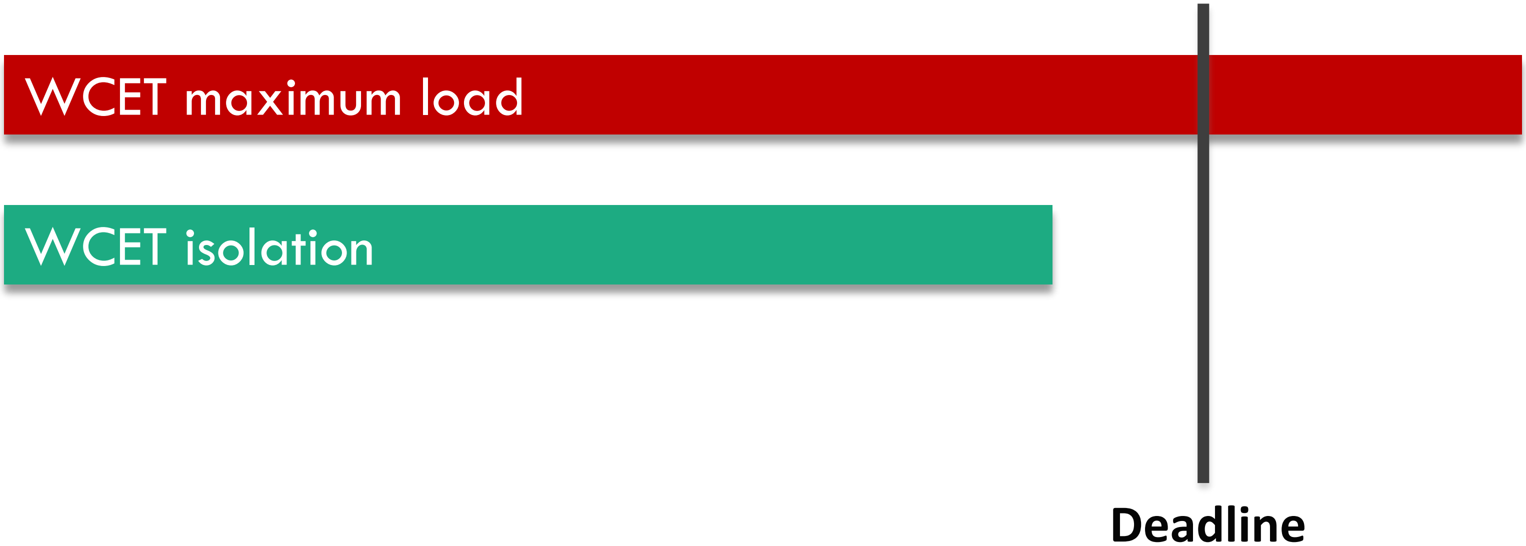
WCET > D_C

WCET ≤ D_C

Contribution

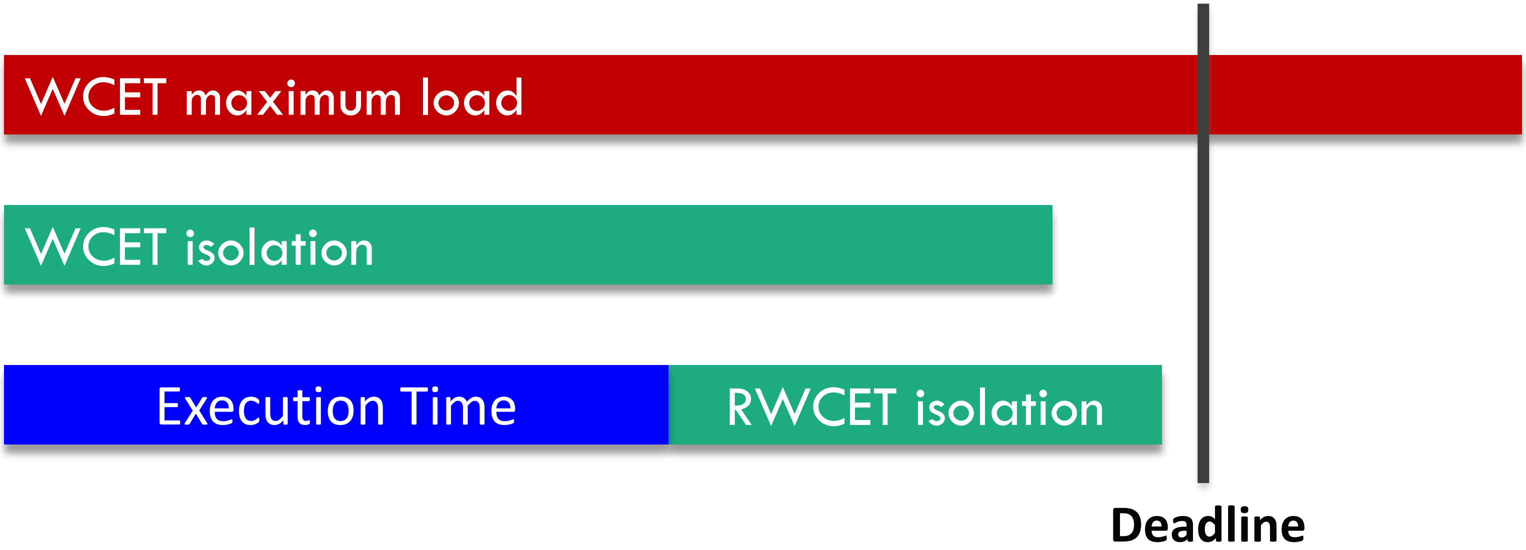
- Existing approaches [Anderson ‘10, Mollison’10]: Not directly applicable

Contribution



- ✗ Existing approaches [Anderson '10, Mollison'10]: Not directly applicable
- ✓ Safe: Only the most critical tasks on 1 core
  - ✗ Under-utilization of the system resources

Contribution



- ✘ Existing approaches [Anderson '10, Mollison'10]: Not directly applicable
- ✔ Safe: Only the most critical tasks on 1 core
  - ✘ Under-utilization of the system resources
- ✔ Proposed: All tasks as long as it is safe OR switch to isolated execution
  - ✔ Improved system resources utilization

Proposed methodology: RWCET computation

- Design-time:
  - Values for all points/iterations
    - Full unrolling of application
    - Prohibitive space overhead
  - Approximation
    - Reduced gain

Proposed methodology: RWCET computation

- Design-time:
  - Values for all points/iterations
    - Full unrolling of application
    - Prohibitive space overhead
  - Approximation
    - Reduced gain
- Run-time:
  - Significant number of paths
  - Prohibitive time overhead

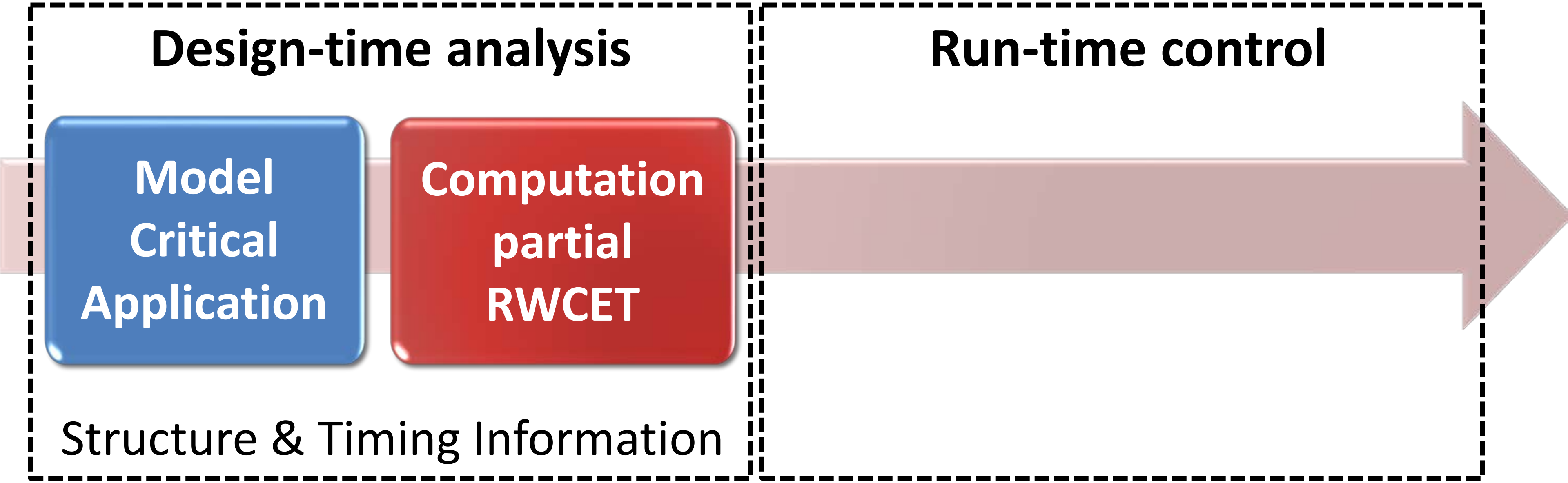
Proposed methodology: RWCET computation

- Design-time:
  - Values for all points/iterations
    - Full unrolling of application
    - Prohibitive space overhead
  - Approximation
    - Reduced gain
- Run-time:
  - Significant number of paths
  - Prohibitive time overhead

Design-time analysis	**Run-time control**

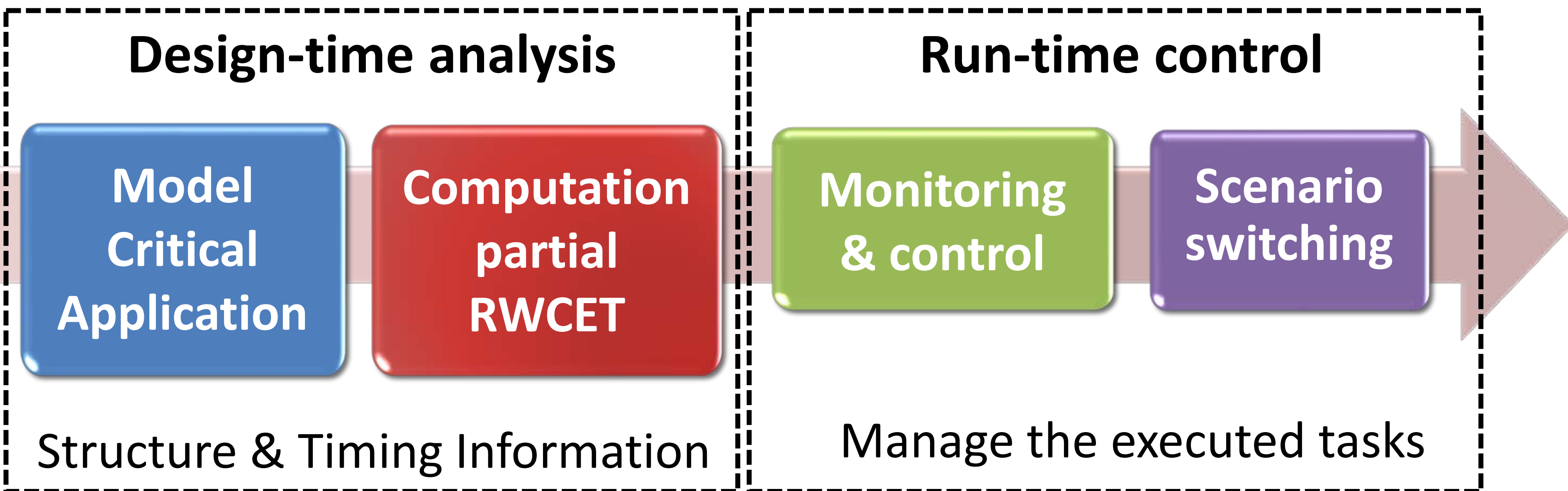
Proposed methodology: RWCET computation

- Design-time:
  - Values for all points/iterations
    - Full unrolling of application
    - Prohibitive space overhead
  - Approximation
    - Reduced gain
- Run-time:
  - Significant number of paths
  - Prohibitive time overhead



Proposed methodology: RWCET computation

- Design-time:
  - Values for all points/iterations
    - Full unrolling of application
    - Prohibitive space overhead
  - Approximation
    - Reduced gain
- Run-time:
  - Significant number of paths
  - Prohibitive time overhead



Design-time analysis

Model of critical application

- Set of functions: $S = \{F_0, \dots, F_n\}$
- Each function
  - Extended Control Flow Graph (ECFG)
    - Nodes: Binary instructions with 1 observation point
    - Edges: Ordering of Instruction execution

Model of critical application

- Set of functions: $S = \{F_0, \dots, F_n\}$
- Each function
  - Extended Control Flow Graph (ECFG)
    - Nodes: Binary instructions with 1 observation point
    - Edges: Ordering of Instruction execution

```
int main(void){
  int i;
  int A[10];
  for (i=0; i<10; i++){
    A[i]=i;
  }
  return EXIT_SUCCESS;
}
```

```
pushq       %rbp
.cfi_def_cfa_offset 16
.cfi_offset 6, -16
movq        %rsp, %rbp
.cfi_def_cfa_register 6
movl         $0, -52(%rbp)
jmp         .L2 .
        .L3:
movl        -52(%rbp), %eax
cltq
movl        -52(%rbp), %edx
movl        %edx, -48(%rbp,%rax,4)
addl        $1, -52(%rbp)
        .L2:
cmpl        $9, -52(%rbp)
jle         .L3
movl        $0, %eax
popq        %rbp
.cfi_def_cfa 7, 8
ret
```

Model of critical application

- Set of functions: $S = \{F_0, \dots, F_n\}$
- Each function
  - Extended Control Flow Graph (ECFG)
    - Nodes: Binary instructions with 1 observation point
    - Edges: Ordering of Instruction execution

```
int main(void){
  int i;
  int A[10];
  for (i=0; i<10; i++){
    A[i]=i;
  }
  return EXIT_SUCCESS;
}
```

```
pushq       %rbp
.cfi_def_cfa_offset 16
.cfi_offset 6, -16
movq        %rsp, %rbp
.cfi_def_cfa_register 6

movl        $0, -52(%rbp)
jmp         .L2 .
      .L3:
movl        -52(%rbp), %eax
cltq
movl        -52(%rbp), %edx
movl        %edx, -48(%rbp,%rax,4)
addl        $1, -52(%rbp)
      .L2:
cmpl        $9, -52(%rbp)
jle         .L3
movl        $0, %eax
popq        %rbp
.cfi_def_cfa 7, 8
ret
```

Model of critical application

- Set of functions: $S = \{F_0, \dots, F_n\}$
- Each function
  - Extended Control Flow Graph (ECFG)
    - Nodes: Binary instructions with 1 observation point
    - Edges: Ordering of Instruction execution

```
int main(void){
  int i;
  int A[10];
  for (i=0; i<10; i++){
    A[i]=i;
  }
  return EXIT_SUCCESS;
}
```

```
pushq           %rbp
.cfi_def_cfa_offset 16
.cfi_offset 6, -16
movq            %rsp, %rbp
.cfi_def_cfa_register 6
movl             $0, -52(%rbp)
jmp             .L2 .
        .L3:
movl            -52(%rbp), %eax
cltq
movl            -52(%rbp), %edx
movl            %edx, -48(%rbp,%rax,4)
addl            $1, -52(%rbp)
        .L2:
cmpl            $9, -52(%rbp)
jle             .L3
movl            $0, %eax
popq            %rbp
.cfi_def_cfa 7, 8
ret
```

Model of critical application

- Set of functions: $S = \{F_0, \ldots, F_n\}$
- Each function
  - Extended Control Flow Graph (ECFG)
    - Nodes: Binary instructions with 1 observation point
    - Edges: Ordering of Instruction execution

```
int main(void){
  int i;
  int A[10];
  for (i=0; i<10; i++){
    A[i]=i;
  }
  return EXIT_SUCCESS;
}
```

```
pushq                %rbp
.cfi_def_cfa_offset 16
.cfi_offset 6, -16
movq                 %rsp, %rbp
.cfi_def_cfa_register 6
movl                 $0, -52(%rbp)
jmp                  .L2 .
        .L3:
movl                 -52(%rbp), %eax
cltq
movl                 -52(%rbp), %edx
movl                 %edx, -48(%rbp,%rax,4)
addl                 $1, -52(%rbp)
        .L2:
cmpl                 $9, -52(%rbp)
jle                  .L3
movl                 $0, %eax
popq                 %rbp
.cfi_def_cfa 7, 8
ret
```

Model of critical application

- Set of functions: $S = \{F_0, \dots, F_n\}$
- Each function
  - Extended Control Flow Graph (ECFG)
    - Nodes: Binary instructions with 1 observation point
    - Edges: Ordering of Instruction execution

```
int main(void){
  int i;
  int A[10];
  for (i=0; i<10; i++){
    A[i]=i;
  }
  return EXIT_SUCCESS;
}
```

```
pushq          %rbp
.cfi_def_cfa_offset 16
.cfi_offset 6, -16
movq           %rsp, %rbp
.cfi_def_cfa_register 6
movl            $0, -52(%rbp)
jmp            .L2 .
        .L3:
movl           -52(%rbp), %eax
cltq
movl           -52(%rbp), %edx
movl           %edx, -48(%rbp,%rax,4)
addl           $1, -52(%rbp)
        .L2:
cmpl           $9, -52(%rbp)
jle            .L3
movl           $0, %eax
popq           %rbp
.cfi_def_cfa 7, 8
ret
```

Model of critical application

- Set of functions: $S = \{F_0, \dots, F_n\}$
- Each function
  - Extended Control Flow Graph (ECFG)
    - Nodes: Binary instructions with 1 observation point
    - Edges: Ordering of Instruction execution

```
int main(void){
   int i;
   int A[10];
   for (i=0; i<10; i++){
     A[i]=i;
   }
   return EXIT_SUCCESS;
}
```

```
pushq           %rbp
.cfi_def_cfa_offset 16
.cfi_offset 6, -16
movq            %rsp, %rbp
.cfi_def_cfa_register 6

movl             $0, -52(%rbp)
jmp             .L2 .
        .L3:
movl            -52(%rbp), %eax
cltq
movl            -52(%rbp), %edx
movl            %edx, -48(%rbp,%rax,4)
addl            $1, -52(%rbp)
        .L2:
cmpl            $9, -52(%rbp)
jle             .L3
movl            $0, %eax
popq            %rbp
.cfi_def_cfa 7, 8
ret
```

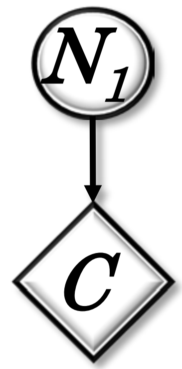
N_1

Model of critical application

- Set of functions: $S = \{F_0, \ldots, F_n\}$
- Each function
  - Extended Control Flow Graph (ECFG)
    - Nodes: Binary instructions with 1 observation point
    - Edges: Ordering of Instruction execution

```
int main(void){
   int i;
   int A[10];
   for (i=0; i<10; i++){
     A[i]=i;
   }
   return EXIT_SUCCESS;
}
```

```
pushq           %rbp
.cfi_def_cfa_offset 16
.cfi_offset 6, -16
movq            %rsp, %rbp
.cfi_def_cfa_register 6
movl             $0, -52(%rbp)
jmp             .L2 .
        .L3:
movl            -52(%rbp), %eax
cltq
movl            -52(%rbp), %edx
movl            %edx, -48(%rbp,%rax,4)
addl            $1, -52(%rbp)
        .L2:
cmpl            $9, -52(%rbp)
jle             .L3
movl            $0, %eax
popq            %rbp
.cfi_def_cfa 7, 8
ret
```

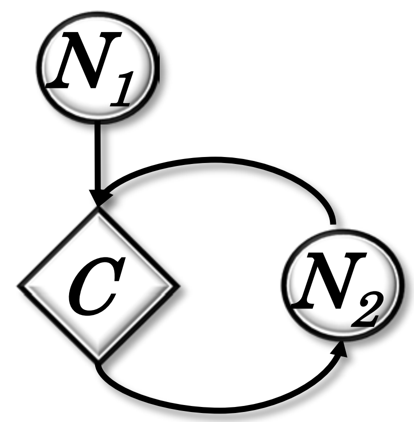


Model of critical application

- Set of functions: $S = \{F_0, \dots , F_n\}$
- Each function
  - Extended Control Flow Graph (ECFG)
    - Nodes: Binary instructions with 1 observation point
    - Edges: Ordering of Instruction execution

```
int main(void){
   int i;
   int A[10];
   for (i=0; i<10; i++){
     A[i]=i;
   }
   return EXIT_SUCCESS;
}
```

```
pushq       %rbp
.cfi_def_cfa_offset 16
.cfi_offset 6, -16
movq        %rsp, %rbp
.cfi_def_cfa_register 6
movl        $0, -52(%rbp)
jmp         .L2 .
      .L3:
movl        -52(%rbp), %eax
cltq
movl        -52(%rbp), %edx
movl        %edx, -48(%rbp,%rax,4)
addl        $1, -52(%rbp)
      .L2:
cmpl        $9, -52(%rbp)
jle         .L3
movl        $0, %eax
popq        %rbp
.cfi_def_cfa 7, 8
ret
```

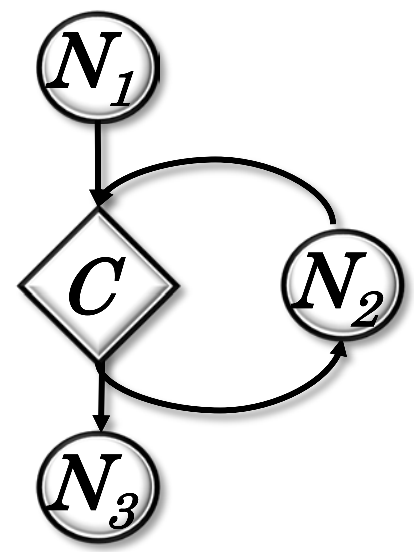


Model of critical application

- Set of functions: $S = \{F_0, \dots, F_n\}$
- Each function
  - Extended Control Flow Graph (ECFG)
    - Nodes: Binary instructions with 1 observation point
    - Edges: Ordering of Instruction execution

```
int main(void){
   int i;
   int A[10];
   for (i=0; i<10; i++){
     A[i]=i;
   }
   return EXIT_SUCCESS;
}
```

```
pushq        %rbp
.cfi_def_cfa_offset 16
.cfi_offset 6, -16
movq         %rsp, %rbp
.cfi_def_cfa_register 6
movl         $0, -52(%rbp)
jmp          .L2 .
        .L3:
movl         -52(%rbp), %eax
cltq
movl         -52(%rbp), %edx
movl         %edx, -48(%rbp,%rax,4)
addl         $1, -52(%rbp)
        .L2:
cmpl         $9, -52(%rbp)
jle          .L3
movl         $0, %eax
popq         %rbp
.cfi_def_cfa 7, 8
ret
```



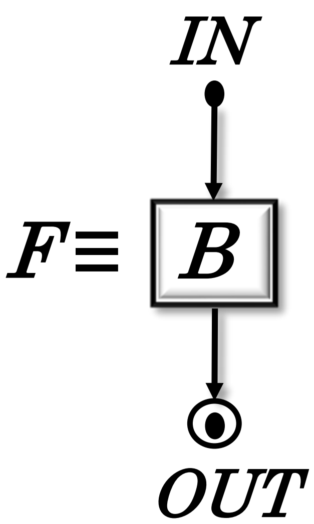
Graph grammar

- Terminal nodes

Graph grammar

- Terminal nodes

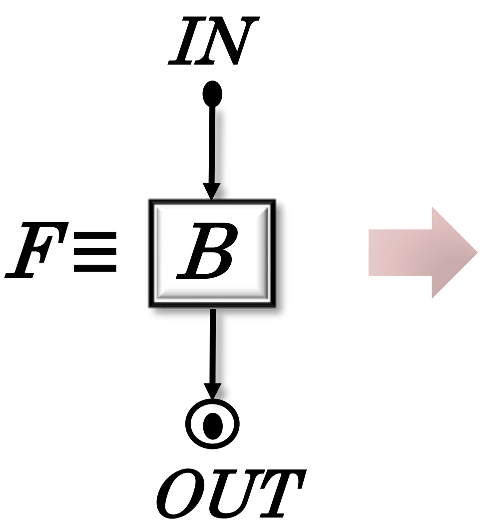
- Function



Graph grammar

- Terminal nodes

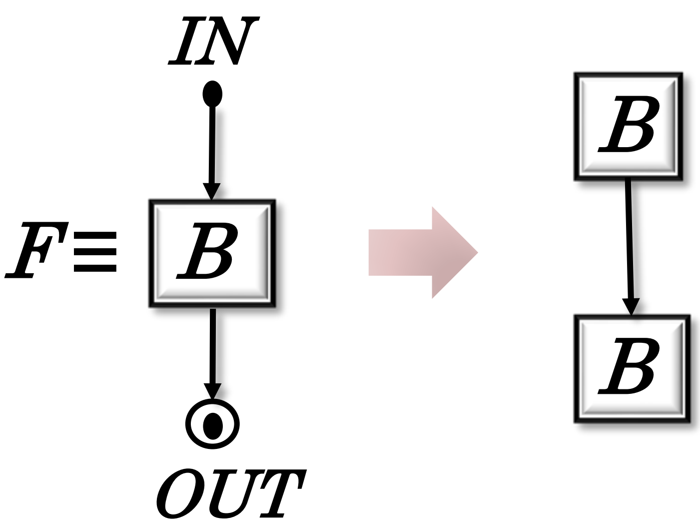
- Function
- Rewriting rules



Graph grammar

- Terminal nodes

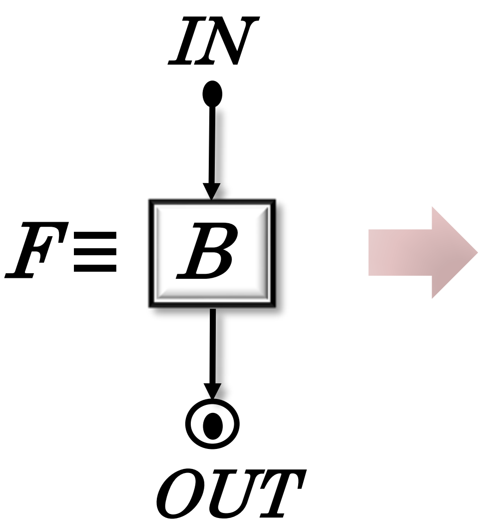
- Function
- Rewriting rules

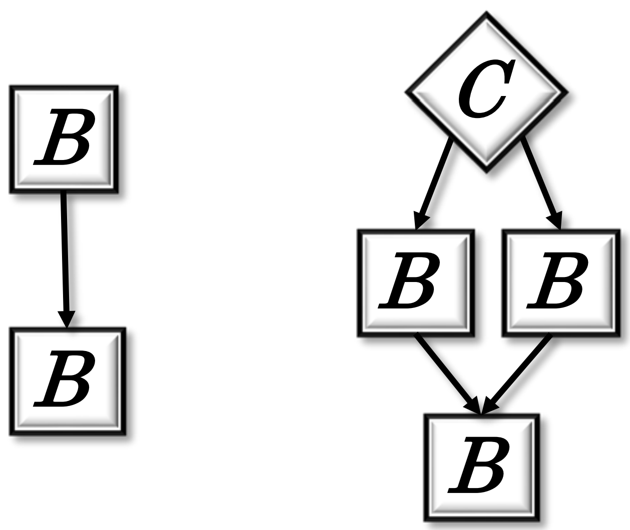


Graph grammar

- Terminal nodes

- Function
- Rewriting rules

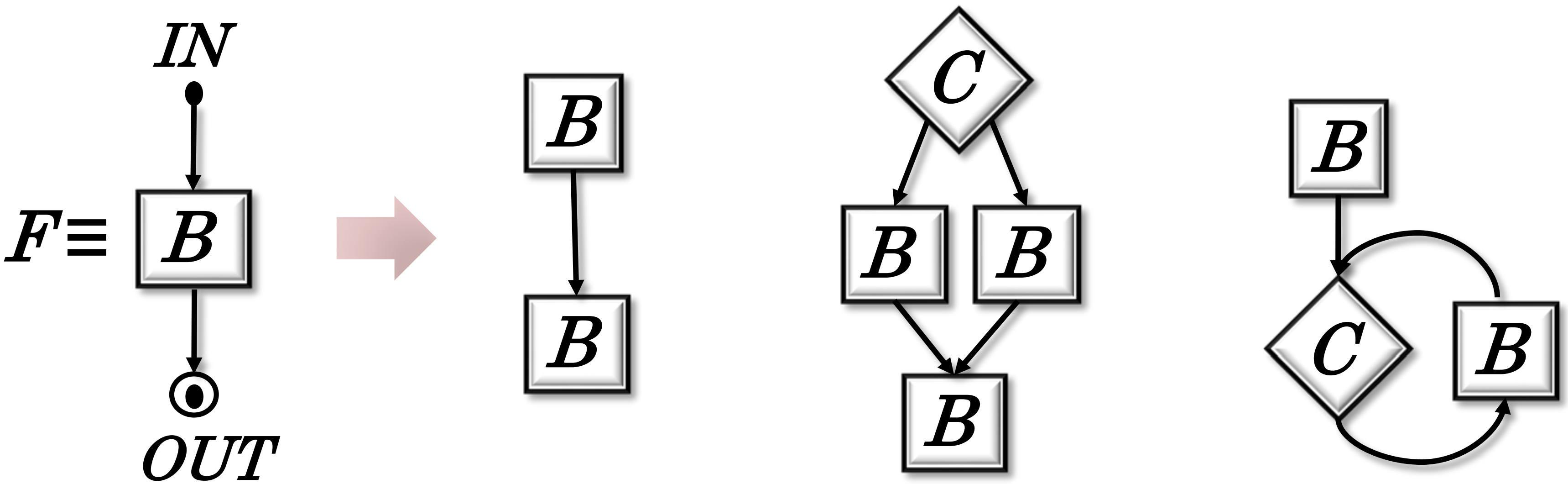




Graph grammar

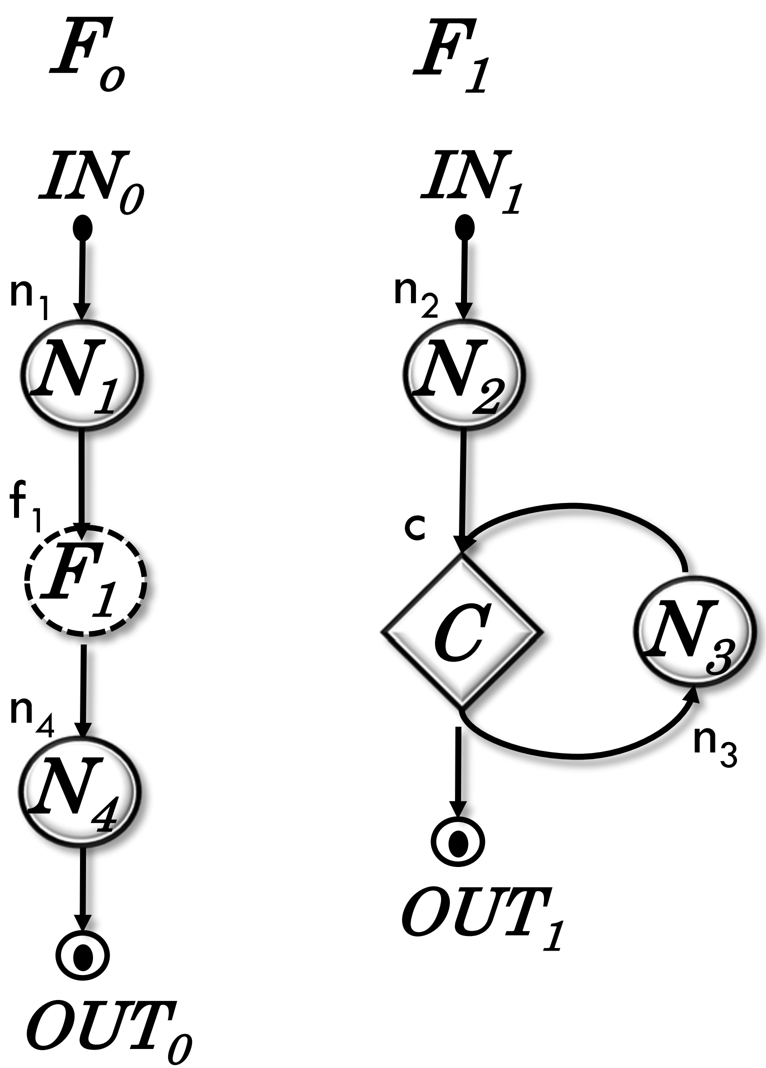
- Terminal nodes

- Function
- Rewriting rules



Structure information: Level, Type, Head

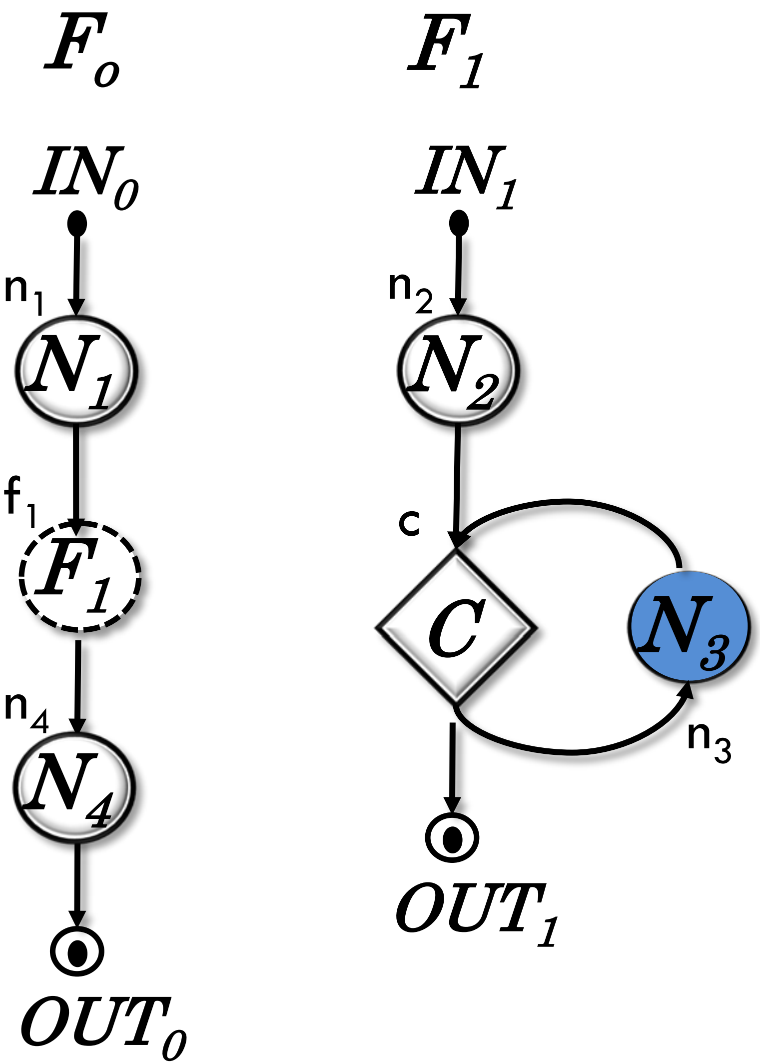
Initialization: S



Point	Level	Type	Head
S			
n_1			
f_1			
n_2			
c			
n_3			
n_4			

Structure information: Level, Type, Head

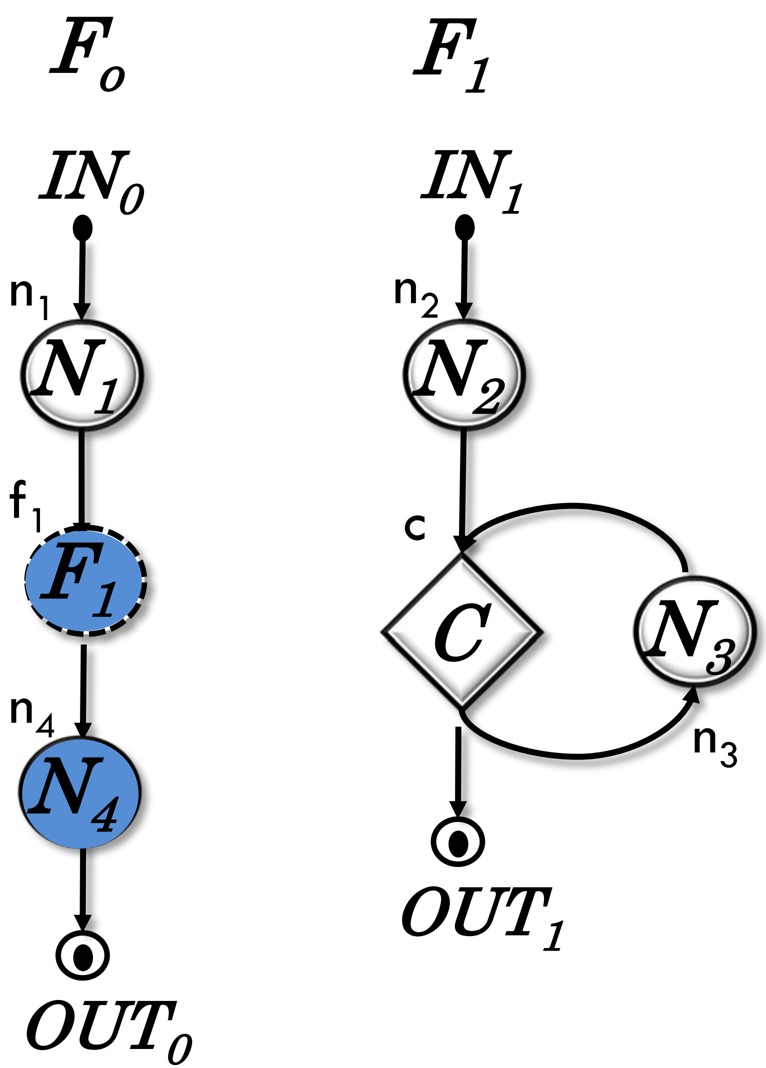
Initialization: S



Point	Level	Type	Head
S	0		
n_1	1		
f_1	1		
n_2	1		
c	1		
n_3	2		
n_4	1		

Structure information: Level, Type, Head

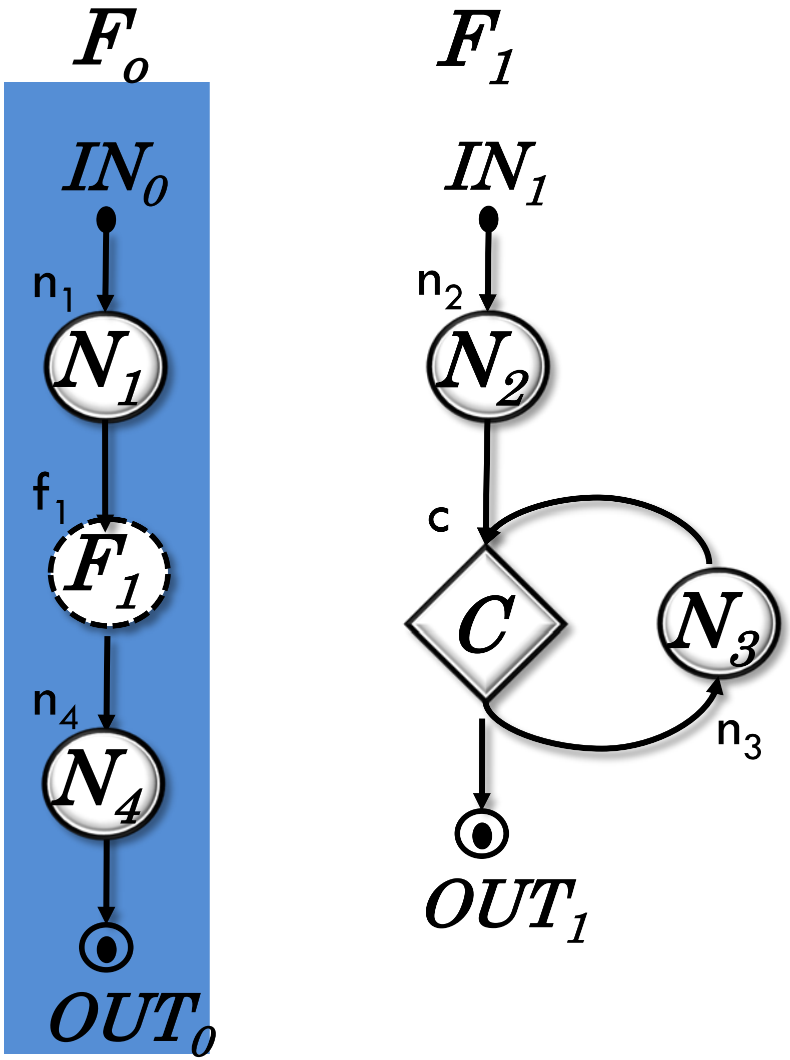
Initialization: S



Point	Level	Type	Head
S	0	-	
n_1	1	-	
f_1	1	F_ENTRY	
n_2	1	-	
c	1	-	
n_3	2	-	
n_4	1	F_EXIT	

Structure information: Level, Type, Head

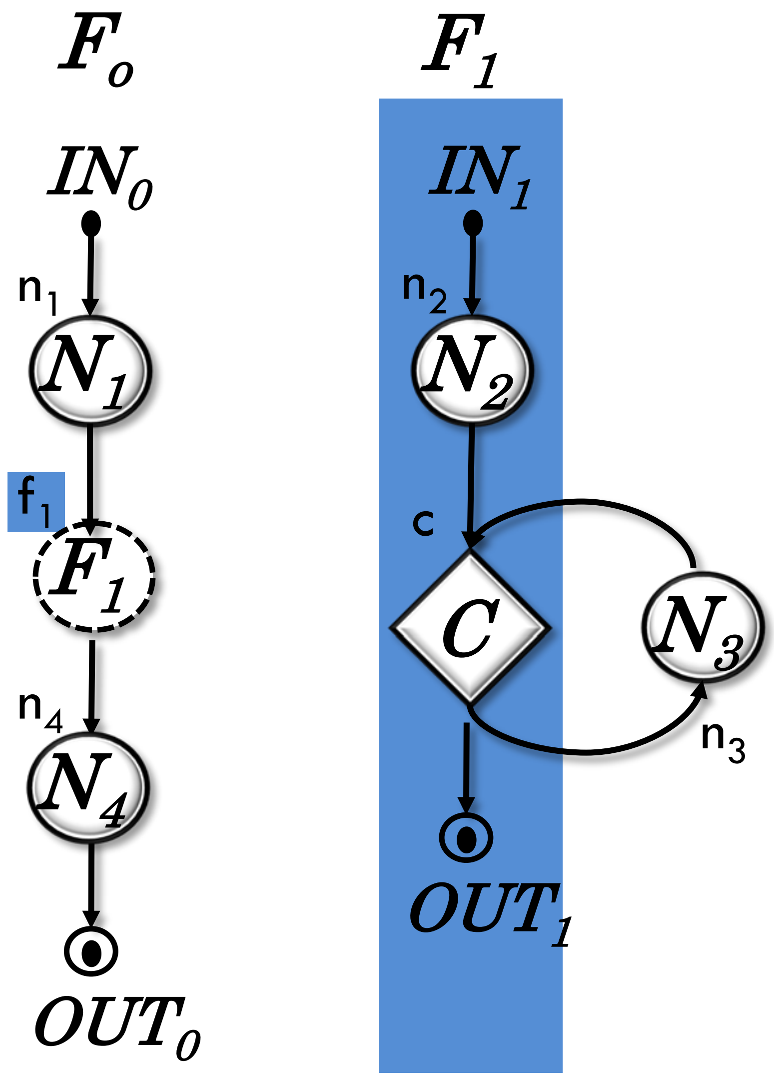
Initialization: S



Point	Level	Type	Head
S	0	-	-
n_1	1	-	S
f_1	1	F_ENTRY	S
n_2	1	-	f_1
c	1	-	f_1
n_3	2	-	c
n_4	1	F_EXIT	S

Structure information: Level, Type, Head

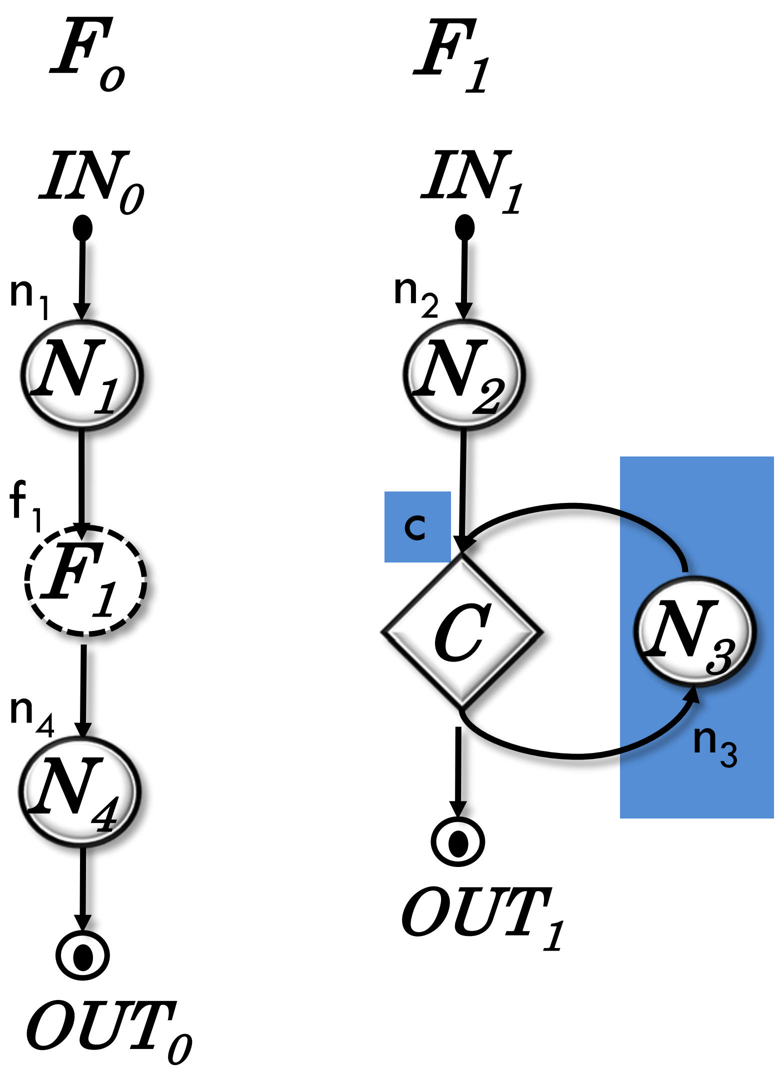
Initialization: S



Point	Level	Type	Head
S	0	-	-
n_1	1	-	S
f_1	1	F_ENTRY	S
n_2	1	-	f_1
c	1	-	f_1
n_3	2	-	c
n_4	1	F_EXIT	S

Structure information: Level, Type, Head

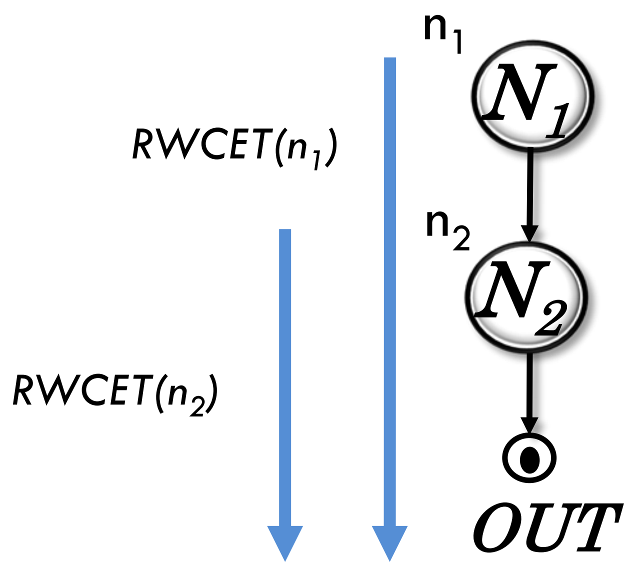
Initialization: S



Point	Level	Type	Head
S	0	-	-
n_1	1	-	S
f_1	1	F_ENTRY	S
n_2	1	-	f_1
c	1	-	f_1
n_3	2	-	c
n_4	1	F_EXIT	S

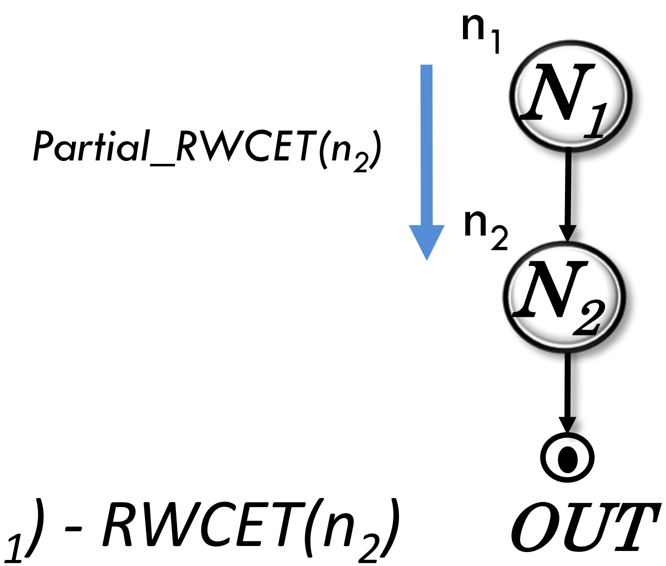
Timing information

- Remaining WCET, *RWCET(x)*
  - *ILP formulation*



Timing information

- Remaining WCET, *RWCET(x)*
  - *ILP formulation*

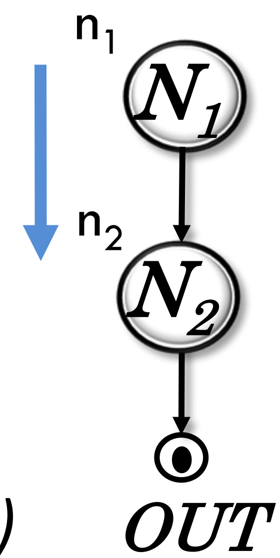


- *Partial_RWCET(n_2) = RWCET(n_1) - RWCET(n_2)*

Timing information

- Remaining WCET, *RWCET(x)*
  - *ILP formulation*

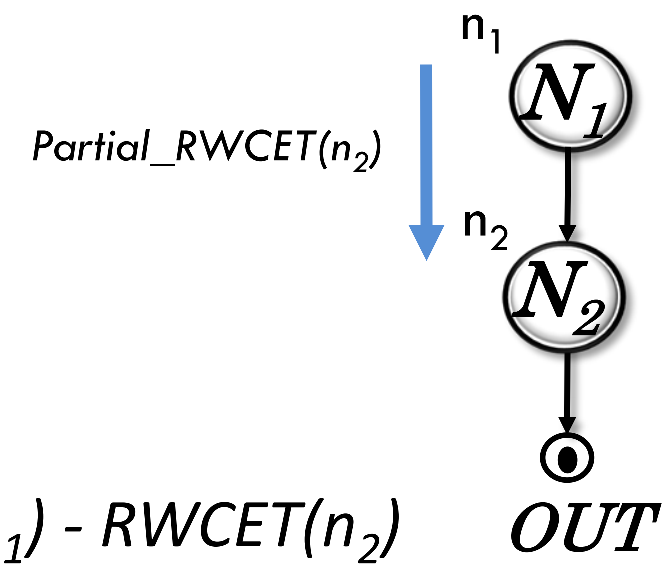
Partial_RWCET(n_2)



- *Partial_RWCET(n_2) = RWCET(n_1) - RWCET(n_2)*
- Isolation scenario: head(x) → x
  - $d_{c\text{-}x} = RWCET(c, j) - RWCET(x, j),$ *for all* $j \leq iter$
  - $w_c = RWCET(c, j) - RWCET(c, j + 1),$ *for all* $j \leq iter$

Timing information

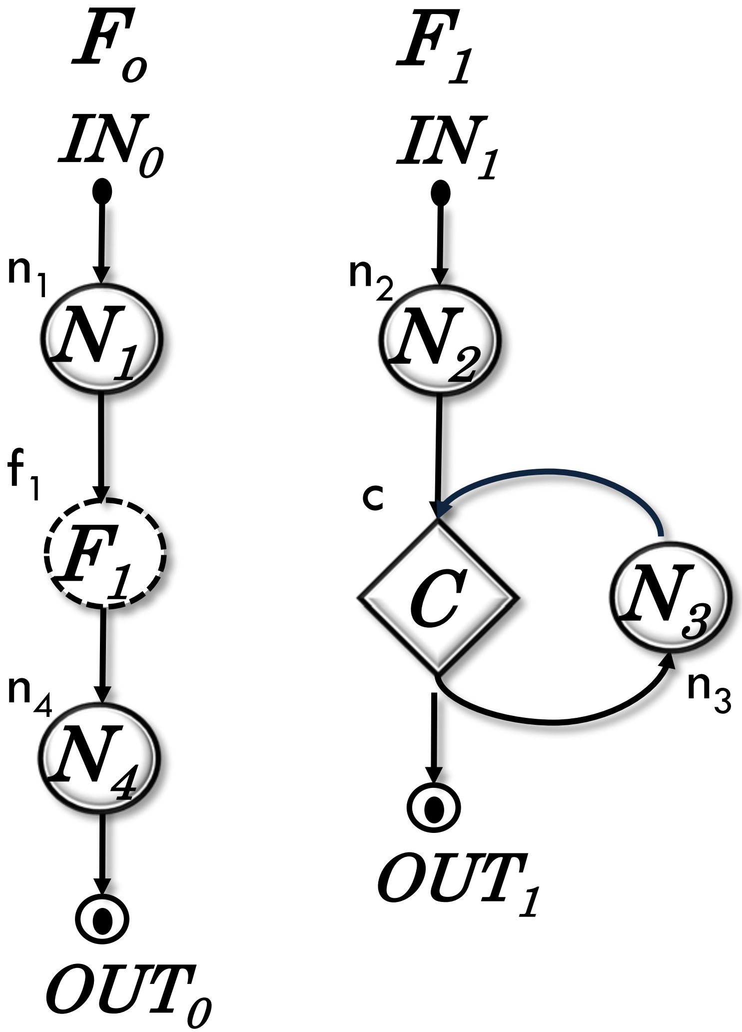
- Remaining WCET, *RWCET(x)*
  - *ILP formulation*



- $Partial\_RWCET(n_2) = RWCET(n_1) - RWCET(n_2)$
- Isolation scenario: head(x) → x
  - $d_{c\text{-}x} = RWCET(c, j) - RWCET(x, j)$, *for all* $j \leq iter$
  - $w_c = RWCET(c, j) - RWCET(c, j + 1)$, *for all* $j \leq iter$
- Maximum load scenario: maximum
  - $W = max(RWCET(x) - RWCET(x+1))$, *for all x*

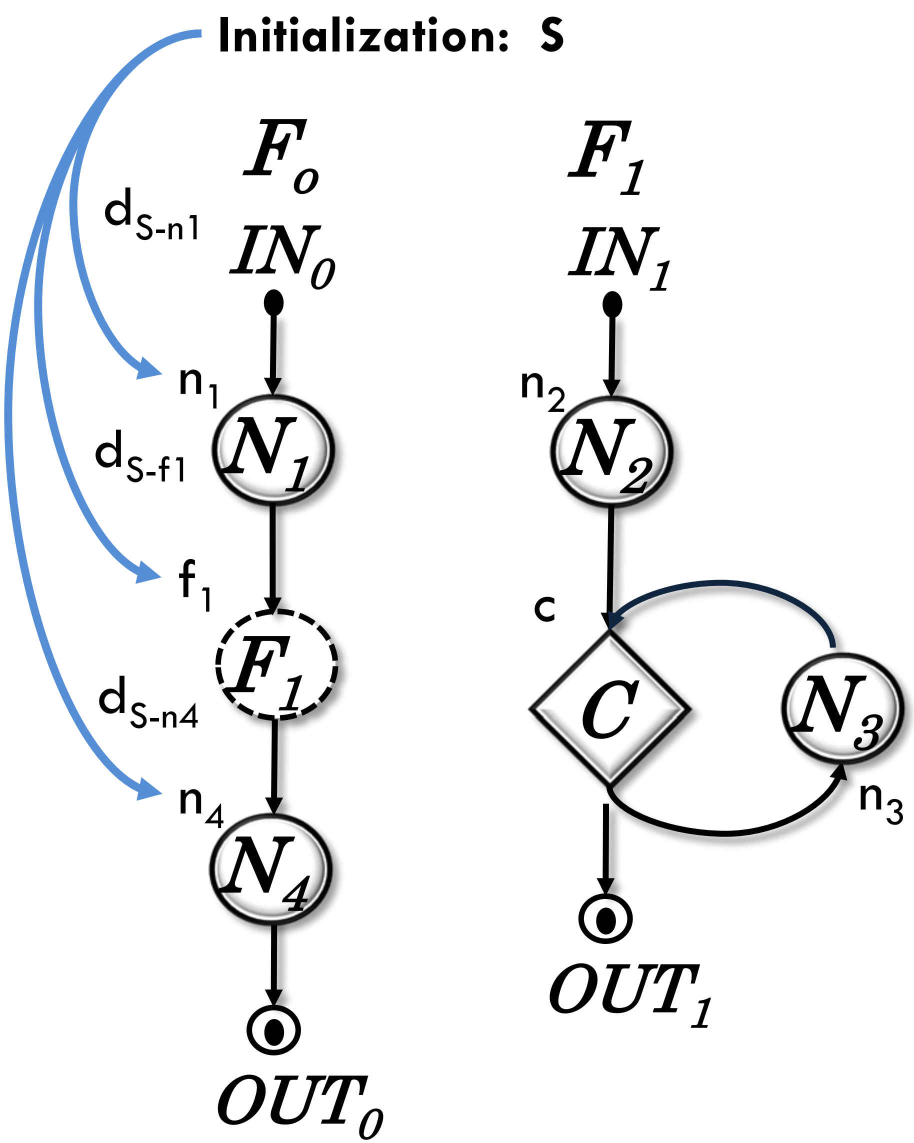
Timing information: Partial RWCETs

Initialization: S



Point	d_{h-x}	w_x
S		
n_1		
f_1		
n_2		
c		
n_3		
n_4		

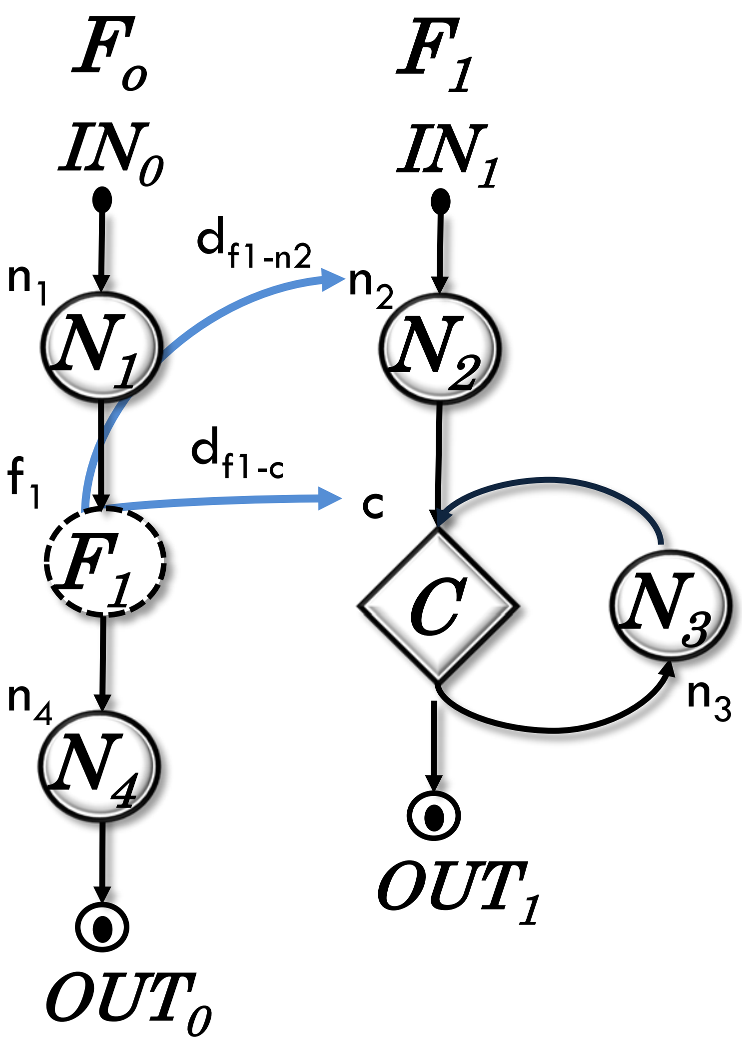
Timing information: Partial RWCETs



Point	d_{h-x}	w_x
S	0	
n_1	d_{S-n1}	
f_1	d_{S-f1}	
n_2	d_{f1-n2}	
c	d_{f1-c}	
n_3	d_{c-n3}	
n_4	d_{S-n4}	

Timing information: Partial RWCETs

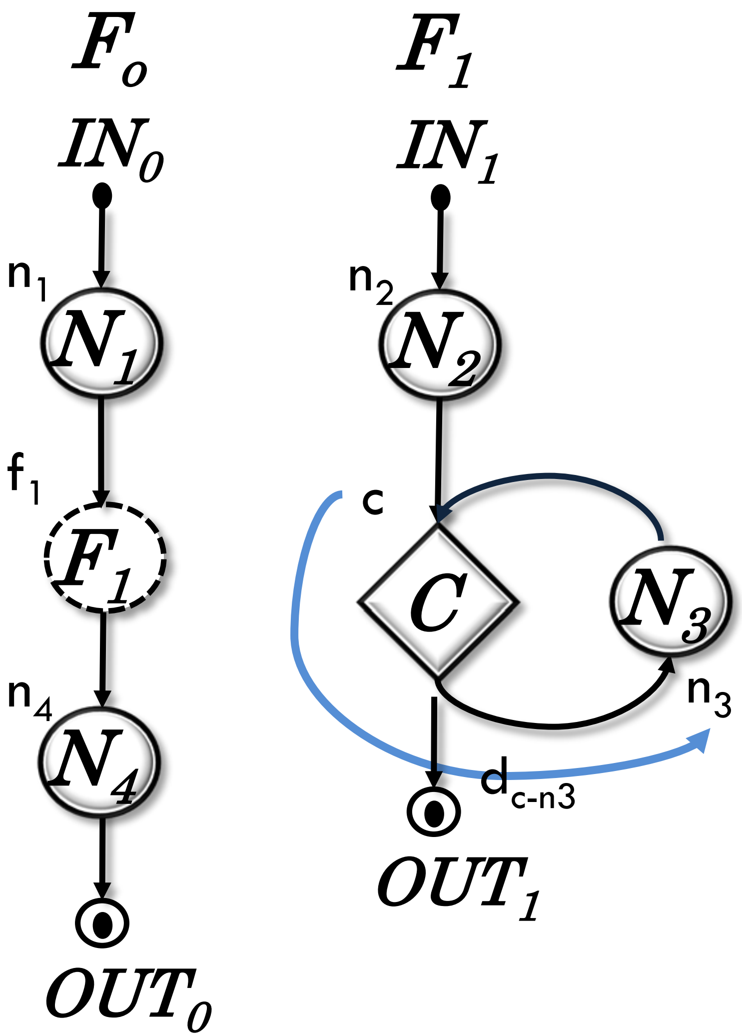
Initialization: S



Point	$d_{h\text{-}x}$	w_x
S	0	
n_1	$d_{S\text{-}n1}$	
f_1	$d_{S\text{-}f1}$	
n_2	$d_{f1\text{-}n2}$	
c	$d_{f1\text{-}c}$	
n_3	$d_{c\text{-}n3}$	
n_4	$d_{S\text{-}n4}$	

Timing information: Partial RWCETs

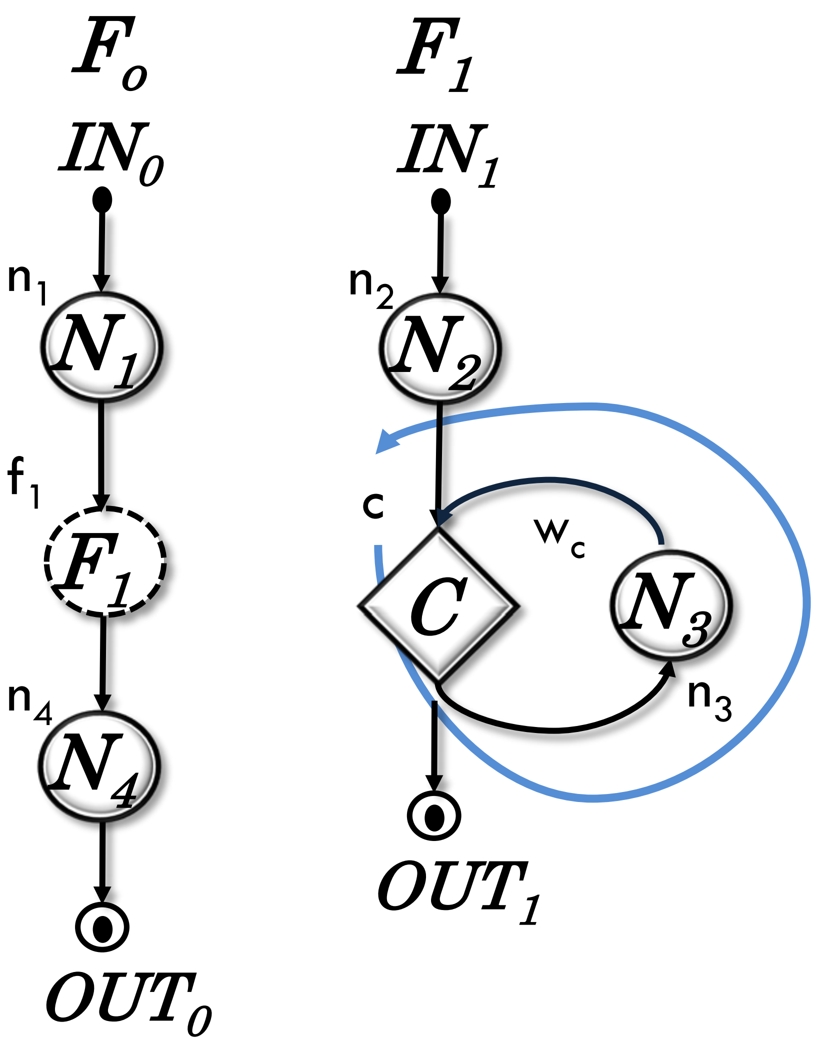
Initialization: S



Point	d_{h-x}	w_x
S	0	
n_1	d_{S-n1}	
f_1	d_{S-f1}	
n_2	d_{f1-n2}	
c	d_{f1-c}	
n_3	d_{c-n3}	
n_4	d_{S-n4}	

Timing information: Partial RWCETs

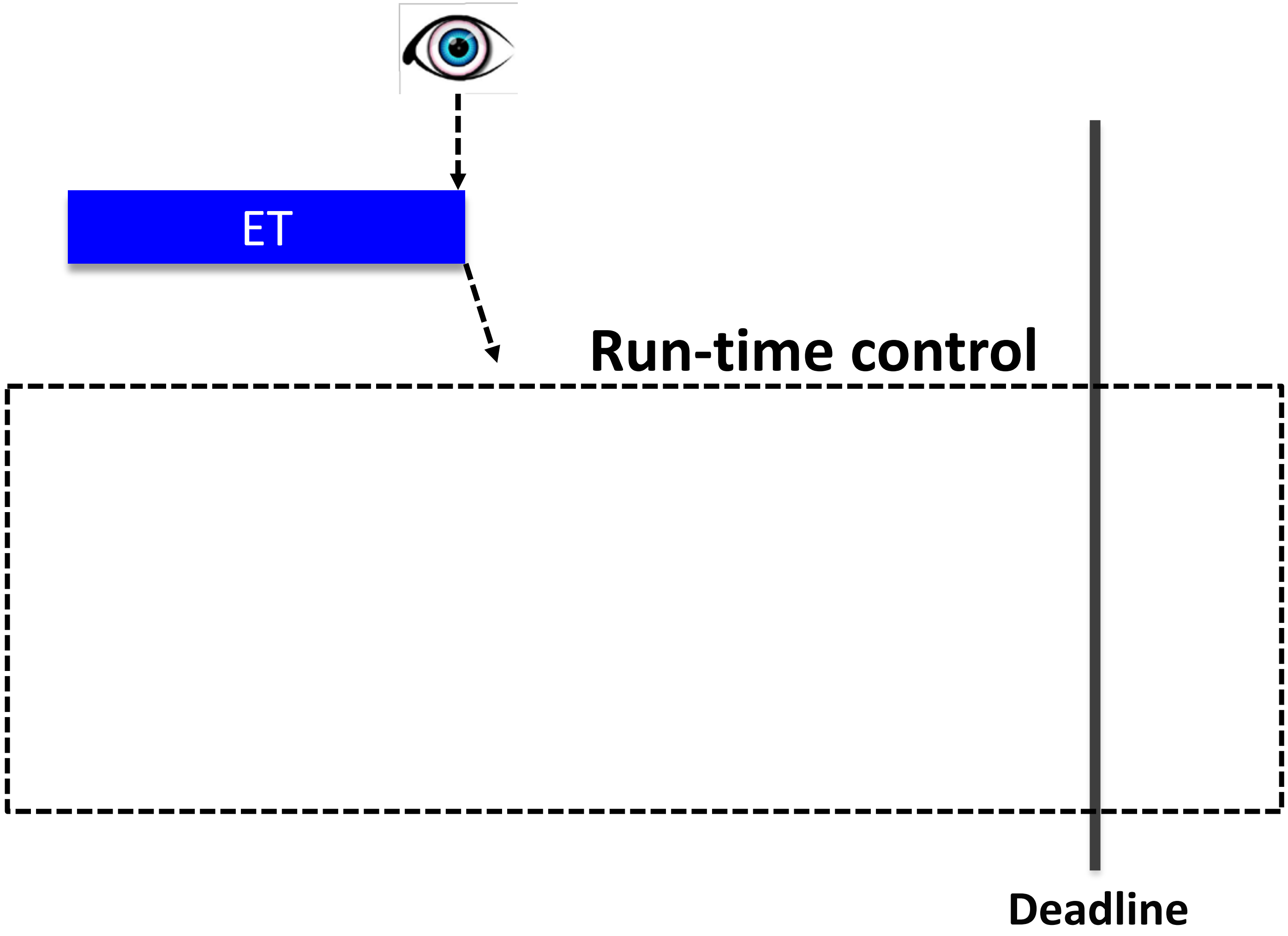
Initialization: S



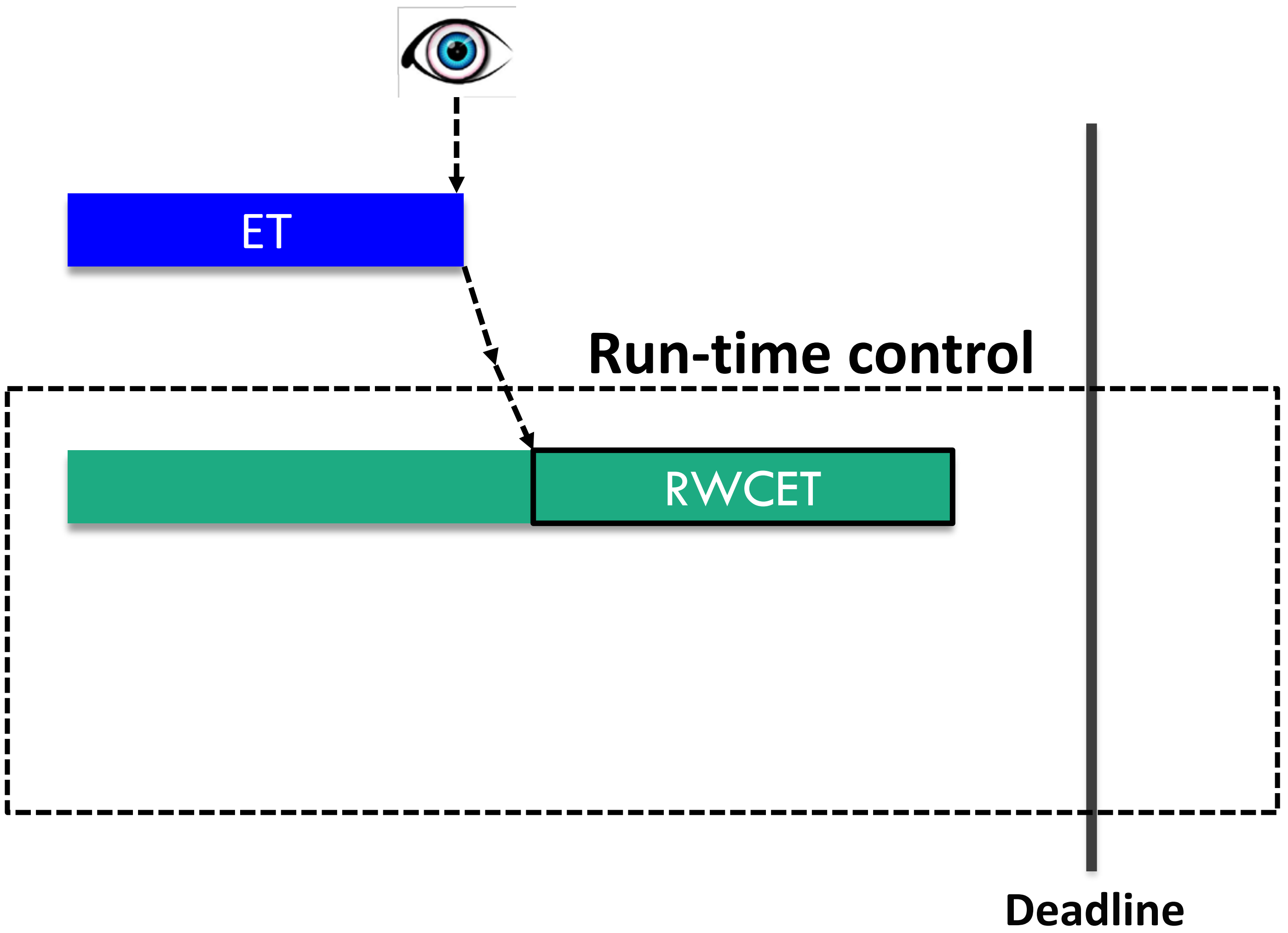
Point	d_{h-x}	w_x
S	0	-
n_1	d_{S-n1}	-
f_1	d_{S-f1}	-
n_2	d_{f1-n2}	-
c	d_{f1-c}	w_c
n_3	d_{c-n3}	-
n_4	d_{S-n4}	-

Run-time control

Observation point



Observation point

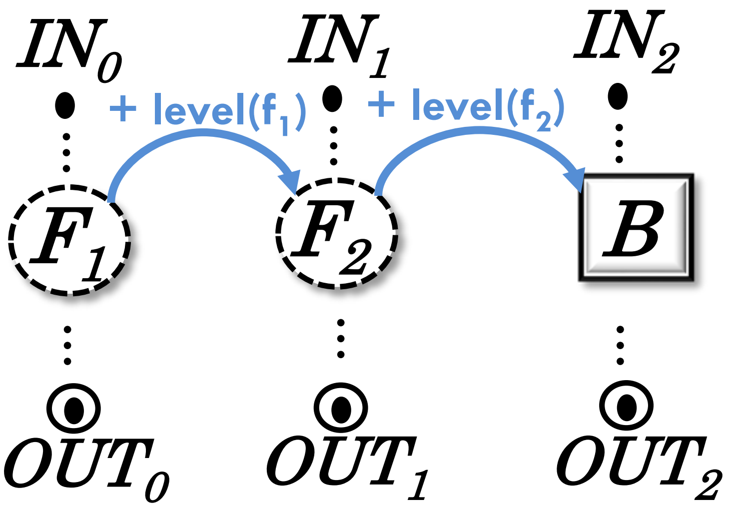


RWCET(x) computation

- *R[ℓ]:*
  - *ℓ=level(x)+offset*
  - Offset: Sum function entry levels

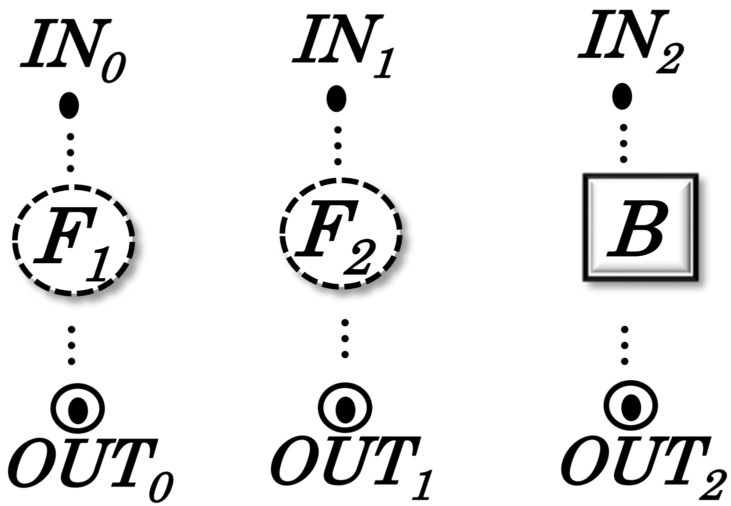
RWCET(x) computation

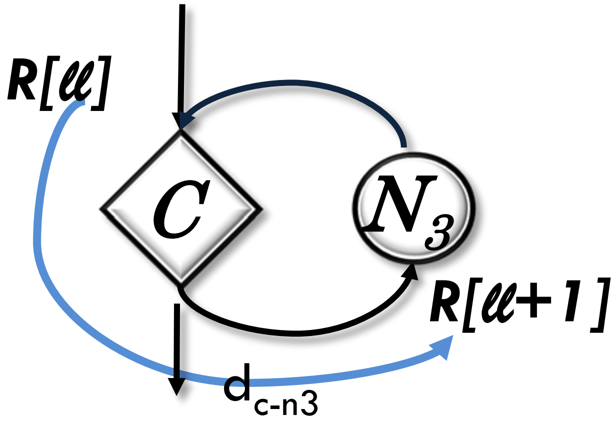
- *R[ℓ]:*
  - *ℓ=level(x)+offset*
  - Offset: Sum function entry levels
- Function entry:
  - *offset = offset + level(f_entry)*



RWCET(x) computation

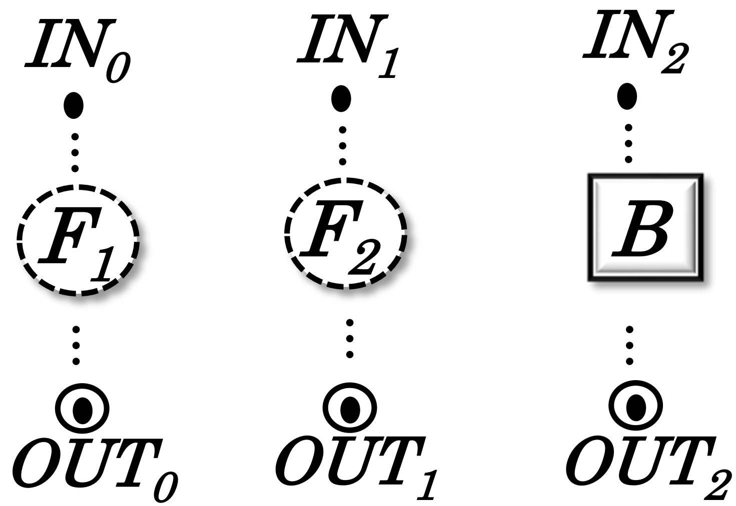
- *R[ℓ]:*
  - *ℓ=level(x)+offset*
  - Offset: Sum function entry levels
- Function entry:
  - *offset = offset + level(f_entry)*
- Function body:
  - Forward direction
    - $R[\ell+1] = R[\ell] - d_{c\text{-}x}$

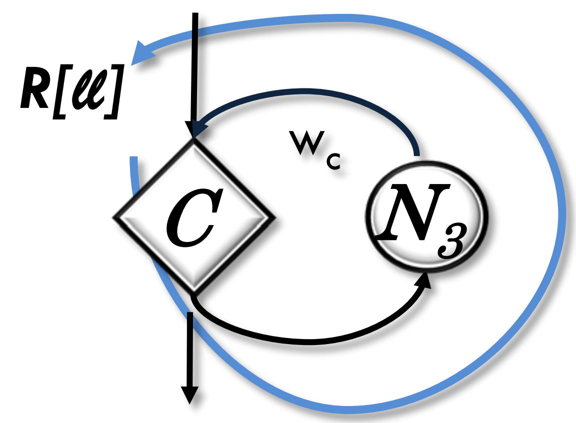




RWCET(x) computation

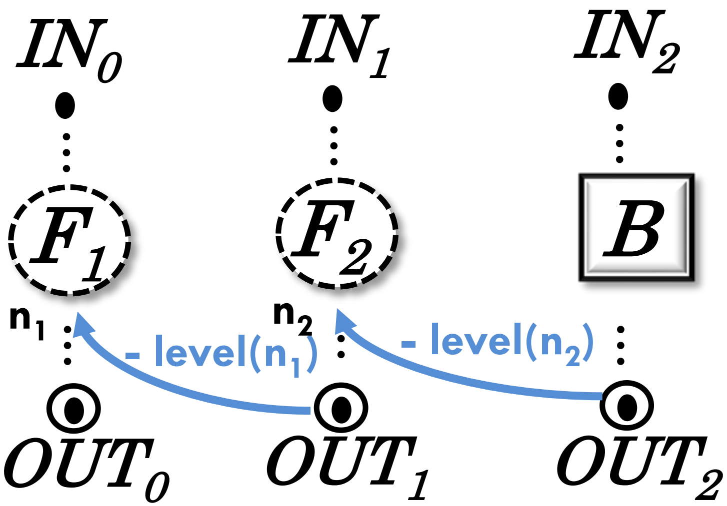
- *R[ℓ]:*
  - *ℓ=level(x)+offset*
  - Offset: Sum function entry levels
- Function entry:
  - *offset = offset + level(f_entry)*
- Function body:
  - Forward direction
    - $R[\ell+1] = R[\ell] - d_{c\text{-}x}$
  - Backward direction
    - $R[\ell] = R[\ell] - w_c$

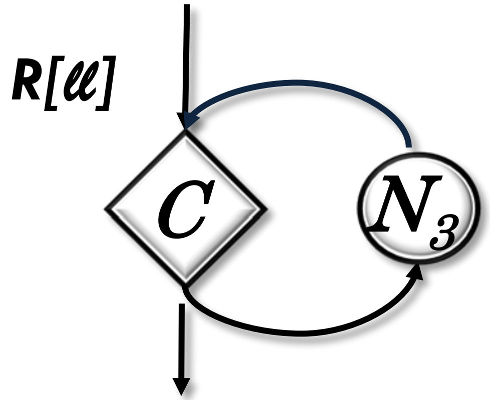




RWCET(x) computation

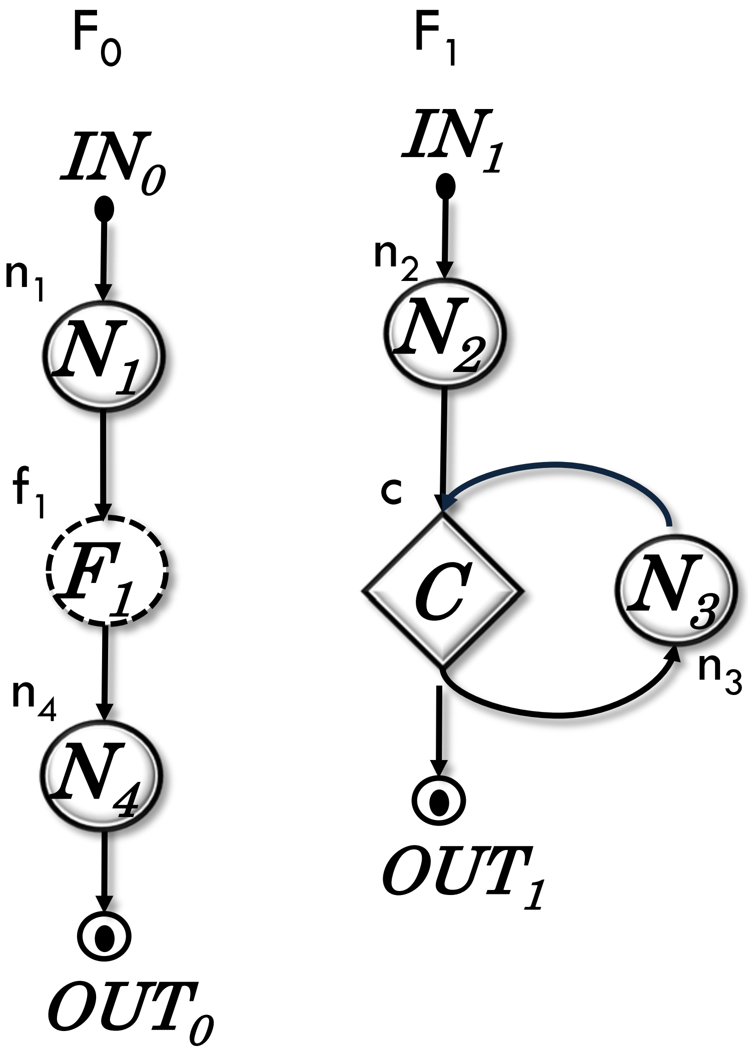
- *R[ℓ]:*
  - *ℓ=level(x)+offset*
  - Offset: Sum function entry levels
- Function entry:
  - *offset = offset + level(f_entry)*
- Function body:
  - Forward direction
    - *R[ℓ+1] = R[ℓ] – $d_{c\text{-}x}$*
  - Backward direction
    - *R[ℓ] = R[ℓ] – w_c*
- Function exit:
  - *offset = offset - level(f_exit)*





Example: RWCET(x)

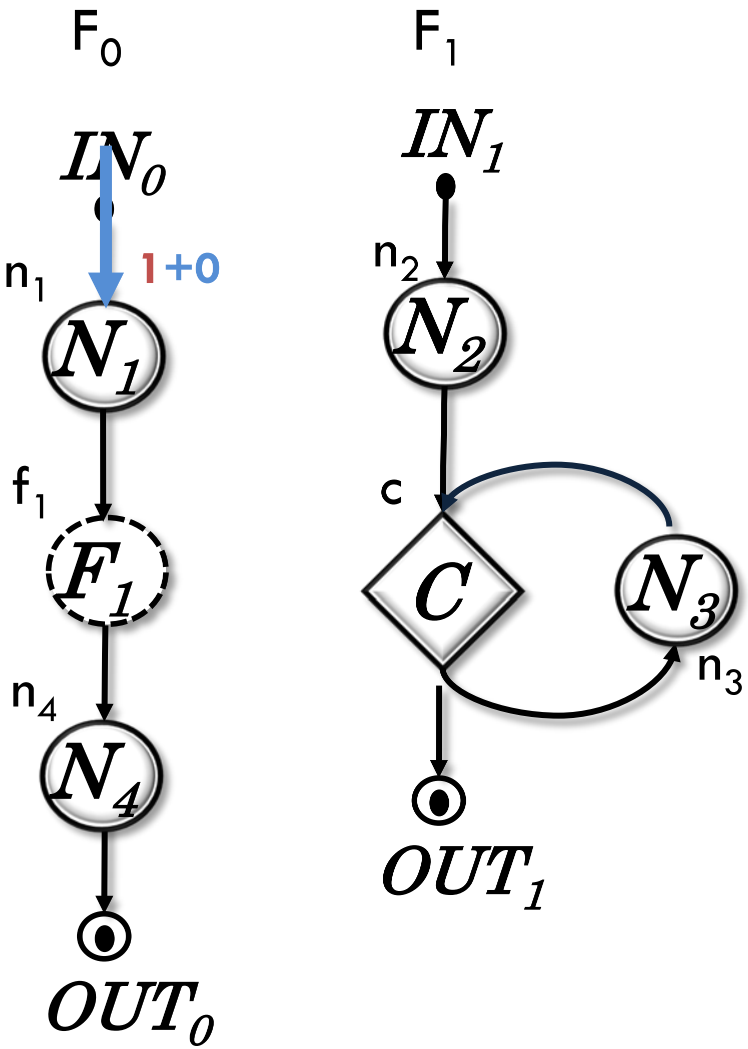
Initialization: S



x	RWCET(x)	Local Level	Offset
S	R[0]=WCET	0	0

Example: RWCET(x)

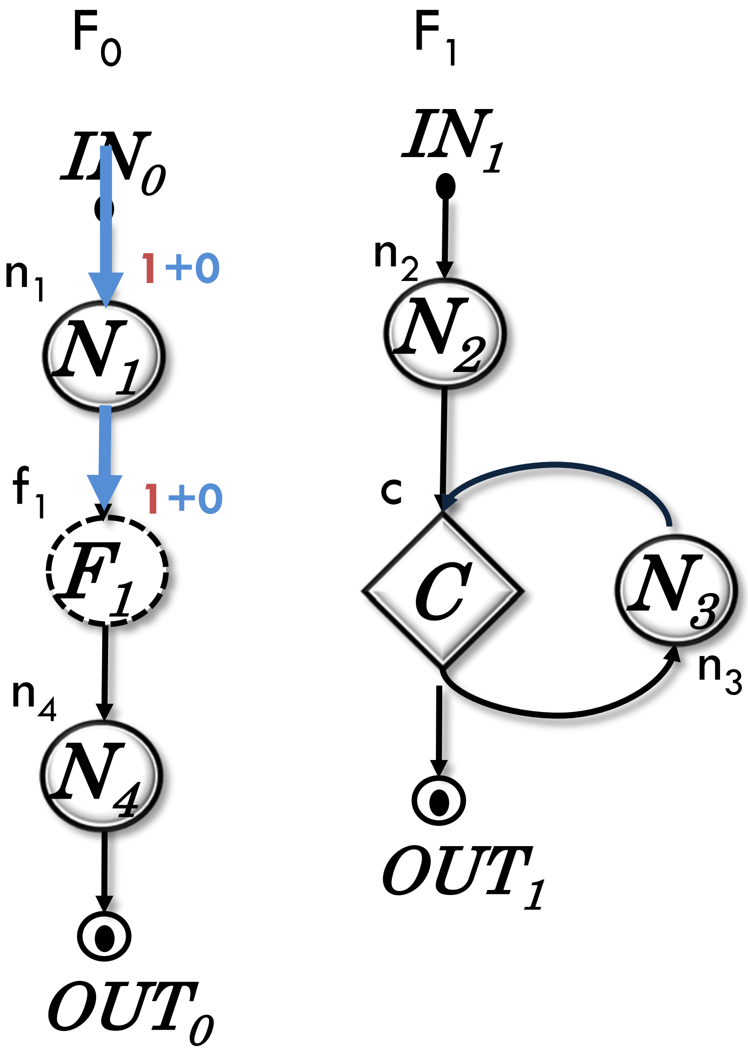
Initialization: S



x	RWCET(x)	Local Level	Offset
S	R[0]=WCET	0	0
n_1	$R[1]=R[0]-d_{S\text{-}n1}$	1	0

Example: RWCET(x)

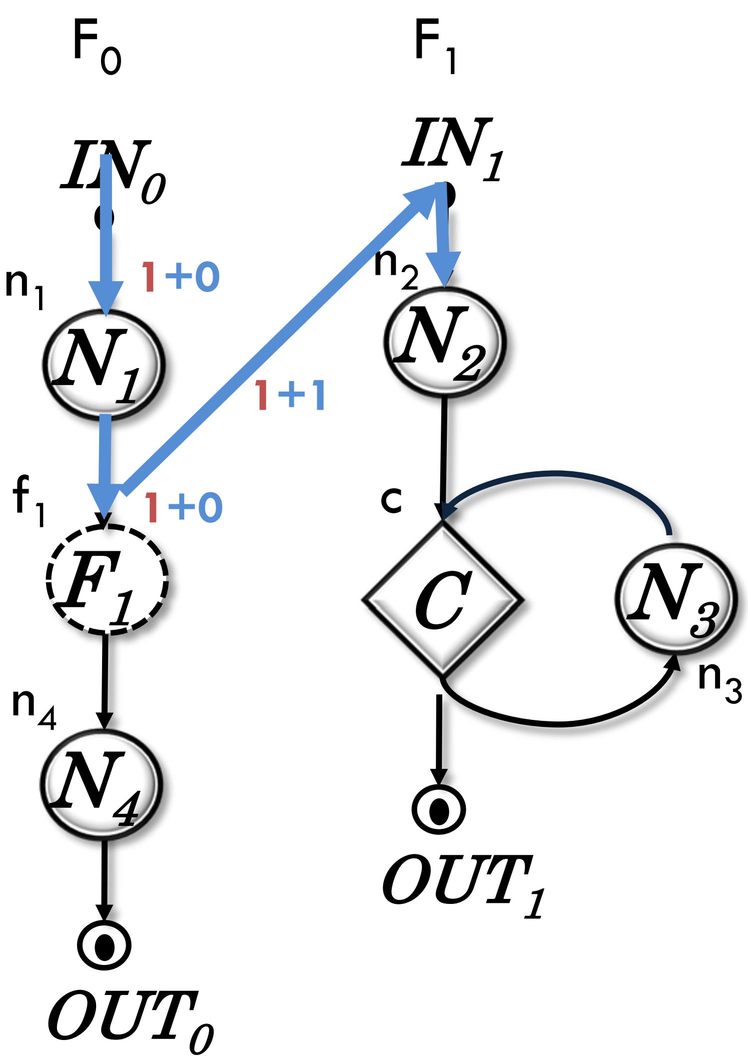
Initialization: S



x	RWCET(x)	Local Level	Offset
S	R[0]=WCET	0	0
n_1	$R[1]=R[0]-d_{S-n1}$	1	0
f_1	$R[1]=R[0]-d_{S-f1}$	1	0

Example: RWCET(x)

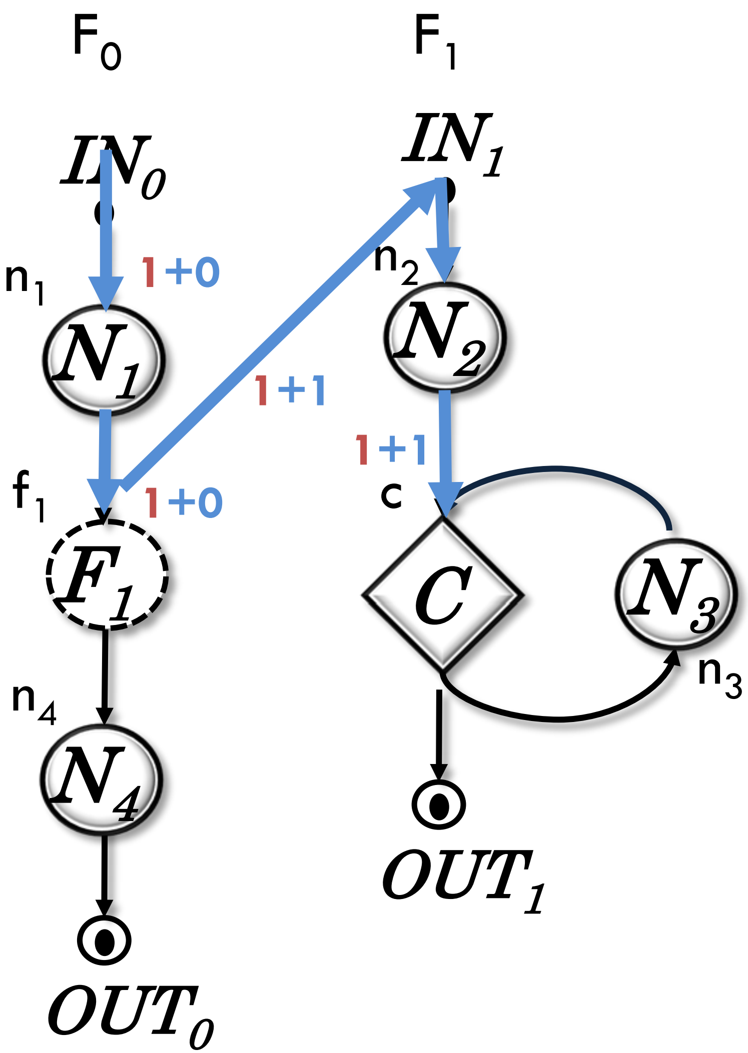
Initialization: S



x	RWCET(x)	Local Level	Offset
S	R[0]=WCET	0	0
n_1	R[1]=R[0]-$d_{S\text{-}n1}$	1	0
f_1	R[1]=R[0]-$d_{S\text{-}f1}$	1	0
n_2	R[2]=R[1]-$d_{f1\text{-}n2}$	2	1

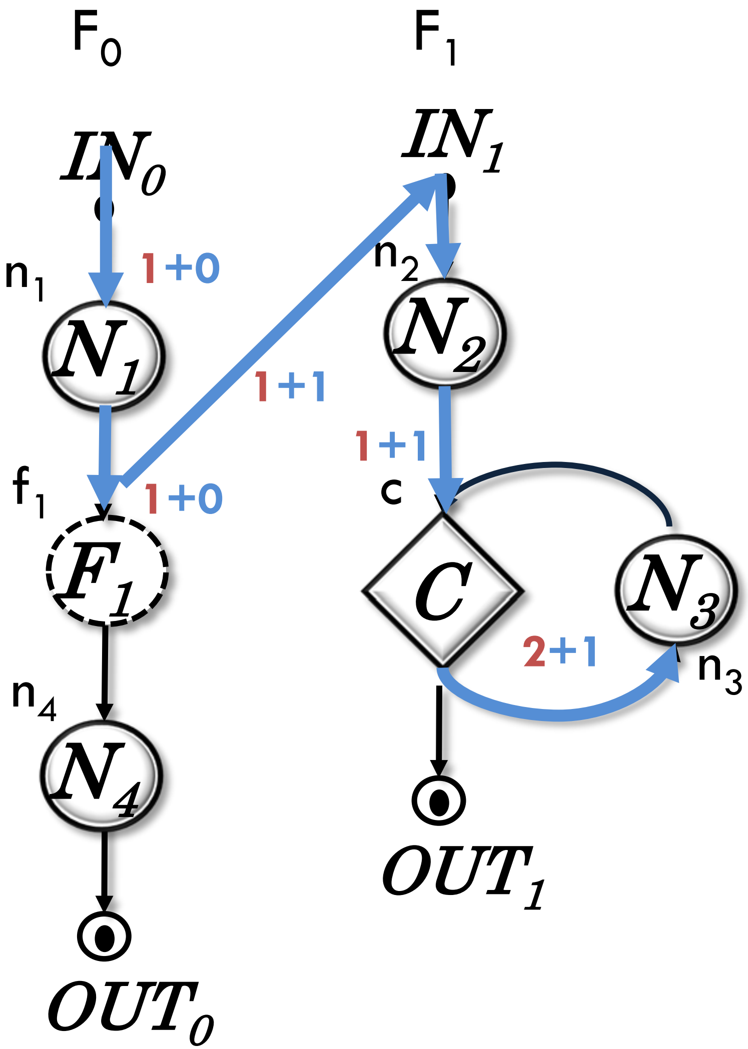
Example: RWCET(x)

Initialization: S



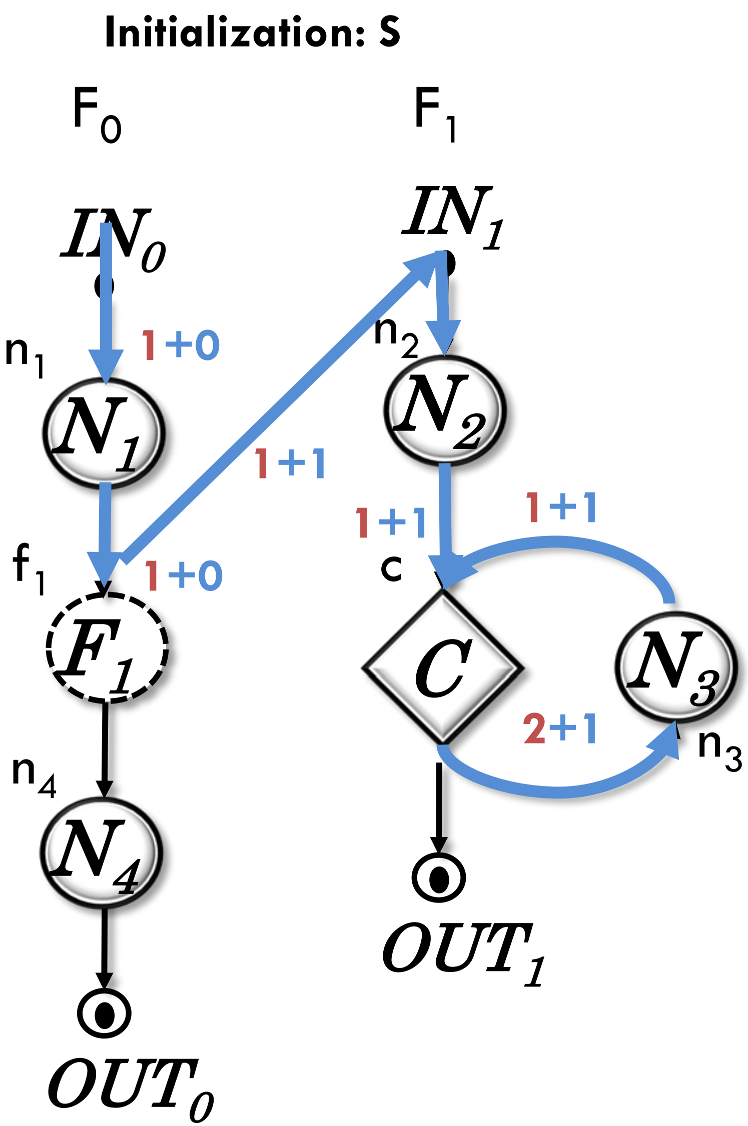
x	RWCET(x)	Local Level	Offset
S	R[0]=WCET	0	0
n_1	$R[1]=R[0]-d_{S-n1}$	1	0
f_1	$R[1]=R[0]-d_{S-f1}$	1	0
n_2	$R[2]=R[1]-d_{f1-n2}$	2	1
c	$R[2]=R[1]-d_{f1-c}$	2	1

Example: RWCET(x)



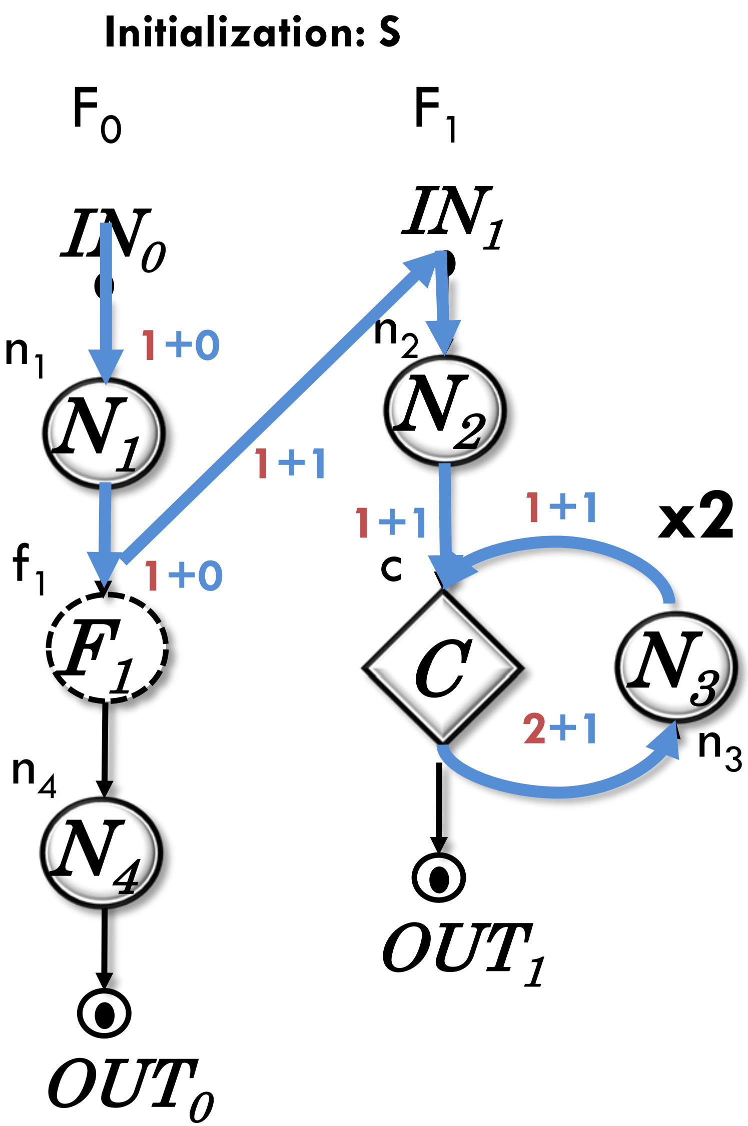
x	RWCET(x)	Local Level	Offset
S	R[0]=WCET	0	0
n_1	$R[1]=R[0]-d_{S-n1}$	1	0
f_1	$R[1]=R[0]-d_{S-f1}$	1	0
n_2	$R[2]=R[1]-d_{f1-n2}$	2	1
c	$R[2]=R[1]-d_{f1-c}$	2	1
n_3	$R[3]=R[2]-d_{c-n3}$	3	1

Example: RWCET(x)



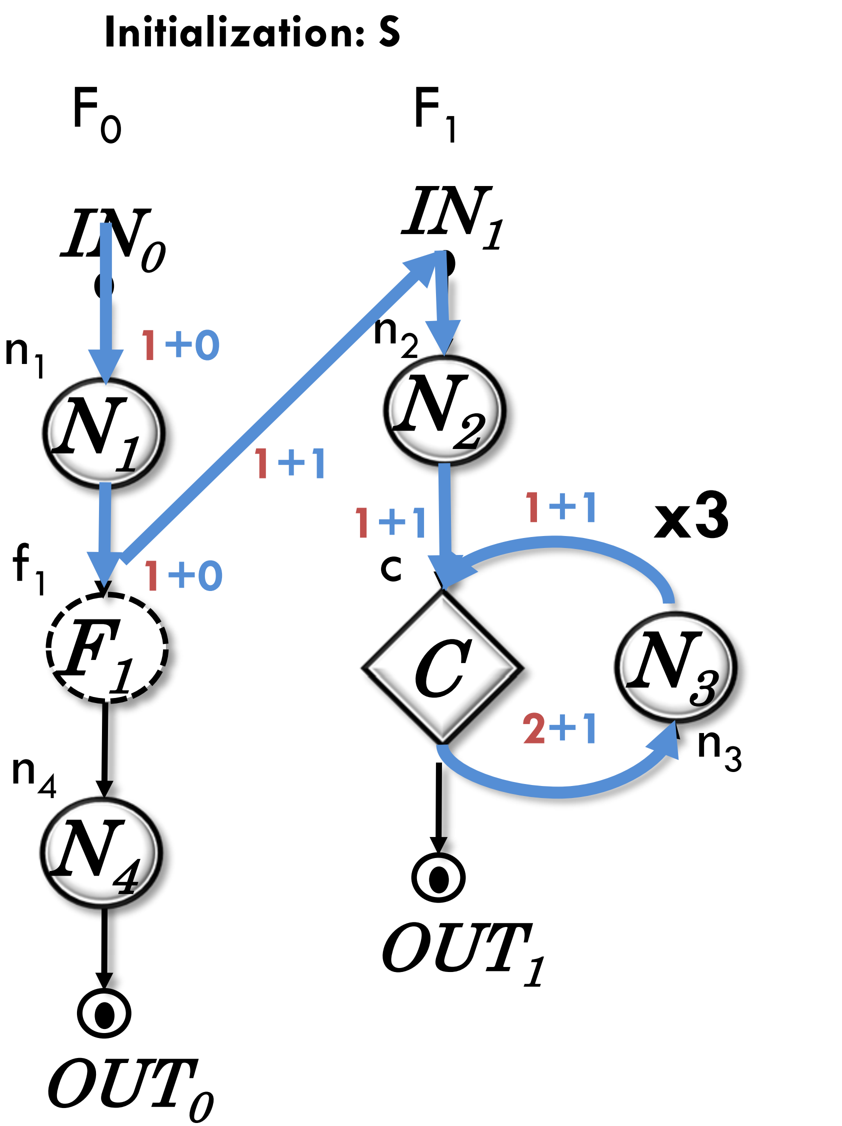
x	RWCET(x)	Local Level	Offset
S	R[0]=WCET	0	0
n_1	$R[1]=R[0]-d_{S-n1}$	1	0
f_1	$R[1]=R[0]-d_{S-f1}$	1	0
n_2	$R[2]=R[1]-d_{f1-n2}$	2	1
c	$R[2]=R[1]-d_{f1-c}$	2	1
n_3	$R[3]=R[2]-d_{c-n3}$	3	1
c	$R[2]=R[2]-w_c$	2	1

Example: RWCET(x)



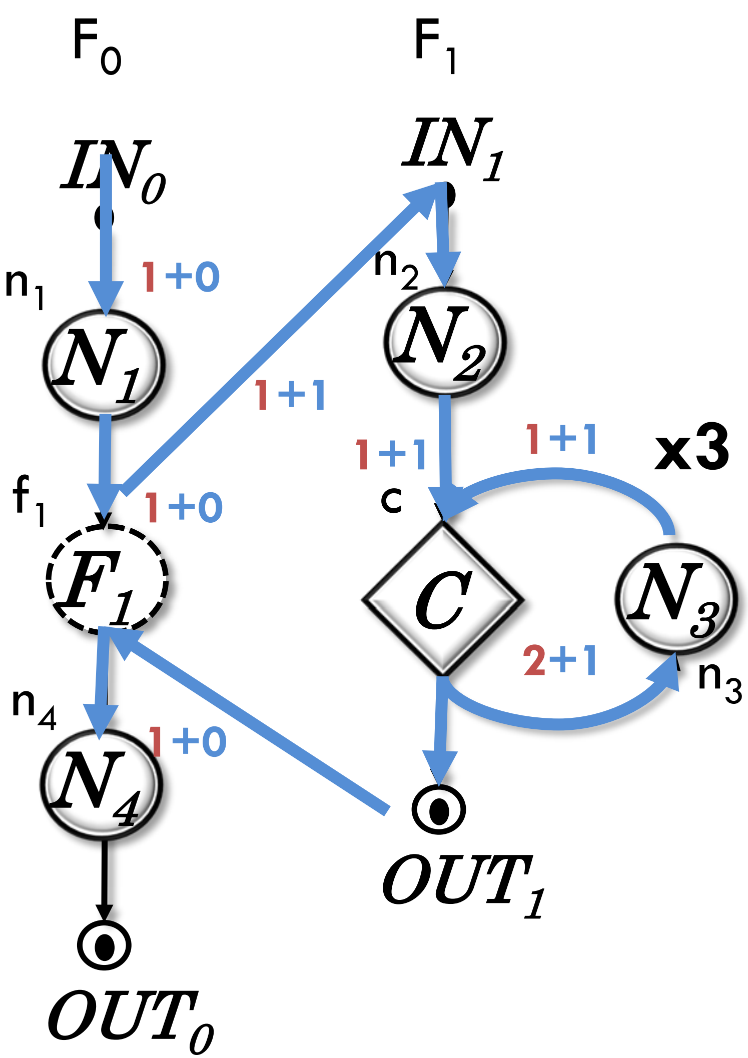
x	RWCET(x)	Local Level	Offset
S	R[0]=WCET	0	0
n_1	$R[1]=R[0]-d_{S-n1}$	1	0
f_1	$R[1]=R[0]-d_{S-f1}$	1	0
n_2	$R[2]=R[1]-d_{f1-n2}$	2	1
c	$R[2]=R[1]-d_{f1-c}$	2	1
n_3	$R[3]=R[2]-d_{c-n3}$	3	1
c	$R[2]=R[2]-w_c$	2	1
n_3	$R[3]=R[2]-d_{c-n3}$	3	1
c	$R[2]=R[2]-w_c$	2	1

Example: RWCET(x)



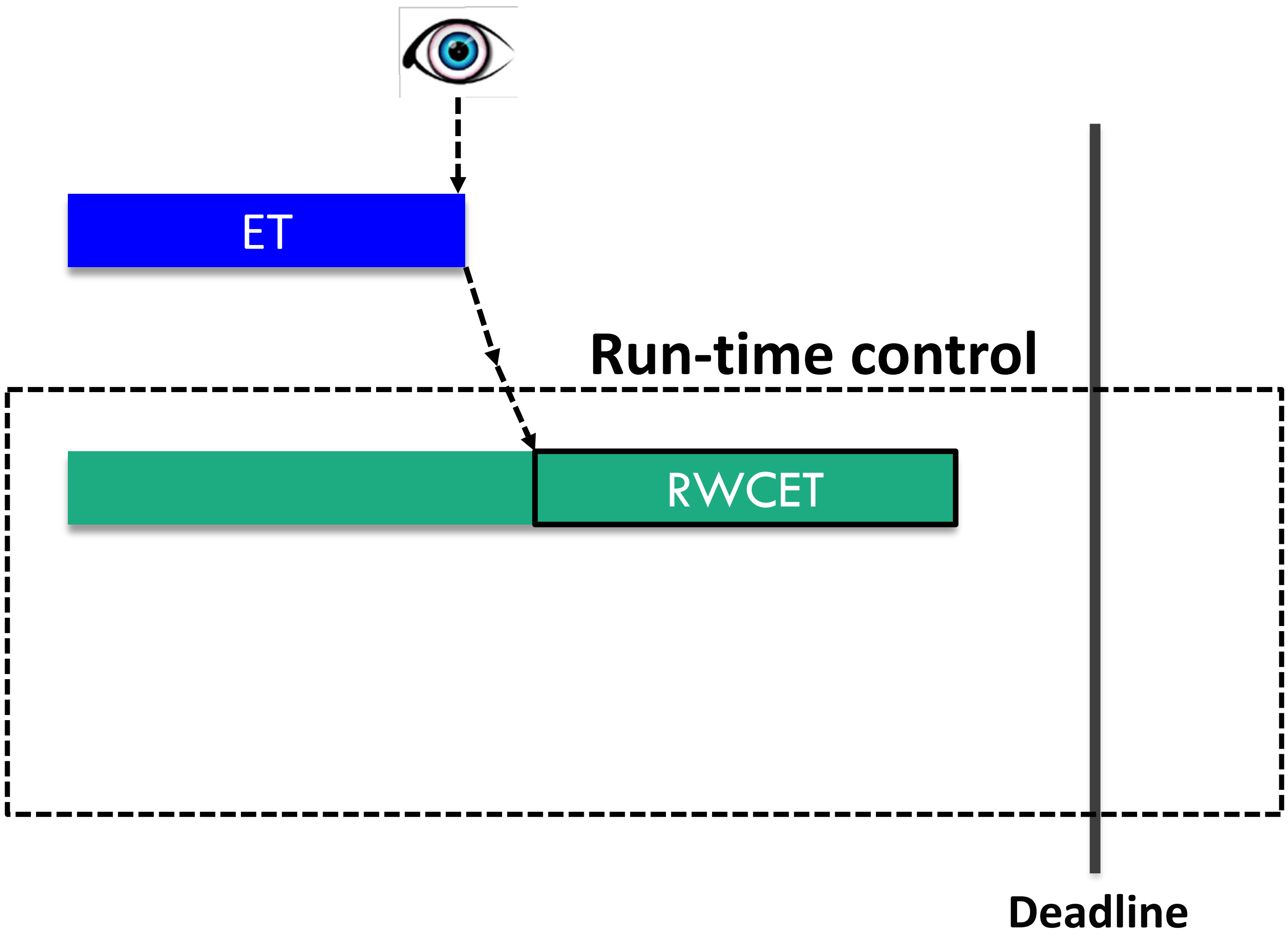
x	RWCET(x)	Local Level	Offset
S	R[0]=WCET	0	0
n_1	$R[1]=R[0]-d_{S-n1}$	1	0
f_1	$R[1]=R[0]-d_{S-f1}$	1	0
n_2	$R[2]=R[1]-d_{f1-n2}$	2	1
c	$R[2]=R[1]-d_{f1-c}$	2	1
n_3	$R[3]=R[2]-d_{c-n3}$	3	1
c	$R[2]=R[2]-w_c$	2	1
n_3	$R[3]=R[2]-d_{c-n3}$	3	1
c	$R[2]=R[2]-w_c$	2	1
n_3	$R[3]=R[2]-d_{c-n3}$	3	1
c	$R[2]=R[2]-w_c$	2	1

Example: RWCET(x)

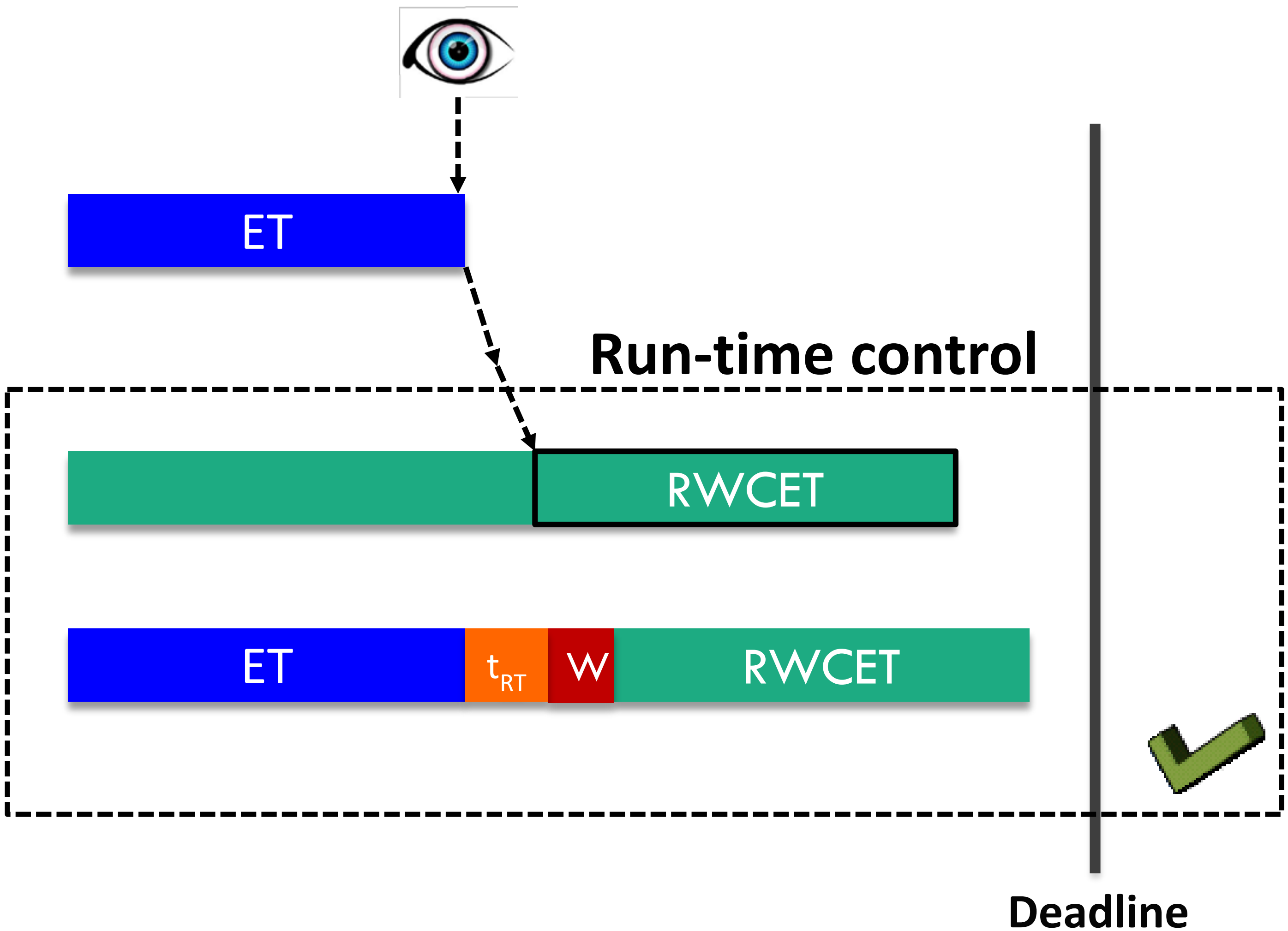


x	RWCET(x)	Local Level	Offset
S	R[0]=WCET	0	0
n_1	$R[1]=R[0]-d_{S-n1}$	1	0
f_1	$R[1]=R[0]-d_{S-f1}$	1	0
n_2	$R[2]=R[1]-d_{f1-n2}$	2	1
c	$R[2]=R[1]-d_{f1-c}$	2	1
n_3	$R[3]=R[2]-d_{c-n3}$	3	1
c	$R[2]=R[2]-w_c$	2	1
n_3	$R[3]=R[2]-d_{c-n3}$	3	1
c	$R[2]=R[2]-w_c$	2	1
n_3	$R[3]=R[2]-d_{c-n3}$	3	1
c	$R[2]=R[2]-w_c$	2	1
n_4	$R[1]=R[0]-d_{S-n4}$	1	0

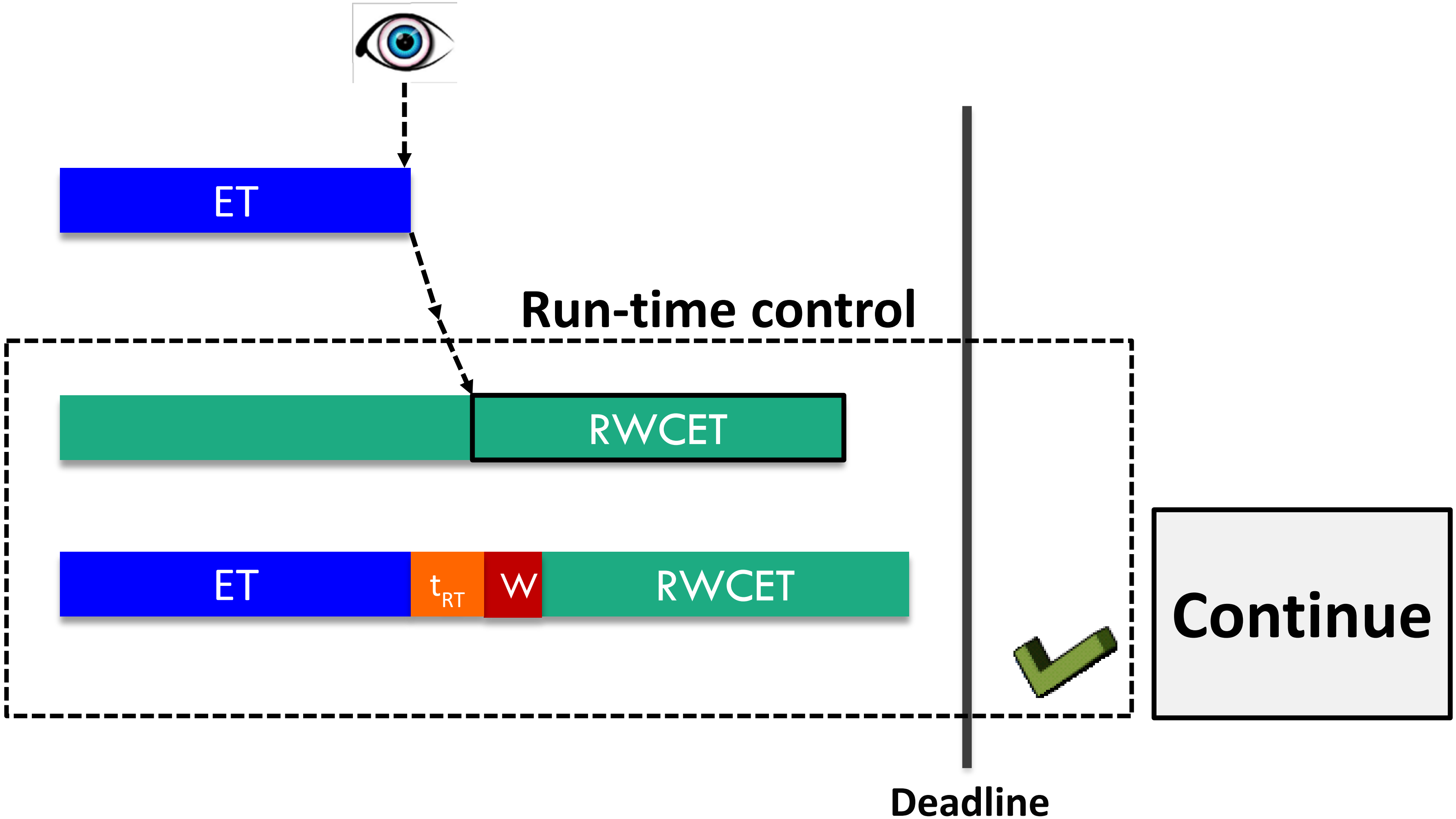
Observation point



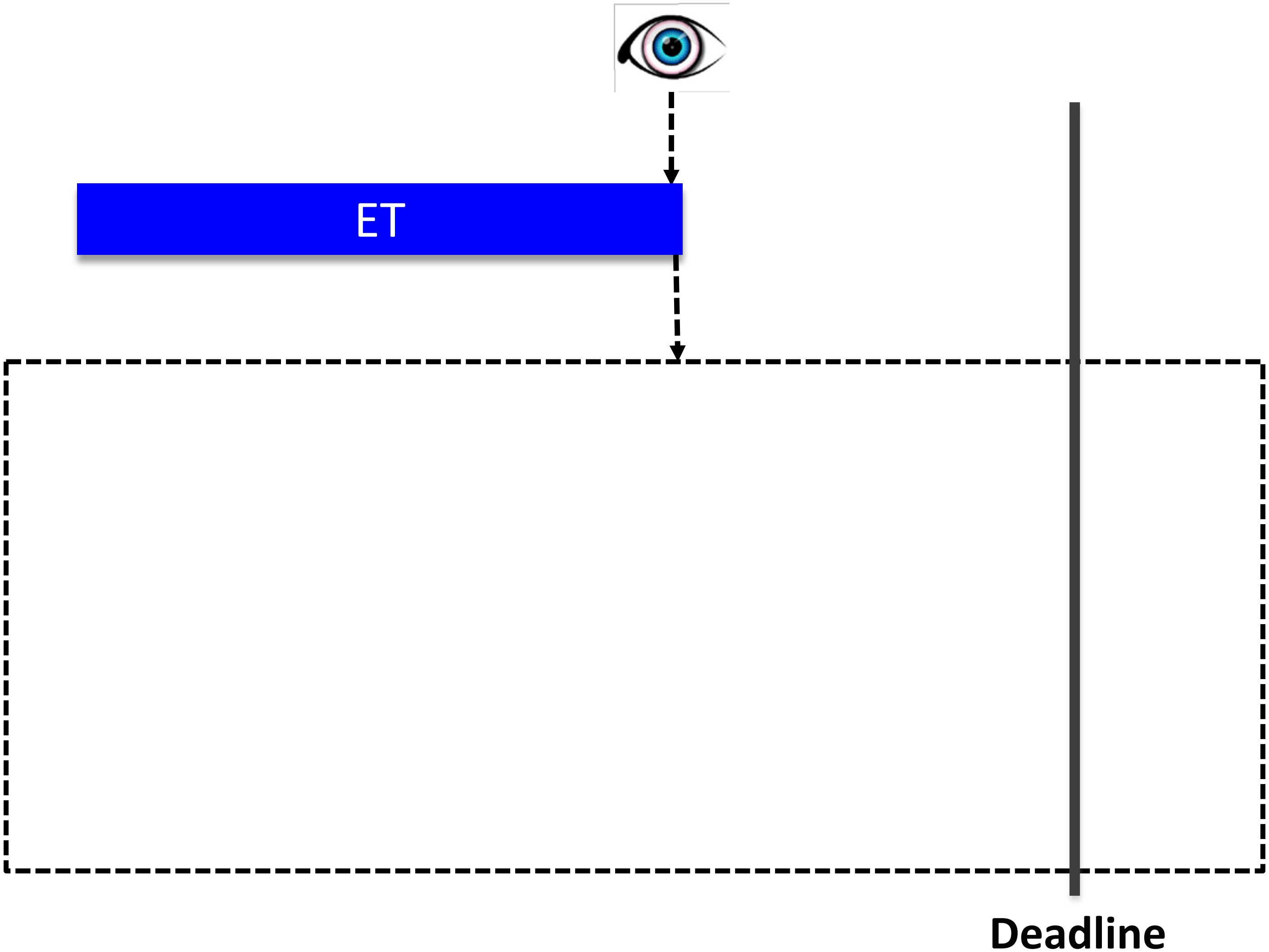
Observation point



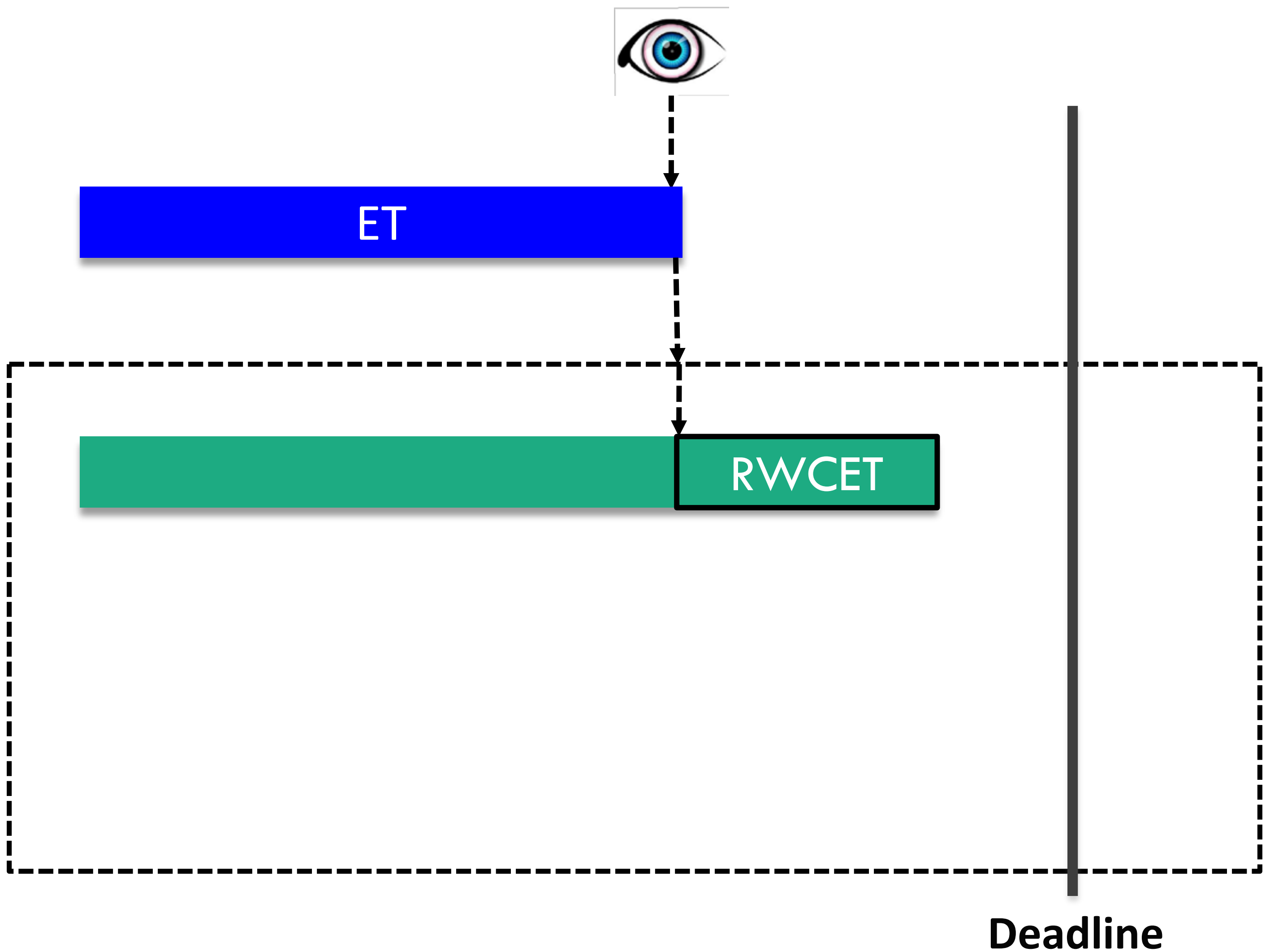
Observation point



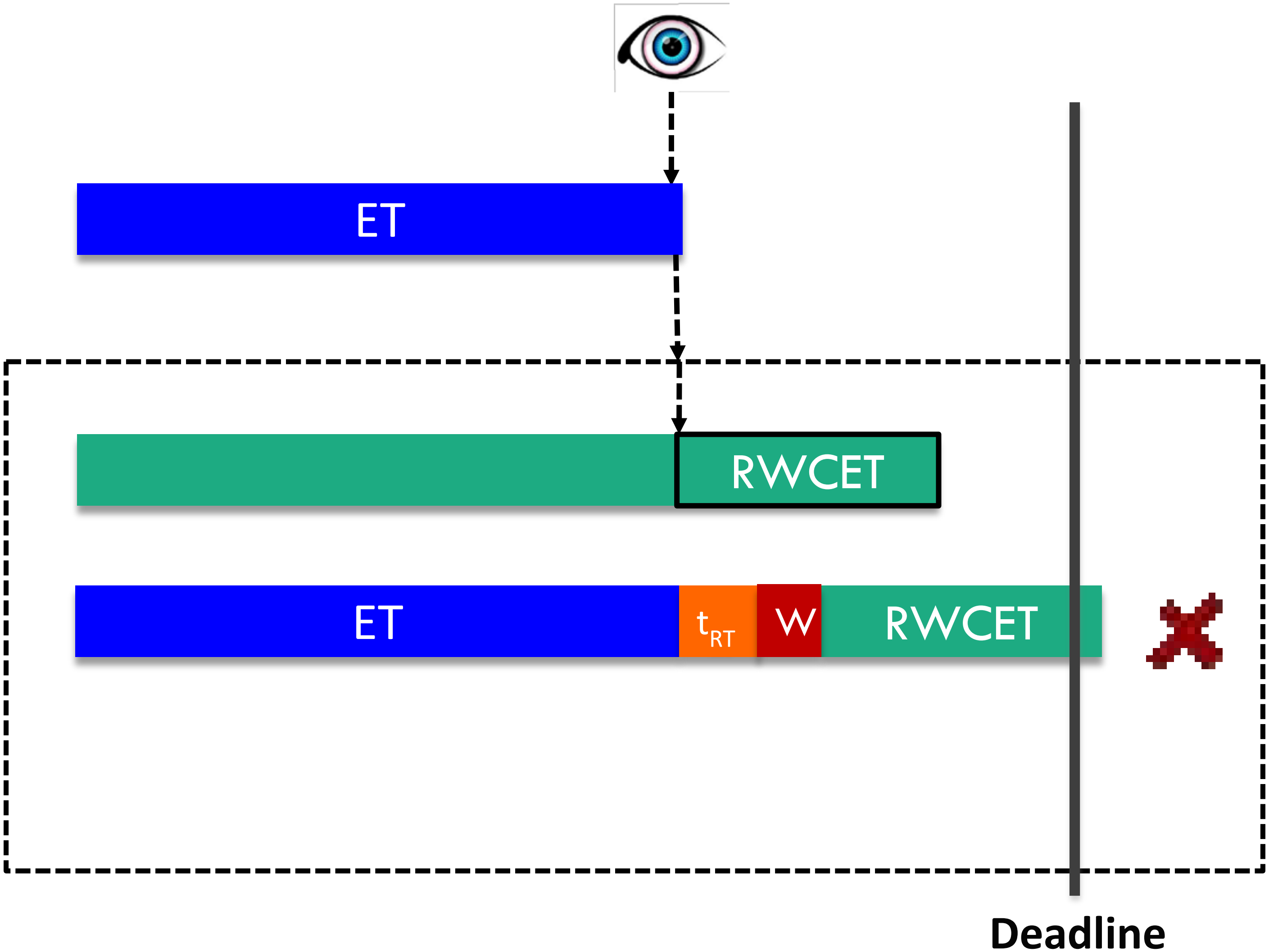
... later observation point



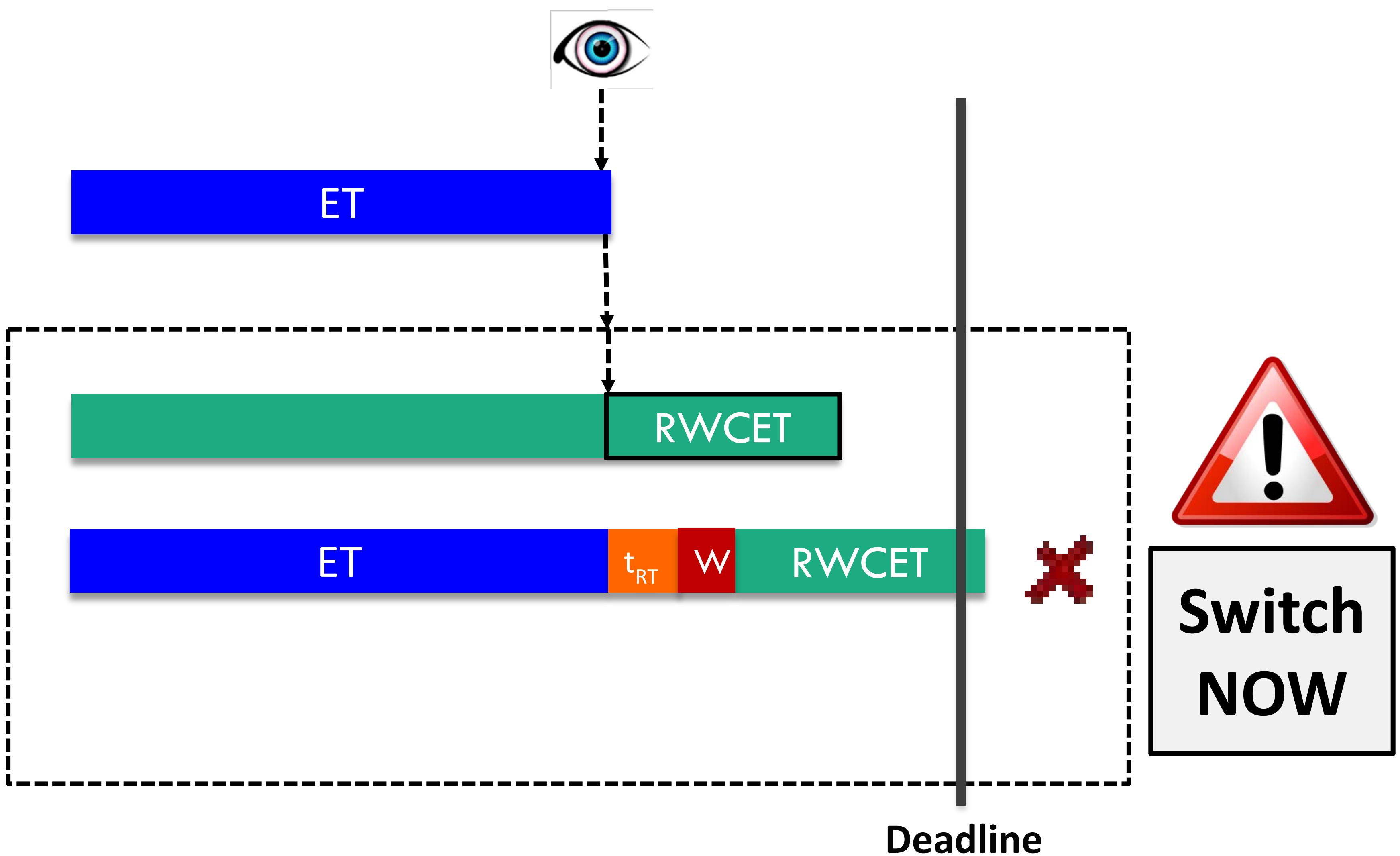
... later observation point



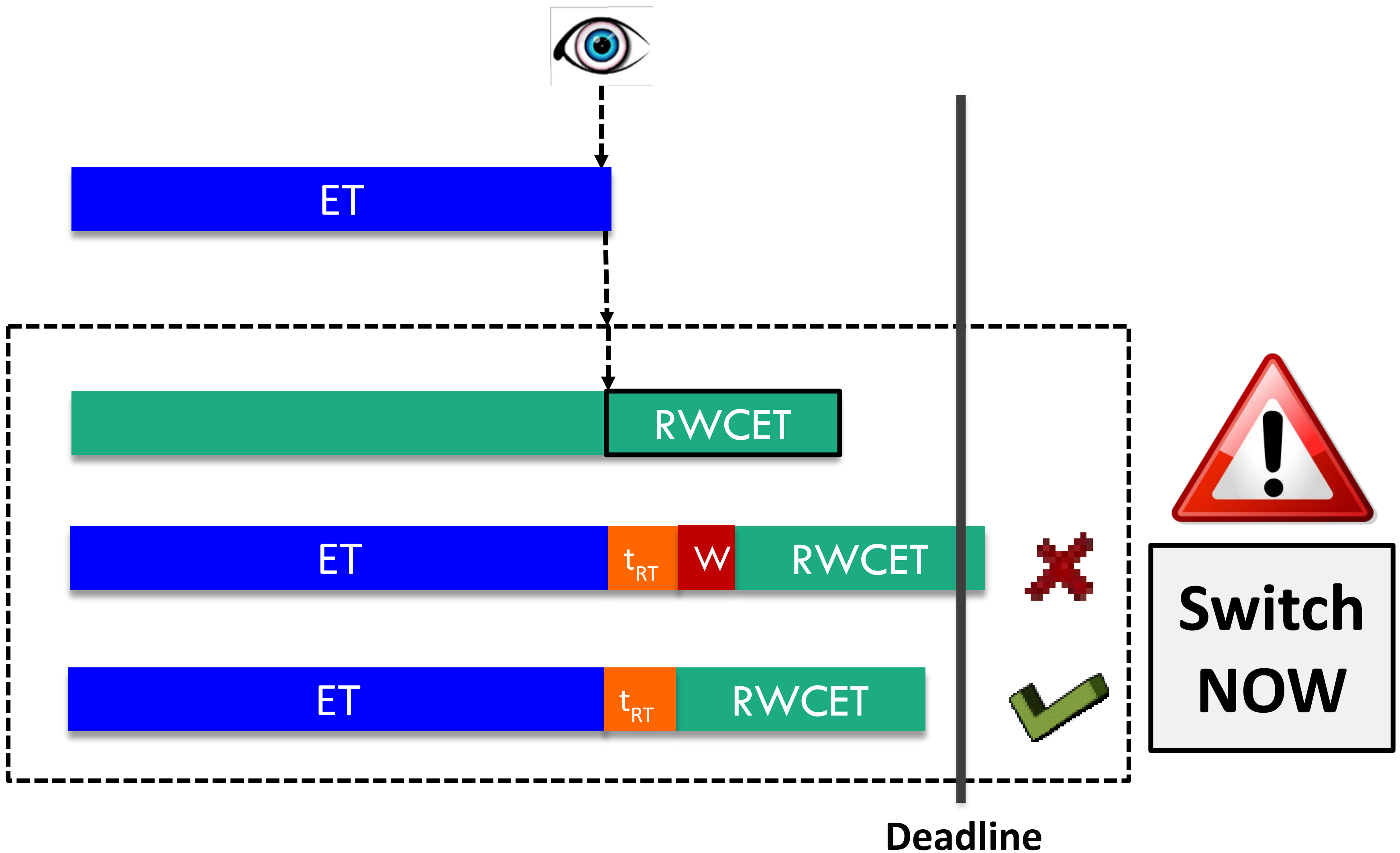
... later observation point



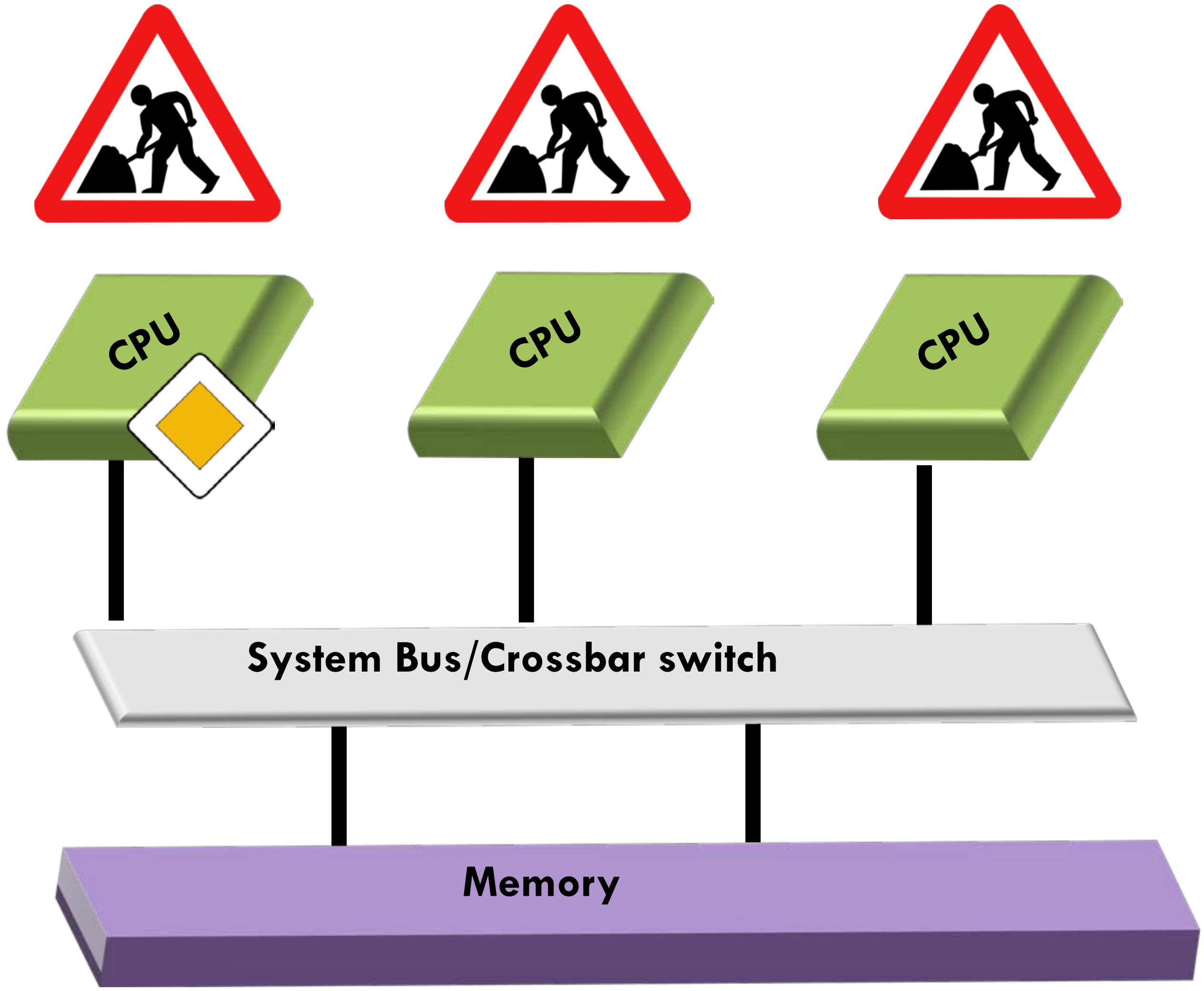
... later observation point



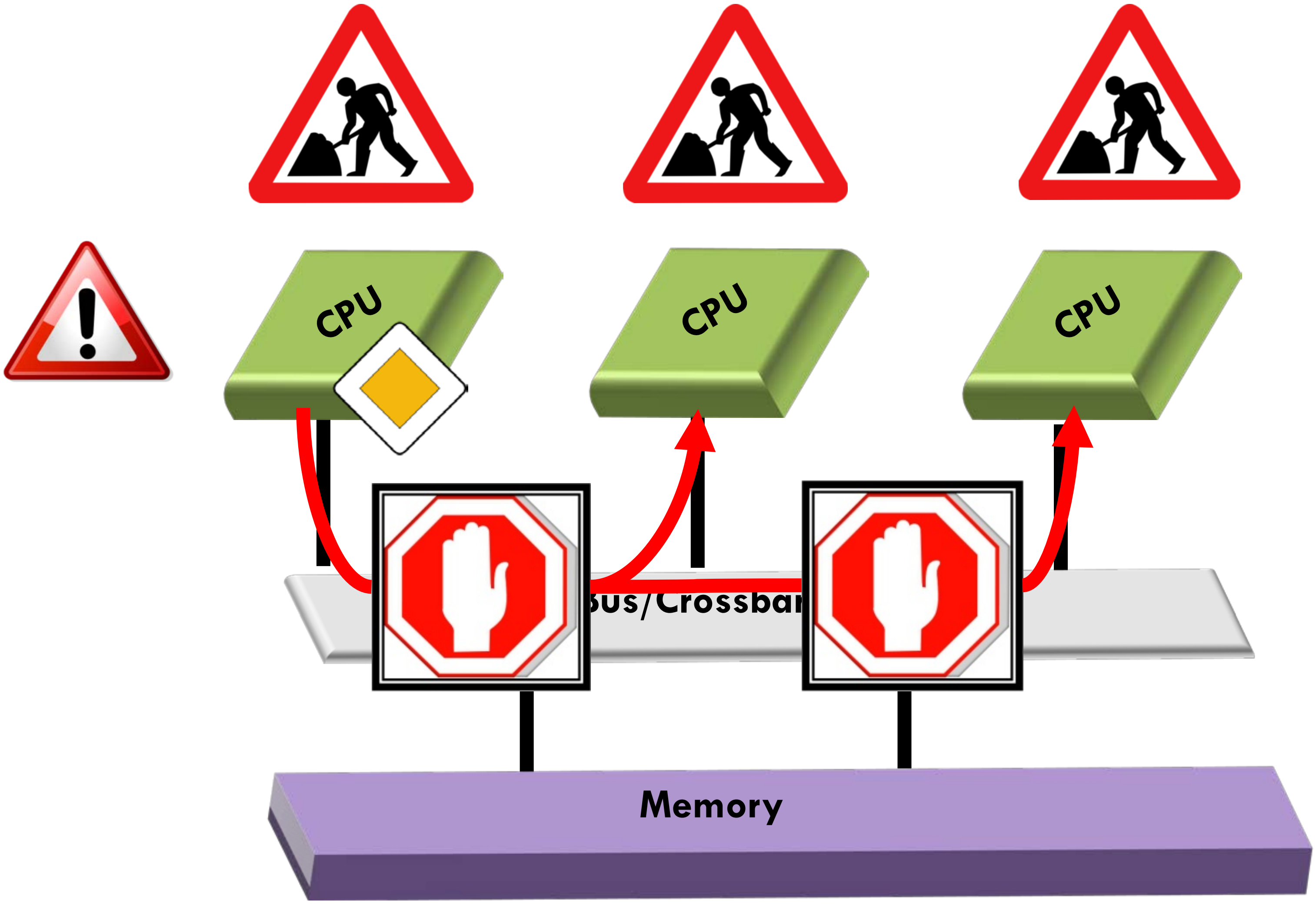
... later observation point



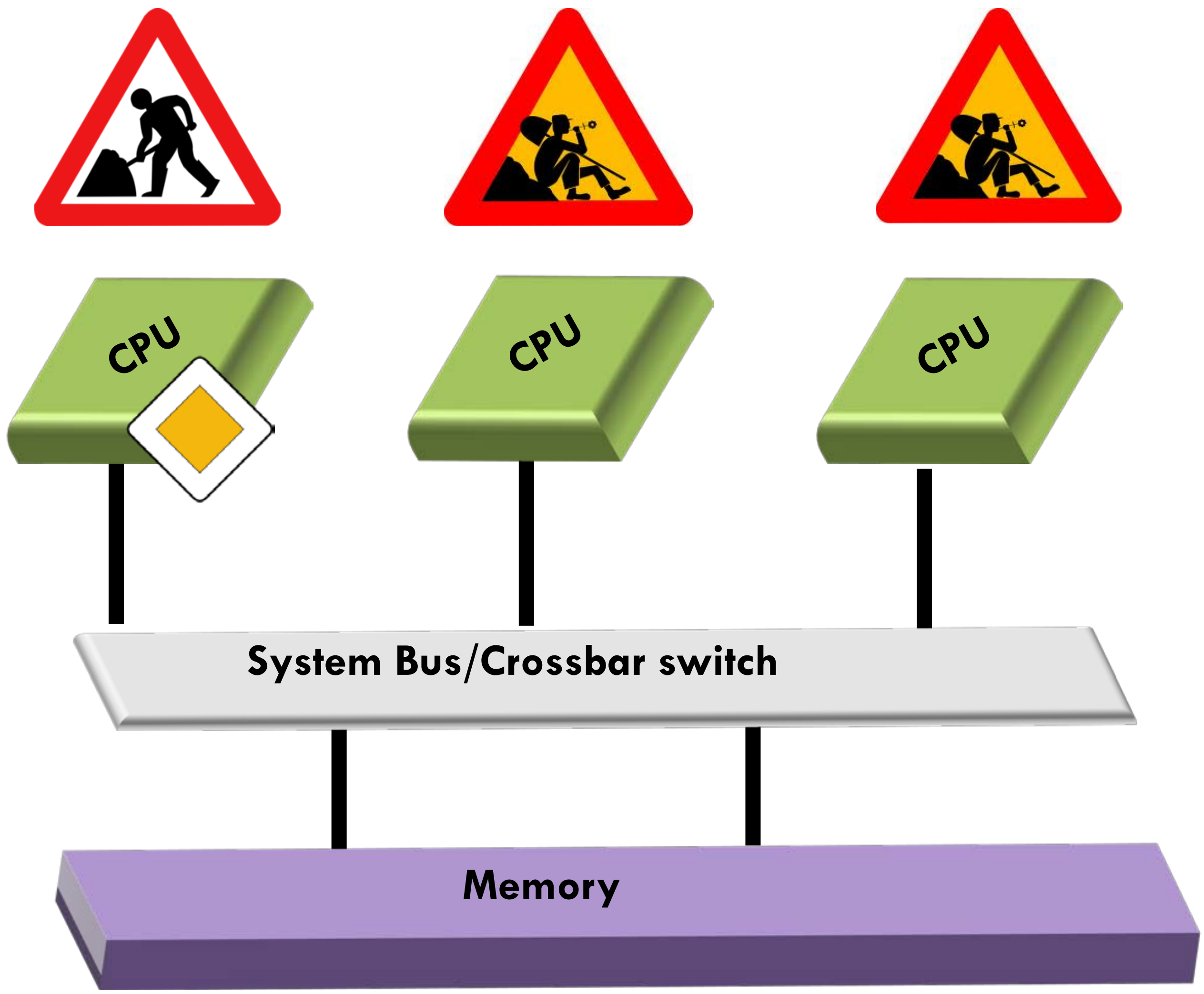
Switch to isolation



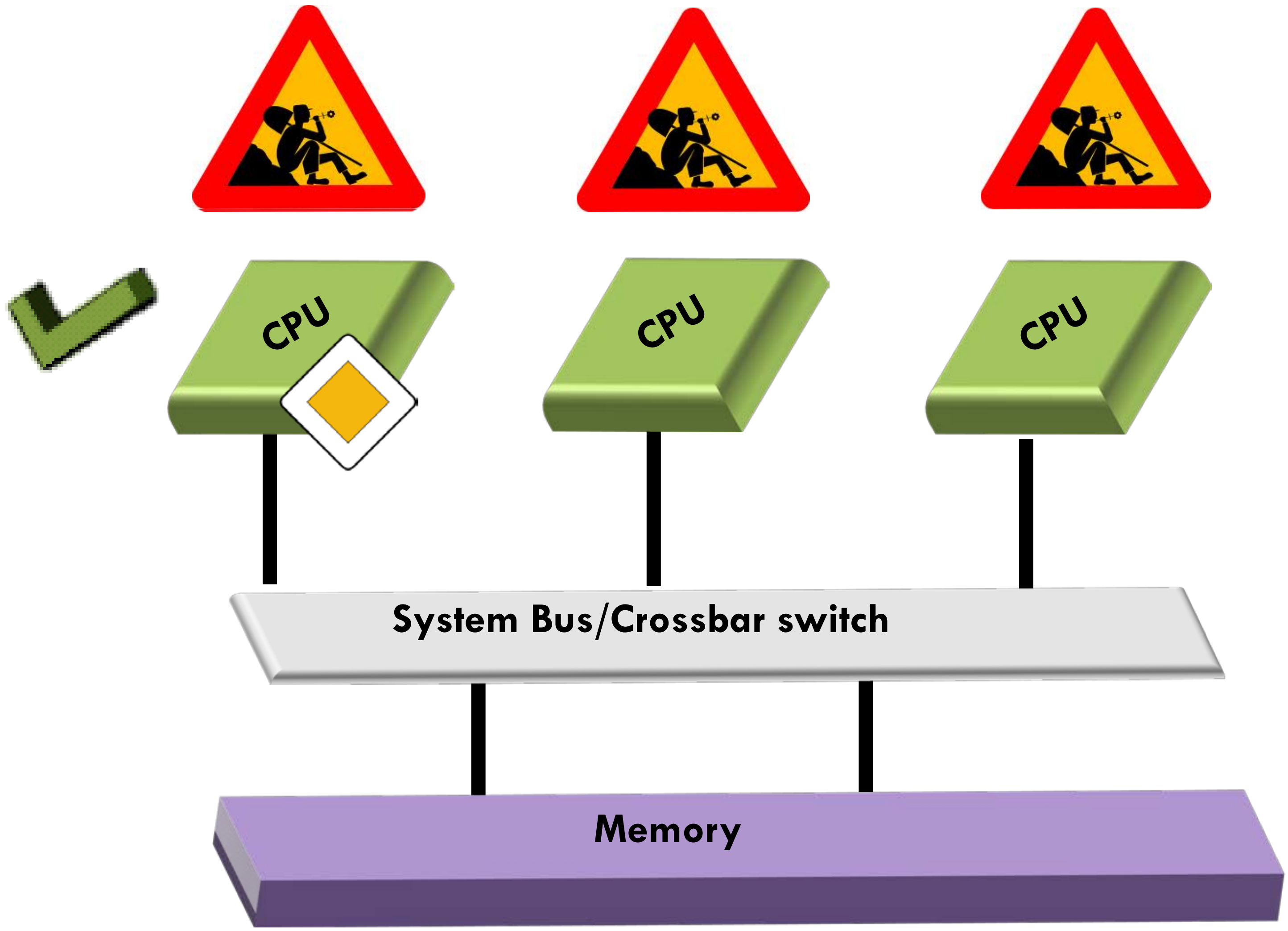
Switch to isolation



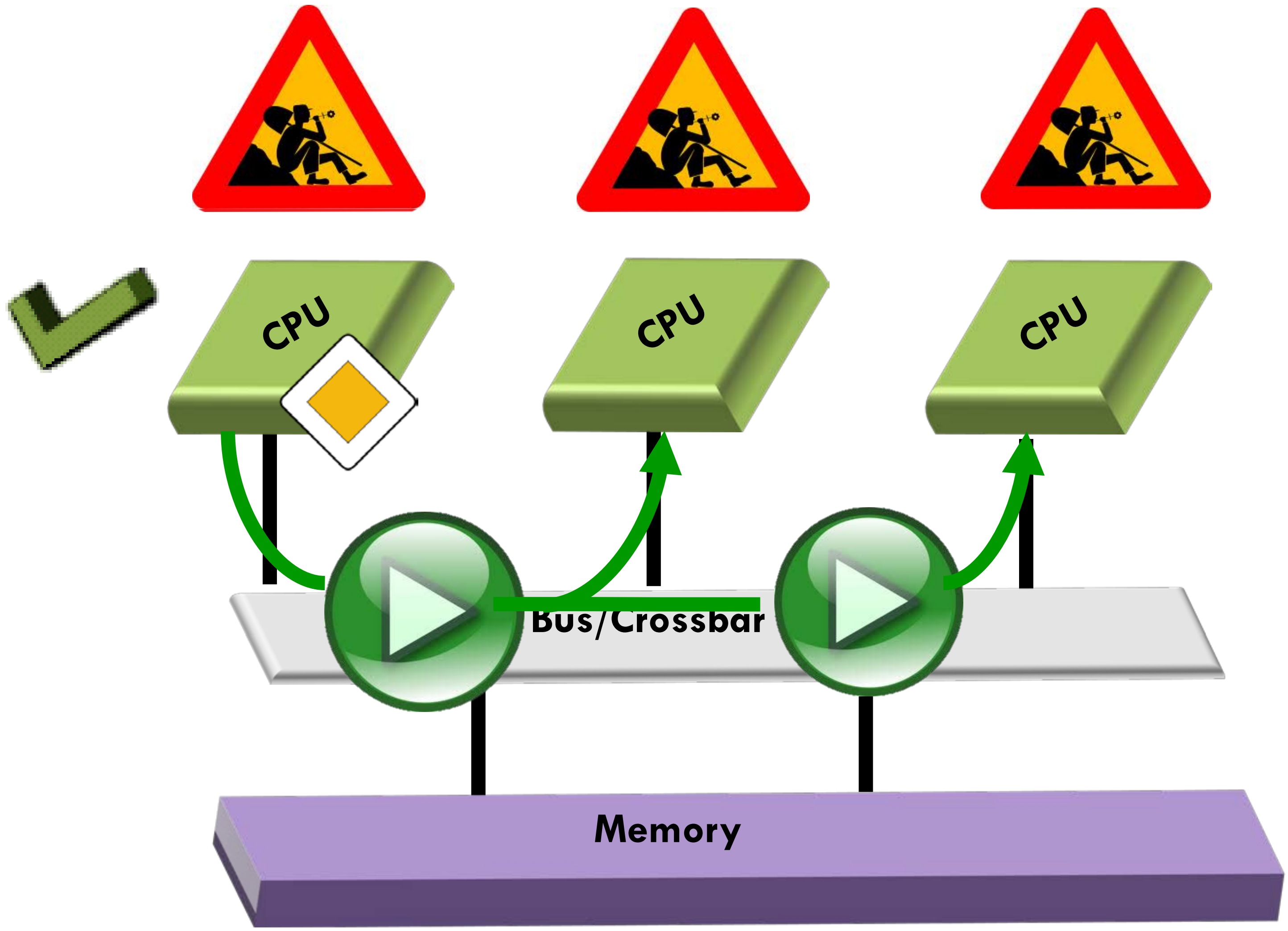
Switch to isolation



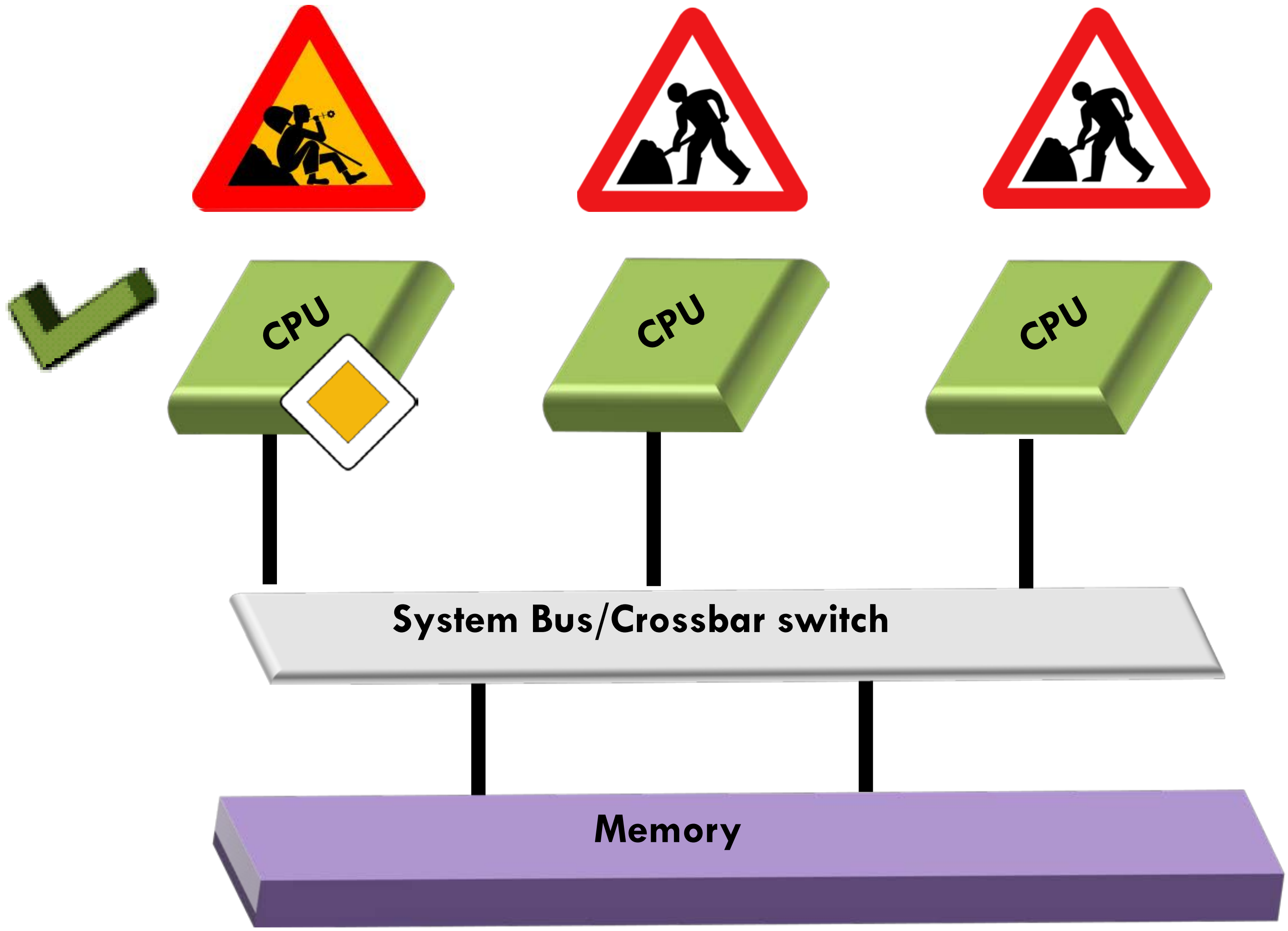
Switch to isolation



Switch to isolation



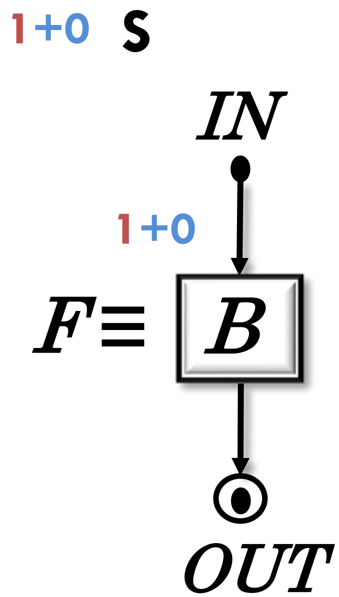
Switch to isolation



Proofs & Results

RWCET(x)

- Structural Induction
- Base case

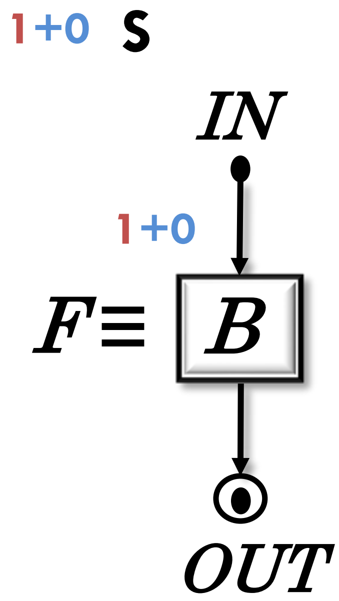


$R[0] = RWCET$

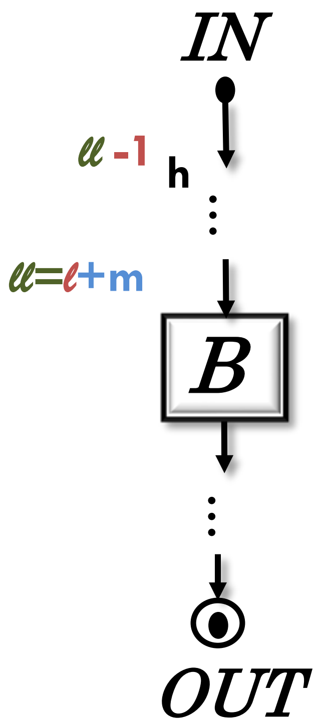
$R[1] = R[0] – d_{s\text{-}b} = R[0] - (RWCET - RWCET(b)) = RWCET(b)$

RWCET(x)

- Structural Induction
- Base case



- Hypothesis

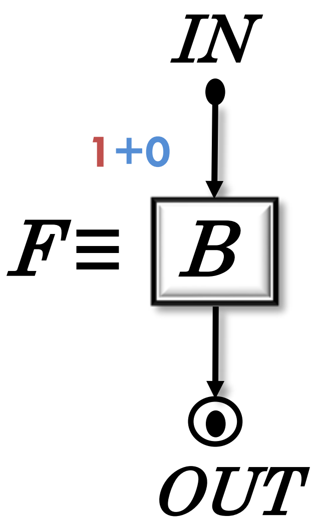


$R[\ell\ell] = RWCET(b)$
$R[\ell\ell\text{-}1] = RWCET(h)$

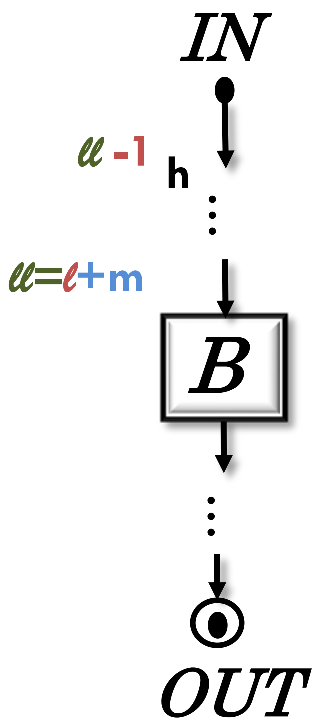
RWCET(x)

- Structural Induction
- Base case

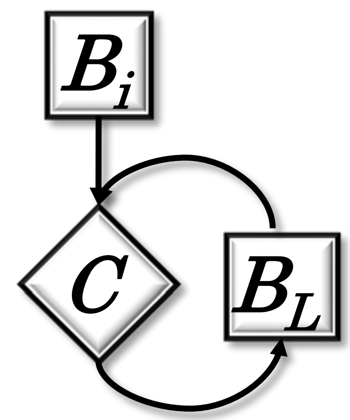
1+0 s



- Hypothesis



- Induction step:
  - Rewriting rules



$R[\ell\ell] = RWCET(b_i)$

$0 \rightarrow R[\ell\ell] = R[\ell\ell-1] - d_{h-c} = R[\ell\ell-1] - (RWCET(h) - RWCET(c)) = RWCET(c)$

$j \rightarrow R[\ell\ell] = R[\ell\ell] - w_c = RWCET(c) - j * w_c = RWCET(c, j)$

Safety condition

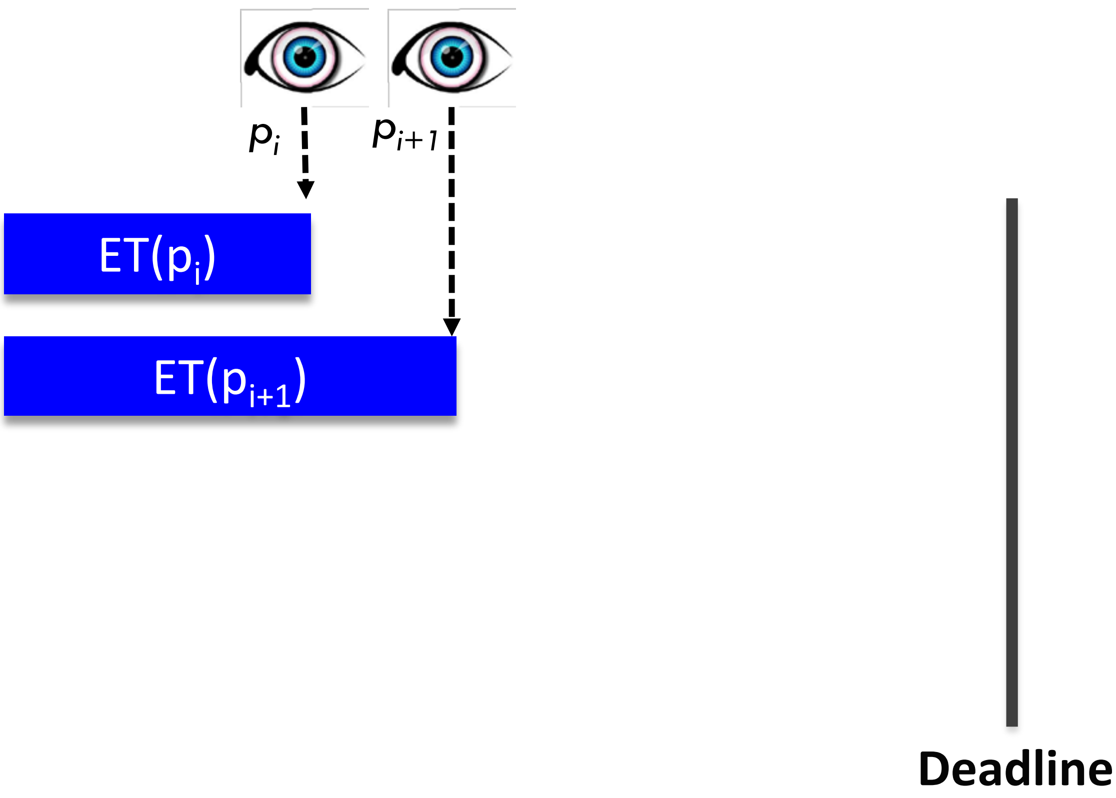
The critical task always respects its deadline, if:

1. $WCET_{iso} \leq D_C$
2. Switch: $$ET(x) + t_{RT} + W + RWCET(x) > D_C \quad (1)$$

Safety condition

The critical task always respects its deadline, if:

1. $WCET_{iso} \leq D_C$
2. Switch: $ET(x) + t_{RT} + W + RWCET(x) > D_C$ (1)

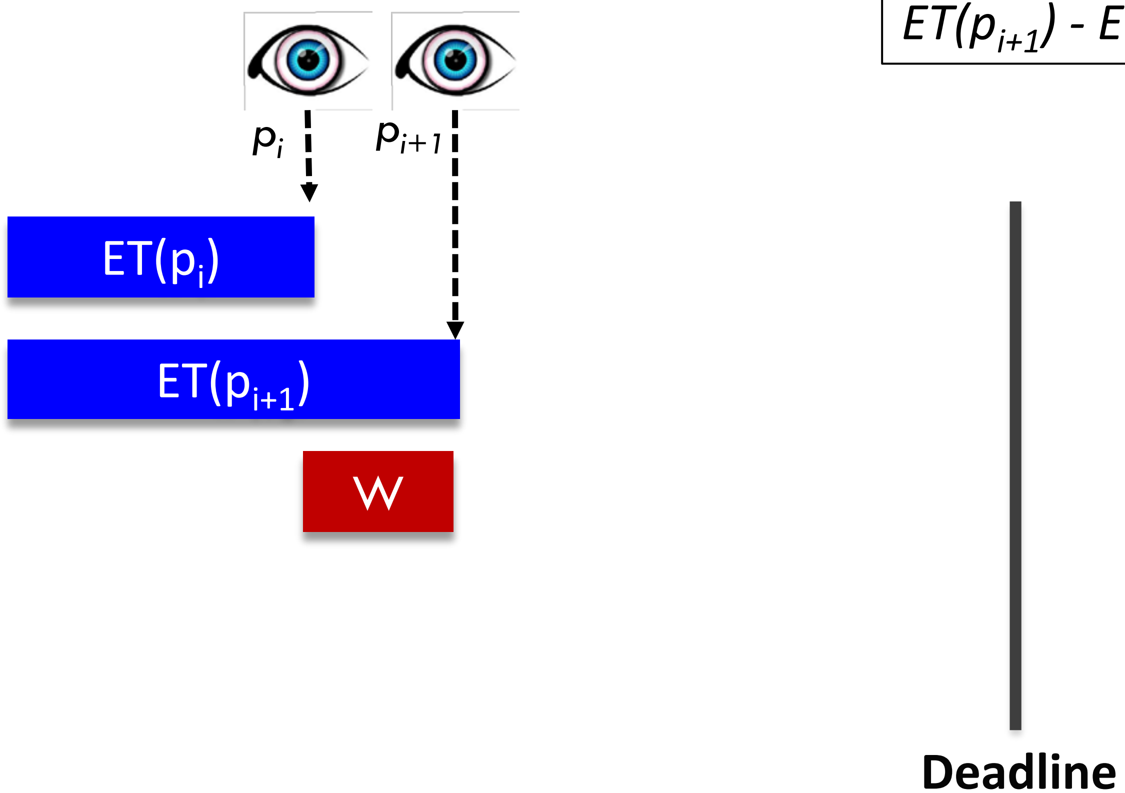


Safety condition

The critical task always respects its deadline, if:

1. $WCET_{iso} \leq D_C$
2. Switch: $ET(x) + t_{RT} + W + RWCET(x) > D_C$ (1)

$ET(p_{i+1}) - ET(p_i) \leq W$ (2)



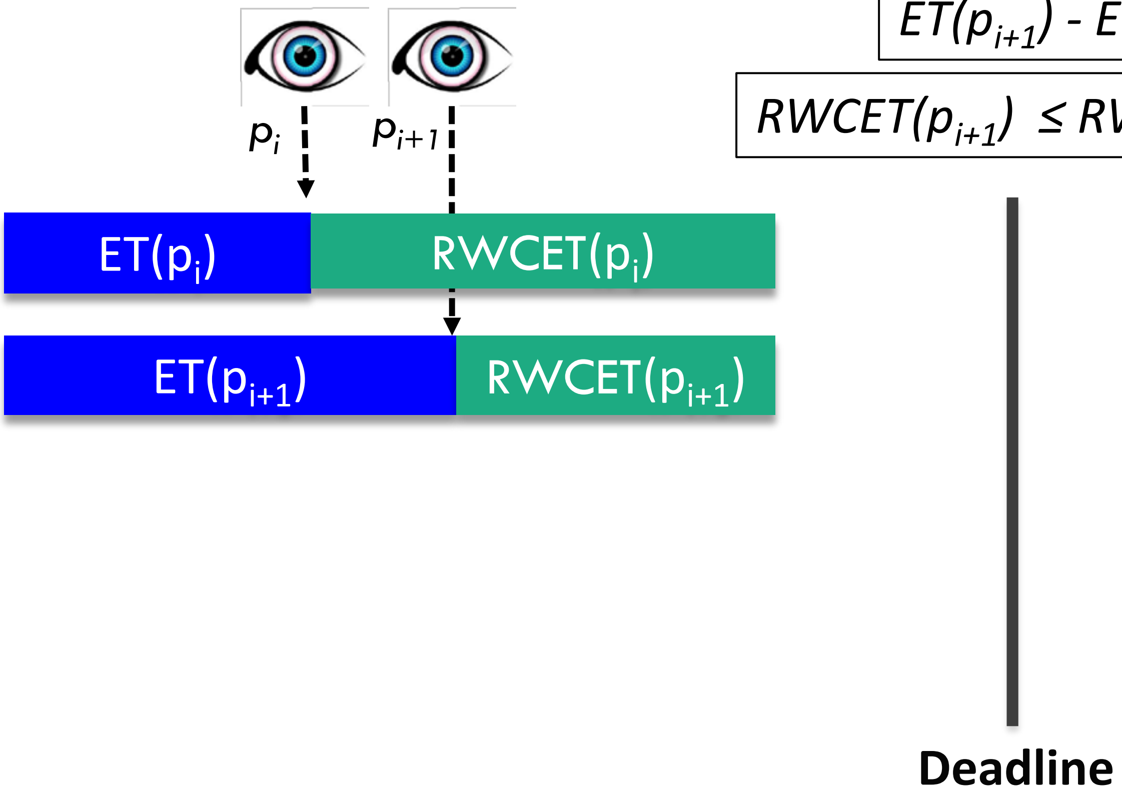
Safety condition

The critical task always respects its deadline, if:

1. $WCET_{iso} \leq D_C$
2. Switch: $ET(x) + t_{RT} + W + RWCET(x) > D_C$ (1)

$ET(p_{i+1}) - ET(p_i) \leq W$ (2)

$RWCET(p_{i+1}) \leq RWCET(p_i)$ (3)



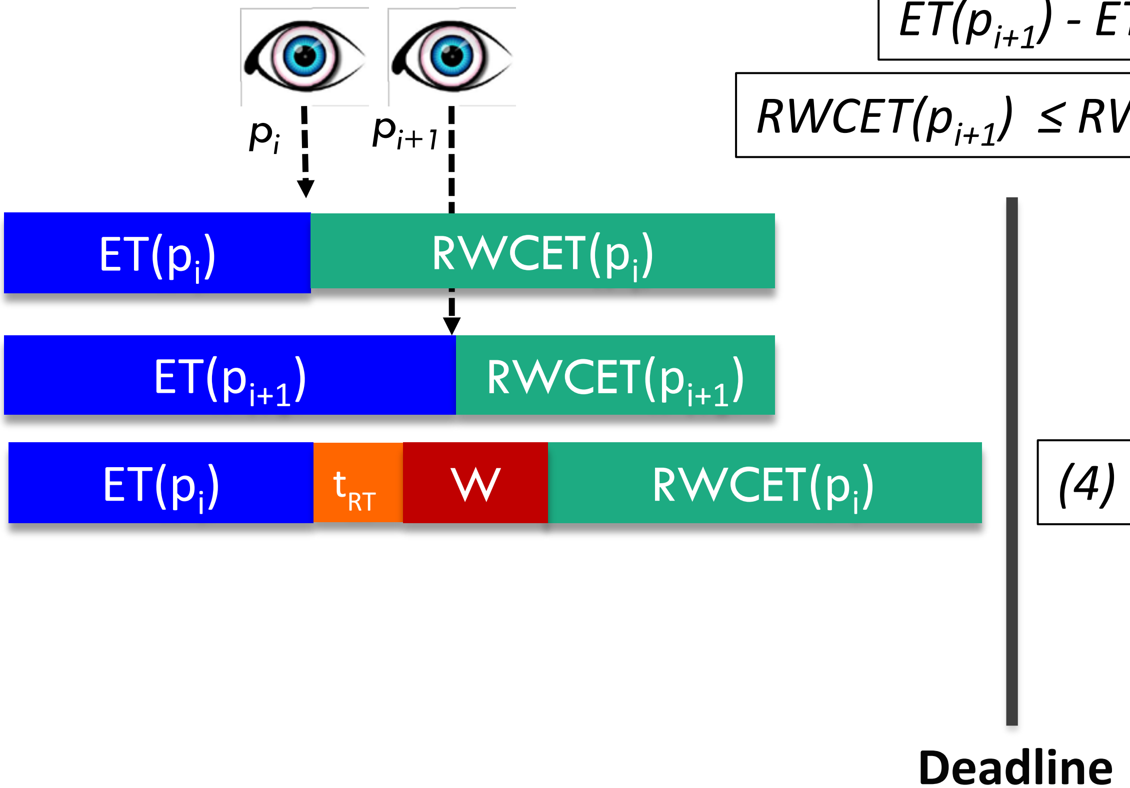
Safety condition

The critical task always respects its deadline, if:

1. $WCET_{iso} \leq D_C$
2. Switch: $ET(x) + t_{RT} + W + RWCET(x) > D_C$ (1)

$ET(p_{i+1}) - ET(p_i) \leq W$ (2)

$RWCET(p_{i+1}) \leq RWCET(p_i)$ (3)



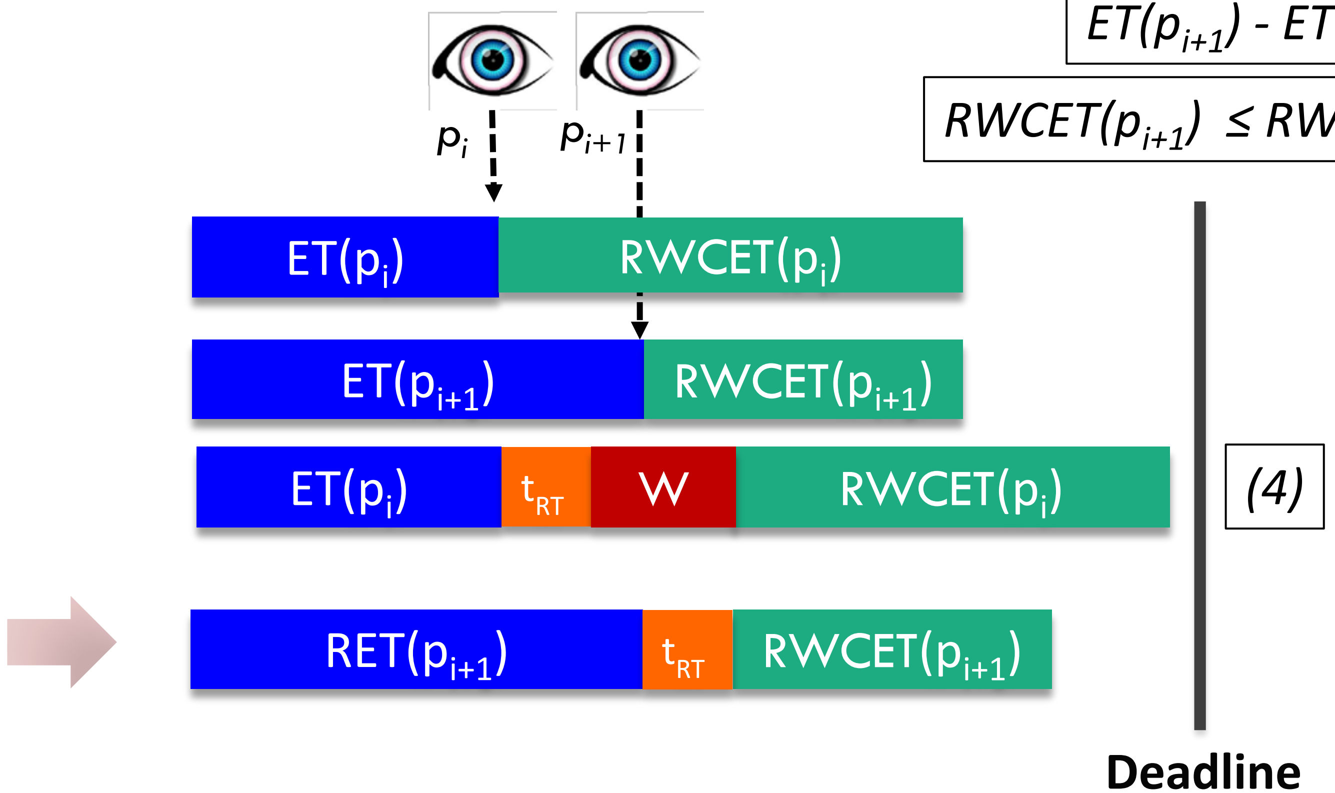
Safety condition

The critical task always respects its deadline, if:

1. $WCET_{iso} \leq D_C$
2. Switch: $ET(x) + t_{RT} + W + RWCET(x) > D_C$ (1)

$ET(p_{i+1}) - ET(p_i) \leq W$ (2)

$RWCET(p_{i+1}) \leq RWCET(p_i)$ (3)



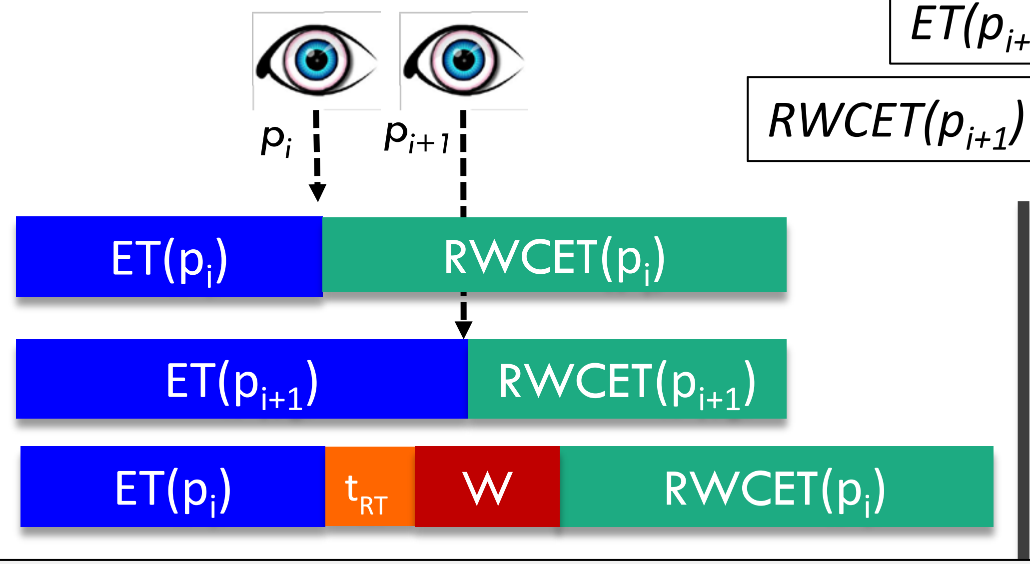
Safety condition

The critical task always respects its deadline, if:

1. $WCET_{iso} \leq D_C$
2. Switch: $ET(x) + t_{RT} + W + RWCET(x) > D_C$ (1)

$ET(p_{i+1}) - ET(p_i) \leq W$ (2)

$RWCET(p_{i+1}) \leq RWCET(p_i)$ (3)



(4)

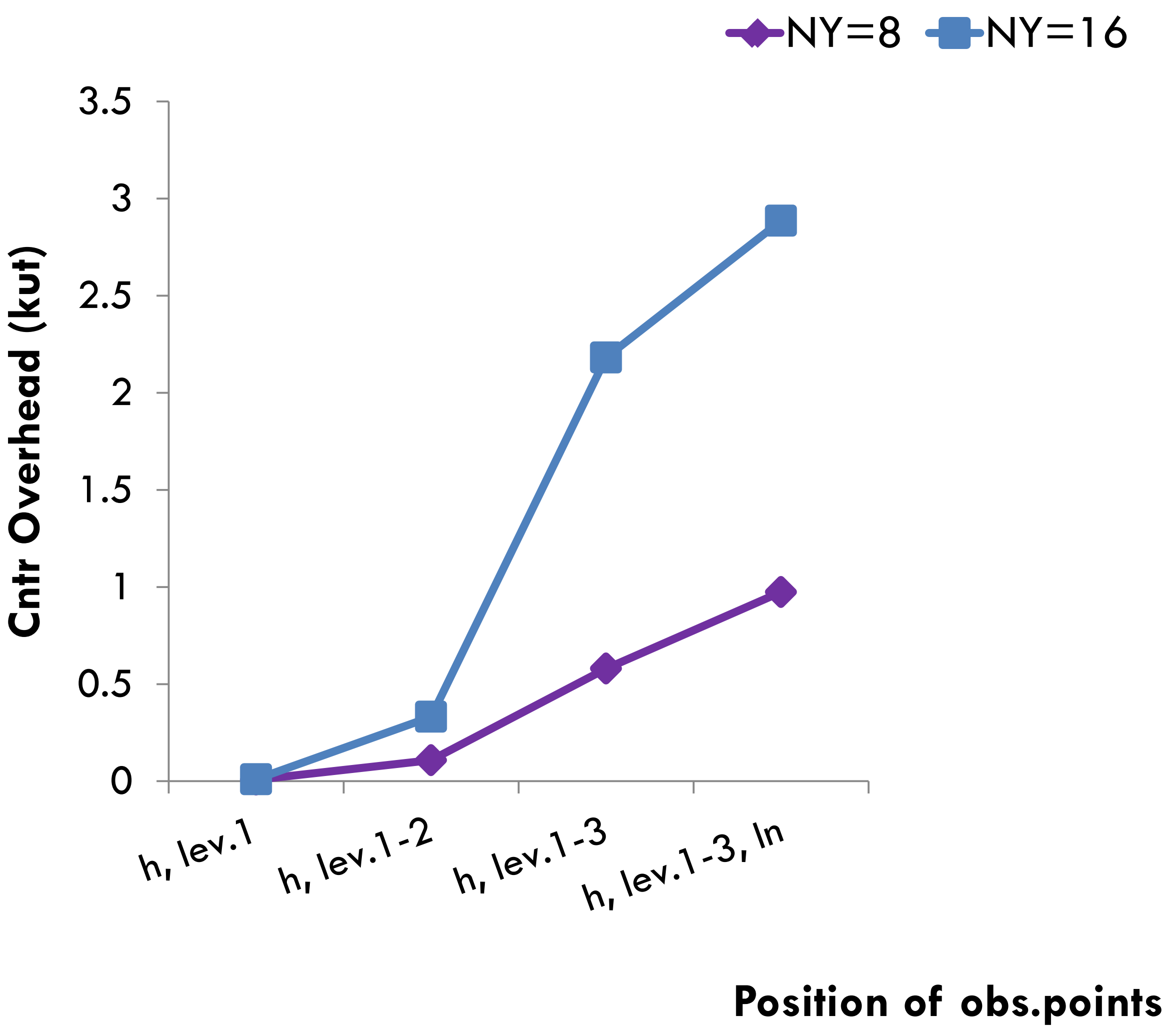
(2) → $ET(p_{i+1}) + t_{RT} + RWCET(p_{i+1}) \leq W + ET(p_i) + t_{RT} + RWCET(p_{i+1})$

(4) → $ET(p_{i+1}) + t_{RT} + RWCET(p_{i+1}) \leq D_C - RWCET(p_i) + RWCET(p_{i+1})$

(3) → $\mathbf{ET(p_{i+1}) + t_{RT} + RWCET(p_{i+1}) \leq D_C}$

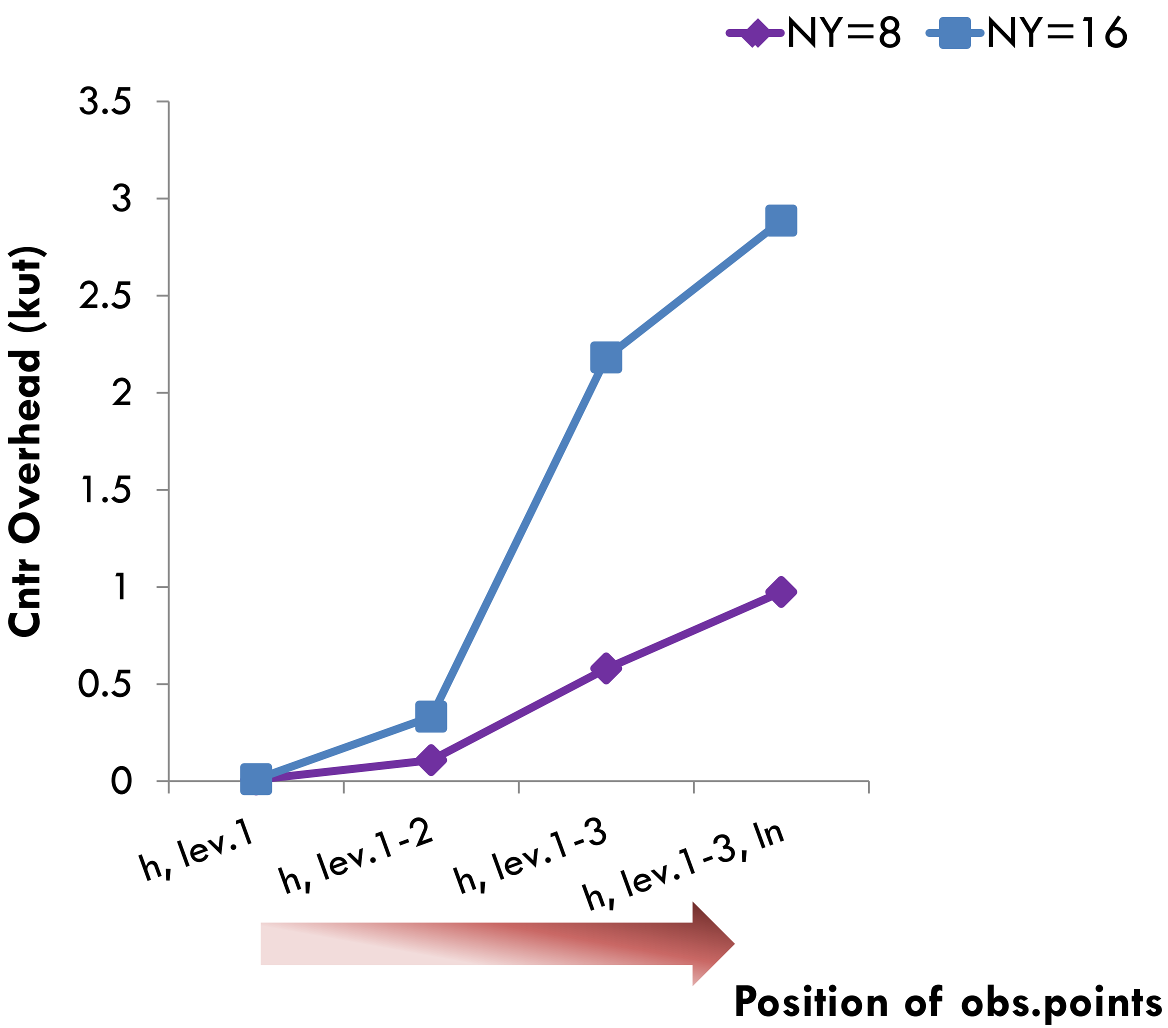
Simulation

Trmm



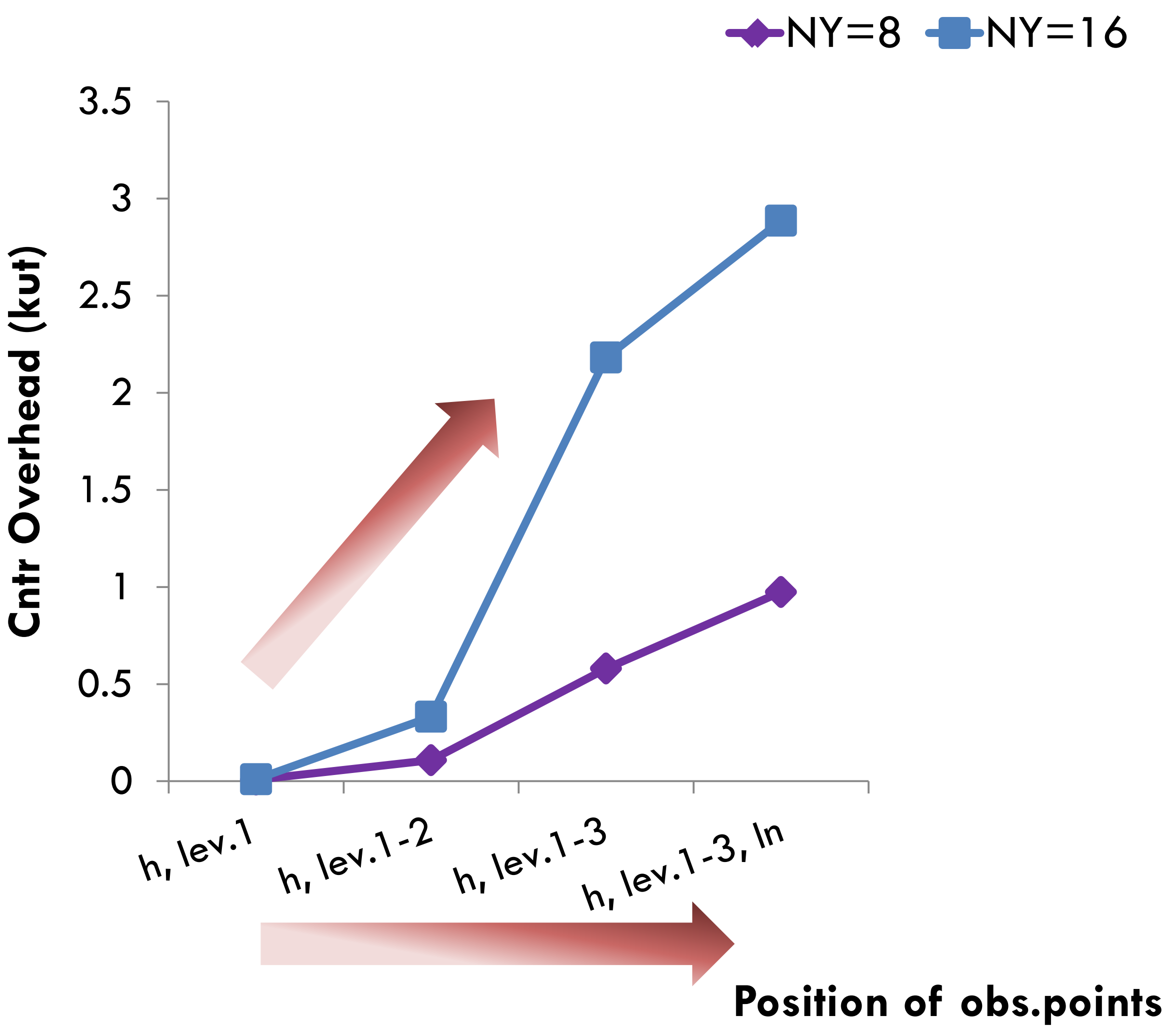
Simulation

Trmm



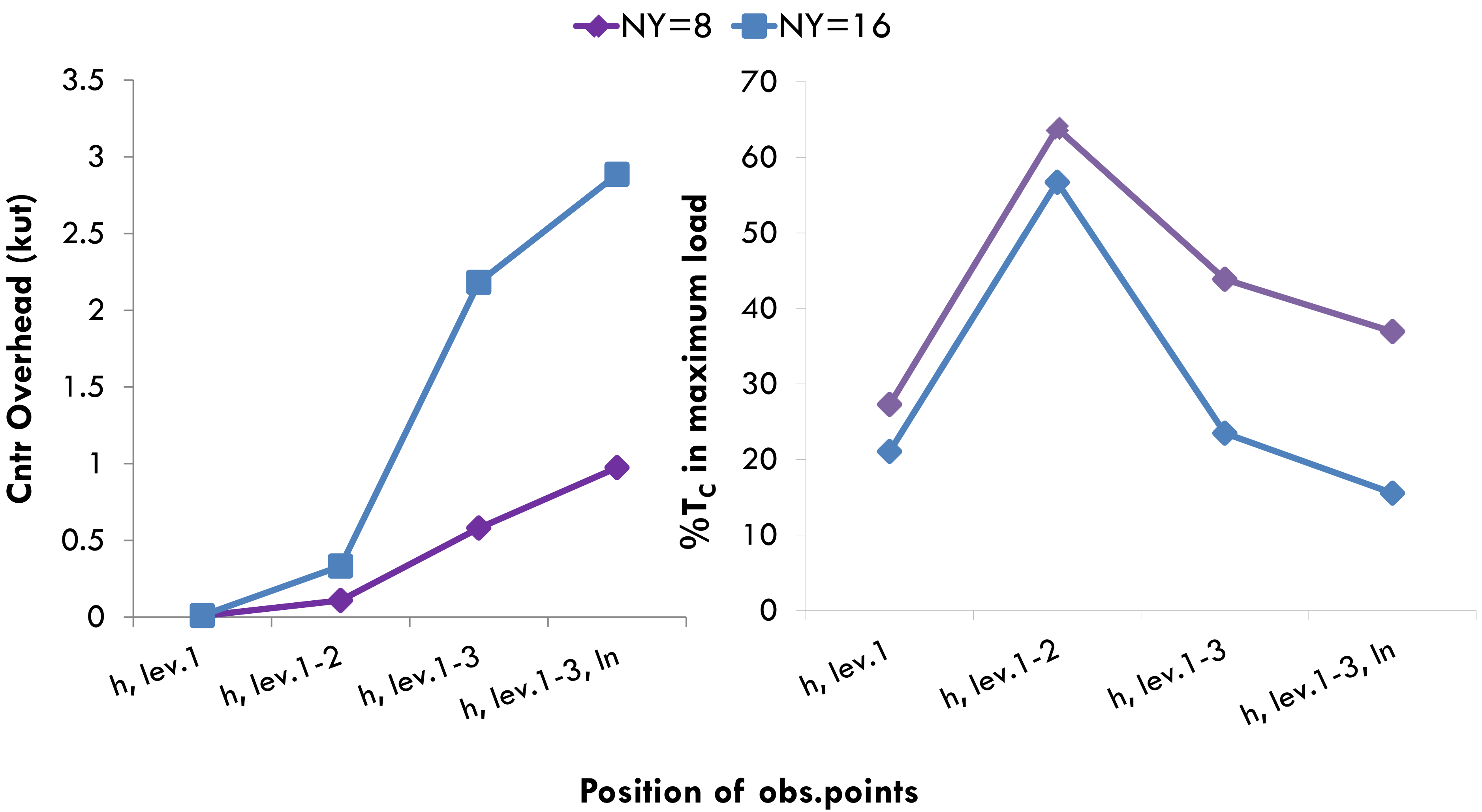
Simulation

Trmm



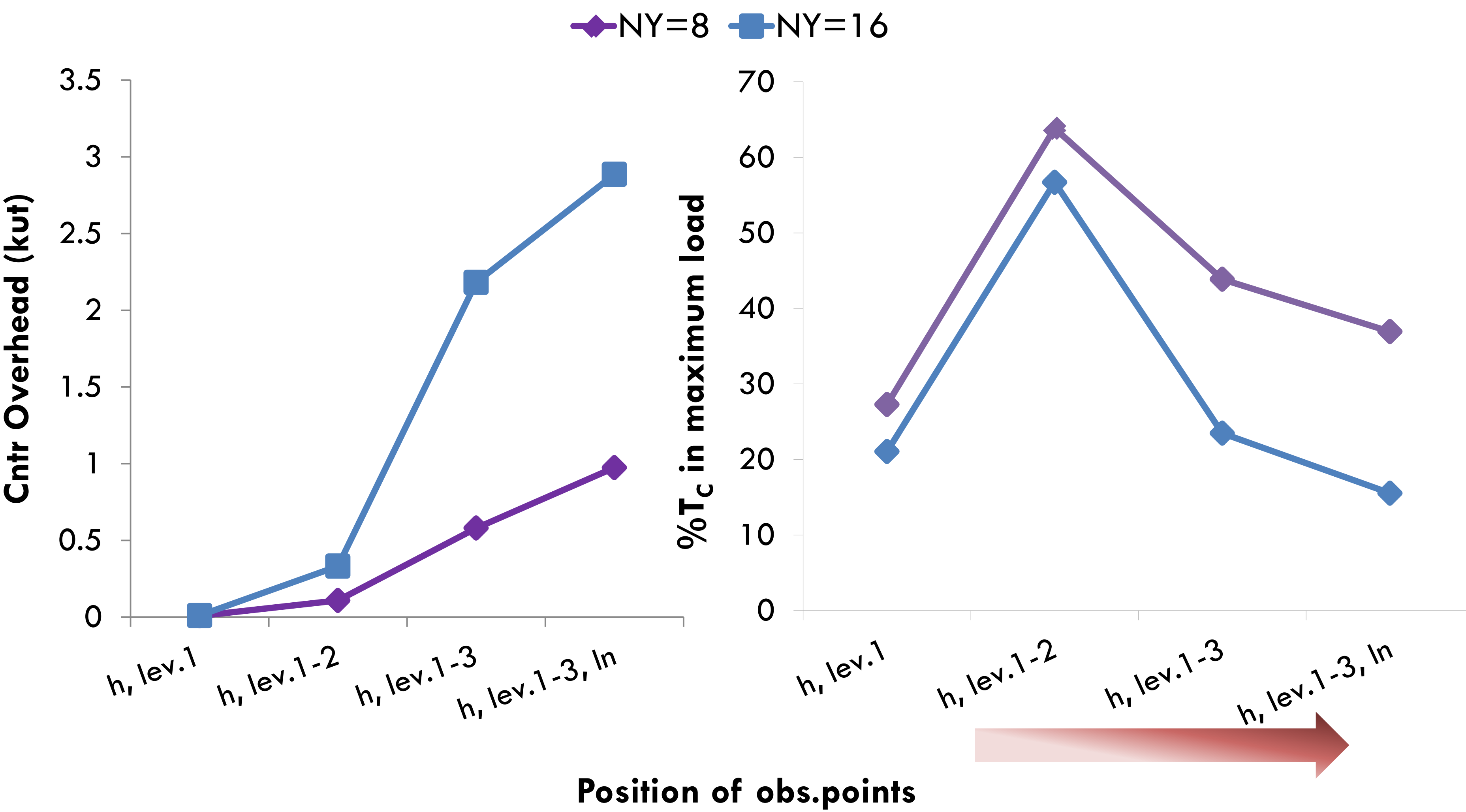
Simulation

Trmm



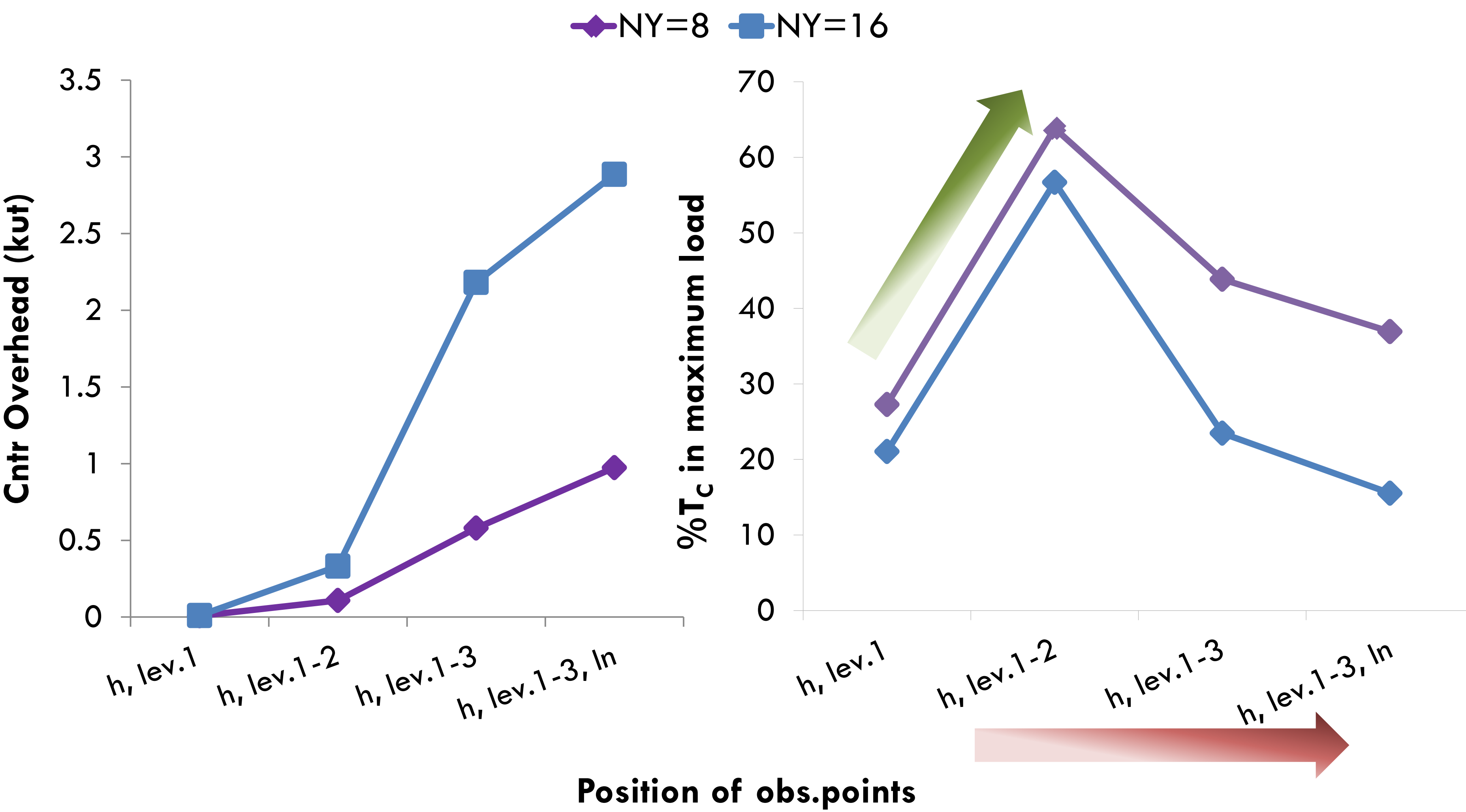
Simulation

Trmm



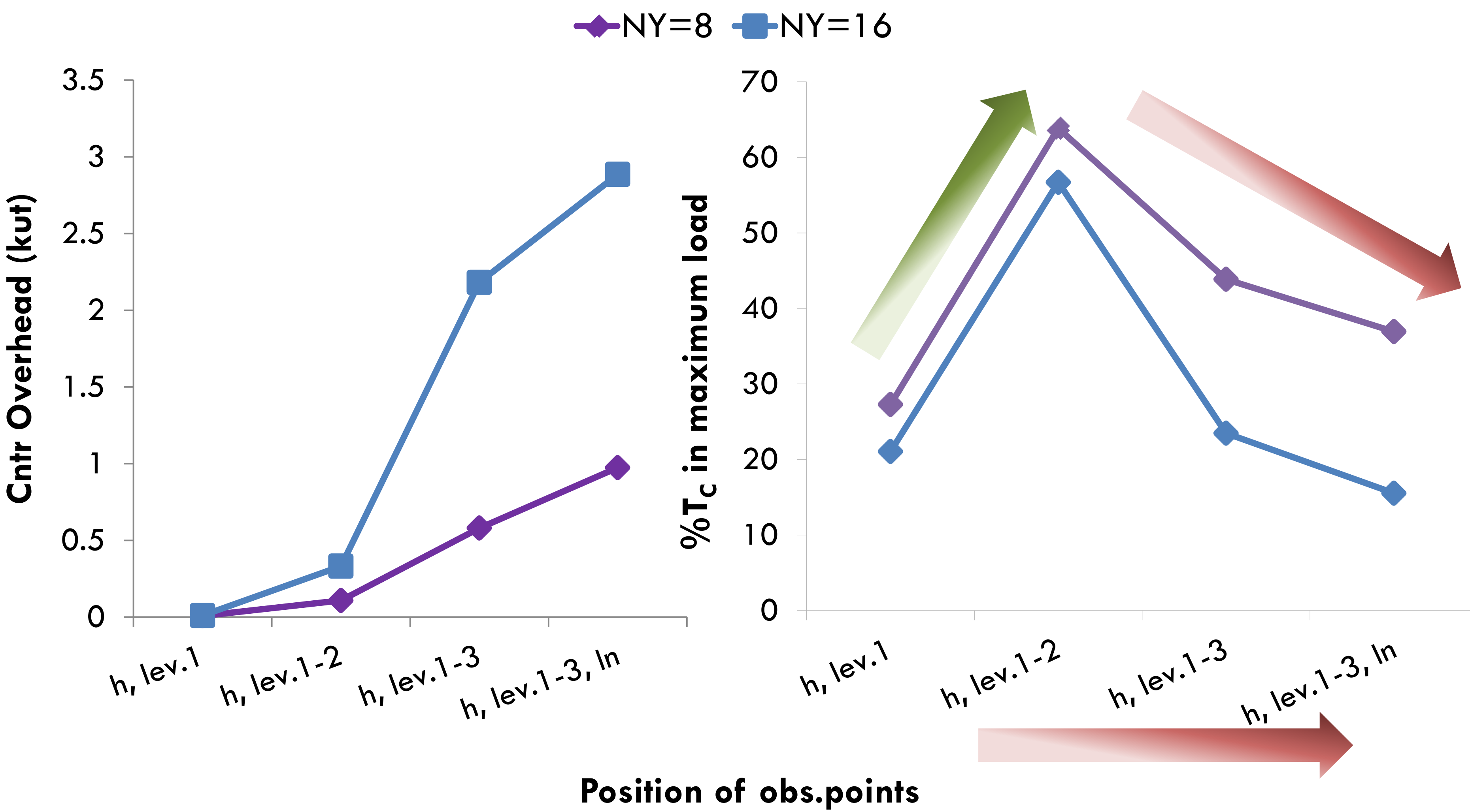
Simulation

Trmm



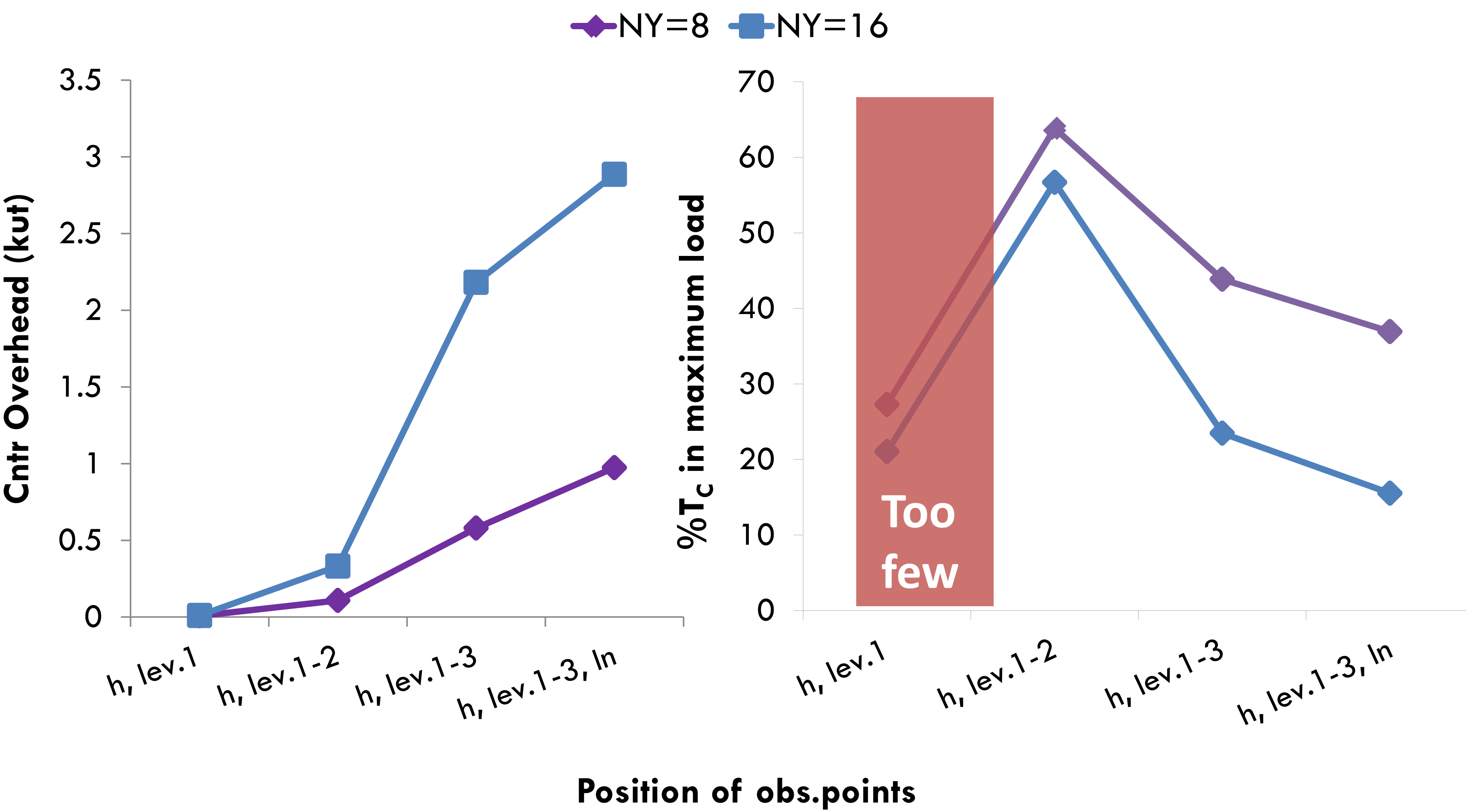
Simulation

Trmm



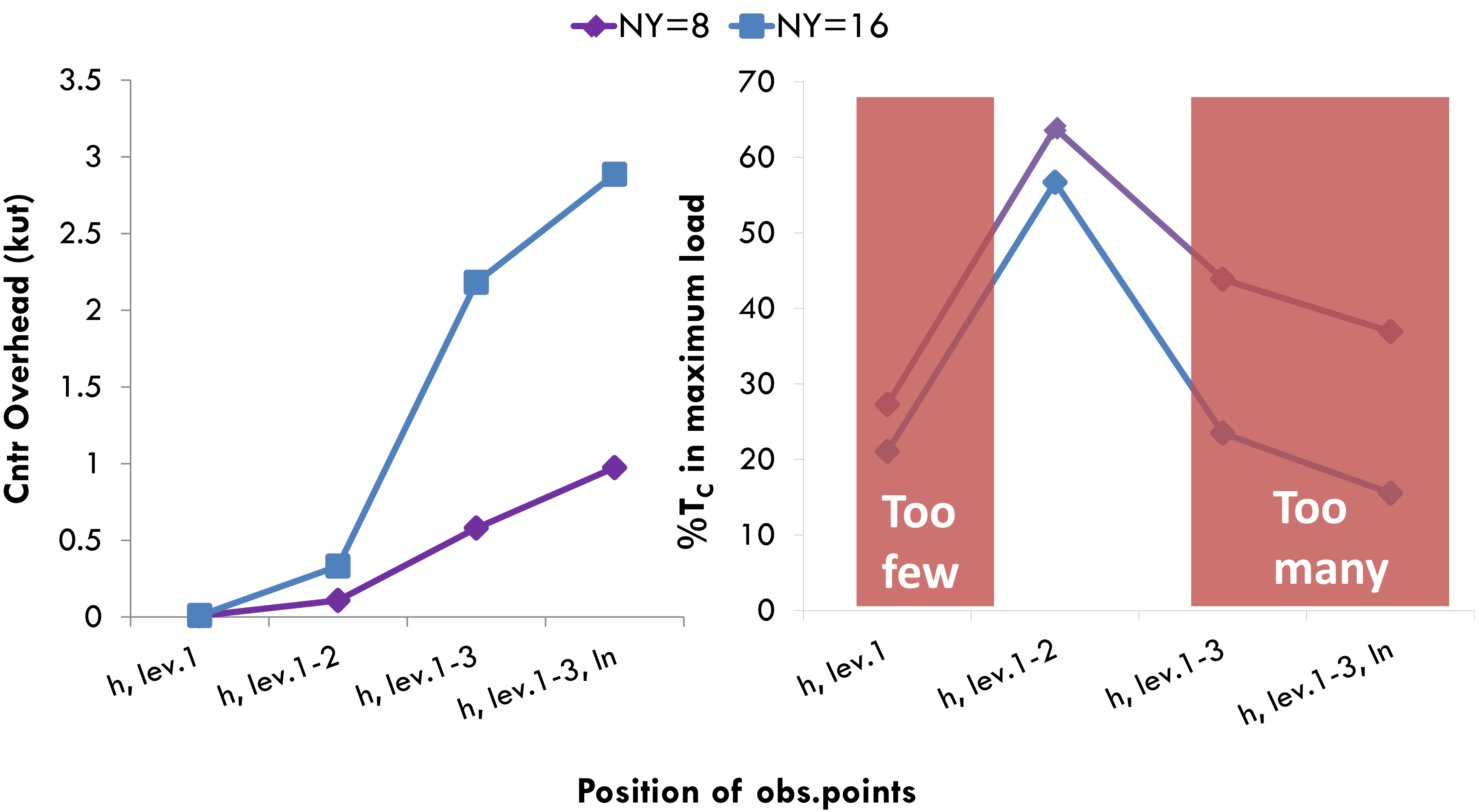
Simulation

Trmm



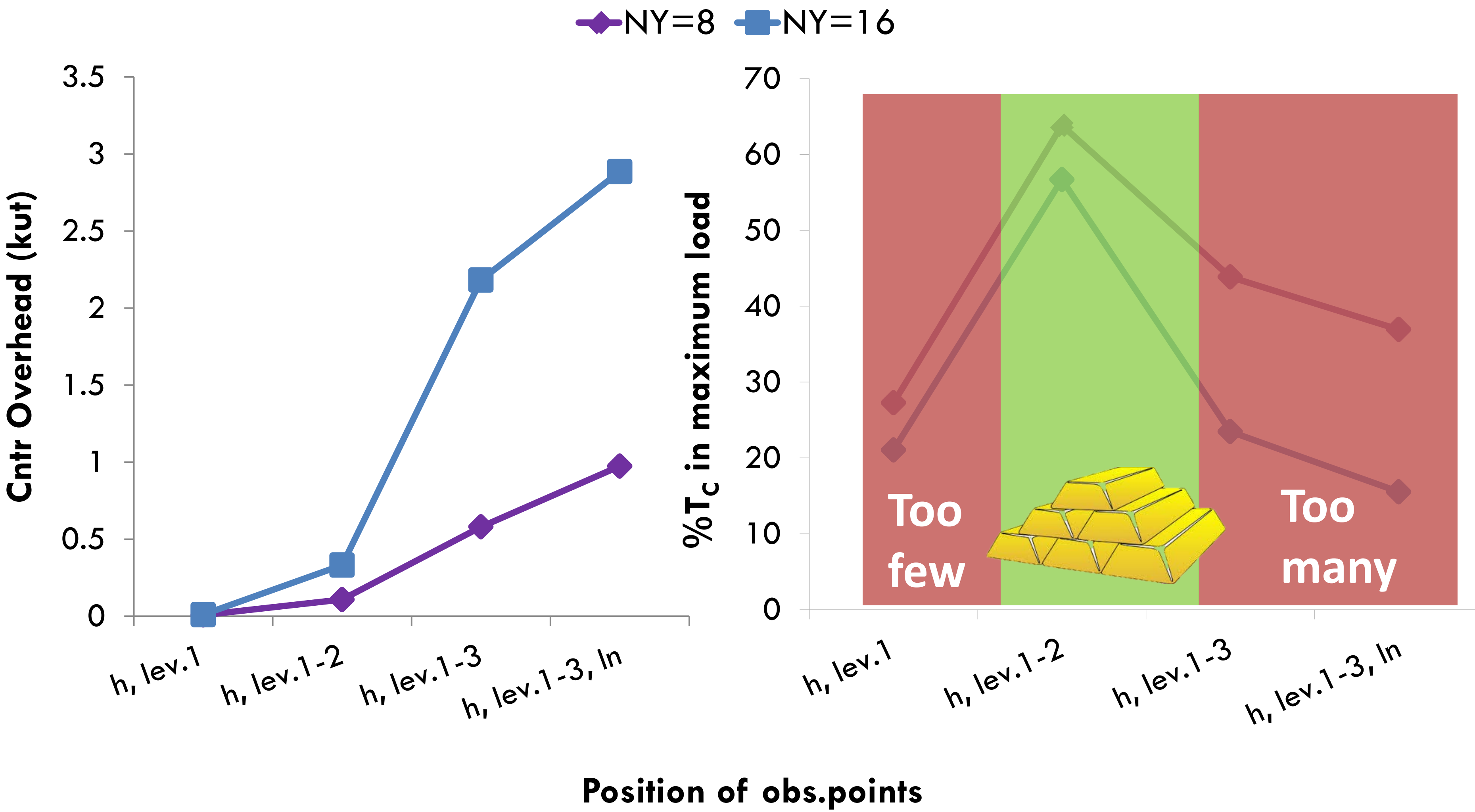
Simulation

Trmm



Simulation

Trmm



Conclusions & Future directions

- Conclusions:
  - Design-time: Pre-compute structure & timing information
  - Run-time: Use info to manage the executed tasks
  - Formally present and prove our approach
- Current work & Future directions:
  - Implementation to a real multicore platform (gain 4.5)
  - Extension to more than one critical tasks
  - Methodology for position & sampling of observation points
  - Extension to:
    - Several criticality layers
    - WCET with low & high assurance levels
    - Time & Space partitioning

THANK YOU

angeliki.kritikakou@onera.fr
www.kritikakou.com

QUESTIONS?