About Verimag

Established in January 1993, as a joint research lab between Verilog SA and the partners of the IMAG Institute (CNRS, INPG, UJF).

1993 – 1997

- Industrialization of the Lustre language, through the SCADE environment, currently commercialized by Esterel technologies, and widely used in many application areas, including avionics, critical process control, energy production and transport.
- Transfer of results on verification in testing, integrated in the ObjectGeode environment, for SDL.

1997- present :

- Public research lab, associated with CNRS, INPG, UJF.
- Staff: 30 permanent, 10 under contract, 35 Ph. Students

About Verimag - Scientific Policy

Development relying on long-term, strategic industrial partnerships, with main players in the area (eg: Airbus, France Télécom, STMicroelectronics). These aim to ensure a sufficient level of resources, to support basic research and tight connection to the evolution of practical industrial problems.

International Collaboration, through mobility of researchers (eg: summer visits, short and longer term visitors).

Strong participation in national, European, and international R&D projects

- Verimag plays a regional role through the Minalogic French “Pôle de Compétitivité”.
- Coordination of European research in embedded systems, through the Artist1 Accompanying Measure in FP5, and the Artist2 NoE in FP6.

About Verimag - Teams

Synchronous Languages and Systems (Florence Maraninchi)

- Symbolic verification techniques and abstract interpretation
- Test,simulation, early execution
- Component-based design of SoC
- Model-driven implementation techniques for synchronous systems

Distributed and Complex Systems (Yassine Lakhnech)

- Component-based design of embedded systems (theory, tools)
- Validation of distributed systems (SW and system verification&testing)
- Security (protocols, policy testing, mobile code ver, Certification M&T)
- Adaptive systems (QoS control for multimedia systems)

Timed and Hybrid Systems (Oded Maler)

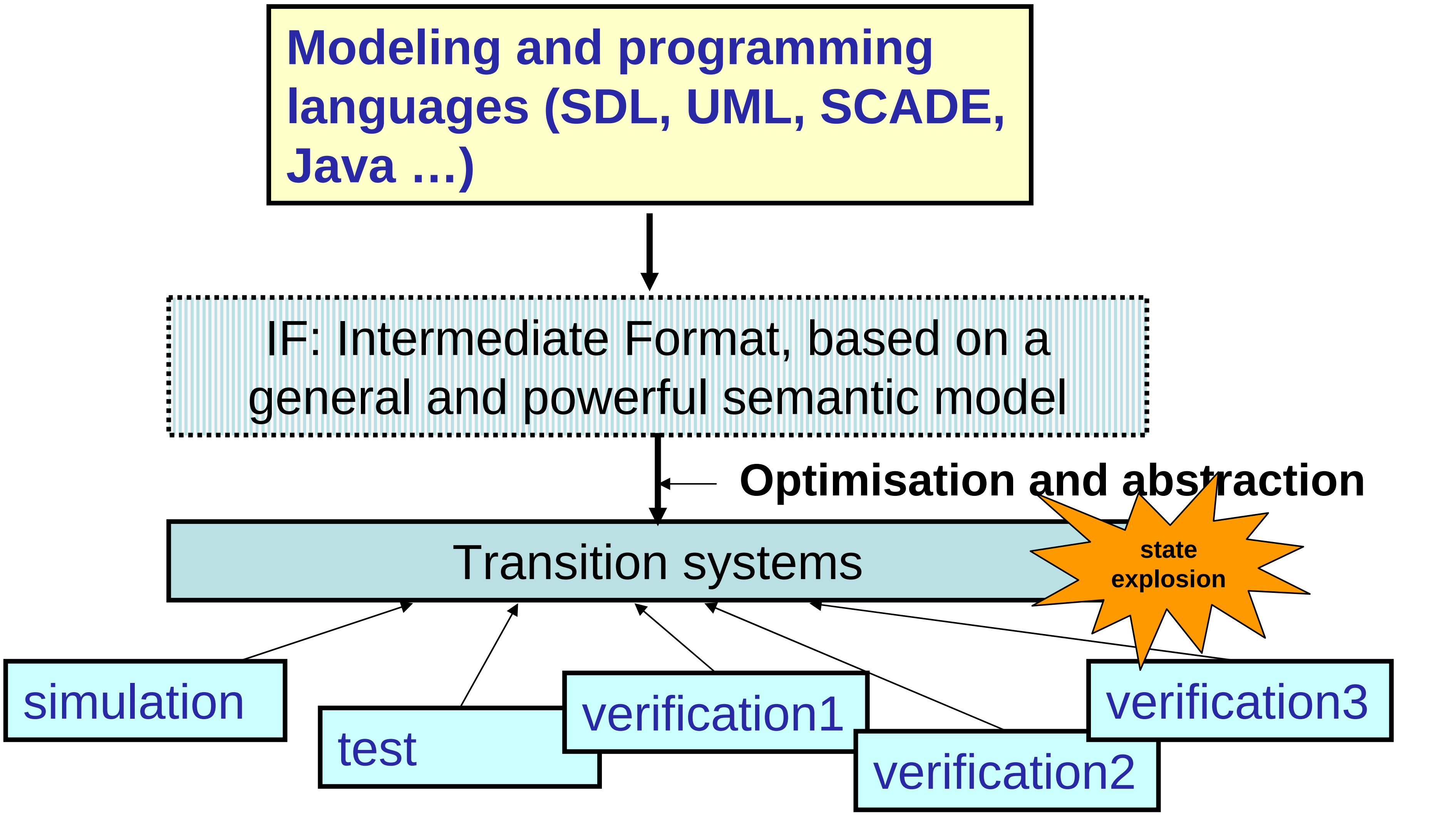
- Specification, verification, testing and code generation
- Timing and schedulability analysis of large systems
- Property-aware implementation of embedded systems,

Modeling and Validation of Real-Time Systems

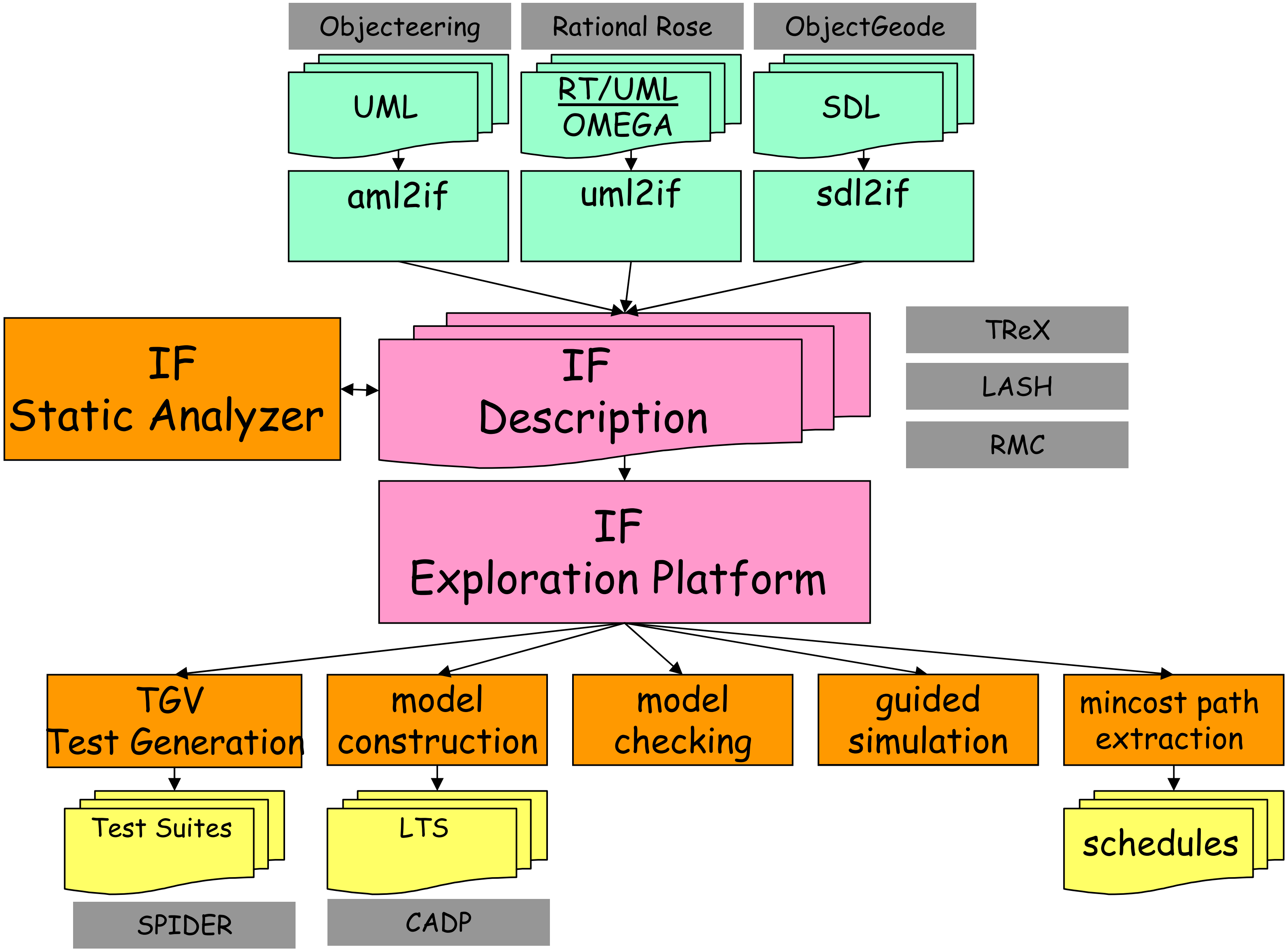
Alpine Verification Meeting
EPFL,October 6, 2005

Joseph Sifakis
VERIMAG and ARTIST2 NoE
sifakis@imag.fr

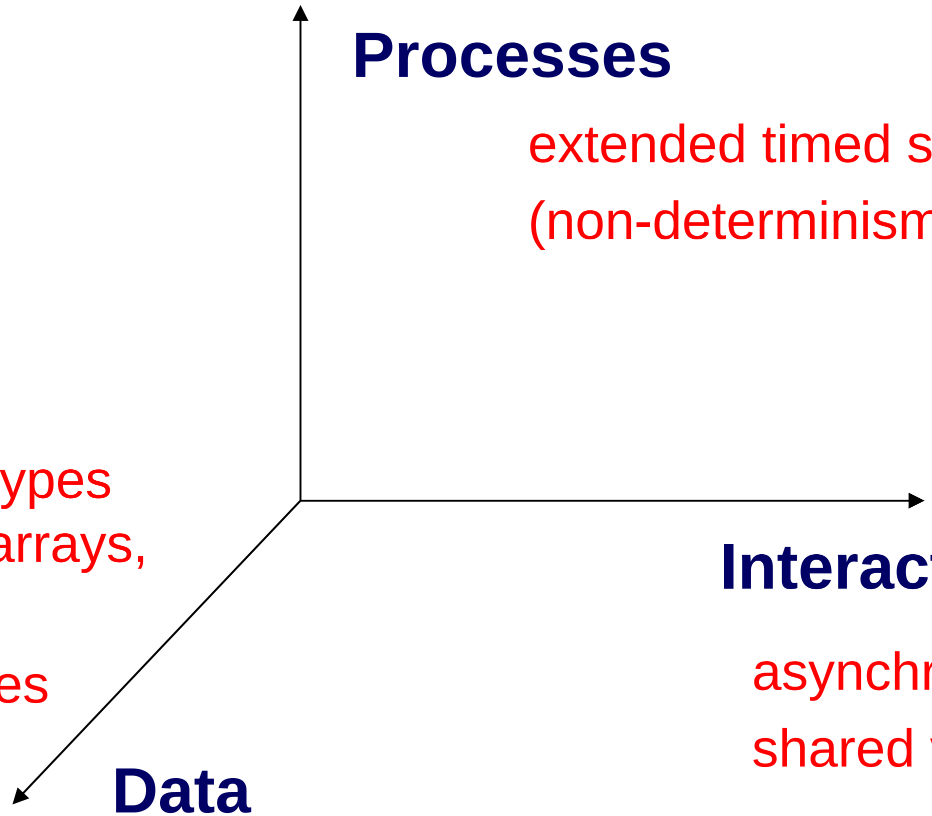
The IF toolset: approach



IF toolset: overall architecture



IF notation: System description



Processes

extended timed systems

(non-determinism, dynamic creation)

Interactions

asynchronous channels

shared variables

Data

predefined data types
(basic types, arrays, records)

abstract data types

IF notation: System description

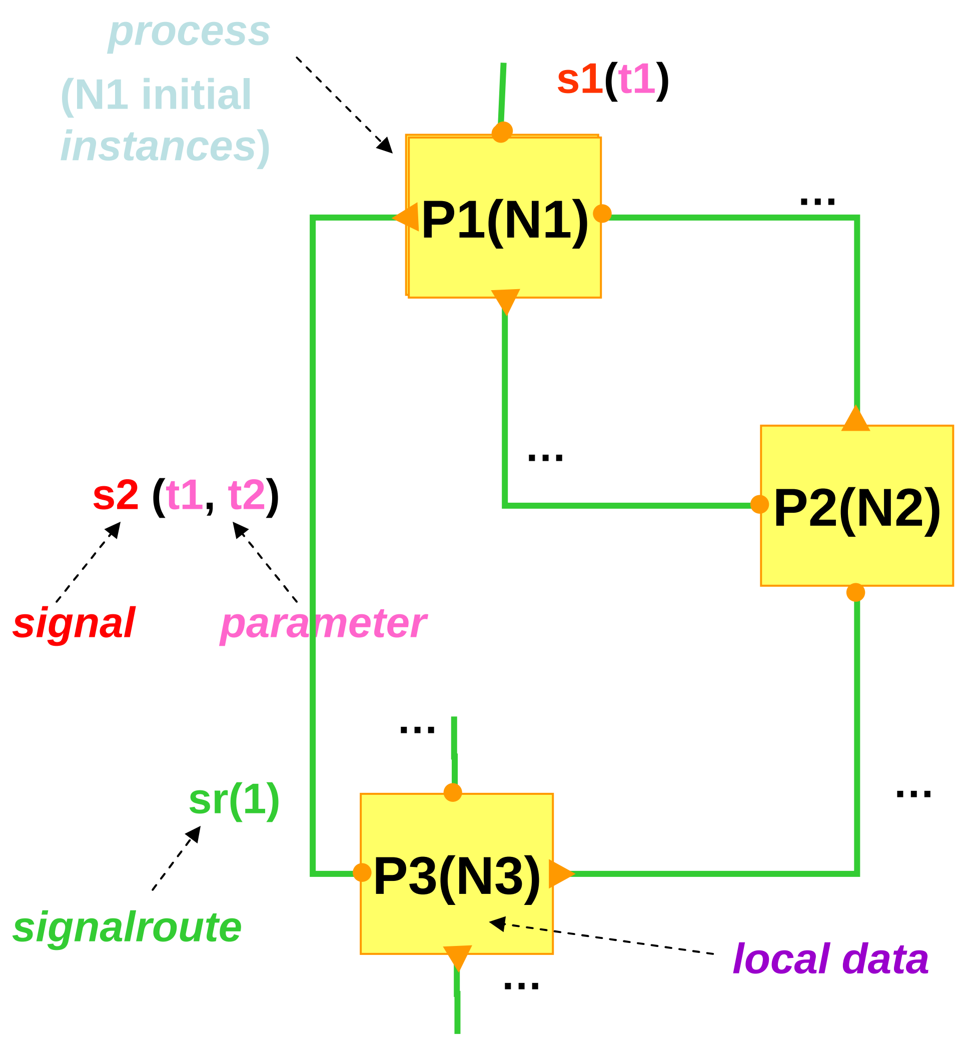
const N1 = ... ;   **//** constants
type t1 = ... ;   // types

signal s2(t1, t2),   // signals

// signalroutes
signalroute sr1(1) ... // route attributes
from P1 to P3

// processes
process P1(N0)
...   // **data** +
behaviour
endprocess;

...
process P3(N3)
...
endprocess;



IF notation: Process description

Process = hierarchical, timed systems with actions

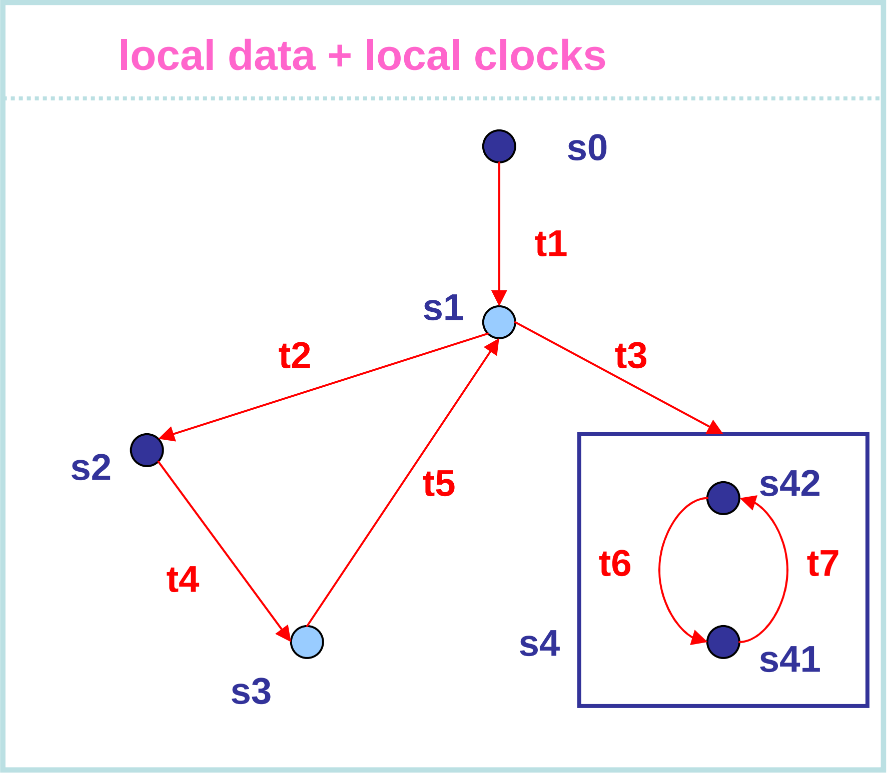
```
process P1(N1);
fpar ... ;                                  <- parameters

                                            <- local data
// types, variables, constants,
procedures

state s0 ... ;                              <- state
        ...  // transition t1
endstate;                                   <- outgoing transitions

state s1 #unstable...;
        ... // transitions t2, t3
endstate;

   ...   // states s2, s3, s4
endprocess;
```



P1(N1)

IF notation: Process description-transition

transition = *urgency* + trigger + body

state s0
…

t1

urgency *eager*
provided x!=10;
when c2 >= 4;
input update(m);
body ….
nextstate s1;

…
endstate;

urgency

untimed guard

timed guard

signal consumption from the process buffer

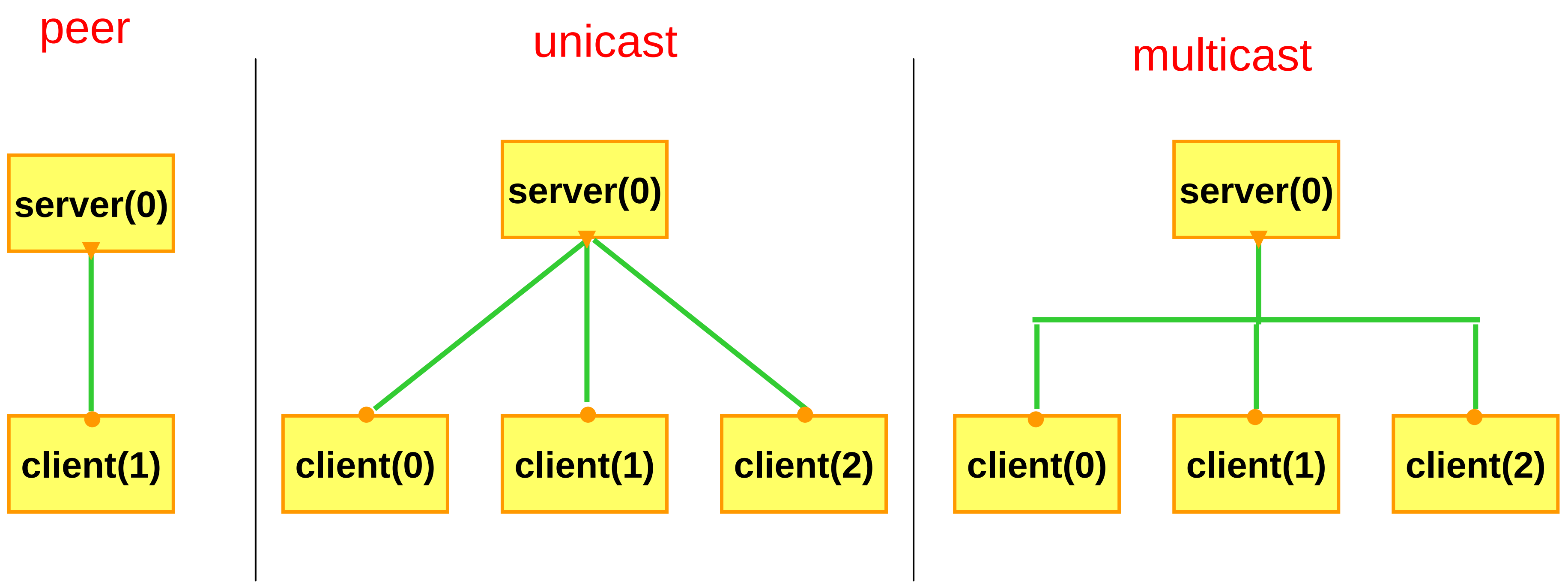
= **trigger**

statement list

statement = data assignment
message emission,
process or signalroute creation or destruction, …

sequential. conditional, or iterative composition

IF notation: interactions - delivery policies



to one specific instance

to a randomly chosen instance

to all instances

IF notation: timed behavior

- operations on clocks
  - set to value
  - deactivate
  - read the value into a variable
- timed guards
  - comparison of a clock to an integer
  - comparison of a difference of two clocks to an integer

```
state send;
  output sdt(self,m,b) to {receiver}0;
  set t:=  10;
  nextstate wait_ack;
endstate;

state wait_ack;
  input ack(sender,c);
  ...
  when 10 <t<20 ;
  ...
endstate;
```

IF notation: dynamic priorities

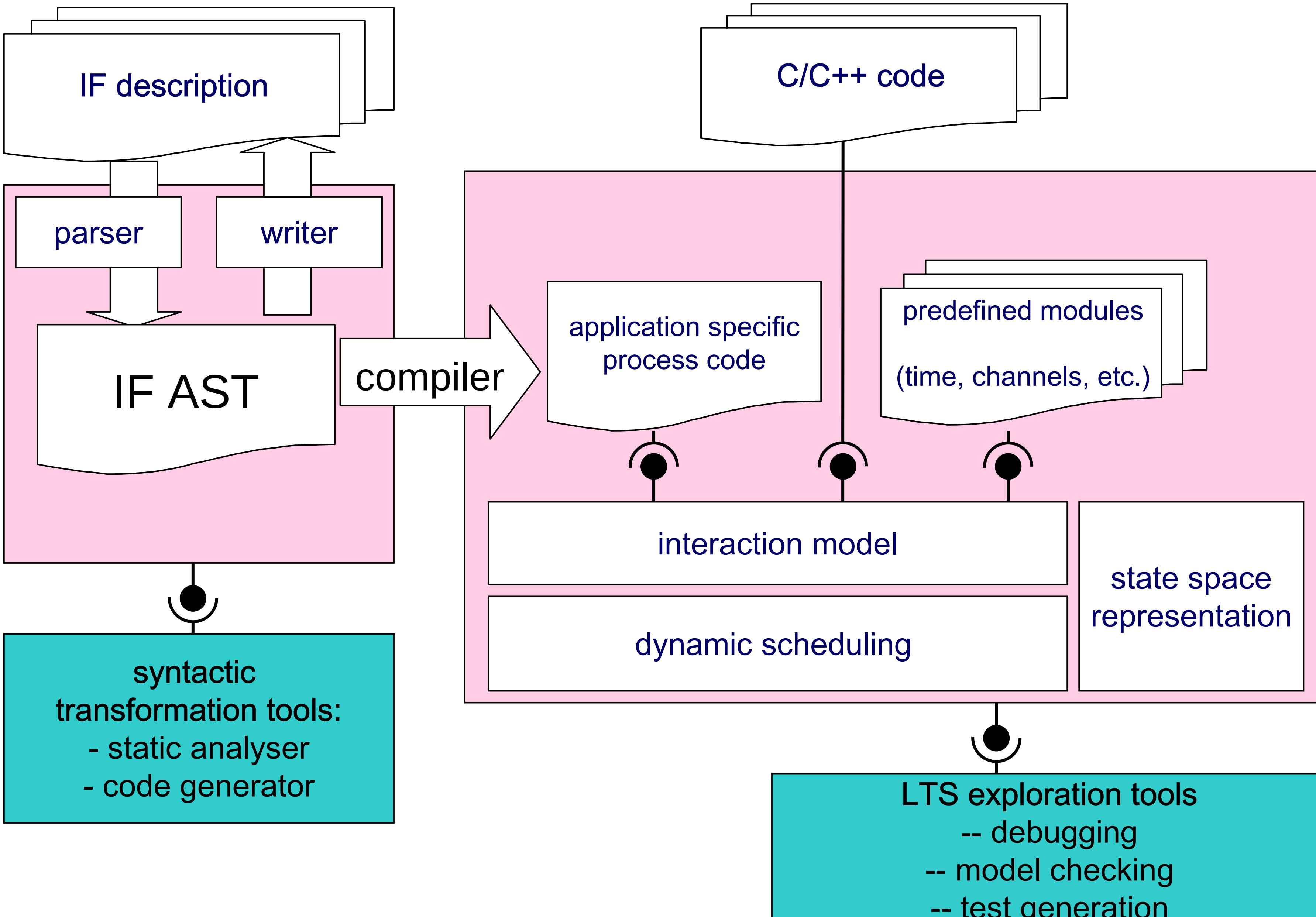
- priority order between process instances p1, p2 (free variables ranging over the active process set)

priority_rule_name : p1 < p2 **if** *condition*(p1,p2)

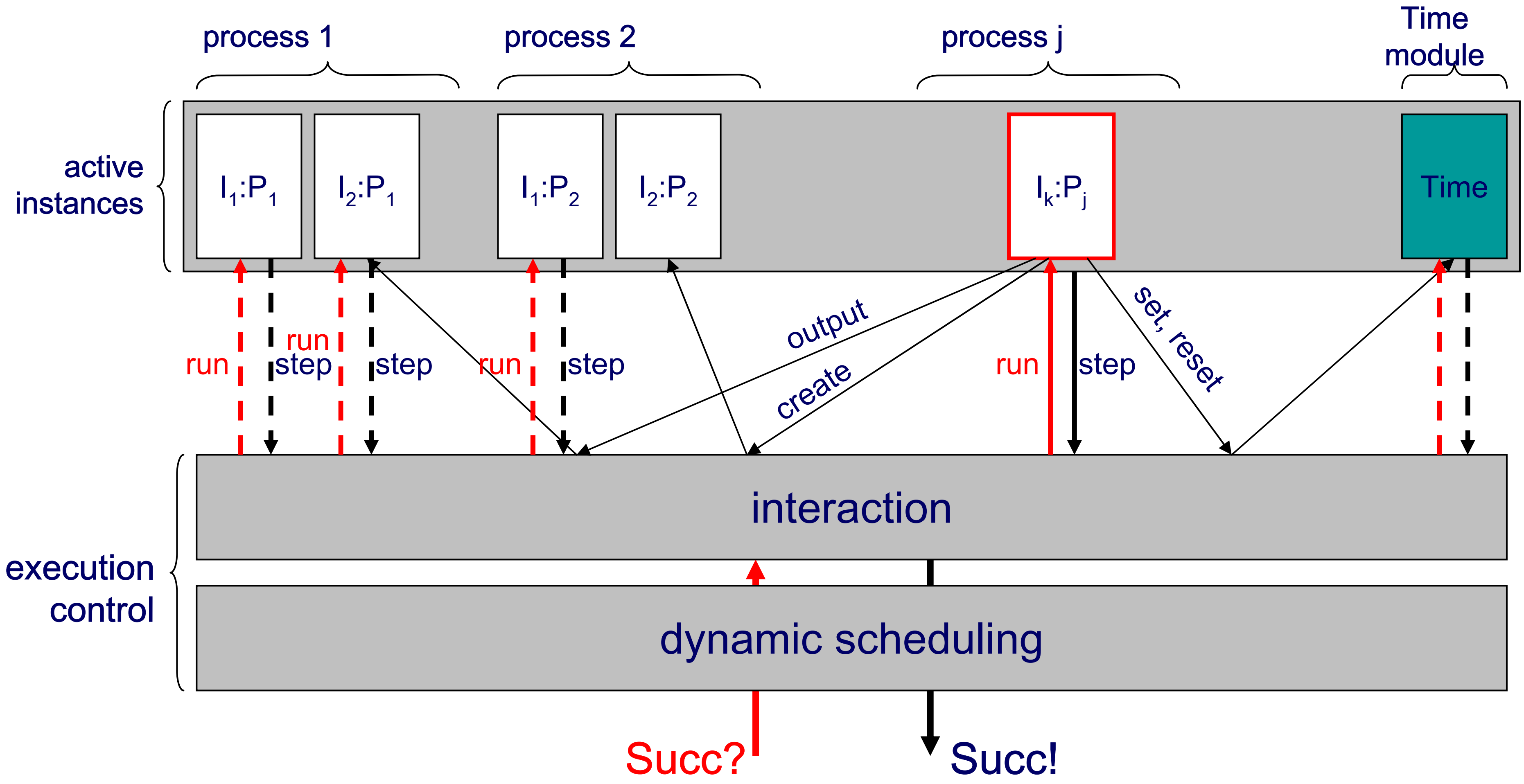
- semantics: *only maximal enabled processes can execute*

- scheduling policies
  - fixed priority: p1 < p2 if p1 instanceof T and p2 instanceof R
  - run-to-completion: p1 < p2 if p2 = manager(0).running
  - EDF: p1 < p2 if Task(p2).timer < Task(p1).timer (p1)

Core components



Core components: exploration platform



Case studies: protocols

SSCOP

Service Specific Connection Oriented Protocol

M. Bozga et al. **Verification and test generation for the SSCOP Protocol**. In *Journal of Science of Computer Programming - Special Issue on Formal Methods in Industry*. Vol. 36, number 1, January 2000.

MASCARA

Mobile Access Scheme based on Contention and Reservation for ATM

case study proposed in VIRES ESPRIT LTR

S. Graf and G. Jia. **Verification Experiments on the Mascara Protocol**. In M.B. Dwyer (Ed.) *Proceedings of SPIN Workshop 2001, Toronto, Canada*. LNCS 2057.

PGM

Pragmatic General Multicast

case study proposed in ADVANCE IST-1999-29082

Case studies: asynchronous circuits

timing analysis

O. Maler et al. **On timing analysis of combinational circuits**. In *Proceedings of the 1st workshop on formal modeling and analysis of timed systems, FORMATS'03, Marseille, France*.

functional validation

D. Borrione et al. **Validation of asynchronous circuit specifications using IF/CADP**. In *Proceedings of IFIP Intl. Conference on VLSI, Darmstadt, Germany*

Ariane 5 Flight Program

joint work with EADS Lauchers

M. Bozga, D. Lesens, L. Mounier. **Model-checking Ariane 5 Flight Program**. In *Proceedings of FMICS 2001, Paris, France*.

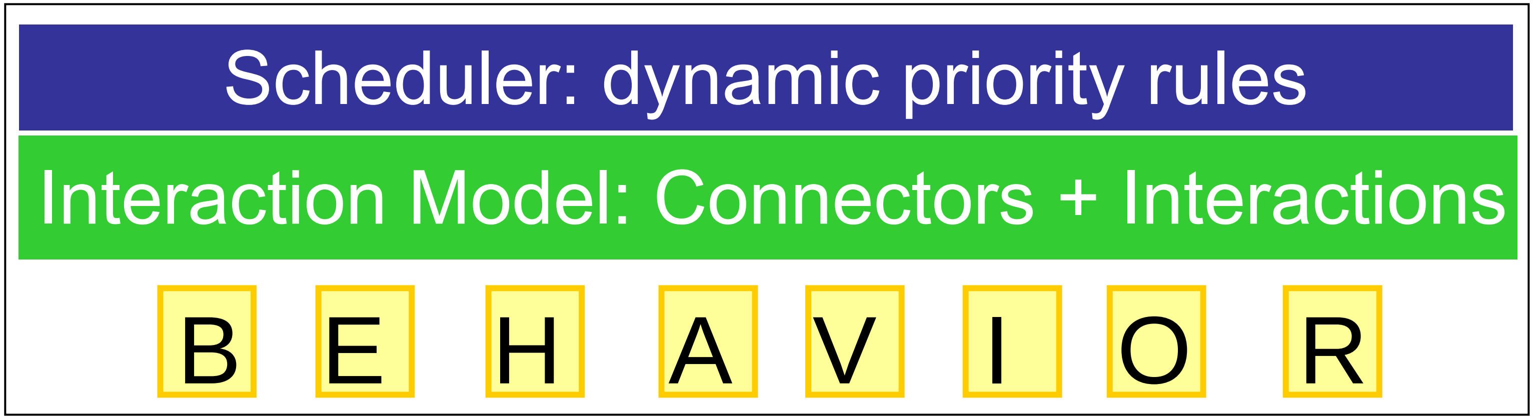
K9 Rover Executive

S.Tripakis et al. **Testing conformance of real-time software by automatic generation of observers**. In *Proceedings of Workshop on Runtime Verification, RV'04, Barcelona, Spain*.

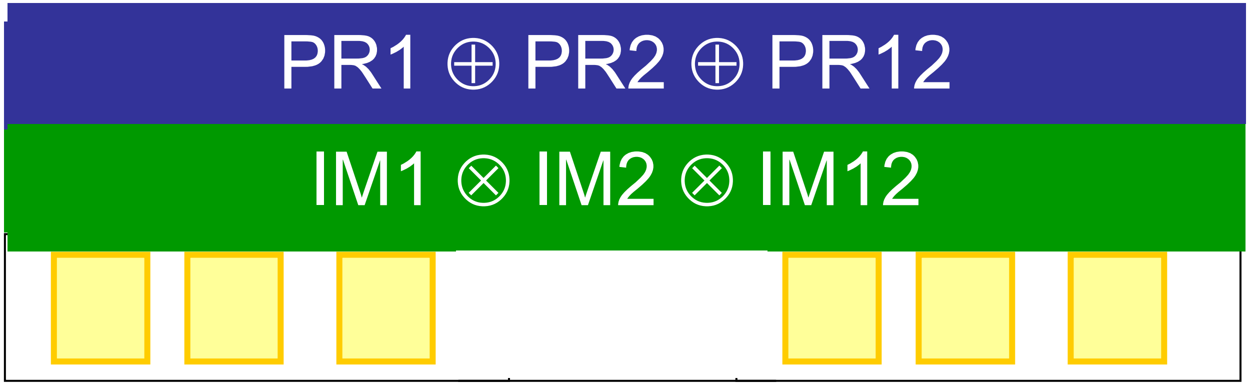
Akhavan et al. **Experiment on Verification of a Planetary Rover Controller**. In *Proceedings of 4th International Workshop on Planning and Scheduling for Space, IWPSS'04, Darmstadt, Germany.*

Component-based modeling – The BIP framework

Layered component model



Composition (incremental description)



Component-based modeling - the BIP framework

Encompass heterogeneity of interaction and execution and rely on a **minimal set of constructs and principles** e.g. interaction models + dynamic priorities

Clear separation between behavor and architecture.

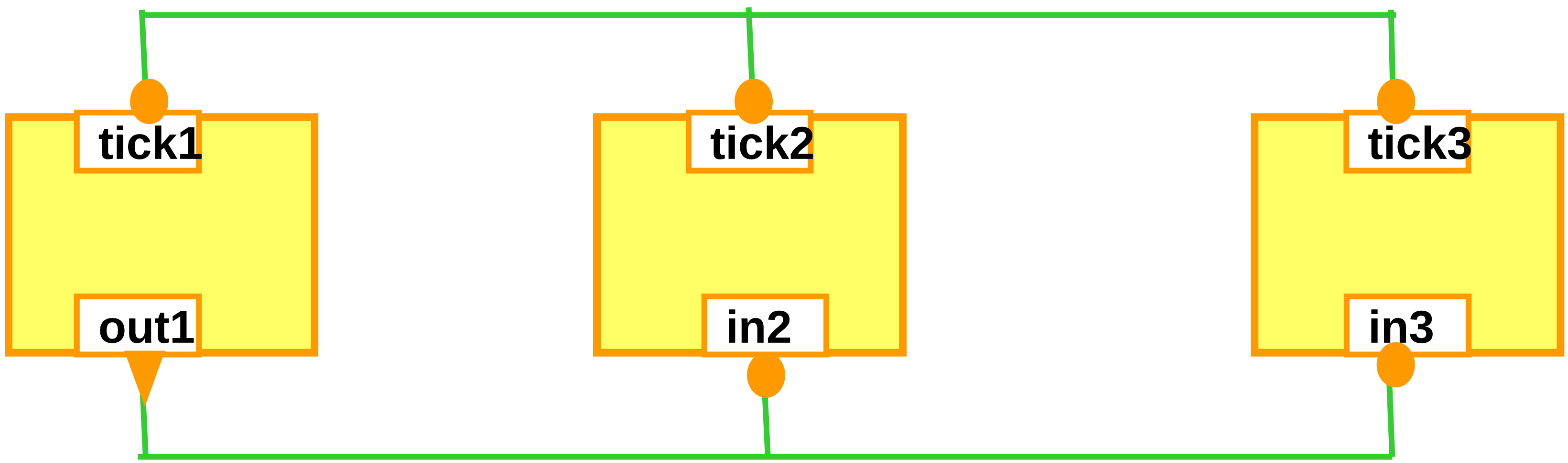
- Architecture is a first class entity
- Correctness-by-construction techniques for deadlock-freedom and liveness, based on suficient conditions on structure (mainly)

Development of a language independent execution/state exploration platform

Interaction models

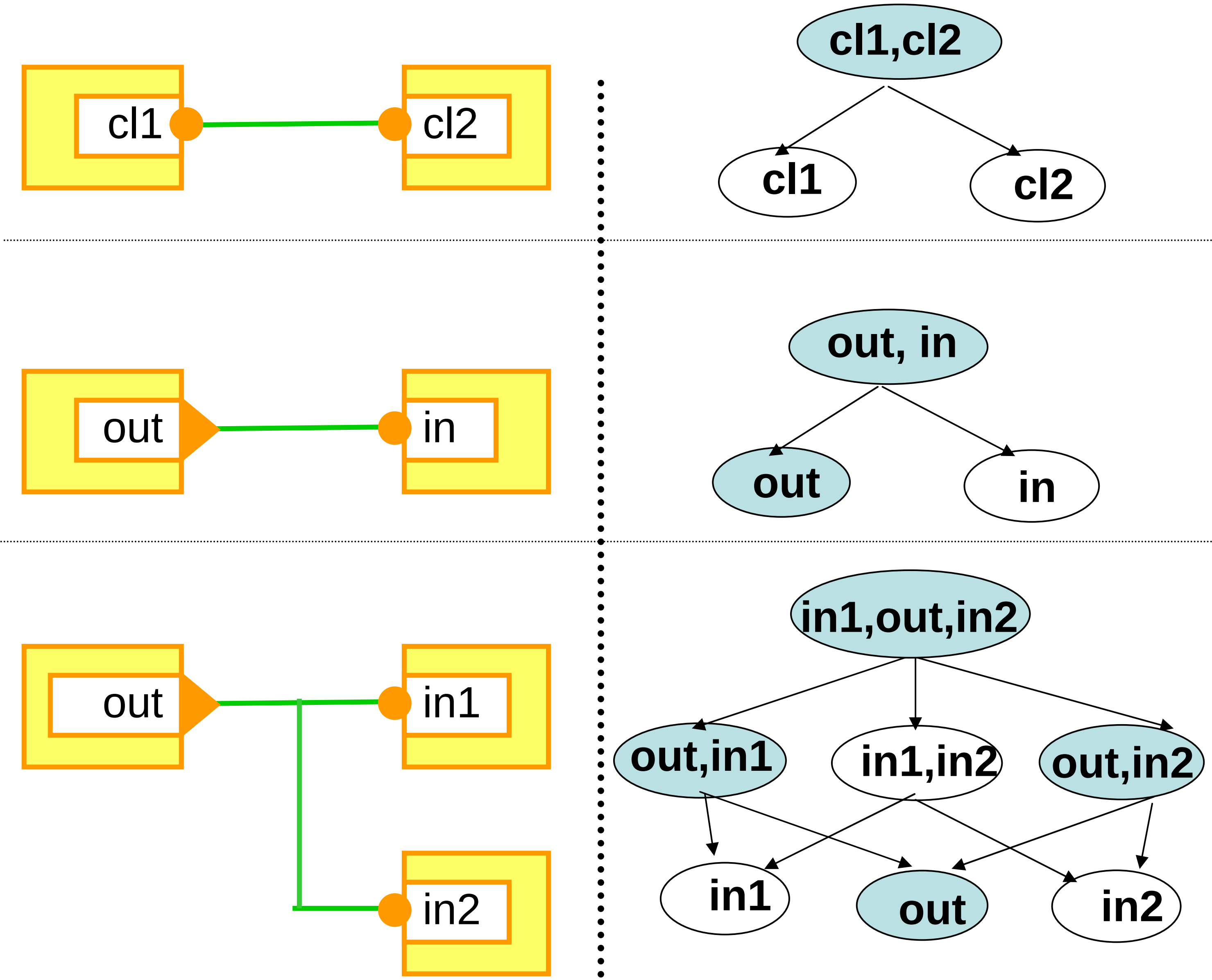
Connectors are maximal sets of compatible actions

Interactions are subsets of connectors; they are defined by using typing **(complete ▼ , incomplete ●) :** either they are maximal or they contain some complete interaction



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

Interaction models - examples

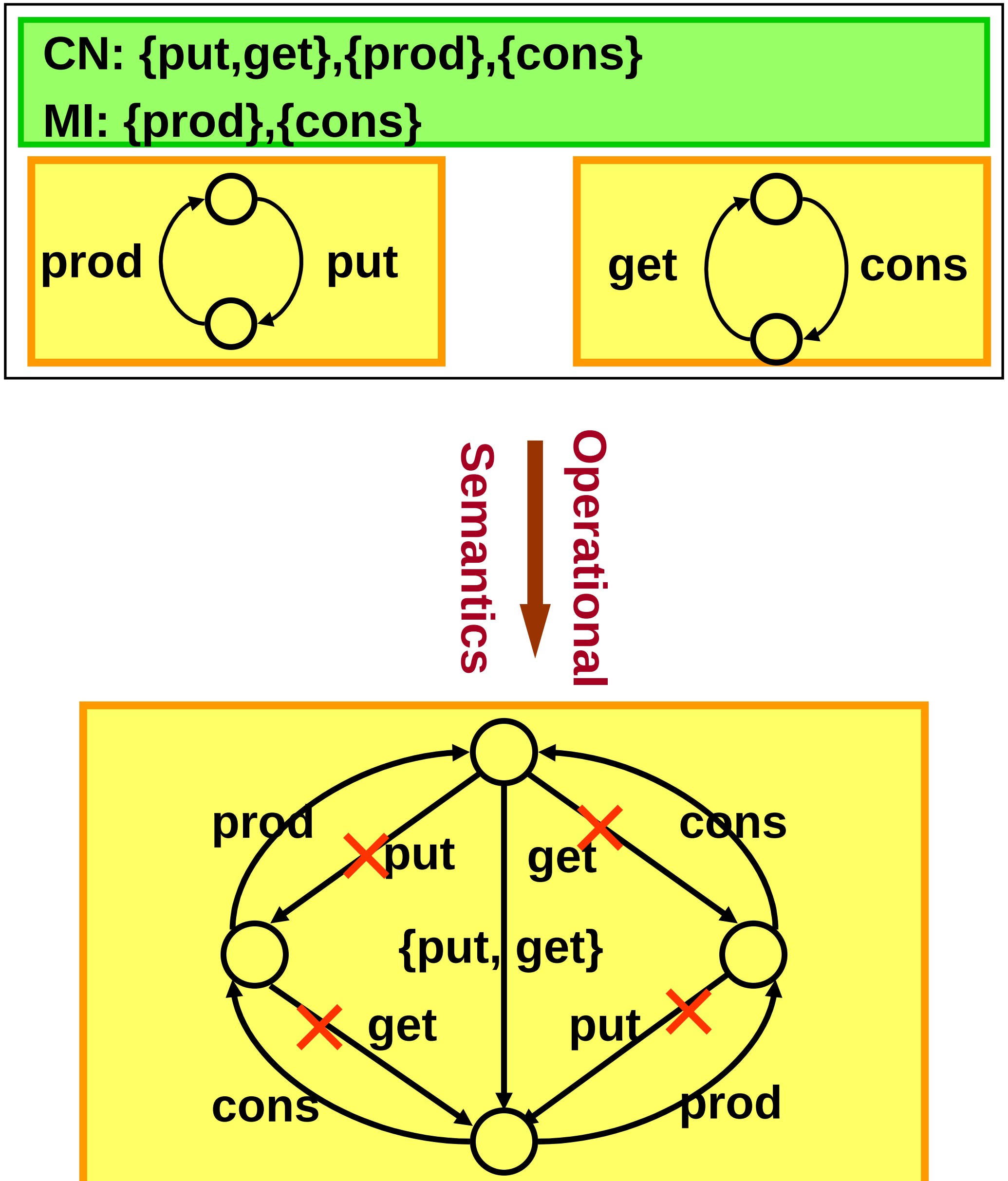


CN:{cl1,cl2}
MI: ∅

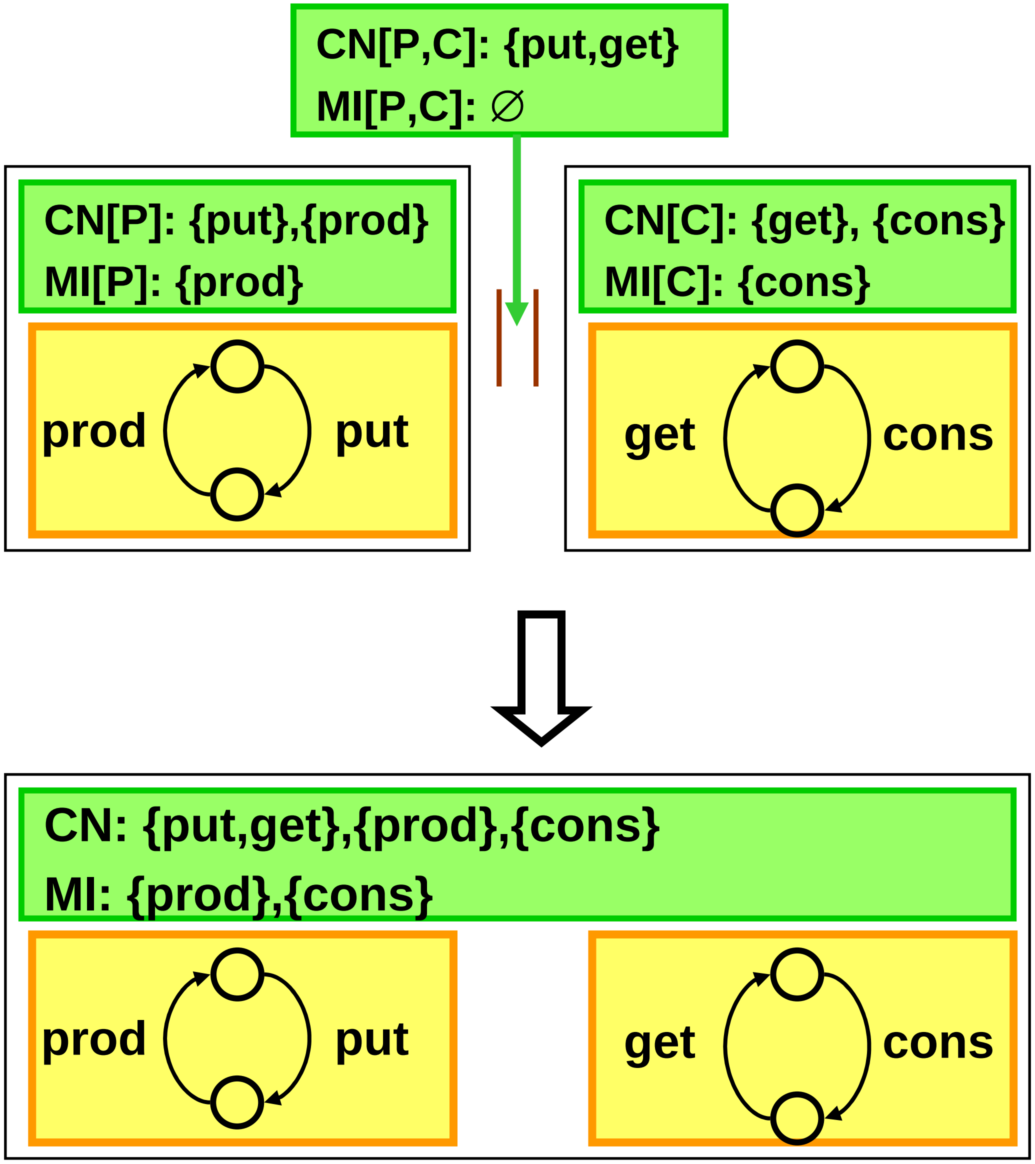
CN:{out,in}
MI: {out}

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

Interaction models – operational semantics

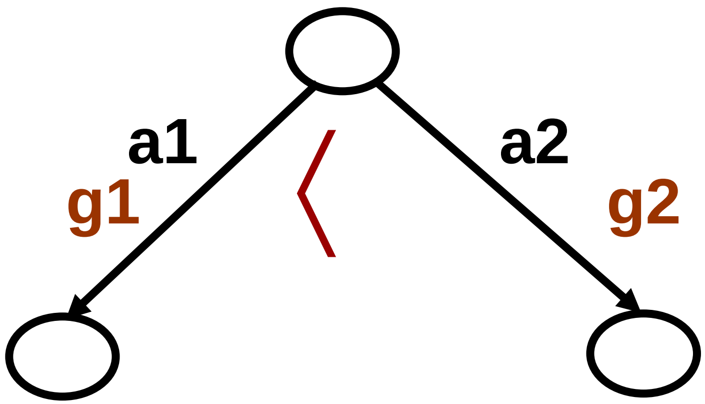


Interaction models - composition



Priority Systems

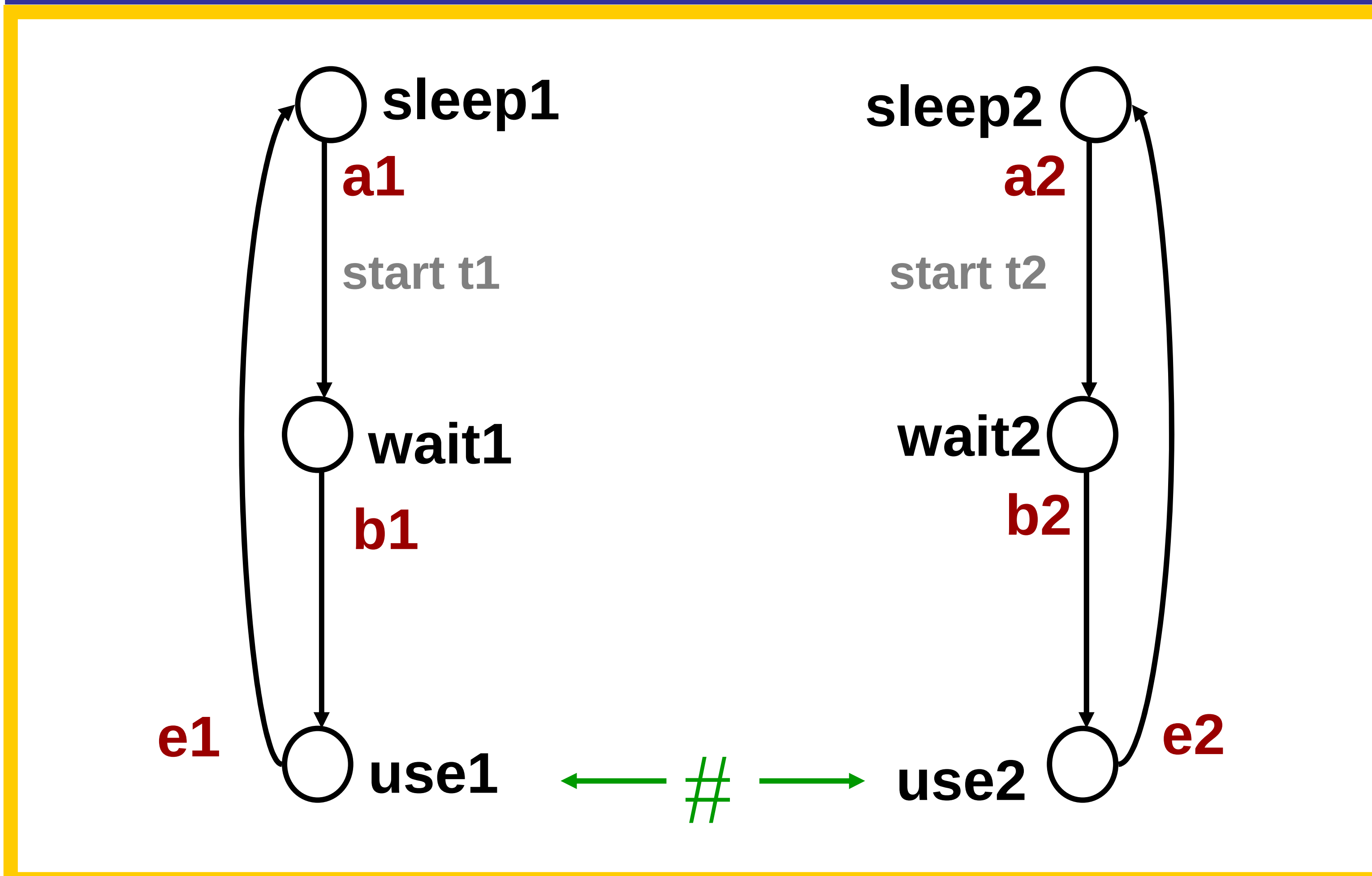
Priority system = Behavior + A set of dynamic priority rules



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

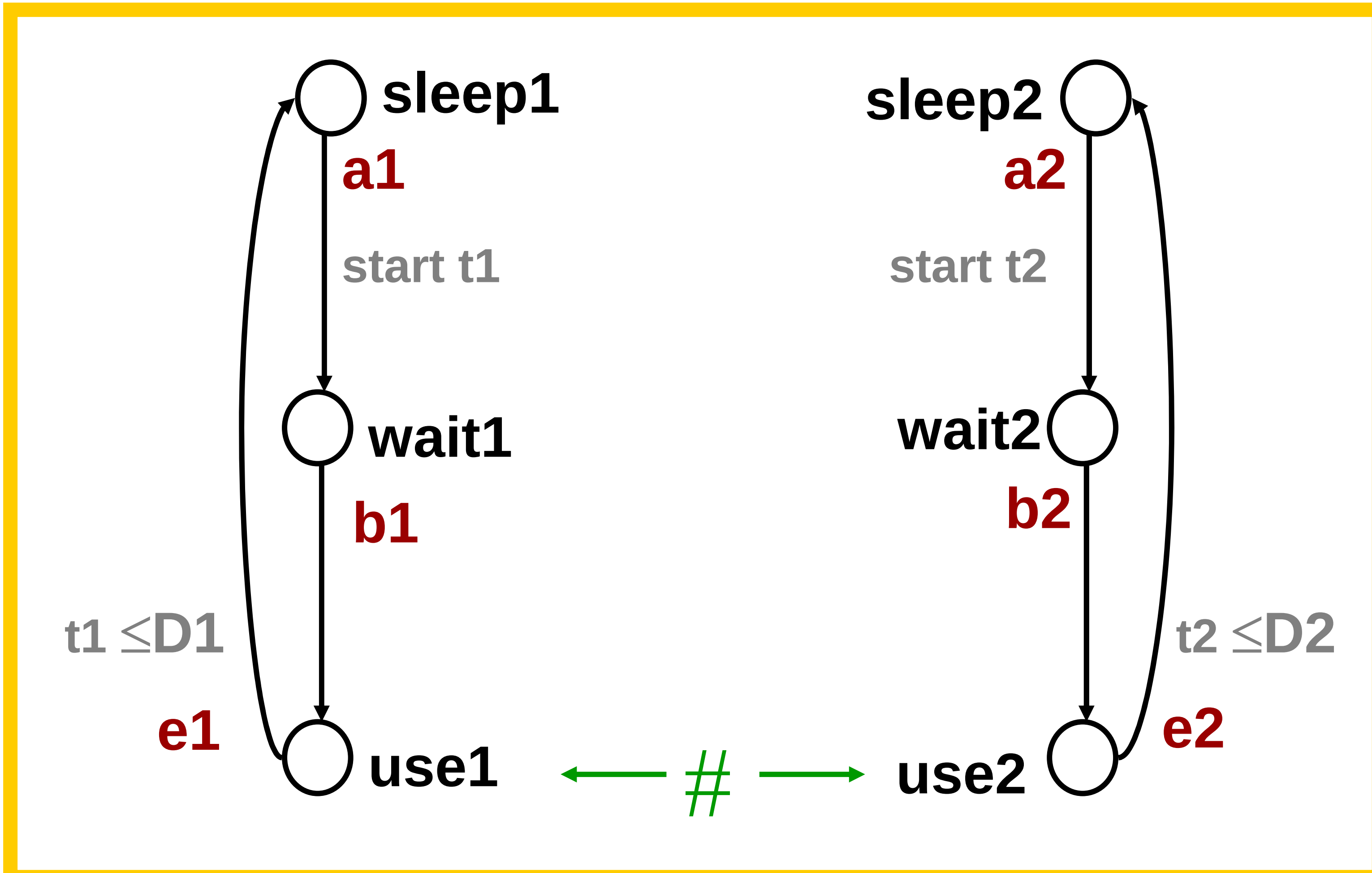
Priority Systems - FIFO policy

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

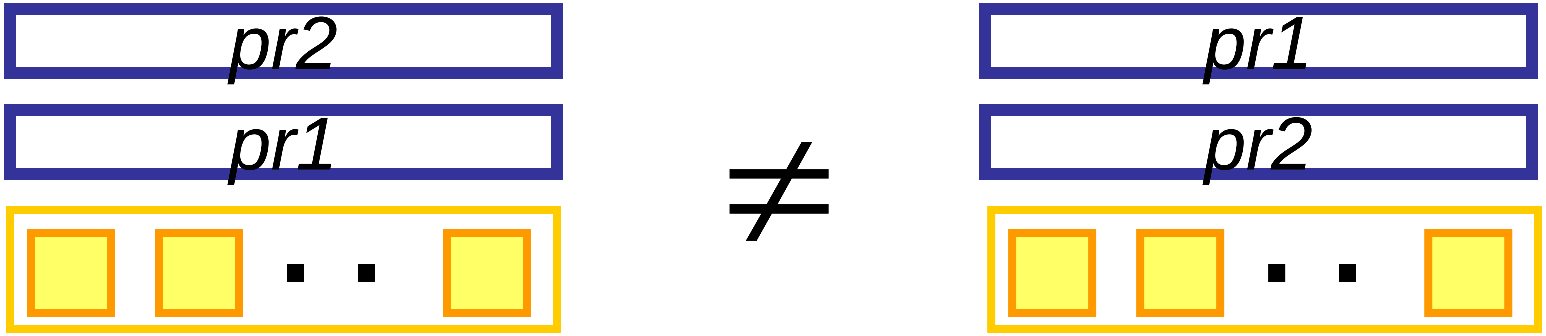


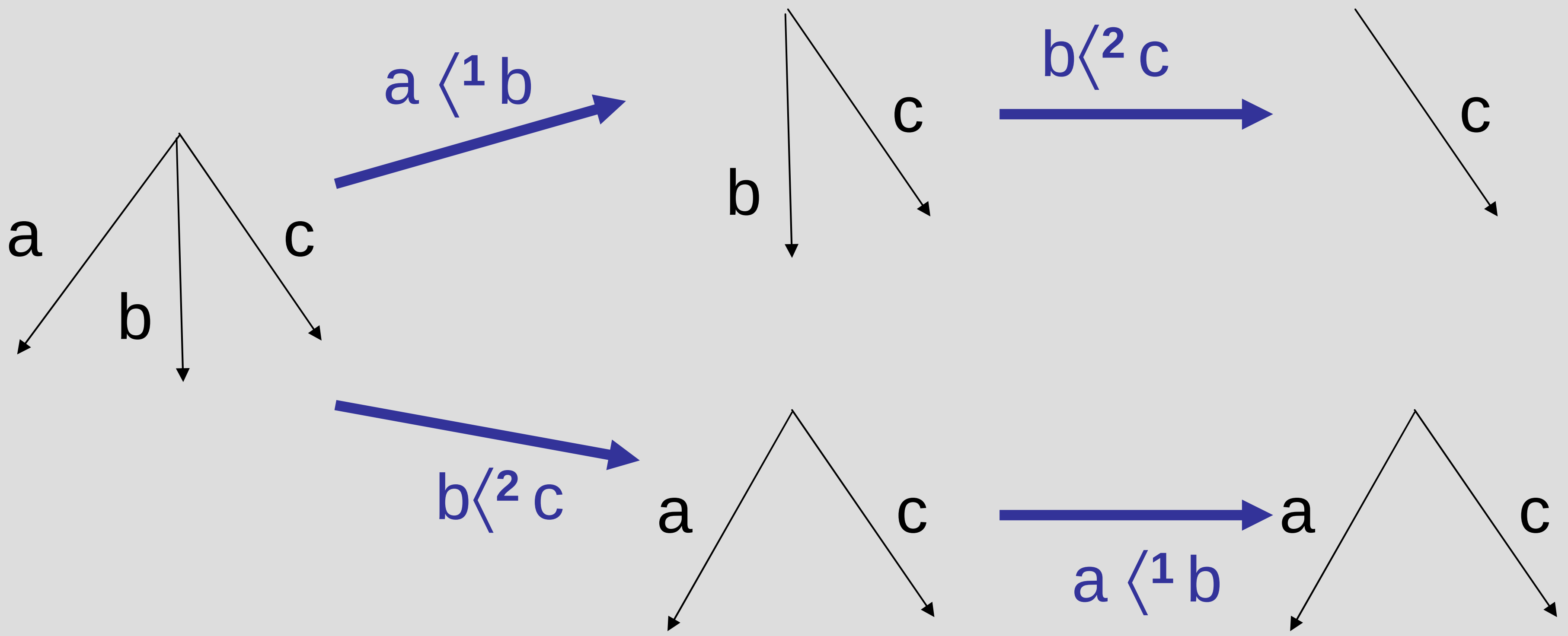
Priority Systems - EDF policy

$D1-t1 \leq D2-t2 \rightarrow b2 \langle b1$ $D2-t2 \leq D1-t1 \rightarrow b1 \langle b2$

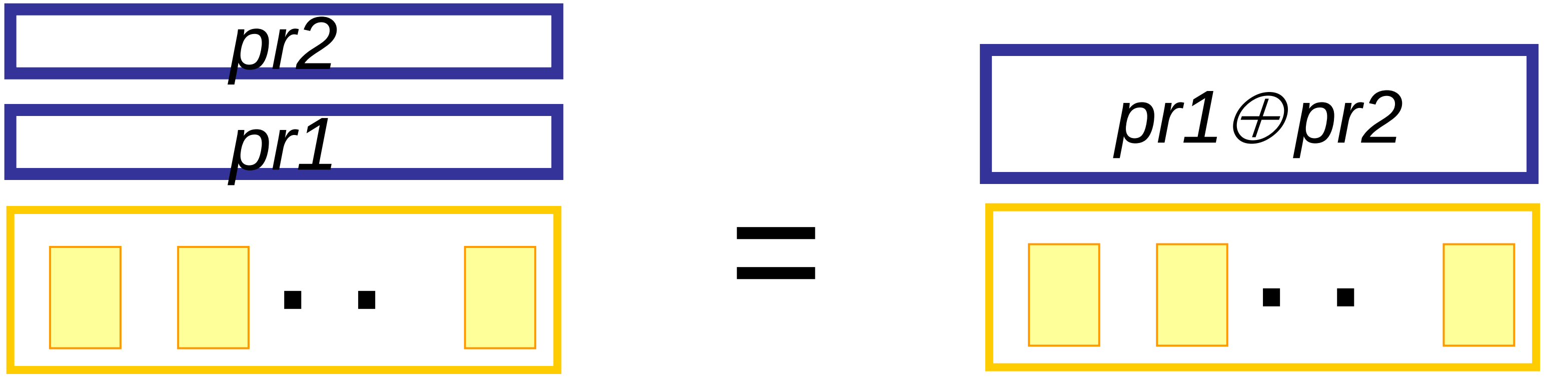


Priority Systems - Composition of priorities





We take:



pr1⊕ pr2 is the least priority containing pr1∪pr2

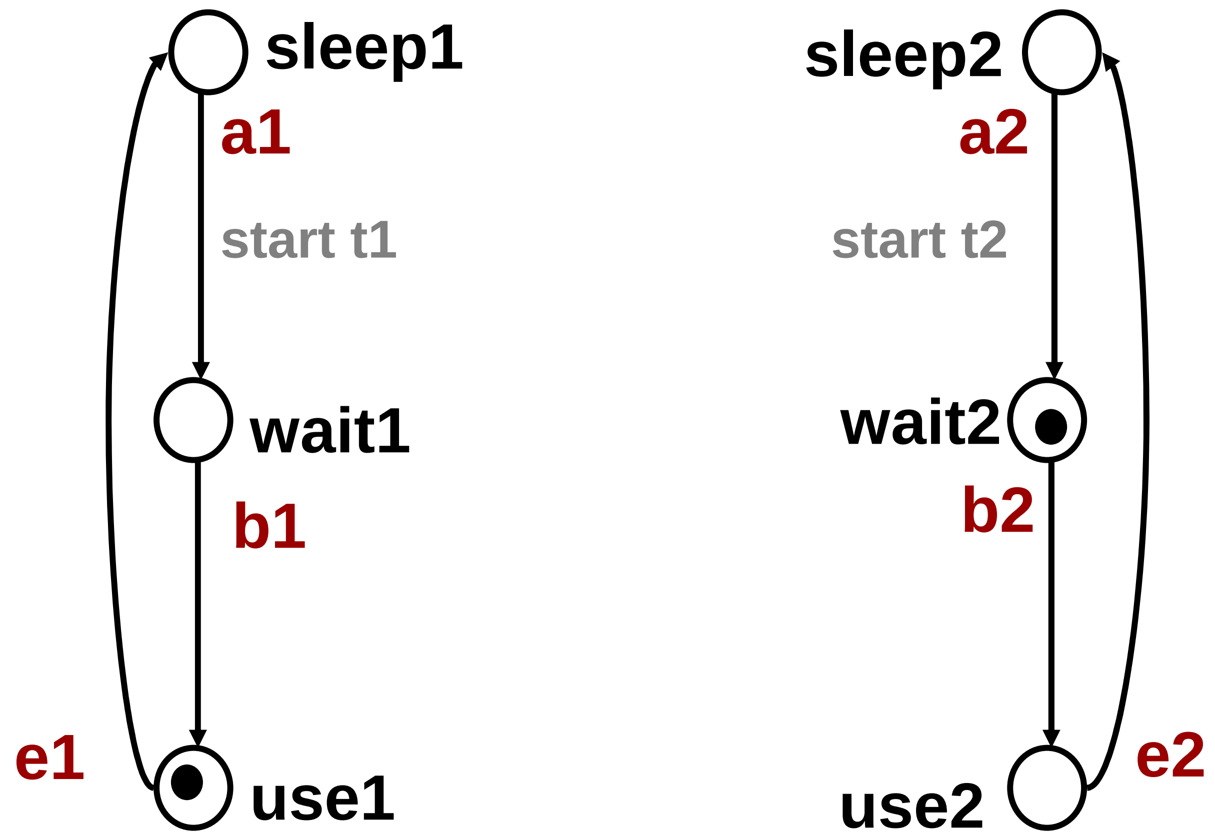
Results :

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

Priority Systems : mutual exclusion + FIFO

t1≤ t2 → b1〈 b2   t2≤ t1 → b2〈 b1

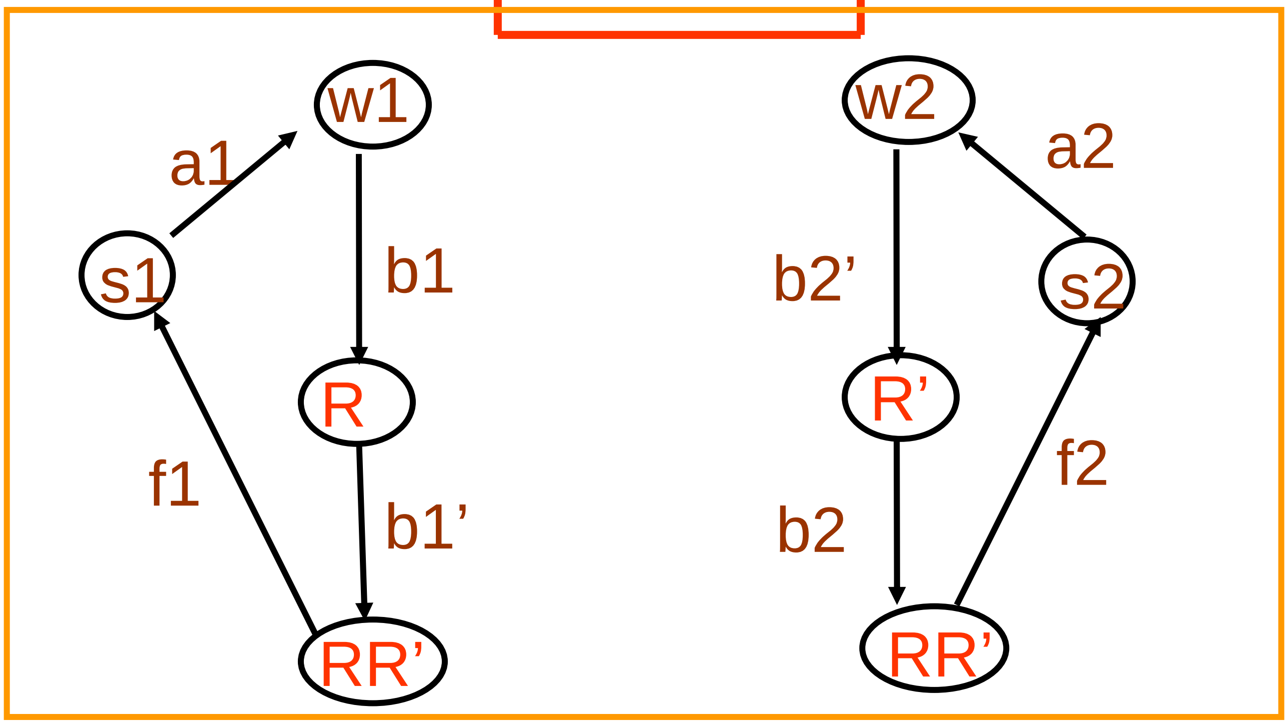
true → b1〈 e2   true → b2〈 e1



Priority systems – mutual exclusion

Mutex on R : b1 ⟨ f2 b2 ⟨ { f1, b1'}

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



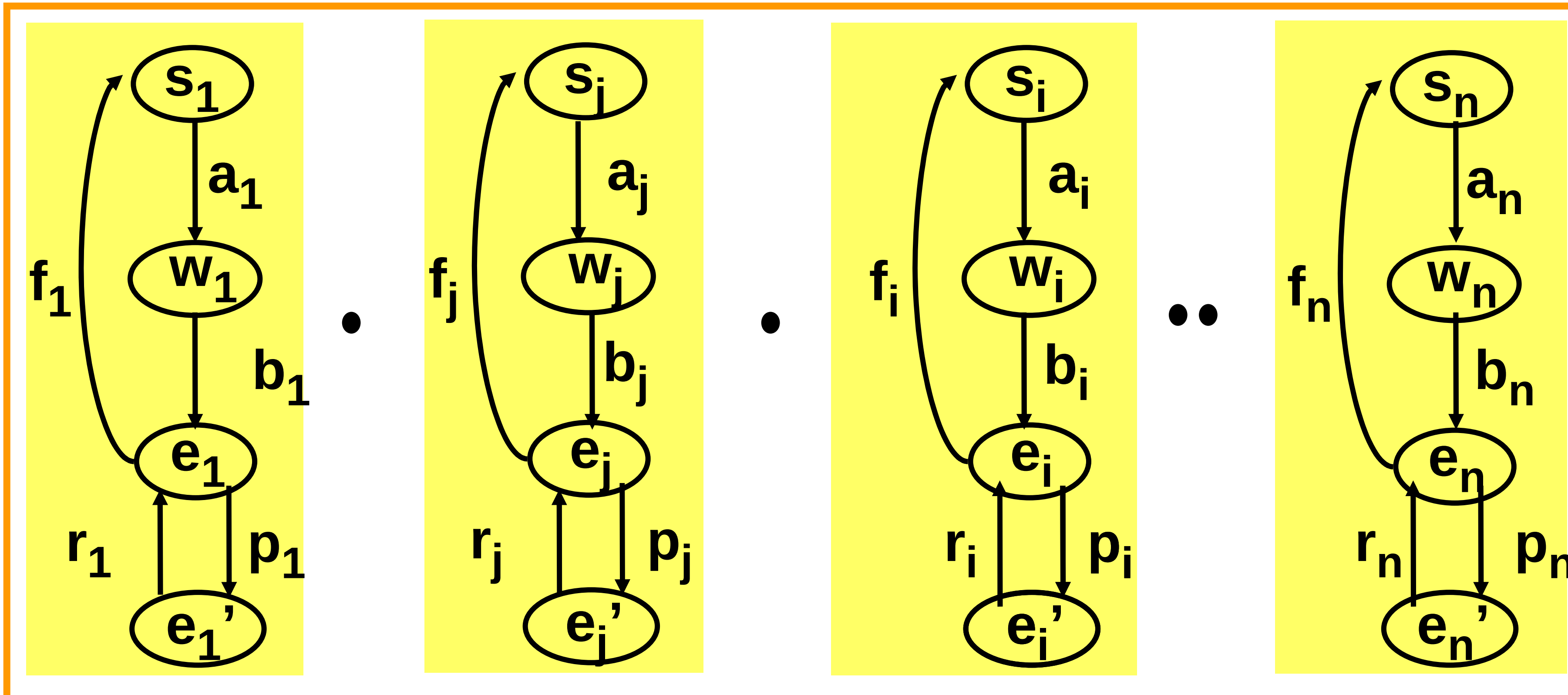
Risk of deadlock: The composition is not a priority order !

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 $n \geq i,j \geq 1$ **MI: a_i, f_i, b_i** for $n \geq i \geq 1$

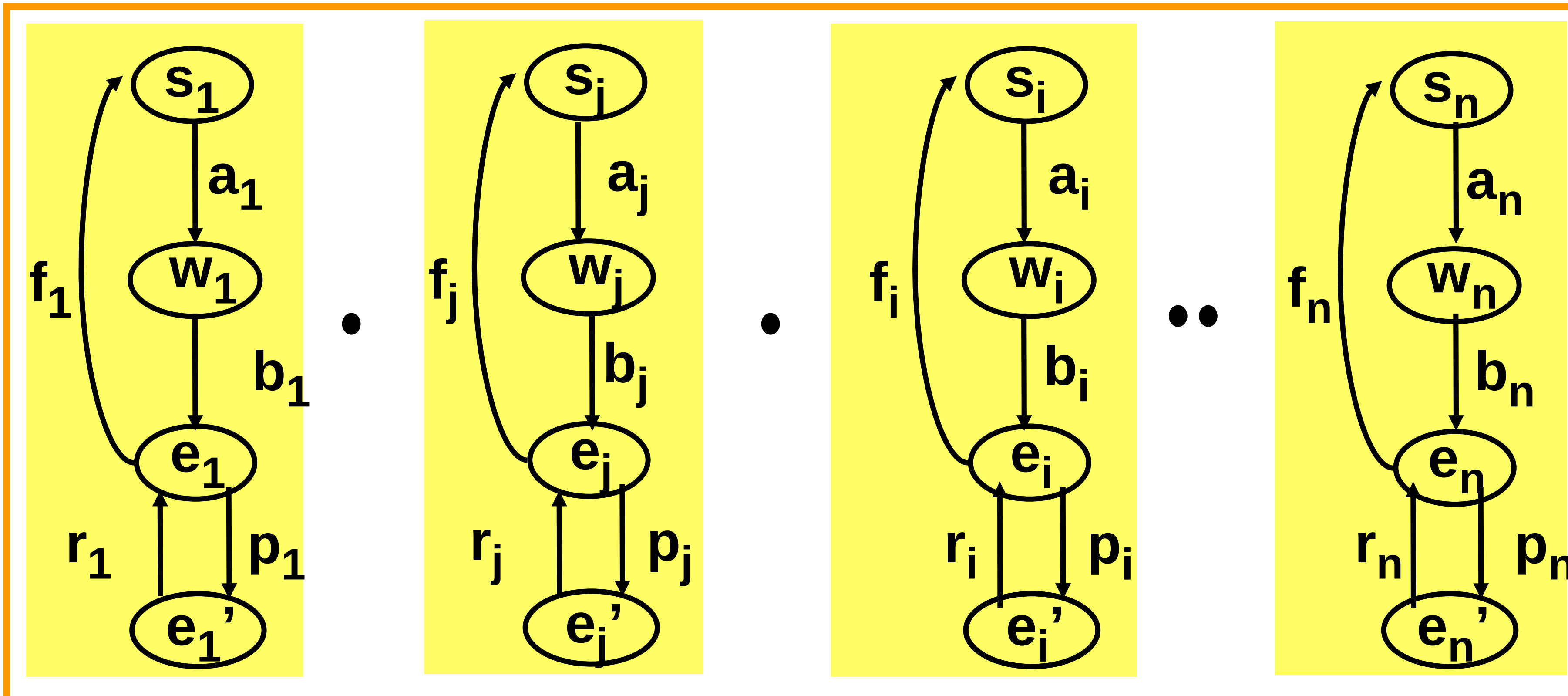


The BIP framework - fixed priority preemptive scheduling (2)

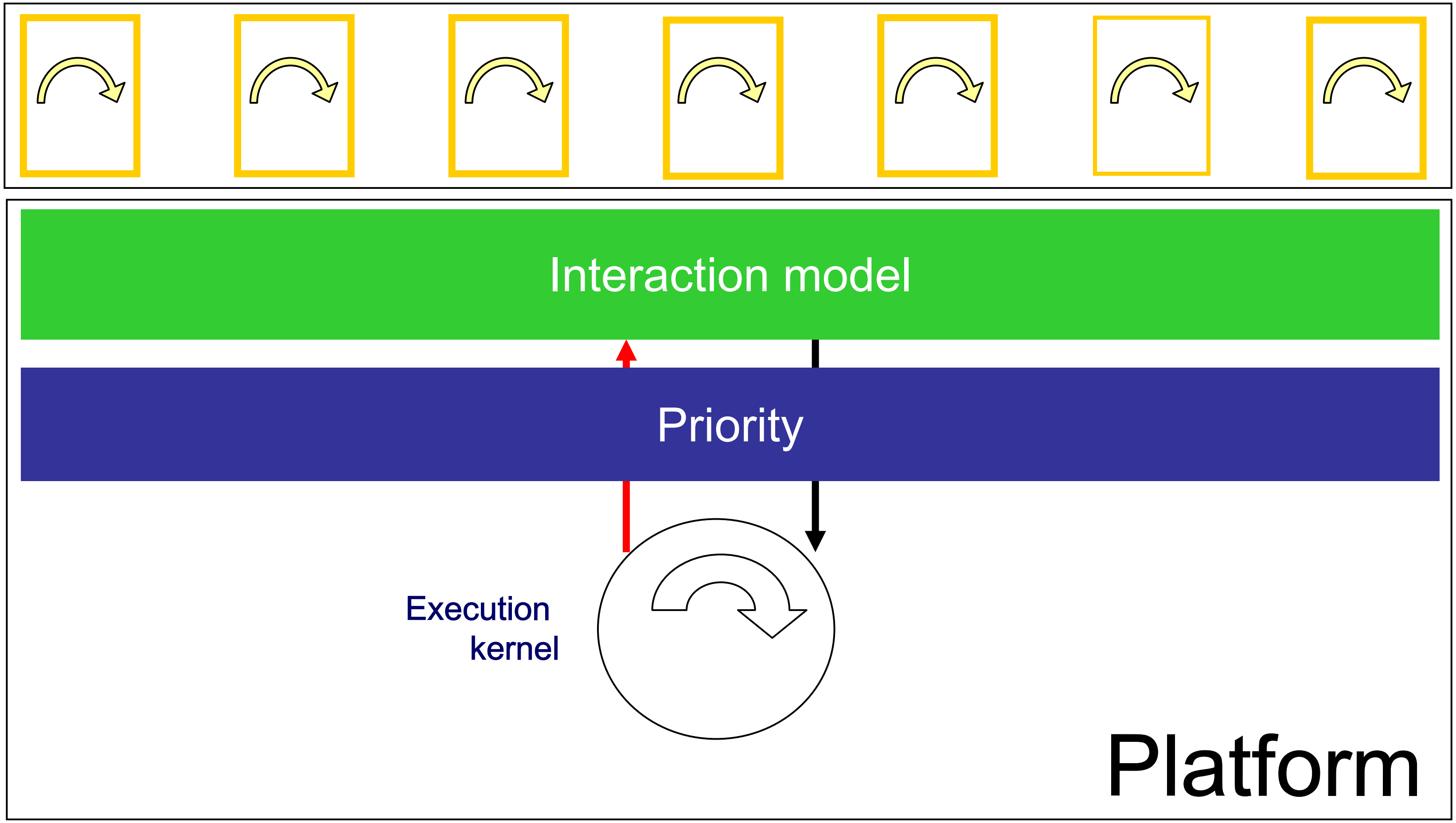
$b_i \langle b_j$, $r_i \langle r_j$, $r_i \langle b_j$, $b_i \langle r_j$ (access to the ressource – pty inherited by composition)

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

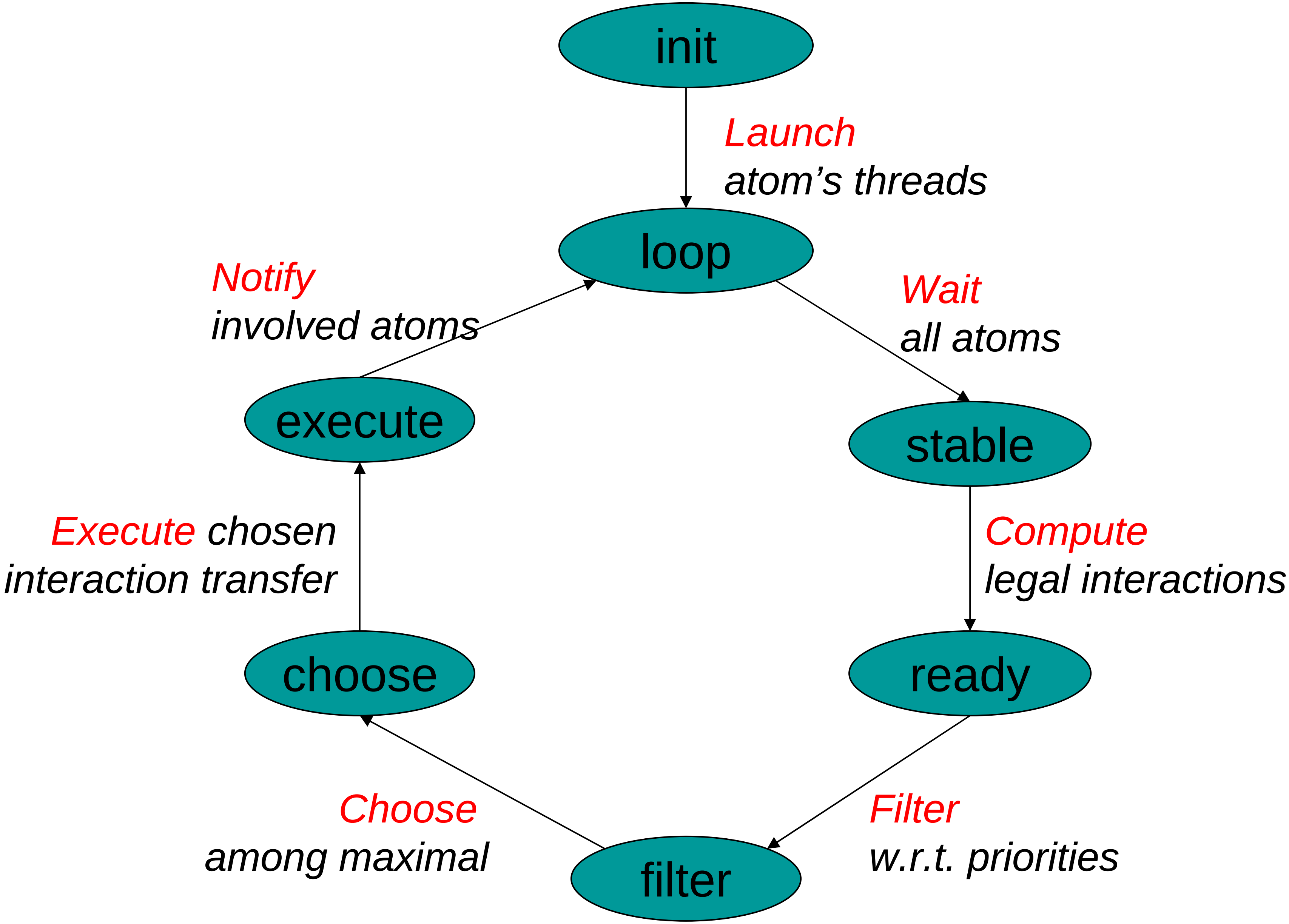
$\{b_i , r_i\} \langle \{f_j \ p_j\}$ $n \geq l, j \geq 1$ (Mutual exclusion)



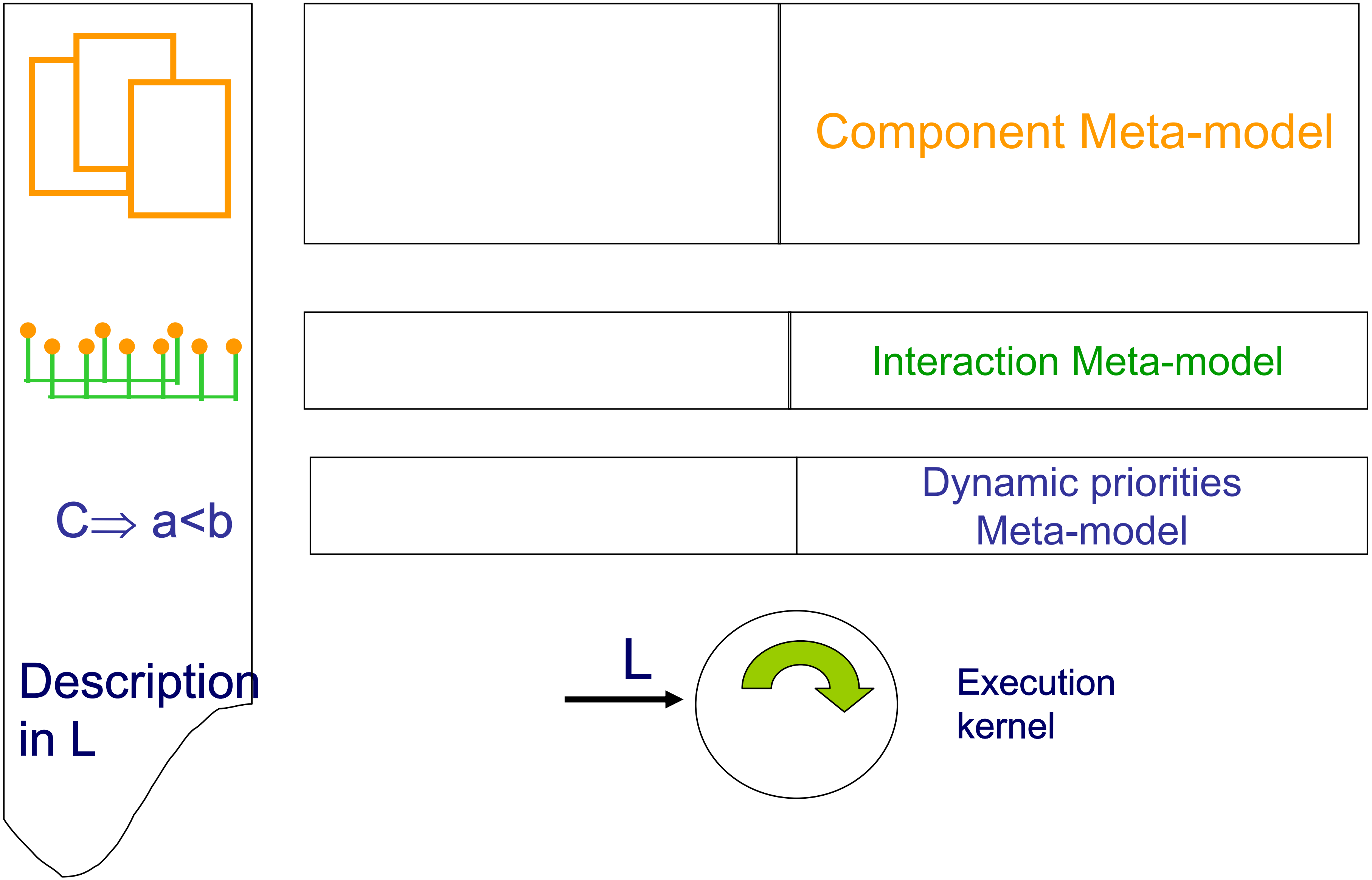
The execution platform



The BIP framework – implementation: the kernel



The execution platform



Implementation - 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 …..

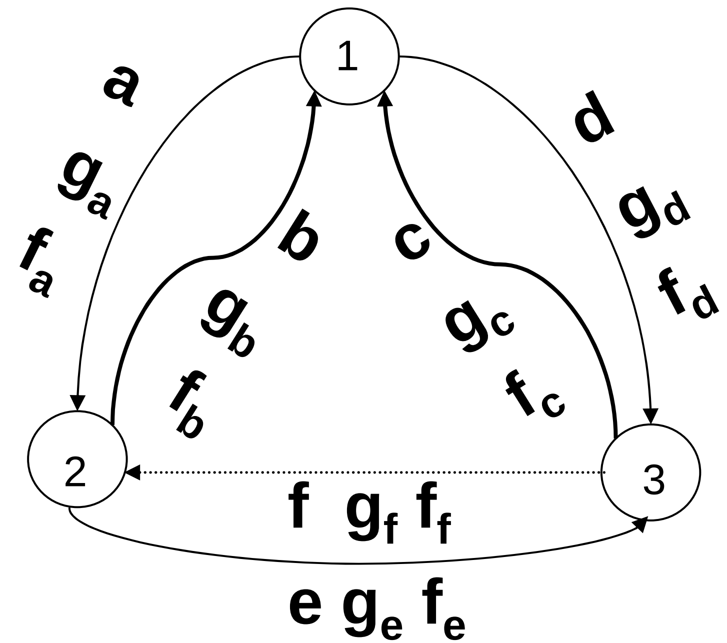
        ……….

    state sn

        on .. ..

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:

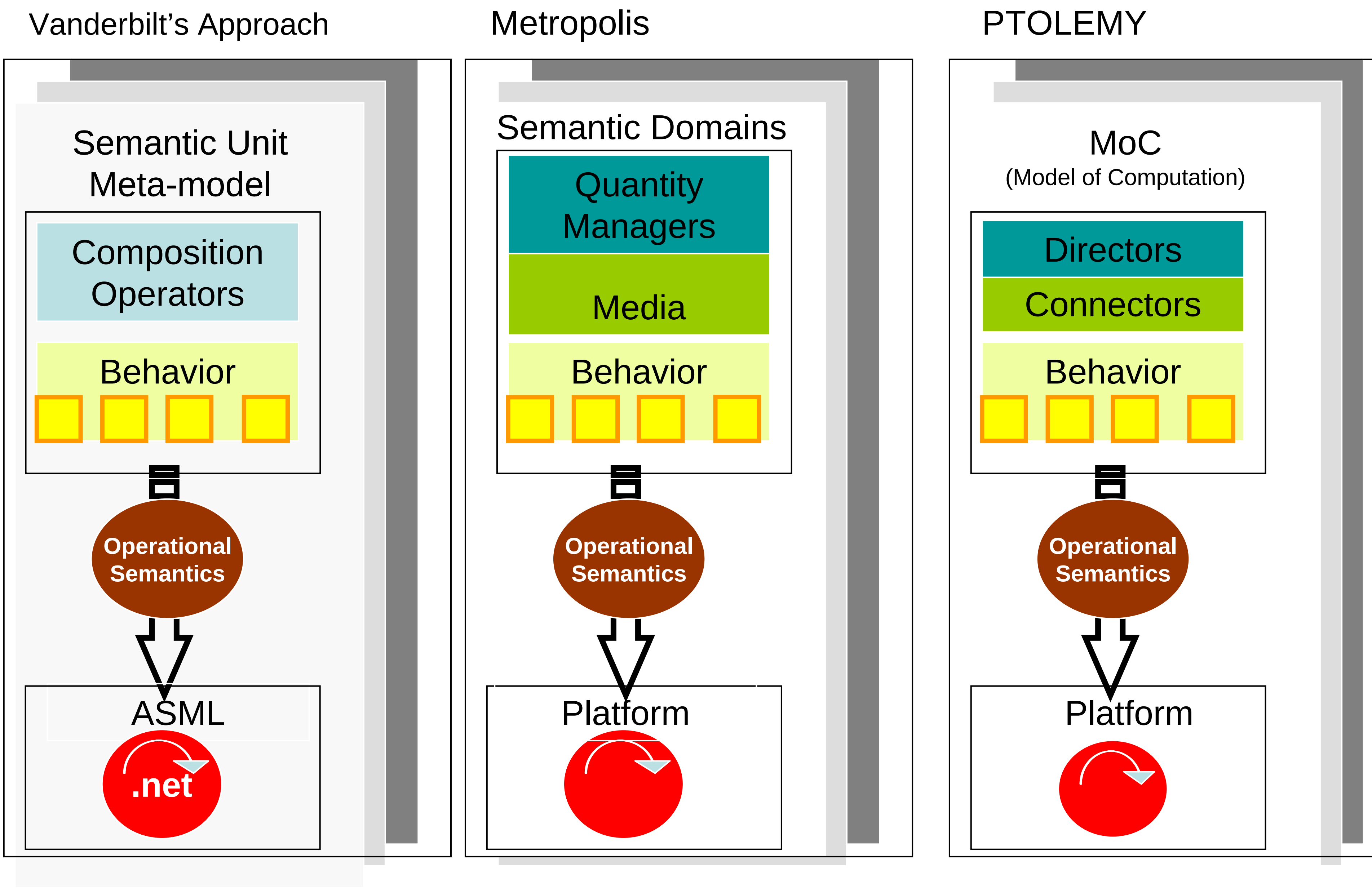
on $\alpha 1$ **provided** $g_{\alpha 1}$ **do** $f_{\alpha 1}$
on $\alpha 2$ **provided** $g_{\alpha 2}$ **do** $f_{\alpha 2}$

Priorities: PR

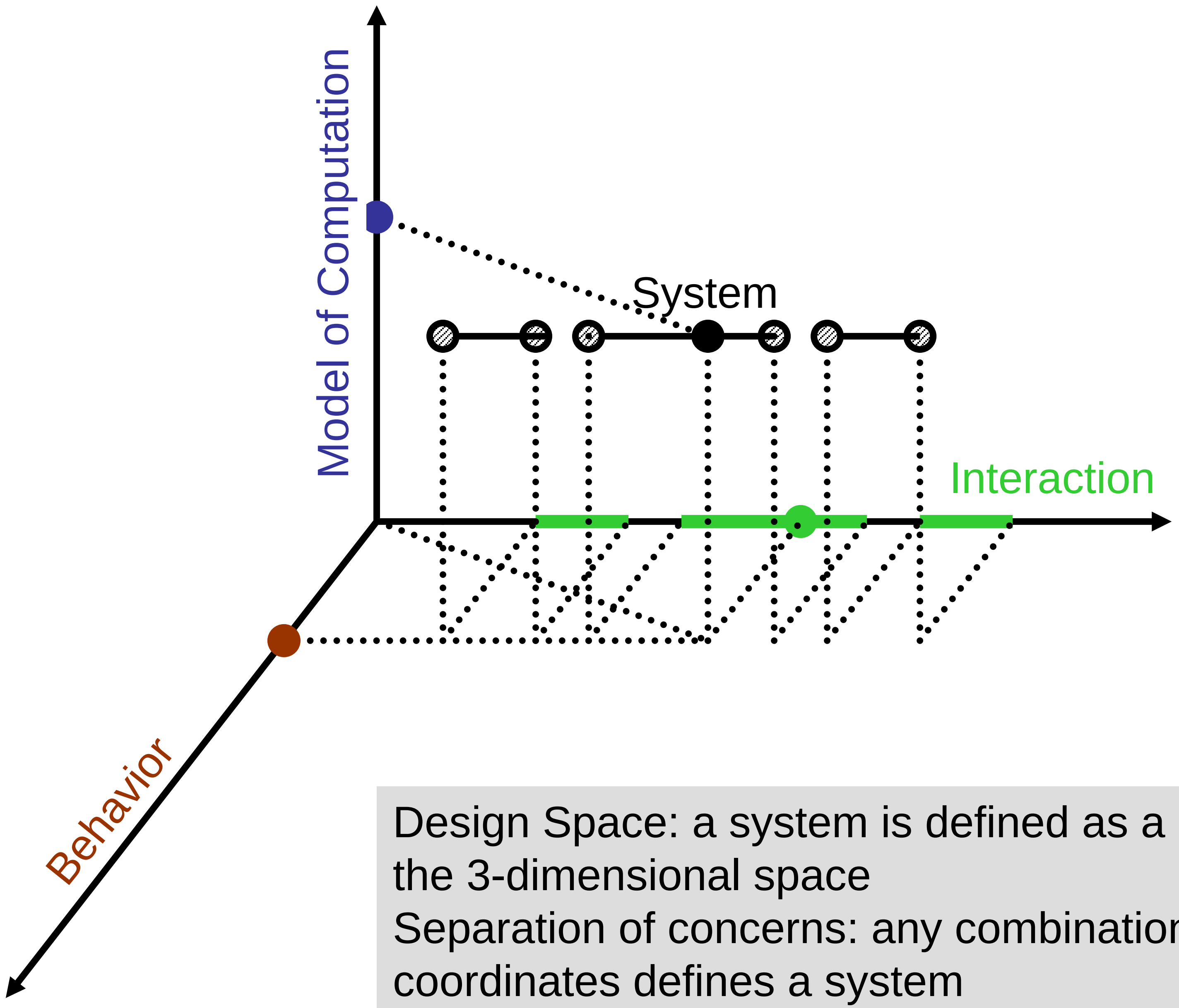
if C1 **then** {(α1, α2), (α3, α4) , … }
if C2 **then** {(α,…), (α,…) , … }

if Cn **then** {(α,…), (α,…) , … }

Discussion – related approaches

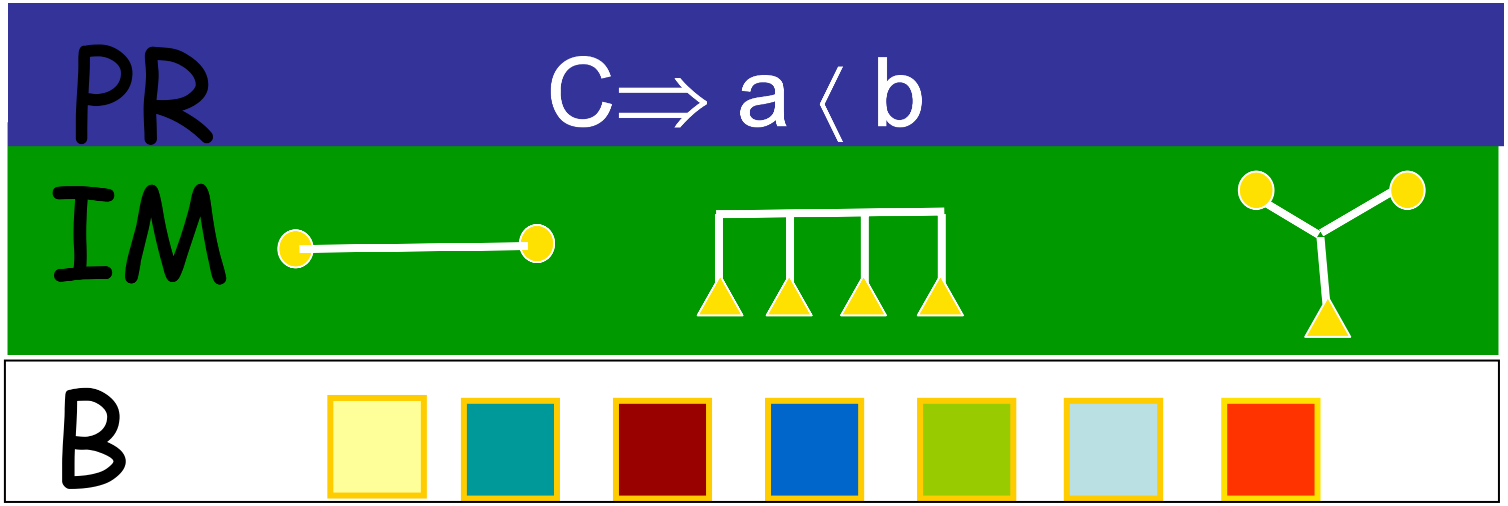


Discussion – separation of concerns



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

Discussion – future work: 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