Component-based Construction of Heterogeneous Real-time Systems in BIP

(“MoCC”s and related issues in BIP)

Joseph Sifakis

VERIMAG

In collaboration with: A. Basu, M. Bozga, G. Goessler

MoCC - Models of Computation and Communication
Zurich, November 16-17, 2006

Key-issues: Component-based construction

Develop a rigorous and general basis for real-time system design and implementation:

- Concept of component and associated composition operators for incremental description and correctness by construction

- Concept for real-time architecture encompassing heterogeneity, paradigms and styles of computation e.g.
    - Synchronous vs. asynchronous execution
    - Event driven vs. data driven computation
    - Distributed vs. centralized execution

- Automated support for component integration and generation of glue code meeting given requirements

Key-issues: Component-based construction
Existing approaches

- Theory such as process algebras and automata
- SW Component frameworks, such as
    - Coordination languages extensions of programming languages : Linda, Javaspaces, TSpaces, Concurrent Fortran, NesC
    - Middleware e.g. Corba, Javabeans, .NET
    - Software development environments: PCTE, SWbus, Softbench, Eclipse
- System modeling languages: SystemC, Statecharts, UML, Simulink/Stateflow, Metropolis, Ptolemy

Lack of

- *frameworks treating interactions and system architecture as first class entities that can be composed and analyzed (usually, interaction by method call)*
- *rigorous models for behavior and in particular aspects related to time and resources.*

Key issues: Heterogeneity [Henzinger&Sifakis, FM06]

Heterogeneity of interaction

- Atomic or non atomic
- Rendezvous or Broadcast
- Binary or n-ary

Heterogeneity of execution

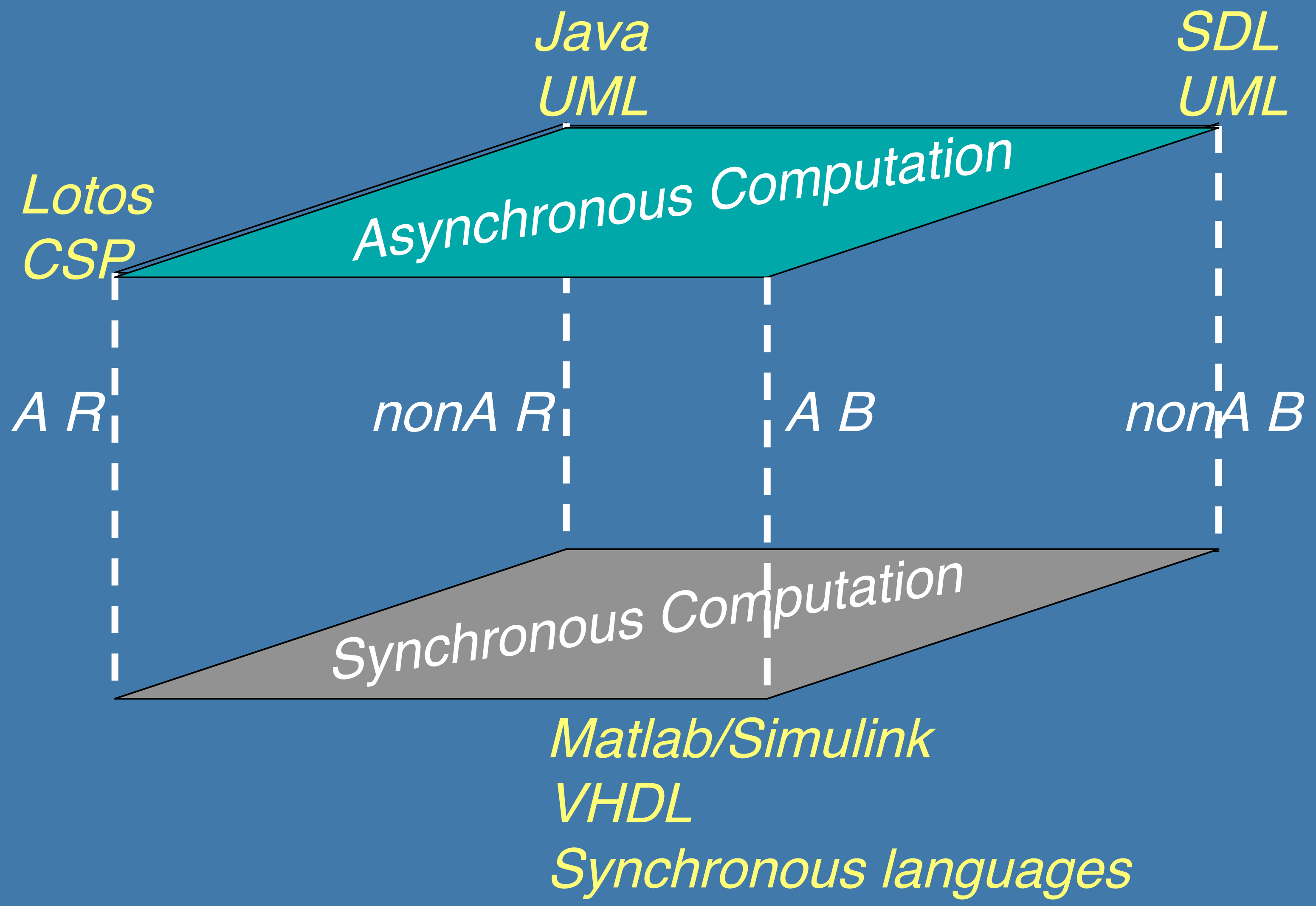
- Synchronous execution
- Asynchronous execution
- Combinations of them

Heterogeneity of abstraction e.g. granularity of execution

*We need a framework **directly** encompassing heterogeneity*

Key issues: Heterogeneity - Example

A: Atomic interaction *R: Rendezvous* *B: Broadcast*

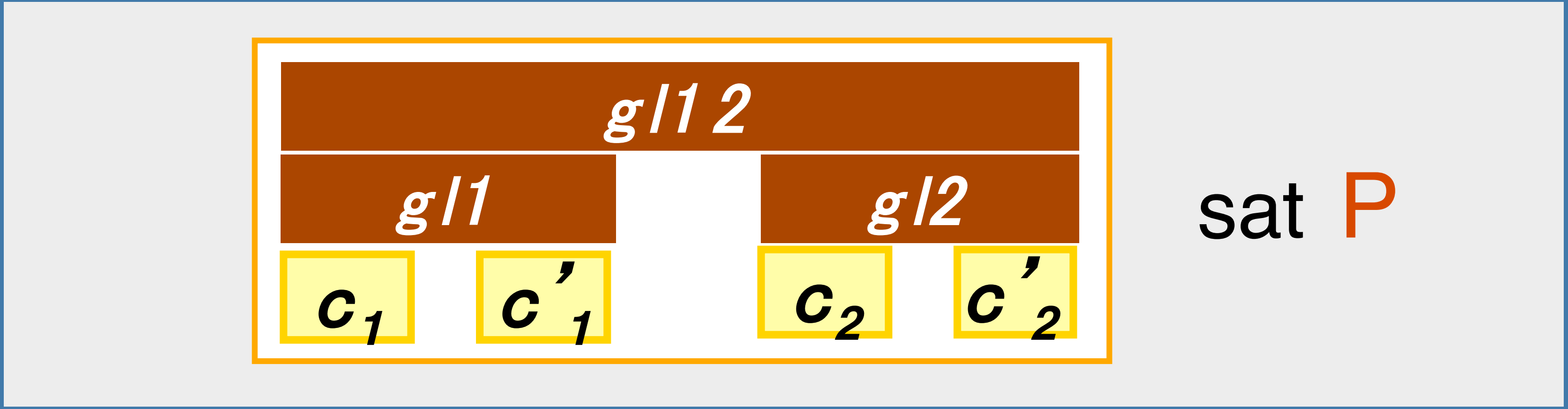


Overview

- About component-based construction
- Interaction modeling
- Priority modeling
- Implementation
- Modeling systems in BIP
- Discussion

Component-based construction – Formal framework

Build a component *C* satisfying a given property *P,* from
- $\mathcal{C}_0$ a set of *atomic* components modeling behavior
- $\mathcal{GL} = \{gl_1, \ldots, gl_i, \ldots\}$ a set of glue operators on components



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

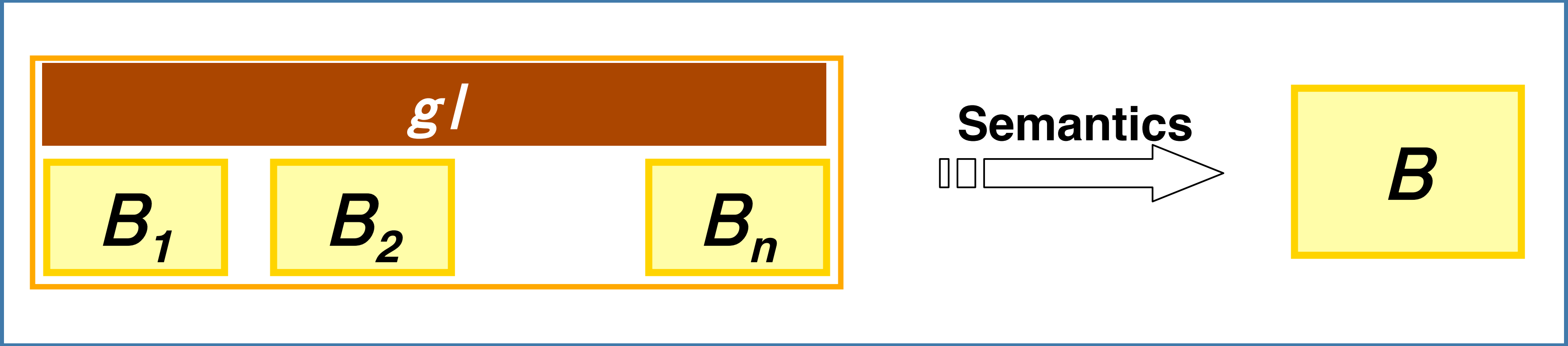
$$gl(C_1, C_2, .., C_n)|\ A_1 \textit{ refines } C_1$$

Component-based construction – Formal framework

Semantics:

• Atomic components → behavior

• Glue operators transform sets of components into components



The process algebra paradigm

• 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 semantics

Component-based construction - Requirements

Find sets of glue operators meeting the following requirements:

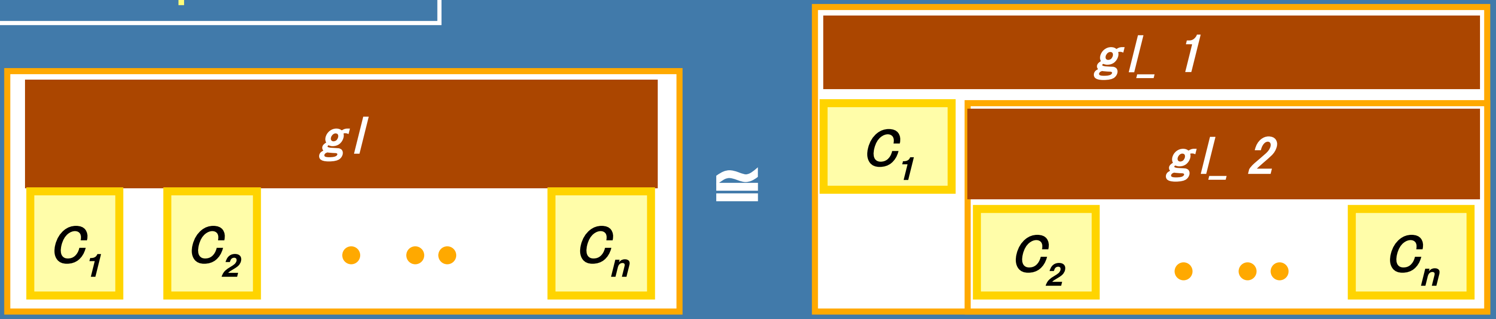
1. Incremental description

2. Correctness-by-construction

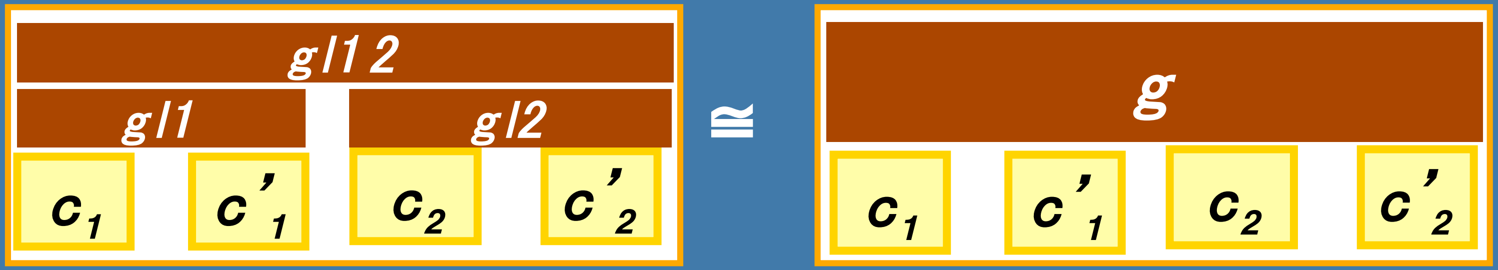
3. Expressiveness (discussed later)

Component-based construction – Incremental description

1. Decomposition



2. Flattening



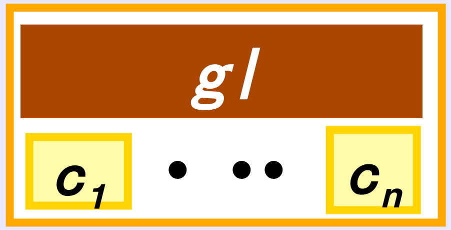
Flattening can be achieved by using a (partial) associative operation $\oplus$ on GL

Component-based construction - Correctness by construction : Compositionality

Building correct systems

from correct components

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

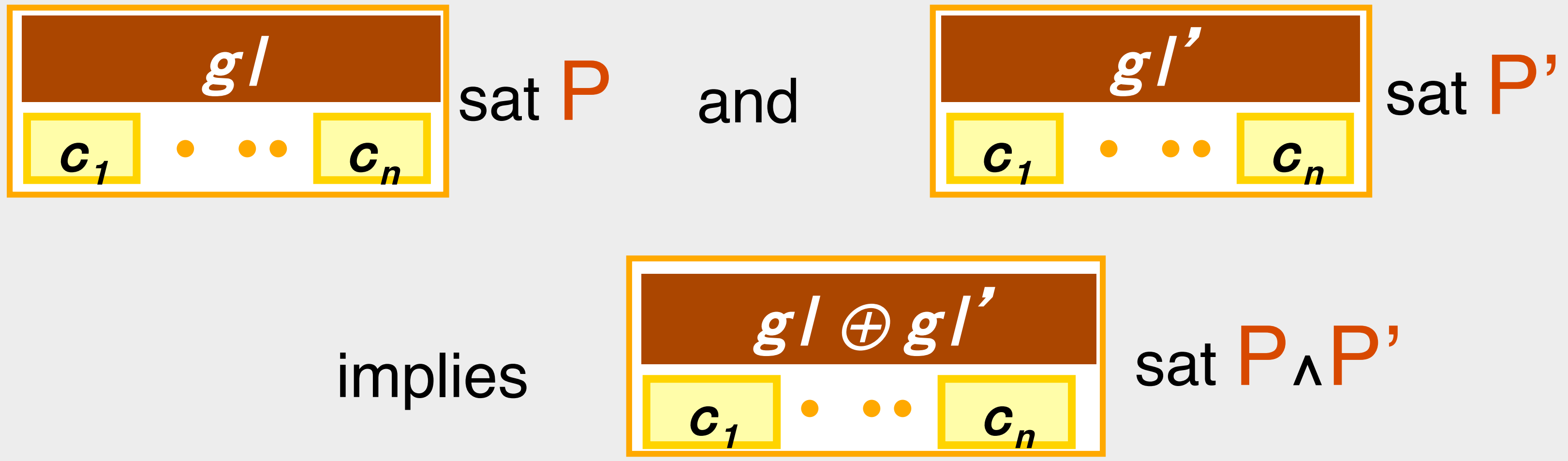


sat $\widetilde{gl}(P_1, .., P_n)$

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

Component-based construction - Correctness by construction : Composability

Integrated components preserve essential properties

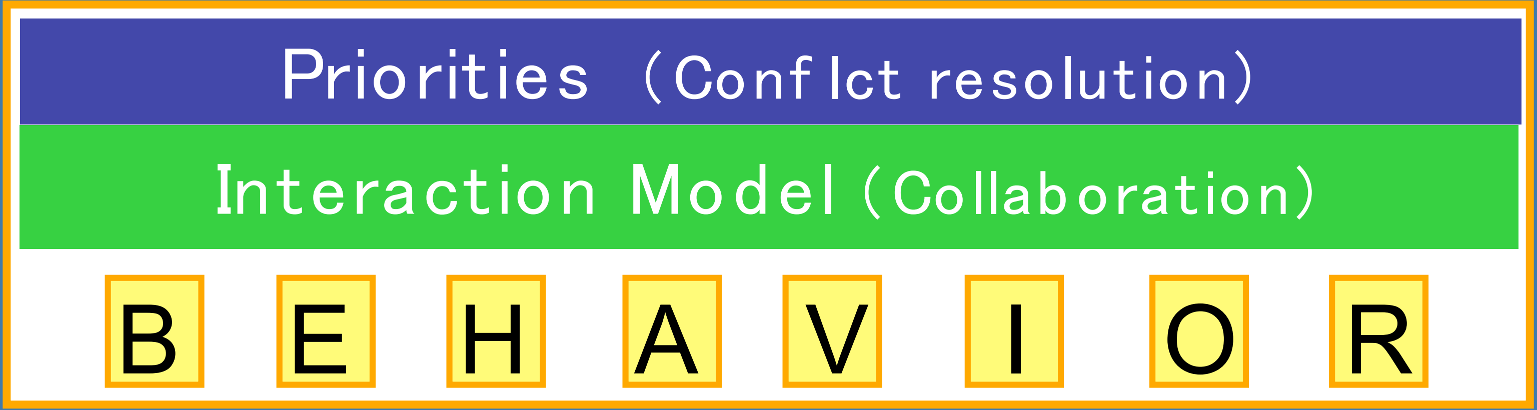


Composability means non interference of properties of integrated components. Lack of results for guaranteeing property stability e.g.

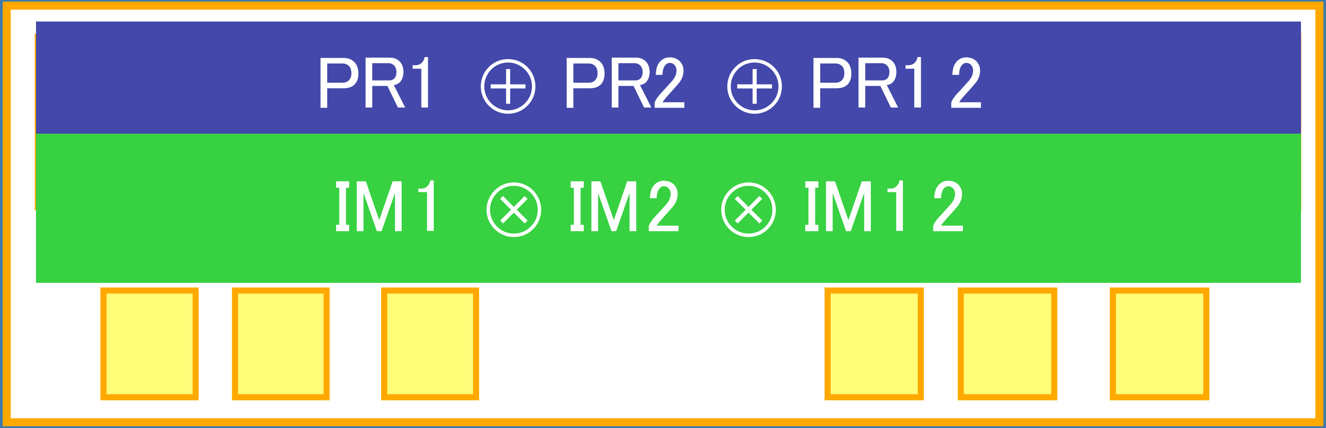
- *non composability of scheduling algorithms*
- *feature interaction*

Component-based construction – The BIP framework

Layered component model


Priorities (Conf lct resolution)
Interaction Model (Collaboration)
B E H A V I O R

Composition (incremental description)



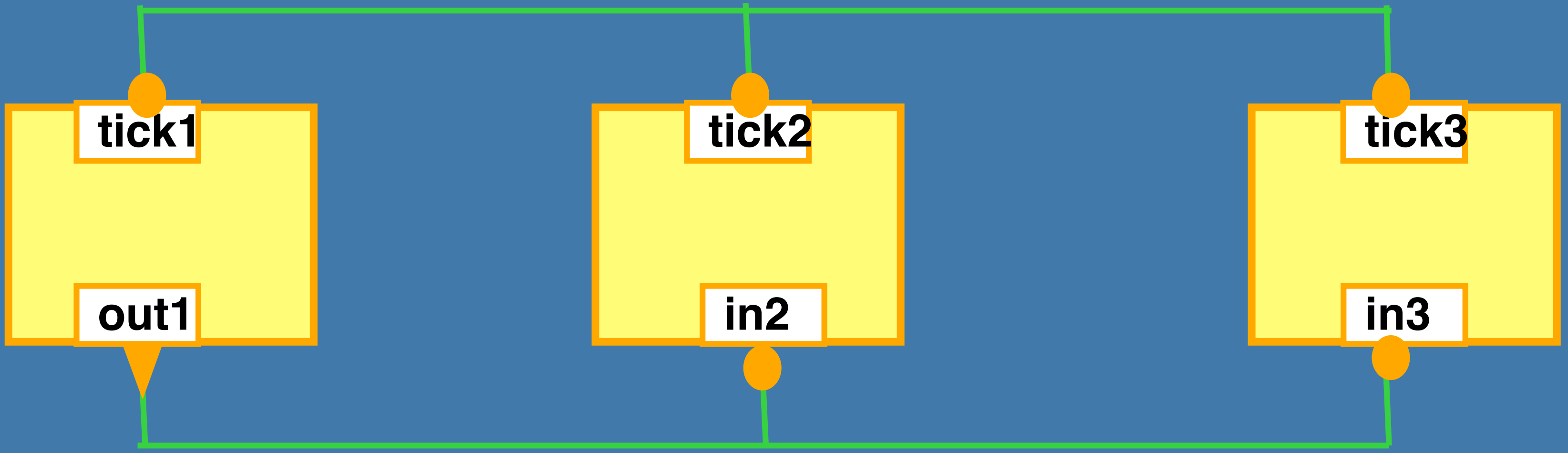
Overview

- About component-based construction
- Interaction modeling
- Priority modeling
- Implementation
- Modeling systems in BIP
- Discussion

Interaction modeling

- A ***connector*** is a set of ports which can be involved in an interaction

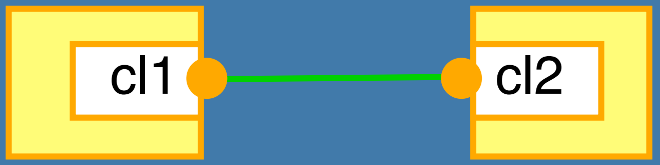
- Port attributes (***complete*** ▼, ***incomplete*** ●) are used to distinguish between rendezvous and broadcast.
- An ***interaction*** of a connector is a set of ports such that: either it contains some complete port or it is maximal.



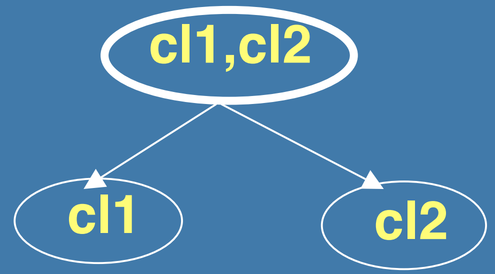
Interactions:

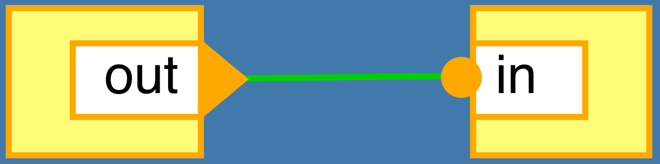
{tick1,tick2,tick3} {out1} {out1,in2} {out1,in3} {out1,in2, in3}

Interaction modeling - Examples

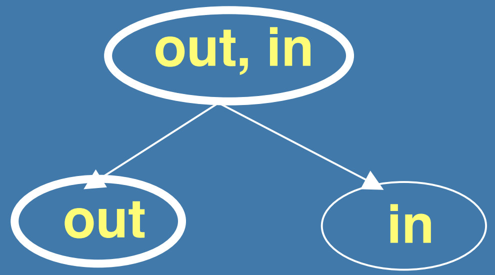


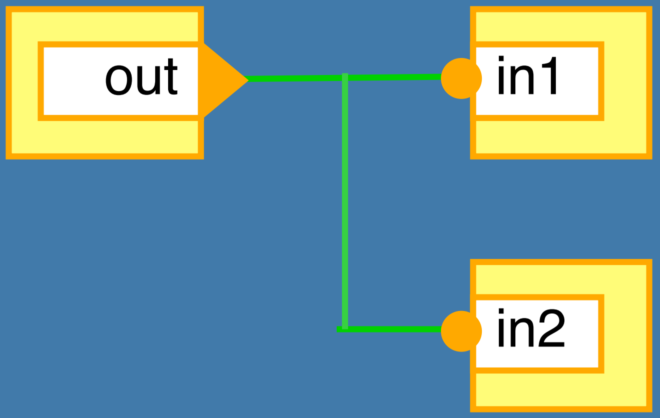
CN:{cl1,cl2}
CP: ∅



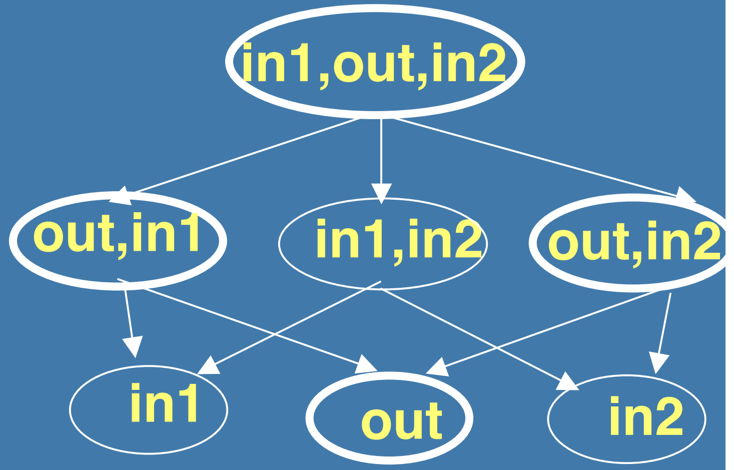


CN:{out,in}
CP: {out}





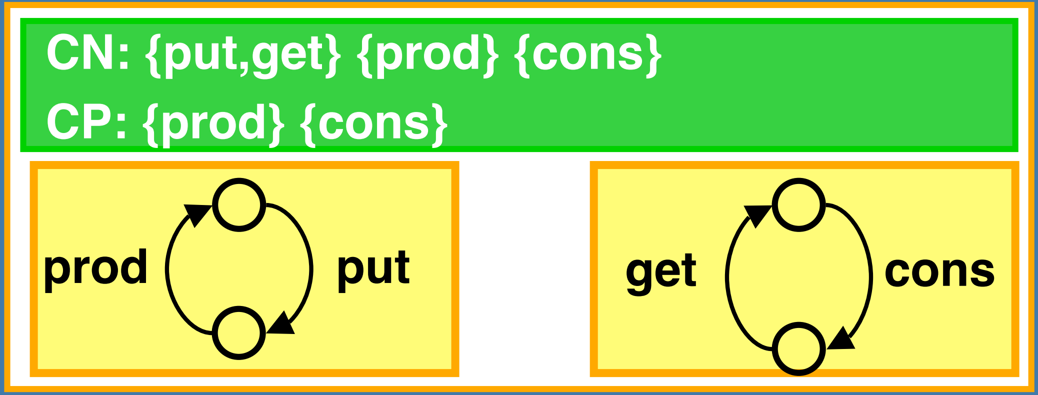
CN:{in1,out,in2}
CP: {out}

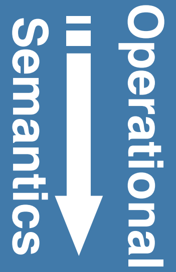


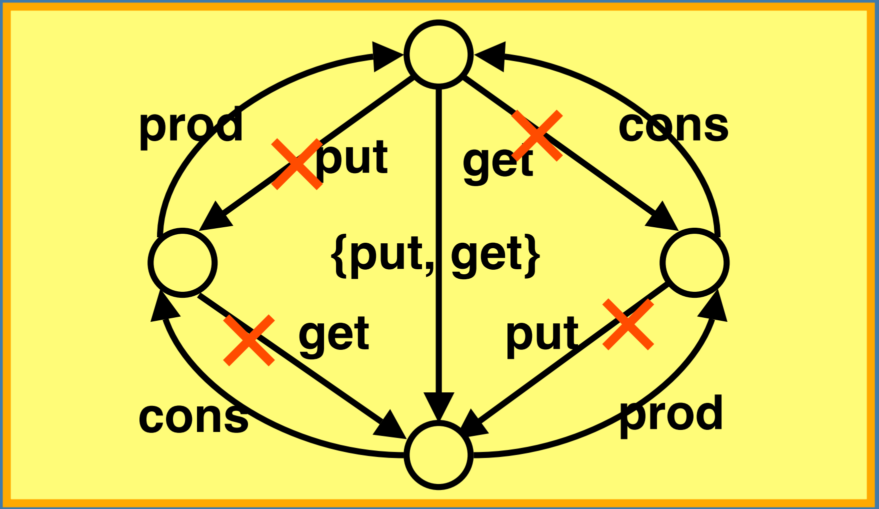
Interaction modeling – Operational semantics

CN: {put,get} {prod} {cons}

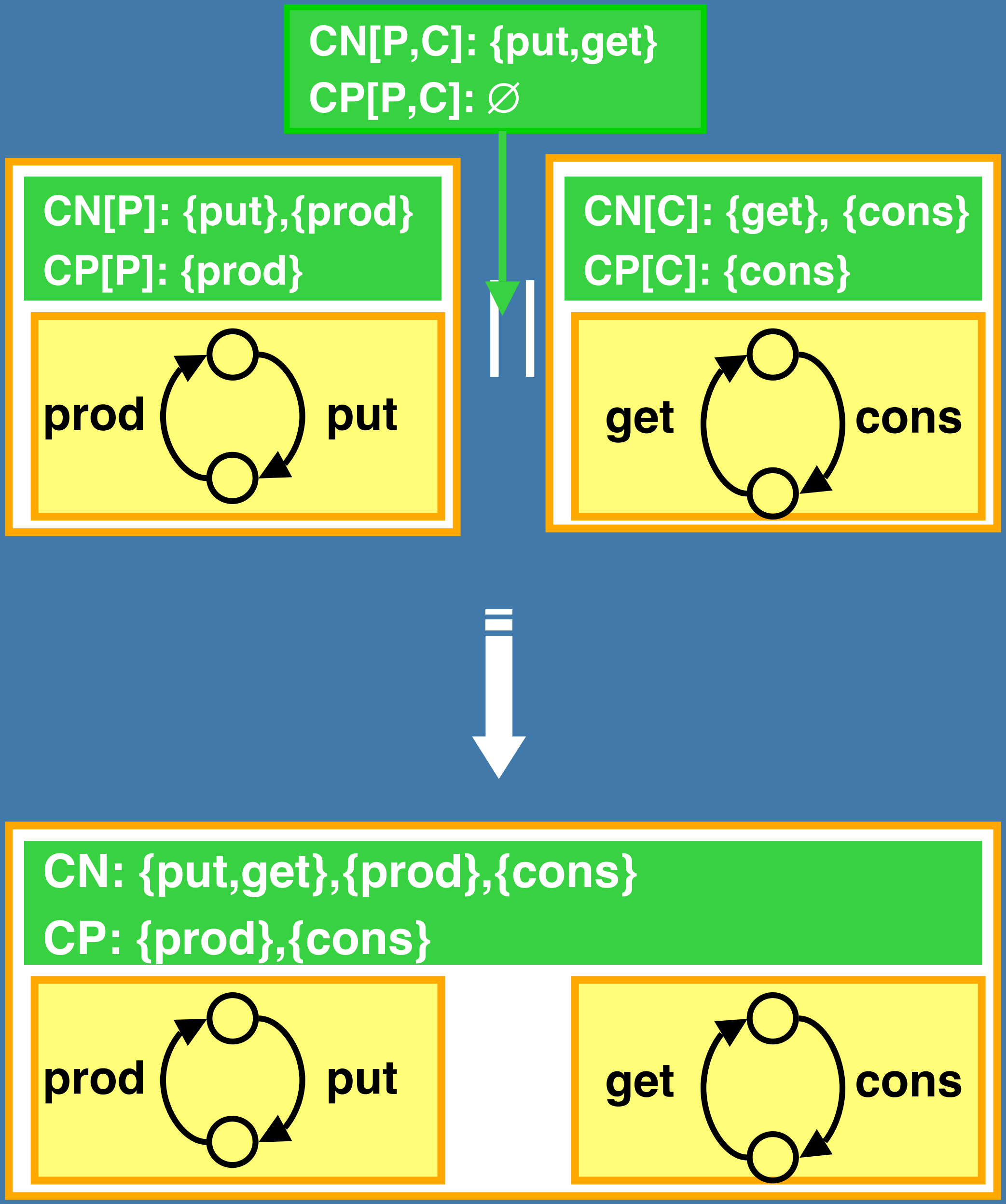
CP: {prod} {cons}







Interaction modeling – Incremental Composition



Overview

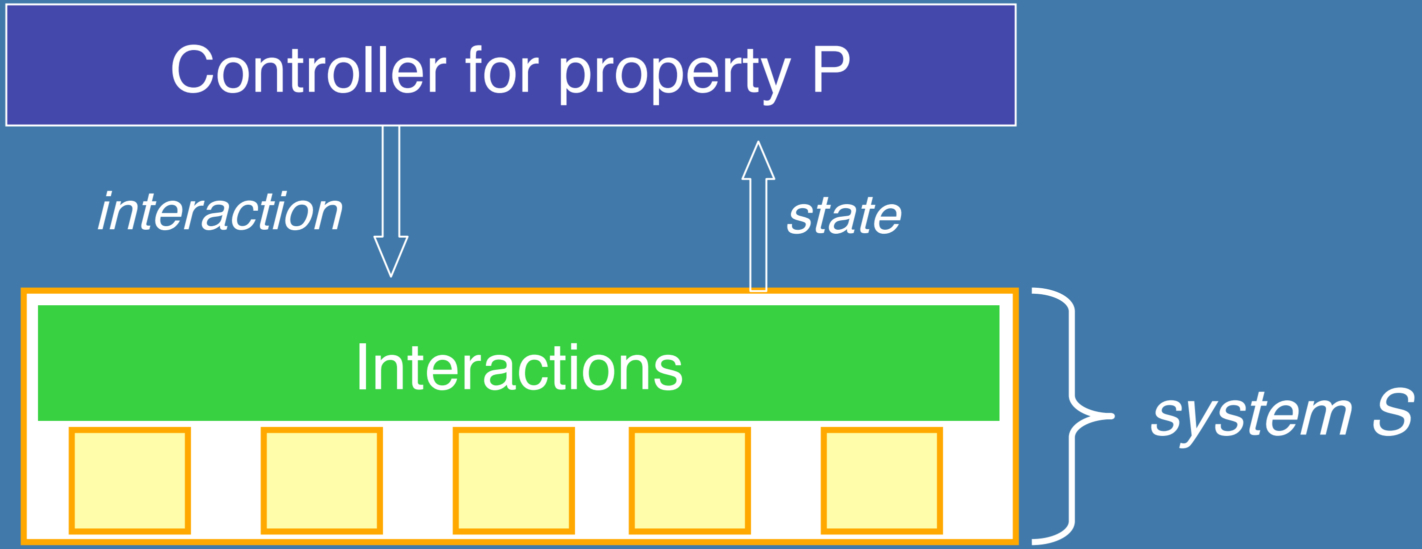
Priorities

Priorities are a powerful tool for restricting non-determinism:

- they allow straightforward modeling of urgency and scheduling policies for real-time systems
- run to completion and synchronous execution can be modeled by assigning priorities to threads
- they can advantageously replace (static) restriction of process algebras

Priorities – Priorities as controllers

A controller restricts the behavior (non determinism) of system S to enforce a property P



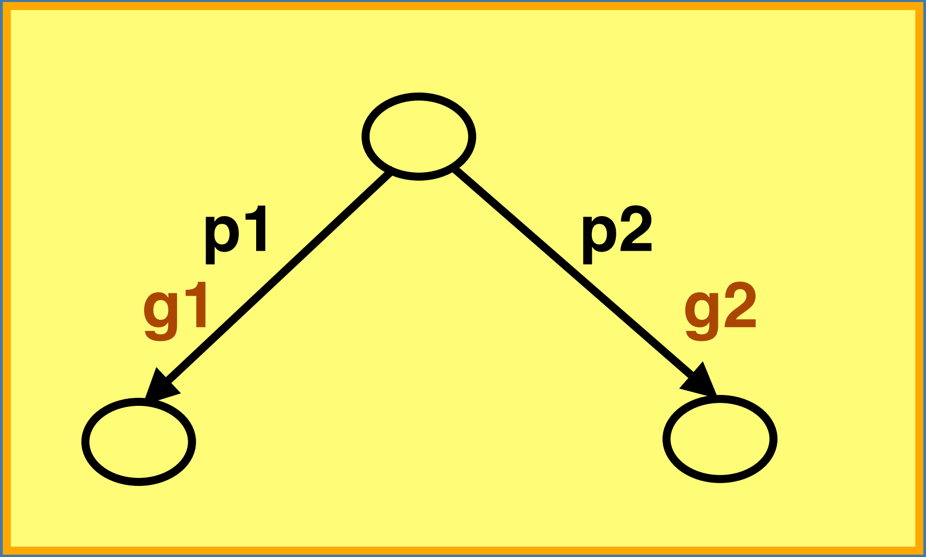
Results [Goessler&Sifakis, FMCO2003] :

• Restrictions induced by controllers enforcing deadlock-free state invariants can be described by dynamic priorities

• Conversely, for any restriction induced by dynamic priorities there exists a controller enforcing a deadlock-free state invariant

Priorities - Definition

Priority rules

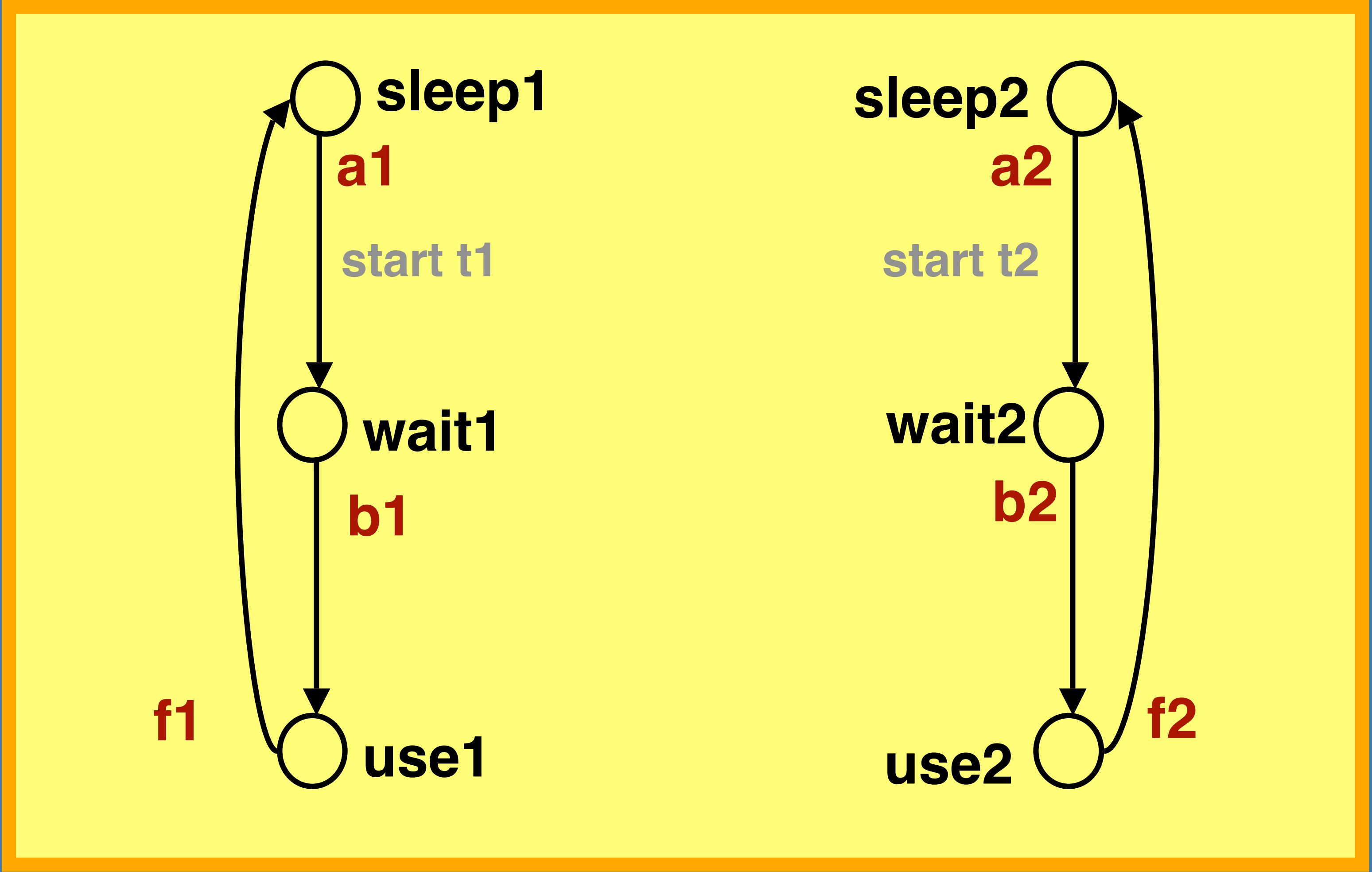


Priority rule	Restricted guard g1’
true → p1 ⟨ p2	g1’ = g1 Ù Ø g2
C → p1 ⟨ p2	g1’ = g1 Ù Ø(C Ù g2)

Priorities – Example: Mutual exclusion + FIFO policy

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

$true \rightarrow b1 \langle f2$ $true \rightarrow b2 \langle f1$



Overview

- About component-based construction
- Interaction modeling
- Priority modeling
- Implementation
- Modeling systems in BIP
- Discussion

Implementation – the BIP language: atomic component

```
component C
port  complete: p1, … ;  incomplete: p2, …
data  {# int x, float y, bool z, …. #}
init {# z=false; #}
    behavior
             state s1
                          on p1 provided g1 do f1 to  s1'
                          ……………….      ……
                          on pn provided gn do fn to  sn'

             state s2
                          on …..
               ….

             state sn
                          on ….
    end
end
```

Implementation – the BIP language: connectors and priorities

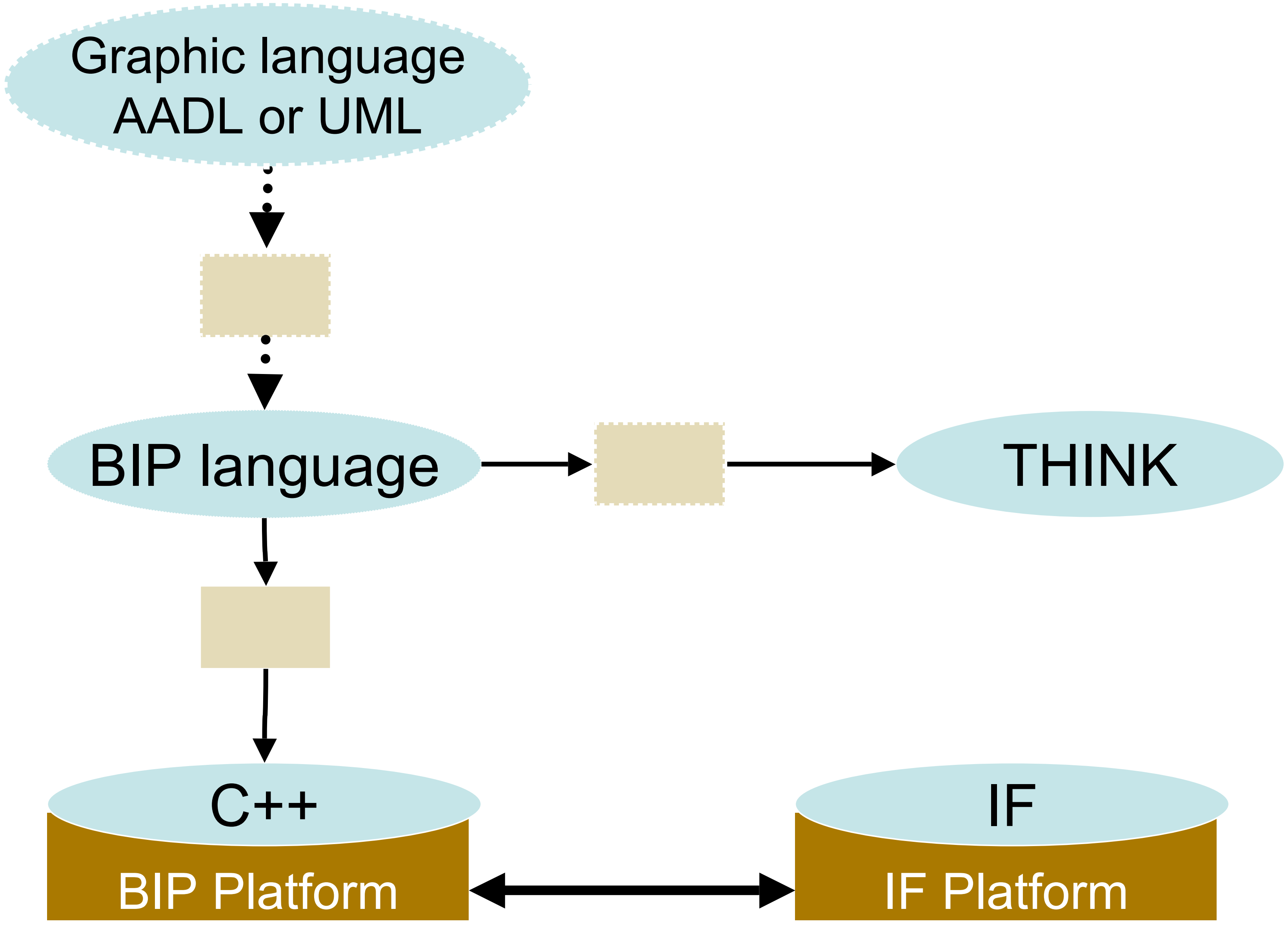
```
connector BUS= {p, p', … , }
complete()
   behavior
        on a1 provided $g_{a1}$ do $f_{a1}$
        ……….
        on an provided $g_{an}$ do $f_{an}$
   end
```

```
priority PR
        if  C1 (a1 < a2), (a3 < a4) , …
        if  C2 (a < …), (a <…) , …
        …
        if  Cn (a <…), (a <…) , …
```

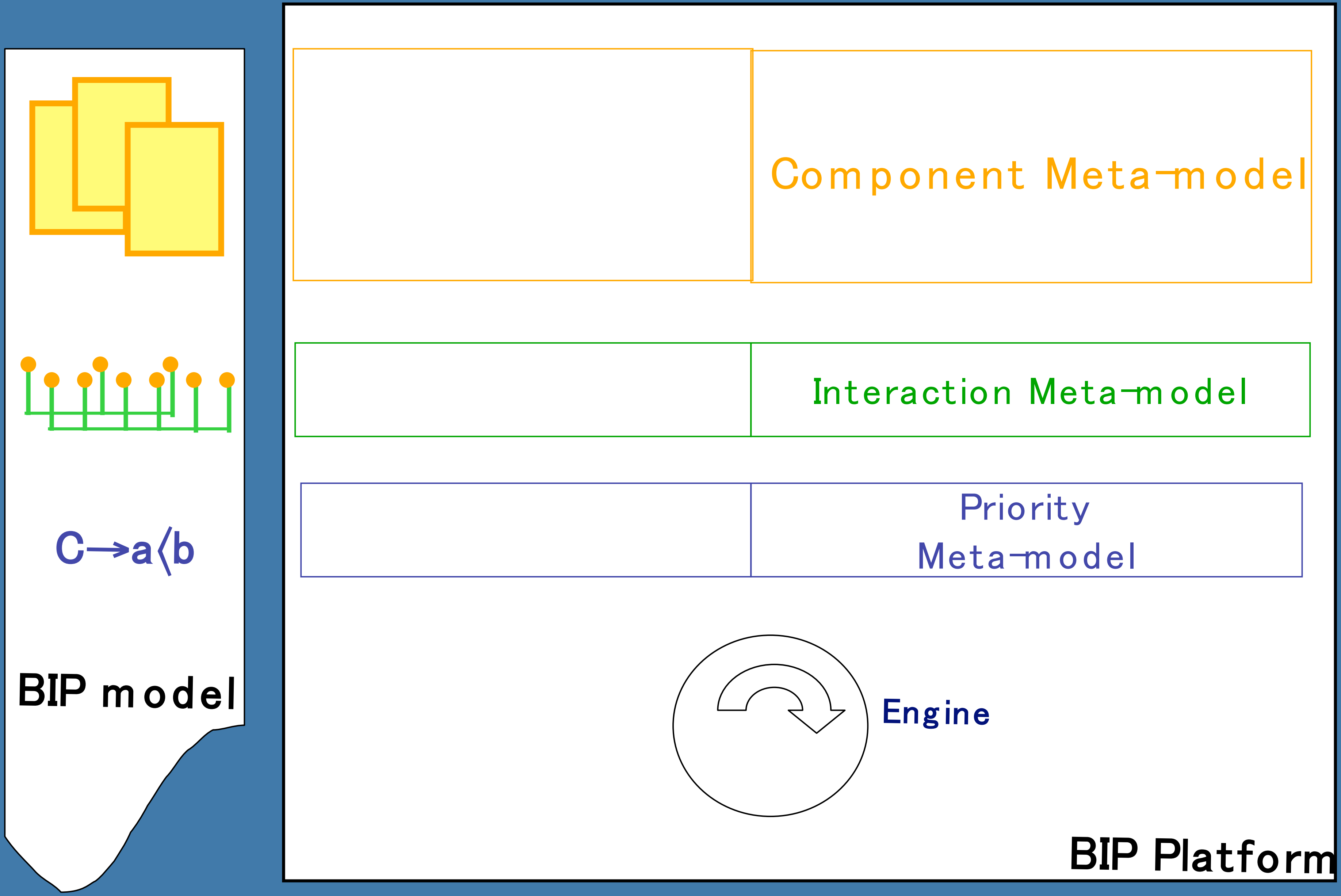
Implementation – the BIP language: compound component

```
component name
    contains c_name1 i_name1(par_list)
        ……
    contains c_namen i_namen(par_list)

    connector name1
    ……
    connector namem

    priority name1
    ……
    priority namek
    end
```

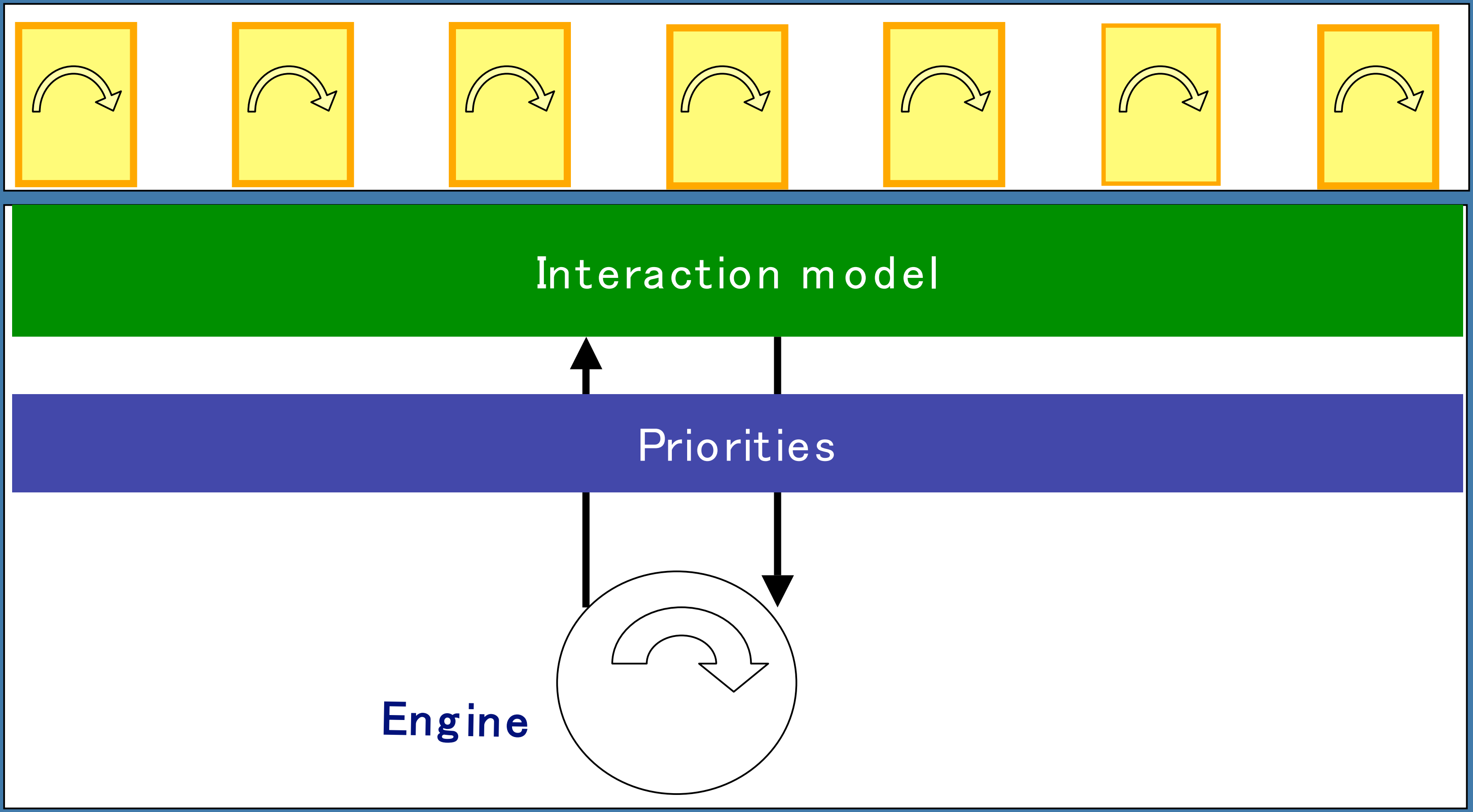
Implementation – the BIP toolset



Implementation – C++ code generation for the BIP platform

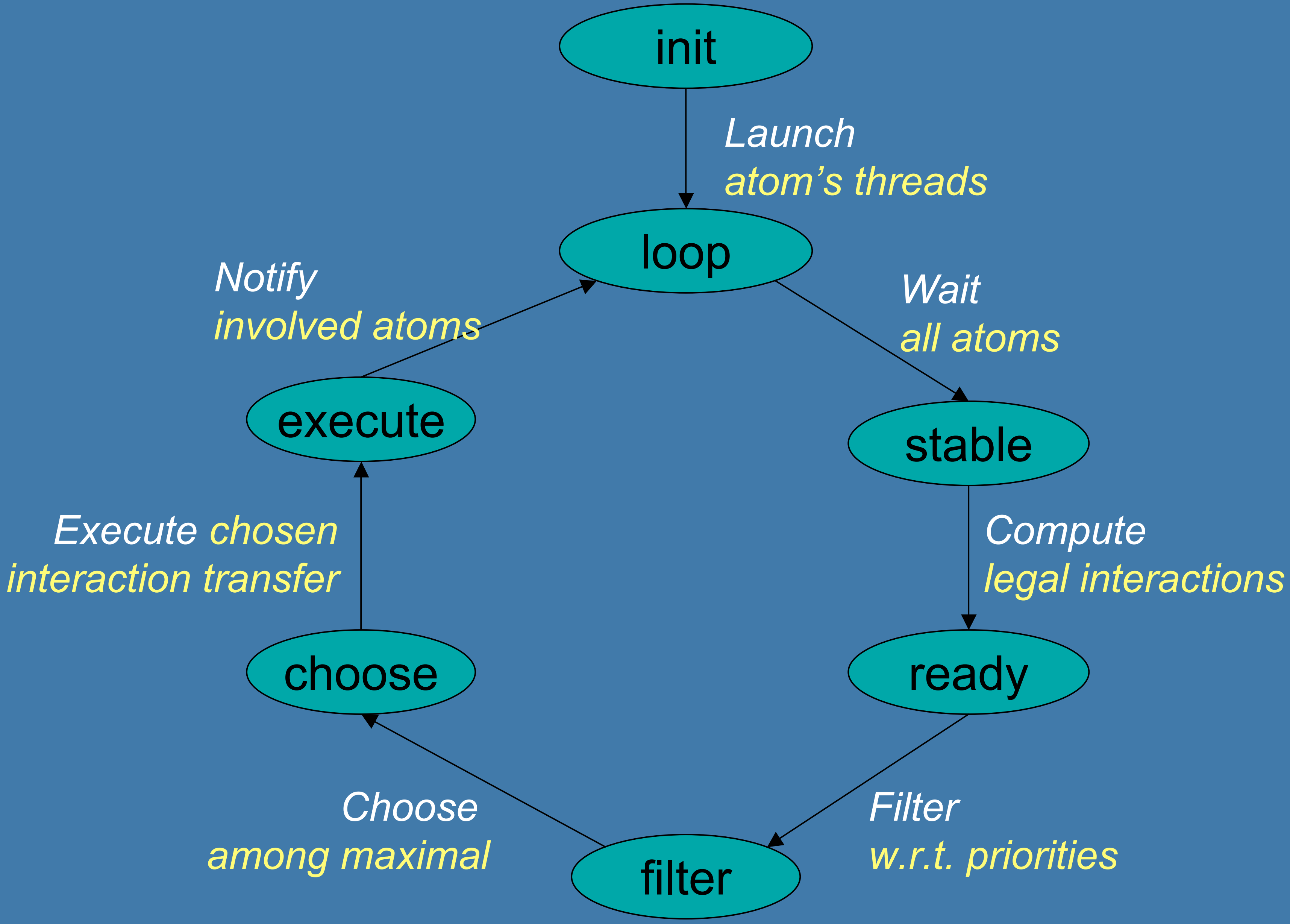


Implementation – The BIP platform



- Code execution and state space exploration features
- Implementation in C++ on Linux using POSIX threads
- Thread assignments preserve semantics

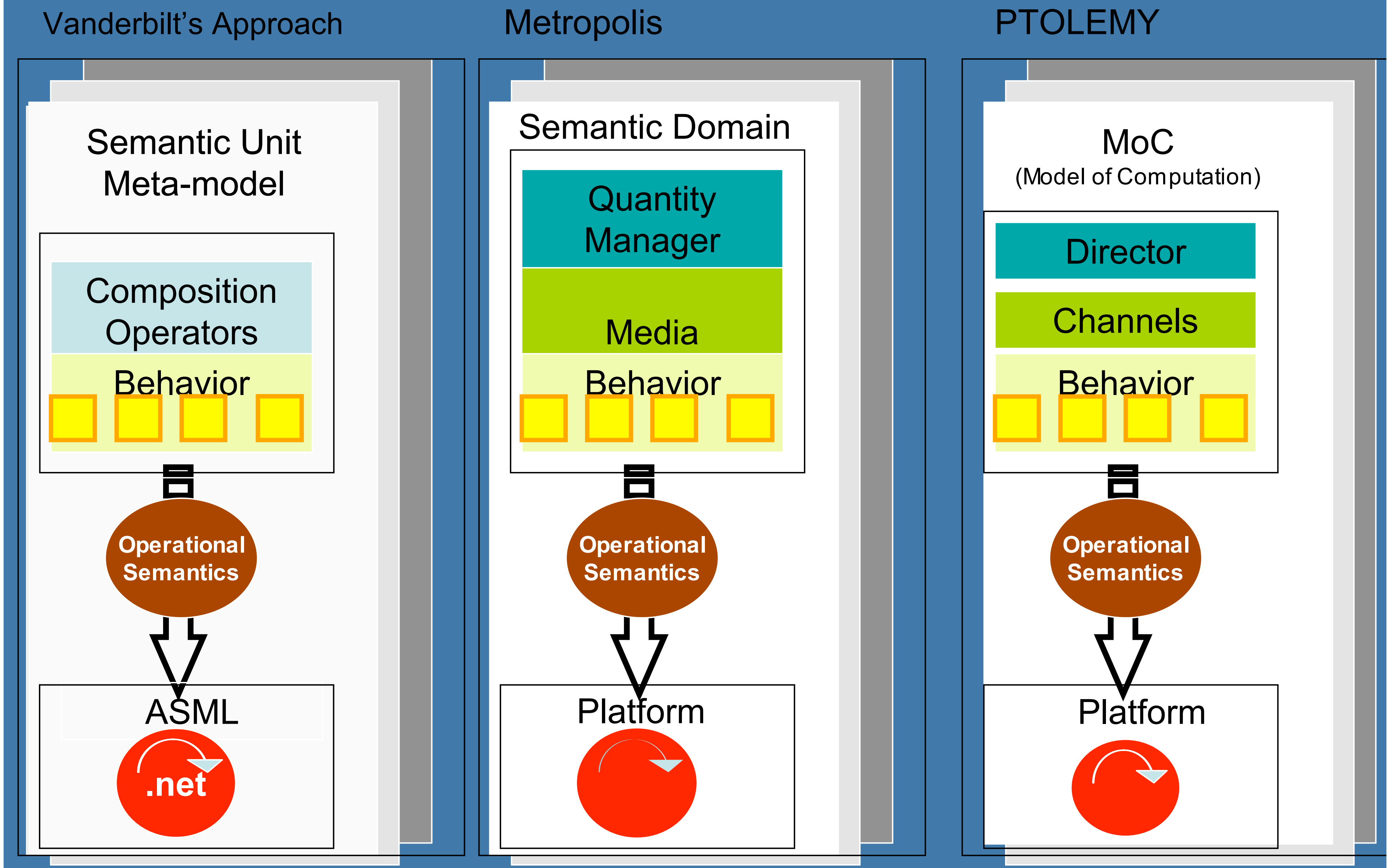
Implementation – The BIP platform: The engine



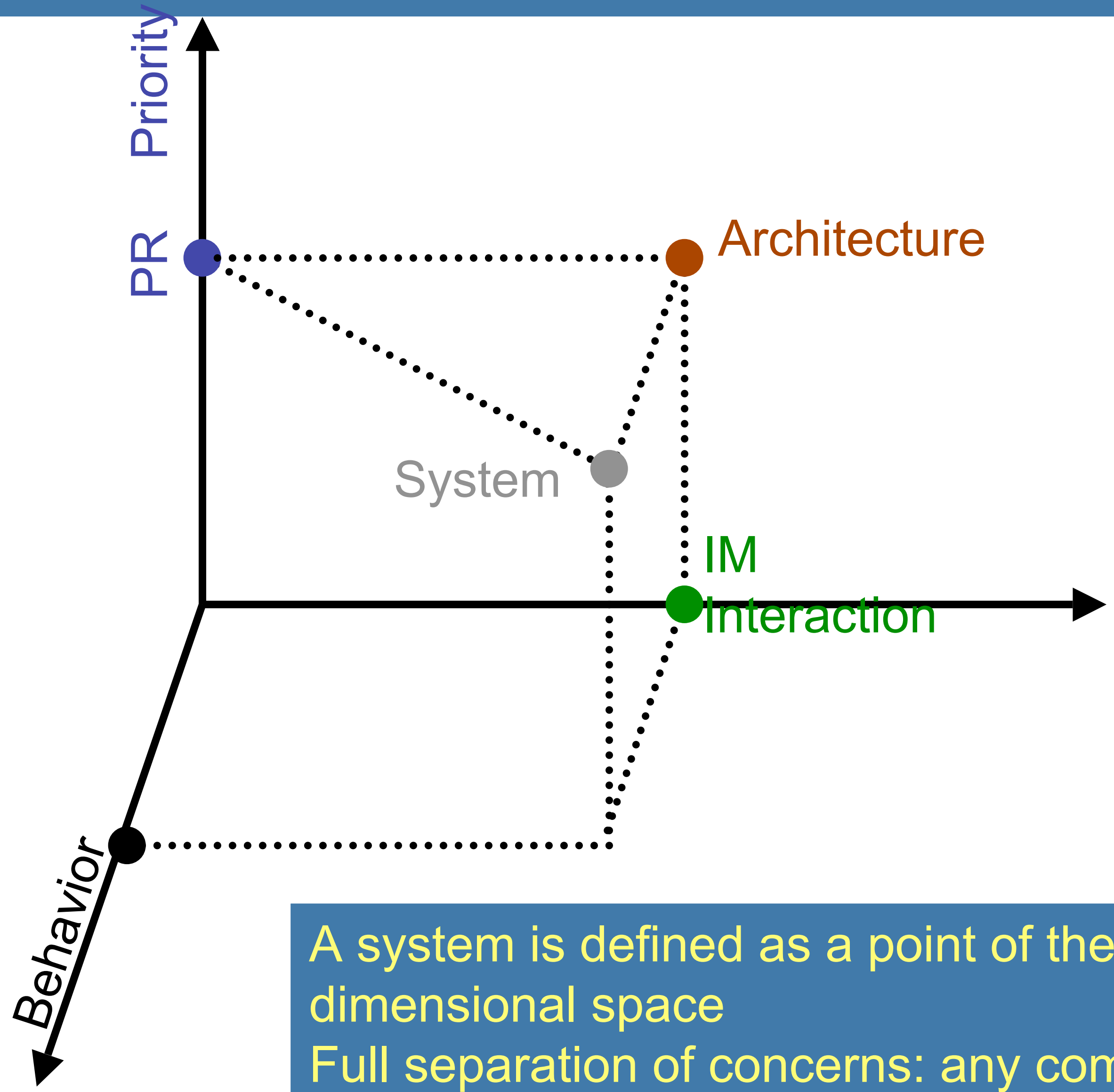
Overview

- About component-based construction
- Interaction modeling
- Priority modeling
- Implementation
- Modeling systems in BIP
- Discussion

Modeling in BIP– Other approaches encompassing heterogeneity

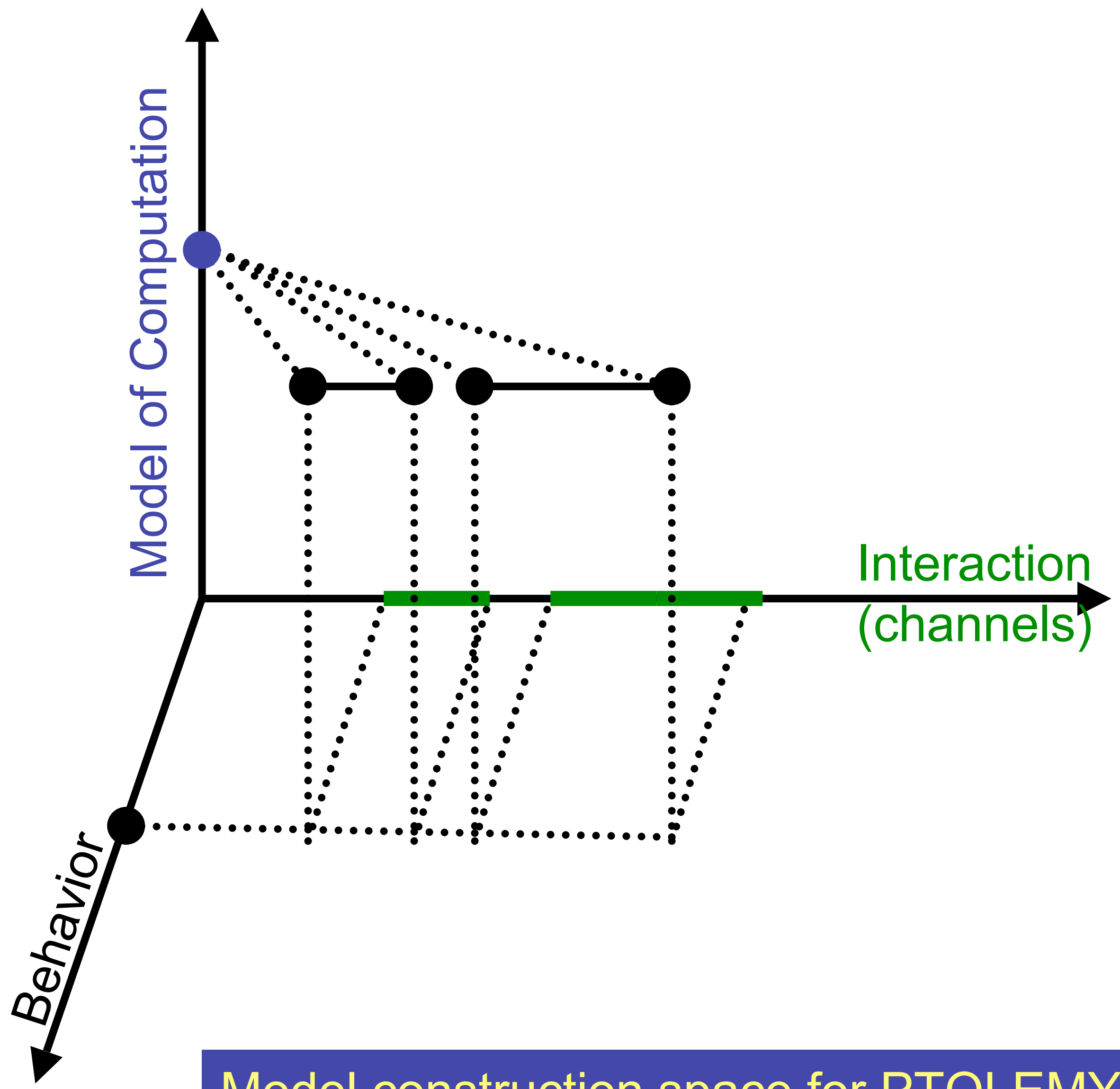


Modeling in BIP– System construction space



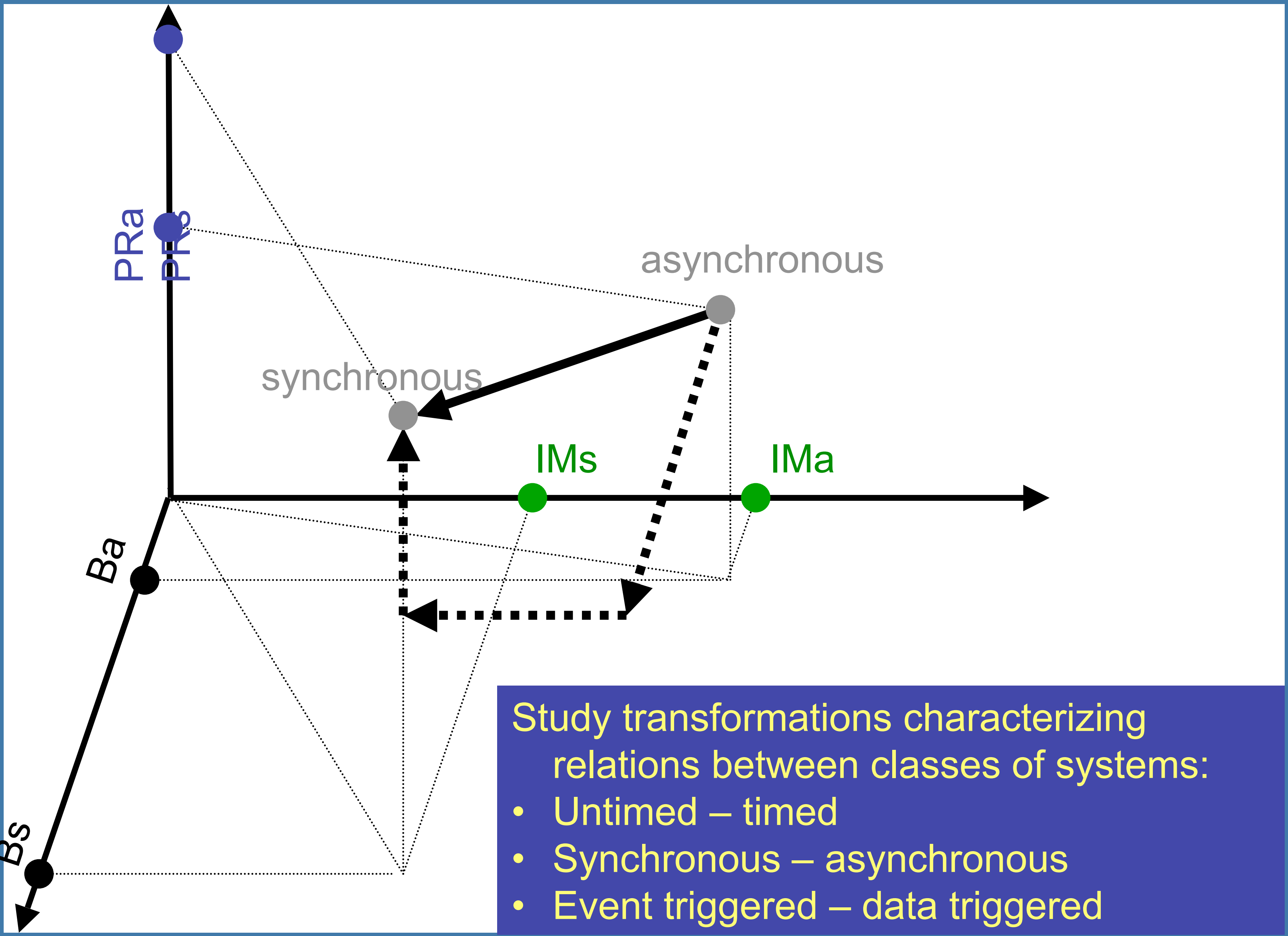
A system is defined as a point of the 3-dimensional space
Full separation of concerns: any combination of coordinates defines a system

Modeling in BIP – System construction space (2)



Model construction space for PTOLEMY

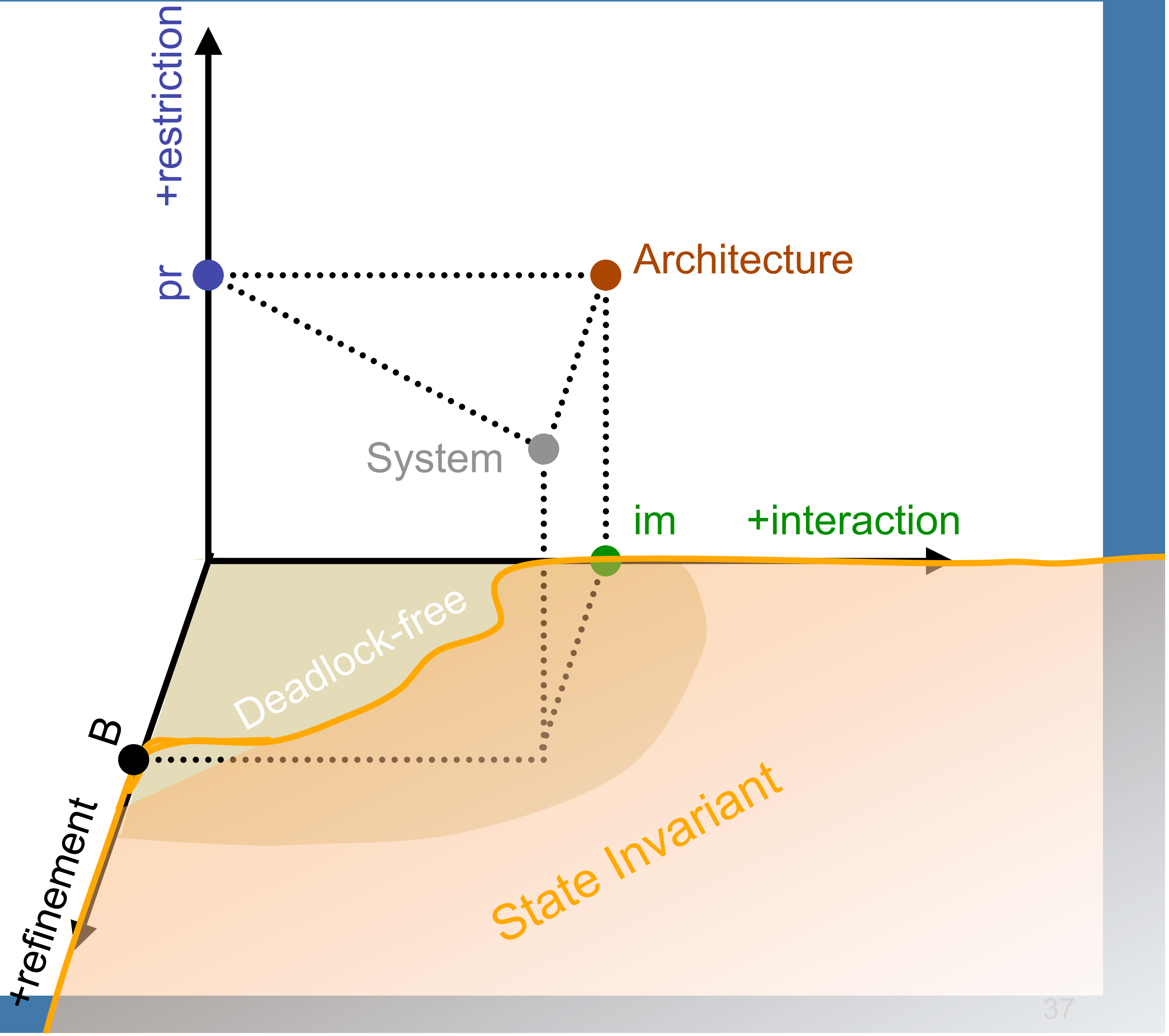
The BIP framework – Relating classes of components



Study transformations characterizing relations between classes of systems:

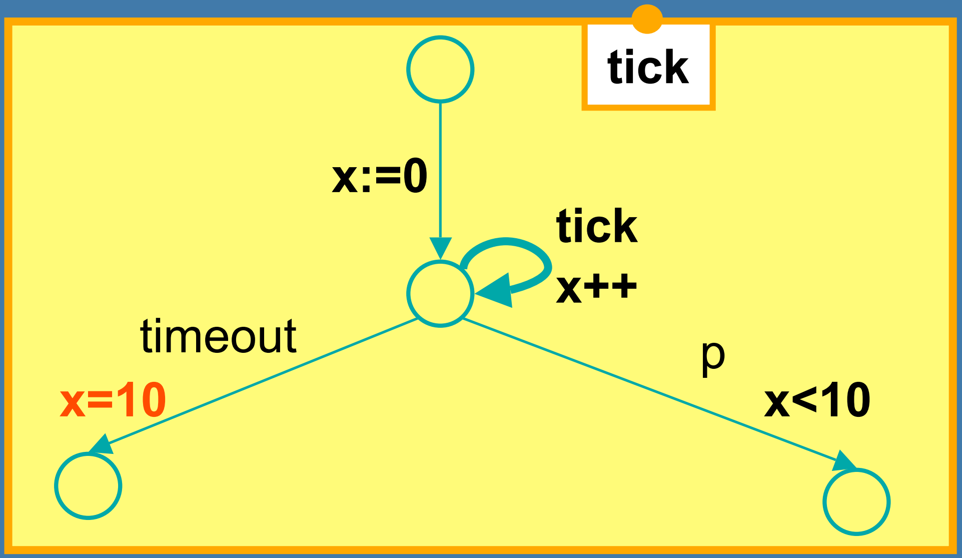
- Untimed – timed
- Synchronous – asynchronous
- Event triggered – data triggered

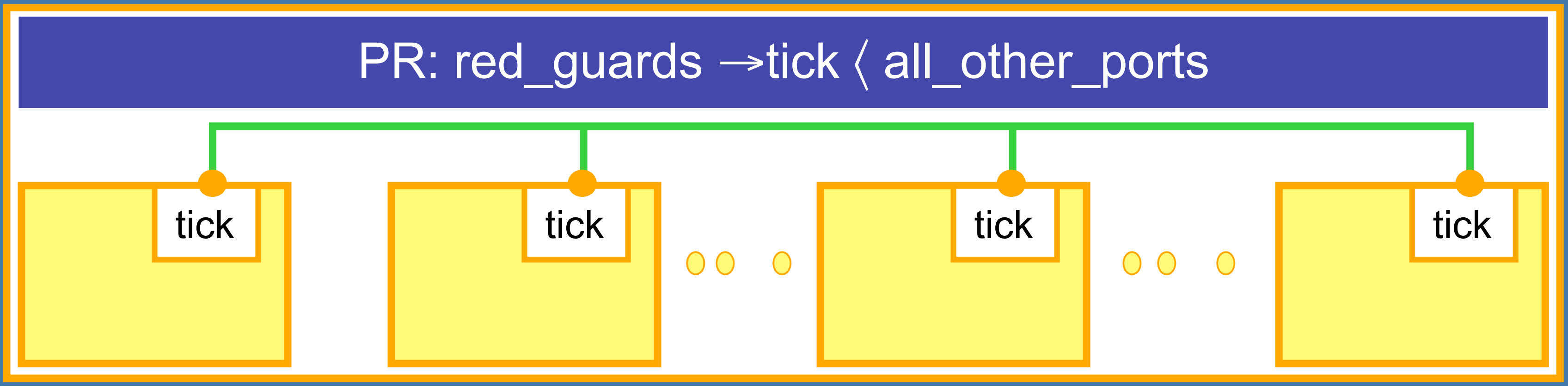
Modeling in BIP – Property preservation



Modeling in BIP – Timed systems

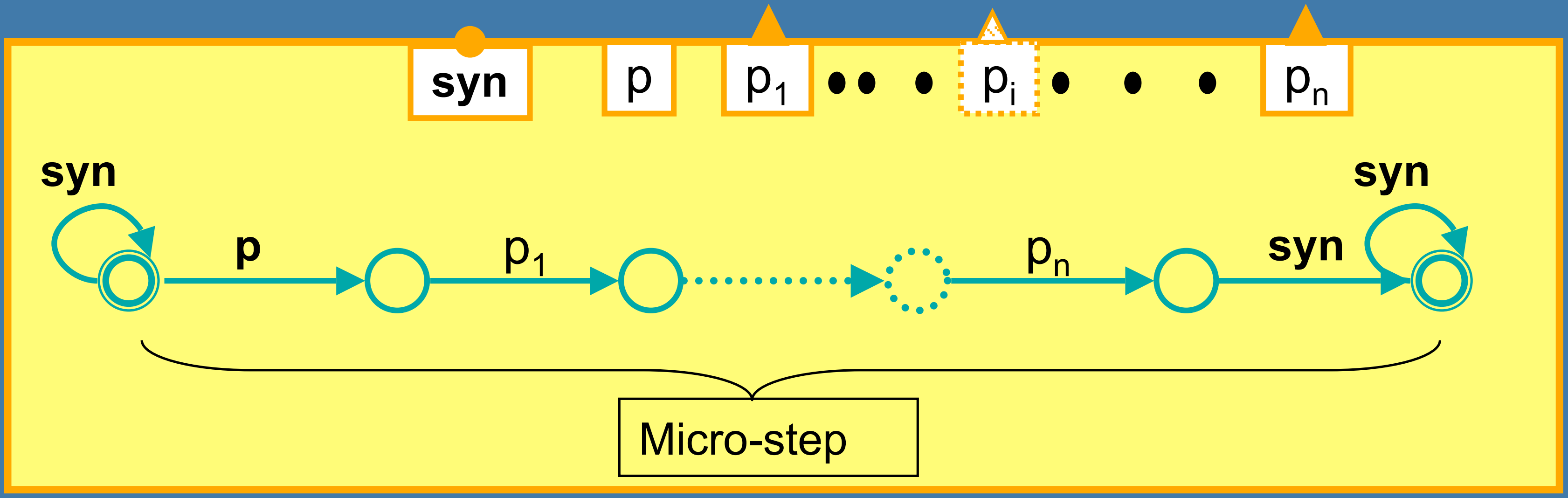
Timed Component



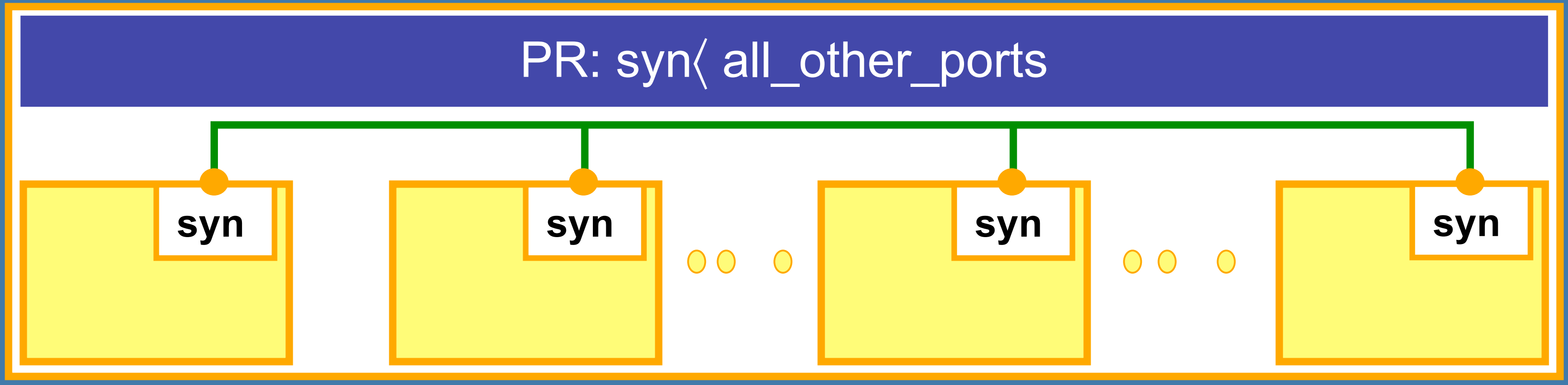


Timed architecture

Modeling in BIP – Synchronous systems



Synchronous component



Synchronous architecture

Overview

- About component-based construction
- Interaction modeling
- Priority modeling
- Implementation
- Modeling systems in BIP
- Discussion

Discussion – Semantic frameworks

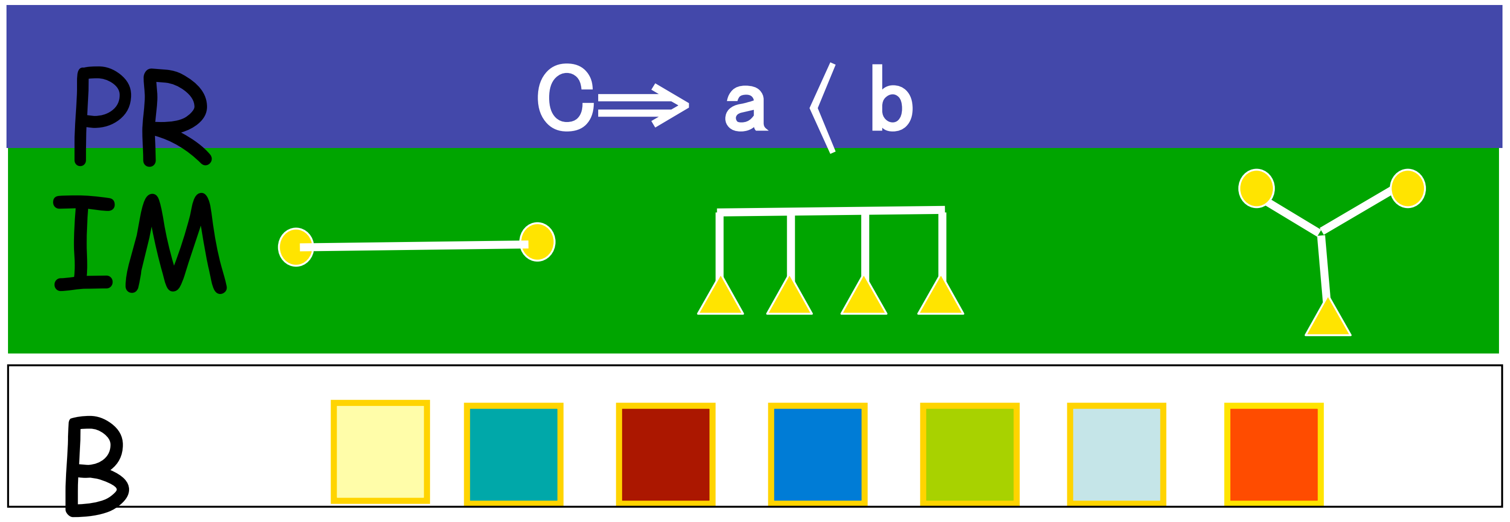
Denotational semantics:
elegant and powerful but we absolutely need associated executable semantic models to be able to faithfully apply theory in methods and tools

Operational semantics:
inherent difficulties to deal with concurrency and resource modeling

For both:
We need « high level » semantic frameworks where structure is a first class entity.

Discussion – Structural Expressiveness

Find a notion of expressiveness different from existing ones which completely ignore structure e.g. all finite state formalisms are equally expressive



For given B, IM and PR which coordination problems can be solved (without modifying behavior of atomic components)?

Discussion – Structural Expressiveness [Sifakis SEFM05]

- Study Component Algebras CA= (**B**, GL,⊕, ≅), where
  - (GL,⊕) is a commutative monoid
  - ≅ is a congruence compatible with operational semantics

- Given two component algebras defined on the same set of atomic components,

CA1 **is more expressive** than CA2 if ∀P ∀B1, .,Bn

∃gl2∈GL2. gl2(B1, .,Bn) sat P ⇒ ∃ gl1∈GL1. gl1(B1, …Bn) sat P

Discussion – Summary for BIP

Framework for component-based construction encompassing heterogeneity and relying on a **minimal set of constructs and principles**

Clear separation between structure (interaction +priority) and behavior

- Structure is a first class entity
- Layered description => separation of concerns => incrementality
- Correctness-by-construction techniques for deadlock-freedom and liveness, based (mainly) on sufficient conditions on the structure

Discussion - Work directions for BIP

Theory

- An algebraic framework based on structural expressiveness
- Correctness by construction
- Model transformation techniques – relating classes of systems

Methodology

- Using BIP as a programming model
- Modeling architectures in BIP

BIP toolset Implementation

- Generation of BIP models from system description languages such as SysML (IST/SPEEDS project), AADL and SystemC (ITEA/Spices project)
- Code generation and optimization for various platforms
- Validation techniques

More about BIP:

- http://www-verimag.imag.fr/index.php?page=tools
- Email to Joseph.Sifakis@imag.fr

THANK YOU