

# Composing RT Objects: A Case for Petri Nets and Girard's Linear Logic

**François Taïani**, Mario Paludetto (LAAS-CNRS, Toulouse, F),  
Jerôme Delatour (ESEO, Angers, F)



# Plan

- Introduction: Time, Objects and Composition
- Objects and Petri Nets. A good Match?
- Using a Mutant Logic to the Rescue
- Example
- Conclusion and Outlook

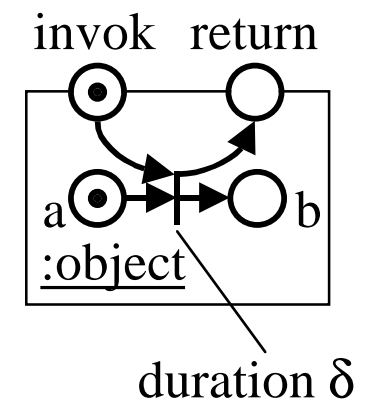
# Time, Objects and Composition

- Object = “Divide and Conquer” Unit  
(cross-project: reuse, cross-company: COTS)
- Integration Pitfall  $\Rightarrow$  Contract Based Development
- Real Time OO Systems  $\Rightarrow$  Time Sensitive Contracts
- (Time Sensitive) Reuse Question:  
*“Does that Object match my (Time Sensitive) Needs?”*

Checking of Temporal Interoperability  
Object / Environment needed.

# Objects and Petri Nets

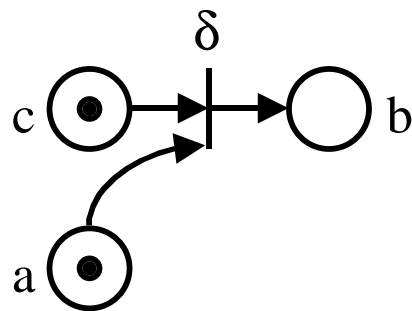
- Petri Net = Good for Control, Bad for Data
- Petri Net in Object:
  - Abstraction of Actual Behavior
  - Orders Object Needs / Services naturally  
(Exclusion, Precedence, ...) (*Logical Time*)
  - Timed Transitions (*Quantitative Time*)
  - Token flows  $\approx$  Threads



# Using a Mutant Logic to the Rescue (I)

“... in which truths get lost once they are used.”

[Girard90]



“A at time  $\alpha$  with C at time  $\gamma$ ”

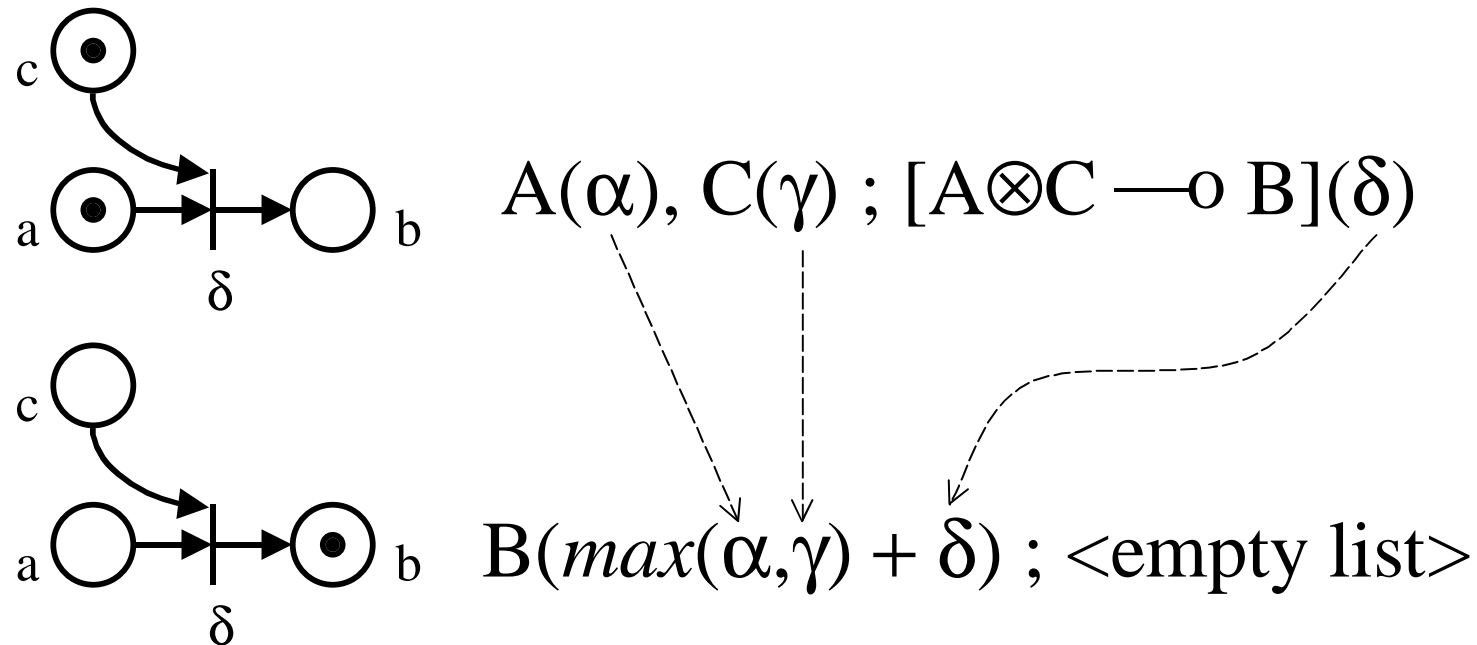
$A(\alpha), C(\gamma)$

“ (A and C) produces B, in duration  $\delta$ ”

$[A \otimes C \multimap B](\delta)$

# Using a Mutant Logic to the Rescue (II)

- A Rewriting Step:

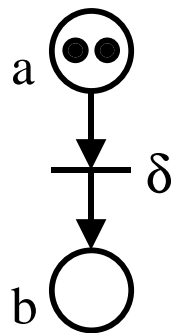


# Using a Mutant Logic to the Rescue (III)

- Girard's Linear Logic  $\approx$   
PN Unfolding with  $(max, +)$  Analysis
- Decisive Features :
  - Symbolic Duration Times
  - Multi-Instantiation of the same Structure ( $\approx$ Threads)
  - (Finite) Loops
  - No State Explosion due to Concurrency

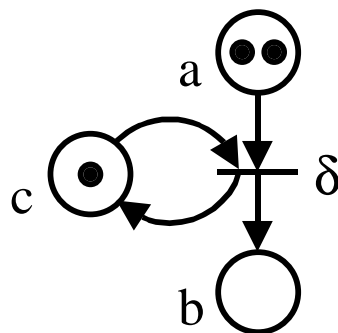
# Using a Mutant Logic to the Rescue (IV)

- Multi-Instantiation of the same Structure



$$\begin{aligned}
 & \mathbf{A}(0) , \mathbf{A}(0) ; 2 \times [\mathbf{A} \multimap \mathbf{B}](\delta) \\
 \rightsquigarrow & \mathbf{B}(\delta) , \mathbf{A}(0) ; [\mathbf{A} \multimap \mathbf{B}](\delta) \\
 \rightsquigarrow & \mathbf{B}(\delta) , \mathbf{B}(\delta) ; \cdot
 \end{aligned}$$

$$\Rightarrow \mathbf{WCET} = \delta$$



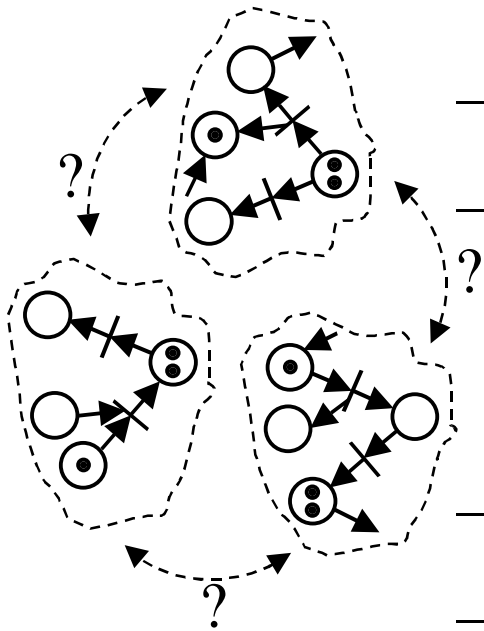
$$\begin{aligned}
 & \mathbf{A}(0) , \mathbf{A}(0) , \mathbf{C}(0) ; 2 \times [\mathbf{A} \otimes \mathbf{C} \multimap \mathbf{B} \otimes \mathbf{C}](\delta) \\
 \rightsquigarrow & \mathbf{A}(0) , \mathbf{C}(\delta) , \mathbf{B}(\delta) ; [\mathbf{A} \otimes \mathbf{C} \multimap \mathbf{B} \otimes \mathbf{C}](\delta) \\
 \rightsquigarrow & \mathbf{B}(2\delta) , \mathbf{C}(2\delta) , \mathbf{B}(\delta) ; \cdot
 \end{aligned}$$

$$\Rightarrow \mathbf{WCET} = 2\delta$$



# Using a Mutant Logic to the Rescue (v)

- No State Explosion due to Concurrency



— Collection of Partial States

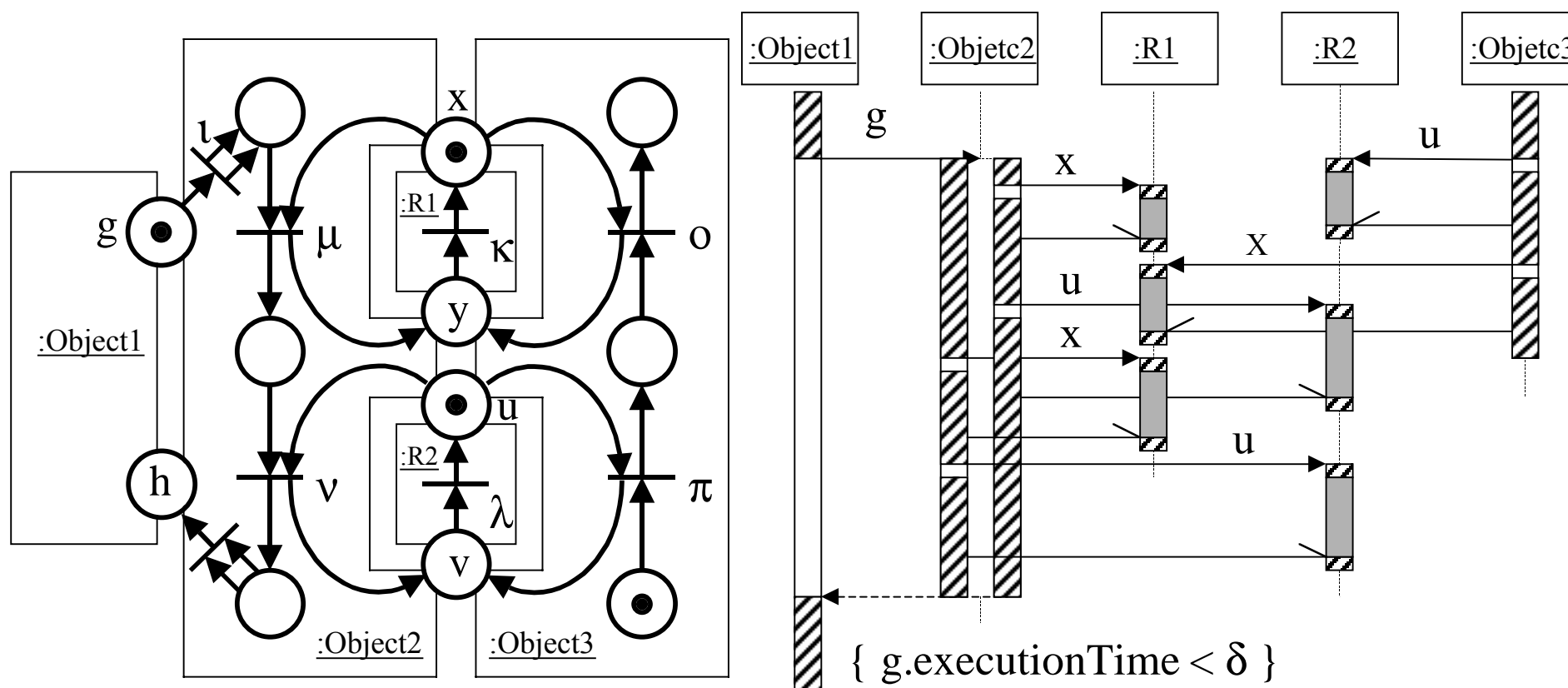
— No Assumptions about possible “Simultaneity”  
... until explicit Synchronization needed



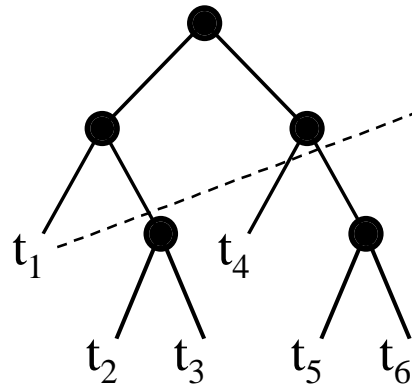
— Plain Application of Rules yields Correct Result

— Case Distinction only at “Conflict Points”

# Example: Problem



# Example: Results



$$t_1 = v + \max \left( 2\mu + \kappa + \max(\iota, \pi + \omicron + \kappa), \right. \\ \left. v + \lambda + \max(\mu + \max(\iota, \pi + \omicron + \kappa), \pi + \lambda) \right)$$

 $\Rightarrow$ 

{ g.executionTime <  $\delta$  } fulfilled as soon as

$$\text{WCET}_{\text{model}} = \max(t_1, t_2, t_3, t_4, t_5, t_6) < \delta$$

● = Conflict (i.e. Choice)  
Point

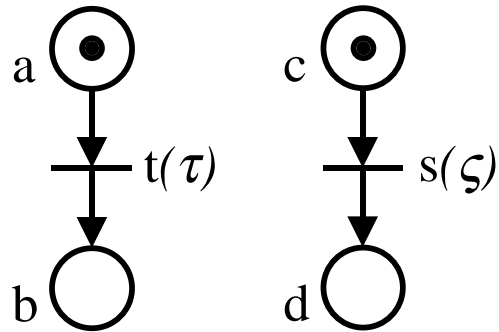
| = Plain Rewriting Sequence :

something ; something  
 $\rightsquigarrow$  something ; something  
 $\rightsquigarrow$  something ; something  
 $\rightsquigarrow$  something ; something

# Conclusion and Outlook

- Temporal Constraint at one Object Interface  
→ Temporal Requirements on whole System
- Further Developments:
  - Multiple Constraints? Choice of Context?
  - Scalability Study on Larger Examples
  - Tool Development (started)
  - Connection to Concrete Distributed OO Frameworks.

# Annex: Partial Orders



$$=_{\text{def}} [A \multimap B](\tau)$$

$$=_{\text{def}} [C \multimap D](\zeta)$$

- Interleaving:  $t ; s$  and  $s ; t$

- Girard's LL:

If no Conflicts:

→ All rewriting Sequences are equivalent.

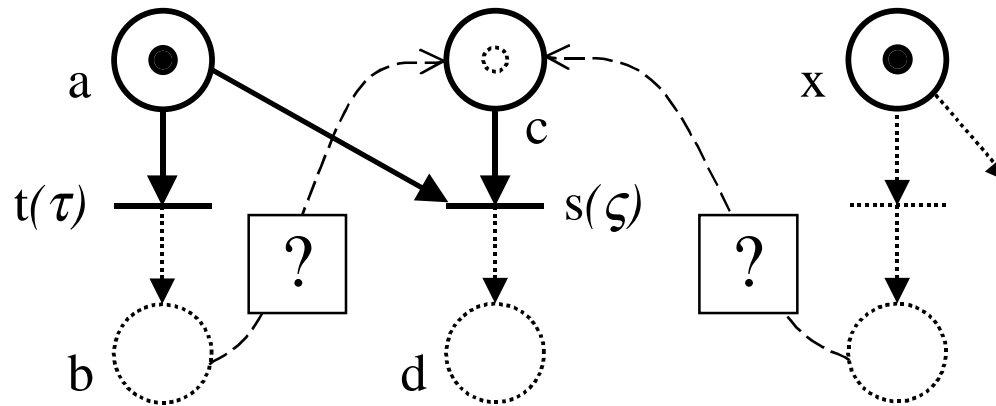
→ Only one does it all :

$$A(\alpha) \quad , \quad C(\gamma) \quad ; \quad T, S$$

$$\leadsto B(\alpha+\tau) \quad , \quad C(\gamma) \quad ; \quad S$$

$$\leadsto B(\alpha+\tau) \quad , \quad D(\gamma+\zeta) \quad ; \quad .$$

# Annex: Potential Conflict Point



Question 1: Fire **t** ?, or wait for possible **C** ?

Question 2: Can a **C** be produced without **t** being fired?

→ Solution: Fire **t**, set flag to study causality  $t \rightarrow C$ ,  
and identify actual conflict afterwards, if any.