The Quest for Correctness - Beyond *a posteriori* Verification

SPIN 09
Grenoble, June 27th 2009

Joseph Sifakis
VERIMAG Laboratory
in collaboration with
A. Basu, S. Bensalem, S. Bliudze, B. Bonakdarpour, M. Bozga, M. Jaber, M. Gallien. H. Nguyen, V. Sfyrla, R. Yan

Correctness by checking vs. Correctness by construction

Building systems which are correct with respect to given requirements is the main challenge for all engineering disciplines

Correctness can be achieved:

- Either by checking that a system or a model of a system meets given requirements
- Or by construction by using results such as algorithms, protocols, architectures e.g. token ring protocol, time triggered architecture

A big difference between Computing Systems Engineering and disciplines based on Physics is the importance of *a posteriori* verification for achieving correctness

OVERVIEW

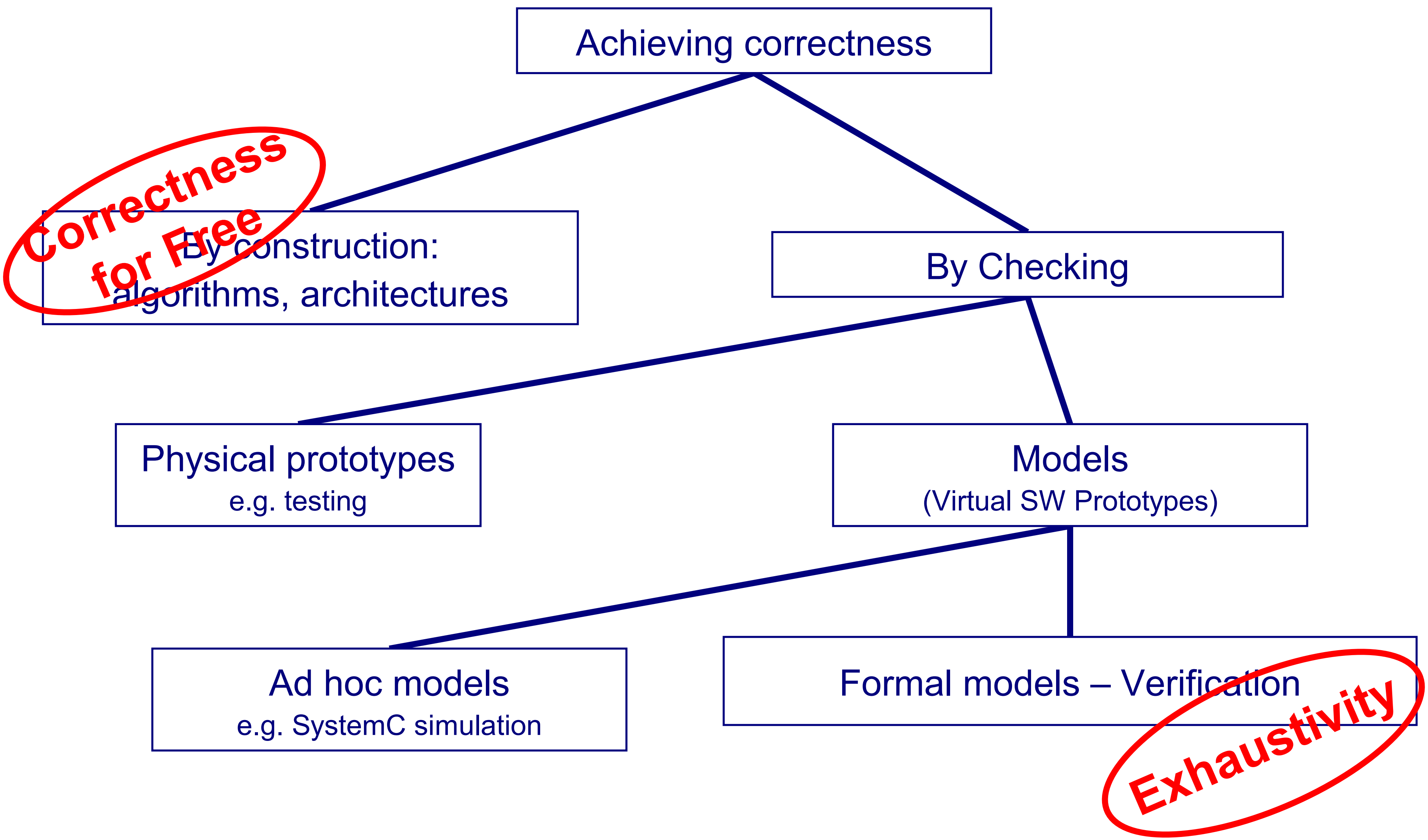
- Current status

- Beyond a posteriori verification
  - Component-based Construction
  - The BIP Component Framework
  - Verification at Design Time

- Conclusion

Achieving Correctness

Correctness: a system is correct if it meets its requirements



A posteriori Verification

System Model

Requirements

Should be:

- **faithful** *e.g. whatever property is satisfied for the model holds for the real system*
- generated **automatically** from system descriptions

Verification Method

YES, NO, DON'TKNOW

Should be:

- **consistent** *e.g. there exists some model satisfying them*
- **complete** *e.g. they tightly characterize the system's behavior*

- As a rule, for infinite state models all non trivial properties are undecidable e.g. bounded memory
- Intrinsically high complexity for finite state models (<u>state explosion problem</u>)

Beyond a posteriori Verification – Idea

Develop “divide and conquer” verification techniques

- Taking advantage of system structure and its properties e.g. for particular
  - architectures (e.g. client-server, star-like, time triggered)
  - programming models (e.g. synchronous, data-flow)
  - execution models (e.g. event triggered preempable tasks)

- For specific classes of properties such as deadlock-freedom, mutual exclusion, timeliness

Beyond *a posteriori* Verification – Principles

- Component-based and faithful construction of models from heterogeneous components

- Tight coupling between design and verification - Achieving correctness through
  - Constructivity: compositionality/composability techniques
  - Incrementality: reusing proofs for constituents
  - Property-preserving transformations

- Minimalistic verification framework
  - Focus on state invariants and deadlock-freedom

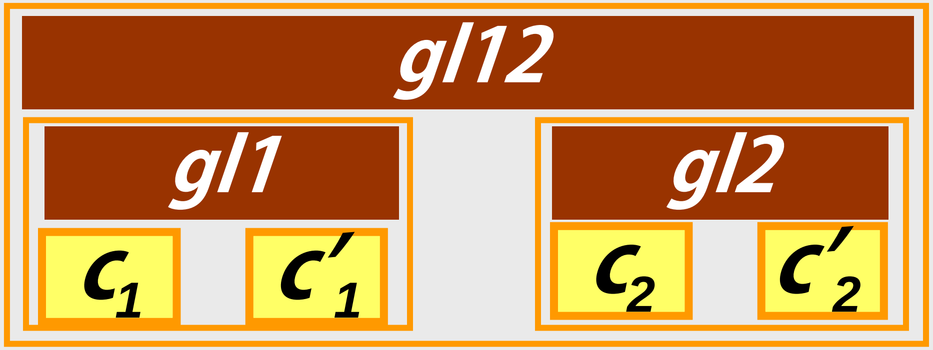
- Current status
- Beyond a posteriori verification
  - Component-based Construction
  - The BIP Component Framework
  - Verification at Design Time
- Conclusion

O V E R V I E W

Component-based Construction

Build a component C satisfying given requirements *f, from*

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

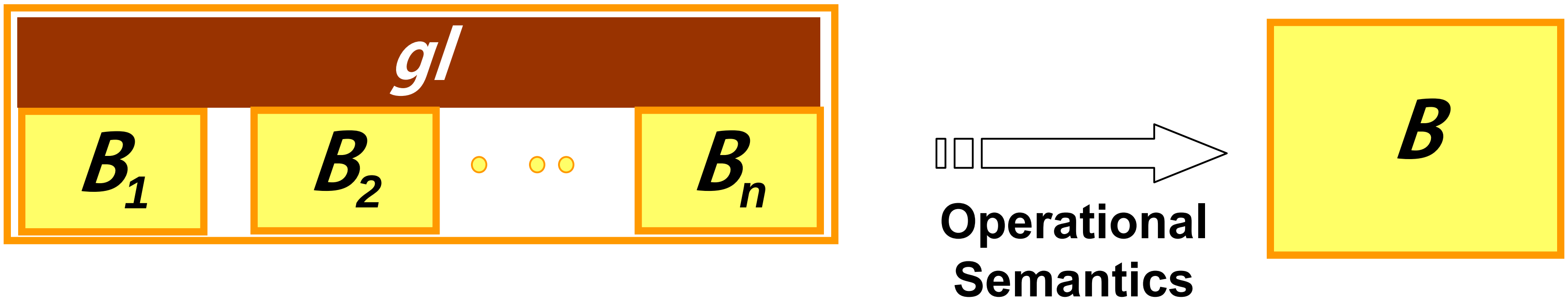


satisfies f

- Move from <u>single low-level</u> composition operators e.g. automata-based to <u>families of high-level</u> composition operators e.g. protocols, controllers
- We need a <u>unified composition paradigm</u> for describing and analyzing the coordination between components to formulate <u>heterogeneous</u> system designs in terms of tangible, well-founded and organized concepts

Glue Operators – Operational Semantics

We use operational semantics to define the meaning of a composite component – glue operators are "behavior transformers"



Glue Operators

- build interactions of composite components from the actions of the atomic components e.g. parallel composition operators
- can be specified by using a family of derivation rules (the Universal Glue)

Glue Operators – Operational Semantics

A **glue operator** is a set of derivation rules of the form

$$\frac{\{q_i \text{ - } a_i \rightarrow_i q'_i\}_{i \in I} \qquad \{\neg\, q_k \text{ - } a_k \rightarrow_k\}_{k \in K}}{(q_1, \ldots, q_n) \text{ - } a \rightarrow (q'_1, \ldots, q'_n)}$$

- $I, K \subseteq \{1, \ldots n\}$, $I \neq \emptyset$, $K \cap I = \emptyset$
- $a = \bigcup_{i \in I} a_I$ is an interaction
- $q'_i = q_i$ for $i \notin I$

Notice that, non deterministic choice and sequential composition are not glue operators

A **glue** is a set of glue operators

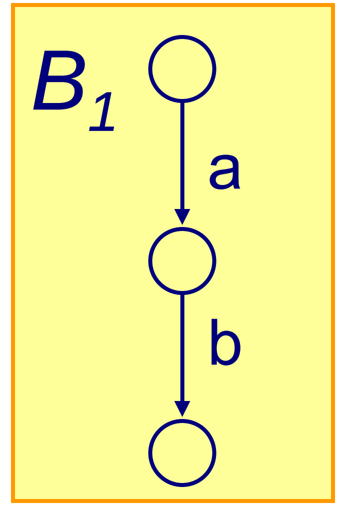
Glue Operators – Operational Semantics: Example

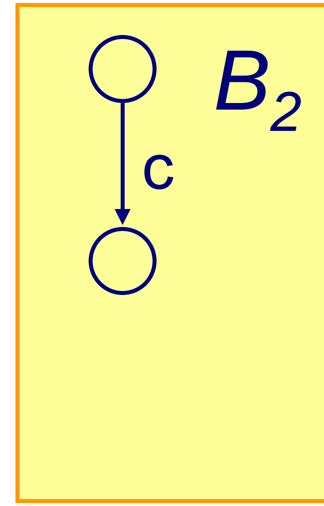
gl is defined by

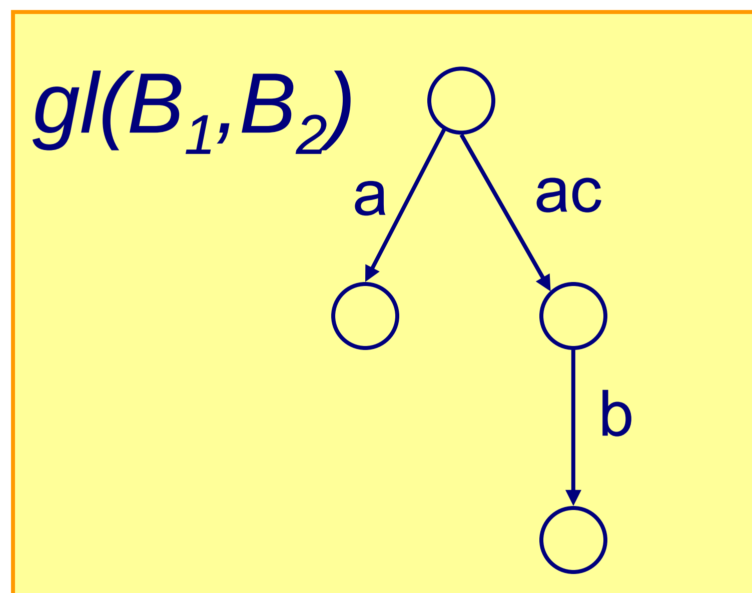
$$\frac{q_1 \xrightarrow{a} q'_1}{q_1 q_2 \xrightarrow{a} q'_1 q_2}$$

$$\frac{q_1 \xrightarrow{a} q'_1 \qquad q_2 \xrightarrow{c} q'_2}{q_1 q_2 \xrightarrow{ac} q'_1 q'_2}$$

$$\frac{q_1 \xrightarrow{b} q'_1 \qquad \neg\, q_2 \xrightarrow{c}}{q_1 q_2 \xrightarrow{b} q'_1 q_2}$$

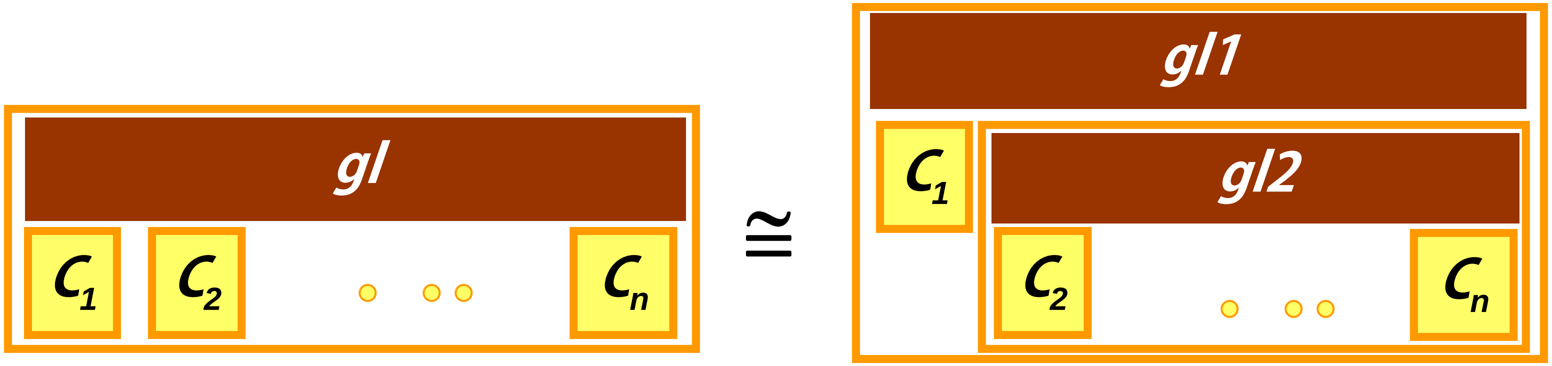




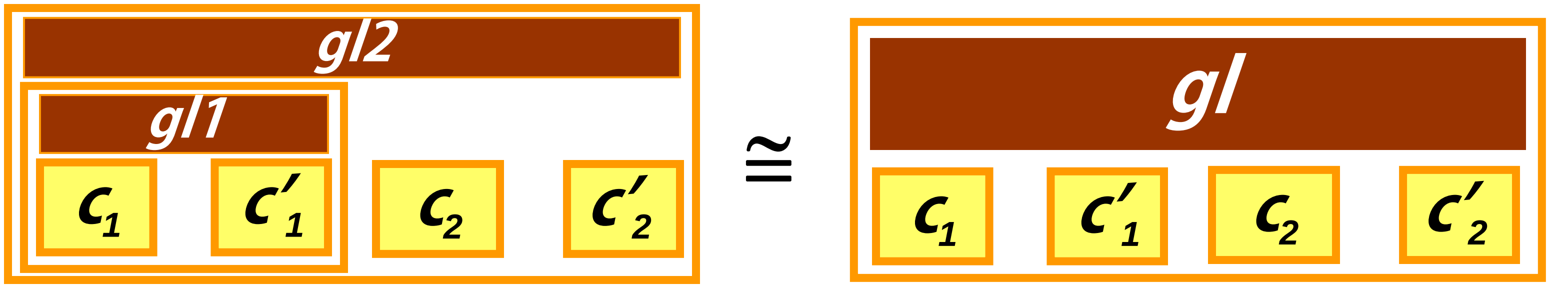


Glue Operators – Incremental Construction

1. Decomposition



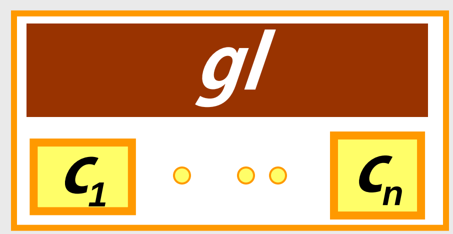
2. Flattening



Glue Operators – Compositionality

Build correct systems from correct components: rules for proving global properties from properties of individual components

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



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

We need compositionality results for the preservation of progress properties such as deadlock-freedom and liveness as well as extra-functional properties

Glue Operators – Composability

Essential properties of components are preserved when they are composed

sat P and

sat P'

implies

sat $P \wedge P'$

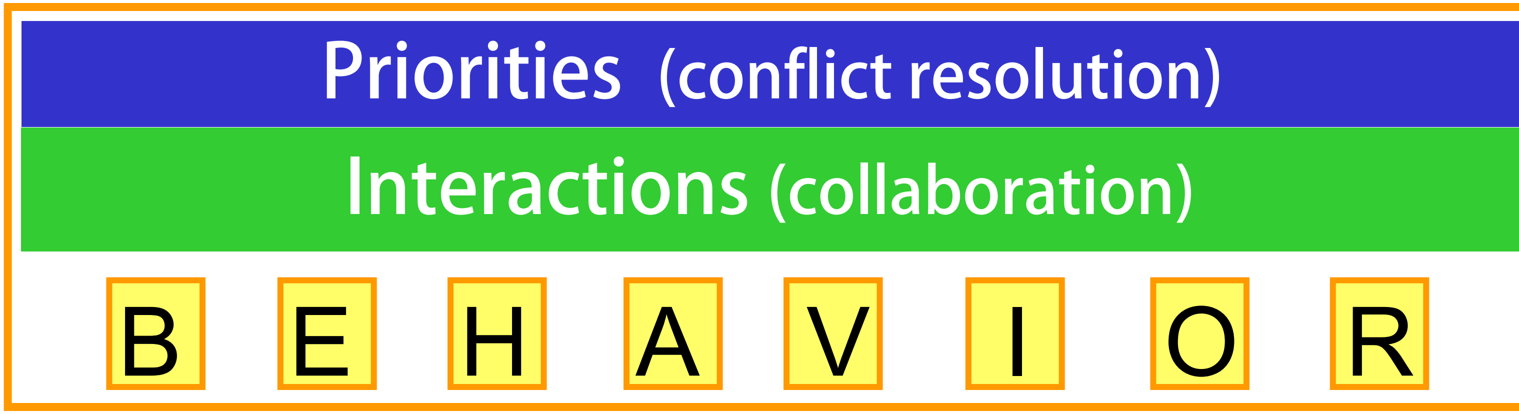
Property stability phenomena are poorly understood.
We need composability results e.g. non interaction of features in middleware, composability of scheduling algorithms, composability of web services

- Current status
- Beyond a posteriori verification
  - Component-based Construction
  - The BIP Component Framework
  - Verification at Design Time
- Conclusion

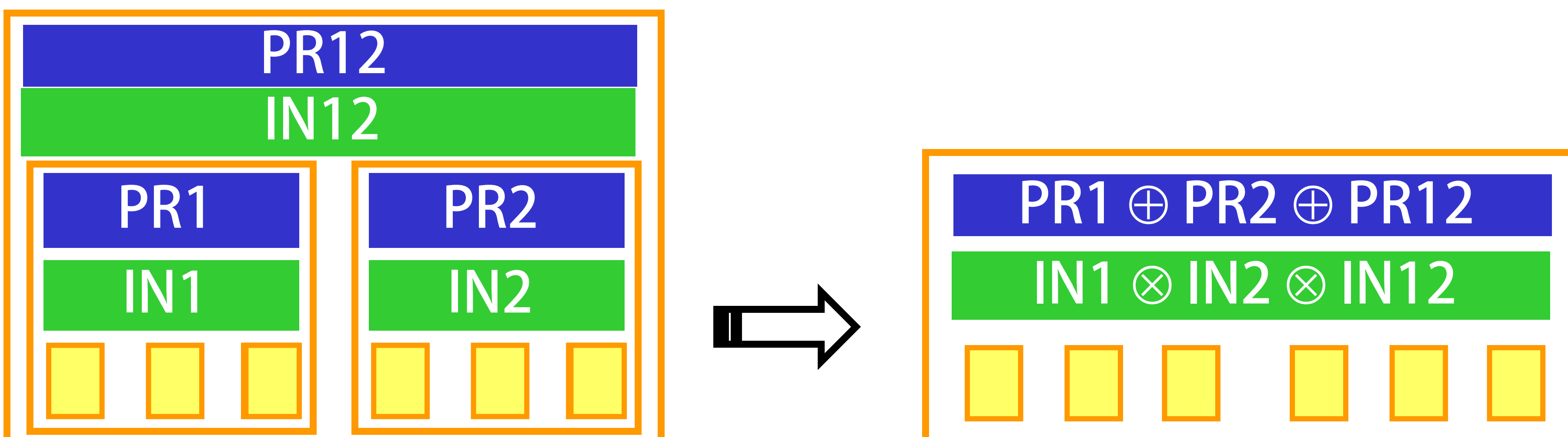
OVERVIEW

BIP – Basic Concepts

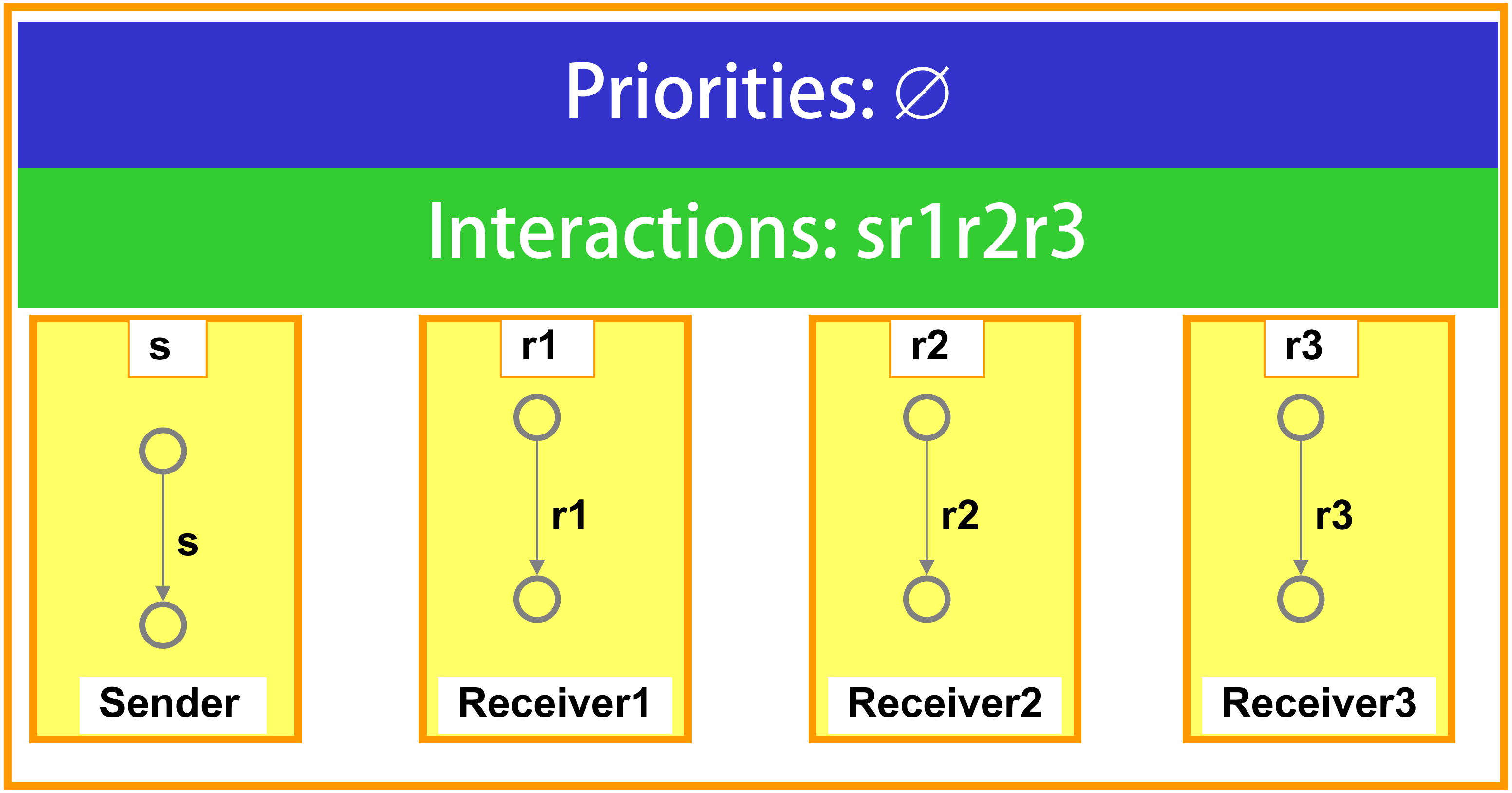
Layered component model



Composition operation parameterized by glue IN12, PR12

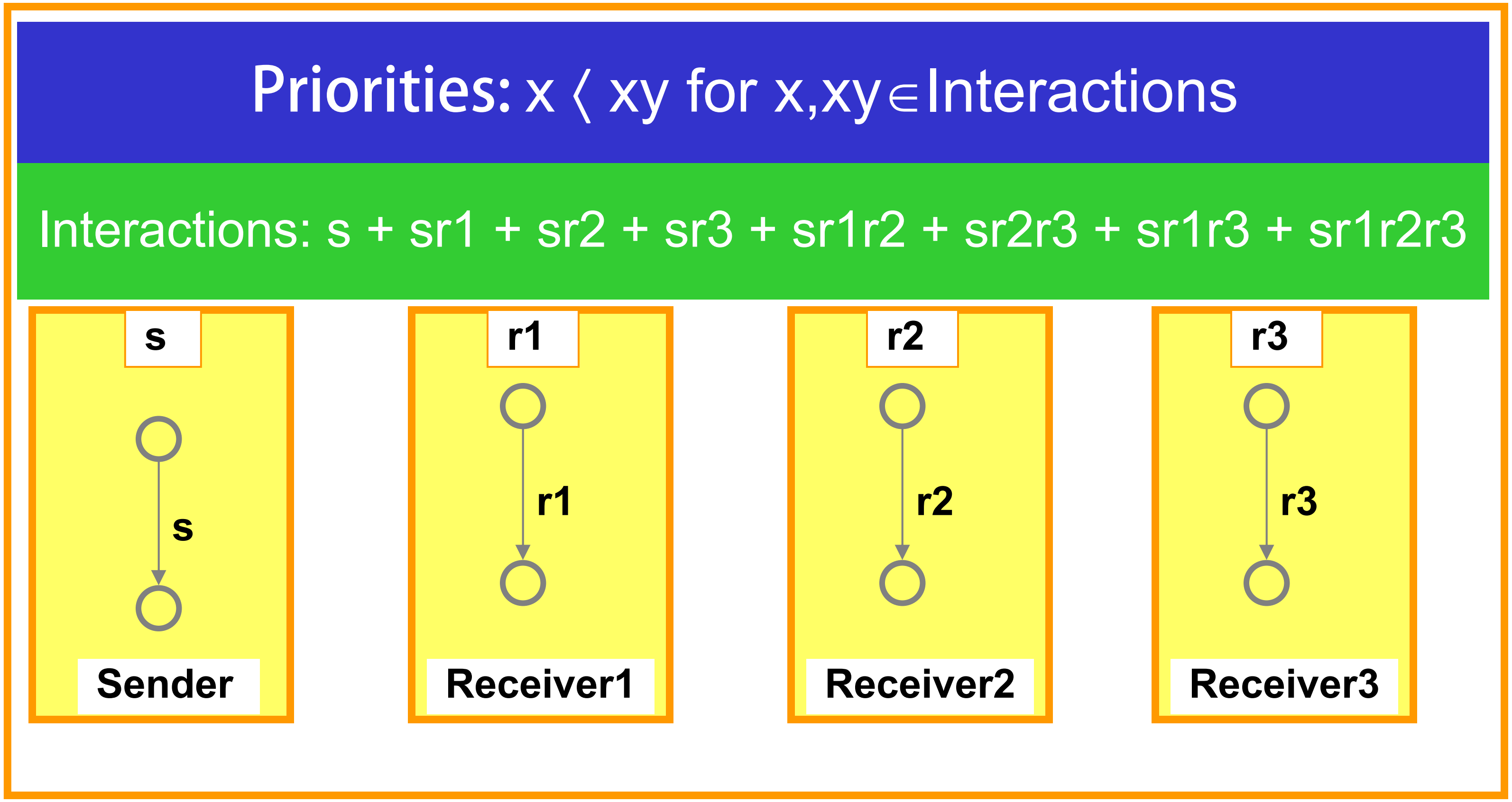


BIP – Basic Concepts



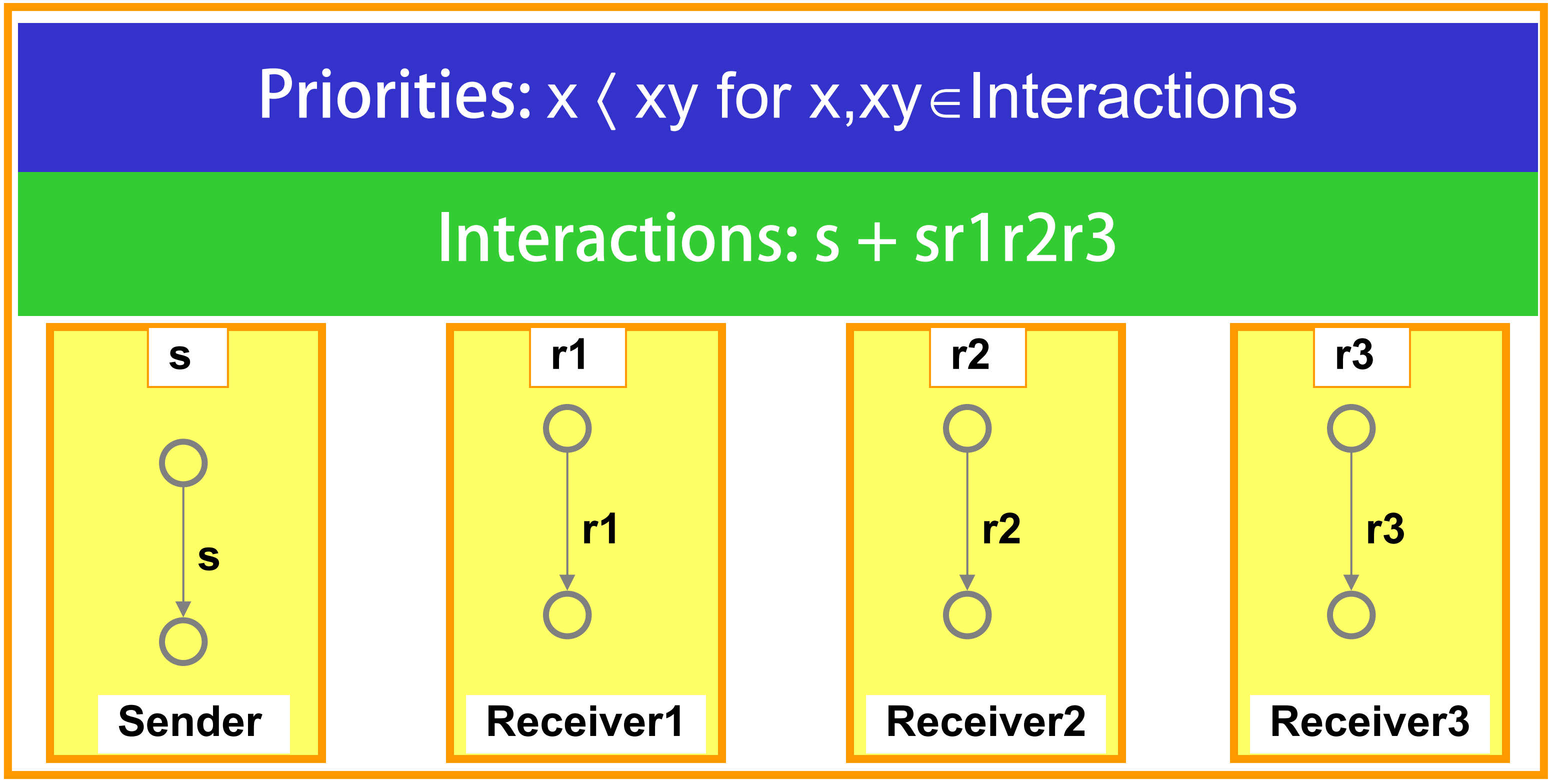
Rendezvous

BIP – Basic Concepts



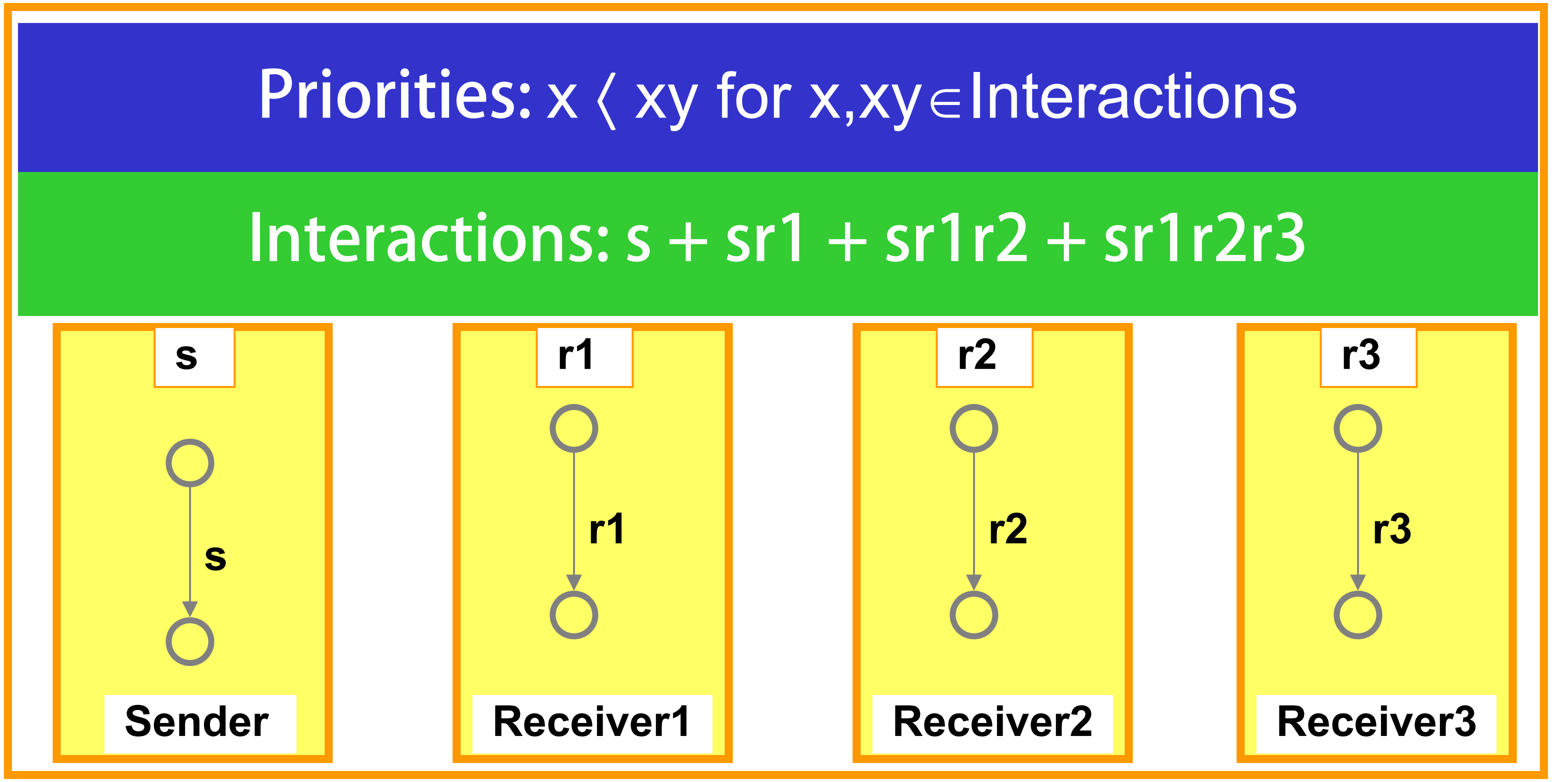
Broadcast

BIP – Basic Concepts



Atomic Broadcast

BIP – Basic Concepts



Causal Chain

BIP – Basic Concepts: Semantics

- a set of atomic components $\{B_i\}_{i=1..n}$ where $B_i = (Q_i, 2^{Pi}, \rightarrow_i)$
- a set of interactions γ
- priorities π, partial order on interactions

$$\left.\begin{array}{l}\{B_i\}_{i=1..n}\\ \gamma\\ \pi\end{array}\right\} \pi\, \gamma\, (B1,.., Bn)$$

Interactions

$$\frac{a \in \gamma \wedge \forall i \in [1,n] \;\; q_i \text{-}\, a \cap P_i \rightarrow_i q'_i}{(q_1 ,.., q_n) \text{-}\, a \rightarrow_\gamma (q'_1 ,.., q'_n) \text{ where } q'_l = q_l \text{ if } a \cap P_i = \emptyset}$$

Priorities

$$\frac{q\text{-}\, a \rightarrow_\gamma q' \wedge \neg (\exists\, q\text{-}\, b \rightarrow_\gamma \wedge a \,\pi\, b\,)}{q\text{-}\, a \rightarrow_\pi q'}$$

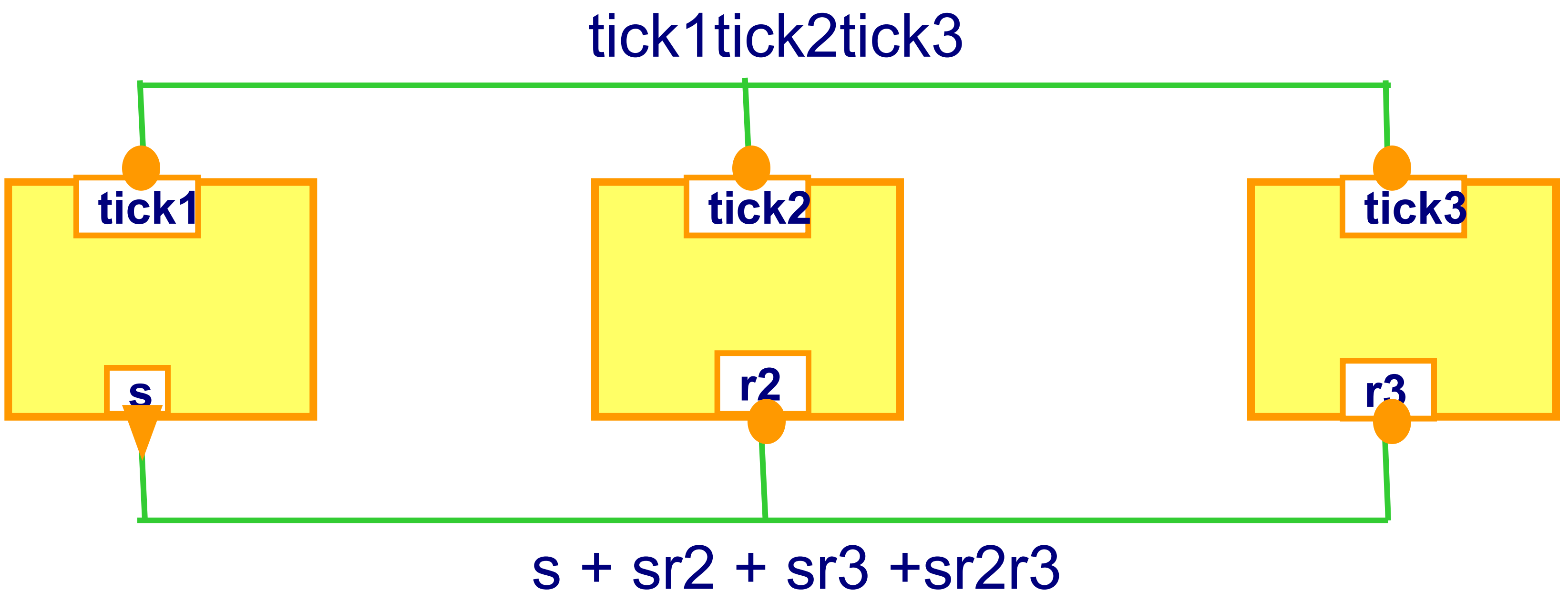
OVERVIEW

- Current status
- Beyond a posteriori verification
  - Component-based Construction
  - The BIP Component Framework
  - Verification at Design Time
- Conclusion

Modeling Interactions – Simple Connectors

Express interactions by combining two protocols: rendezvous and broadcast

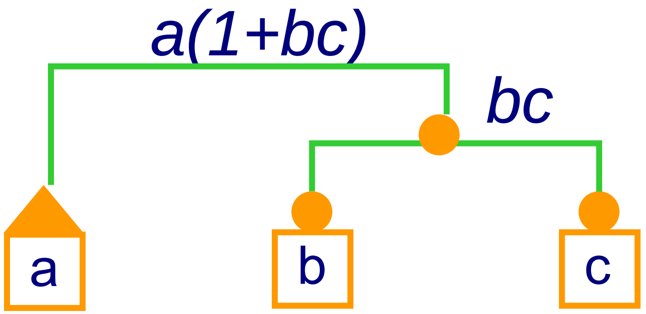
- A ***connector*** is a set of ports that can be involved in an interaction
- Port attributes (***trigger*** ▼, ***synchron*** ●) are used to model rendezvous and broadcast.
- An ***interaction*** of a connector is a set of ports such that: either it contains some trigger or it is maximal.



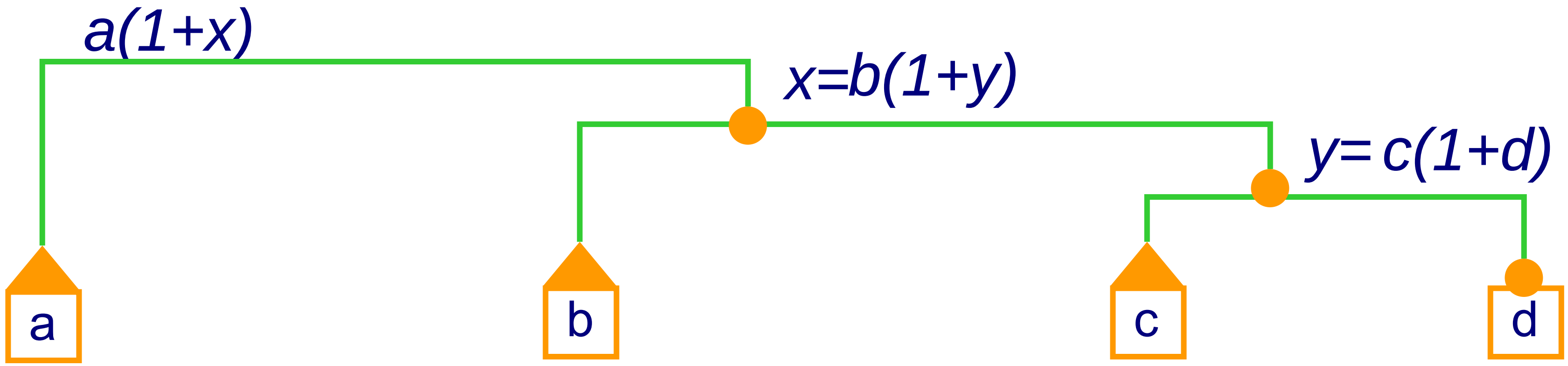
Modeling Interactions – Hierarchical Connectors

Atomic Broadcast:

a+abc

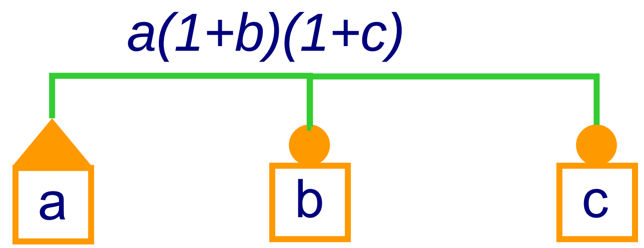


Causality chain: a+ab+abc+abcd



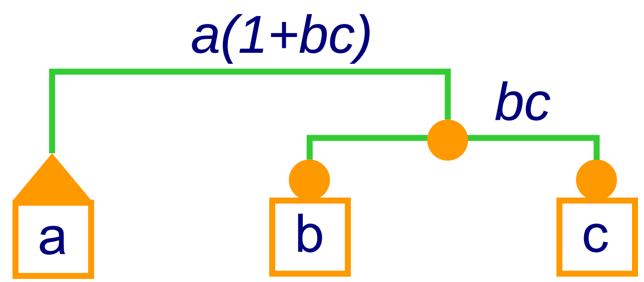
Broadcast

a’bc



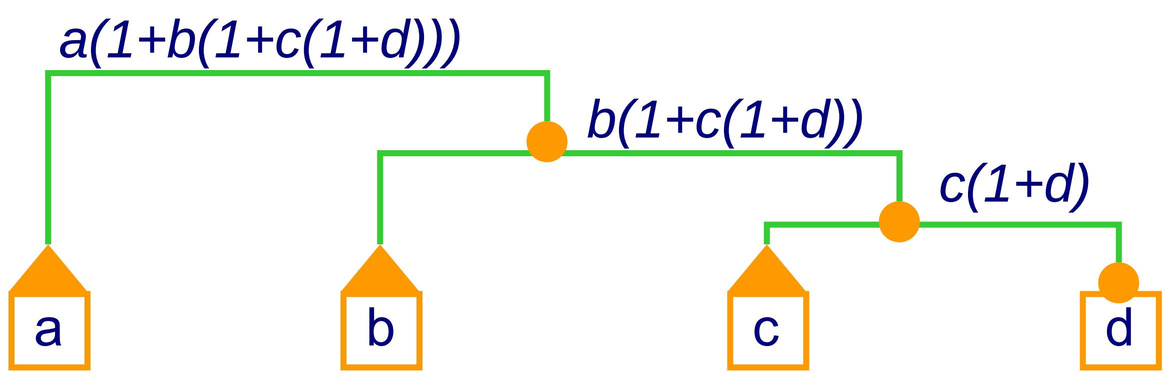
Atomic Broadcast

a’[bc]

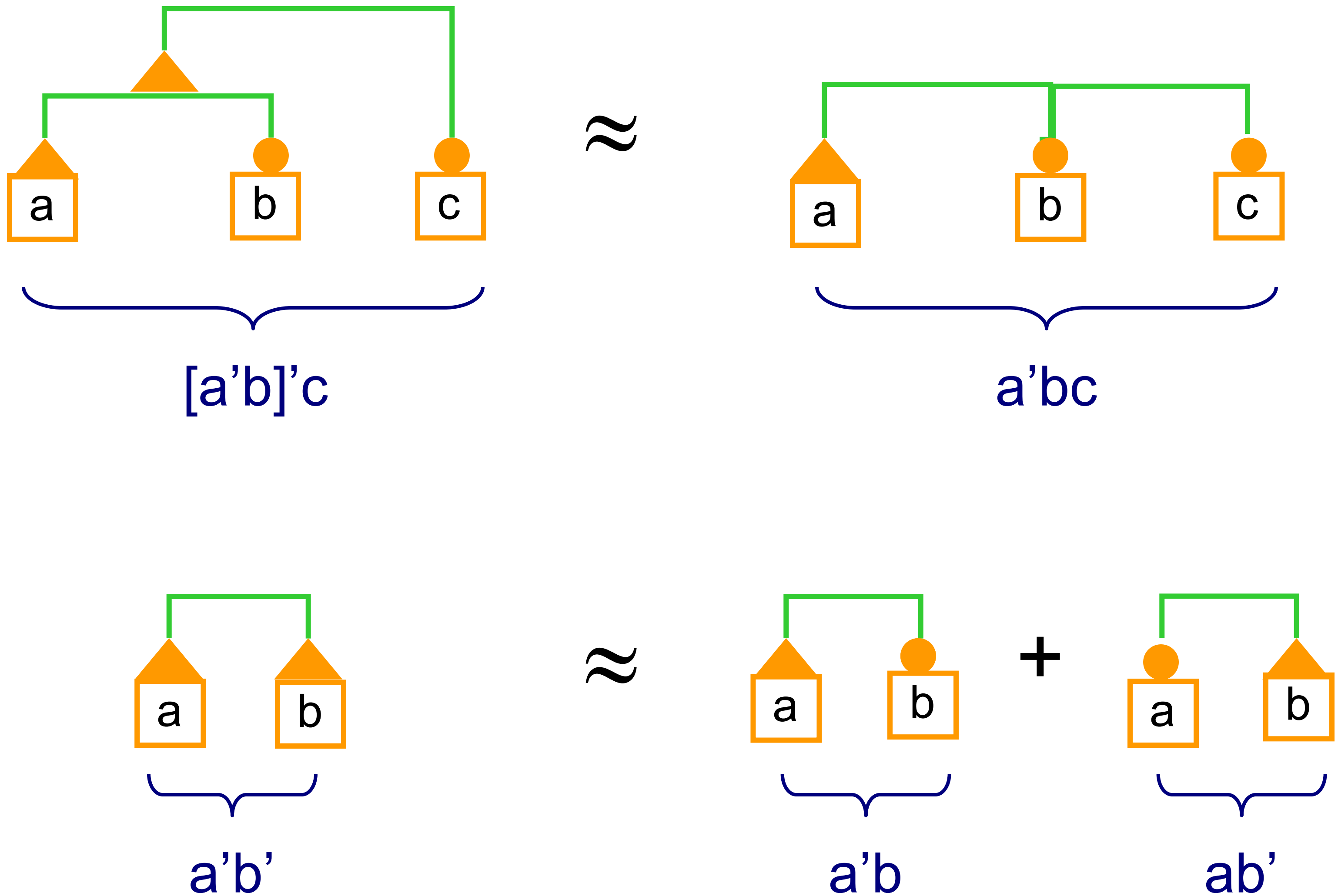


Causality chain

a’[b’[c’d]]



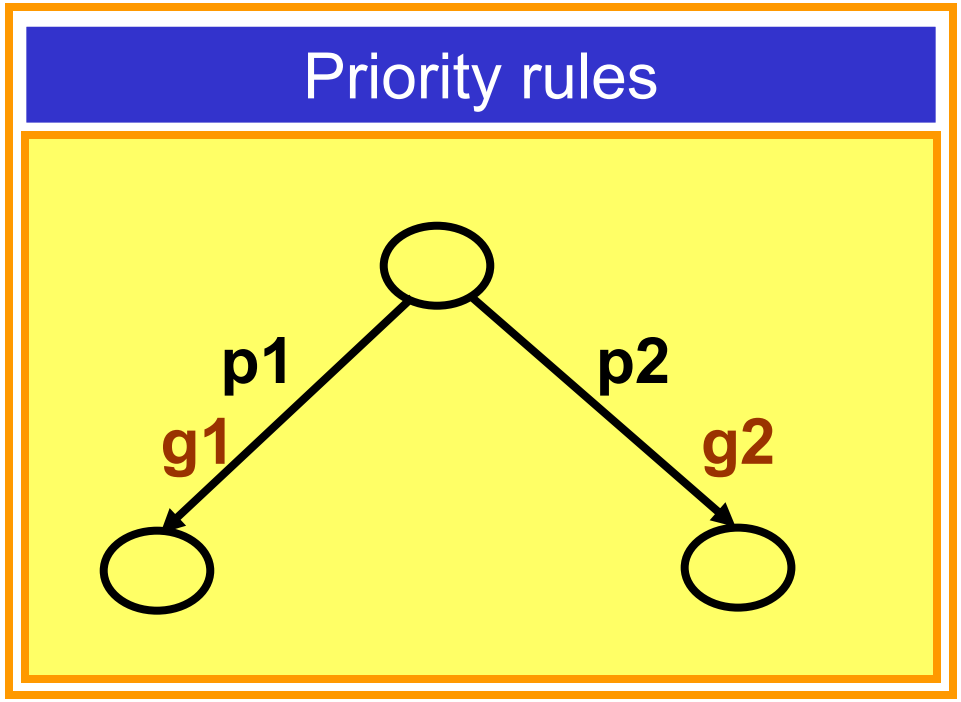
Modeling Interactions – The Algebra of Connectors



- Current status
- Beyond a posteriori verification
  - Component-based Construction
  - The BIP Component Framework
  - Verification at Design Time
- Conclusion

O V E R V I E W

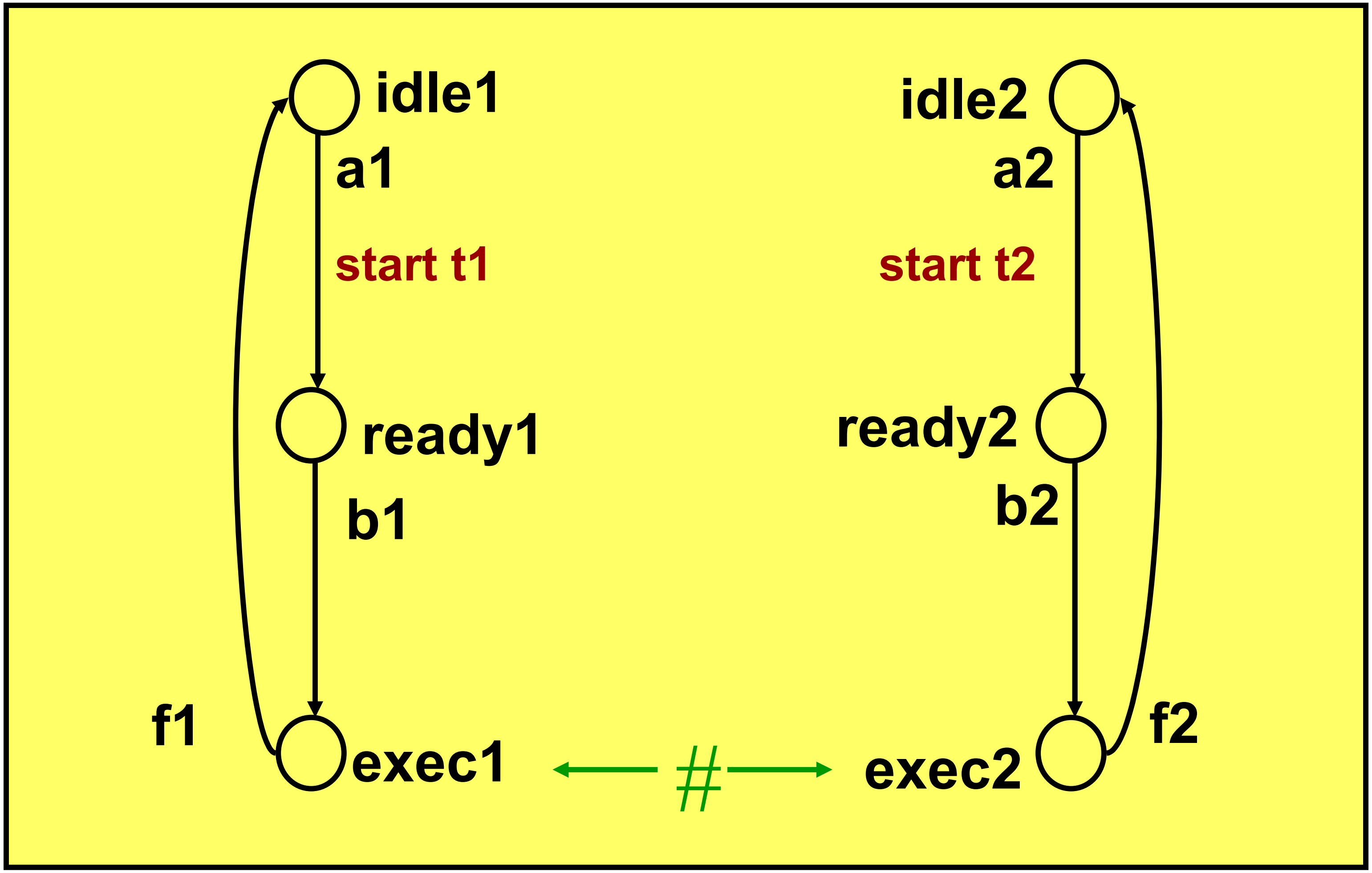
Modeling Priorities – Definition



Priority rule	**Restricted guard g1'**
$true \rightarrow p1 \langle p2$	$g1' = g1 \wedge \neg g2$
$C \rightarrow p1 \langle p2$	$g1' = g1 \wedge \neg(C \wedge g2)$

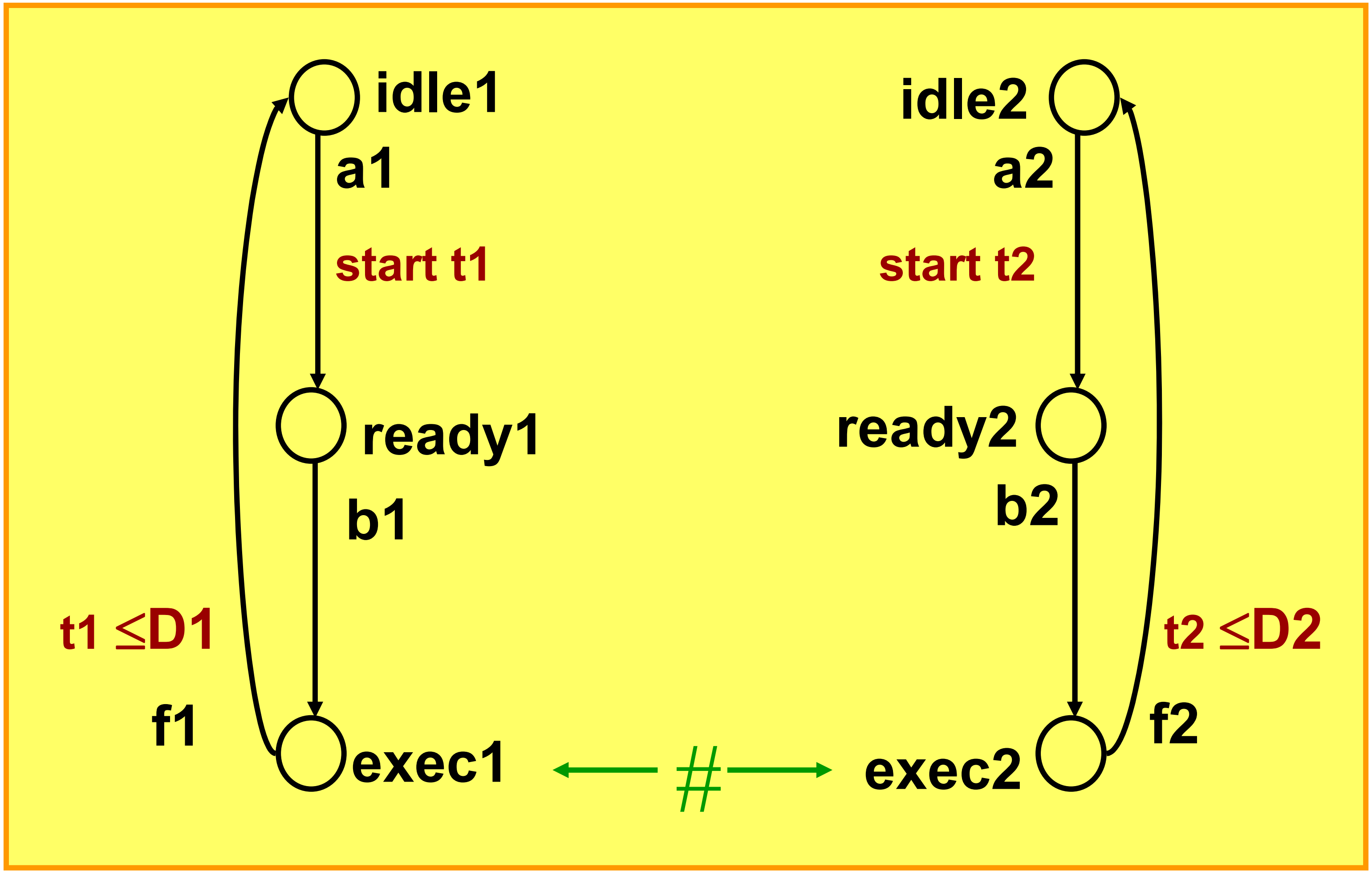
Modeling Priorities – FIFO policy

PR : $t1 \leq t2 \rightarrow b1 \langle b2 \qquad t2 < t1 \rightarrow b2 \langle b1$

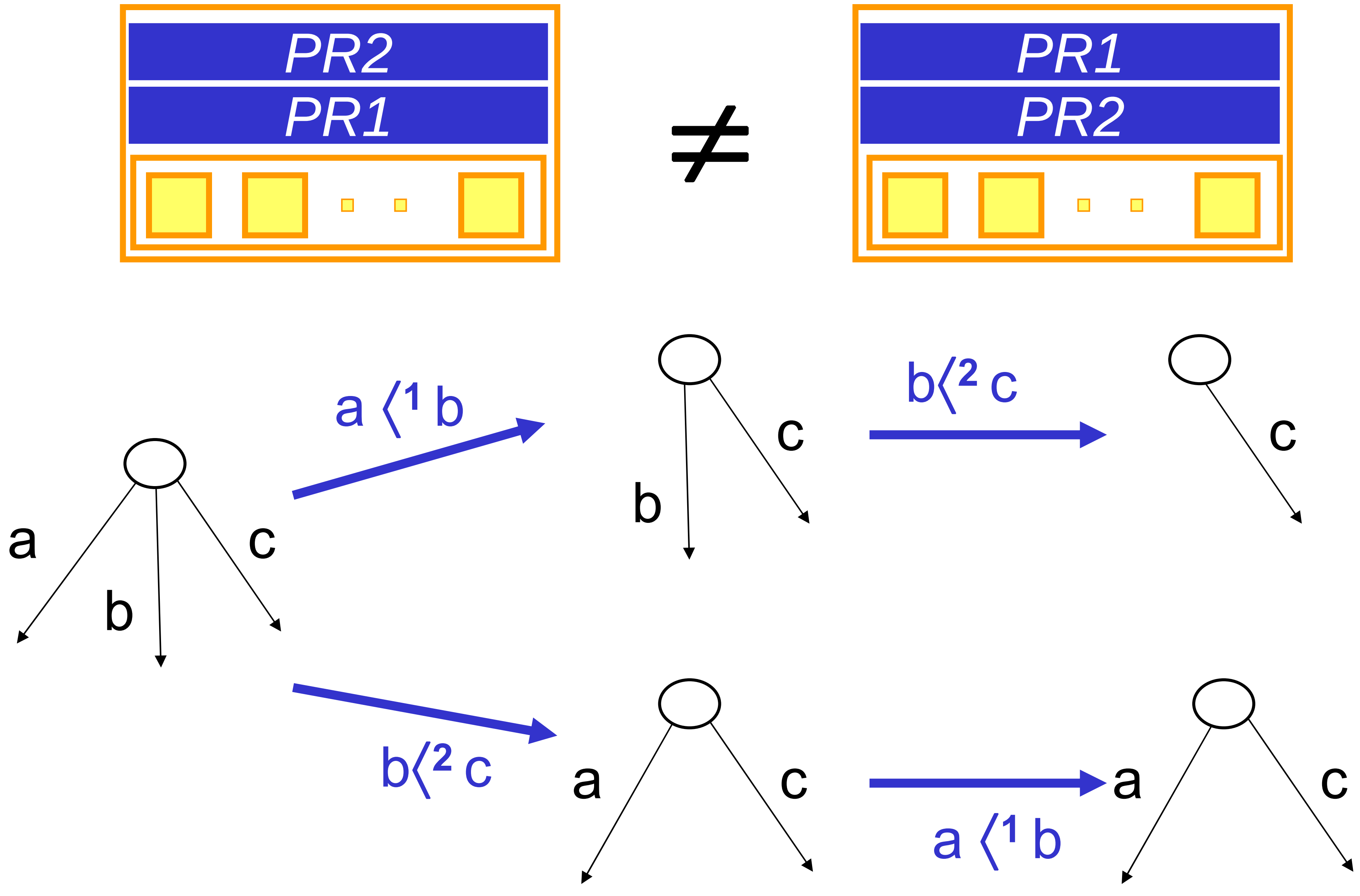


Modeling Priorities – EDF policy

PR: $D1\text{-}t1 \leq D2\text{-}t2 \rightarrow b2 \langle b1$ $D2\text{-}t2 < D1\text{-}t1 \rightarrow b1 \langle b2$

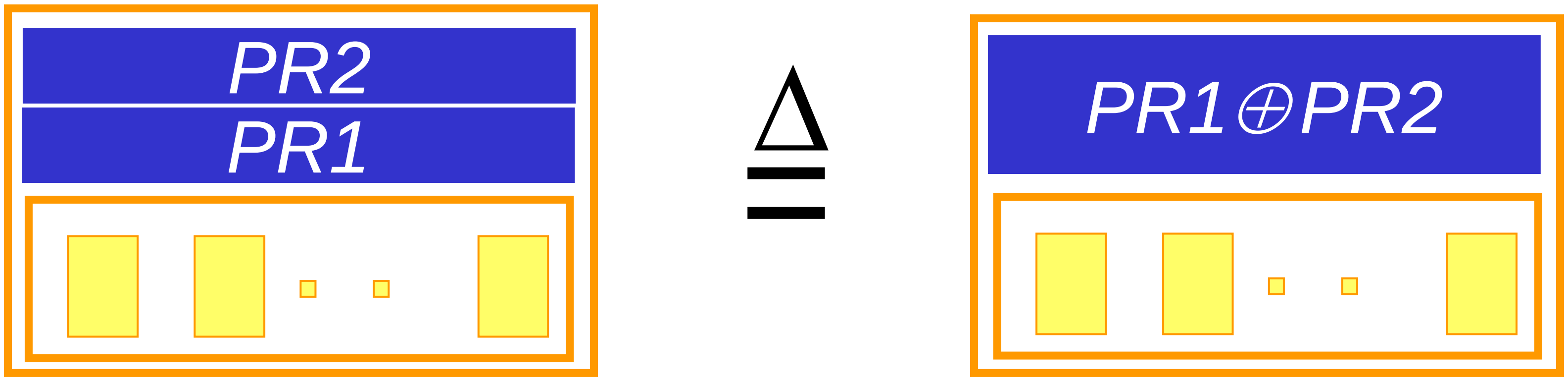


Modeling Priorities – Composability



Modeling Priorities – Composability

We take:



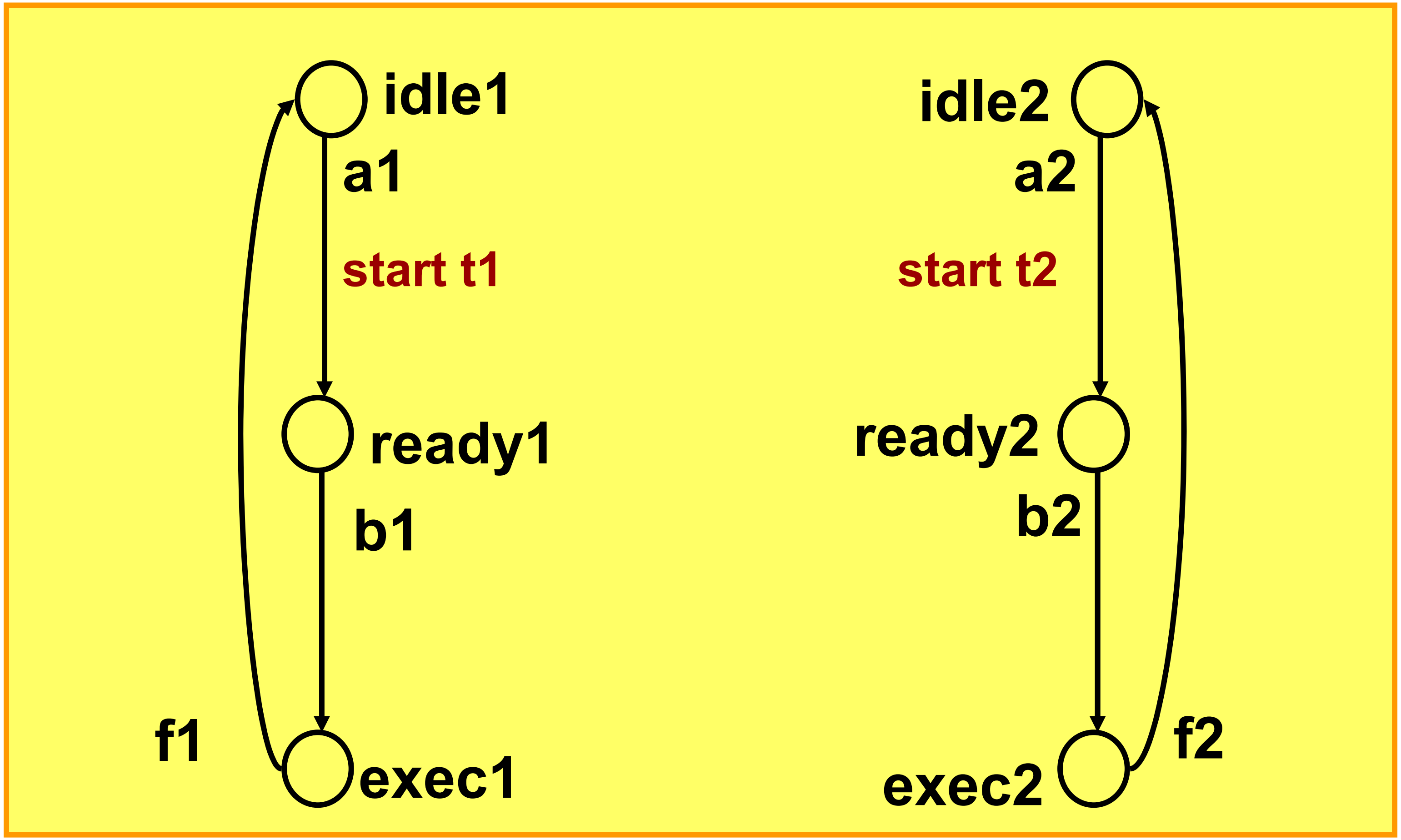
PR1⊕ PR2 is the least priority containing PR1∪PR2

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

Modeling Priorities – Mutual Exclusion + FIFO policy

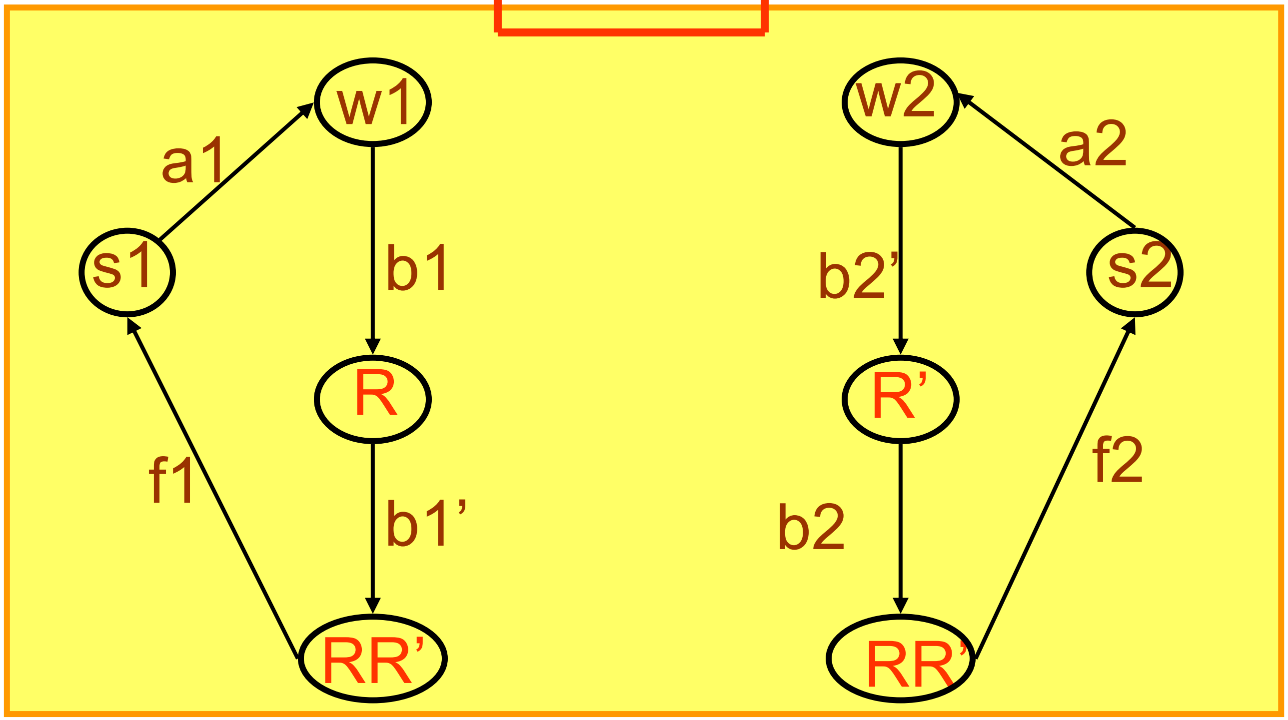
$t1 \leq t2 \rightarrow b1 \langle b2$	$t2 < t1 \rightarrow b2 \langle b1$
$true \rightarrow b1 \langle f2$	$true \rightarrow b2 \langle f1$



Modeling Priorities – Mutual Exclusion: Example

PR : b1 ⟨ f2   b2 ⟨ { f1, b1'}   (mutex on R)

PR': b2' ⟨f1   b1' ⟨ { f2, b2 }   (mutex on R')



Risk of deadlock: PR⊕PR' is not defined

OVERVIEW

- Current status
- Beyond a posteriori verification
  - Component-based Construction
  - The BIP Component Framework
  - Verification at Design Time
- Conclusion

Expressiveness for Component-based Systems

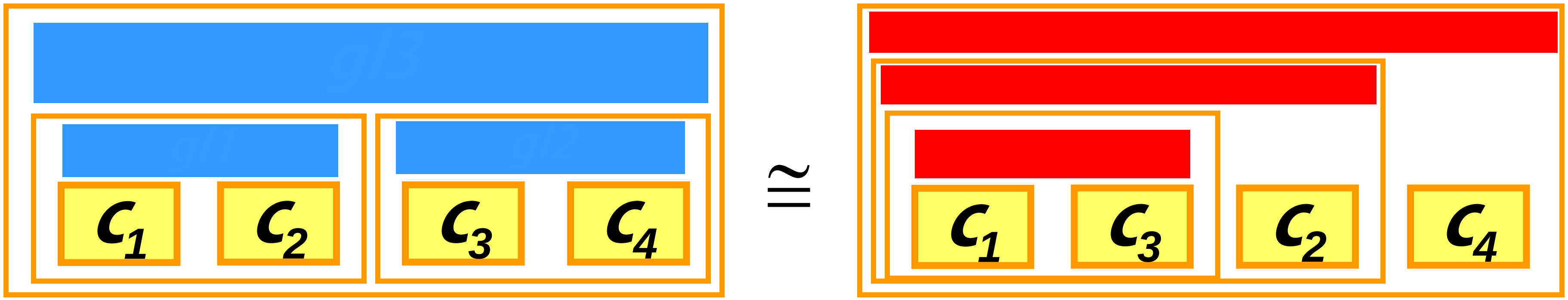
- Different from the usual notion of expressiveness!
- Based on strict separation between glue and behavior

Given two glues G_1 , G_2

G_2 ***is strongly more expressive than*** G_1

if for any component built by using G_1 and $\mathcal{C}_0$

there exists an equivalent component built by using G_2 and $\mathcal{C}_0$



Expressiveness for Component-based Systems

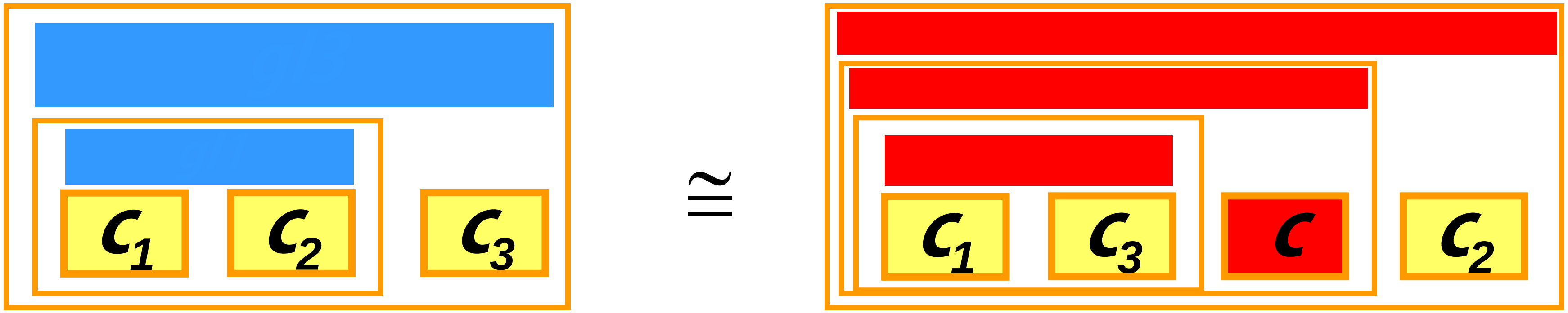
Given two glues G_1 , G_2

G_2 ***is weakly more expressive than*** G_1

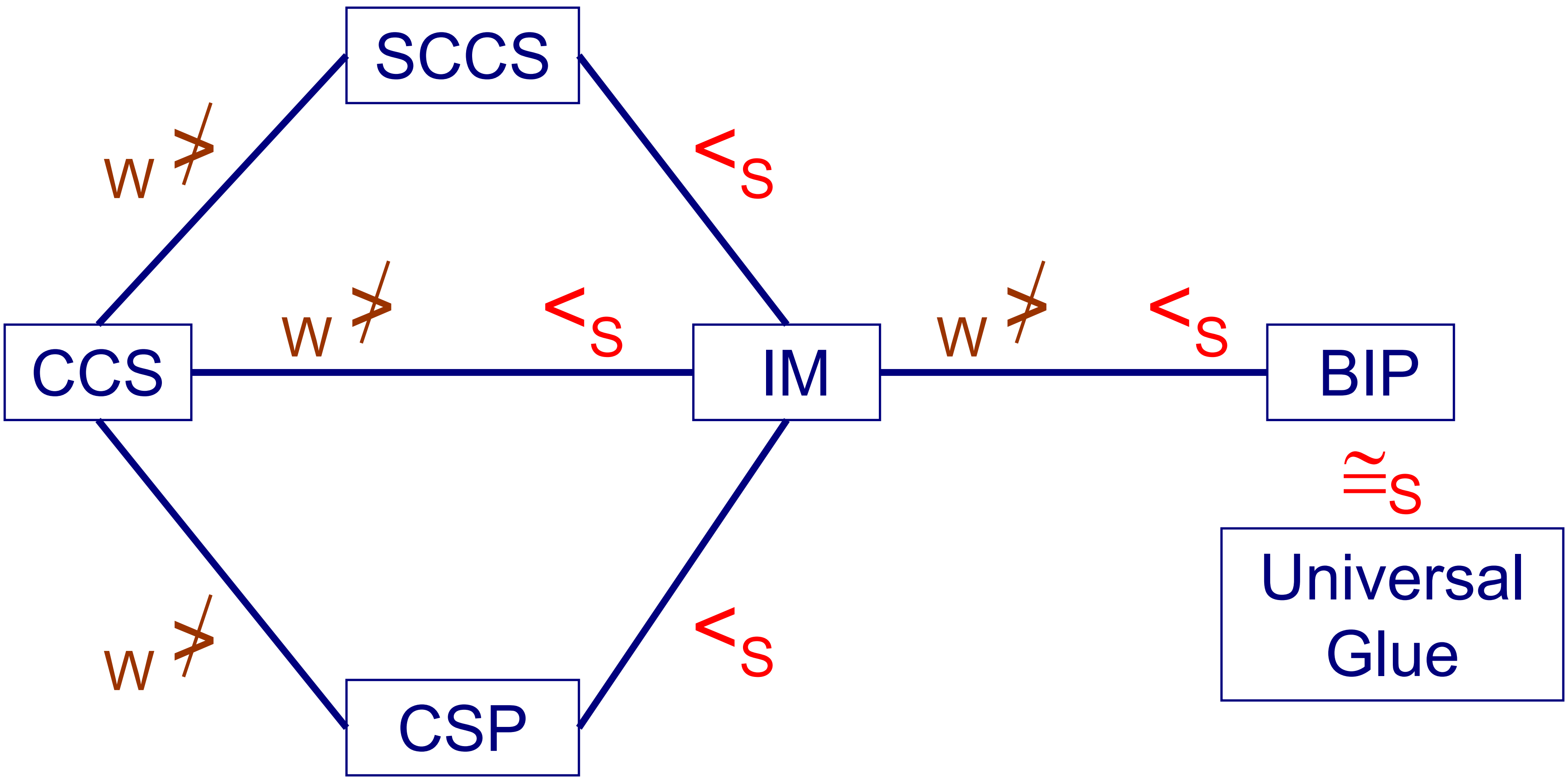
if for any component built by using G_1 and $\mathcal{C}_0$

there exists an equivalent component built by using G_2 and $\mathcal{C}_0 \cup \mathcal{C}$

where $\mathcal{C}$ is a finite set of coordination behaviors.



Expressiveness for Component-based Systems



[Bliudze&Sifakis, Concur 08]

O V E R V I E W

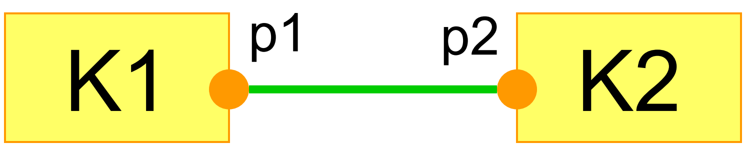
- Current status

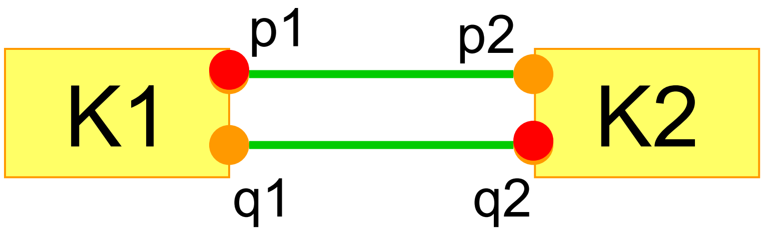
- Beyond a posteriori verification
  - Component-based Construction
  - The BIP Component Framework
  - Verification at Design Time

- Conclusion

Compositional Verification

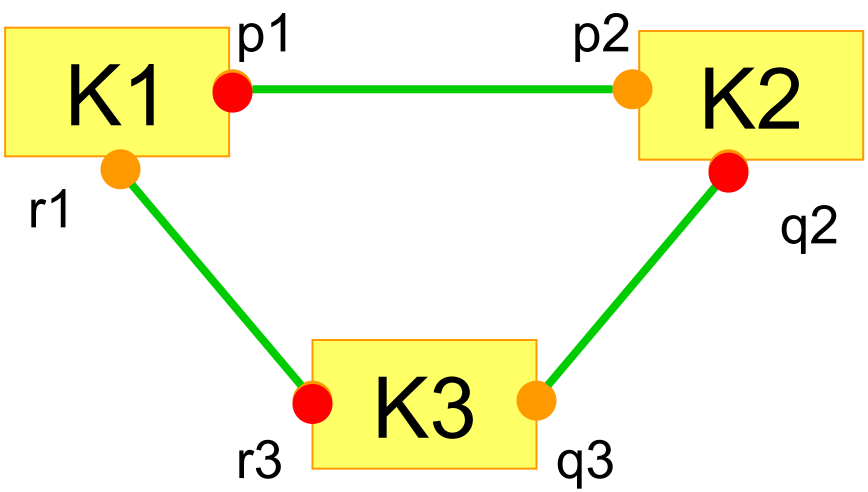
Verify global deadlock-freedom of a system
by separate analysis of the components and of the architecture.





Potential deadlock

$D = en(p1) \wedge \neg en(p2) \wedge en(q2) \wedge \neg en(q1)$

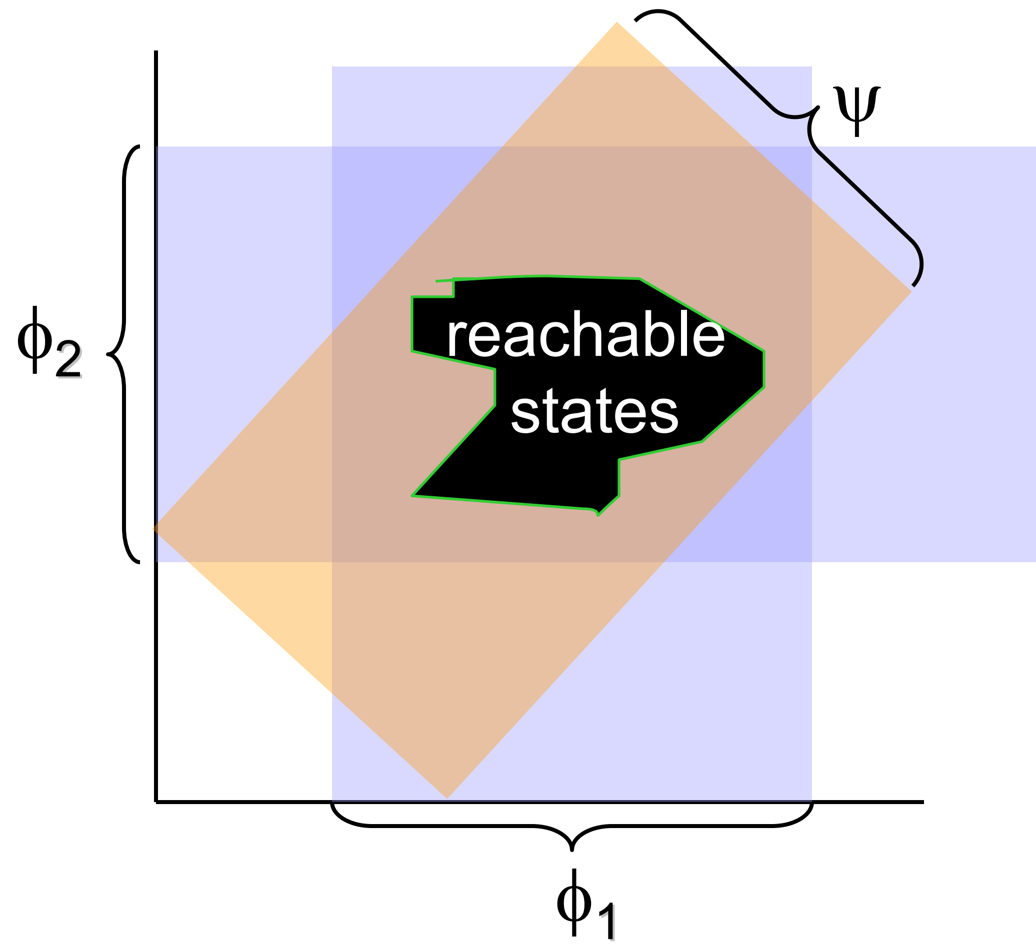


Potential deadlock

$D = en(p1) \wedge \neg en(p2) \wedge en(q2) \wedge \neg en(q3) \wedge en(r3) \wedge \neg en(r1)$

Compositional Verification: D-Finder

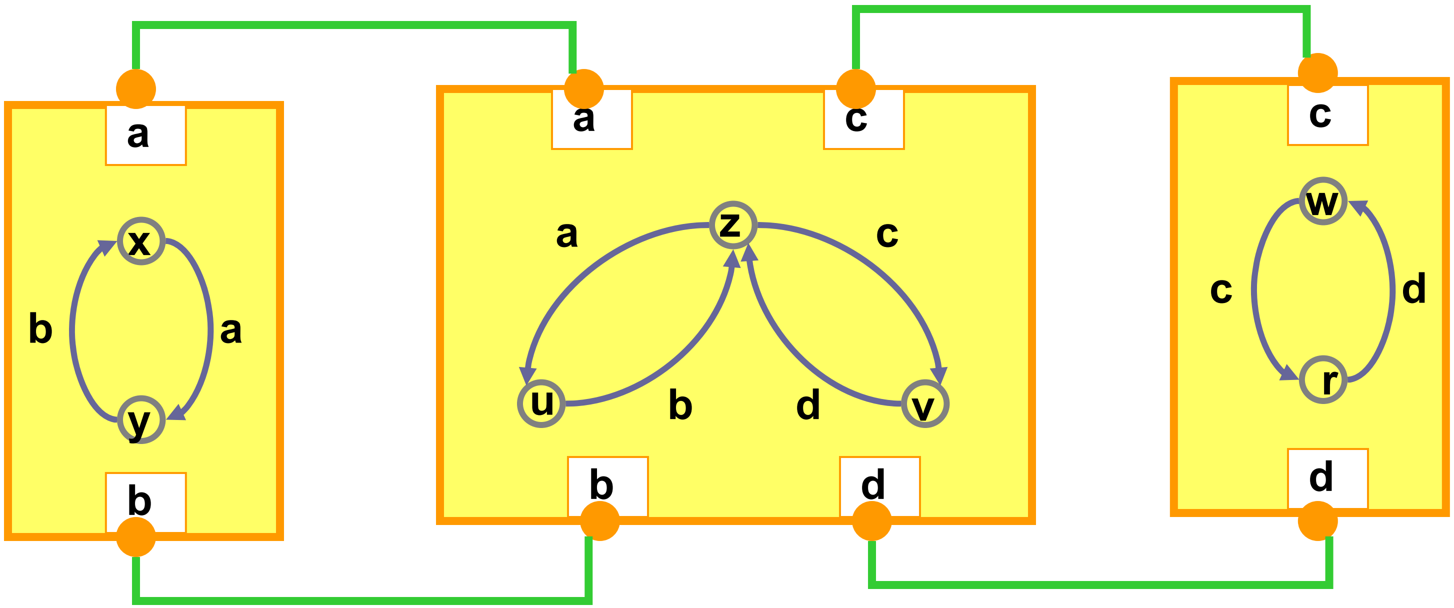
$$\frac{B_1 \models \Box\, \phi_1 \quad B_2 \models \Box\, \phi_2 \quad \psi \in II(\gamma(B_1, B_2), \phi_1, \phi_2) \quad \phi_1 \wedge \phi_2 \wedge \psi \Rightarrow \chi}{\gamma(B_1, B_2) \models \Box \chi}$$



Method:

Eliminate potential deadlocks D by computing compositionally global invariants χ such that $\chi \wedge D = \text{false}$

Compositional Verification – D-Finder



$x \Rightarrow y \vee u$
$y \Rightarrow x \vee z$

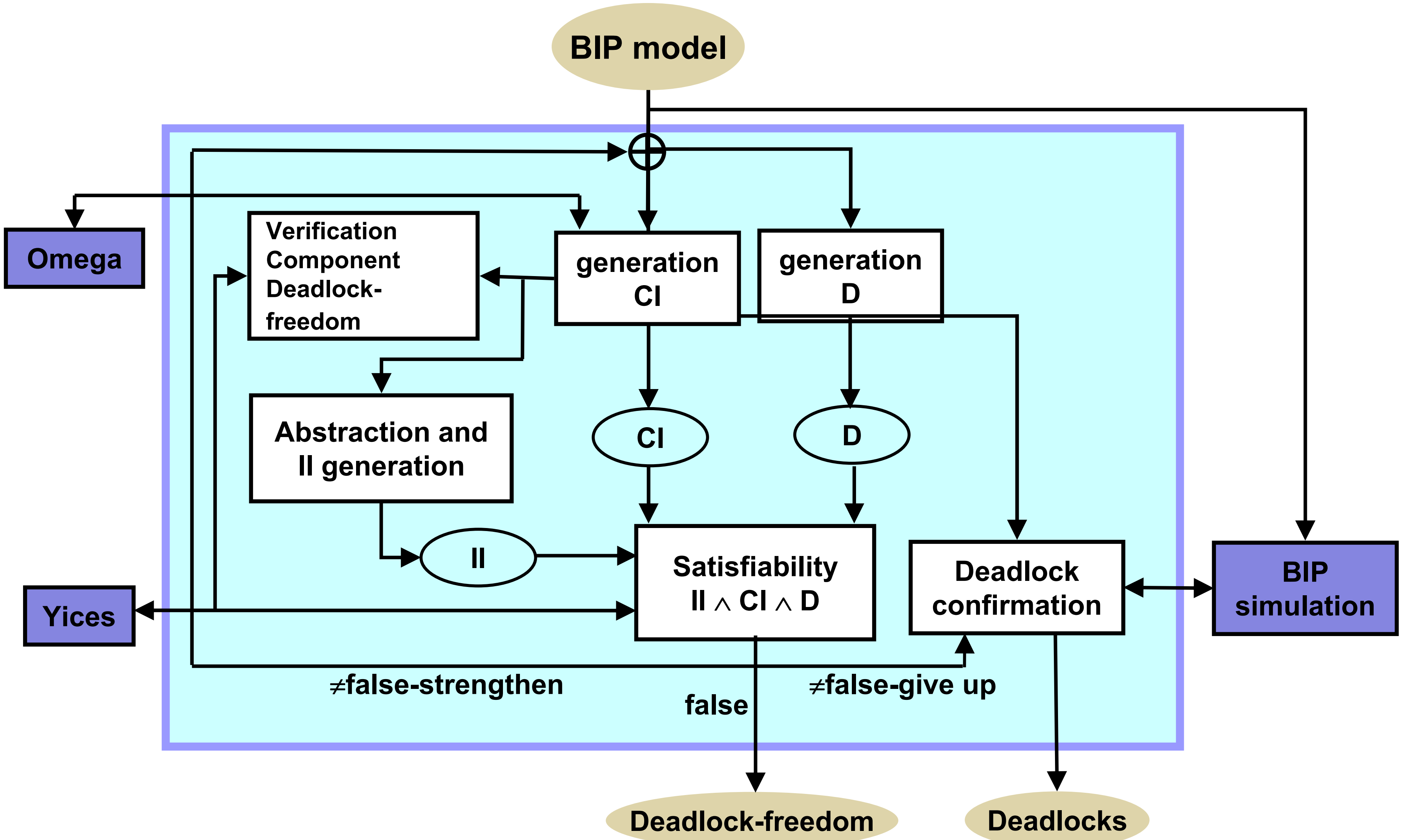
$z \Rightarrow (y \vee u) \wedge (v \vee r)$
$u \Rightarrow x \vee z$
$v \Rightarrow w \vee z$

$w \Rightarrow (v \vee r)$
$r \Rightarrow w \vee z$

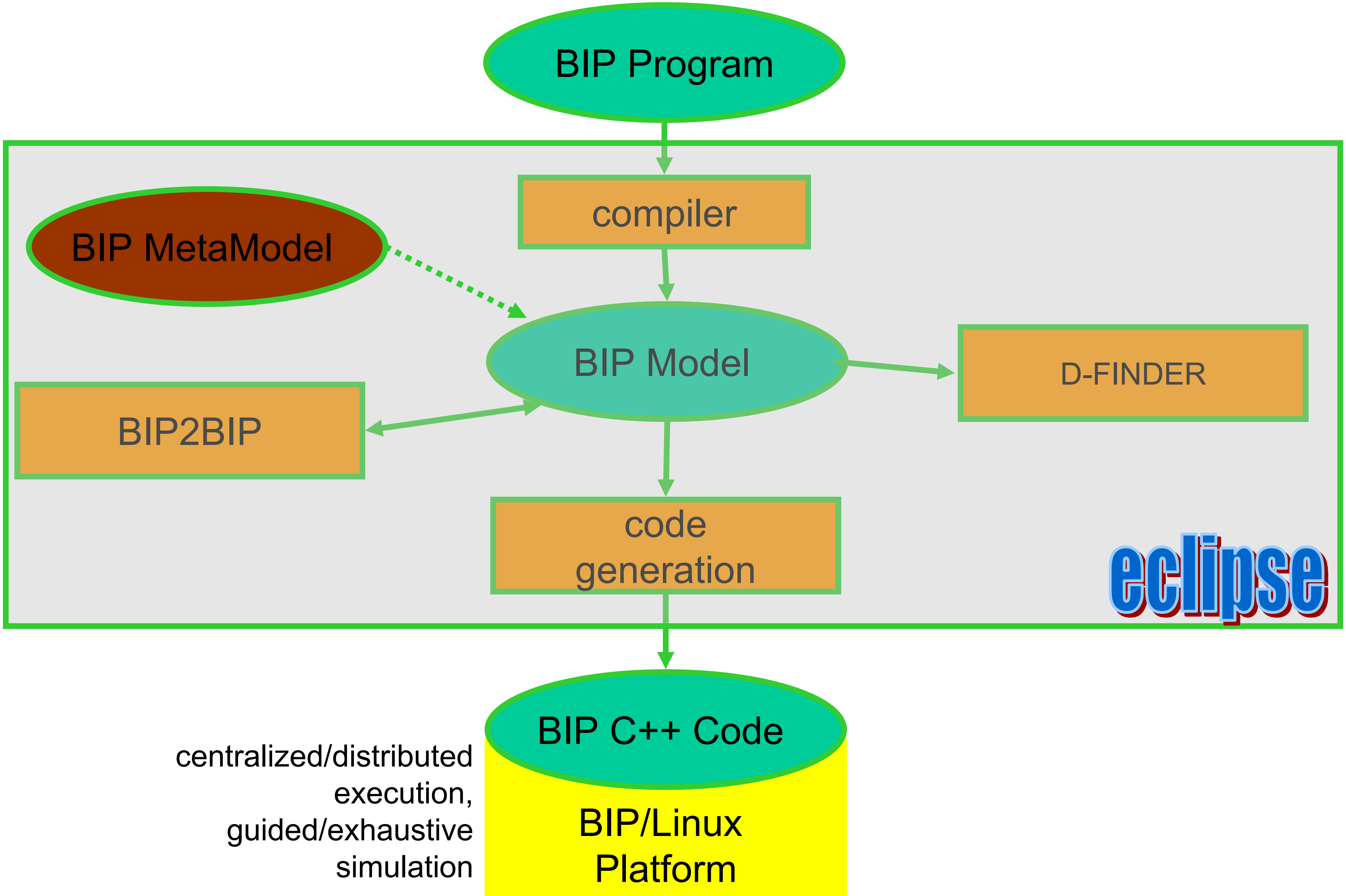
Minimal solutions define invariants :

- Component invariants: $x \vee y$, $z \vee u \vee v$, $w \vee r$
- Interaction invariants: $x \vee u$, $z \vee y \vee v$, $z \vee y \vee r$, $z \vee u \vee r$, $w \vee v$

Compositional Verification – D-Finder



Overall BIP Toolset Architecture



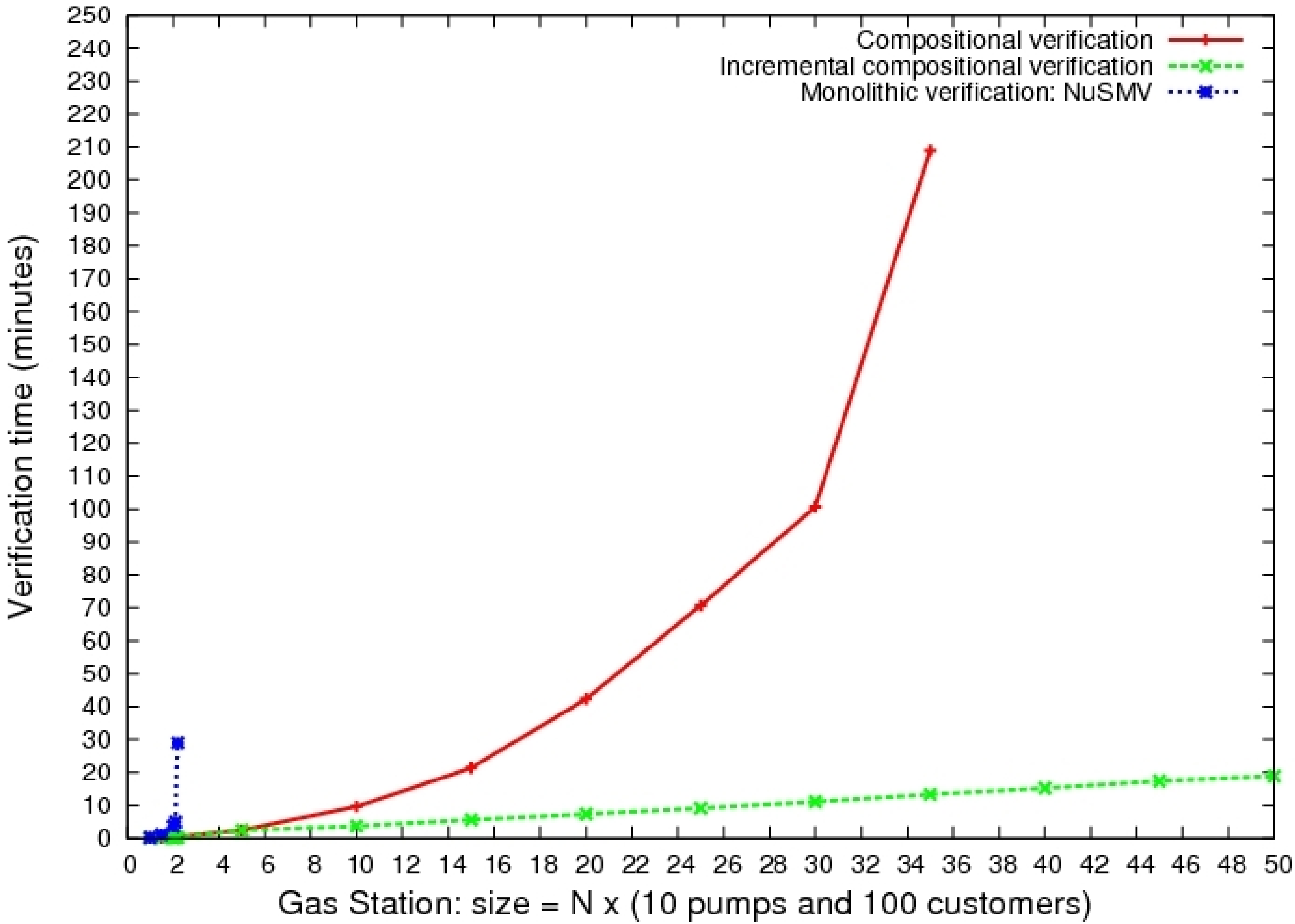
centralized/distributed execution, guided/exhaustive simulation

Compositional Verification – D-Finder

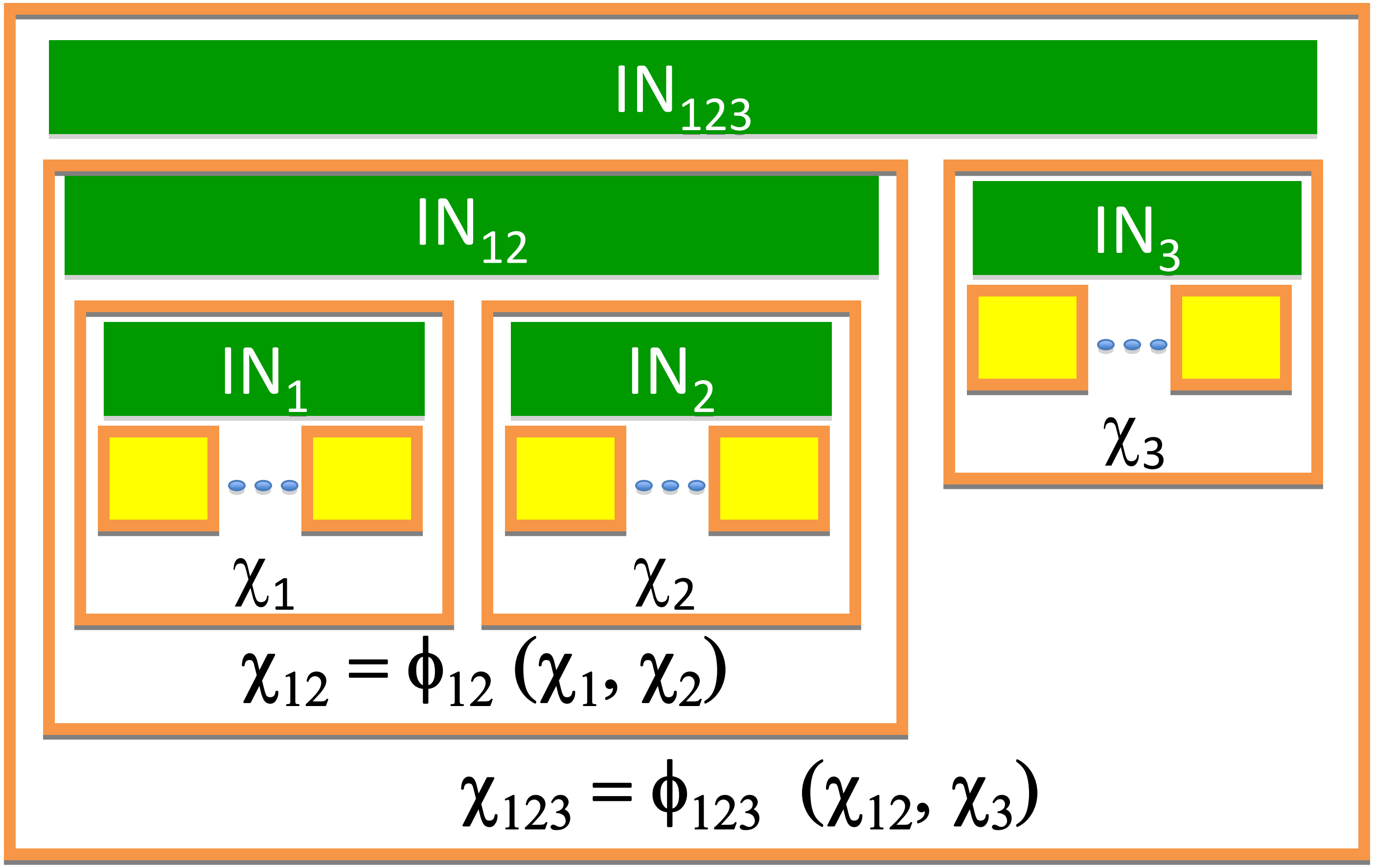
Example	Number of Comp	Number of Ctrl States	Number of Bool Variables	Numb of Int Var	Number Potential Deadlocks	Number Remaining Deadlocks	Verification Time
Temperature Control (2 rods)	3	6	0	3	8	3	3s
Temperature Control (4 rods)	5	10	0	5	32	15	6s
UTOPAR (40 cars,256 CU)	297	795	40	242	--	0	3m46s
UTOPAR (60 cars, 625 CU)	686	1673	60	362	--	0	25m29s
R/W(10000 readers)	10002	20006	0	1	--	0	36m10s
Philosophers (13000)	26000	65000	0	0	--	3	38m48s
Philosophers (10000)	20000	50000	0	0	--	3	29m30s
Smokers (5000)	5001	10007	0	0	--	0	14m
Gas stations (500 pumps, 5000 customers)	5501	21502	0	0	--	0	18m55s

Results obtained by using the D-Finder tool: http://www-verimag.imag.fr/~thnguyen/tool/

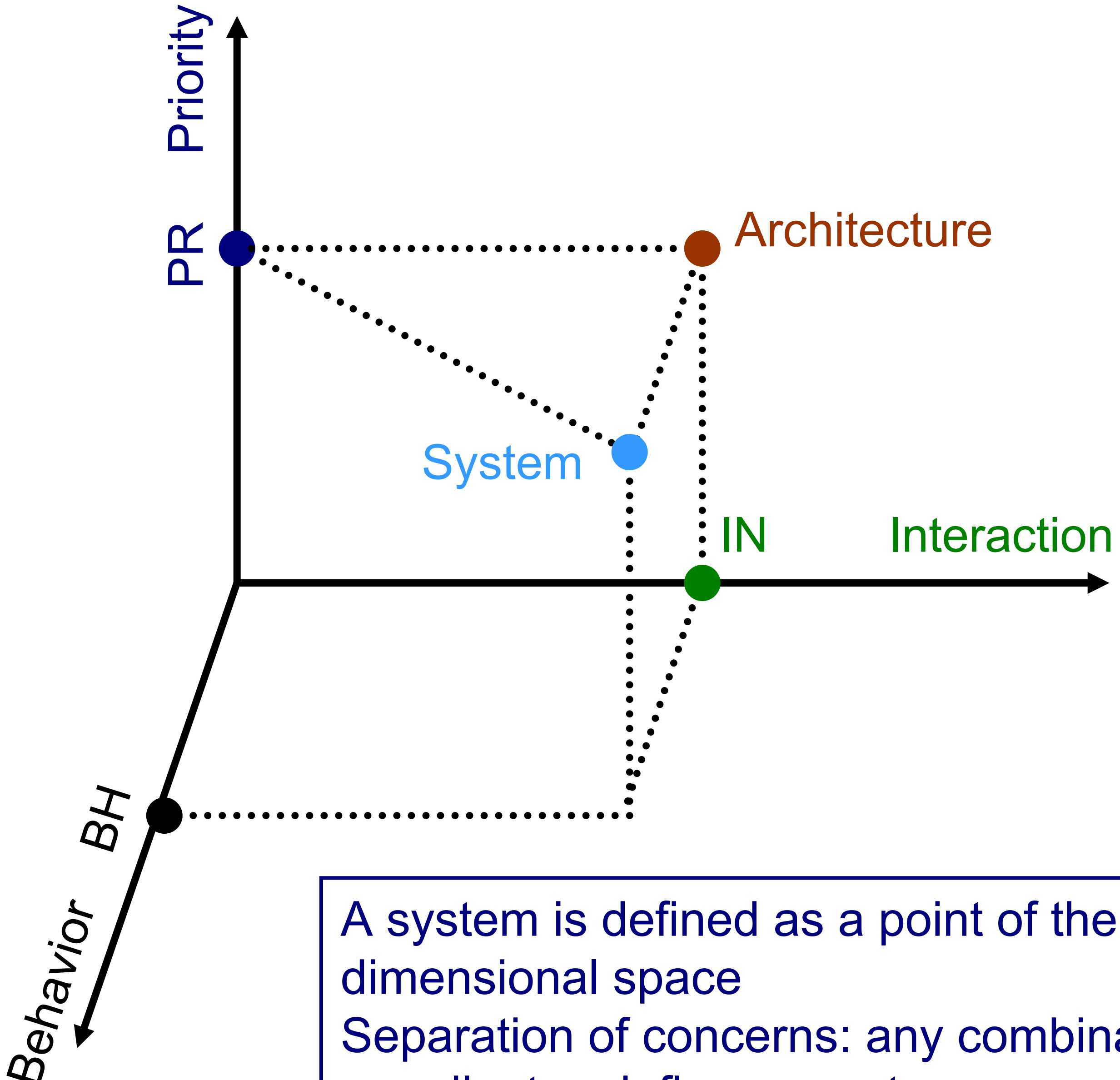
Compositional Verification – D-Finder



Incremental Verification

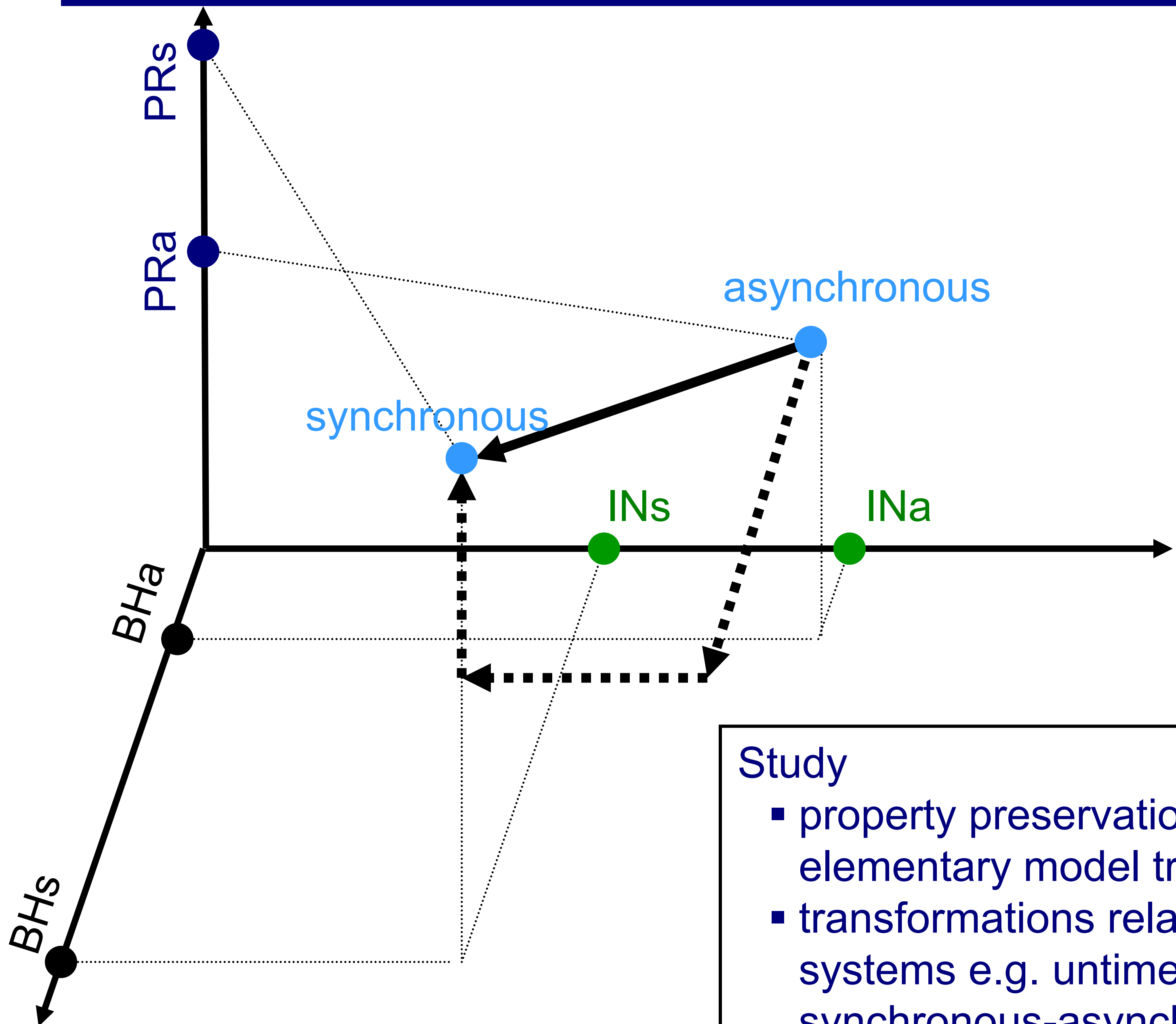


System Construction Space



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

System Construction Space – Incrementality



Study

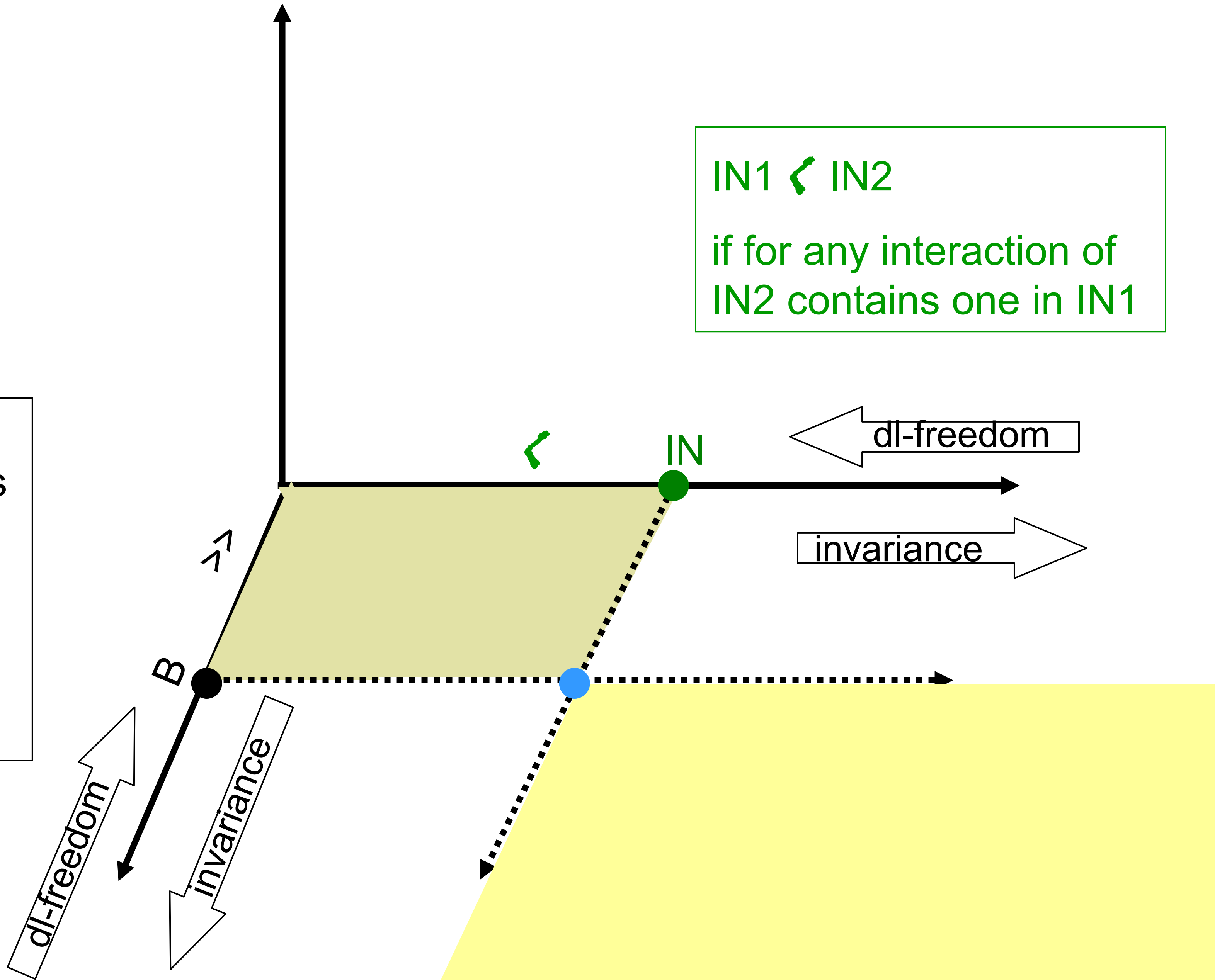
- property preservation results by elementary model transformations
- transformations relating classes of systems e.g. untimed-timed, synchronous-asynchronous

System Construction Space – Incrementality

IN1 $\prec$ IN2

if for any interaction of IN2 contains one in IN1

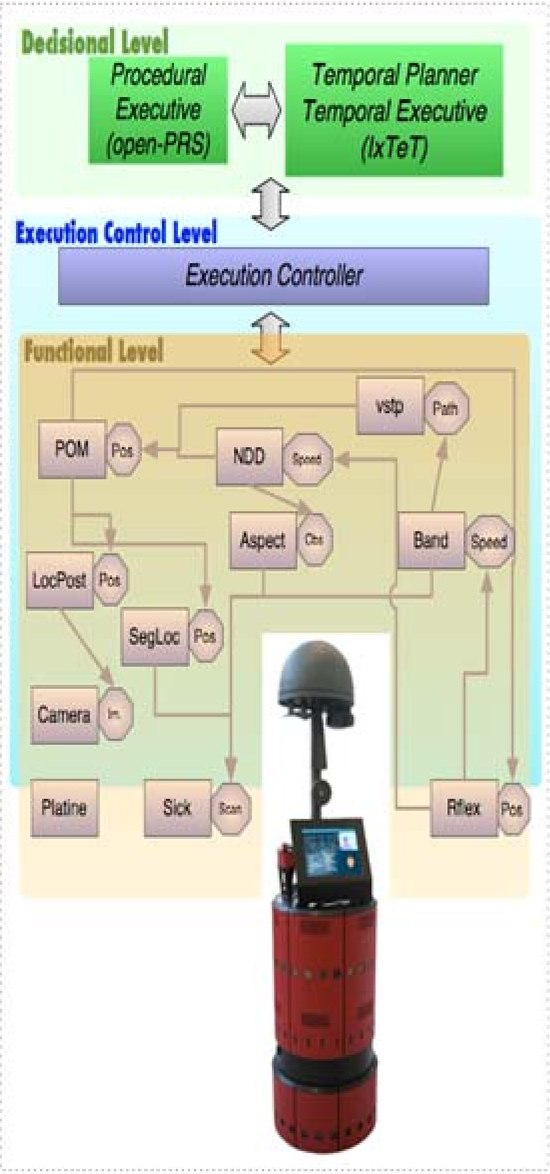
B1<<B2 if B1 simulates B2 and the simulation relation is total

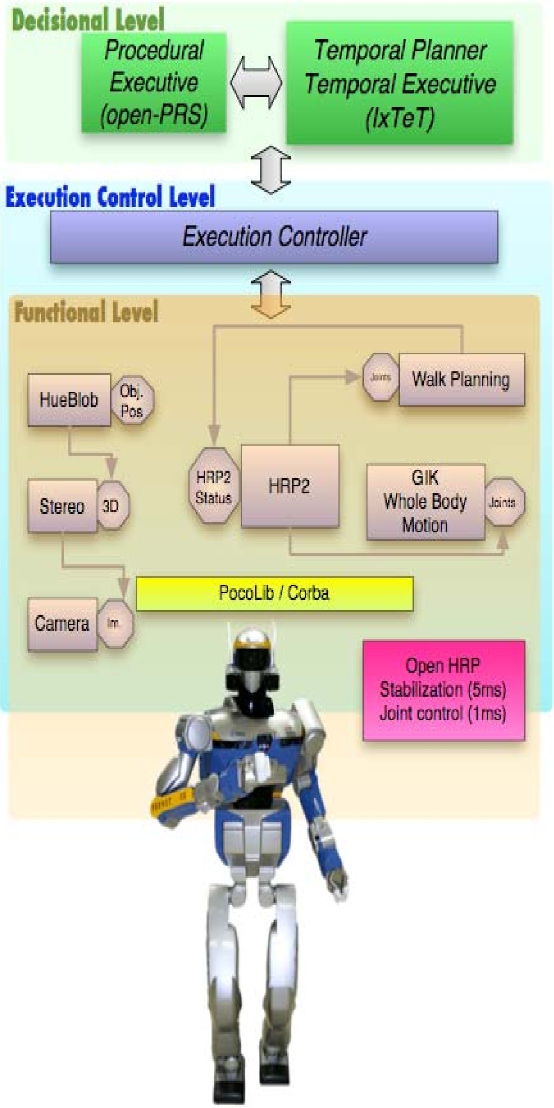


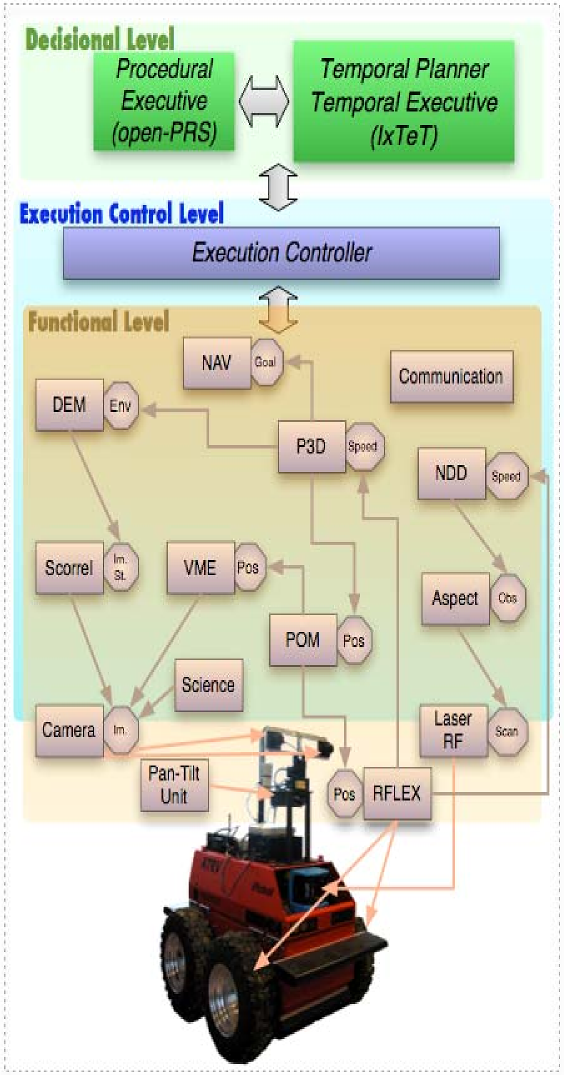
OVERVIEW

- Current status
- Beyond a posteriori verification
  - Component-based Construction
  - The BIP Component Framework
  - Verification at Design Time
- Conclusion

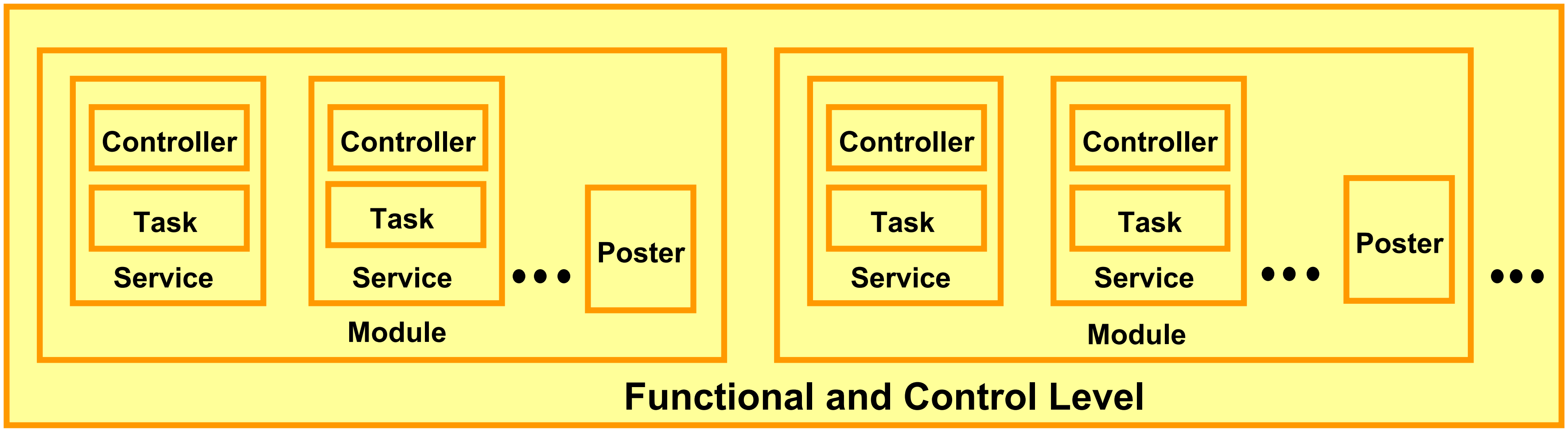
Autonomous Systems







The DALA Robot – Componentization



Functional and Control Level ::= Module$^+$

Module ::= Service$^+$. Poster

Service ::= Service Controller . Service Task

Service Controller ::= Event Triggered Controller | Cyclic Controller

Cyclic Controller ::= Event Triggered Controller . Cyclic Trigger

Service Task ::= Timed Task | Untimed Task

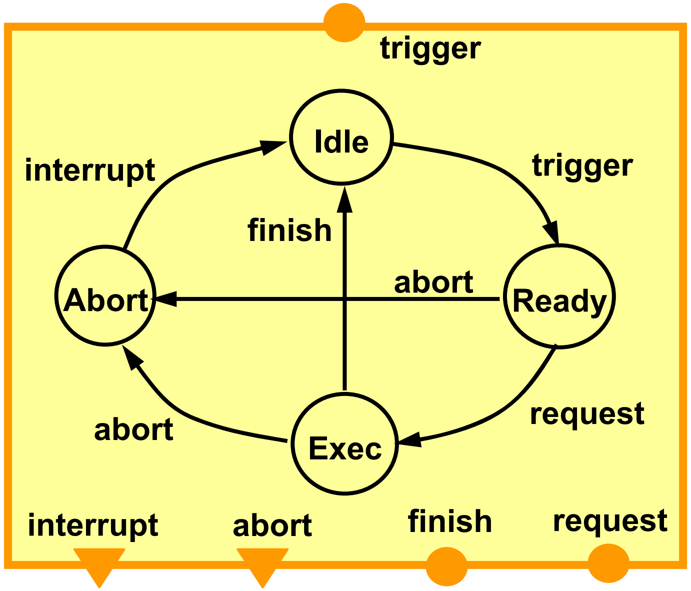
The DALA Robot – Event Triggered Controller

Idle: the Service is idle

Ready: checks the possibility for starting a new Task of the Service

Exec: execution of the Task of the Service

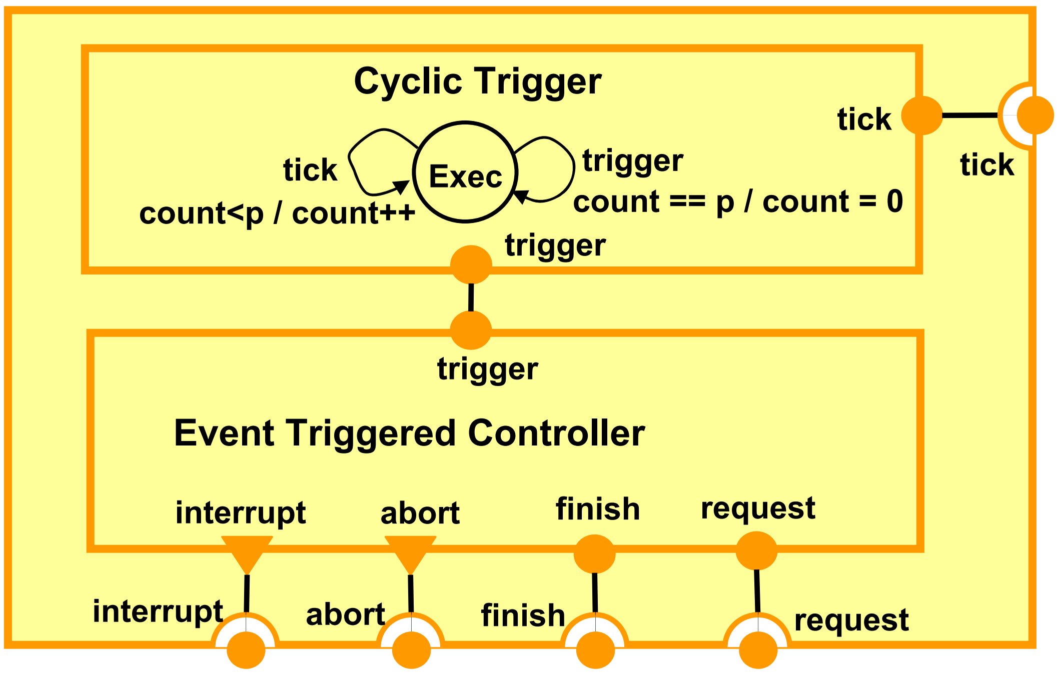
Abort: Service is aborted



The DALA Robot – Cyclic Controller

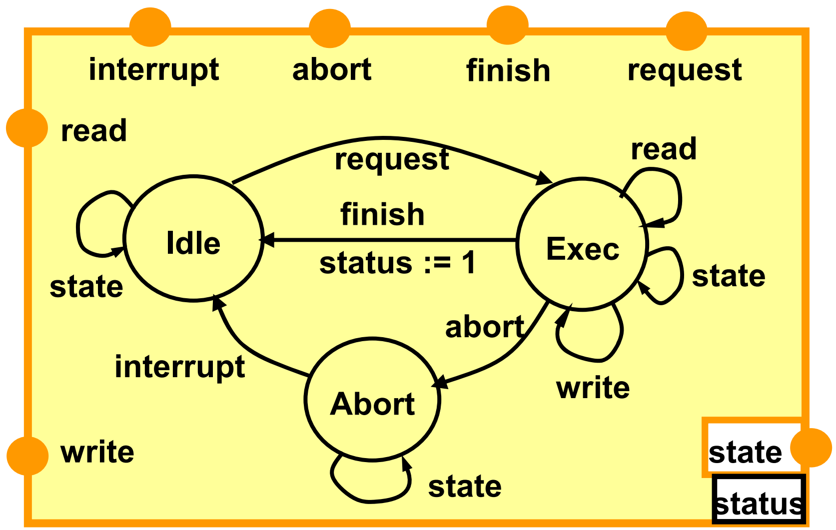
Cyclic Controller ::=
Event Triggered Controller . Cyclic Trigger

The Cyclic Trigger starts the Event Triggered Controller every period p



The DALA Robot – Untimed Task

Triggered by *request*

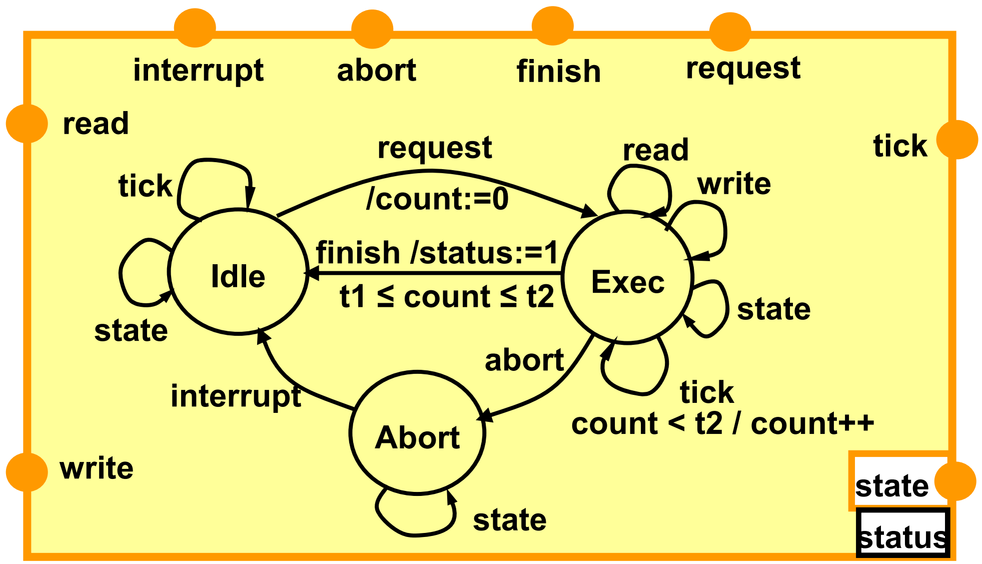


The variable ***status*** specifies the previous state of Task

- ***status == 1*** : Task successfully executed
- ***status == 0*** : Task aborted

The DALA Robot – Timed Task

- Obtained from an Untimed Task
- Its execution time is in [t1,t2]

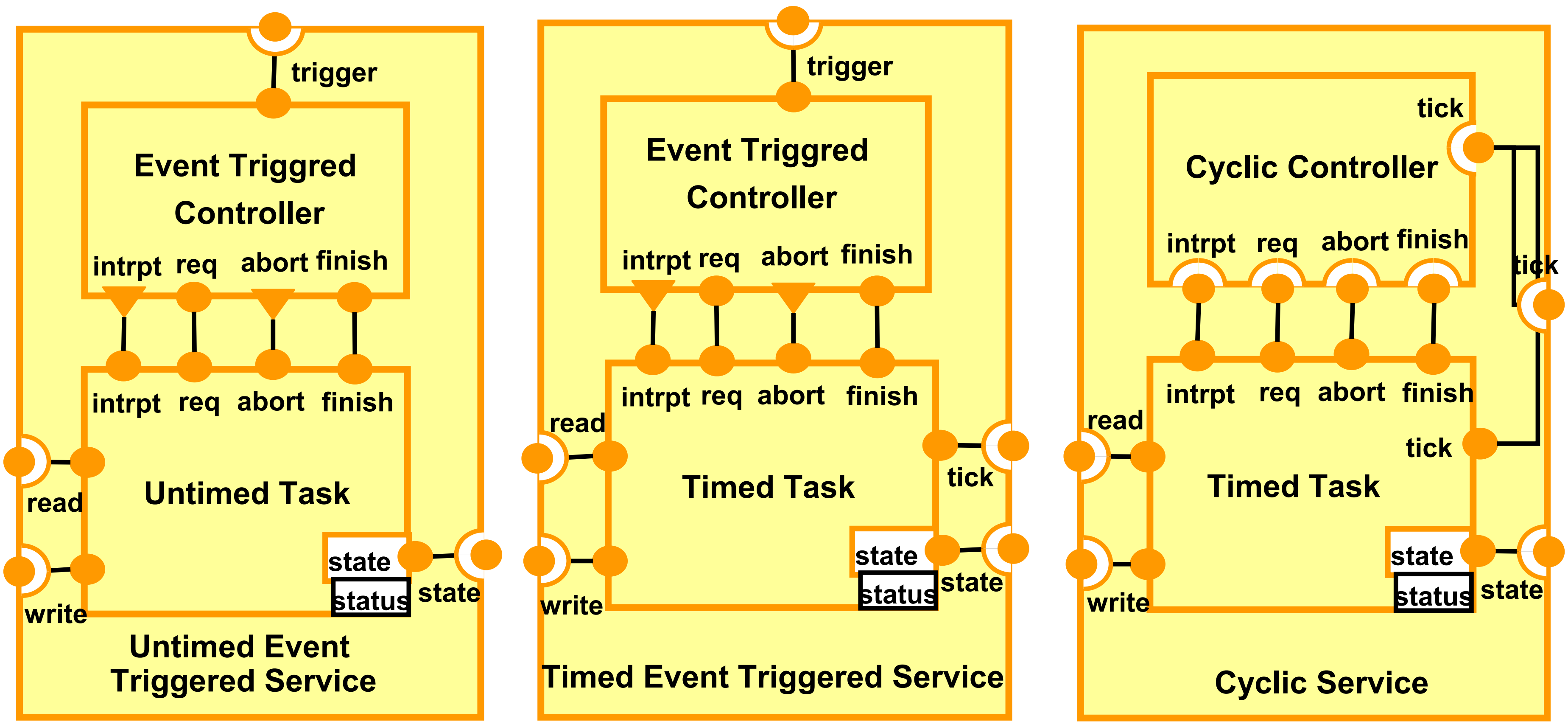


The DALA Robot – Different types of Services

Untimed Event Triggered Service ::= Event Triggered Controller. Untimed Task

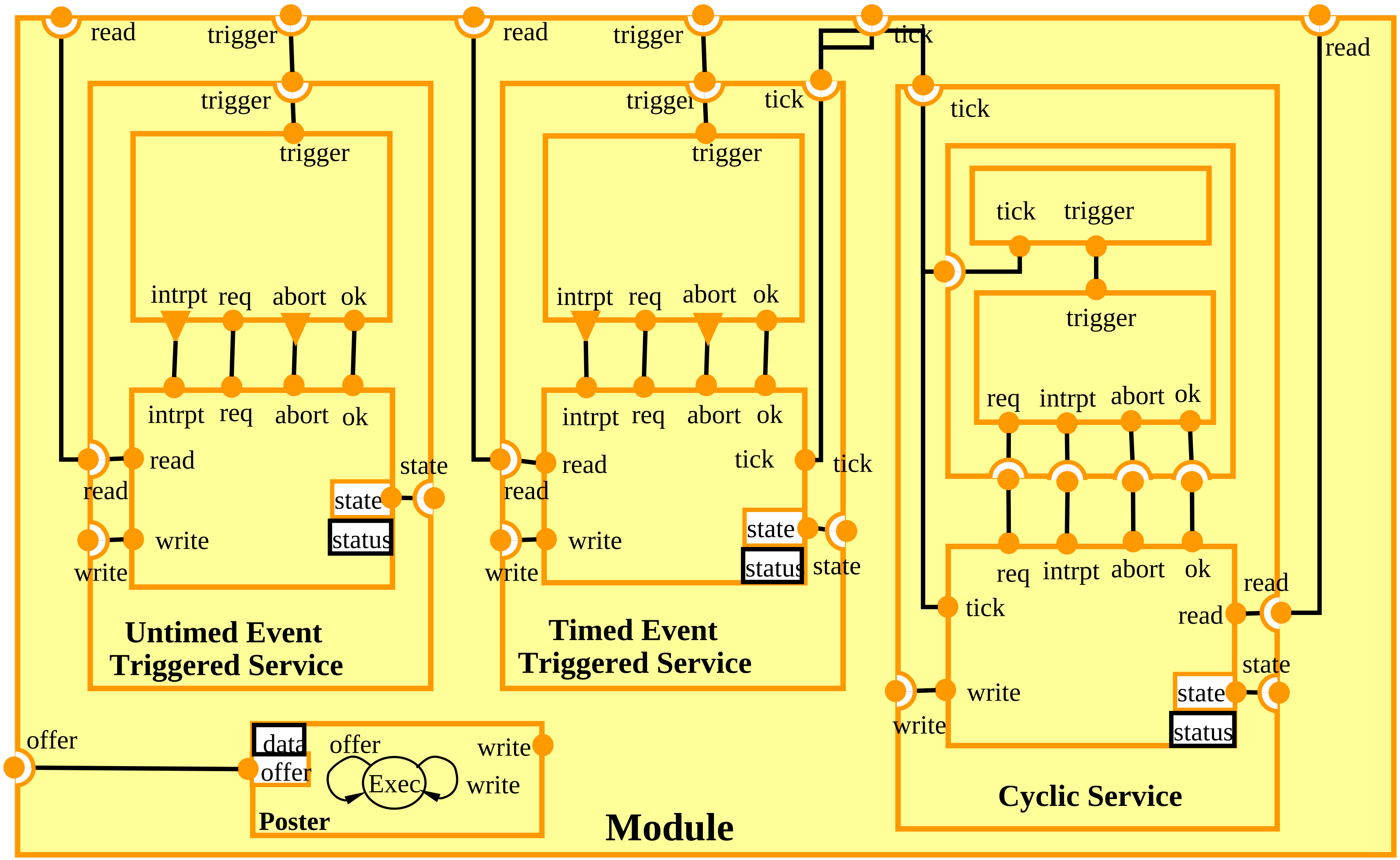
Timed Event Triggered Service ::= Event Triggered Controller. Timed Task

Cyclic Service ::= Cyclic Controller . Timed Task



The DALA Robot – A Module

A module composed of 3 services and a poster



O V E R V I E W

- Current status
- Beyond a posteriori verification
  - Component-based Construction
  - The BIP Component Framework
  - Verification at Design Time
- Conclusion

Conclusion

- Move from a posteriori verification to verification at design time – adequate framework for component-based construction

- Minimalistic approach for verification – focus on state invariance and deadlock-freedom

- Achieve correctness through
  - Constructivity: compositionality/composability techniques
  - Incrementality: reusing proofs for constituents
  - Property-preserving transformations

THANK YOU