Modeling Real-time Systems

Joseph Sifakis

VERIMAG & ARTSIST2 NoE

RTSS 2004

December 5-8, 2004

Lisbon

Motivation - Modeling

Modeling plays a central role in systems engineering

- Can profitably replace experimentation on actual systems
- Can provide a basis for rigorous system development and implementation (model-based approaches).

Modeling real-time systems

- Raises hard problems about concepts, languages and their semantics e.g. What is an architecture? What is a scheduler? How synchronous and asynchronous systems are related?

- Requires a deep understanding of basic system design issues such as development methodologies (combination of techniques and tools, refinement) and architecture design principles

It's not just playing with graphical tools ….

Outline

Key Research issues

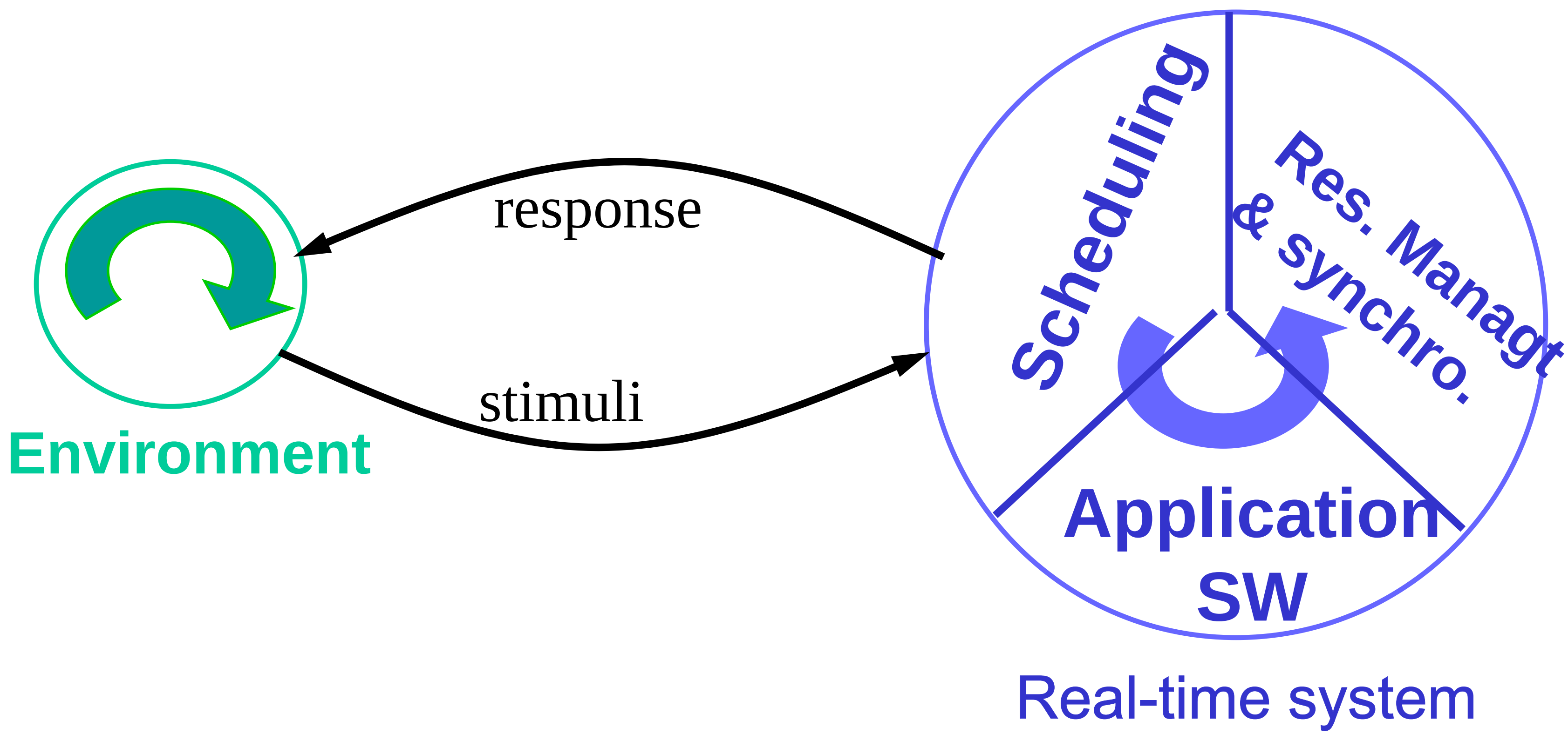
- Modeling Real-time systems
- From application SW to implementations
- Component-based construction

The modeling framework

- Principles
- Interaction models
- Scheduler modeling
- Timed models with priorities

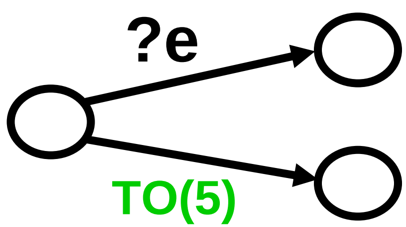
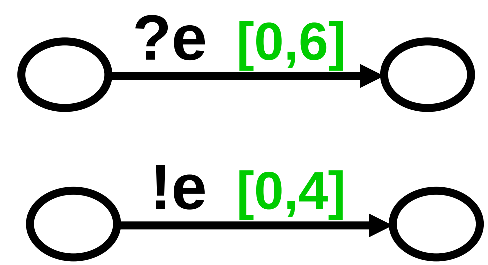
Discussion

Modeling real-time systems

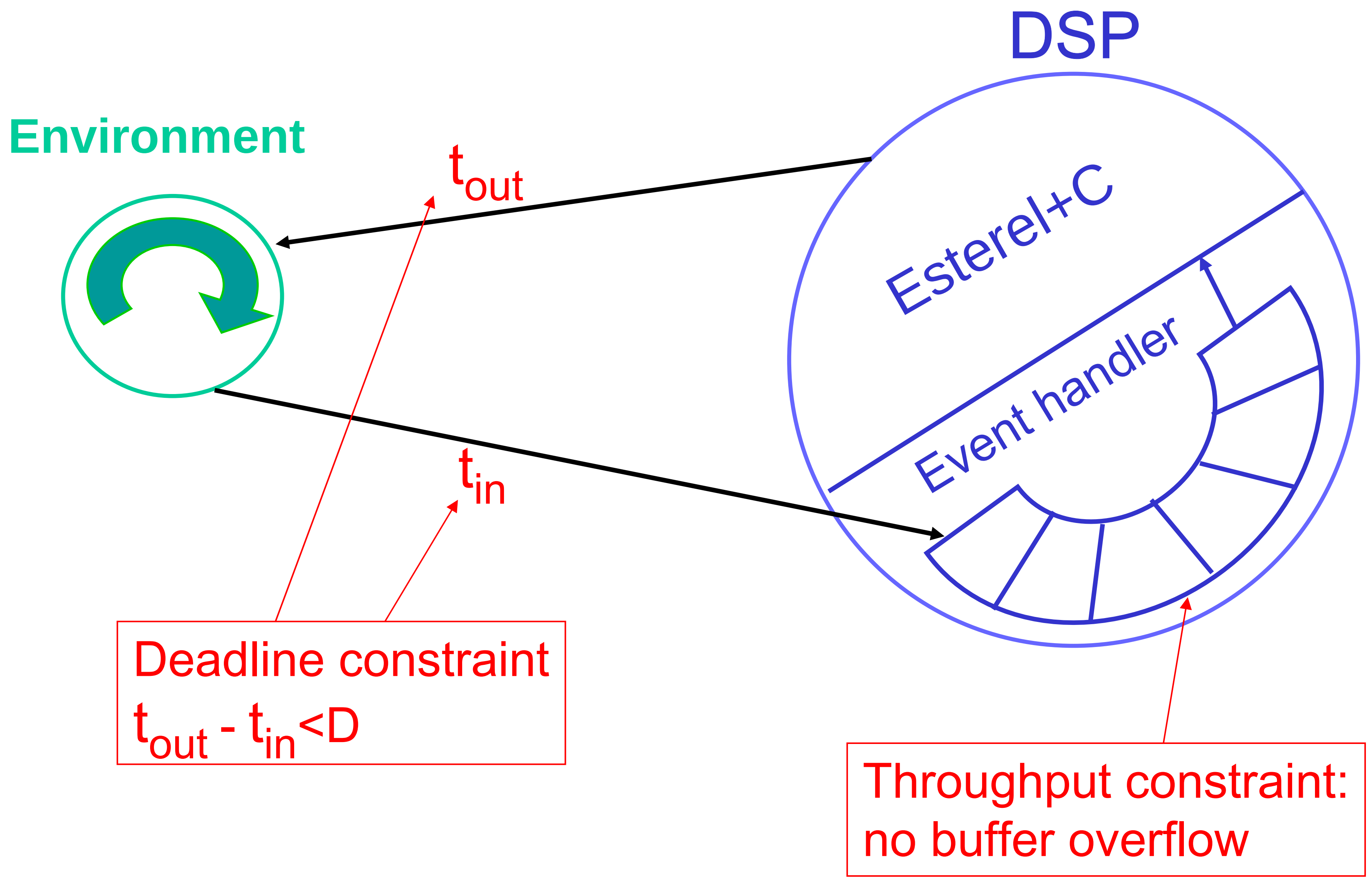


Thesis :
A Timed Model of a RT system can be obtained by "composing" its application SW with timing constraints induced by both its execution and its external environment

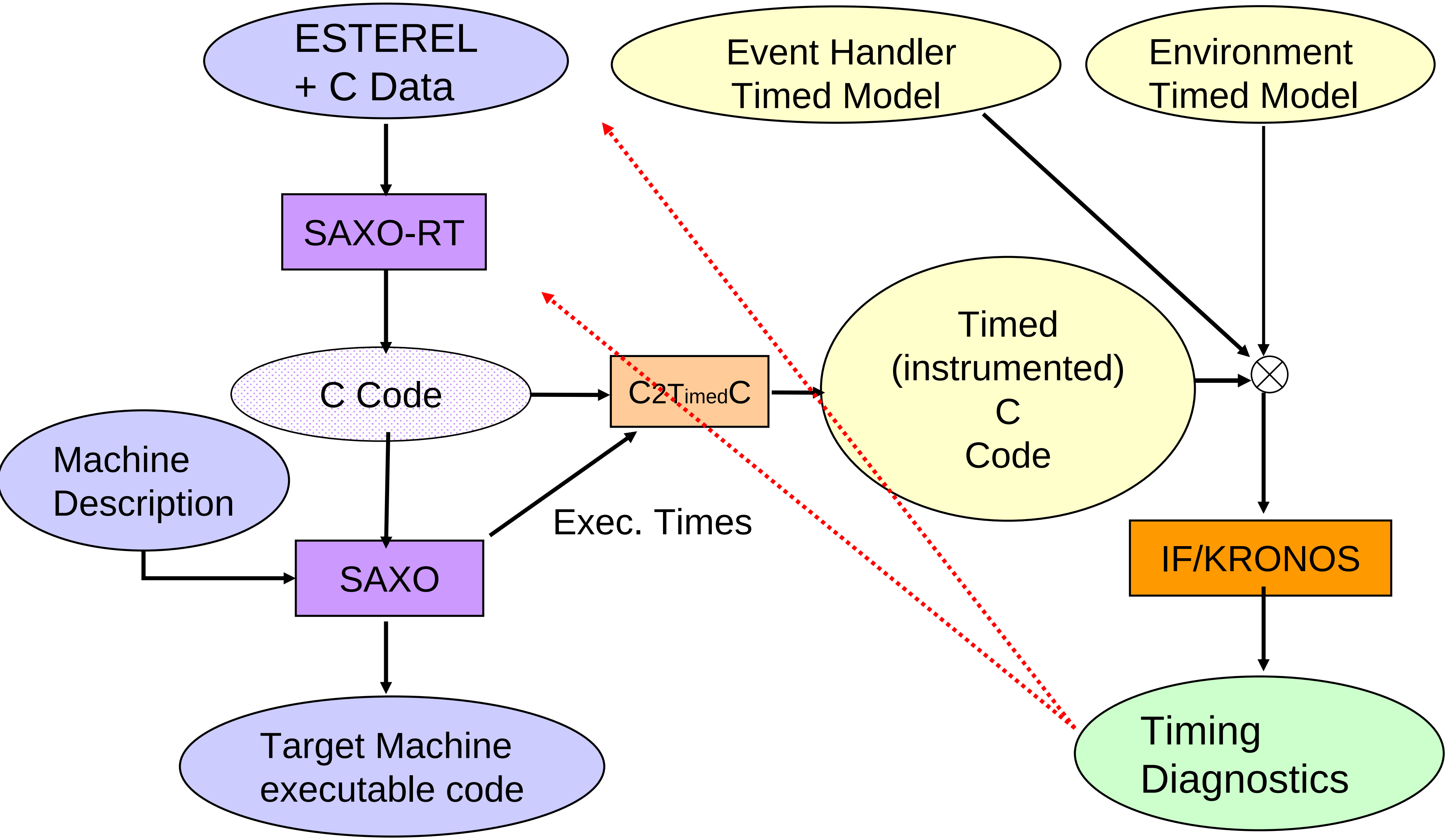
Modeling real-time systems

	Application SW ➡	Timed model
DESCRIPTION	Reactive machine (untimed)	Reactive machine + External Environment + Execution Platform
TIME	Reference to physical (external) time	Quantitative (internal) time Consistency pbs - timelocks
TRIGGERING	Timeouts to control waiting times 	Timing constraints on interactions 
ACTIONS	No assumption about Execution Times Platform-independence	Assumptions about Execution Times Platform-dependence

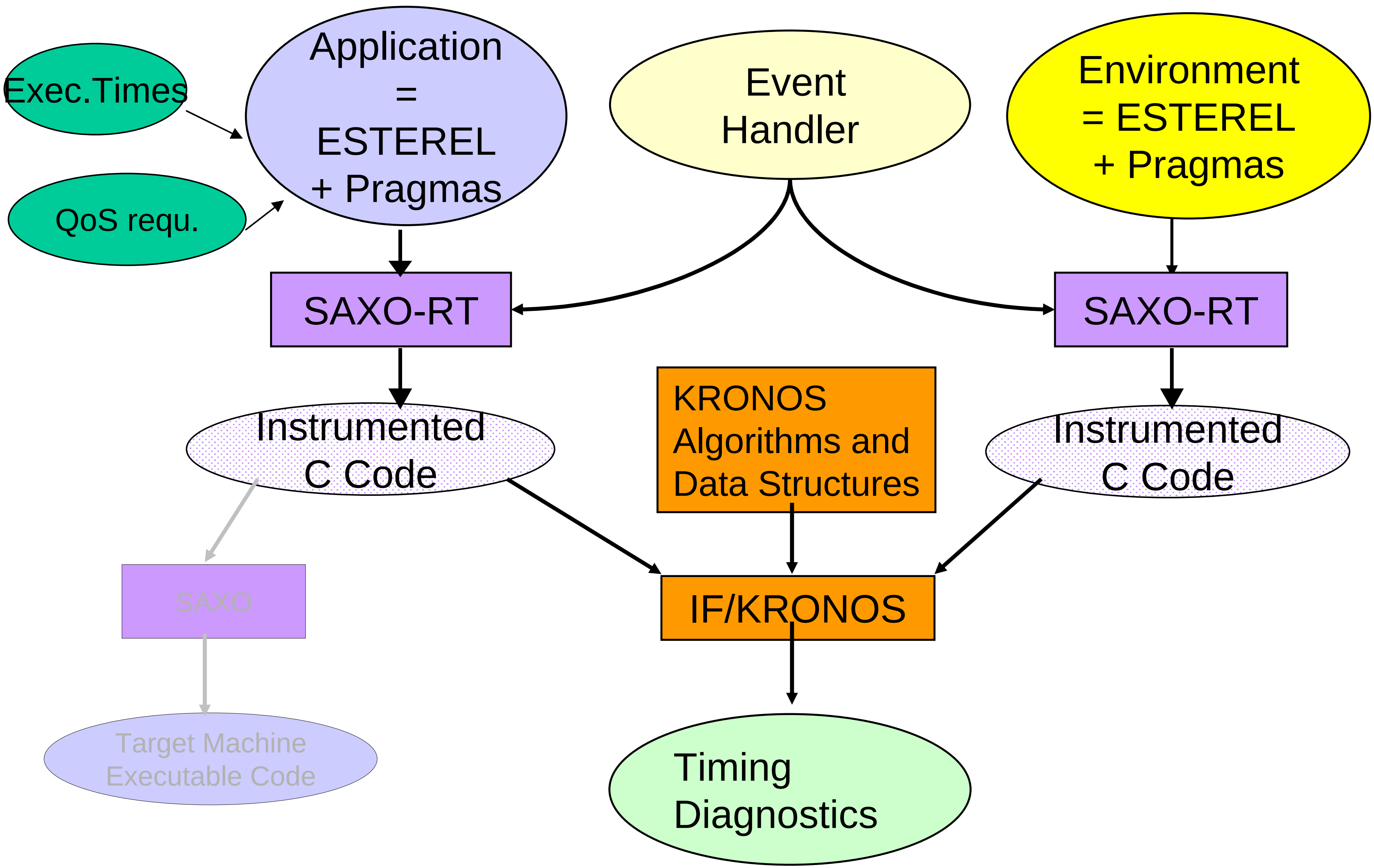
Modeling real-time systems – Taxys (1)



Modeling real-time systems – Taxys (2)



Modeling real-time systems – Taxys(3)



Outline

Key Research issues

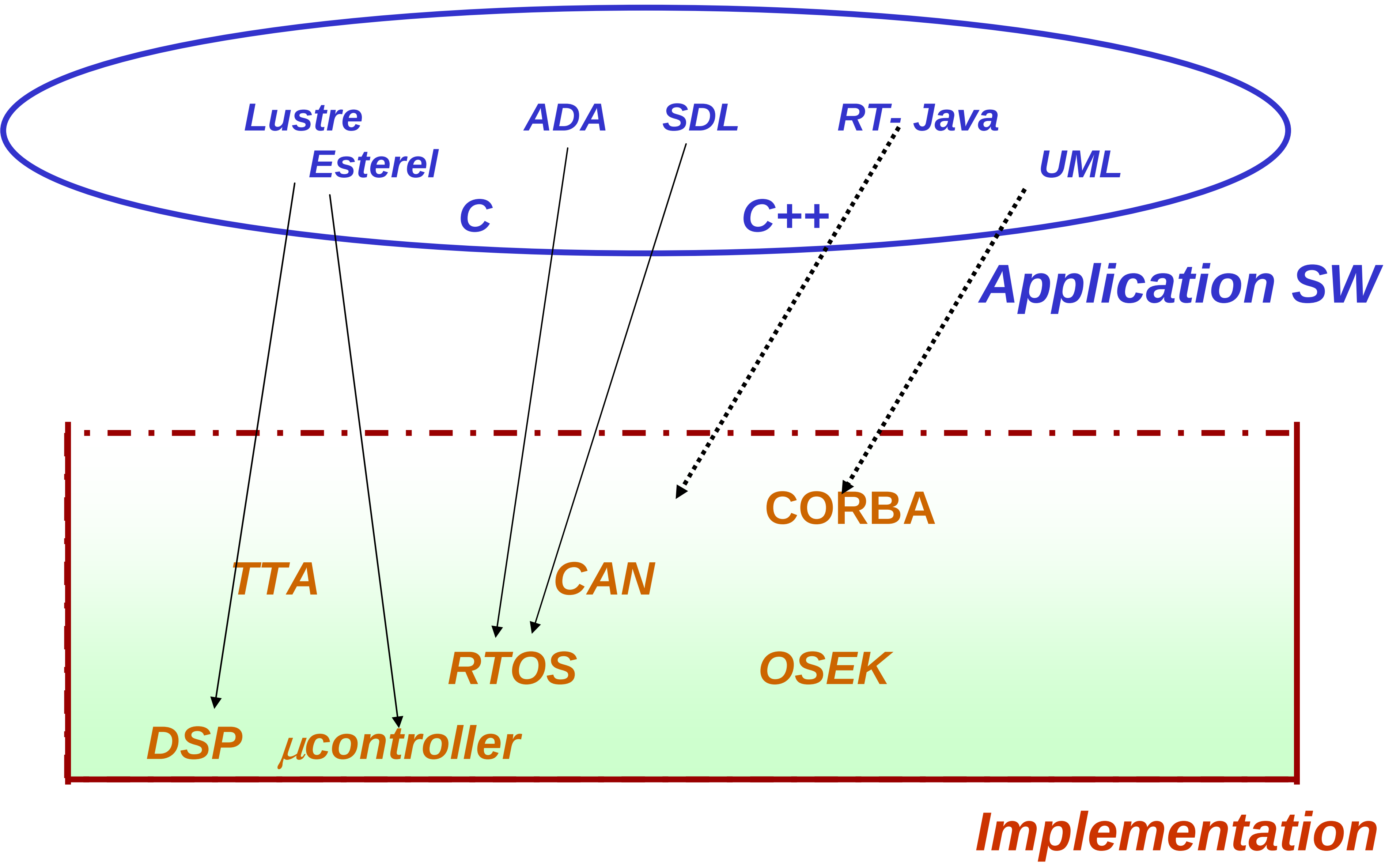
- Modeling Real-time systems
- From application SW to implementations
- Component-based construction

The modeling framework

- Principles
- Interaction models
- Scheduler modeling
- Timed models with priorities

Discussion

From application SW to implementations



From application SW to implementations

Logical abstract time

High level structuring constructs and primitives

Simplifying synchrony assumptions wrt environment

Application SW

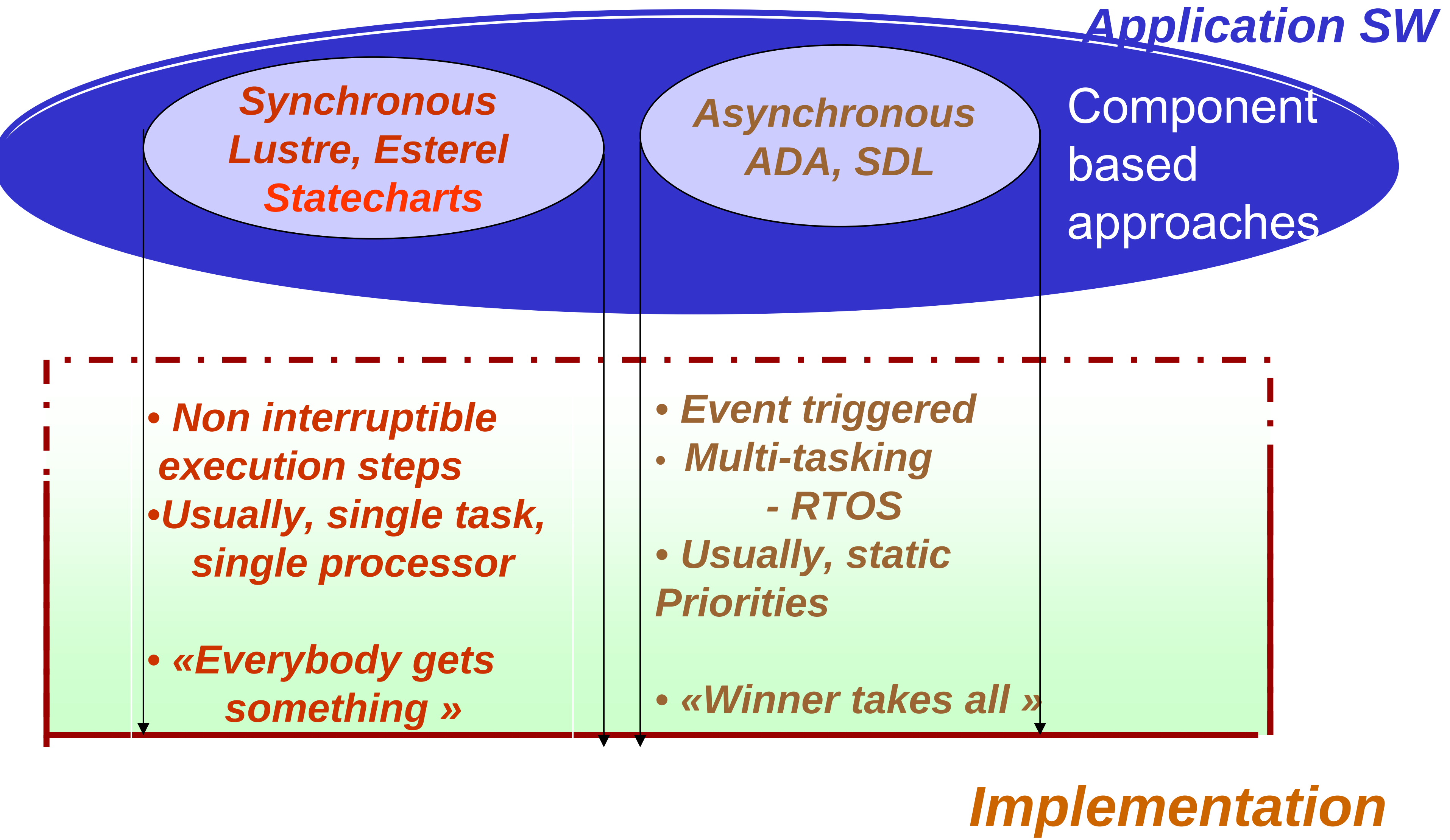
Physical, Non functional properties

Execution times, interaction delays, latency

Task coordination, resource management, scheduling

Implementation

From application SW to implementations – synchronous vs. asynchronous



Outline

Key Research issues

- Modeling Real-time systems
- From application SW to implementations
- Component-based construction

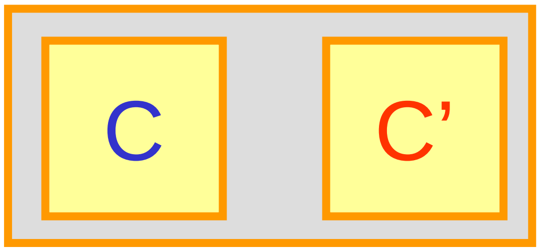
The modeling framework

- Principles
- Interaction models
- Scheduler modeling
- Timed models with priorities

Discussion

Component-based construction

Construction problem: Given a component **C** and a property **P** find **C'** and **glue** such that



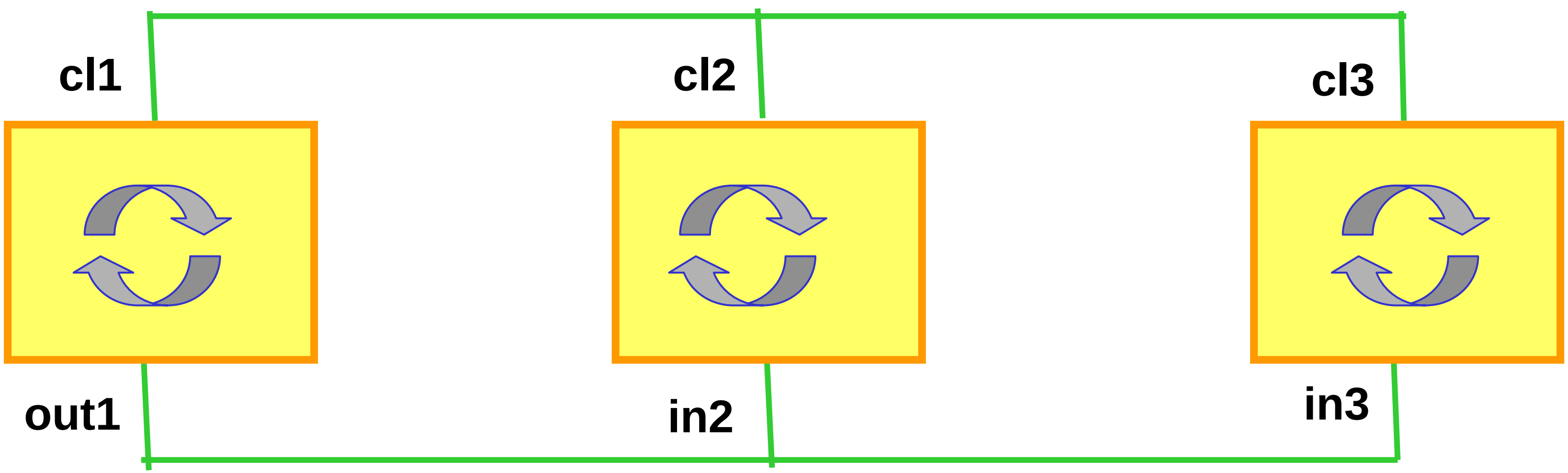
satisfies P

Two key inter-dependent issues:

- Heterogeneity of the glue – need for a unified theory
- Methods guaranteeing correctness by construction for at least some basic properties e.g. deadlock-freedom

Component-based construction – the glue

Assign meaning to diagrams

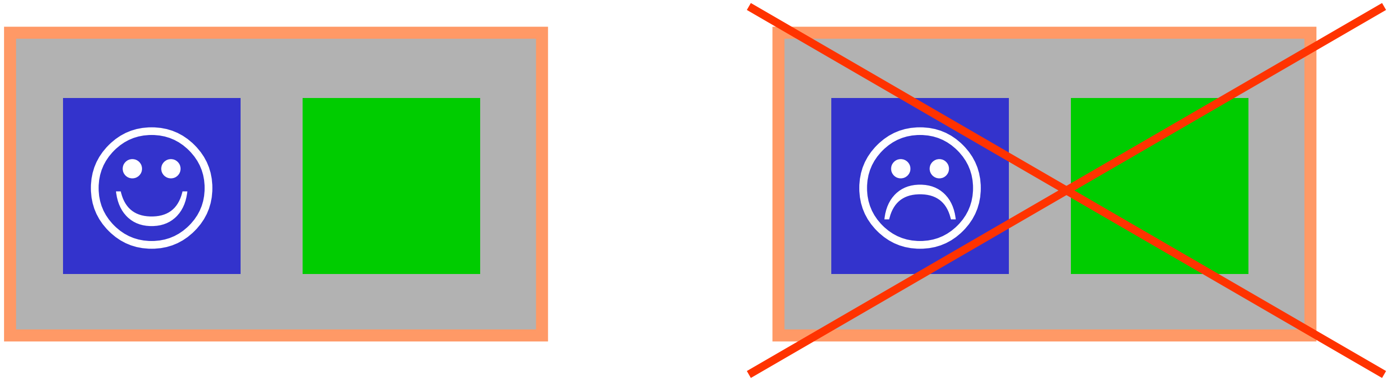


- What are the possible interactions?
- How computation threads of components are related e.g. synchronous vs. asynchronous execution

Correctness by construction - Composability

Make the new without breaking the old

Composability rule: guarantees that a component property is preserved when merged in an environment



- *We badly need composabilty results*
- *Property stability phenomena are poorly understood*
    - *- feature interaction*
    - *- non composability of scheduling algorithms*

Correctness by construction - Compositionality

Build correct systems from correct components

Compositionality rule: guarantees that if the components of a system meet a given property then this property is preserved by composition

We need compositionality results that preserve properties other than safety properties e.g. progress properties

Outline

Key Research issues

- Modeling Real-time systems
- From application SW to implementations
- Component-based construction

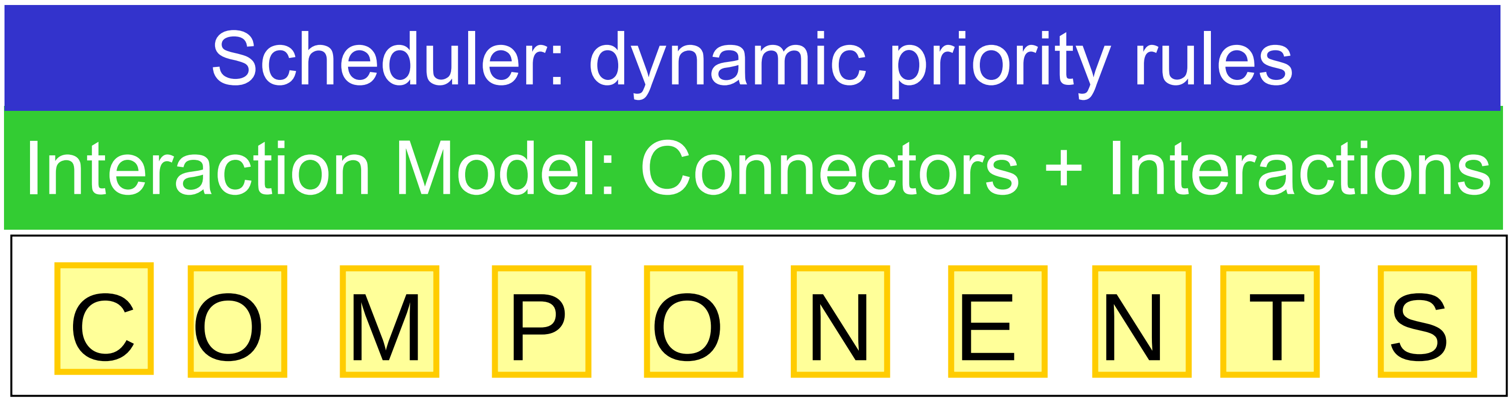
The modeling framework

- Principles
- Interaction models
- Scheduler modeling
- Timed models with priorities

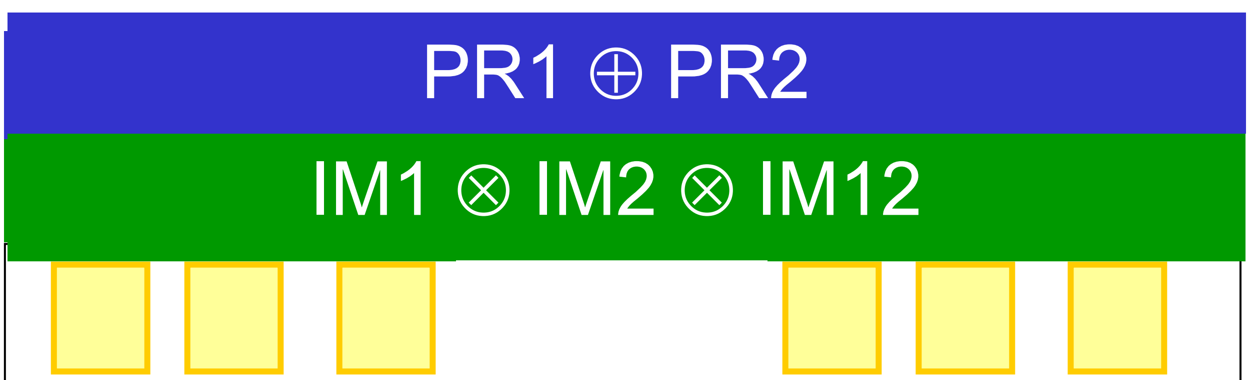
Discussion

Modeling framework principles

Layered description – separating the issues



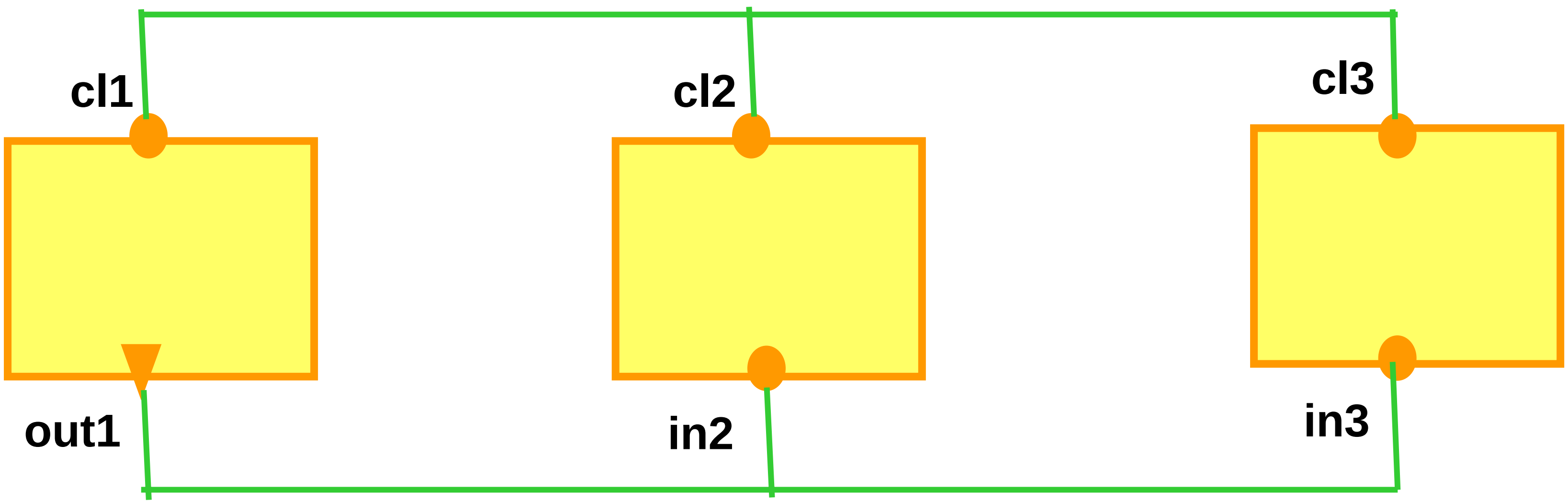
Composition (incremental description)



Modeling framework principles – Heterogeneous interaction

Interactions may

- be **atomic** or non **atomic**
- involve **strong** or **weak** synchronization



- **Strong synchronization**: all the participants must agree

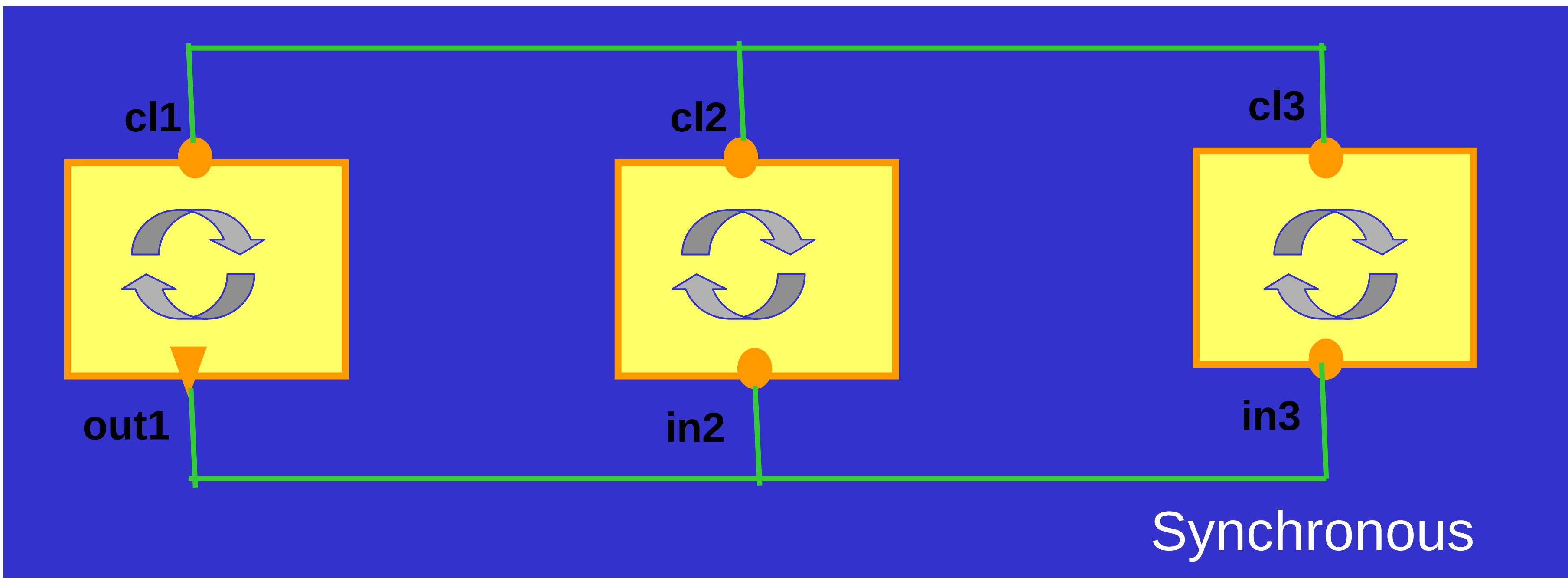
  Example: only {cl1,cl2,cl3} is possible

- **Weak synchronization**: interaction is initiated by a “leader”

  Example: {out1,in2,in3}, {out1,in2}, {out1,in3}, {out1}, are possible

Modeling framework principles – Heterogeneous execution

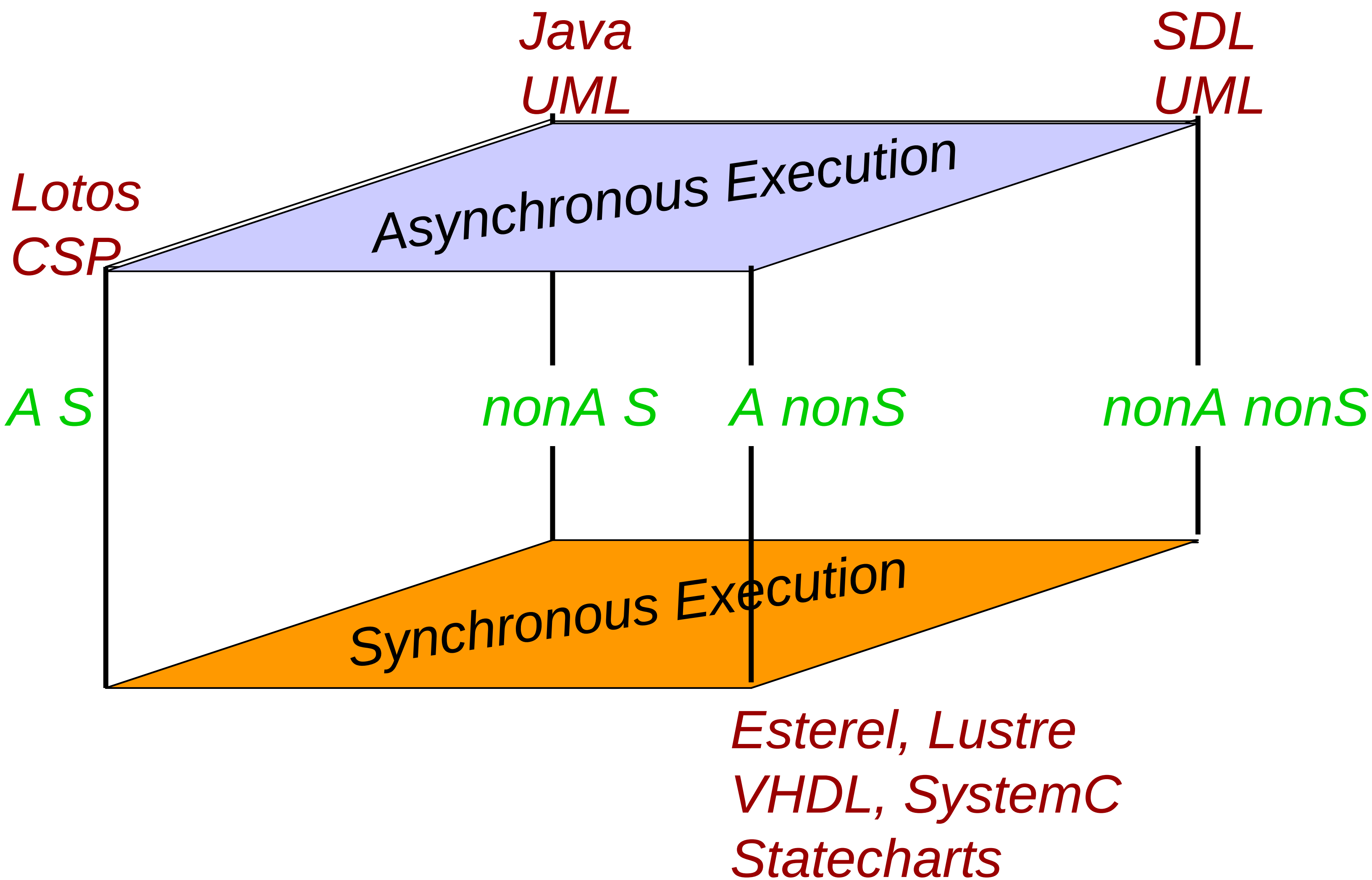
- **Asynchronous:** independent threads modulo interaction constraints
- **Synchronous:** additional strong synchronization constraints enforced by using scheduling mechanisms



Synchronous = Asynchronous + Scheduling

Modeling framework principles - Example

A: Atomic interaction *S: Strong synchronization*



Outline

Key Research issues

- Modeling Real-time systems
- From application SW to implementations
- Component-based construction

The modeling framework

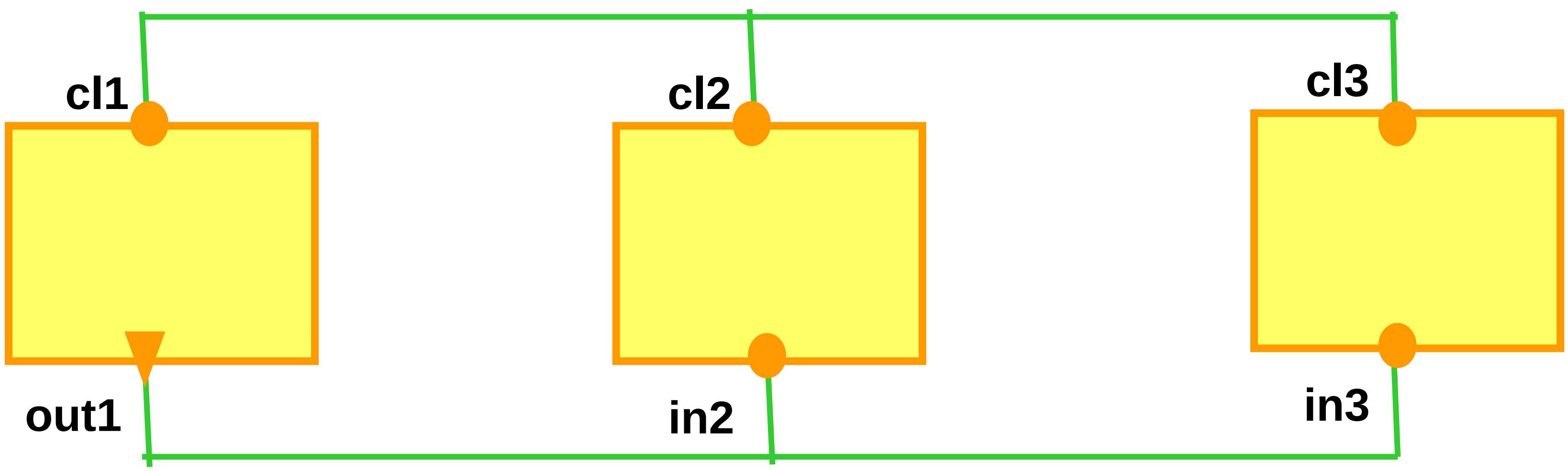
- Principles
- Interaction models
- Scheduler modeling
- Timed models with priorities

Discussion

Interaction models

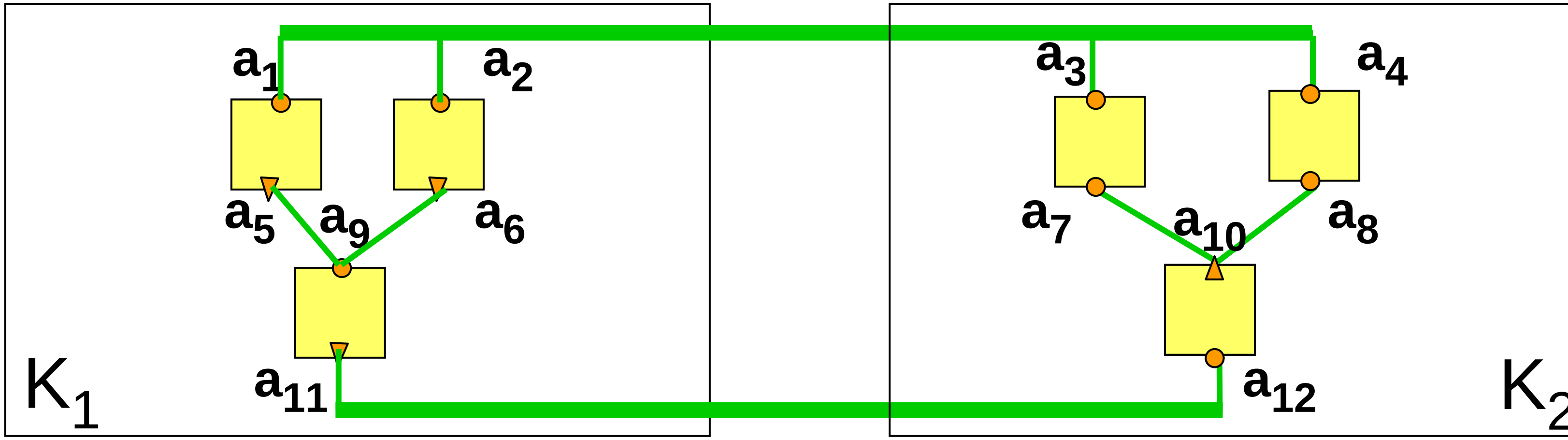
- **Connectors** are maximal sets of compatible actions **-** Interactions are subsets of connectors

- **Actions types (complete ▼, incomplete ●)** determine the set of possible interactions : interactions are either maximal or contain some complete action



Interactions: {cl1,cl2,cl3}, {out1}, {out1,in2}, {out1,in3}, {out1,in2, in3}

Interaction models - Composition



IM[K_1,K_2]:

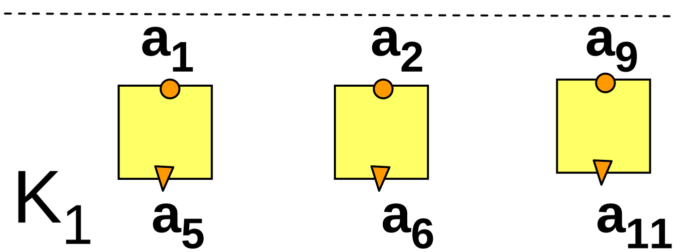
C[K_1,K_2] : {a_1, a_2, a_3, a_4}, {a_{11}, a_{12}}

CI[K_1,K_2] : {a_1,a_2,a_3,a_4}, {a_{11}}, {a_{11},a_{12}}

IM[K_1]:

C[K_1] : {a_1, a_2}, {a_5, a_9},{a_6, a_9}

CI[K_1] : a_5, a_6, a_{11}, {a_5, a_9}, {a_6,a_9}

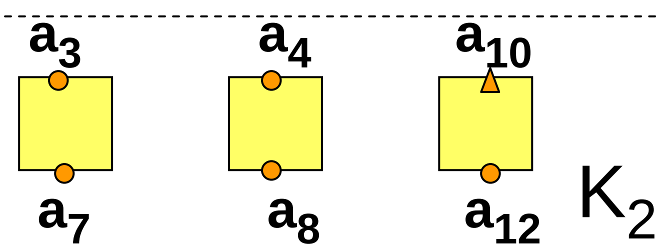


||

IM[K_2]:

C[K_2] : {a_3, a_4}, {a_7, a_{10}}, {a_8, a_{10}}

CI[K_2] : a_{10}, {a_7, a_{10}}, {a_8,a_{10}}



Interaction models – Composition (2)

$IM[K_1,K_2]$:

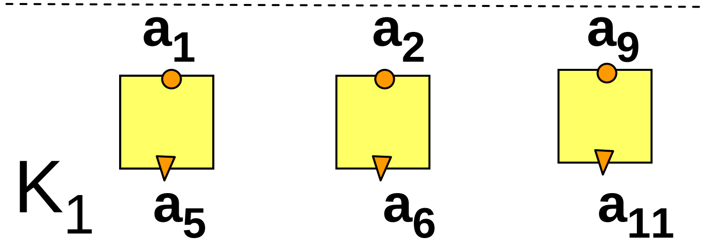
$C[K_1,K_2] : \{a_1, a_2, a_3, a_4\}, \{a_{11}, a_{12}\}$

$CI[K_1,K_2] : \{a_1,a_2,a_3,a_4\}, \{a_{11}\}, \{a_{11},a_{12}\}$

$IM[K_1]$:

$C[K_1] : \{a_1, a_2\}, \{a_5, a_9\},\{a_6, a_9\}$

$CI[K_1] : a_5, a_6, a_{11}, \{a_5, a_9\}, \{a_6,a_9\}$

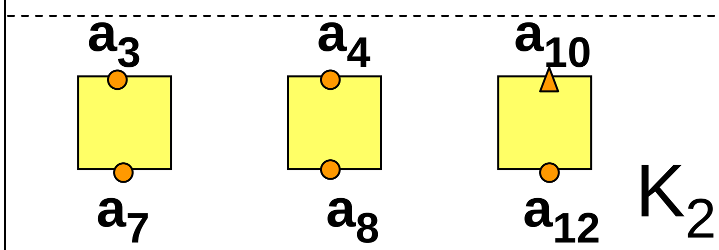


||

$IM[K_2]$:

$C[K_2] : \{a_3, a_4\}, \{a_7, a_{10}\}, \{a_8, a_{10}\}$

$CI[K_2] : a_{10}, \{a_7, a_{10}\}, \{a_8,a_{10}\}$



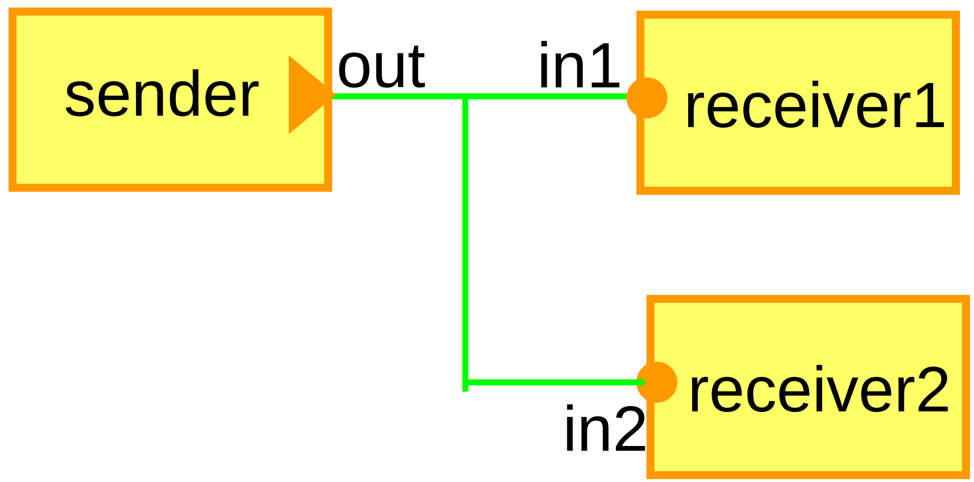
$IM[K_1 \cup K_2] = IM[K_1] \otimes IM[K_2] \otimes IM[K_1,K_2]$

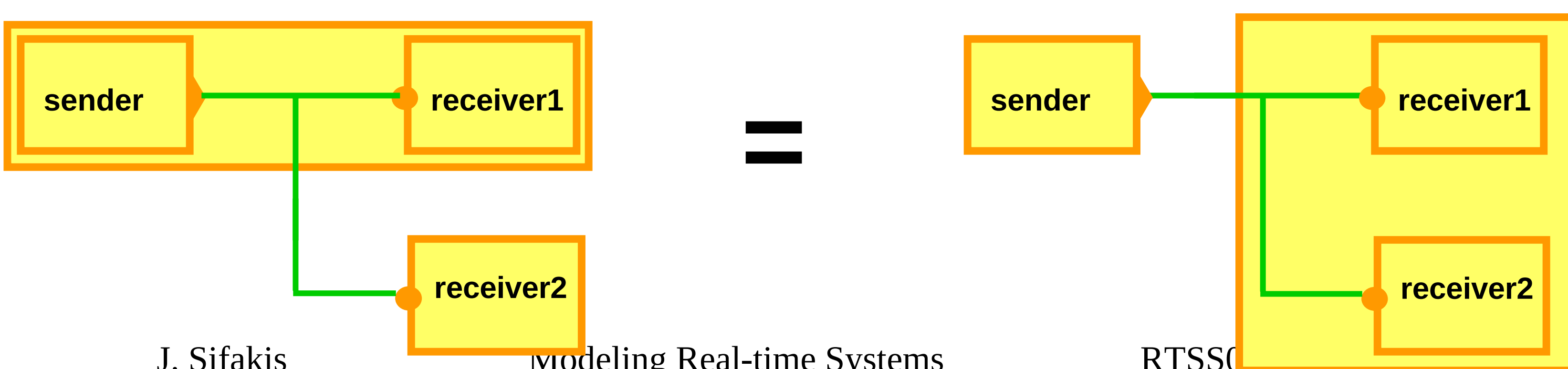
$C[K_1 \cup K_2] = \max\{C[K_1], C[K_2], C[K_1, K_2] \}$

$CI[K_1 \cup K_2] = \min\{CI[K_1], CI[K_2], CI[K_1, K_2] \}$

Interaction models – Associativity

Composition is associative and commutative





Outline

Key Research issues

- Modeling Real-time systems
- From application SW to implementations
- Component-based construction

The modeling framework

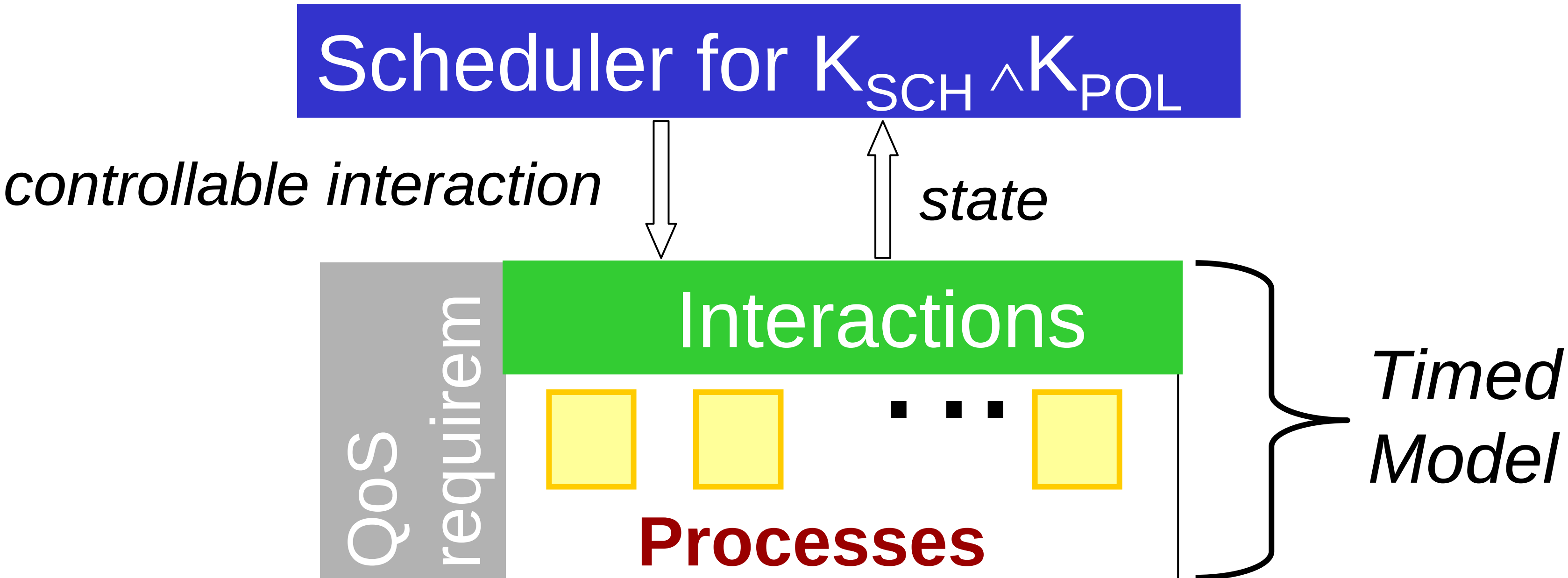
- Principles
- Interaction models
- Scheduler modeling
- Timed models with priorities

Discussion

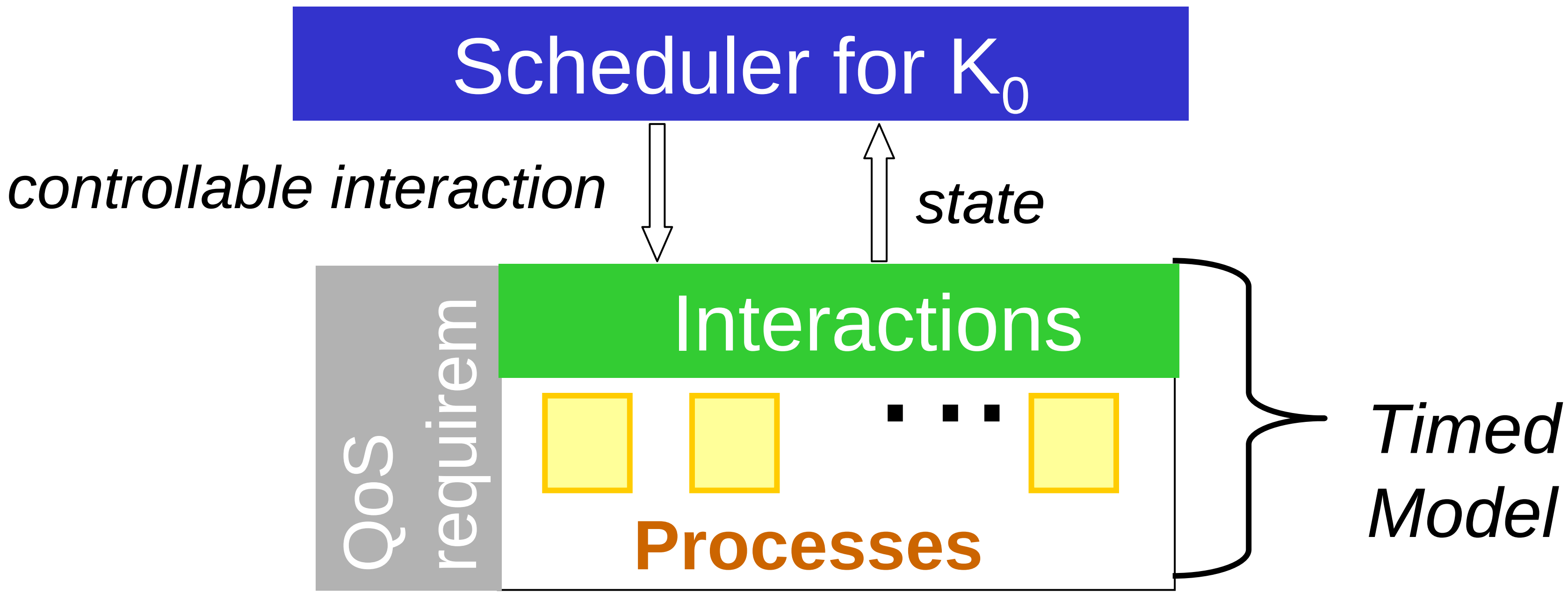
Schedulers - Their role

A scheduler is a controller that restricts access to resources by triggering controllable interactions so as to respect timing constraints (state predicates) $K_0 = K_{SCH} \wedge K_{POL}$

- K_{SCH} timing constraints on process actions
- K_{POL} scheduling policy



Schedulers - Control invariants

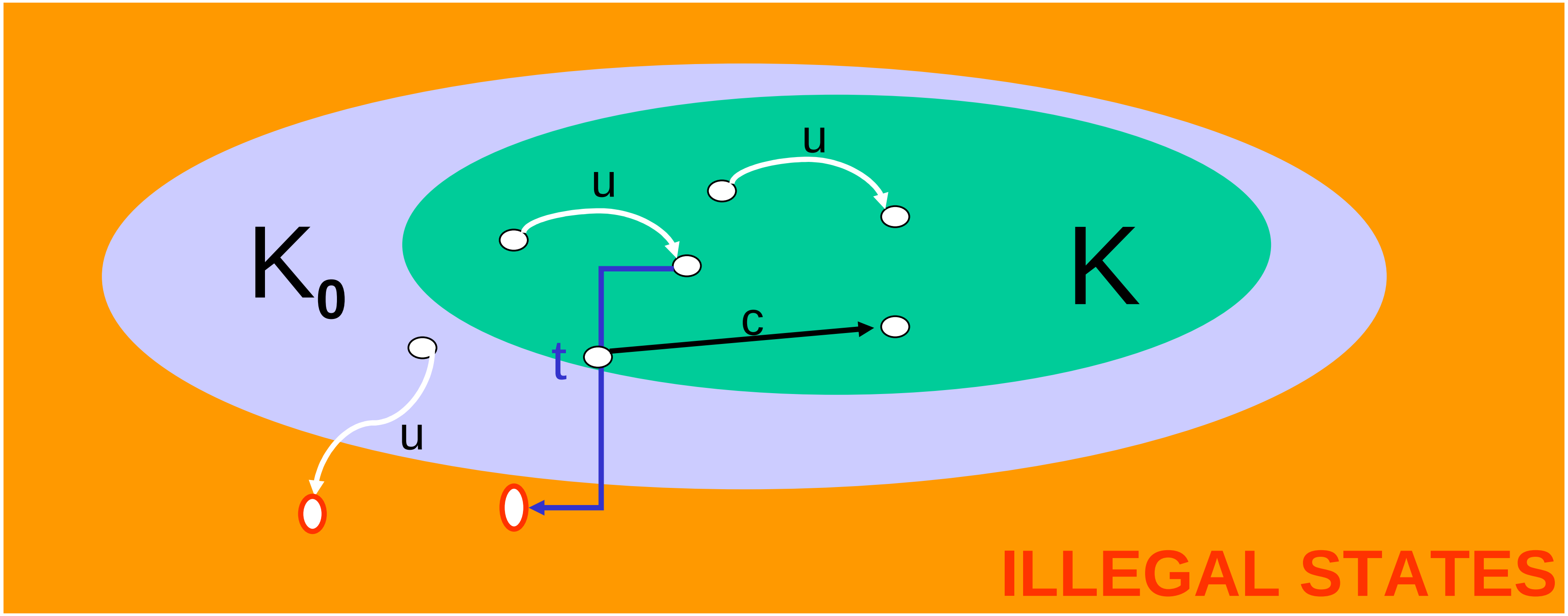


A scheduler for K_0 can be defined by a **control invariant** $K \Rightarrow K_0$

- K contains only deadlock-free states for all processes
- K cannot be violated by uncontrollable interactions
- If from some state time progress can violate K, then there exists from this state controllable action preserving K

Schedulers - control invariants (2)

A control invariant $K \Rightarrow K_0$



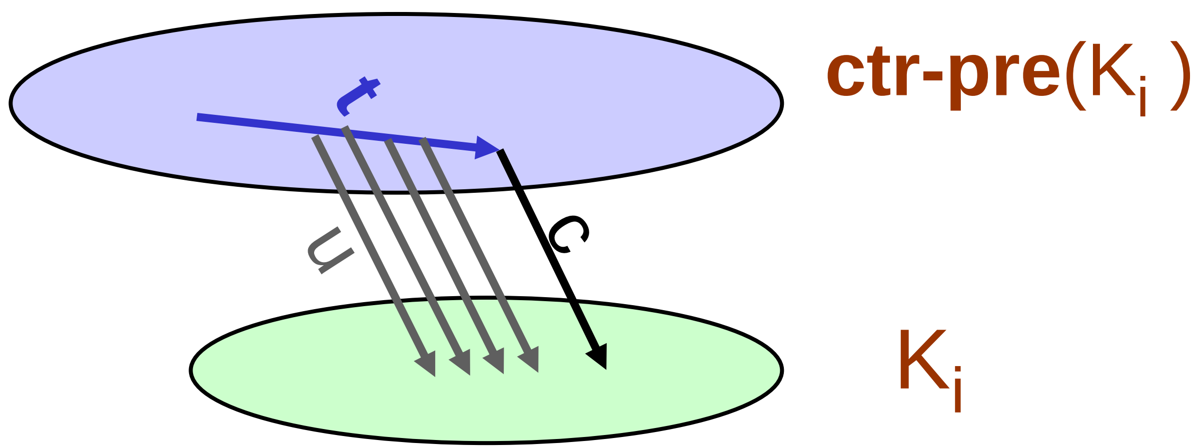
The effect of the scheduler				
Initial system	S	⟶	S/K	Scheduled system
Guard of controllable interaction a	g	⟶	$g \wedge K \wedge wp_a(K)$	

Schedulers - Control invariants : some results

For any system S and timing constraints K_0

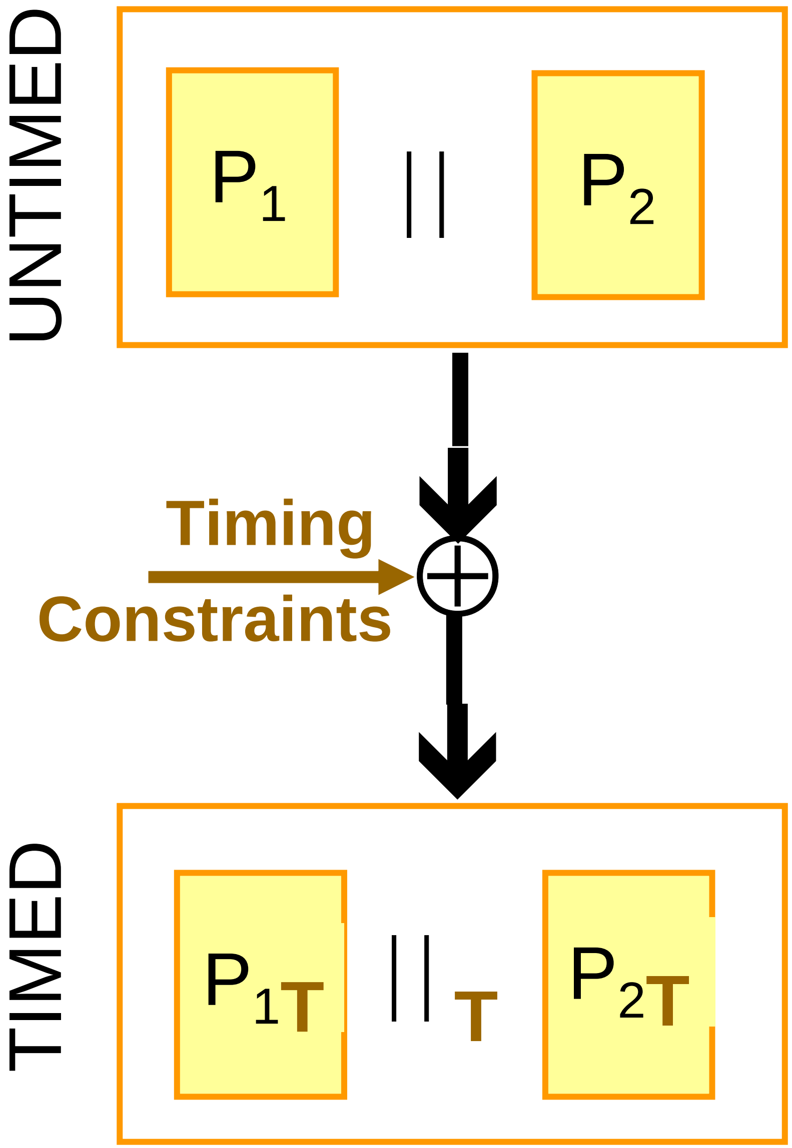
- There exists a maximal control invariant K, $K \Rightarrow K_0$
- K can be computed as the result of a synthesis semi-algorithm

$$\mathrm{SYNTH}(S,K_0) = \lim_I \{K_i\} \text{ where } K_{I+1} = K_I \wedge \textbf{ctr-pre}\,(K_I)$$



- The conjunction of control invariants is not a control invariant - conditions for composability of schedulers

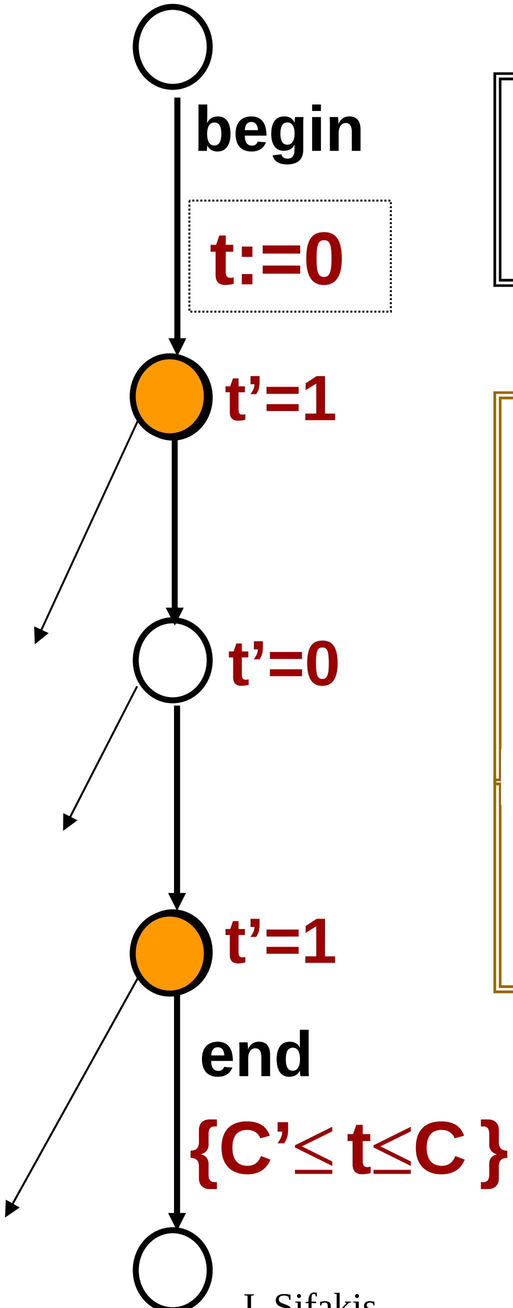
Building timed models



Methodology :

- Avoid over-specification which may lead to inconsistency
- Make explicit all the consequences of the constraints on actions (completeness)
- Define $||_T$ so as to preserve properties such as well-timedness, and deadlock-freedom

Building timed models



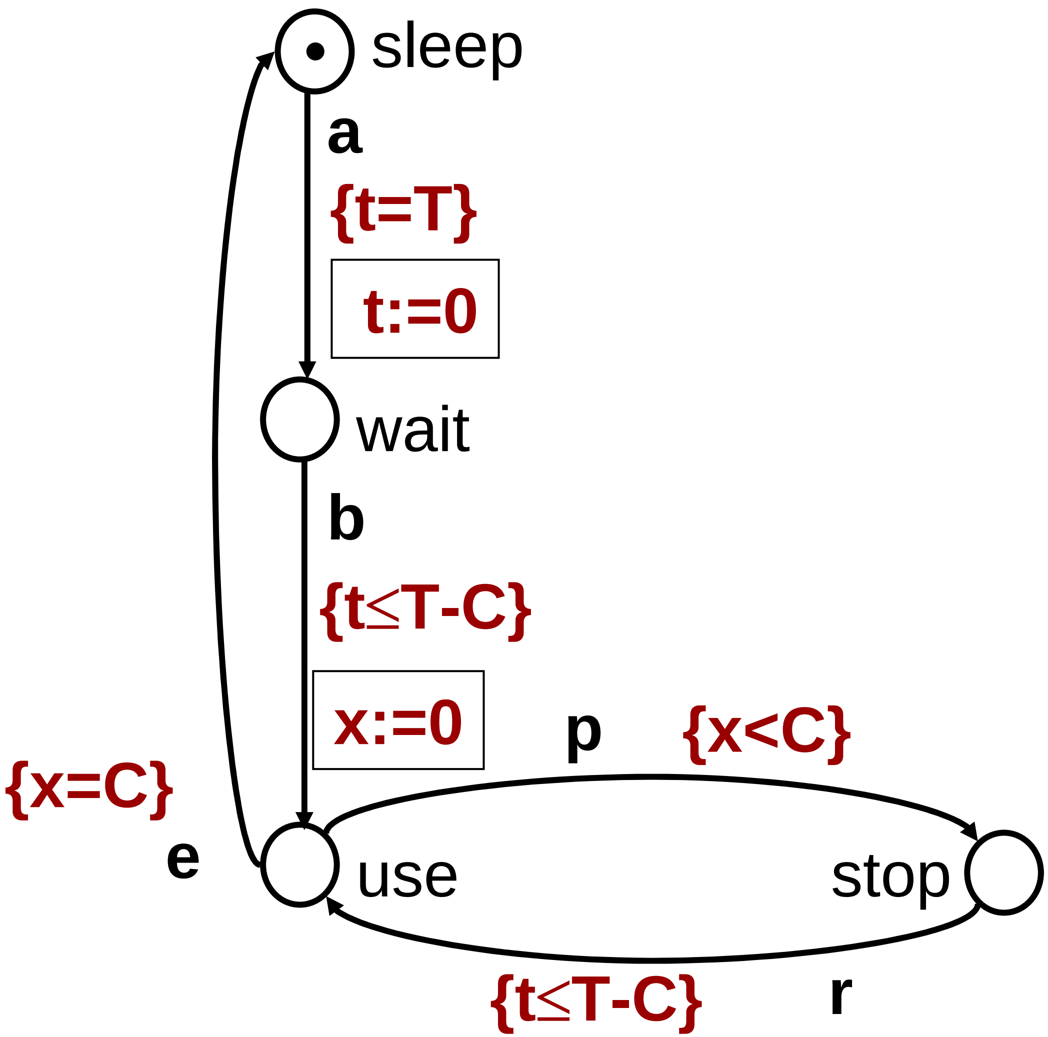
Automata: set of labeled transition on a set of actions

Timers: real-valued variables that can

- be reset (started) and tested at transitions
- increase (derivative =1) or remain unchanged at states (derivative =0)

Building timed models - example

A periodic process of period **T** and completion time **C**



Actions

a: arrive	(u)
b: begin	(c)
e: end	(u)
p: preempt	(c)
r: resume	(c)

t'=x'=1 at all states except stop **(x'=0)**

Scheduler specification : $\mathbf{K_{SCH}}$

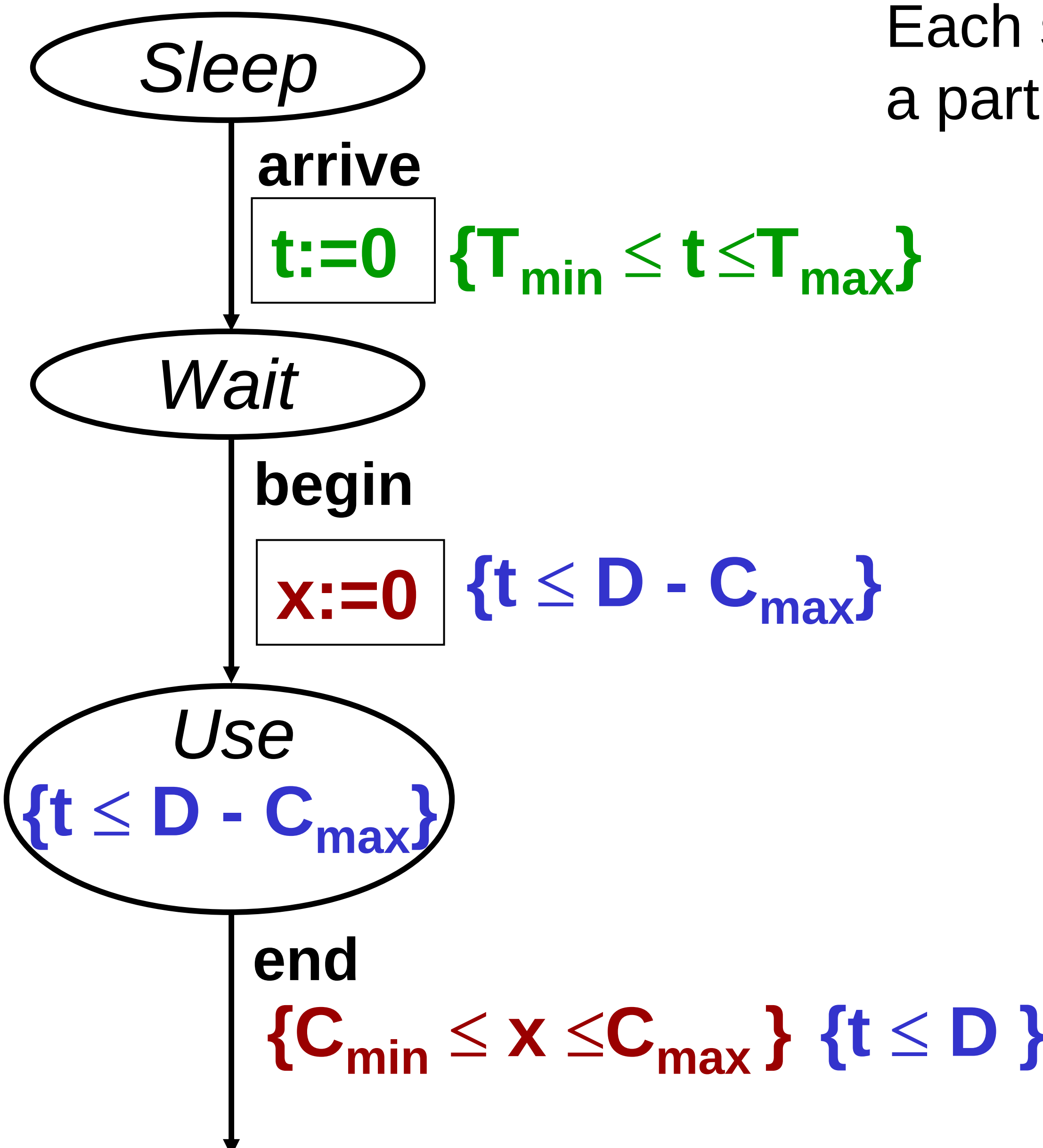
The scheduling constraint $\mathbf{K_{SCH}}$ relates timing constraints of 3 different kinds

- *from the **execution platform** e.g. execution times, latency times*

- *from the **external environment** about arrival times of triggering events e.g. periodic tasks*

- ***user requirements** e.g. QoS, which are timing constraints relating events of the real-time system and events of its environment e.g. deadlines, jitter*

Scheduler specification : $\mathbf{K_{SCH}}$



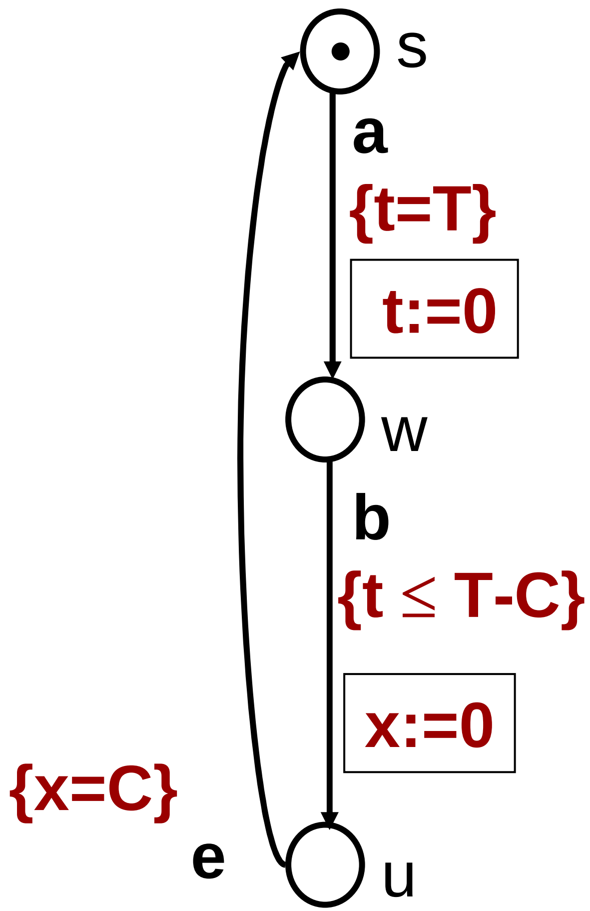
Each shared resource induces a partition {Sleep, Wait, Use}.

Arrival time (t)

Completion time (x)

Deadline D

Scheduler specification: K_{SCH}



$K_{SCH} = \bigwedge_i K^i_{SCH}$

where K^i_{SCH} *expresses the property that no timing constraint is violated in process i.*

For timelock-free process models with bounded guards schedulability boils down to deadlock-freedom of processes

$$K_{SCH} = s \wedge (t \leq T) \vee w \wedge (t \leq T\text{-}C) \vee u \wedge (x \leq C)$$

Scheduler specification : K_{POL}

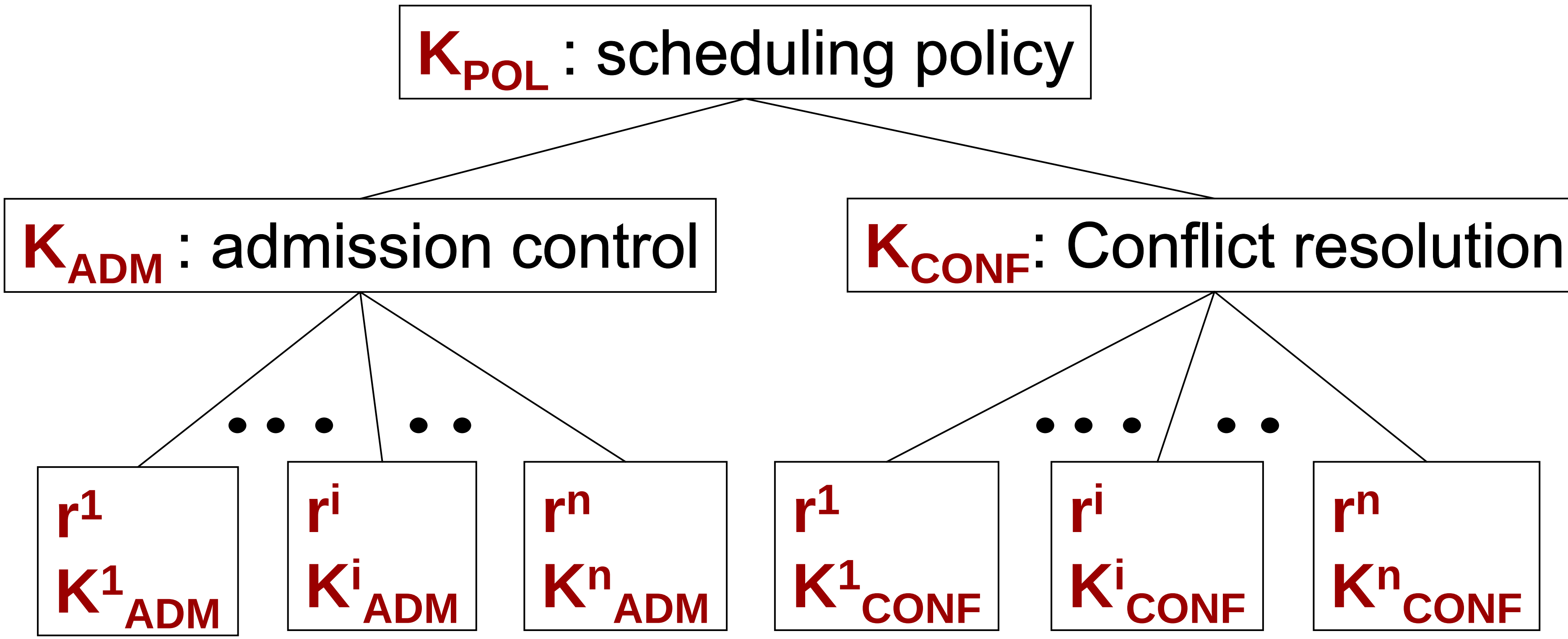
K_{POL} is the conjunction of scheduling policies for the set R of shared resources

$K_{POL} = \bigwedge_{r \in R} K^r_{POL}$ where $K^r_{POL} = K^r_{CONF} \wedge K^r_{ADM}$

- K^r_{CONF} says how conflicts for the acquisition of resource r are resolved e.g. EDF, RMS, LLF

- K^r_{ADM} says which requests for r are considered by the scheduler at a state e.g. masking

Scheduler specification: K_{POL}



Scheduler specification – K_{POL} : example

K_{POL} for the Priority Ceiling Protocol

Admission control: *"Process P is eligible for resource r if the current priority of P is higher than the ceiling priority of any resource allocated to a process other than P"*

Conflict resolution: " *The CPU is allocated to the process with the highest current priority"*

Outline

Key Research issues

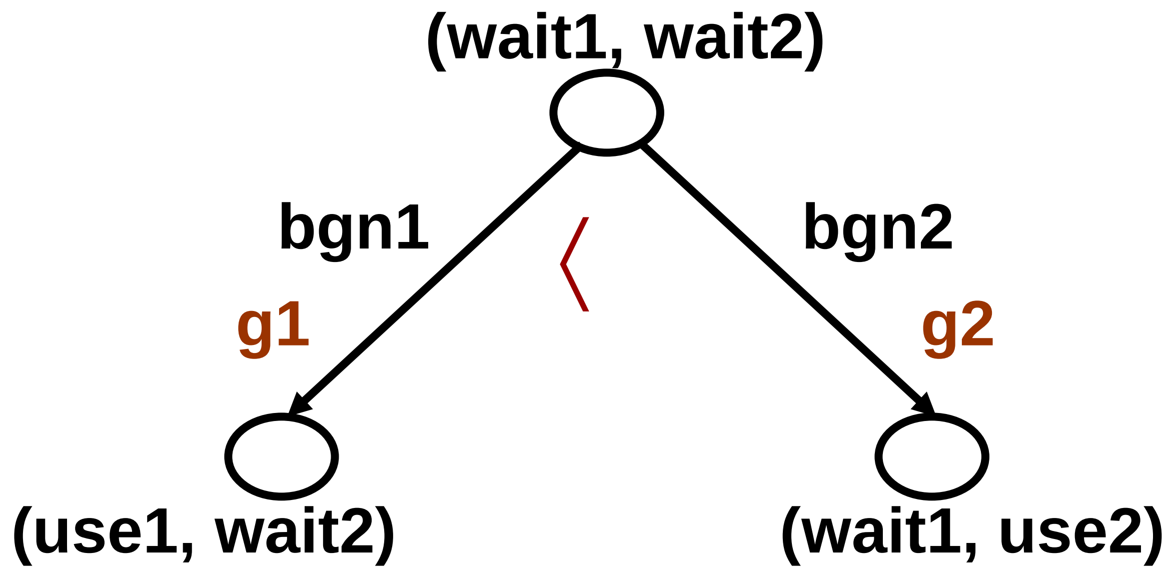
- Modeling Real-time systems
- From application SW to implementations
- Component-based construction

The modeling framework

- Principles
- Interaction models
- Scheduler modeling
- Timed models with priorities

Discussion

Timed models with priorities

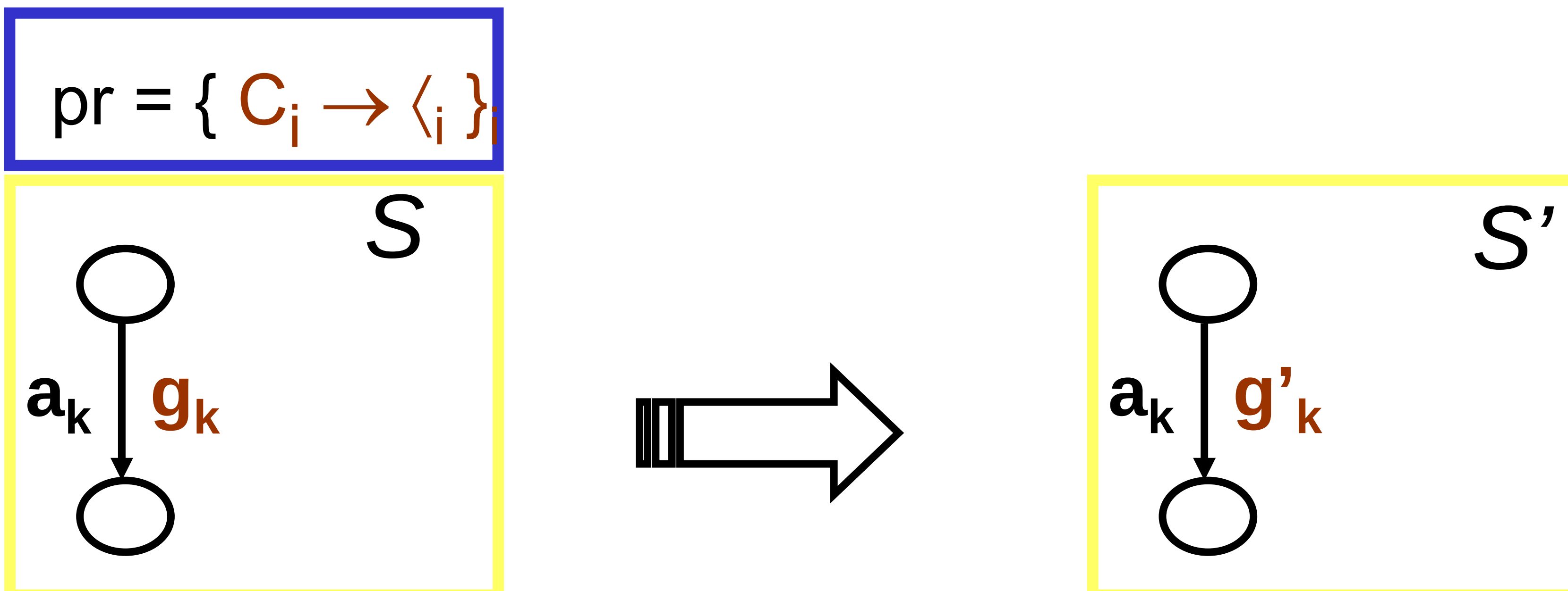


Priority rule	Strengthened guard of bgn1
true $\rightarrow$ bgn1 〈 bgn2	$g1' = g1 \wedge \neg g2$
C $\rightarrow$ bgn1 〈 bgn2	$g1' = g1 \wedge \neg(C \wedge g2)$

Timed models with priorities

A priority order is a strict partial order, $\langle \subseteq A^c \times A$

A set of priority rules, pr = $\{ C_i \rightarrow \langle_i \}_i$ where $\{C_i\}_i$ is a set of disjoint state predicates



$$g'_k = g_k \wedge \bigwedge_{C \rightarrow \langle \in pr} \left(C \Rightarrow \bigwedge_{ak \langle ai} \neg g_i \right)$$

Scheduling and Priorities - results

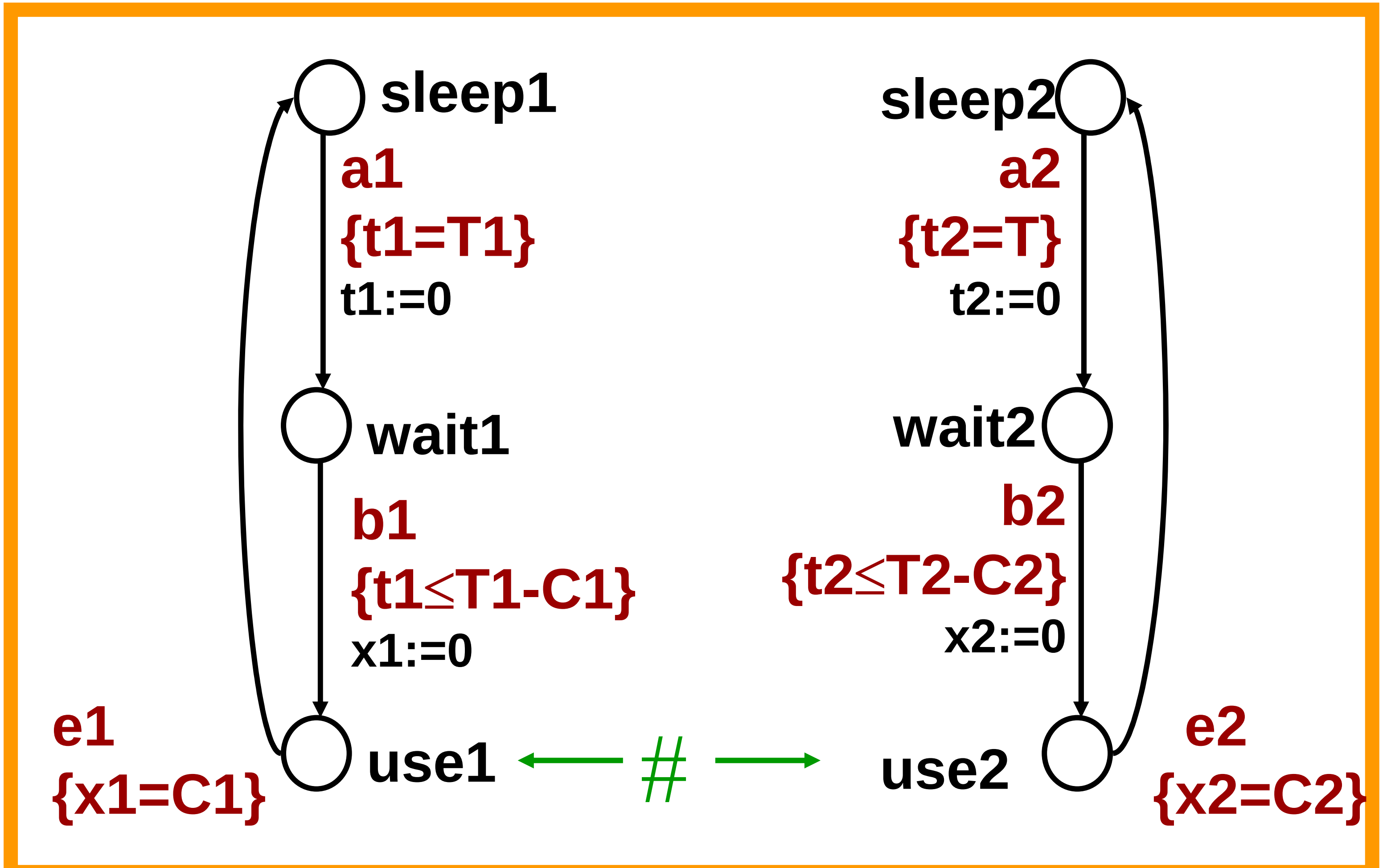
*If **K** is a constraint characterizing a set of deadlock-free states of **S** then there exists a set of priority rules **pr** such that **(S,pr)** preserves **K***

*For any control invariant K of **S** there exists a set of dynamic priority rules **pr** such that the scheduled system **S/K = (S,pr)***

Any feasible scheduling policy K_{POL} induces a restriction that can be described by dynamic priorities

Timed models with priorities: FIFO policy

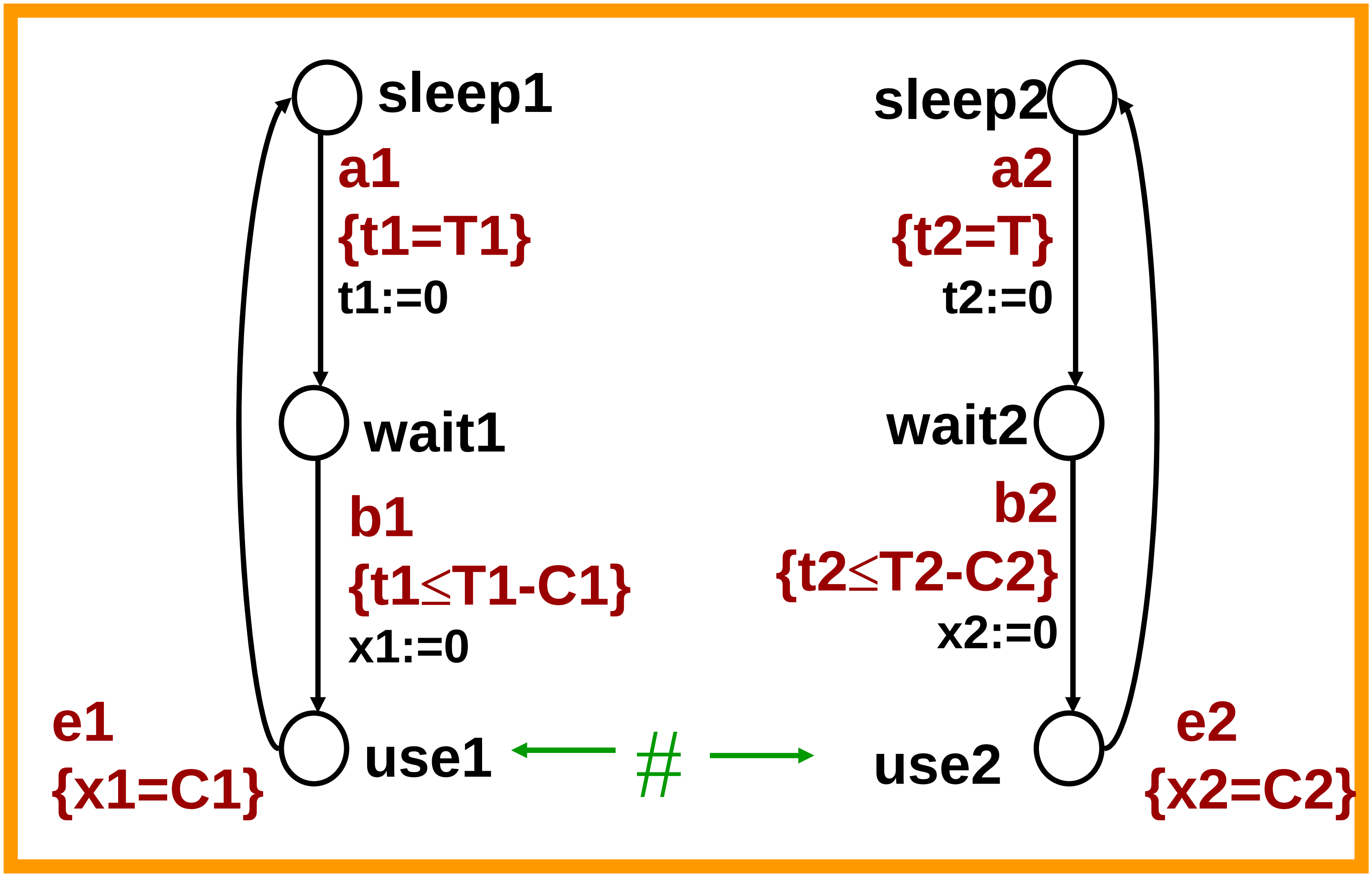
$t1 \leq t2 \rightarrow b1 \langle b2$ **$t2 \leq t1 \rightarrow b2 \langle b1$**



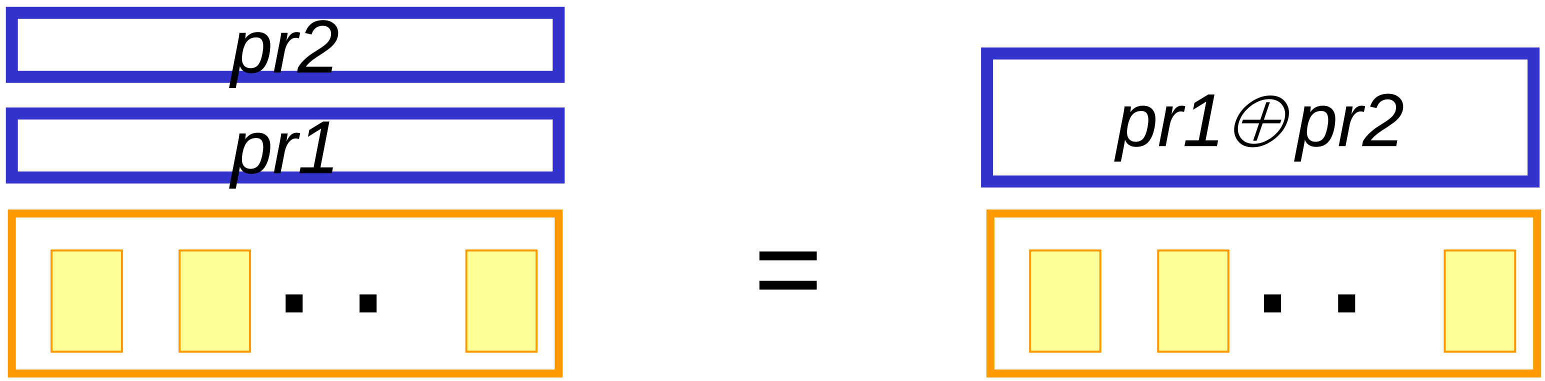
Timed models with priorities : Least Laxity First policy

$L1 \leq L2 \rightarrow b2 \langle b1 \qquad L2 \leq L1 \rightarrow b1 \langle b2$

where Li =Ti-Ci-ti is the laxity of process i



Timed models with priorities: composition of priorities



pr1⊕ pr2 is the least priority order containing pr1∪pr2

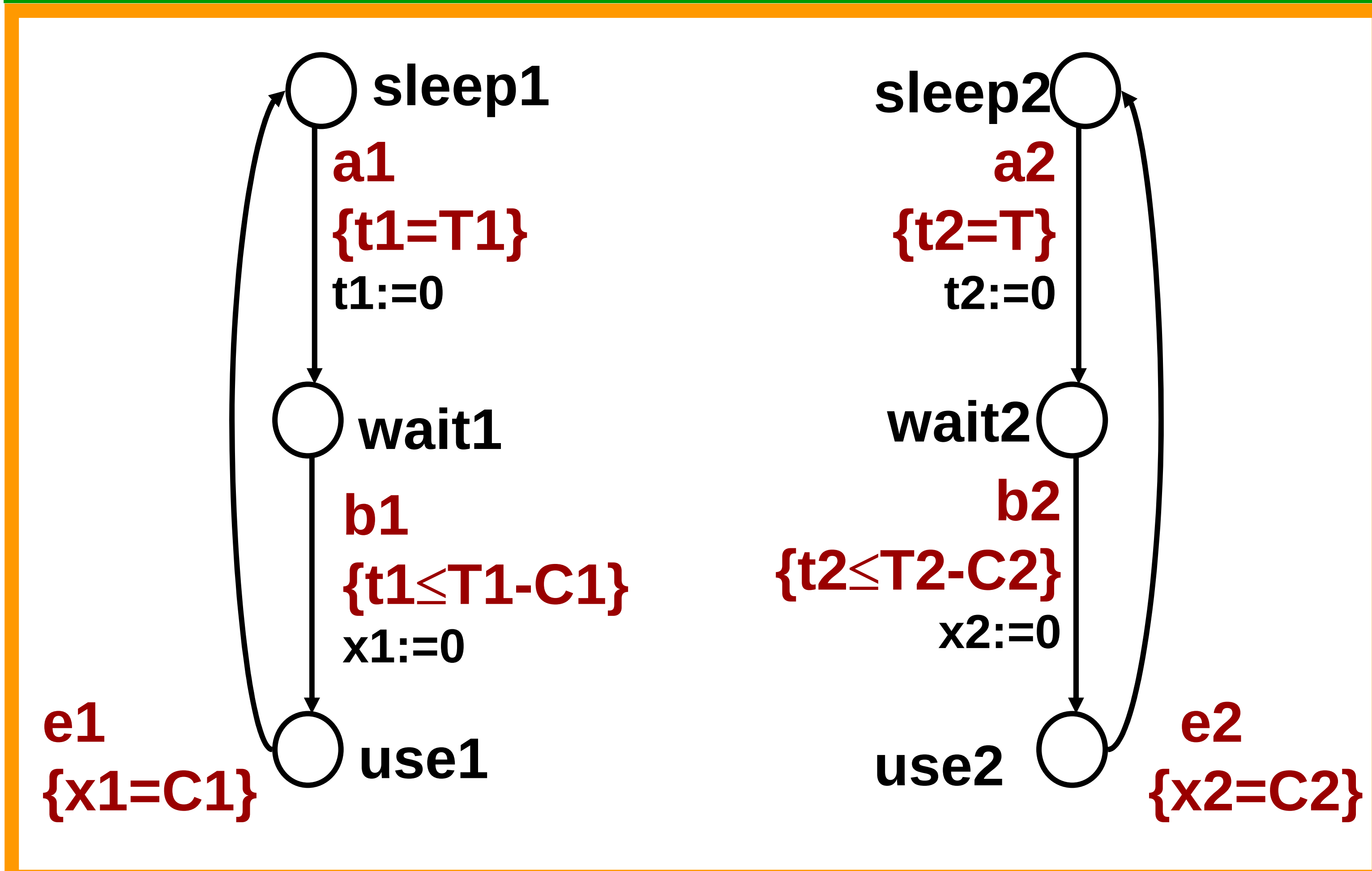
Results :

- The operation ⊕ is partial, associative and commutative
- Sufficient conditions for deadlock-freedom and liveness

Timed models with priorities: mutual exclusion + FIFO

$t1 \leq t2 \rightarrow b1 \langle b2$ $\qquad$ $t2 \leq t1 \rightarrow b2 \langle b1$

$true \rightarrow b1 \langle e2$ $\qquad$ $true \rightarrow b2 \langle e1$



Timed models : Fixed priority preemptive scheduling

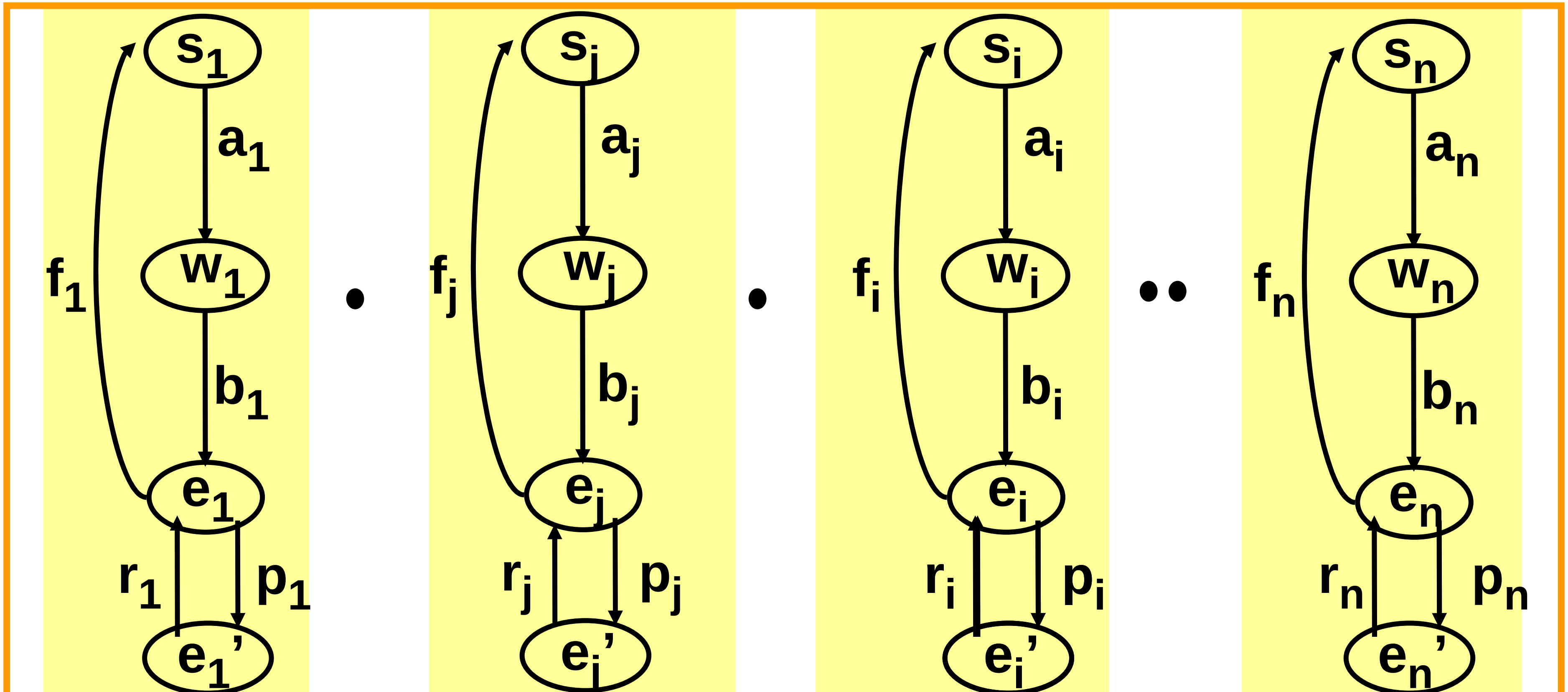
Scheduling policy For $n \geq l > j \geq 1$

$\mathbf{b_i \langle b_j,\ r_i \langle r_j,\ r_i \langle b_j,\ b_i \langle r_j}$ (access to resource)

$\mathbf{\{b_i, p_j\} \langle f_j,\ \{r_i, p_j\} \langle f_j}$ (non pre-emption by lower pty tasks)

Interaction model For $n \geq l > j \geq 1$

$\mathbf{\{b_j, p_i\}, \{r_j, p_i\} \in C}$ $\mathbf{a_i, f_i, b_i \in Cl}$



Key Research issues

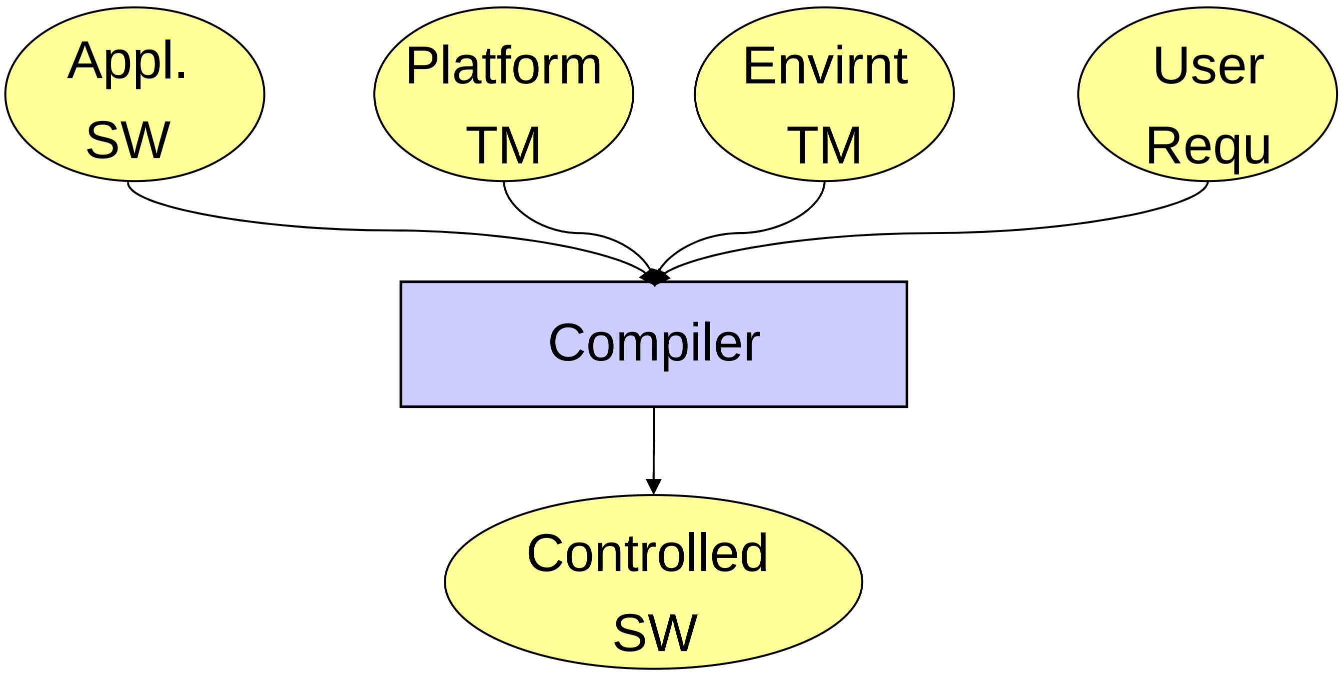
- Modeling Real-time systems
- From application SW to implementations
- Component-based construction

The modeling framework

- Principles
- Interaction models
- Scheduler modeling
- Timed models with priorities

Discussion

Discussion : SW is the model



- Specific and tractable methodology relying on a **minimal set of constructs and principles** e.g. interaction models + priorities

  Layering $\Rightarrow$ separation of concerns $\Rightarrow$ incremental description

- Focus on specific construction principles and rules to ensure correctness constructively, especially for **safety** and **deadlock-freedom**

Discussion : A framework for scheduling

- The controller synthesis paradigm is a basis for a general framework for scheduler specification ($K_{SCH} \wedge K_{POL}$) and design (control invariants)

- Scheduling theory studies sufficient conditions guaranteeing K_{SCH} for particular scheduling policies K_{POL} and interaction models (architectures)

- Extending the model-based approach to encompass scheduler modeling and design

- Scheduler design methodology based on model-checking techniques – scheduling policies can be used to simplify the synthesis problem

Discussion : Dynamic priorities

That’s all you need!

- they allow straightforward modeling of
  - urgency (priority of actions over time progress)
  - scheduling policies
  - schedulers (control invariants)

- run to completion and synchronous execution can be modeled by assigning priorities to threads (implemented in the IF toolset)

- Composability and compositionality results

Discussion : some papers

THEORY

- ‘’ Scheduler modeling based on the controller synthesis paradigm’’ Journal of Real-time Systems, Vol. 23, pp.55-84, 2002
- ‘’A Framework for Scheduler Synthesis’’ RTSS 1999
- ‘’Component-based construction of deadlock-free systems’’, FSTTCS03, LNCS 2194,
- ‘’ Priority Systems’’ Proceedings of FMCO’03, LNCS 3188
- ’’Composition for component-based modeling’’, FMCO 02, LNCS 2852

APPLICATIONS

S. Yovine et al. ‘’A methodology and tool support for generating scheduled native code for real-time Java applications’’ EmSoft 03

- ‘’TAXYS: a tool for the developpment and verification real-time embedded systems’’ CAV'01. LNCS 2102.
- M. Bozga, S. Graf, Il. Ober, Iul. Ober, J. Sifakis "The IF Toolset" Formal Methods for the Design of Real-Time Systems, Sept 2004, LNCS 3185