A Framework for Component-based Construction

Workshop on Distributed Embedded Systems
Leiden, 21-24 November 2005

Joseph Sifakis

VERIMAG and ARTIST2 NoE

sifakis@imag.fr

In collaboration with

Ananda Basu, Marius Bozga

Motivation

Develop a rigorous and general basis for architecture modeling and implementation:

- Study the concept of architecture as a means to organize computation (behavior, interaction, control)
- Define a meta-model for real-time architectures, encompassing specific styles, paradigms, e.g. modeling
  - Synchronous and asynchronous execution
  - Event driven and state driven interaction
  - Distributed computation
  - Architecture styles such as client-server, blackboard architecture
- Provide automated support for component integration and generation of glue code meeting given requirements

Overview

- Component-based construction – the notion of glue
- Interaction Models
- Priorities
- The BIP framework
- Discussion

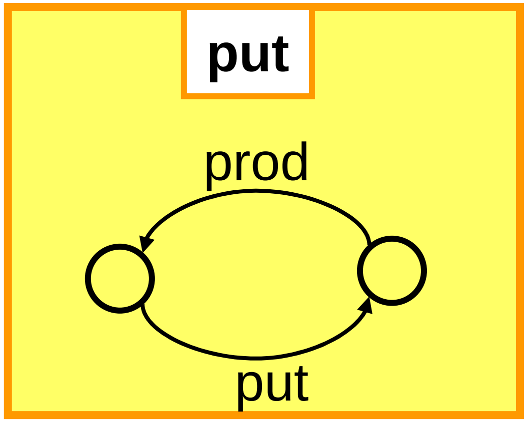
Component-based construction - components

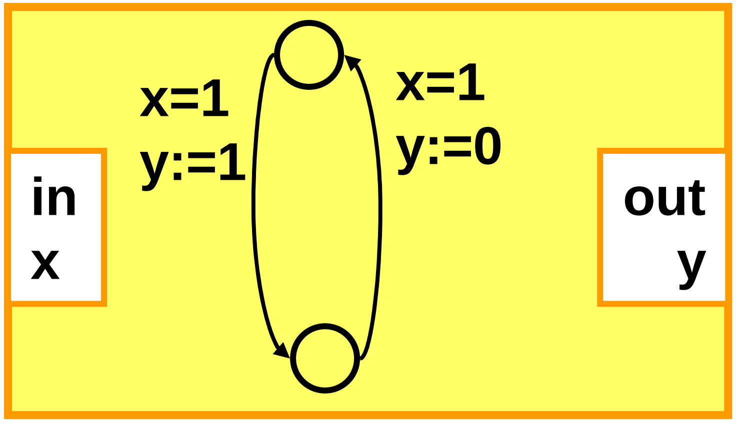
Build systems by **composition of components**

Atomic components are building blocks composed of **behavior** and **interface**

- **Behavior** is a transition system

- **Interface** hides irrelevant internal behavior and provides some adequate abstraction for composition and re-use, e.g. set of action names (ports) and associated variables

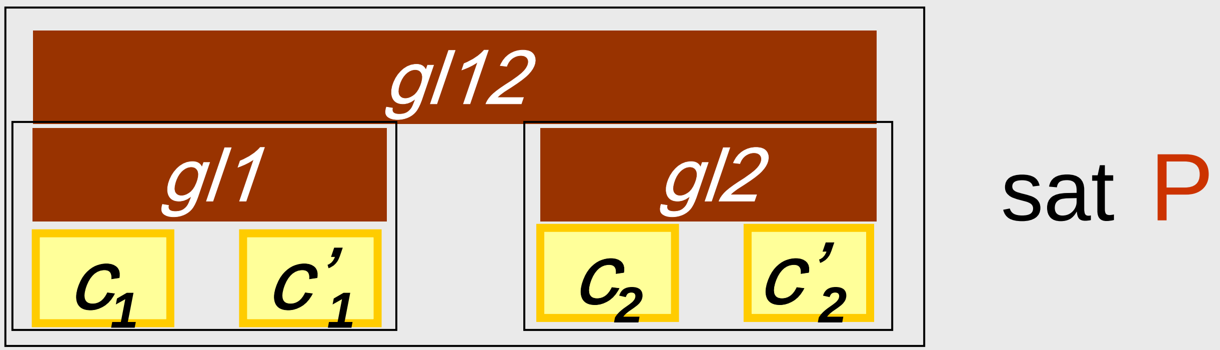




Component-based construction – formal framework

Pb:Build a component C satisfying a given property P, *from*

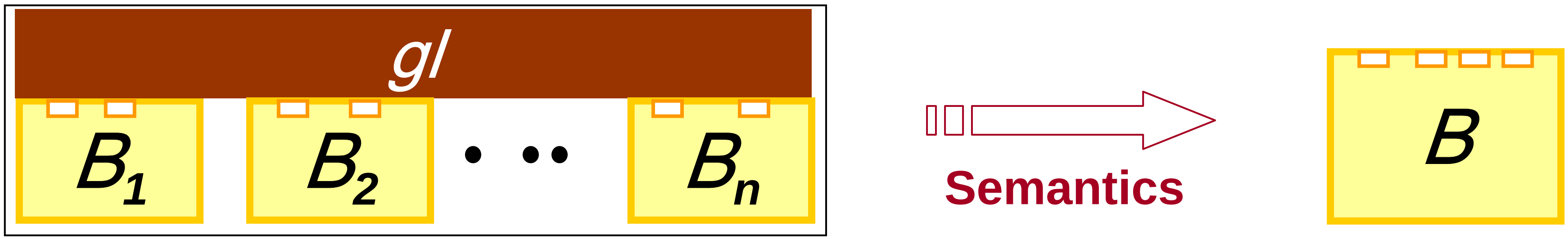
- $\mathcal{C}_0$ a set of atomic components
- $\mathcal{GL} = \{gl_1, \ldots, gl_i, \ldots\}$ a set of glue operators on components



- Components are terms of an algebra of terms ($\mathcal{C}$, $\cong$) generated from $\mathcal{C}_0$ by using operators from $\mathcal{GL}$
- $\cong$ is a congruence compatible with operational semantics

Component-based construction – formal framework

Glue operators transform sets of components into components



Glue operators

- model mechanisms used for communication and control such as protocols, controllers, buses
- restrict the behavior of their arguments, that is

$$gl(C_1, C_2, .., C_n)|A_1 \text{ refines } C_1$$

Component-based construction - requirements

Examples of existing frameworks:

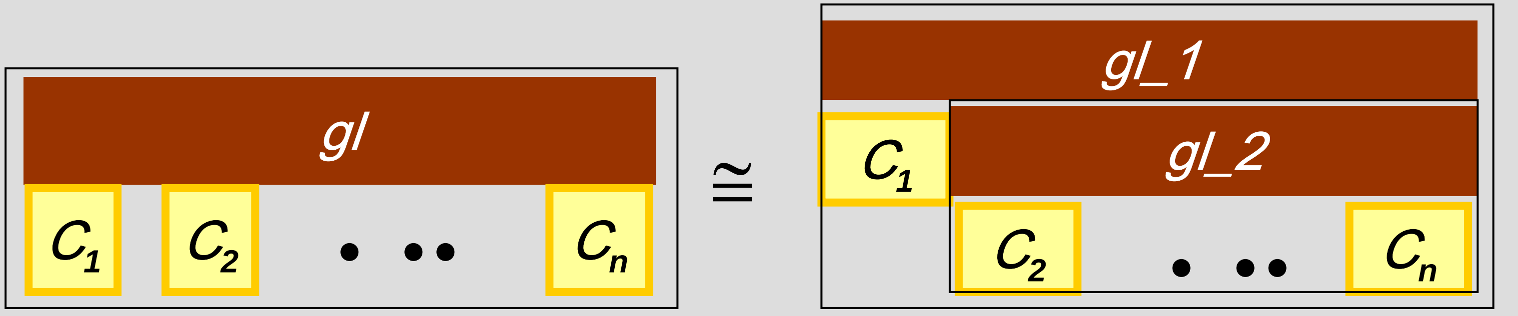
- Sequential functions with logical operators and delay operators for building circuits
- Process algebras
- Distributed algorithms define generic *gl* for a given property *P* e.g. token ring, clock synchronization …

Pb: Find a set of glue operators meeting the following requirements:

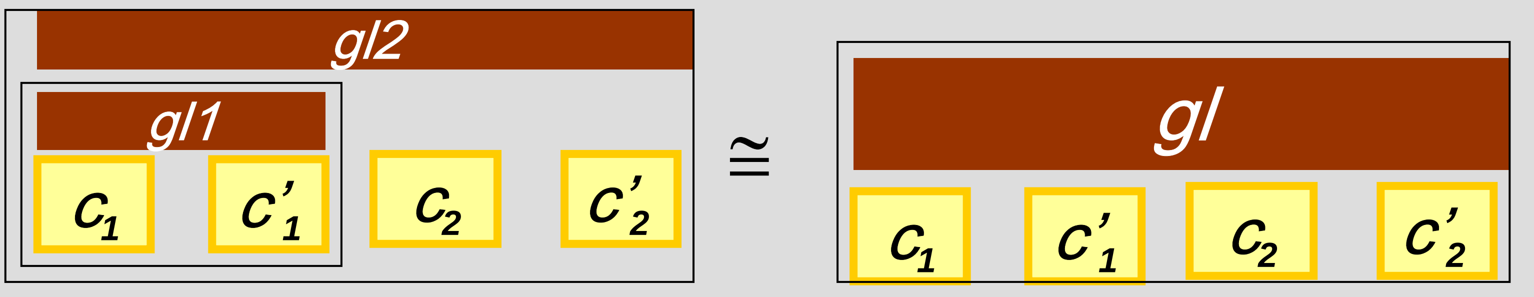
- Expressiveness (discussed later)
- Incremental description
- Correctness-by-construction

Component-based construction – incremental description

1. Decomposition of *gl*



2. Flattening of terms



Flattening can be achieved by introducing an idempotent operation $\oplus$ such that (GL, $\oplus$) is a commutative monoid and

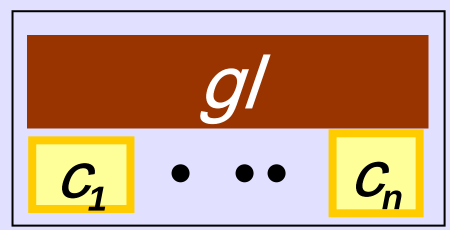
$$gl(gl'(C_1, C_2, .., C_n)) \simeq gl \oplus gl'(C_1, C_2, .., C_n)$$

Component-based construction - Correctness by construction : compositionality

Build correct systems

from correct components

c_i sat P_i implies $\forall gl\ \exists \widetilde{gl}$

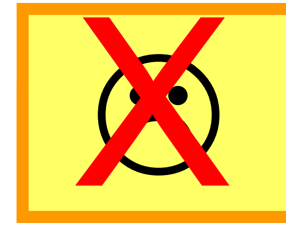


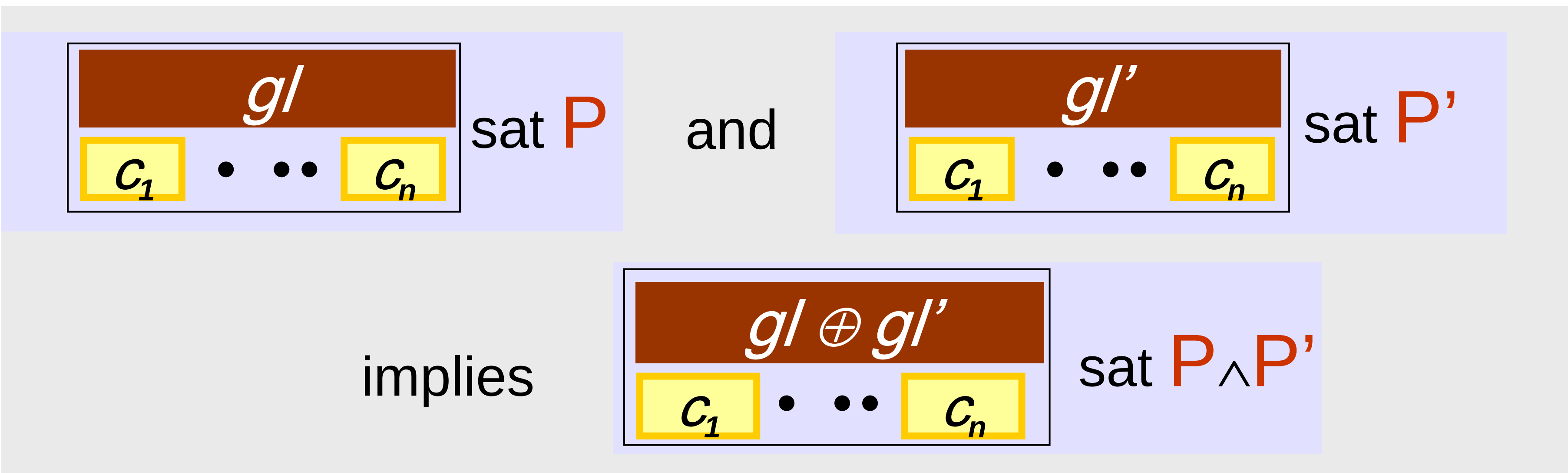
sat $\widetilde{gl}(P_1, .., P_n)$

We need compositionality results about preservation of progress properties such as deadlock-freedom and liveness.

Component-based construction - Correctness by construction : composability

Make the new without breaking the old

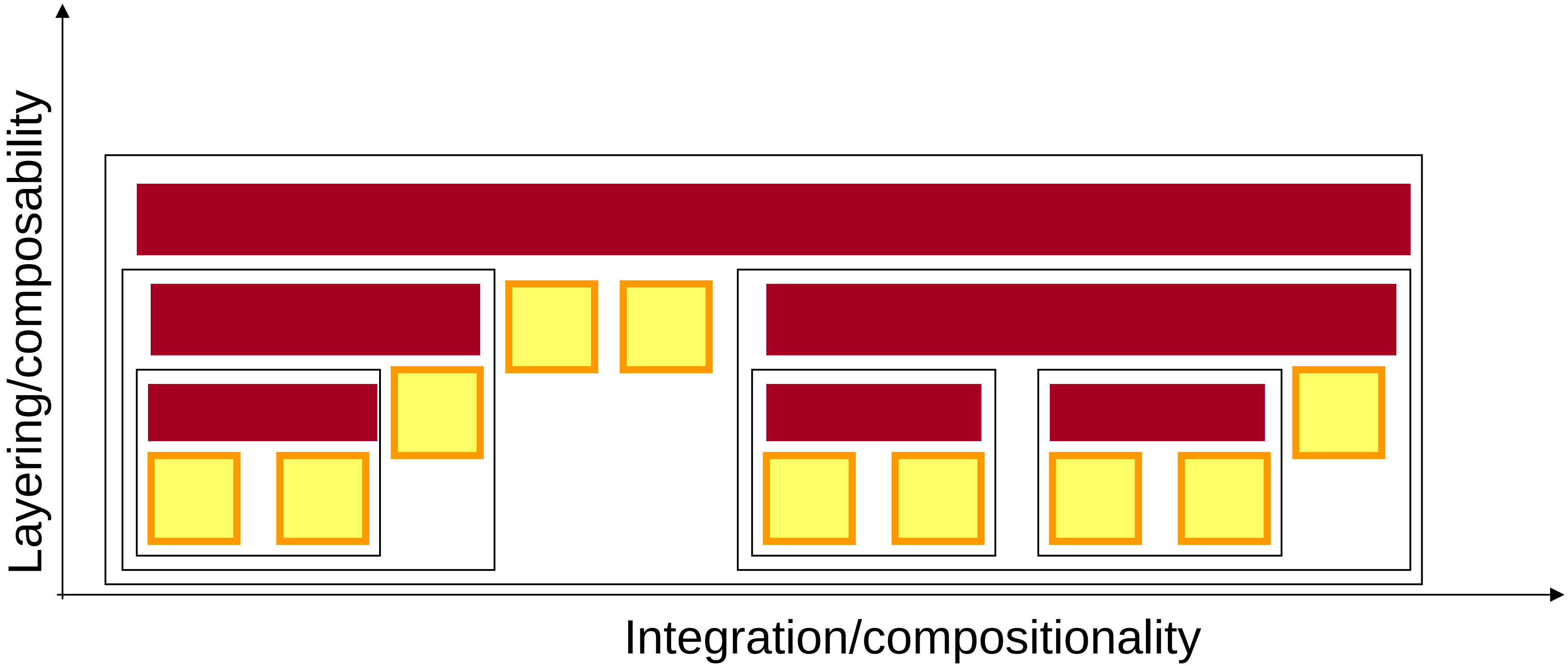




Property stability phenomena are poorly understood

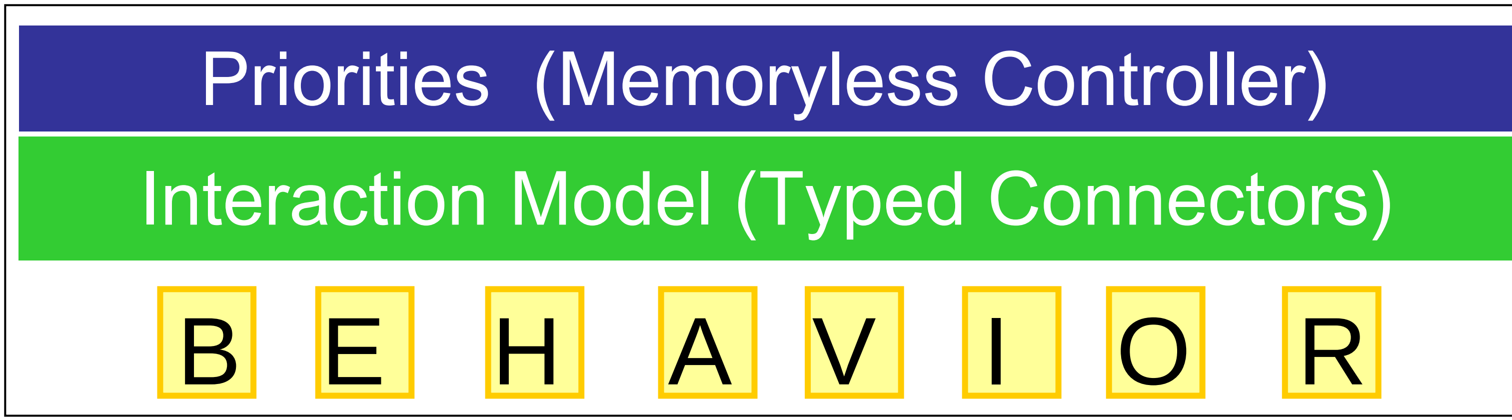
- *feature interaction*
- *non composability of scheduling algorithms*

Component-based construction - compositionality vs. composability

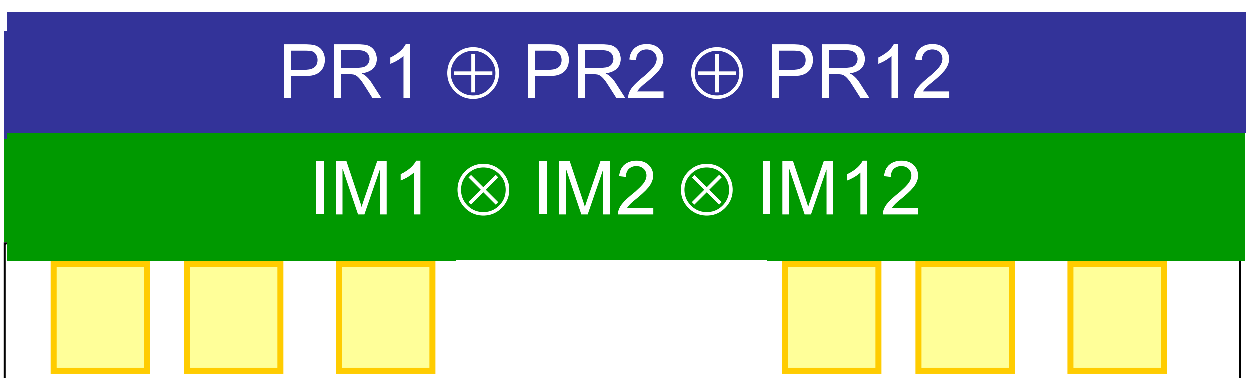


Component-based modeling – The BIP framework

Layered component model



Composition (incremental description)



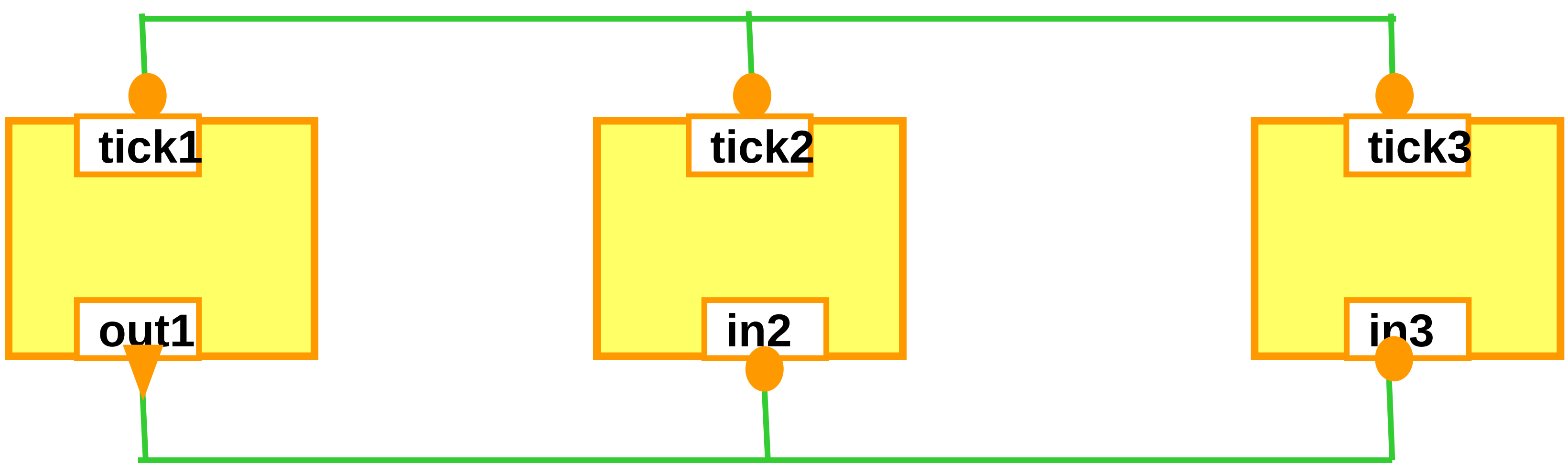
Overview

- Component-based construction – the notion of glue
- Interaction Models
- Priorities
- The BIP framework
- Discussion

Interaction models

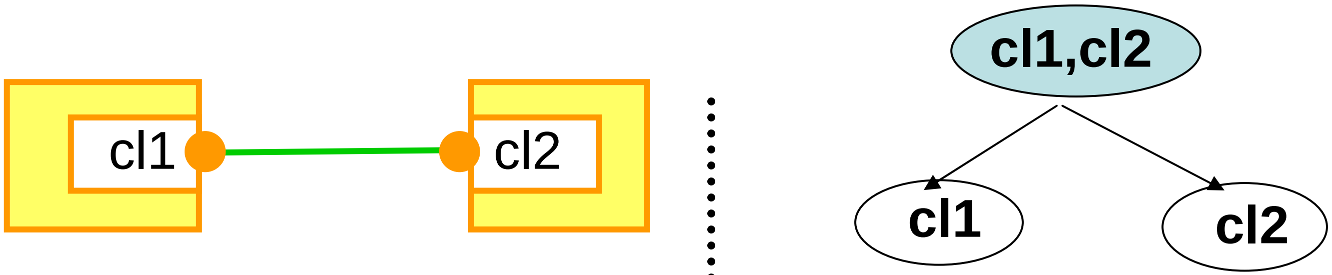
- **A connector** is a maximal set of compatible actions
- An **interaction** is a non empty subset of a connector

- Action types **(complete ▼, incomplete ●)** are used to define which subsets are interactions
- Interactions either contain some complete action or are maximal

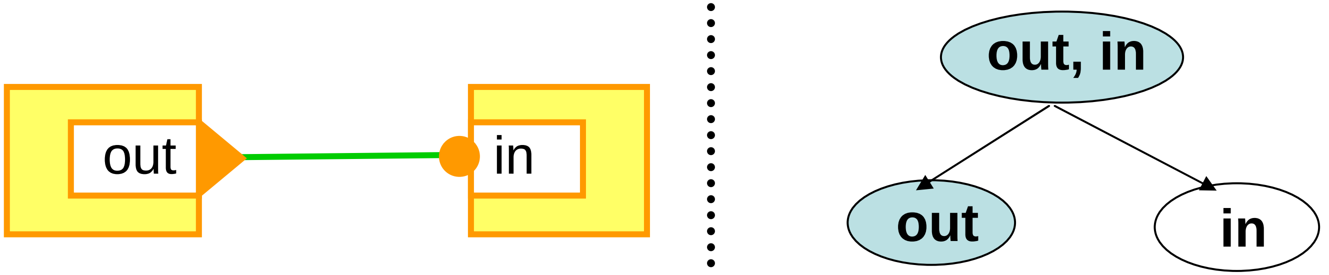


Interactions:
{tick1,tick2,tick3}, {out1}, {out1,in2}, {out1,in3}, {out1,in2, in3}

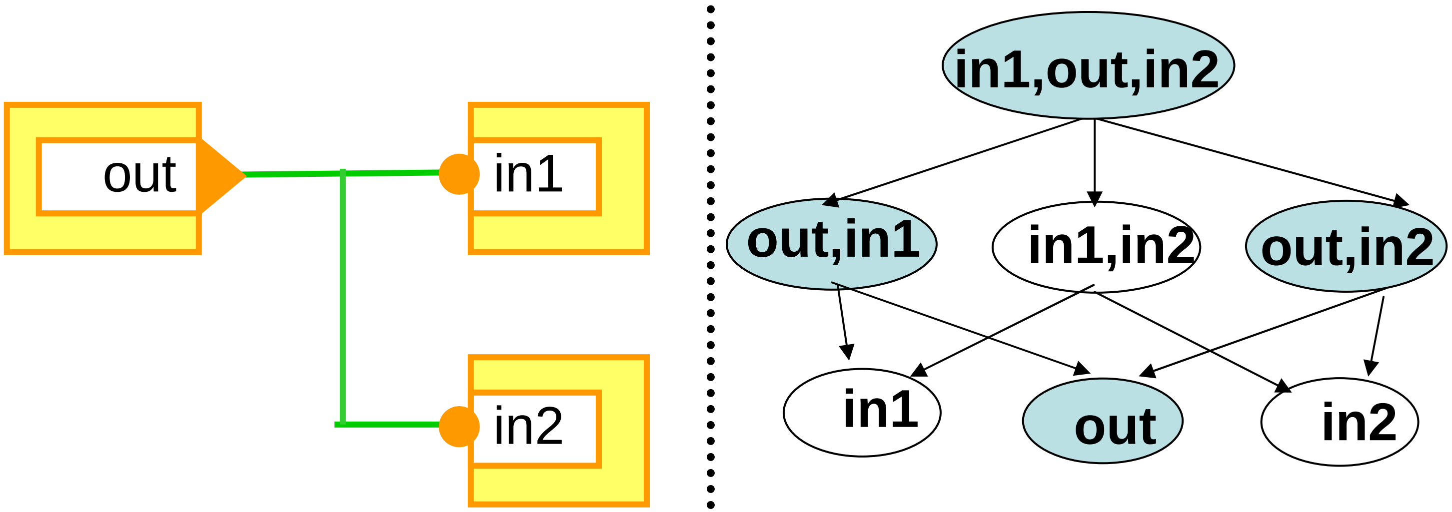
Interaction models - examples



CN:{cl1,cl2}
MCI: ∅

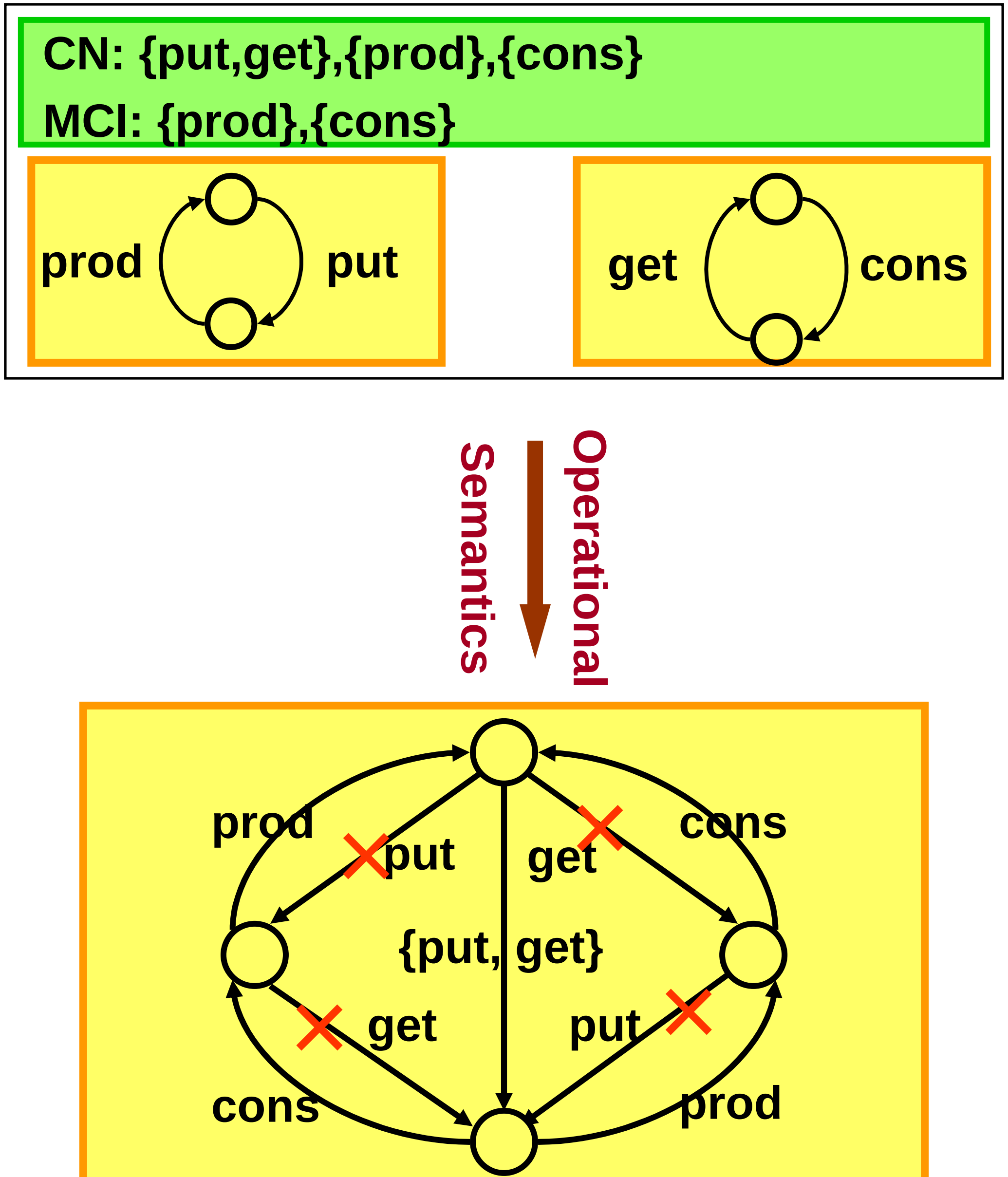


CN:{out,in}
MCI: {out}

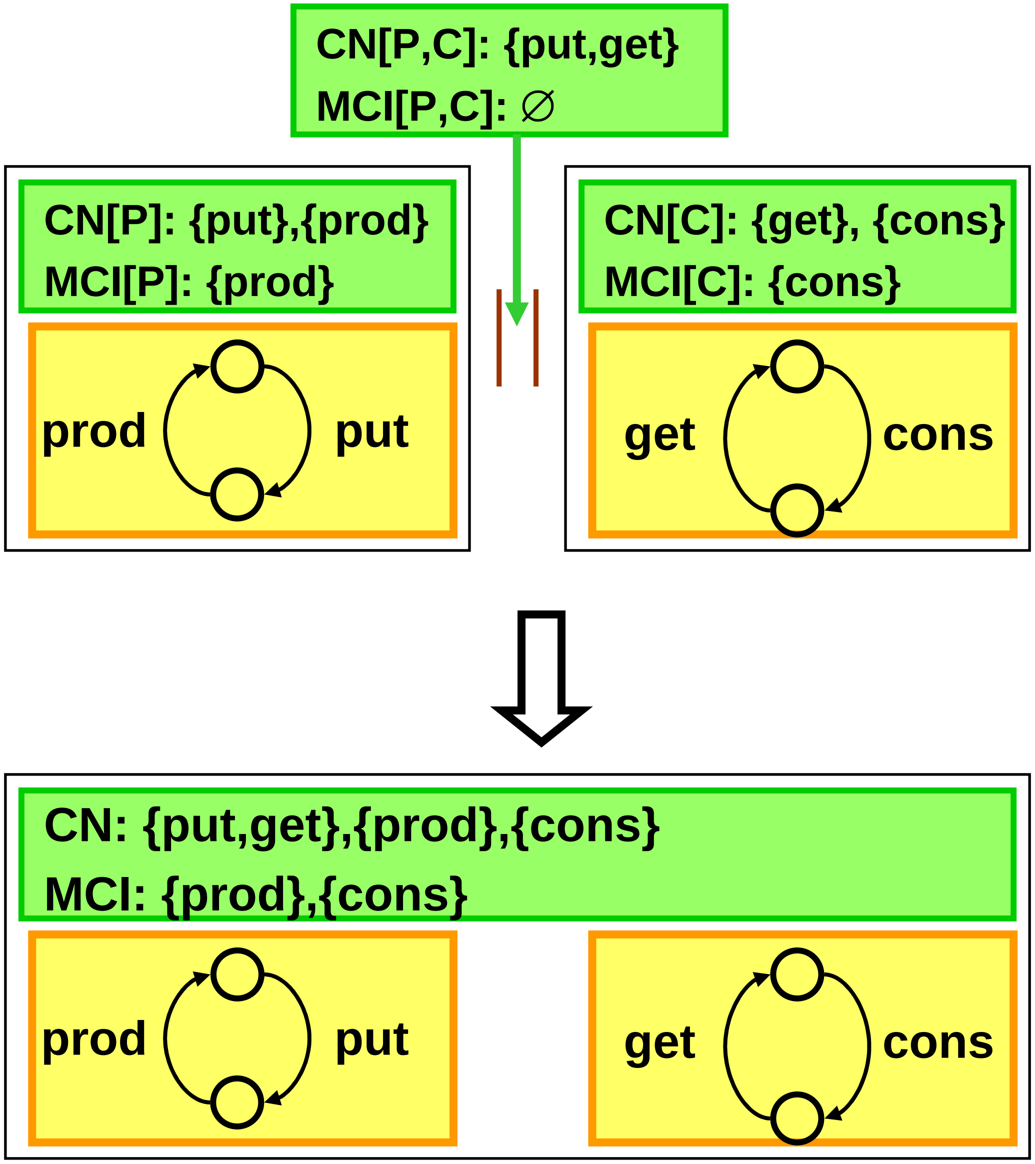


CN:{in1,out,in}
MCI: {out}

Interaction models – operational semantics

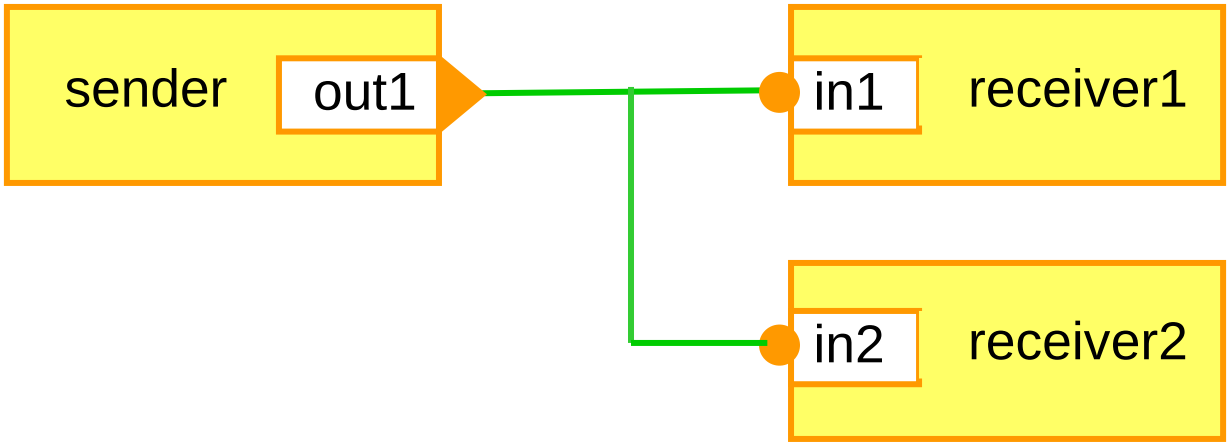


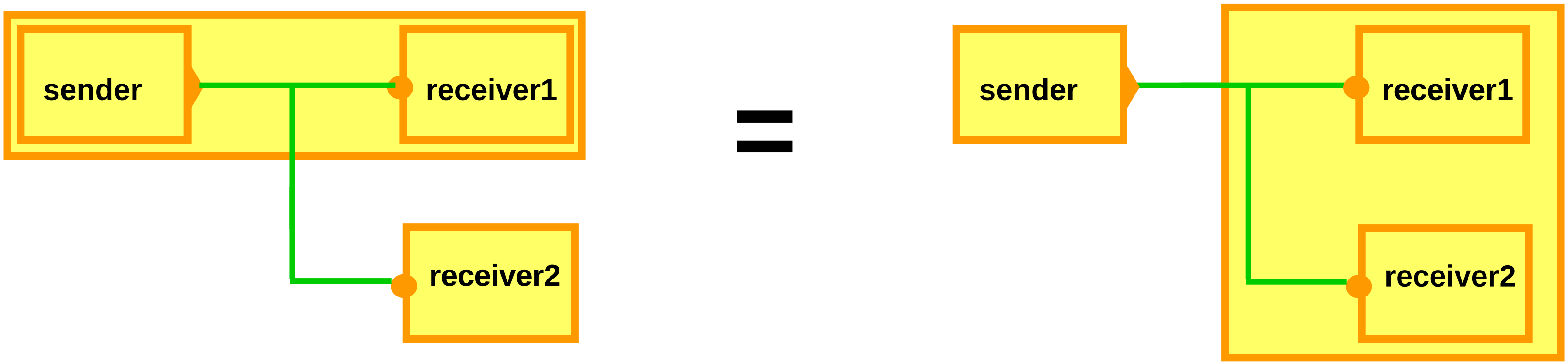
Interaction models - composition



Interaction models – results [Goessler Sifakis 2003]

Incremental commutative composition encompassing blocking and non blocking interaction



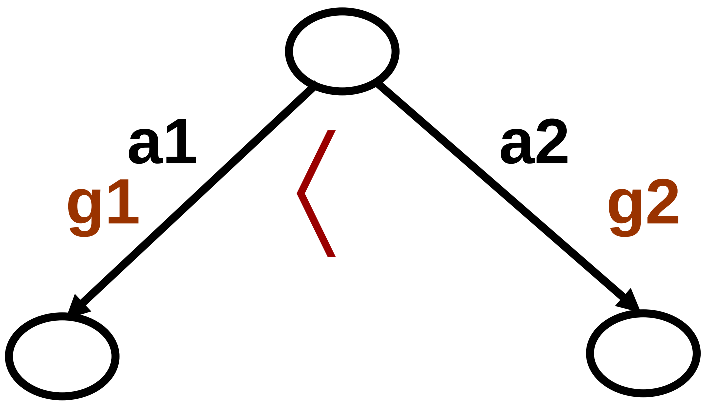


Overview

- Component-based construction – the notion of glue
- Interaction Models
- Priorities
- The BIP framework
- Discussion

Priorities

Restrict non-determinism by using (dynamic) priority rules



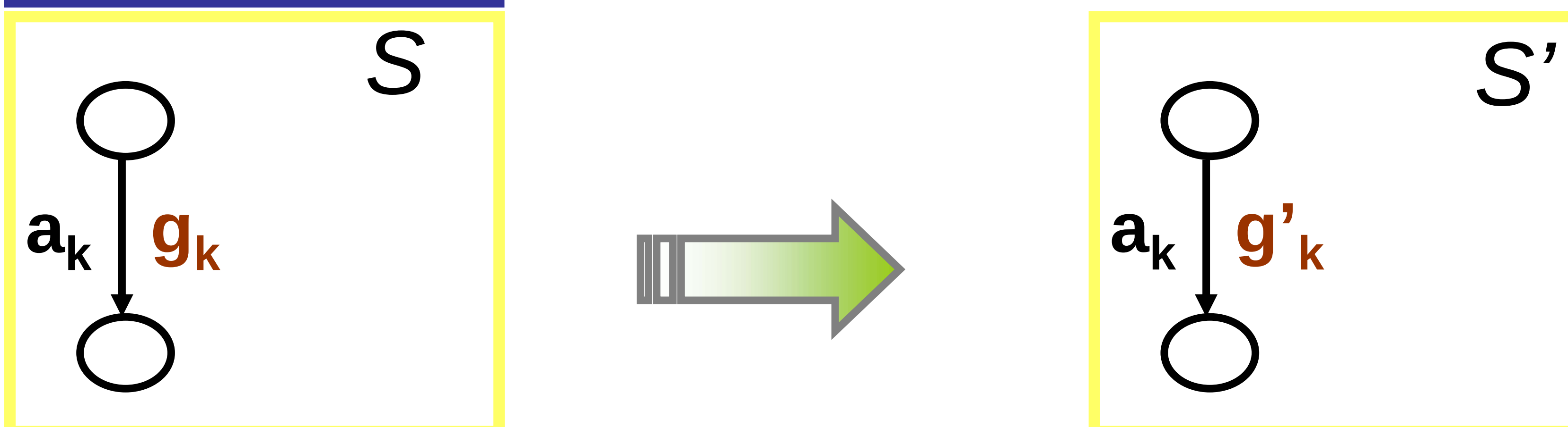
Priority rule	**Restricted guard g1’**
true → a1 〈 a2	$g1' = g1 \wedge \neg g2$
C → a1 〈 a2	$g1' = g1 \wedge \neg(C \wedge g2)$

Priorities

A *priority order* is a strict partial order $\prec \subseteq A^c \times A$

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

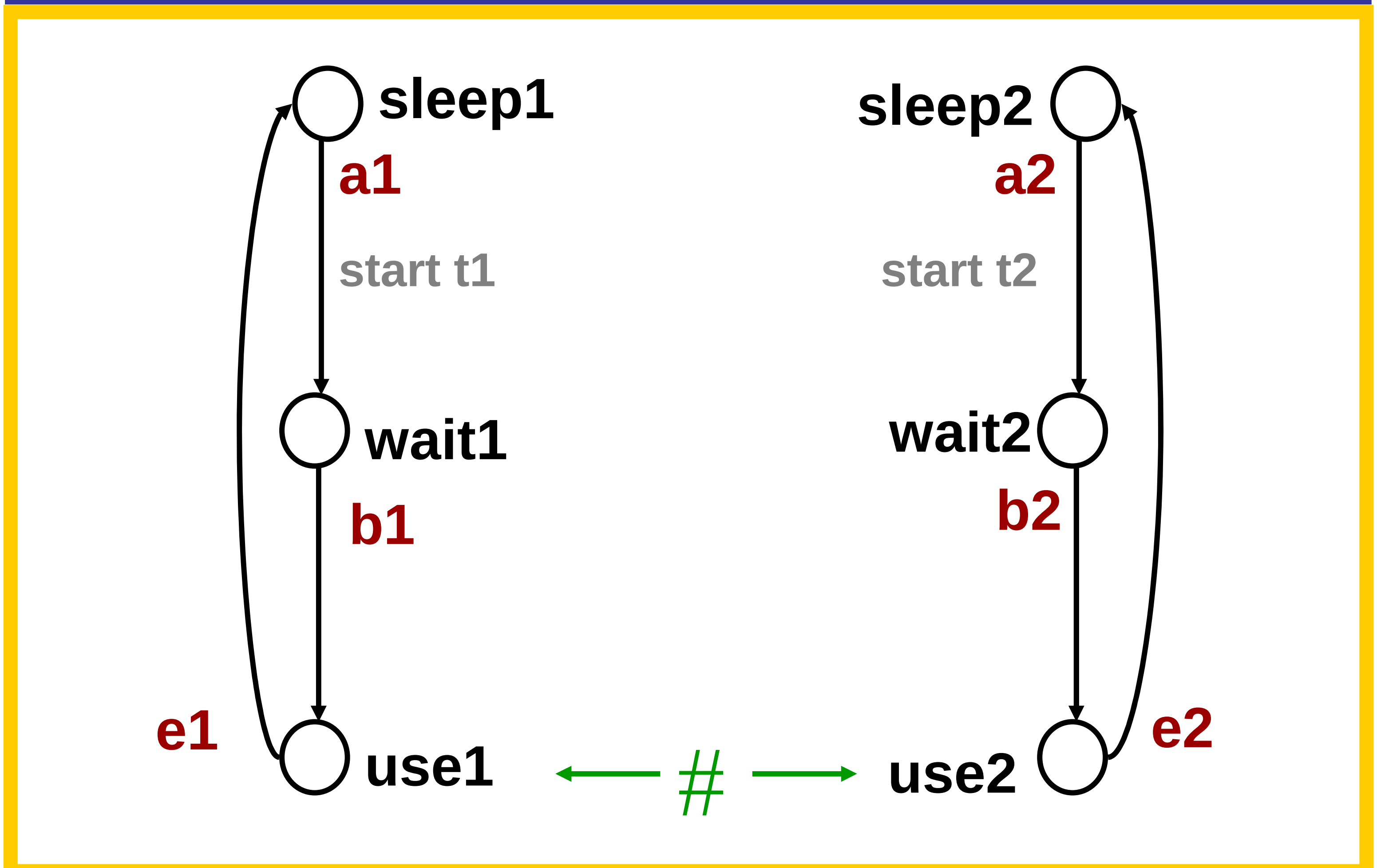
$pr = \{ C_i \rightarrow \prec_i \}_i$



$$g'_k = g_k \wedge \bigwedge_{C \rightarrow \prec \, \in pr} \Big(C \Rightarrow \bigwedge_{ak \prec ai} \neg\, g_i \Big)$$

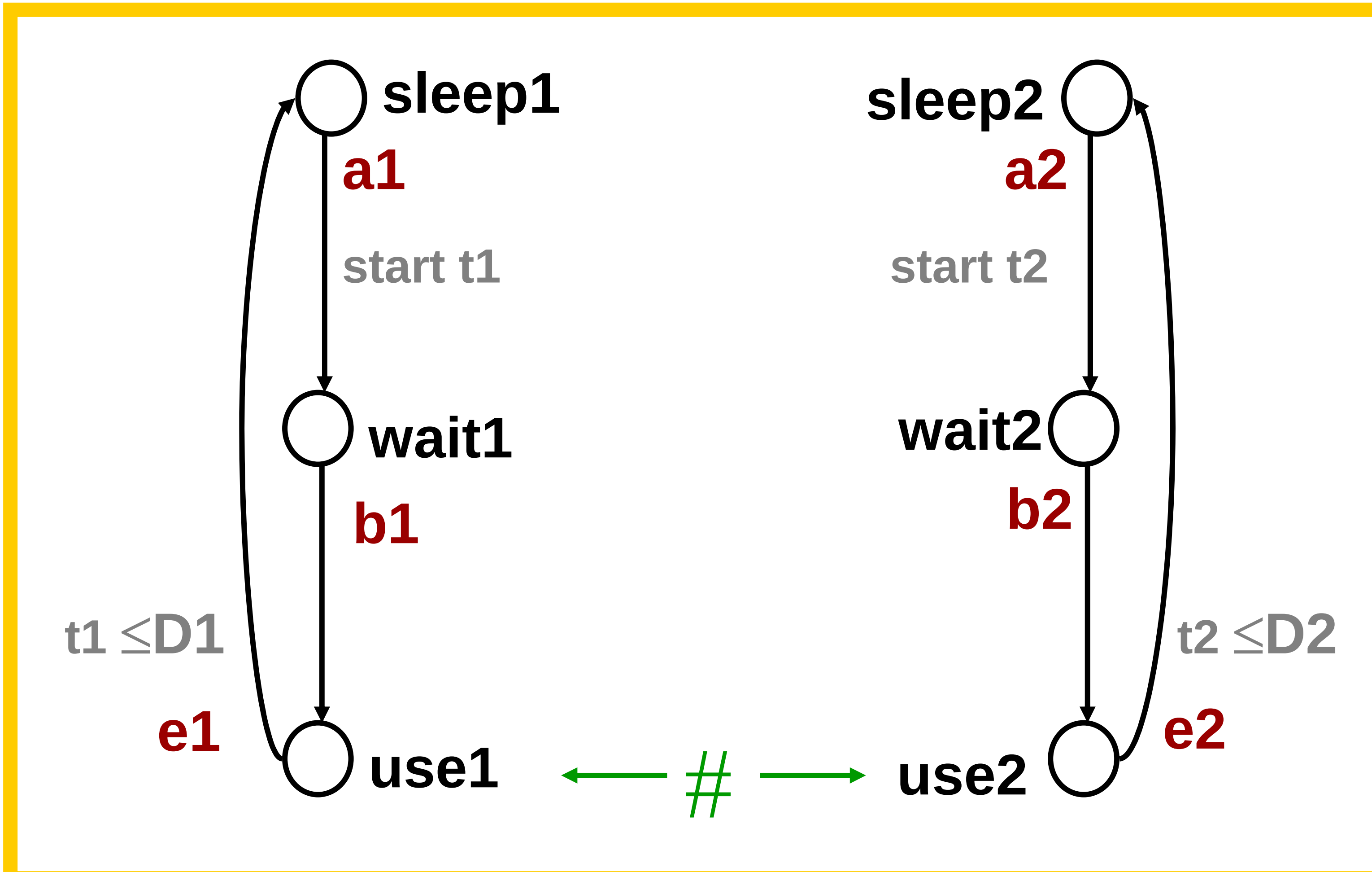
Priorities - FIFO policy

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

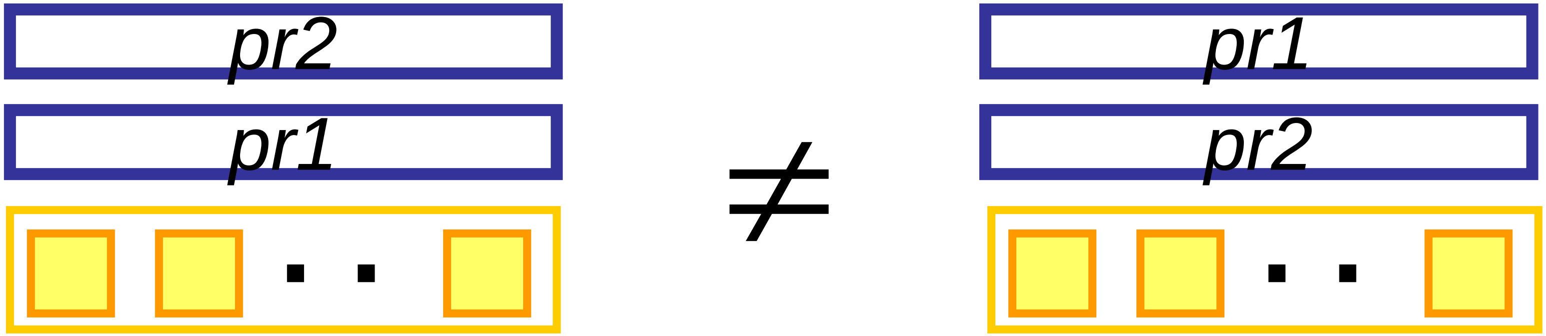


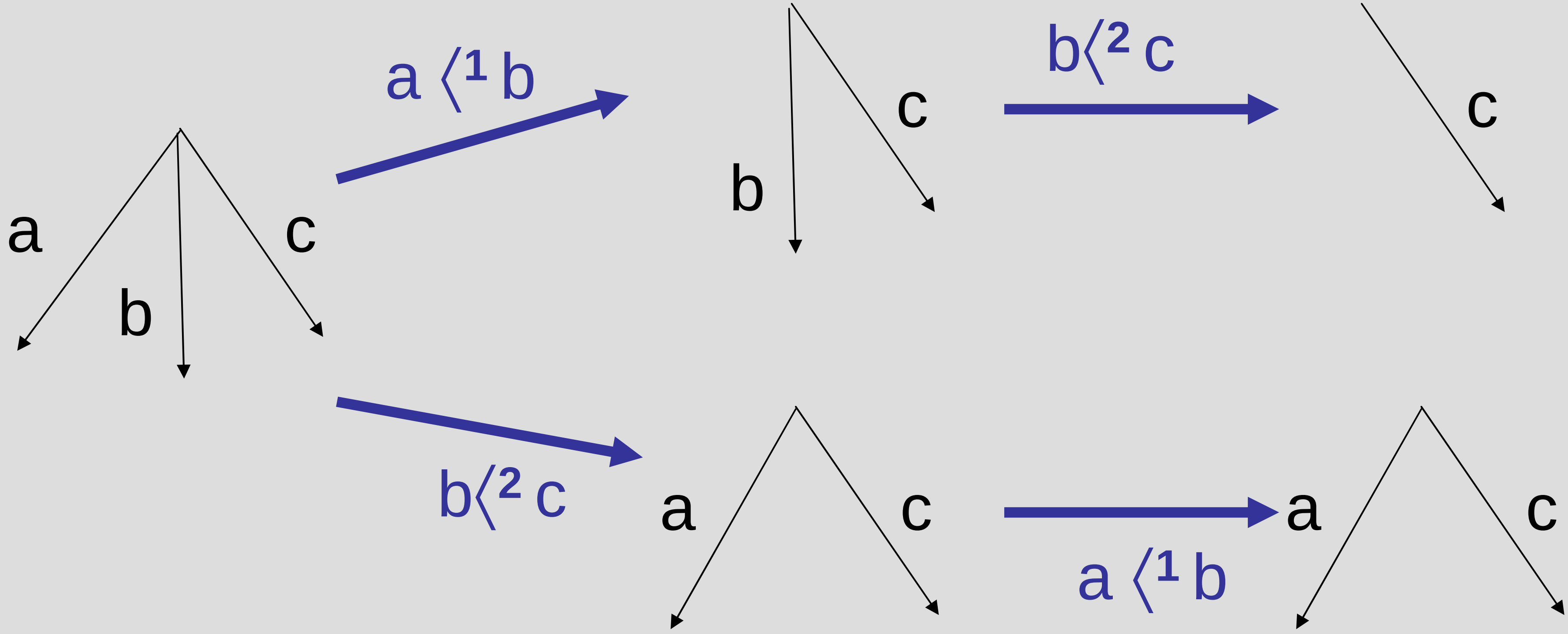
Priorities - EDF policy

$D1\text{-}t1 \leq D2\text{-}t2 \rightarrow b2 \langle\, b1$ $\qquad$ $D2\text{-}t2 \leq D1\text{-}t1 \rightarrow b1 \langle\, b2$



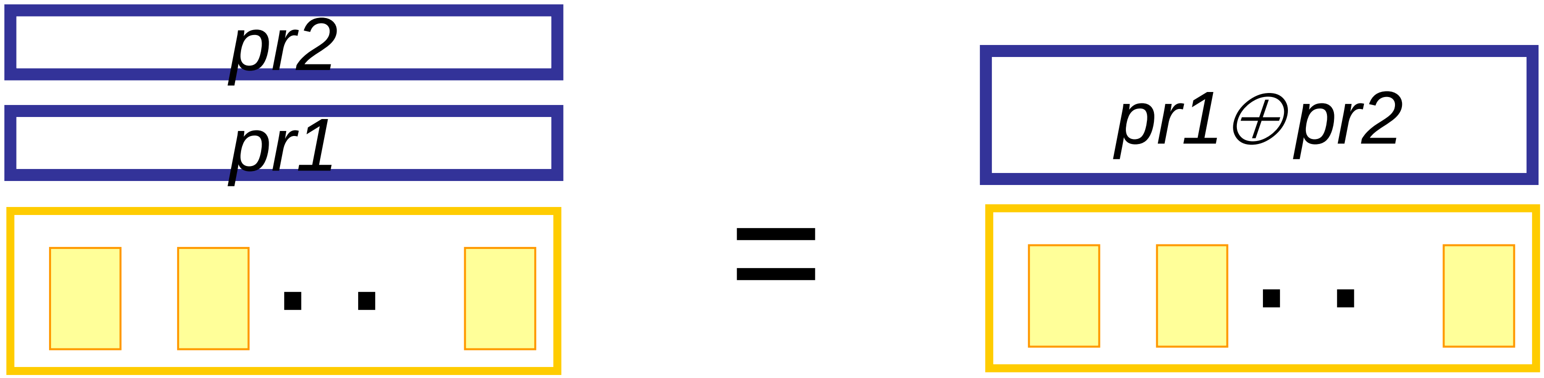
Priorities - Composition





Priorities – Composition (2)

We take:



pr1⊕ pr2 is the least priority containing pr1∪pr2

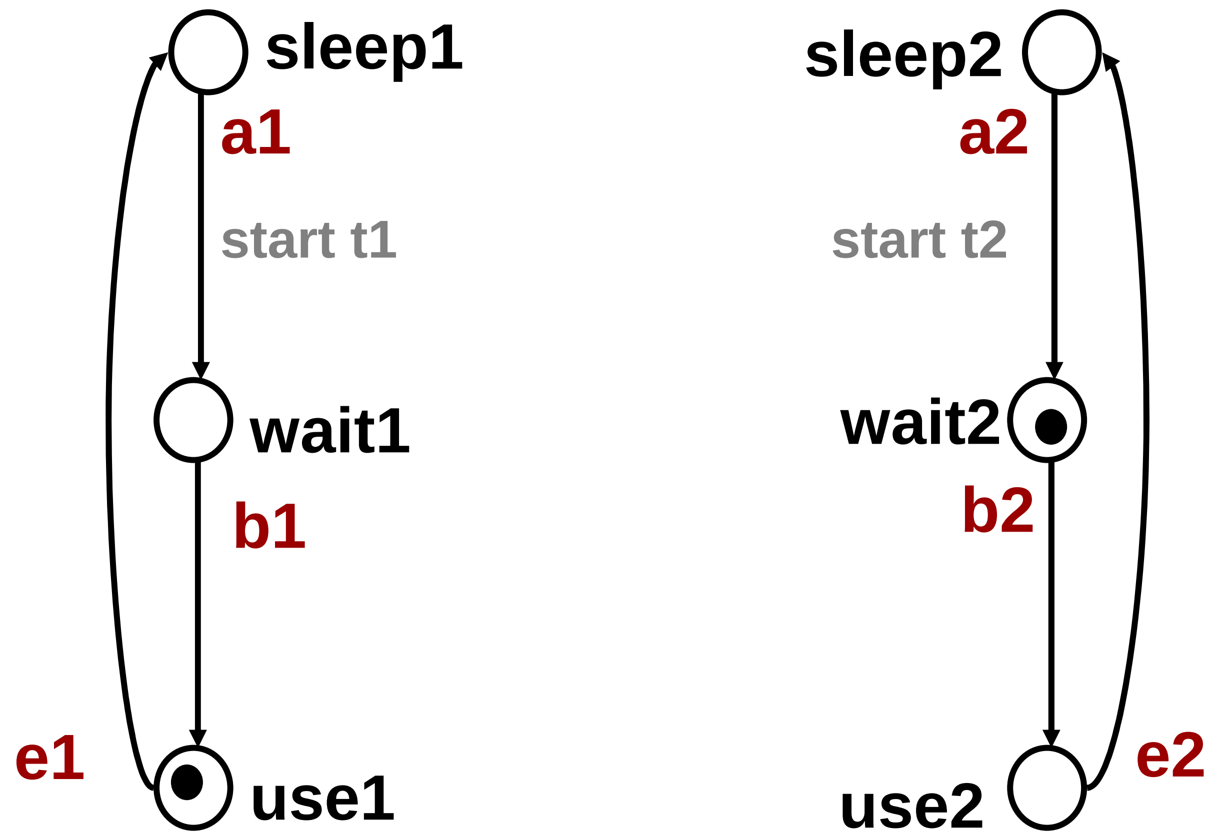
Results :

- The operation ⊕ is partial, associative and commutative
- pr1(pr2(B)) ≠pr2(pr1(B))
- pr1⊕ pr2(B) *refines* pr1∪pr2(B) *refines* pr1(pr2(B))
- Priorities preserve deadlock-freedom

Priorities - mutual exclusion + FIFO

t1≤ t2 → b1〈 b2	t2≤ t1 → b2〈 b1

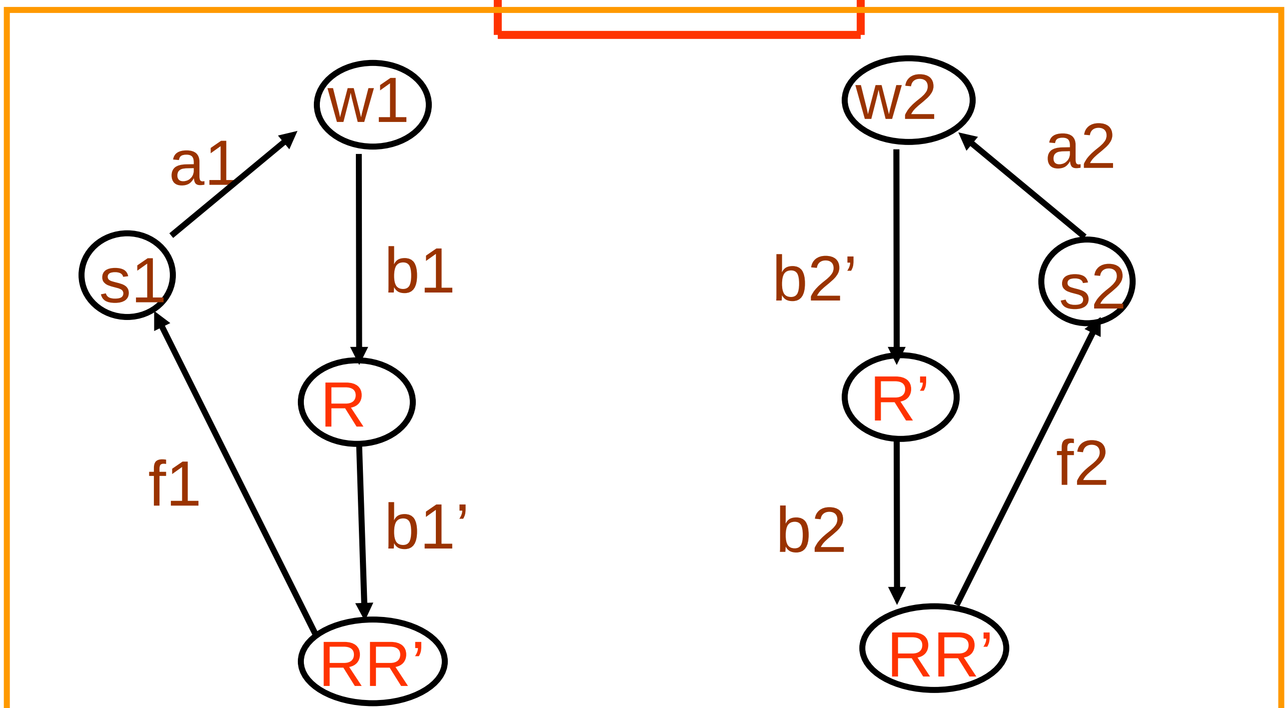
true → b1〈 e2	true → b2〈 e1



Priorities – mutual exclusion: example

Mutex on R : b1 ⟨ f2 b2 ⟨ { f1, b1’}

Mutex on R’ : b1’ ⟨ { f2, b2 } b2’ ⟨f1



Risk of deadlock: The composition is not a priority order !

Overview

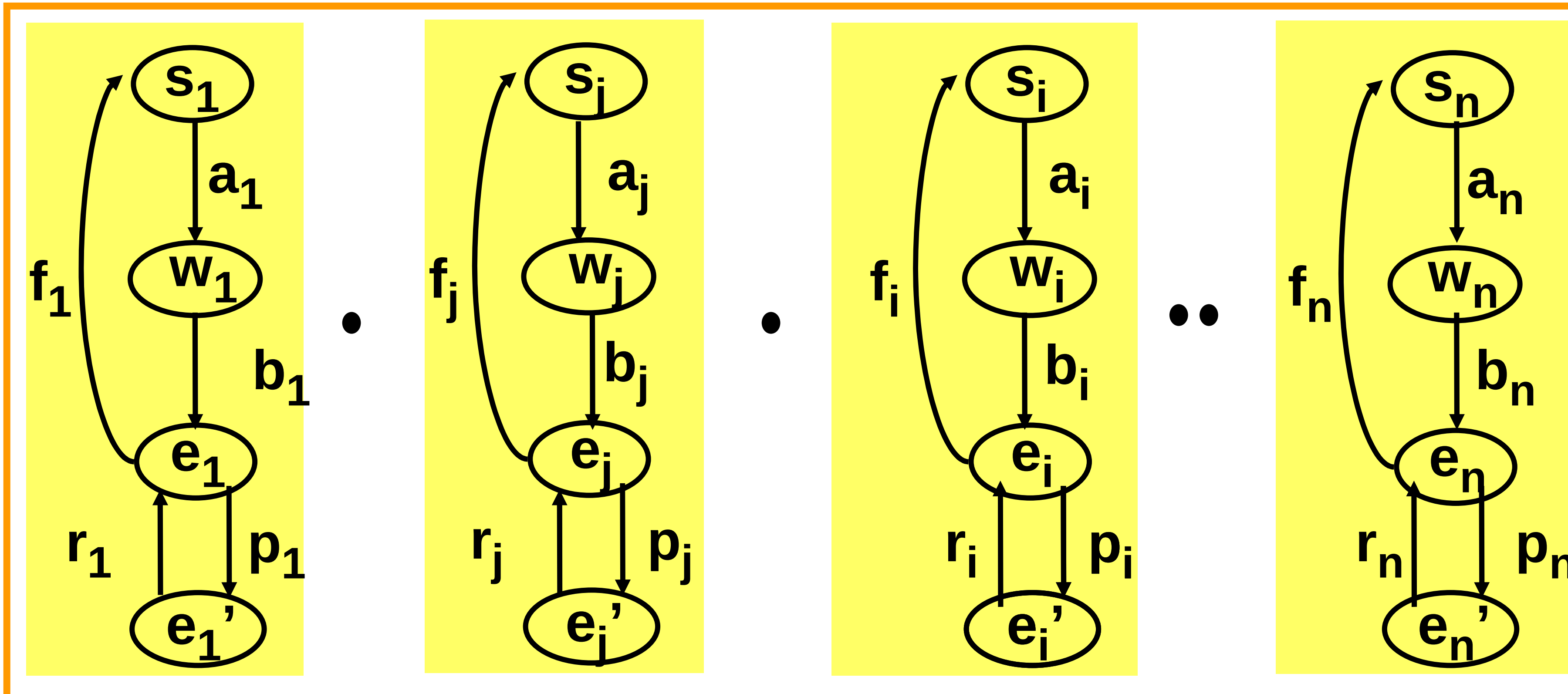
- Component-based construction – the notion of glue
- Interaction Models
- Priorities
- The BIP framework
- Discussion

The BIP framework - fixed priority preemptive scheduling (1)

$b_i \langle b_j$, $r_i \langle r_j$, $r_i \langle b_j$, $b_i \langle r_j$ (access to the resource – priority preserved by composition)

$\{b_i, p_j\} \langle f_j$, $\{r_i, p_j\} \langle f_j$, $n \geq l > j \geq 1$ (non pre-emption by lower pty tasks)

CN: $\{b_i,p_j\}\ \{r_i,p_j\}$ for $n \geq i,j \geq 1$ **MCI:** a_i, f_i, b_i for $n \geq i \geq 1$

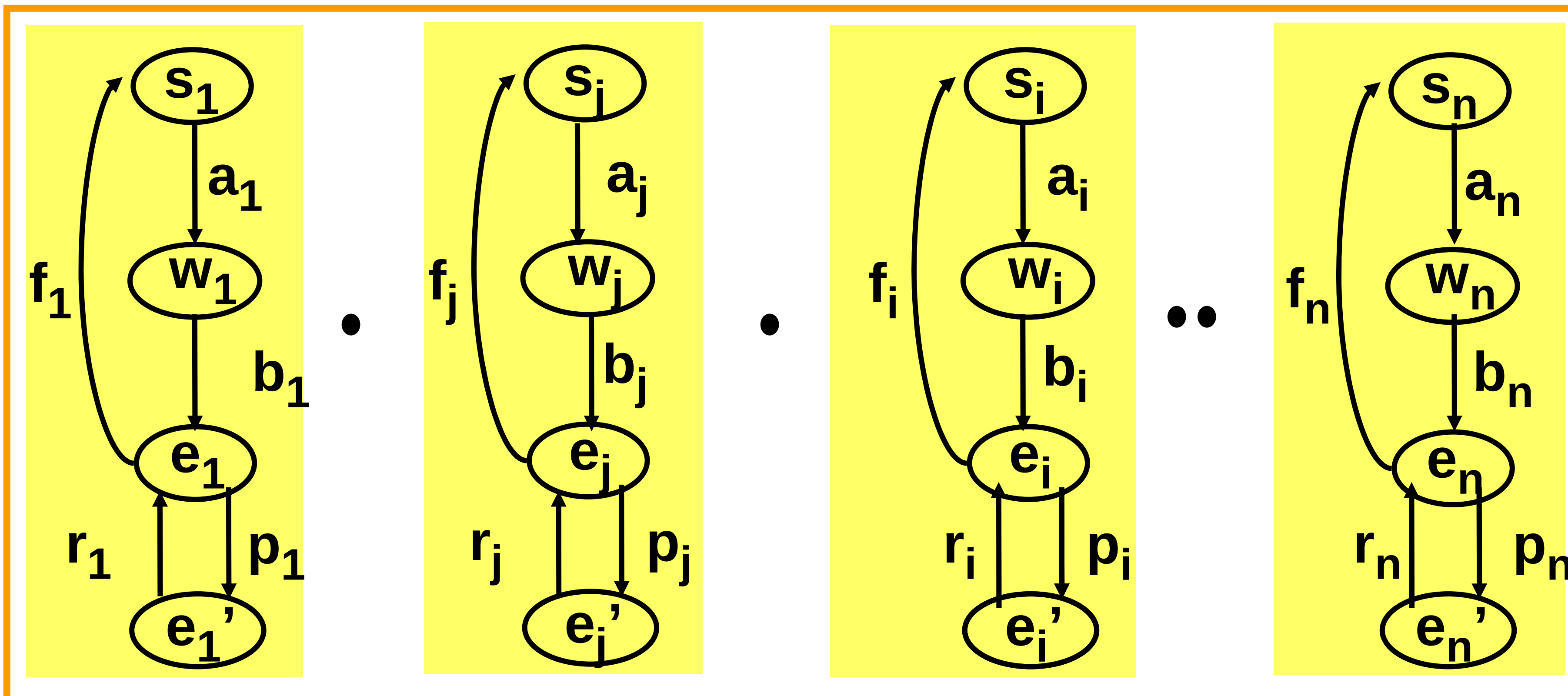


The BIP framework - fixed priority preemptive scheduling (2)

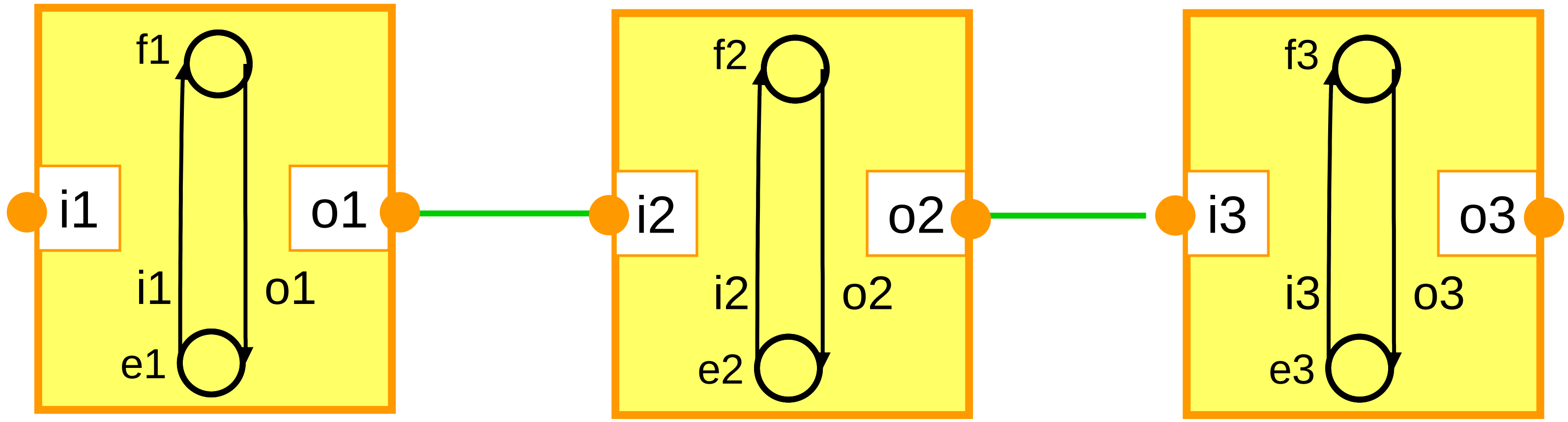
$b_i \langle b_j$, $r_i \langle r_j$, $r_i \langle b_j$, $b_i \langle r_j$ (access to the ressource – pty inherited by composition)

$p_i \langle f_j$, if w_i or e'_i $\quad n \geq l > j \geq 1$ (non pre-emption by lower pty tasks)

$\{b_i, r_i\} \langle \{f_j\ p_j\}$ $\quad n \geq l, j \geq 1$ (Mutual exclusion)

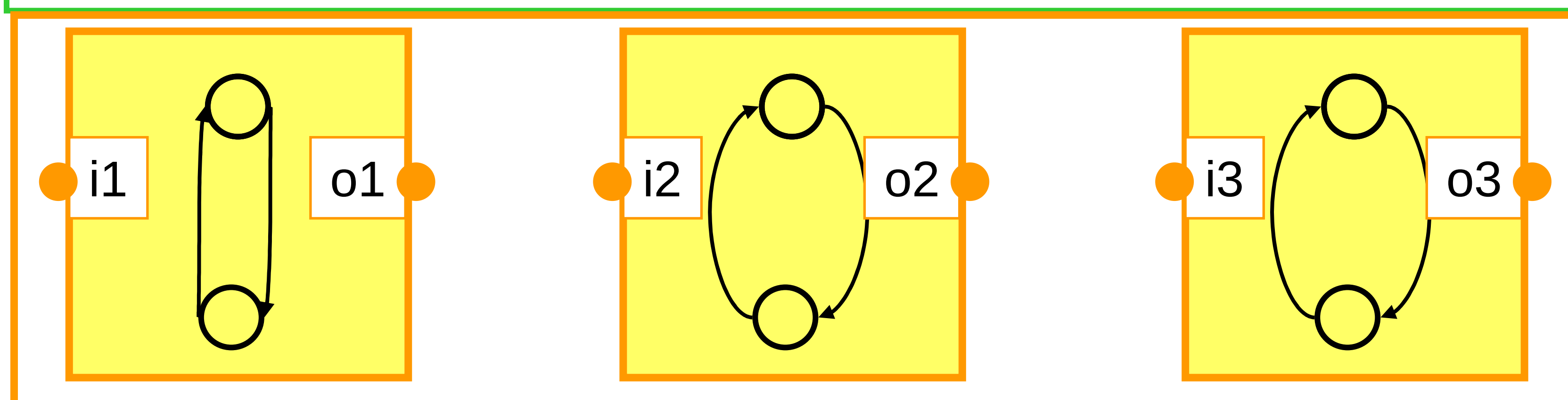


The BIP framework – run to completion

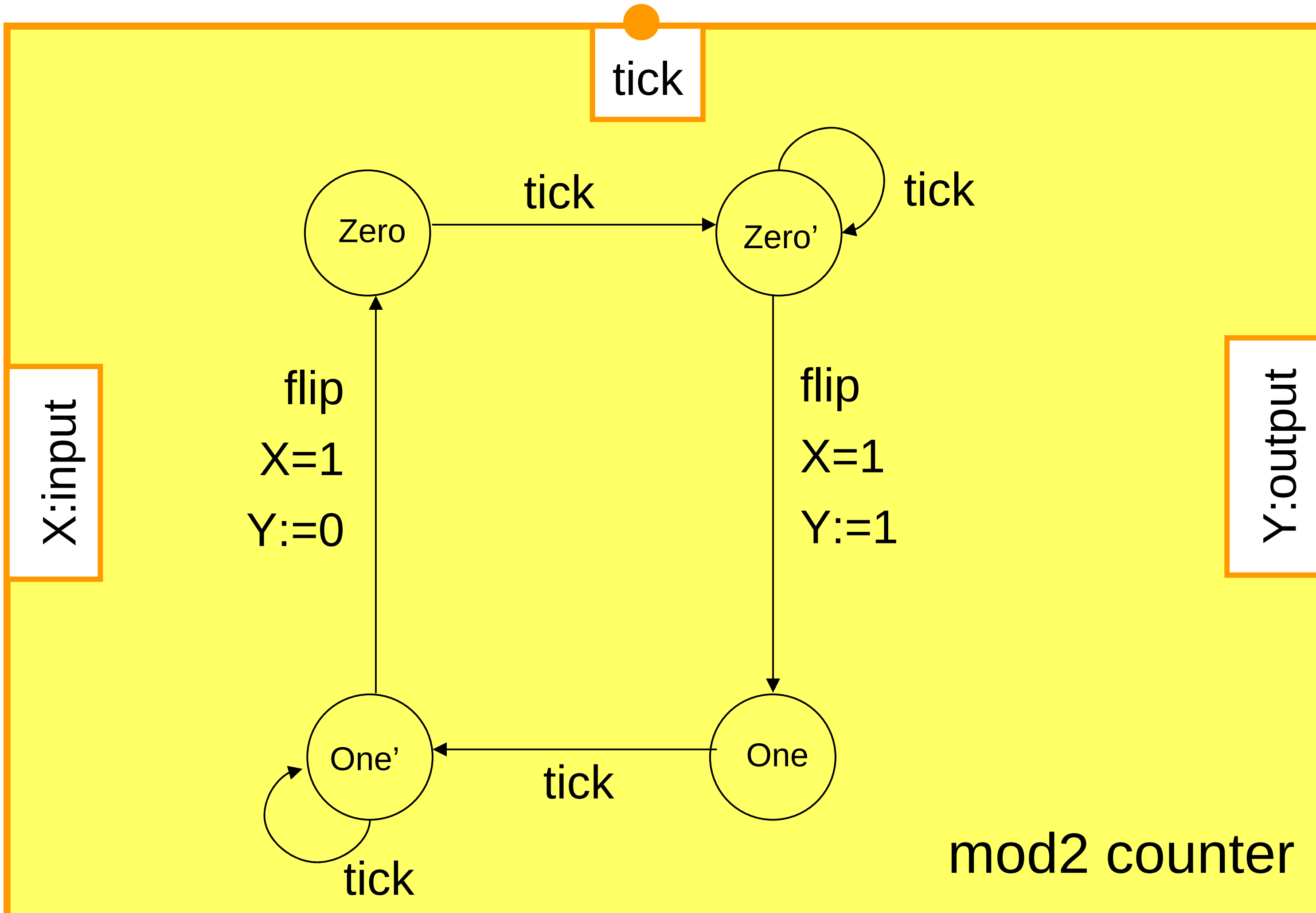


i1 〈 {o1,i2} 〈 {o2,i3} 〈 o3

CN: {o1,i2}, {o2,i3}   MCI: ∅



The BIP framework - modulo-8 counter : atomic component



The BIP framework - modulo-8 counter: the model

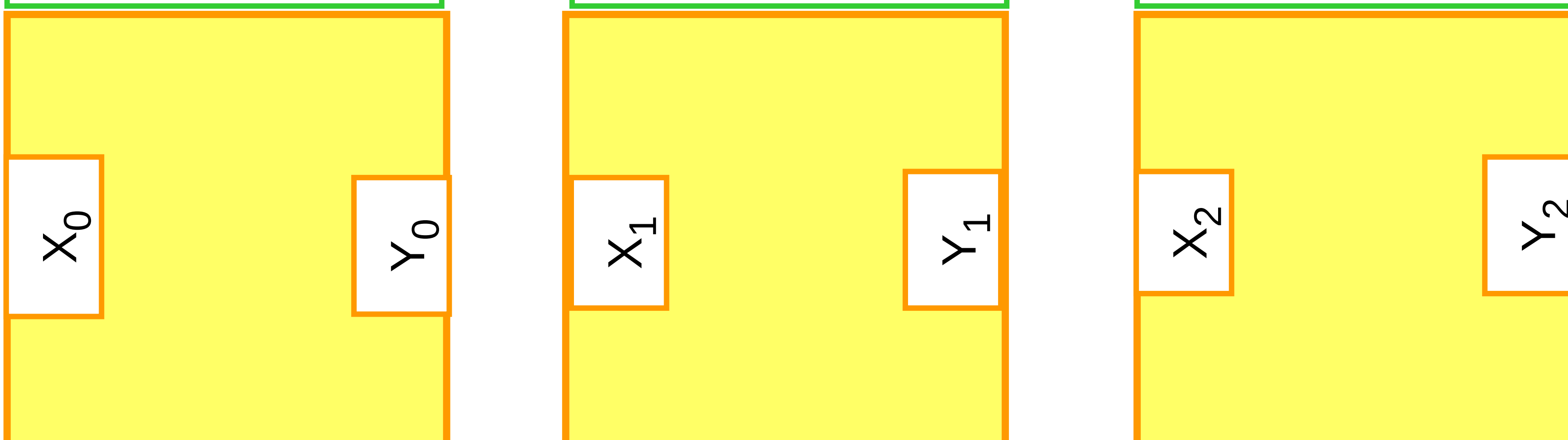
tick $\langle$ flip$_0$, tick $\langle$ flip$_1$, tick $\langle$ flip$_2$

CN: tick={tick$_0$, tick1$_1$, tick$_2$}

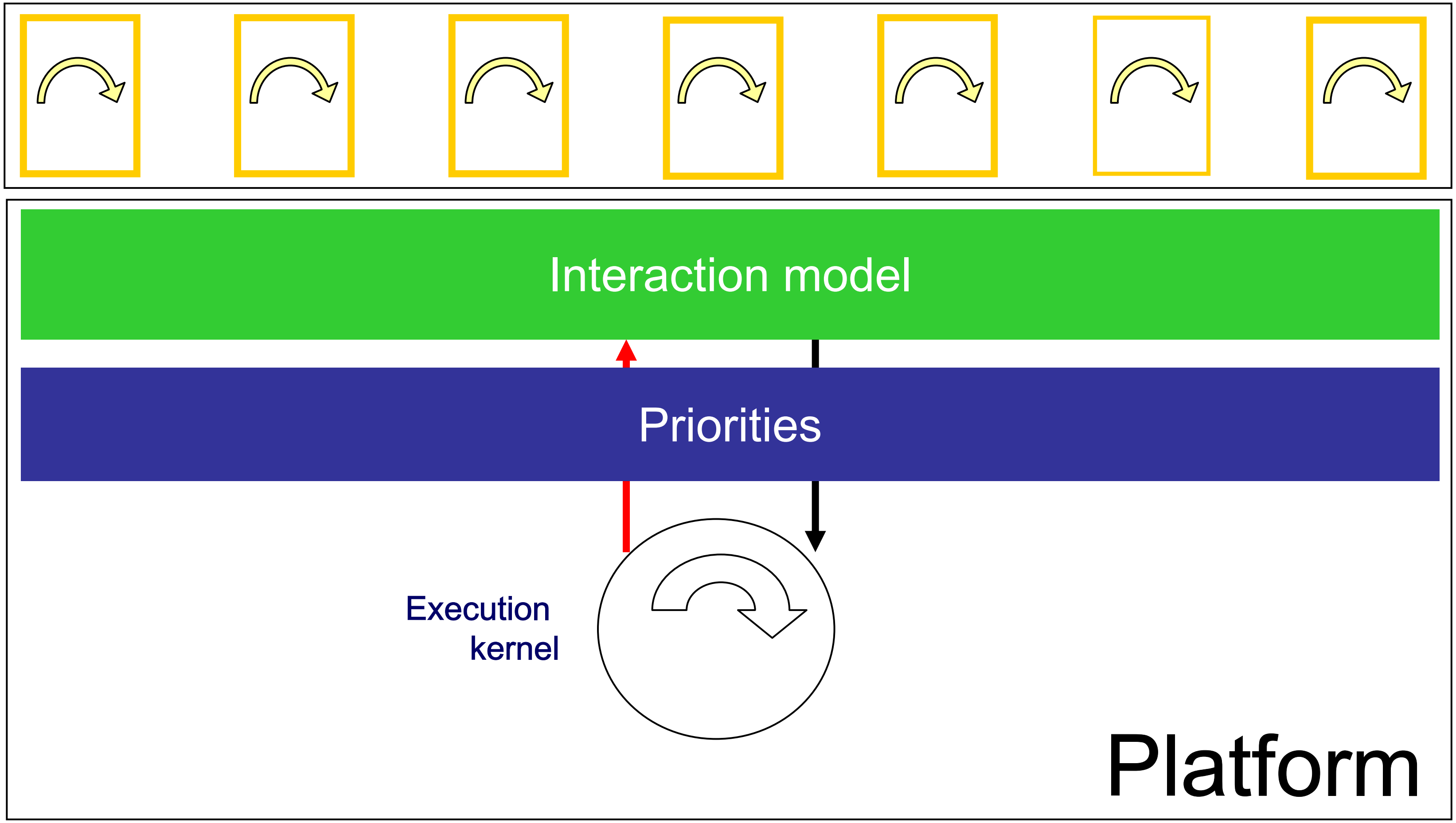
MCI: $\varnothing$

Transfer : $X_1 := Y_0$; $X_2 := Y_1 \wedge Y_0$

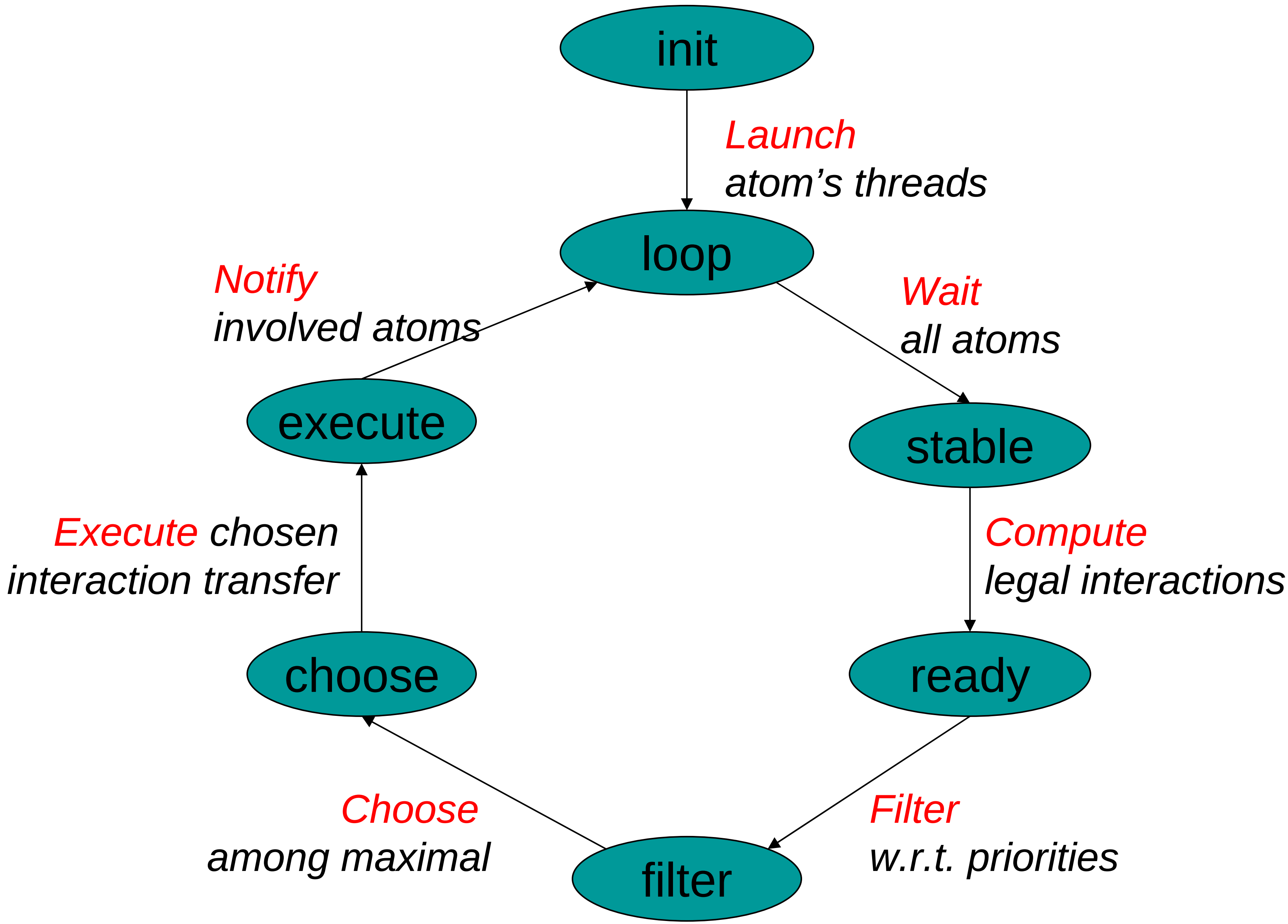
CN: tick$_0$, flip$_0$ **MCI: flip$_0$**	**CN: tick$_1$, flip$_1$** **MCI: flip$_1$**	**CN: tick$_2$, flip$_2$** **MCI: flip$_2$**



The BIP framework - The execution platform



The execution platfrom – the kernel

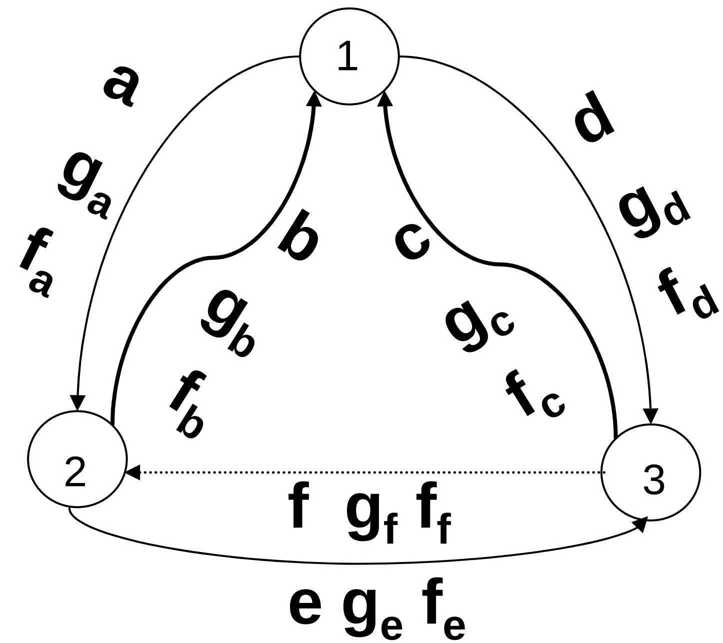


The BIP framework - atomic component: abstract syntax

```
Component: C
Ports: p1,p2, …
Data: x,y,z, ….
Access: (p1,{x,y,z}), (p2,{x,u,v}),

Behavior:
        state s1
                on p1 provided g1 do f1 to state s1’
                ……………….     ……
                on pn provided gn do fn to state sn’

        state s2
                on …..

                ……….

        state sn
                on ..  ..
```

The BIP framework : Implementation - atomic components

```
run() {
 Port* p;
 int state = 1;
 while(true) {
  switch(state) {
   case 1: p = sync(a, g_a, d, g_d);
          if (p == a)
            f_a;  state = 2;
          else
            f_d;  state = 3;
          break;
   case 2: p = sync(b, g_b, e, g_e);
           ...
   case 3:  ...
  }
 }
}
```



Implementation - connectors and priorities: abstract syntax

Connector: BUS={p, p', … , }
complete()
Behavior:
on $\alpha 1$ **provided** $g_{\alpha 1}$ **do** $f_{\alpha 1}$
on $\alpha 2$ **provided** $g_{\alpha 2}$ **do** $f_{\alpha 2}$

Priorities: PR
if C1 **then** {(α1, α2), (α3, α4) , … }
if C2 **then** {(α,…), (α,…) , … }

if Cn **then** {(α,…), (α,…) , … }

Overview

- Component-based construction – the notion of glue
- Interaction Models
- Property enforcement by controllers
- Priorities
- The BIP framework
- Discussion

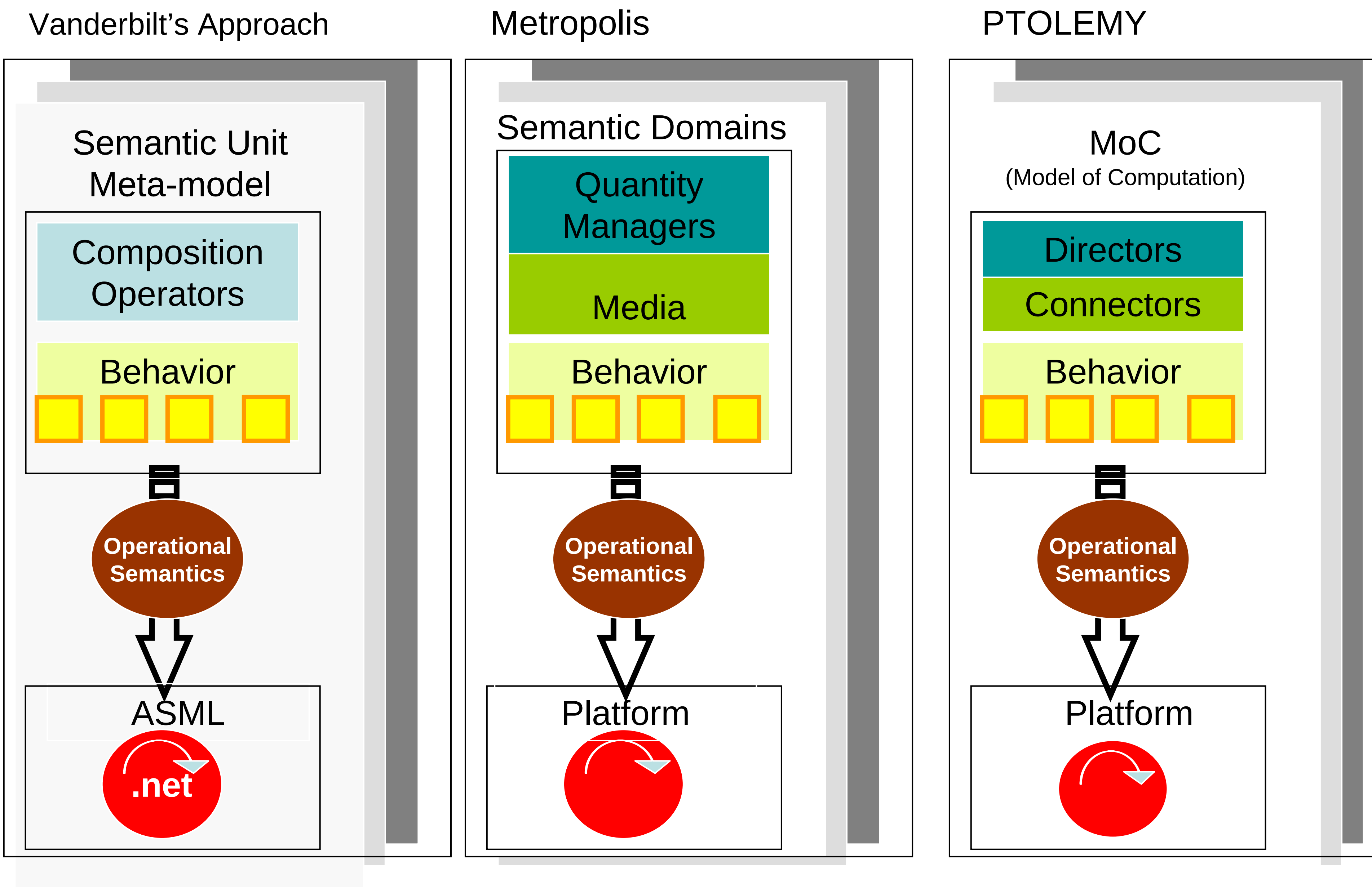
Discussion - Summary

- Framework for component-based modeling encompassing heterogeneity and relying on a **minimal set of constructs and principles** e.g. interaction models + dynamic priorities

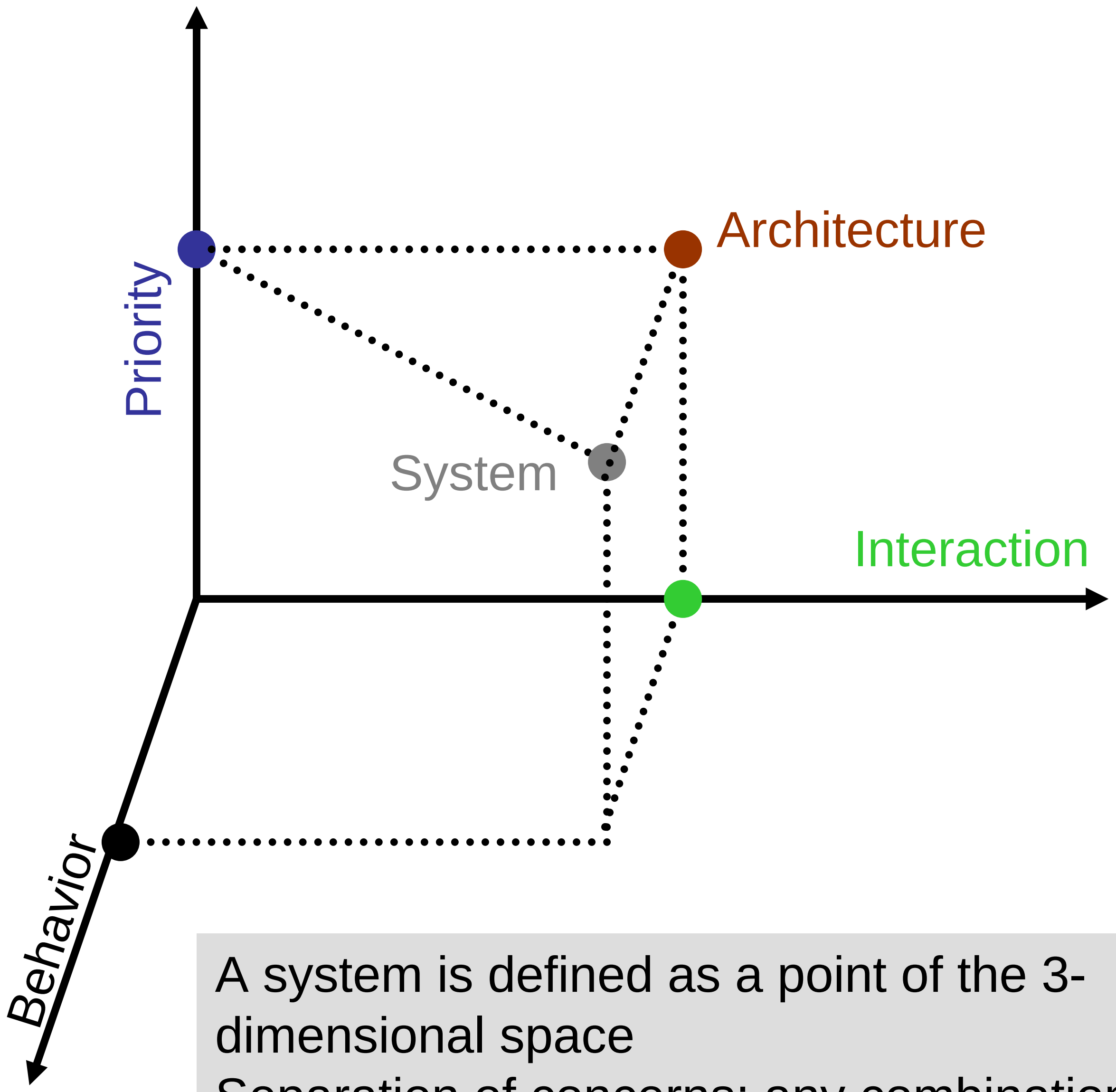
- Clear separation between behavor and architecture
  - Architecture is a first class entity
  - Correctness-by-construction techniques for deadlock-freedom and liveness, based on suficient conditions on architecture (mainly)

- Applications at Verimag
  - IF toolset allows layered description of timed systems,
  - Methodology and tool support for generating scheduled code for real-time applications (work by S. Yovine et al.)

Discussion – related approaches

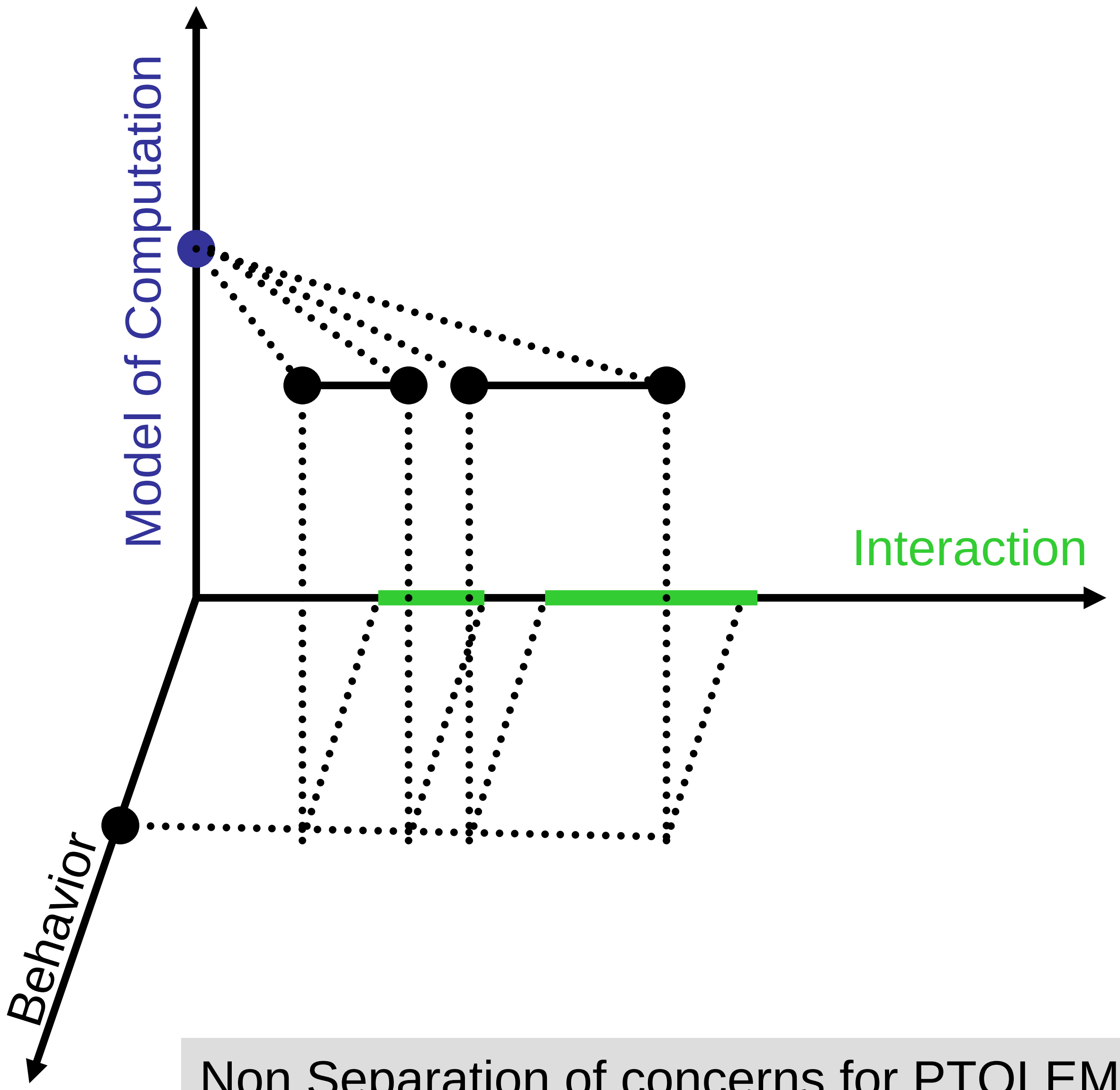


Discussion – construction space



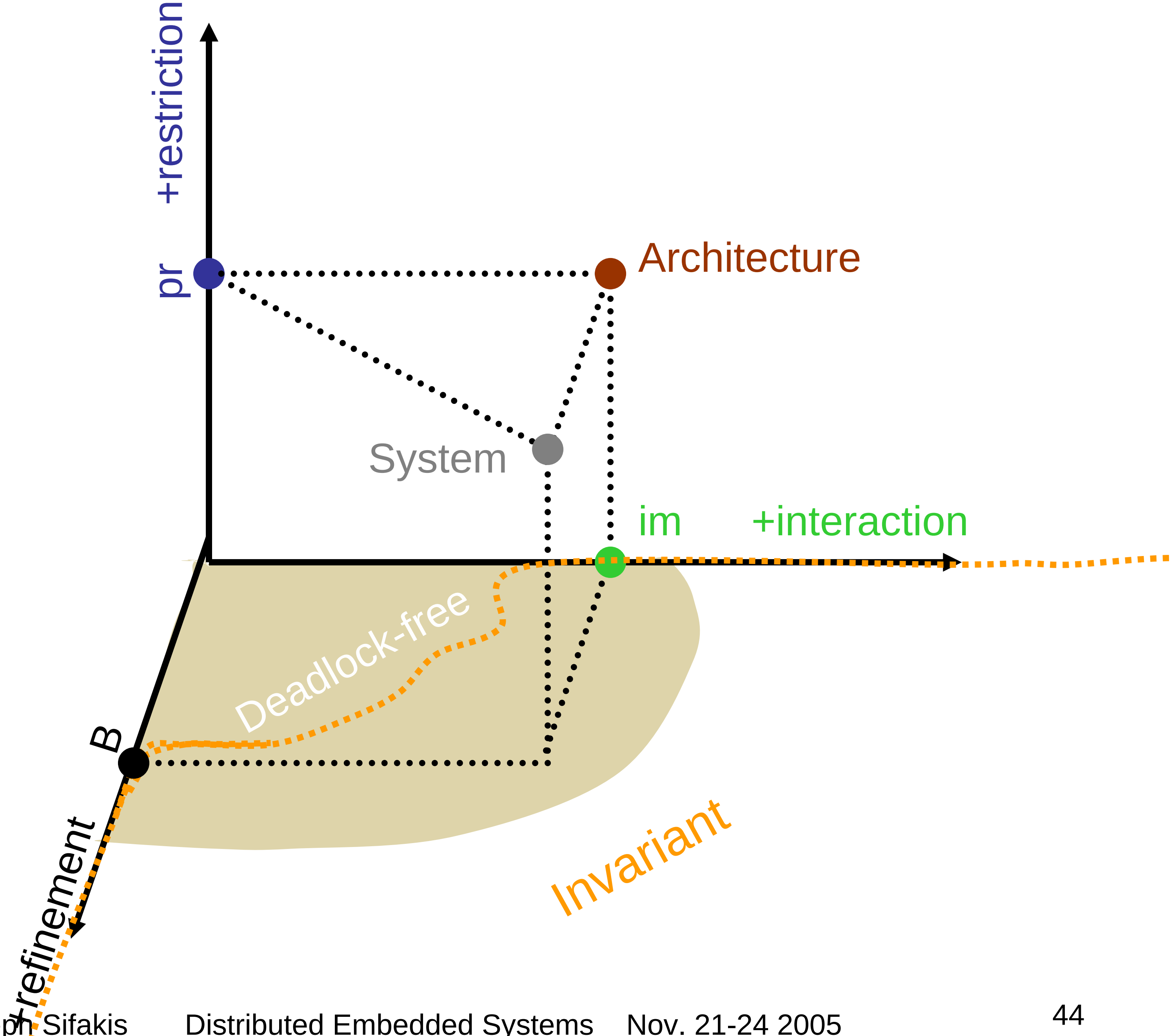
A system is defined as a point of the 3-dimensional space
Separation of concerns: any combination of coordinates defines a system

Discussion – construction space

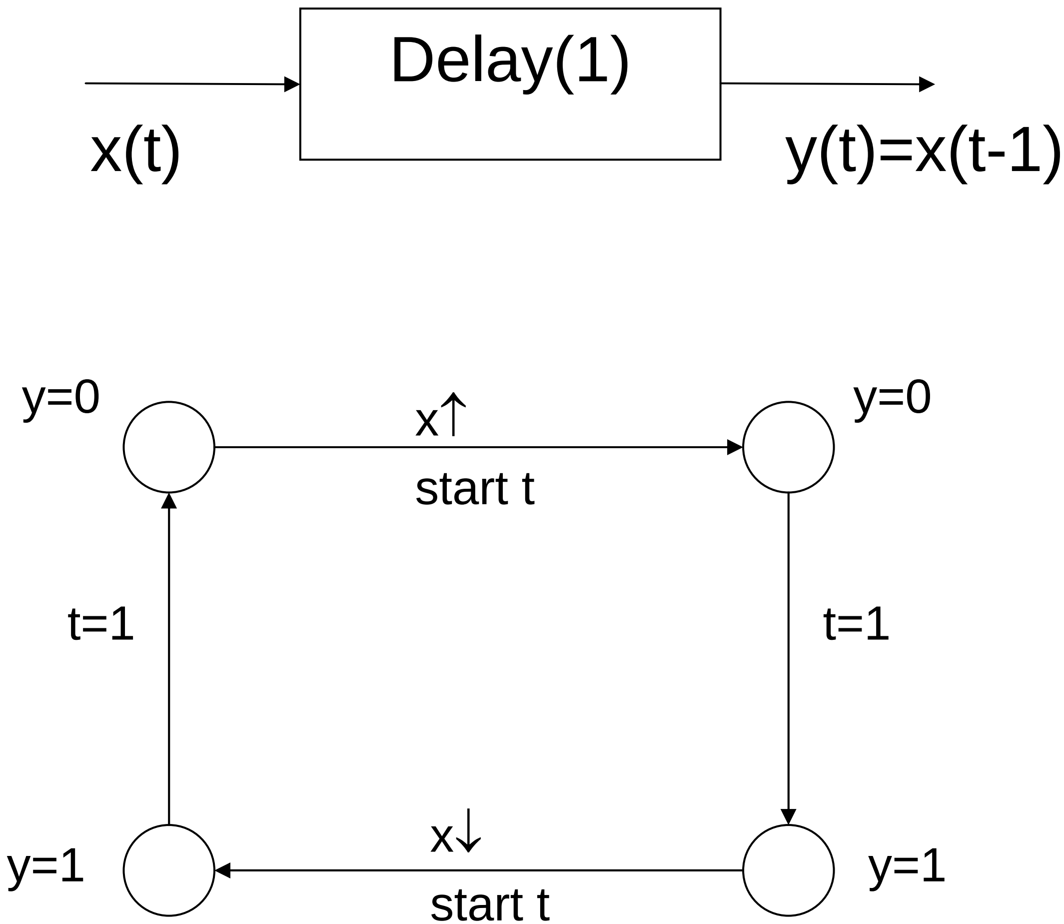


Non Separation of concerns for PTOLEMY

Discussion – construction space: property preservation



Discussion - Computational vs. Analytic Approach



x changes at most once within 1 time unit

Discussion - Computational vs. Analytic Approach

	Computational	Analytic
Non determinism	Yes	No
Executable	Yes	Maybe
Probabilities	Badly	Yes
Analysis	Verification Bounds only for finite state systems	Averages, Bounds
HW design	Maybe	Yes
SW design	Yes	Maybe

Discussion – expressiveness

Study Component Algebras CA= (**B**, GL,$\oplus$, $\cong$)

- (GL,$\oplus$) is a monoid and $\oplus$ is idempotent
- $\cong$ is a congruence compatible with operational semantics

- Study classes of glue operators
- Focus on properties relating $\oplus$ to $\cong$

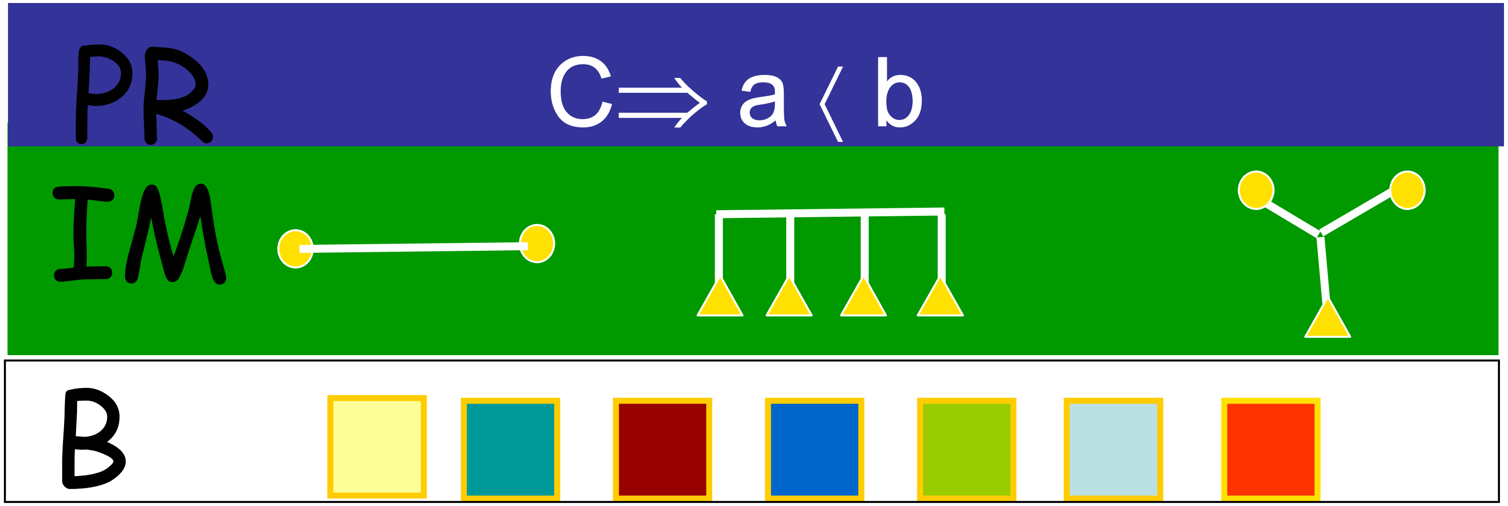
Study notions of **expressiveness** characterizing structure

Given CA_i= (**B**, GL_i,$\oplus_i$, $\cong_i$), i=1,2,

CA_1 **is more expressive** than CA_2 if $\forall$ P

$\exists\, gl_2 \in GL_2\; gl_2(B_1, .., B_n)$ sat P $\Rightarrow \exists\, gl_1 \in GL_1.\; gl_1(B_1, \ldots B_n)$ sat P

Discussion – expressiveness(2)



Example: For given B, IM and PR which coordination problems can be solved?

Notion of expressiveness different from existing ones which

- Either completely ignore structure
- or use operators where separation between structure and behavior seems problematic e.g. hiding, restriction