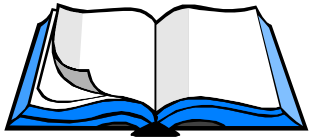
Taking semantics into account for Modeling Real-Time Applications

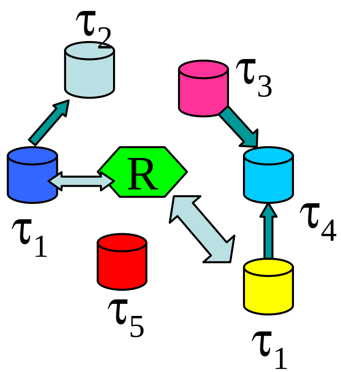
Christian Fotsing[1], Annie Geniet[1,2], Guy Vidal-Naquet[3,4]

1: LISI, ENSMA, France
2: Université de Poitiers, France
3: SUPELEC, Campus de Gif-Sur-Yvette, France
4: Université Paris-Sud, Orsay, France

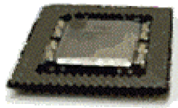
The Scheduling Problem



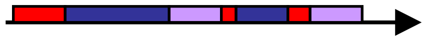
Requirements



A set of tasks (periodic, sporadic..)



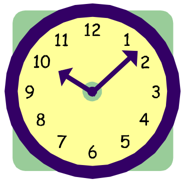
A Schedule

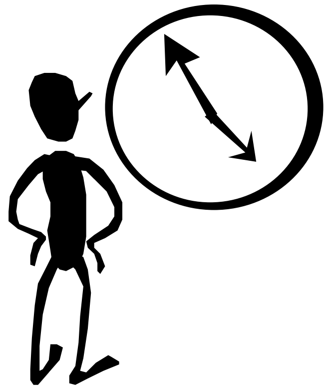


Specifications

functionnal

Temporal





Timing Constraints

General Context

- ✓ Critical real-time application

- ✓ Pre-emptive interacting periodic tasks with conditional statements

- ✓ Pre-runtime scheduling on pre-emptive uniprocessor systems

Structure of a Real-Time Task

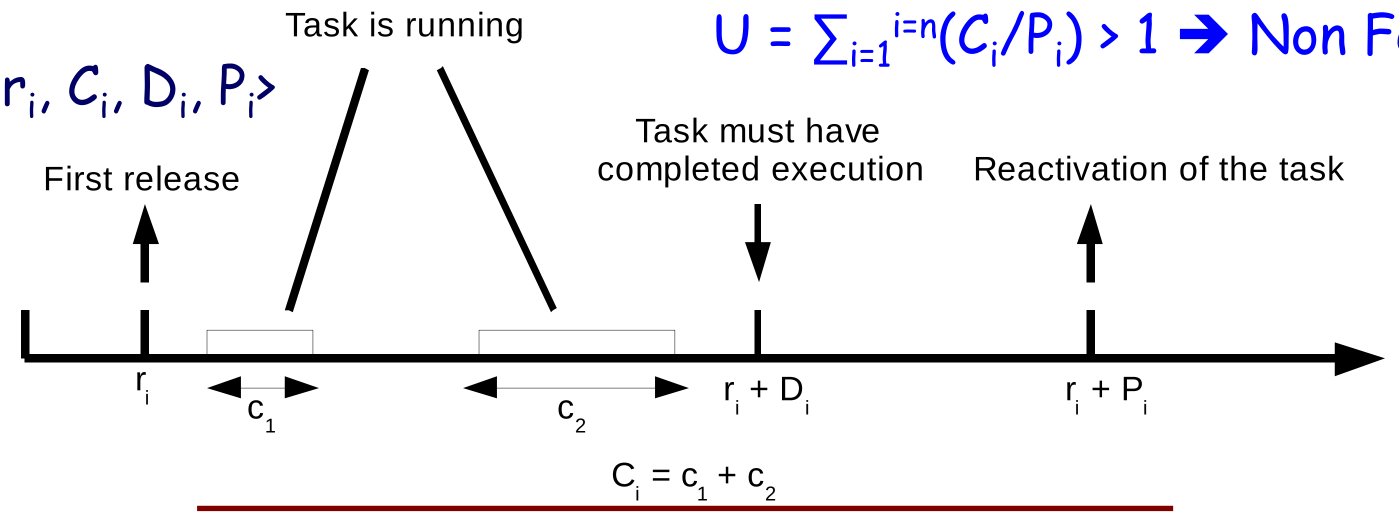
```
Task T
        Input x: integer;
        block(2); -- a functionnal block written in a high-level langage
        Send(message1); -- a real-time primitive
        If (x>3) [duration of test = 1]
        then
                block(1);
        else
                Lock(resource1); -- a real-time primitive
                block(3) ;
                Unlock(resource1); -- a real-time primitive
        endif;
        block(3);
        Receive(message2); -- a real-time primitive
        block(1);
end;
```

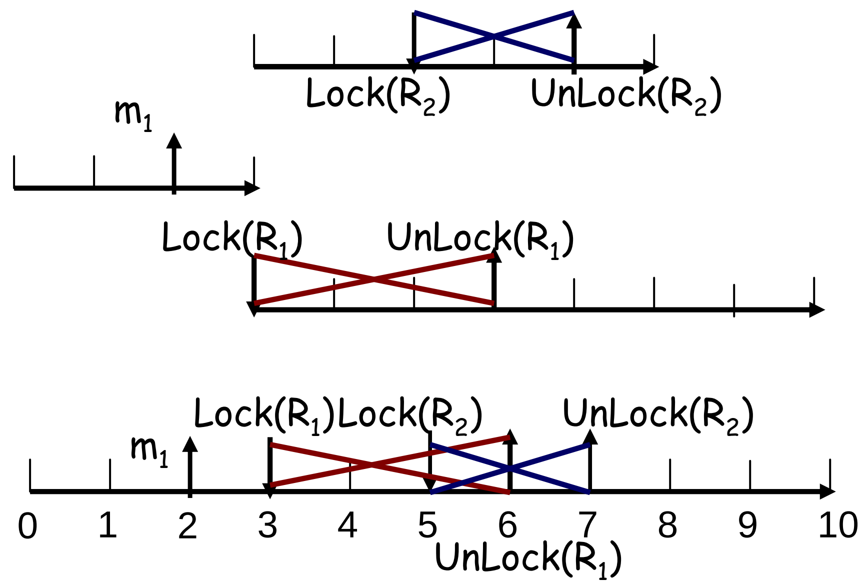
Classical Model versus Tree Model

Liu1973, Babau1996, Niehaus1991, Buttazo1997, Wilhelm and Al2008

$U = \sum_{i=1}^{i=n}(C_i/P_i) > 1$ ➔ Non Feasible

Task T_i <r_i, C_i, D_i, P_i>





Tree view of a Task

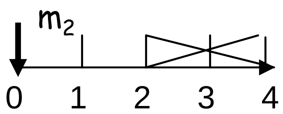
Classical view of a Task

- The classical model is
  - ✓ Overconstrained
  - ✓ Oversized
  - ✓ Uneffective

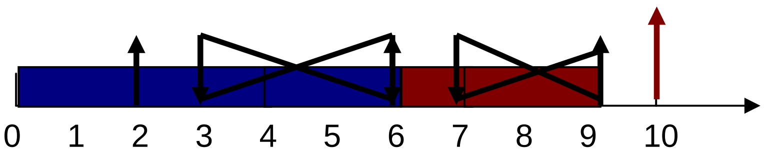
Objectives

Linear (Classical) approach

A task

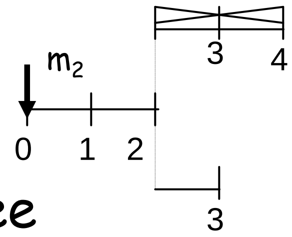


A schedule

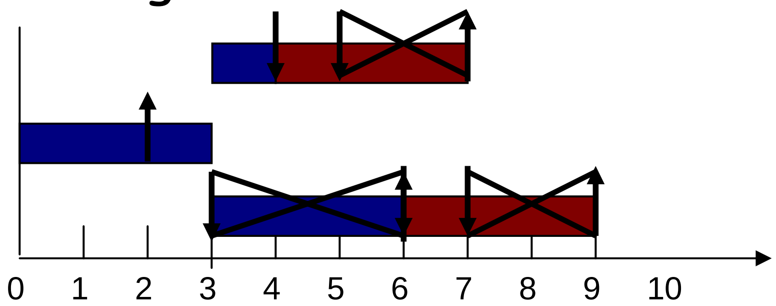


Tree-based (Our) approach

A task



A scheduling tree



+

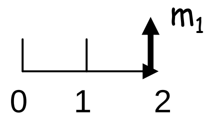
Automatic generation

Motivations of this Work (1/2)

Fotsing and Al2009, Fotsing and Al2010

✓ Task $T_{AcquiredOrder}$

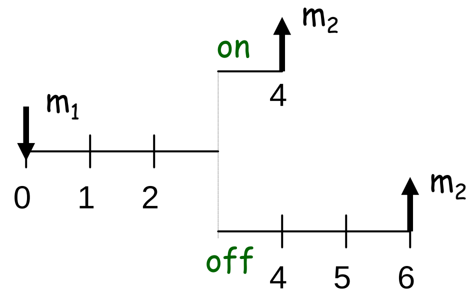
$T_{AcquiredOrder}\langle 0,2,11,11\rangle$

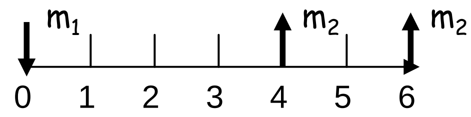


→ No conditional instructions

✓ Task $T_{WiperController}$

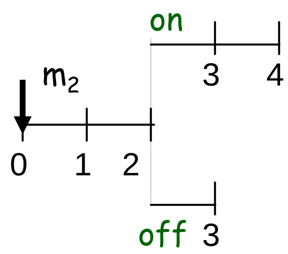
$T_{WiperController}\langle 0,(4,6),11,11\rangle$

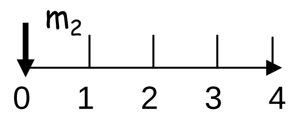




✓ Task T_{Order}

$T_{Order}\langle 0,(4,3),11,11\rangle$





Motivations of this Work (2/2)

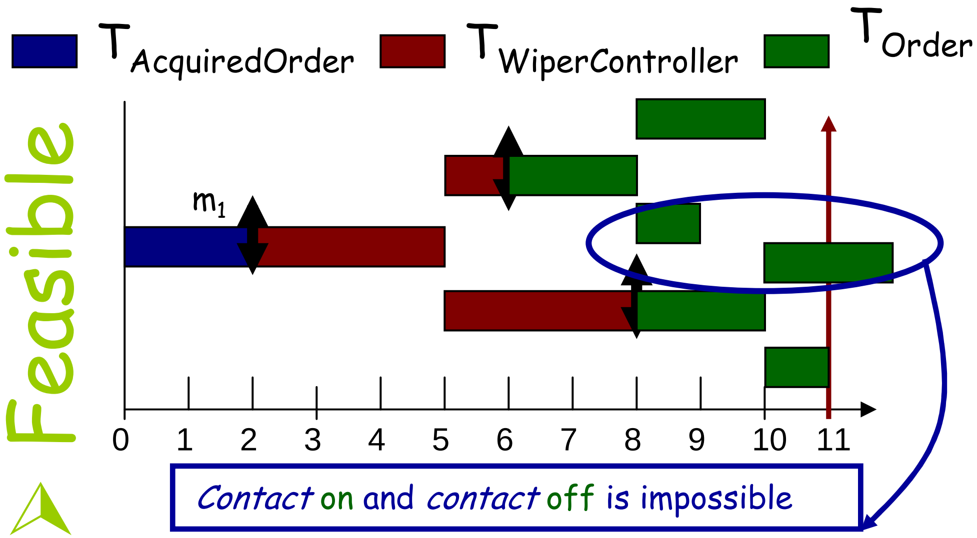
Madhukar and Al2008, Fotsing and Al2009, Fotsing and Al2010

Classical analysis

- ✓ $U = 2/11 + 6/11 + 4/11 = 12/11 > 1$
- ✓ Not valid schedule
  - ➢ Non Feasible

Our analysis

➢ Feasible



Scheduling tree

Linear Unfeasibility Does not Imply Tree-based Unfeasibility

Linear Feasibility Implies Tree-based Feasibility Is Equivalent to Effective Feasibility

Existing Approaches

Aussaguès and Al1998, Baruah1998, Baruah2003

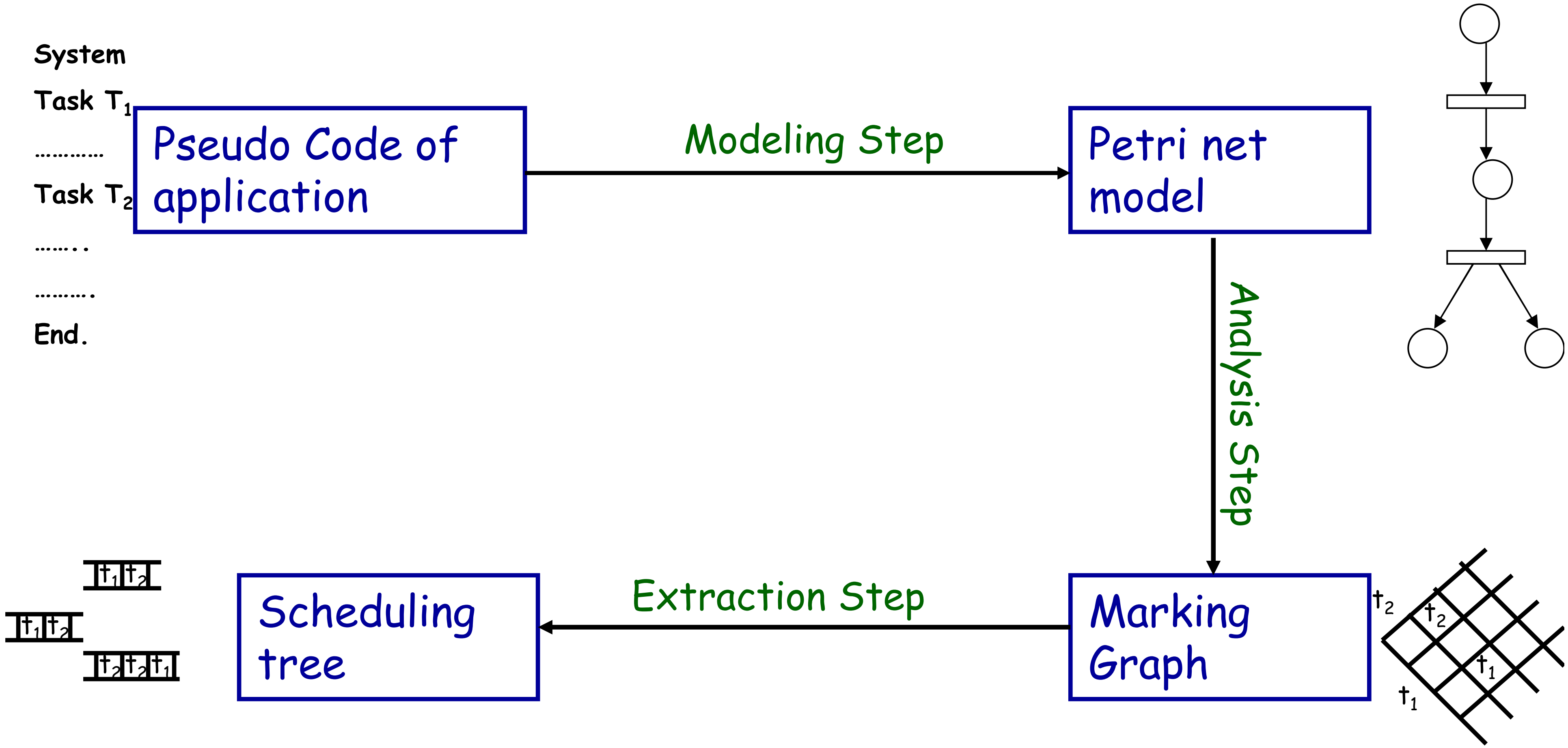
- Aussaguès And David's approach
  - ✓ State-transition diagram

✓ Propose a feasibility study based on ILP technique

- Baruah's approach
  - ✓ Subdivise tasks in sub-tasks

✓ Propose a feasibility study based on DBF function

Our General Methodology



Motivations for Our Choices

- Off line approaches
  - ✓ Not optimal on line scheduling
    - ✓ Describe behavior of applications
- ✓ Scheduling Power
- Petri net based model expressing
  - ✓ Exchange messages
  - ✓ Share resources
  - ✓ Parallelism
  - ✓ Concatenation
- ➢ Automatic Generation of Scheduling Tree

Use of Petri Net

- Some notions of Petri nets
- The Basic Model of Choquet-Geniet and Grolleau
- The Integration of Conditional Tasks
- The Incompatibility Relations between Tests
- The Semantic Layer
- An Illustration of our Model
- Conclusion

Petri Net Model

Choquet-Geniet2006

- ✓ Autonomous Petri nets
- ✓ Colored Petri nets
- ✓ Terminal Marking
- ✓ Maximum firing rule

- M_0: Initial Marking
- Q: Finite Set of places
- T: Finite Set of transitions
- W: QxT U TxQ → N^+, the valuation fonction

Basic Model

Choquet-Geniet and Grolleau2002

Clock System

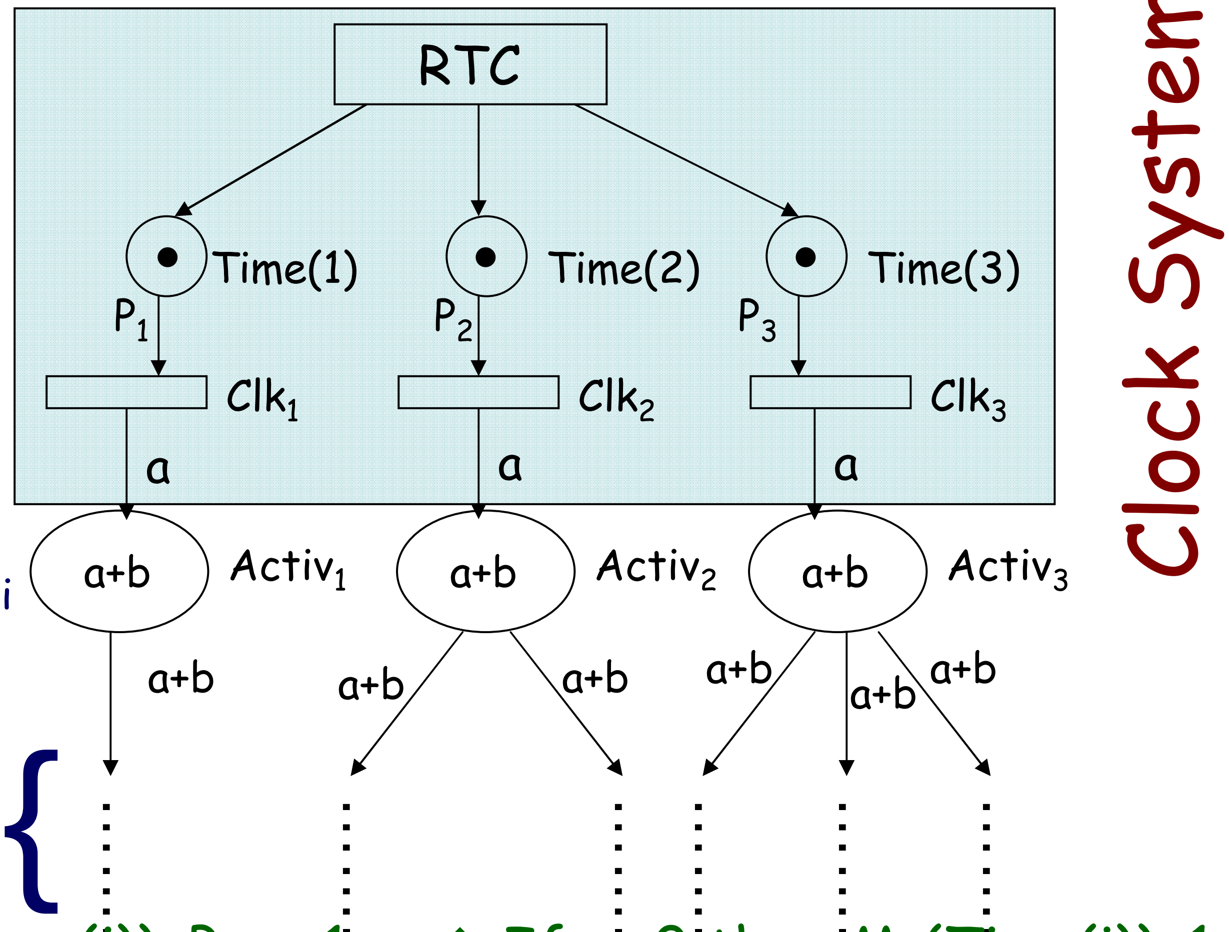
(To insure dynamic of application)

Task System

(To manage tasks and their interactions)

Clock Representation

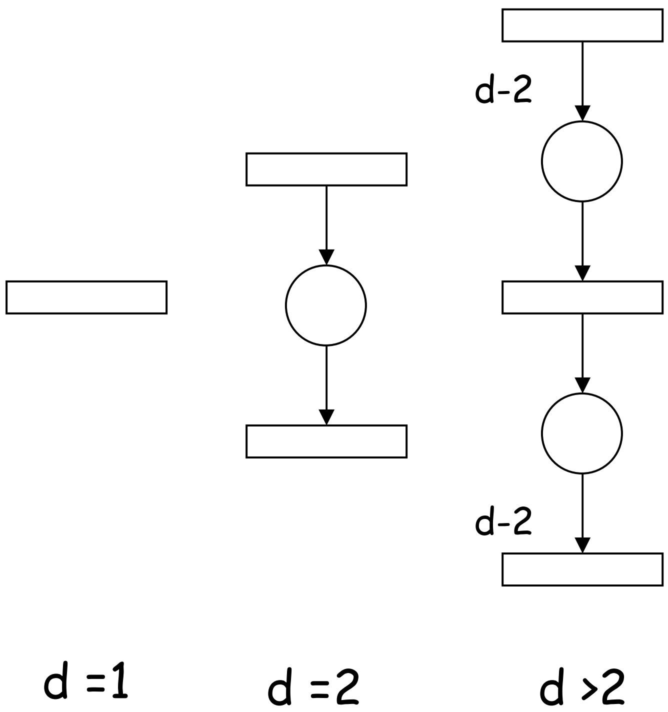
- The firing of RTC model one time unit
  - Local counter
- Fired at each period
- Activation of $Task_i$
  - Body of $Task_i$

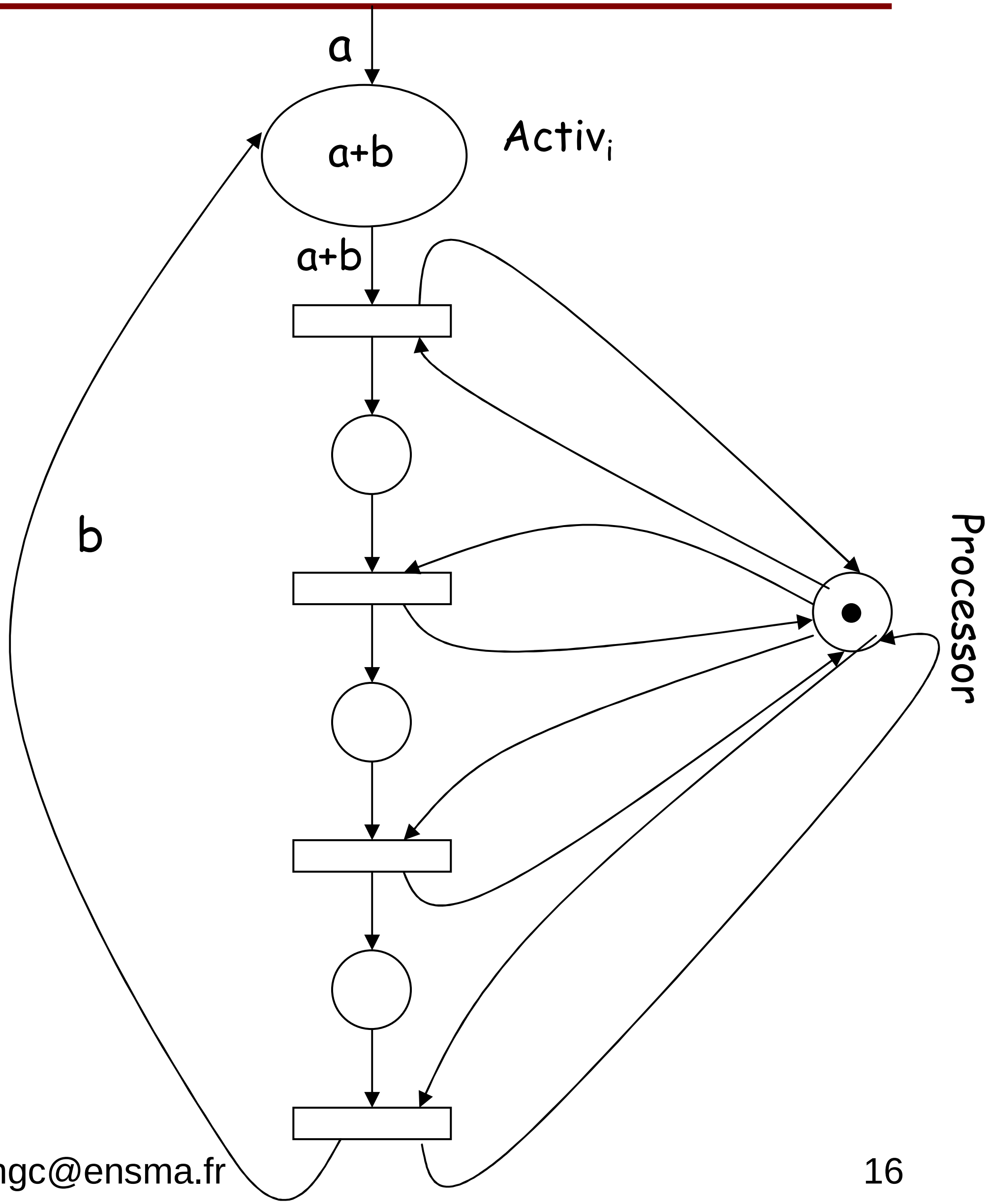


- If $0<r_i<P_i$ then $M_0(Time(i))=P_i-r_i+1$
- If $r_i=0$ then $M_0(Time(i))=1$
- If $0<r_i<P_i$ then $M(Activ_i)=b$
- If $r_i=0$ then $M(Activ_i)=a+b$

Modeling Task T_i as a Path

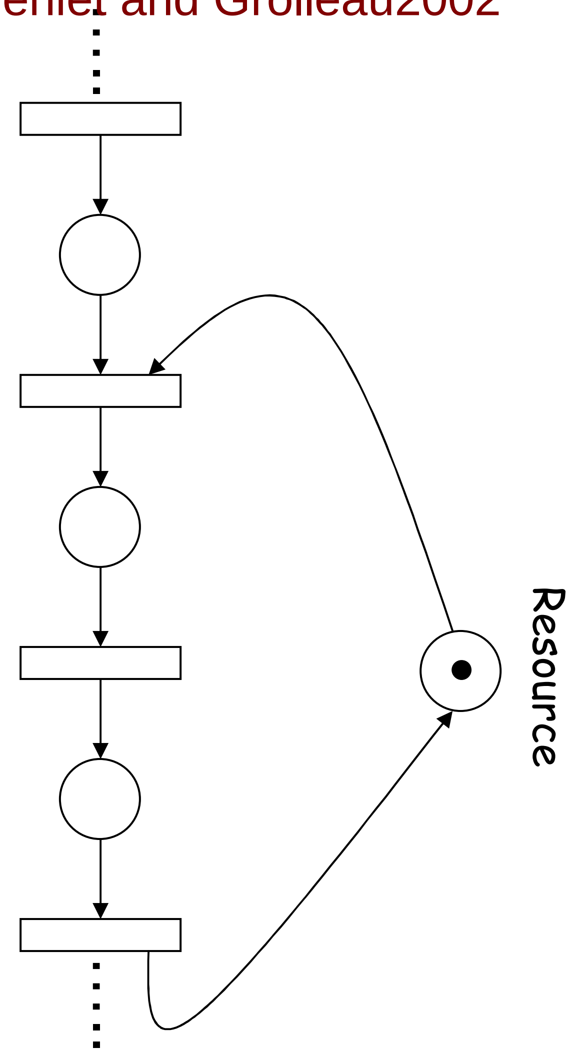
Choquet-Geniet and Grolleau2002

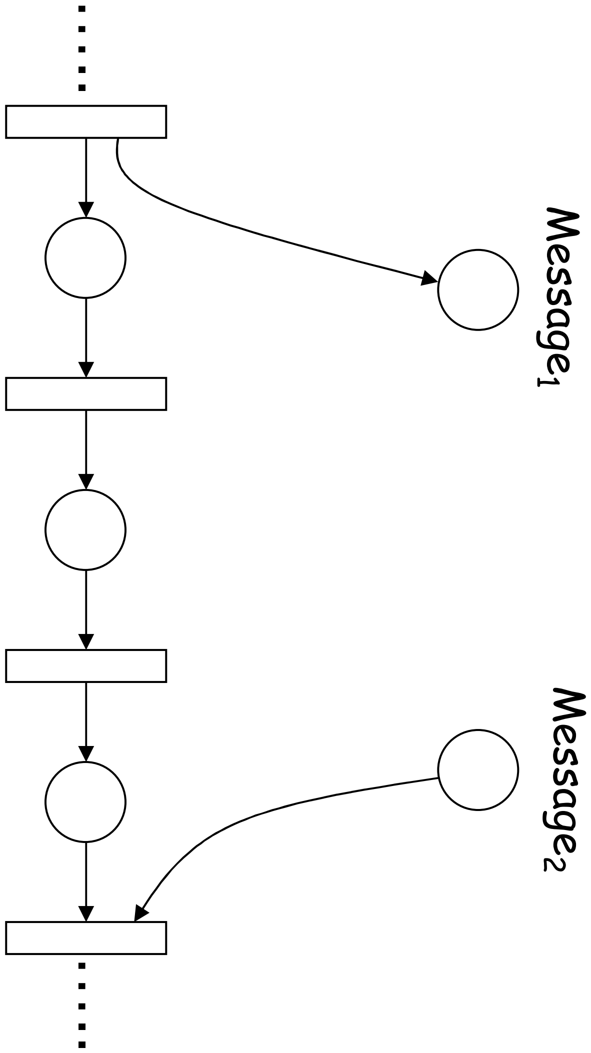




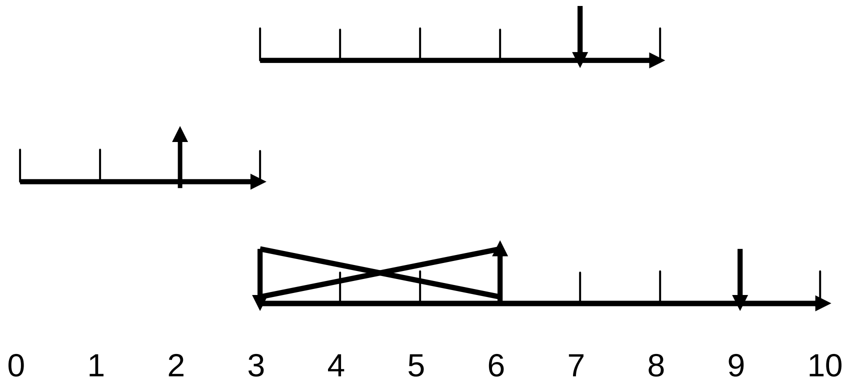
Resources and Messages

Choquet-Geniet and Grolleau2002

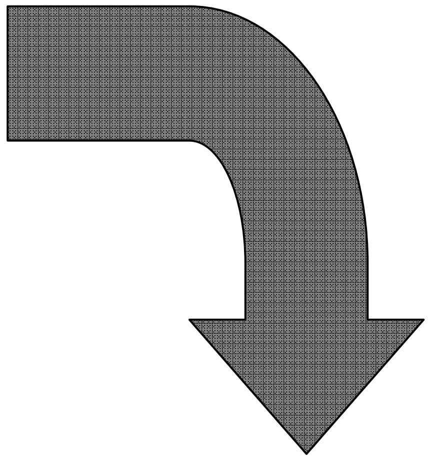




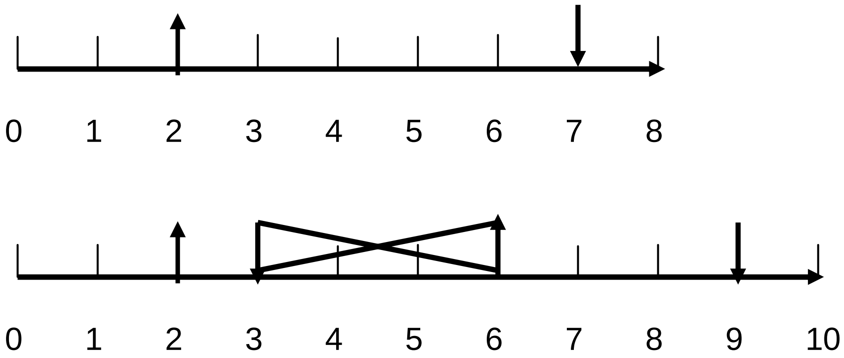
Notion of Paths



Tree-based model of previous Task T



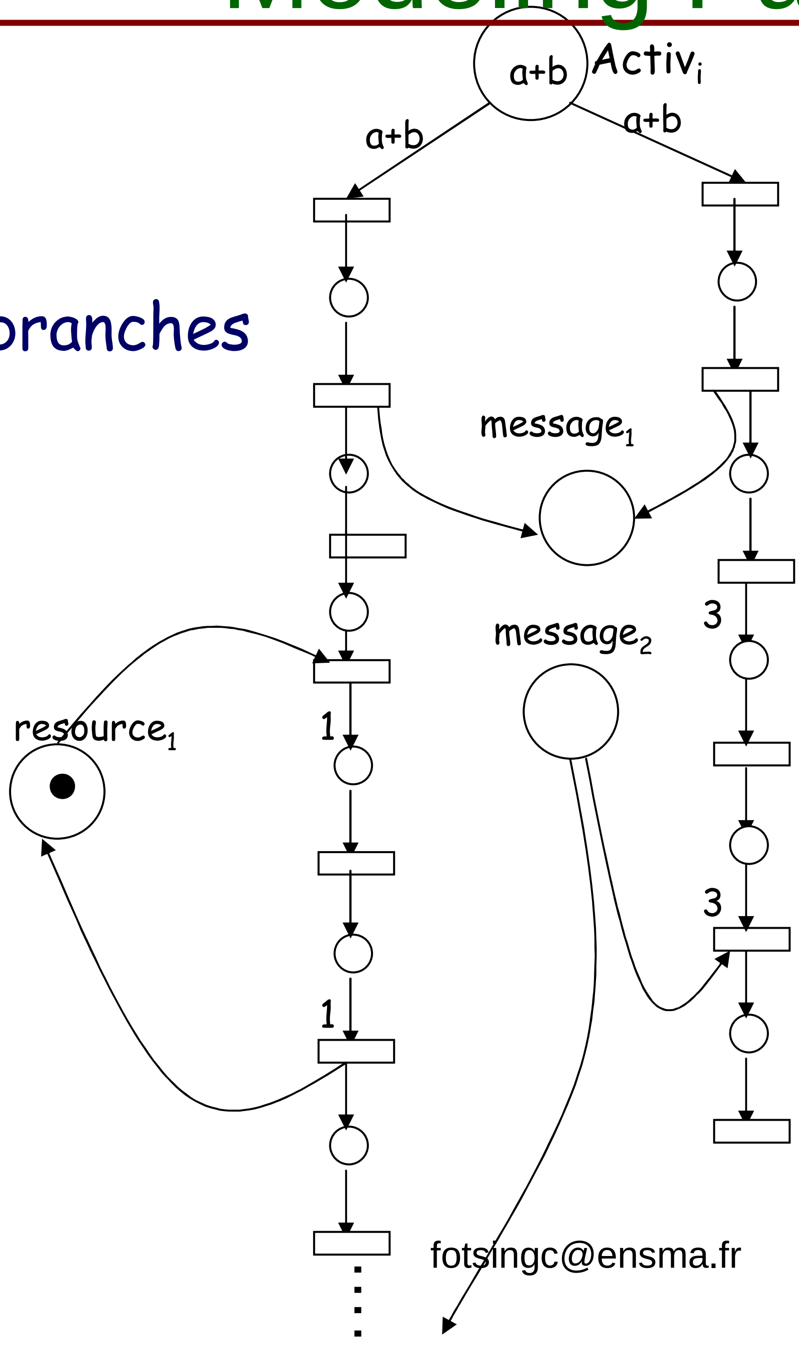
2 Behaviors ⇔ 2 Paths



Used approach for implementation

Modeling Paths

- As many branches as paths



- Paths are in Mutual Exclusion
- Each primitive occurs on it's effective path

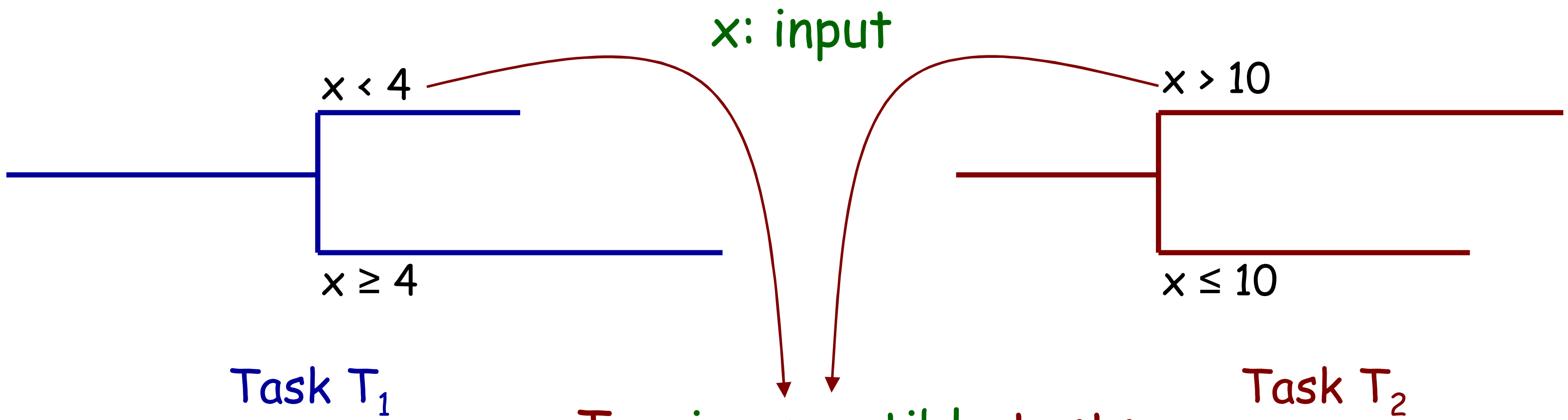
Insuring Respects of Constraints

- A terminal set I

  - ❖ $M(Time(i)) > D_i \Rightarrow M(Activ_i) = b$
  - ❖ $M(Time(i)) = 1 \Rightarrow M(Activ_i) = a+b$ or $M(Activ_i) = b$
    - ➢ To insure Critical Delay
  - ❖ $M(Time(i)) \leq P_i$
    - ➢ To insure Periodicity

Notion of Incompatible Branches

✓ $test_1$ and $test_2$ are incompatible if $test_1 \rightarrow not(test_2)$



- Two incompatible tests

+

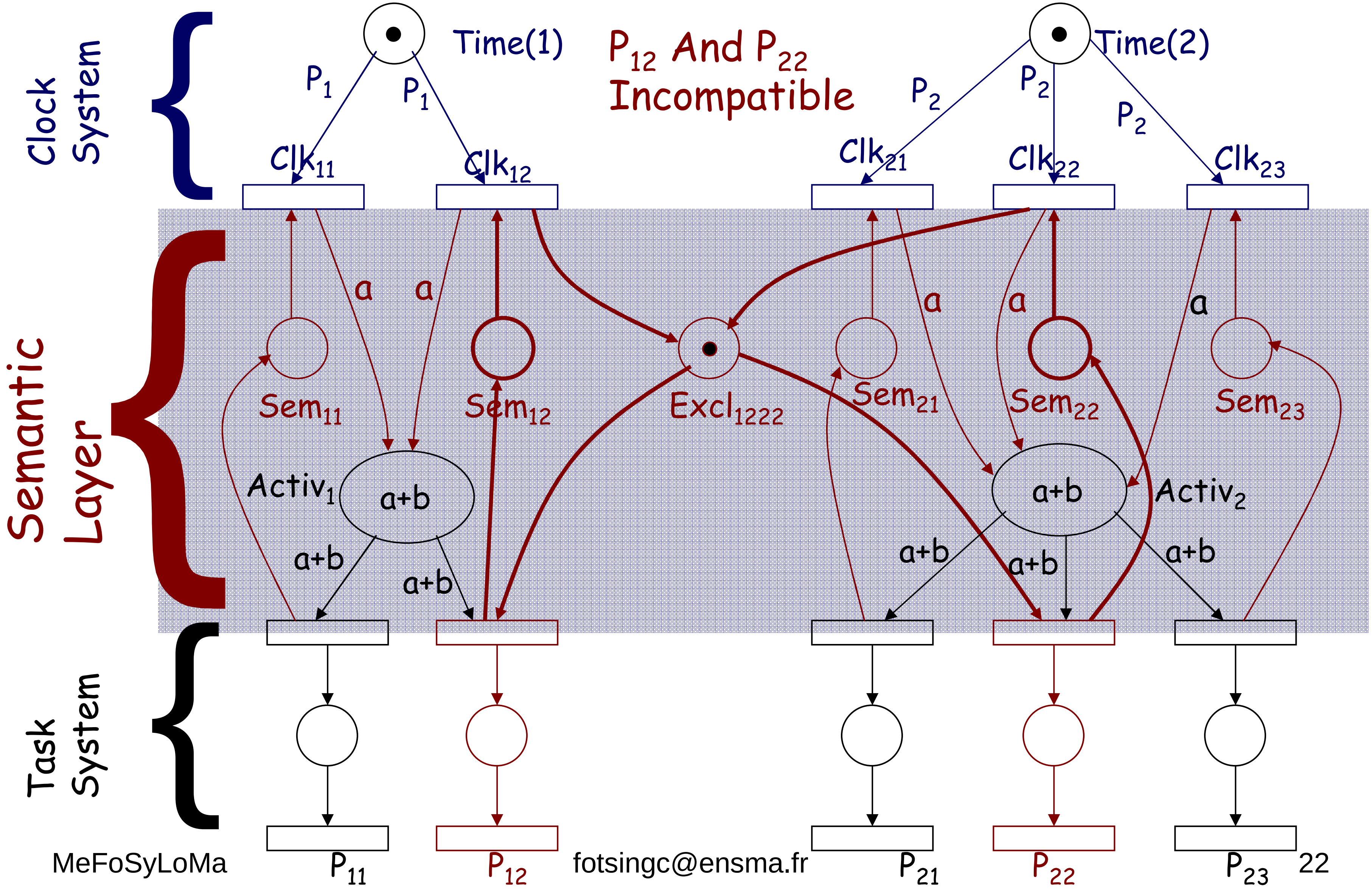
- Same value of input x

+

- $r_1 = r_2$ and $P_1 = P_2$

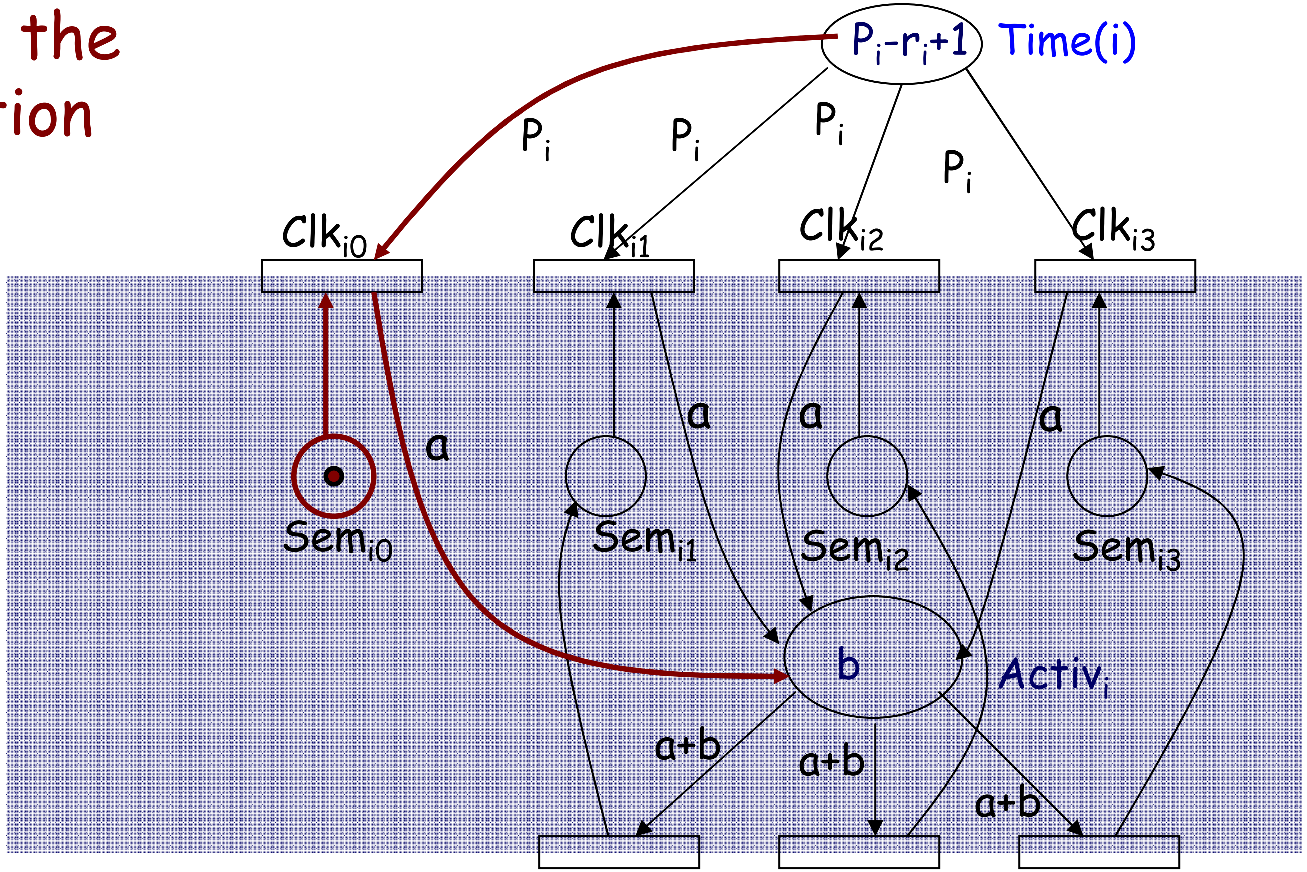
- Then Branch of T_1 is incompatible with Then Branch of T_2

Managing of Semantic of Tests (1/2)



Managing of Semantic of Tests (2/2)

Insure the activation

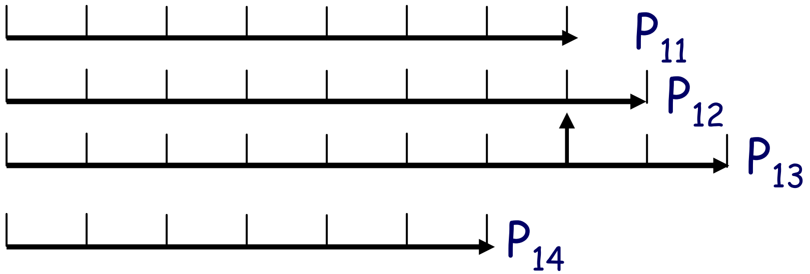


In the case ri ≠ 0

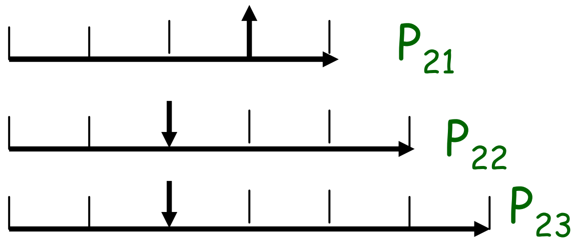
Analysis and Validation

- $M(Sem_{il}) + M(Excl_{ili'l'}) + M(Sem_{i'l'}) = 1$
  - ❖ To insure incompatibility relations
- $\sum_{l=1}^{|Pathi|} M(Sem_{il}) \leq 1$
- $M(Activ_i) = a+b \rightarrow M(Sem_{il}) = 0$
- $M(Time(i)) = 1 \rightarrow M(Excl_{ili'l'}) = 1$
  - ❖ To insure re-initialization of Petri net

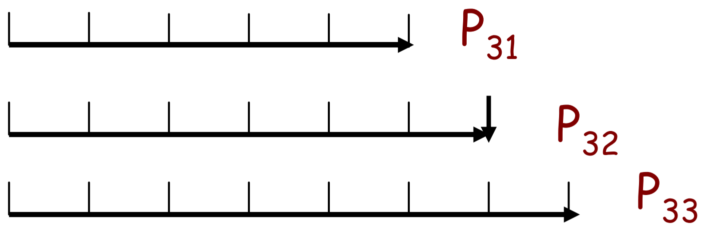
Illustration (1/2)



T_1 <0,(7,8,9,6),22,22,9> as set of paths



T_2 <0,(4,5,6),22,22> as set of paths



T_3 <0,(5,6,7),22,22> as set of paths

P_{13} Incompatible with P_{31}

P_{14} Incompatible with P_{21}

P_{23} Incompatible with P_{33}

Illustration (2/2)

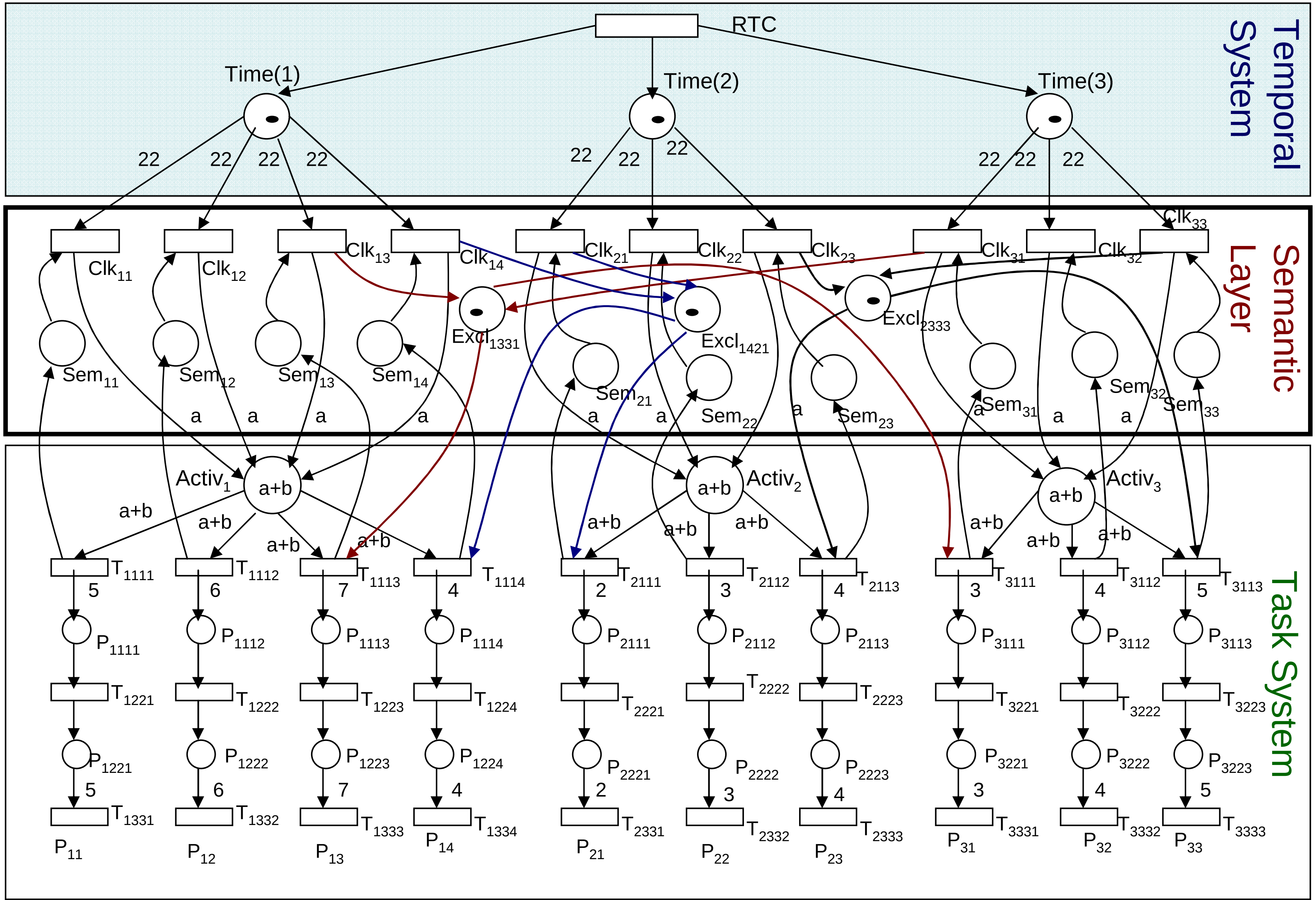


Illustration (2/2)

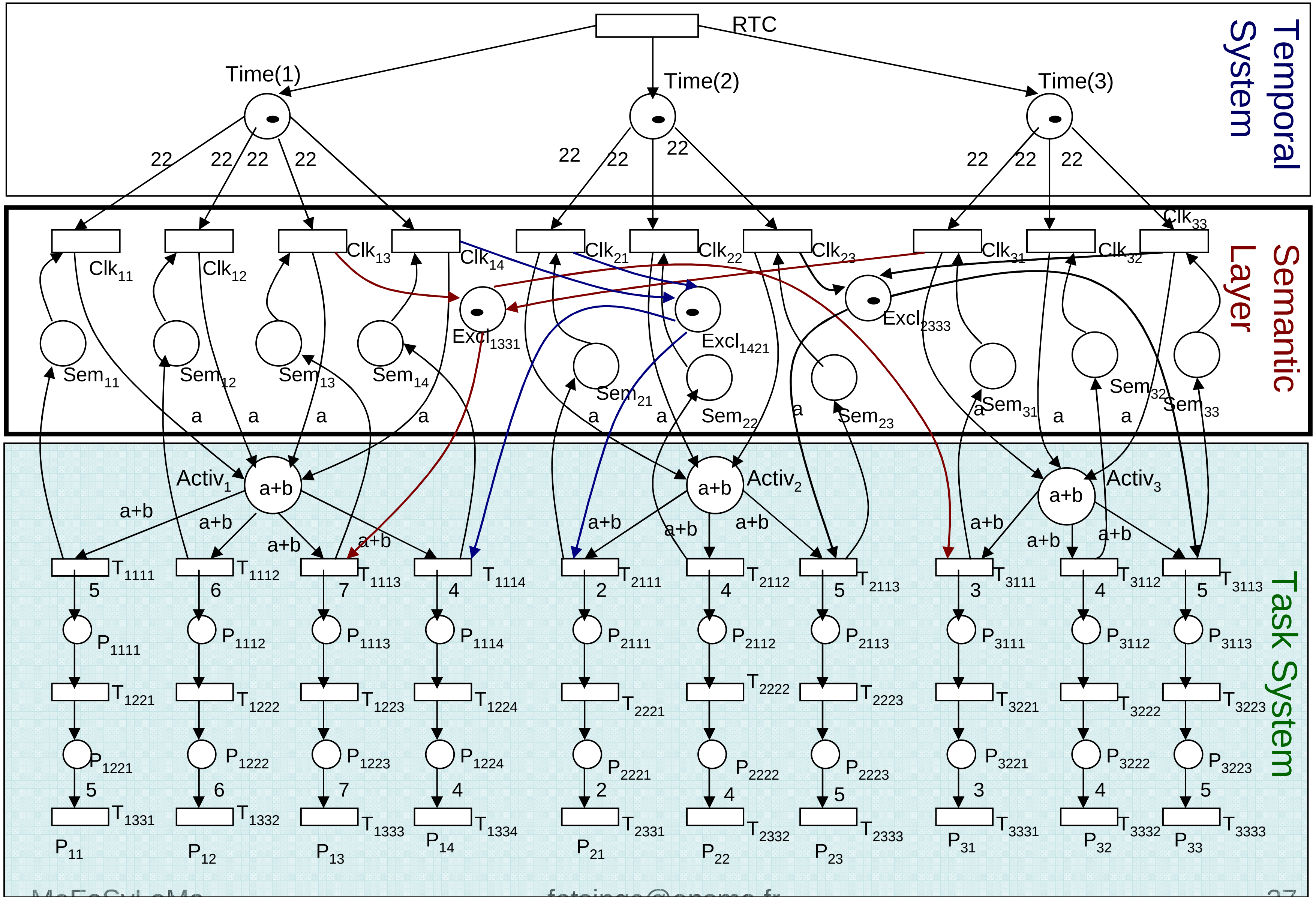
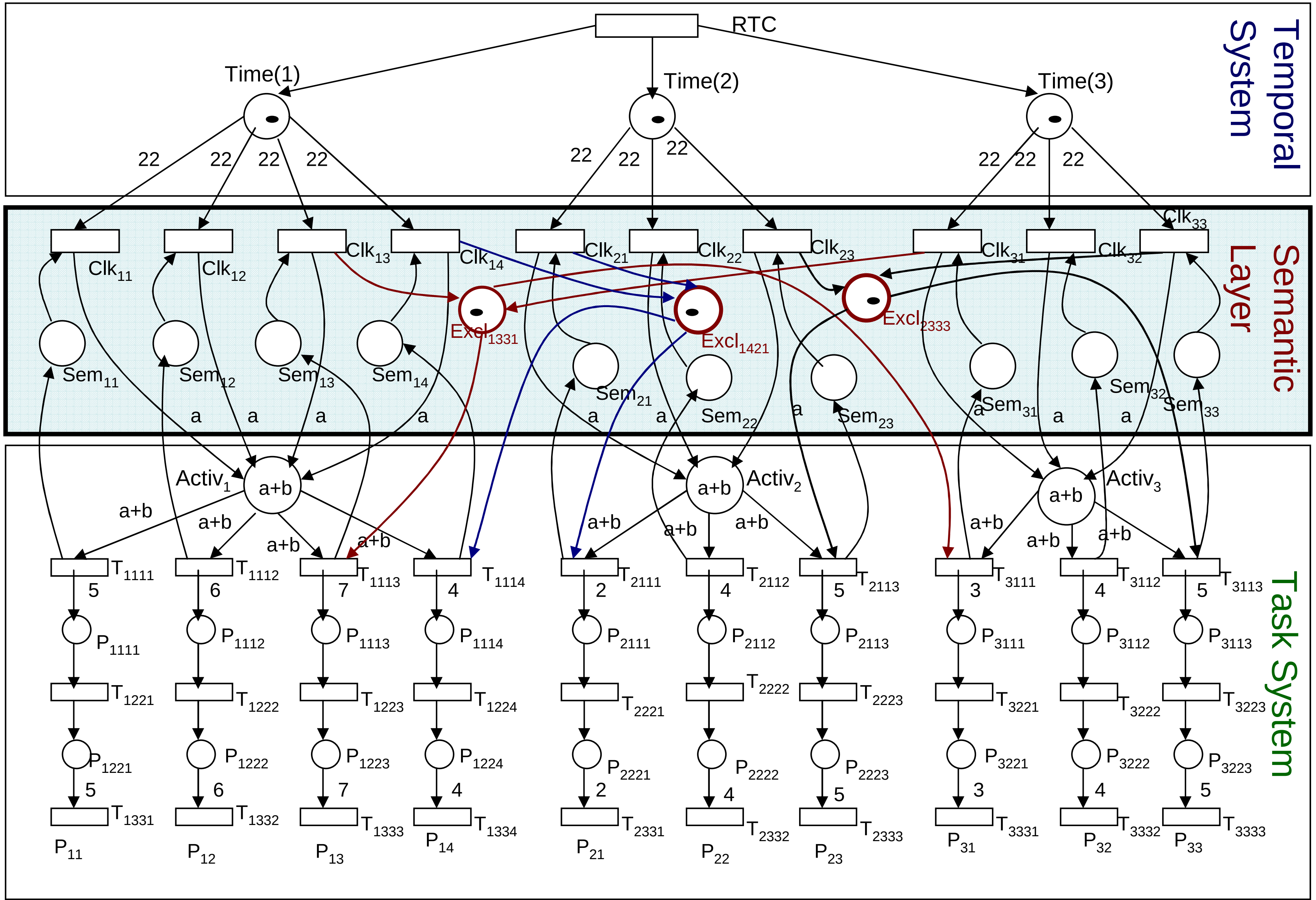
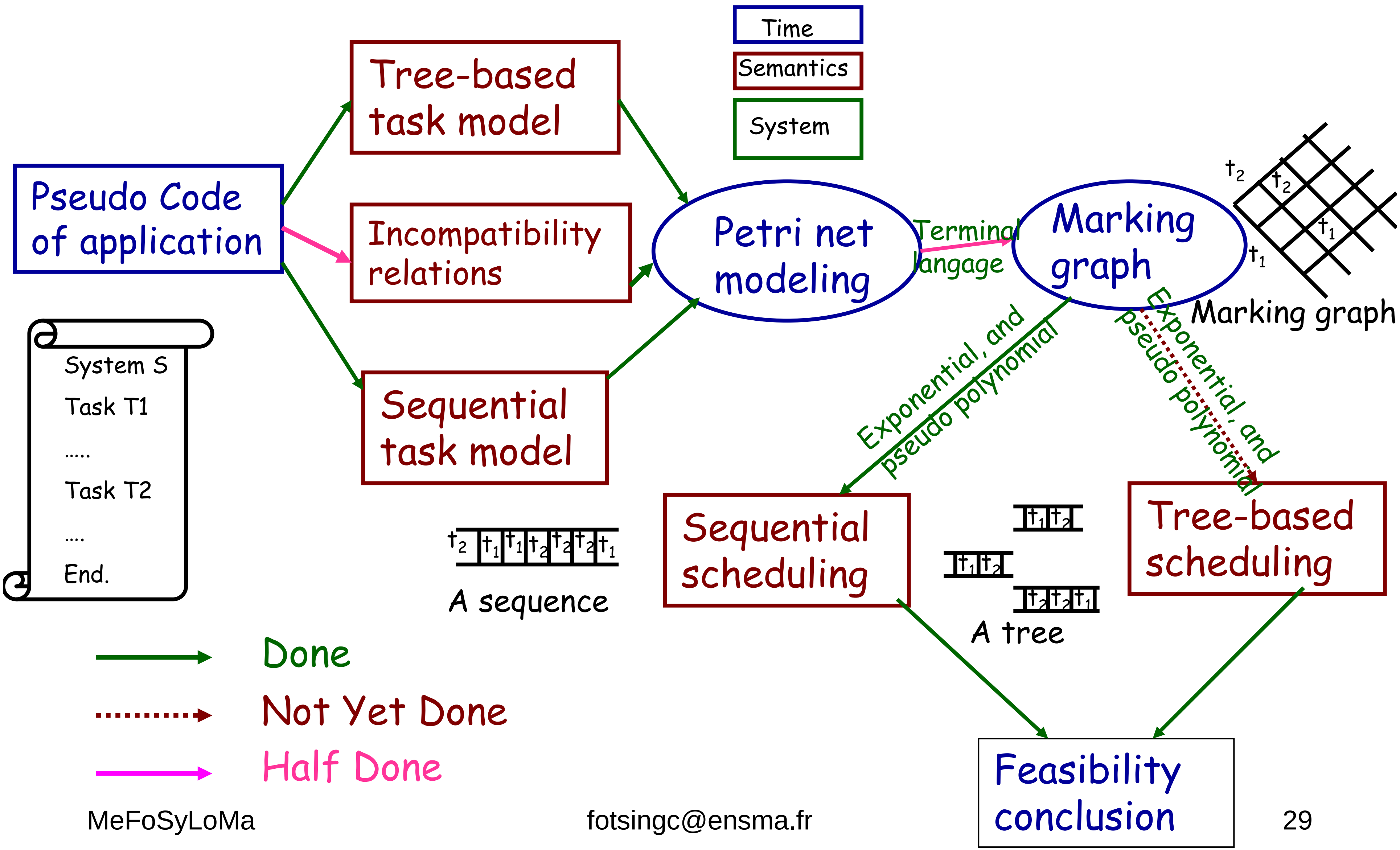


Illustration (2/2)



A Complete tree-based Schedulability Analysis



Conclusion

- Explicit modeling of conditional statements
- Explicit modeling of the semantics of tests
  - Formalization of tree-based schedules
  - Extraction techniques of tree-based schedules
  - Propose a tool for a complete automatic off-line analysis of real-time systems

Bibliography (1/2)

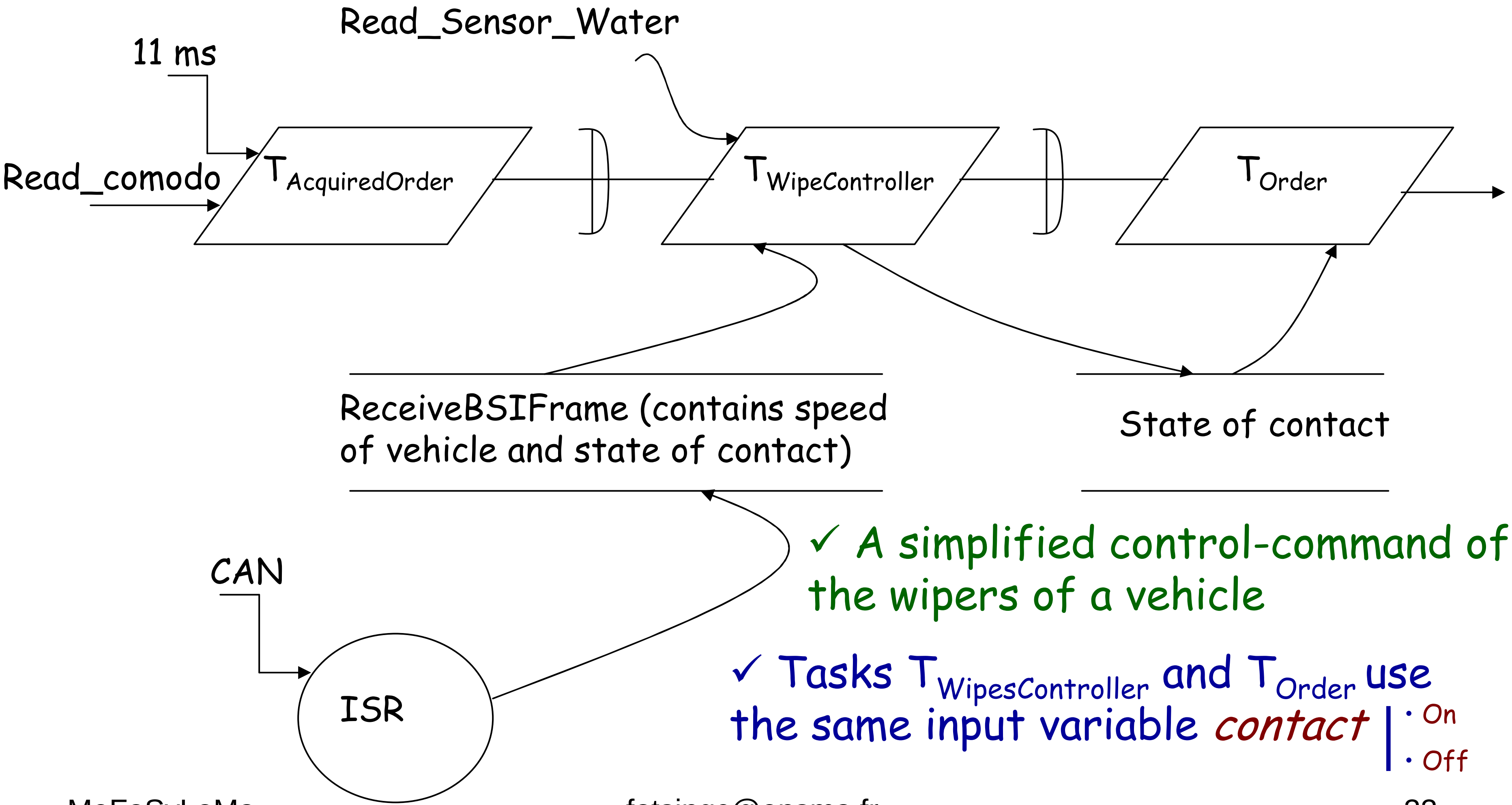
- C.L. Liu and J.W. Layland. Scheduling algorithms for multiprogramming in real-time environnement. Journal of the ACM, 20(1), 1973. PP 46-61.
- E. Grolleau and A. Choquet-Geniet. Off line comptation of real-time schedules using Petri nets. In Discrete Events Dynamic Systems. PP 311-333. Kluwer Academic Publishers, 2002. Manufactured in the Netherlands.
- R. Wilhelm and Al. The worst-case execution time problem—overview of methods and survey of tools. ACM Trans.Embedd.Comput.Syst., 7(3), april 2008.
- A. Choquet-Geniet. Les réseaux de Petri, un outil de modélisation. Sciences Sup, Mars 2006. ISBN 2100491474, Dunod.
- J.P. Babau. Etude du comportement temporel des applications temps-réel à contraintes strictes basée sur une analyse d'ordonnançabilité. PhD thesis, University of Poitiers, France, 1996.
- C. Aussaguès and V. David. A Method and a Technique to Model and Ensure Timeliness in Safety Critical Real-Time Systems. ICCCS'98.
- S. Baruah. Dynamic and Static priority Scheduling of Recurring Ral-time Tasks. Real-Time Systems, 24, PP 93-128, 2003. Kluwer Academic Publishers, 2003. Manufactured in the Netherlands.

Bibliography (2/2)

- A. Maddhukar and Al. Compositional Feasibility Analysis for Conditional Real-Time Task Models. Proc of 11 IEEE International Symposium ISORC. Orlanda, Florida, May 2008.
- C. Fotsing and Al. Tree Scheduling versus Sequential Scheduling. CARS@EDCC, ACM Digital Library, ISBN 978-1-60558-915-2. April 2010.
- S.K. Baruah. Feasibility Analysis of Recurring Branching Tasks. 10th Euromicro Real-Time Systems. Berlin, Germany. ISBN 0-8186-8503-4. June 1998.
- C. Fotsing and Al. A realistic model of real-time systems for efficient scheduling. 33rd Annual IEEE Sofware Engineering Workshop. Swedish, 2009.
- G.C. Buttazo. Hard Real-Time Computing Systems: Predictable Scheduling Algorithms and Applications. Kluwer Academic, 1997.

An Example of Real-Time System (1/2)

➢DARTS Conception of system



- ✓ A simplified control-command of the wipers of a vehicle
- ✓ Tasks $T_{WipesController}$ and T_{Order} use the same input variable *contact* | • On • Off

An Example of Real-Time System (2/2)

Task $T_{AcquiredOrder}$ <0, 2, 11, 11>

All 11 ms do

block(2); - - reads the position of the comodo (position may be stop/1/2/3)

Send(m_1); - - sends the state of the comodo

end;

Task $T_{WipesController}$ <0, 6, 11, 11>

Receive(m_1); - - waits the message giving the state of the comodo

block(2); - - extracts speed and state of contact of vehicule on ReceiveBSIFrame

If (contact = 0) [duration of test = 1] then - - if contact is off

block(1); - - calculation of the order according to the speed and the comodo

Send(m_2); - - sends command 0 to task T_{Order} the state

else

block(3) - - calculation of the order according to the speed and the comodo

Send(m_2); - - sends calculate command to task T_{Order}

endif;

end;

Task T_{order} <0, 4, 11, 11>

Receive(m_2); - - waits the value of the order

block(1); - - extract state of contact of vehicule

If (contact = 0) [duration of test = 1] then - - if contact is off

block(2); - - application of the order on wipers

else

block(1); - - application of the order on wipers

end;