Correct-by-construction Distributed Implementations for BIP

WFCD 09

Grenoble, October 11, 2009

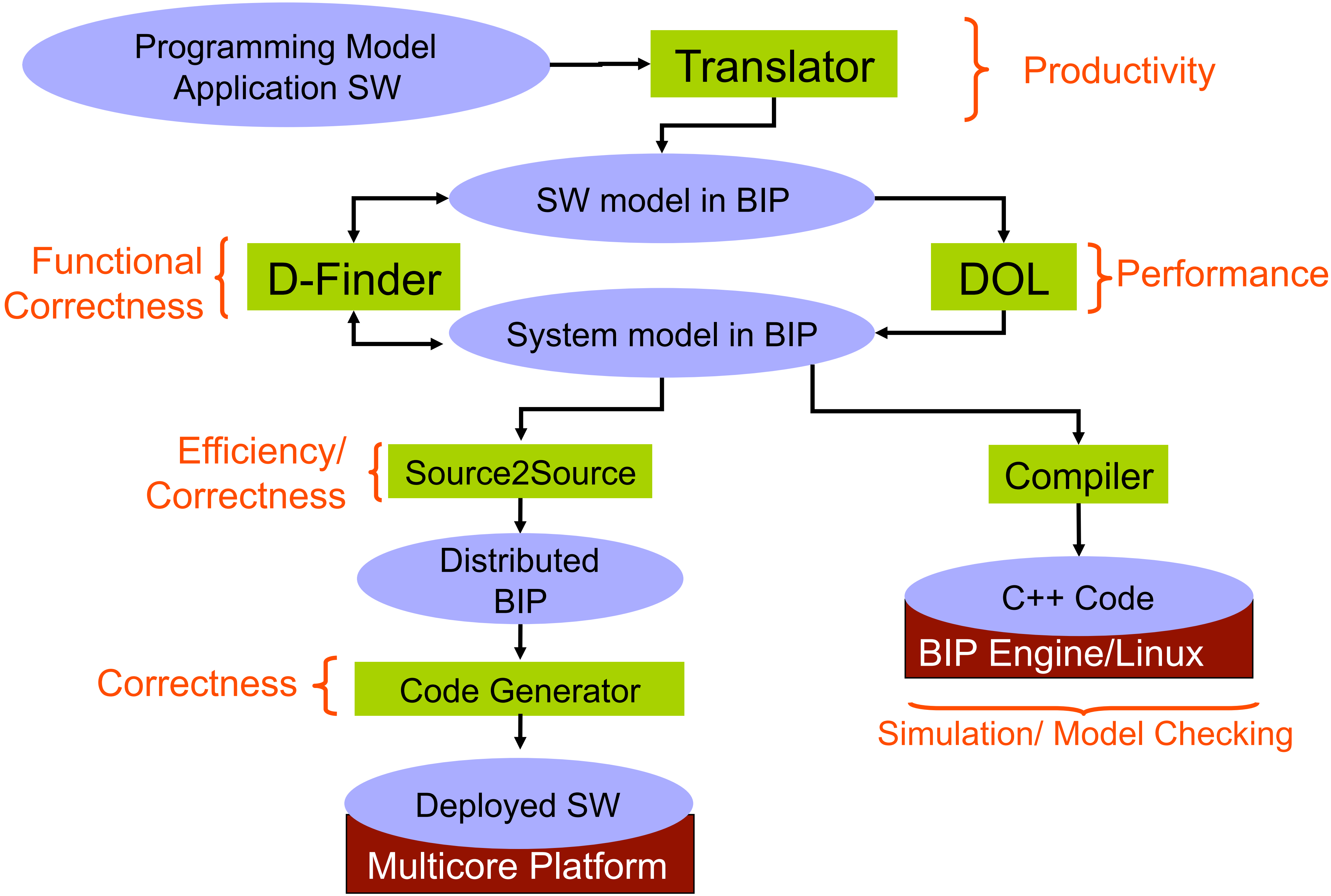
Joseph Sifakis

VERIMAG Laboratory

in collaboration with

A. Basu, S. Bensalem, B. Bonakdarpour, M. Bozga, M. Jaber

System Design Flow



System Design Flow – Methodology

- Use BIP as a unifying semantic model for the various programming models e.g. synchronous, data flow, event driven

- Constructivity results - rules for inferring global properties from properties of constituent components

- Generate from a model of the application software and a model of the target platform, an implementation by using a set of correct-by-construction model transformations
  - preserving functional properties
  - taking into account extra-functional requirements

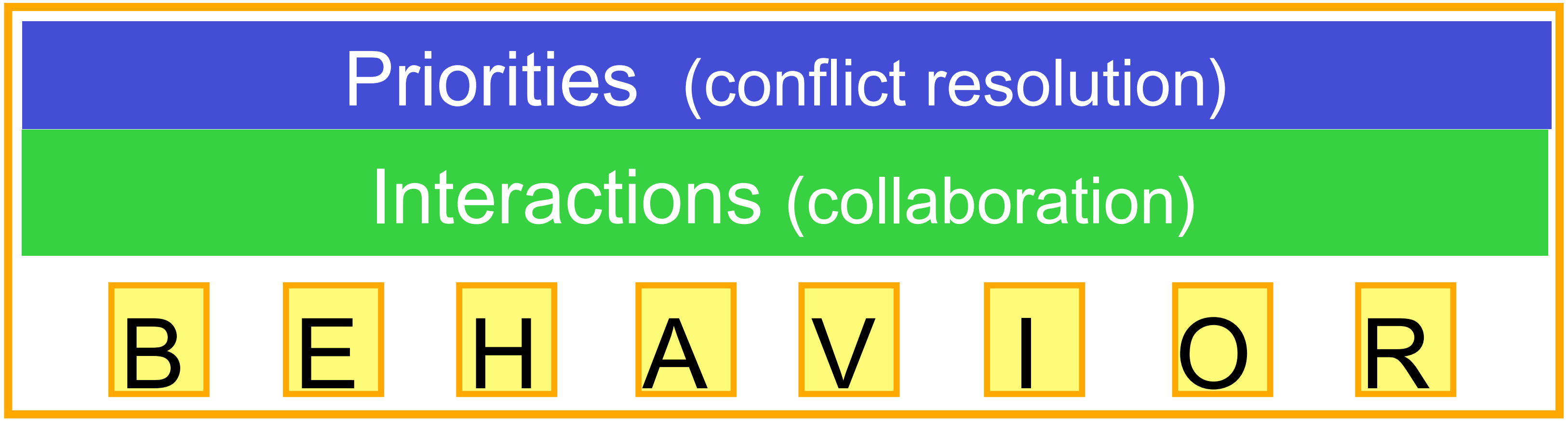
  Types of transformations studied:
  - From BIP to distributed BIP (BIP using asynchronous message passing only)
  - Architecture transformations
  - From BIP (message passing) models to shared memory models e.g. shared/private memory transformations
  - Integrating architecture constraints, e.g. HW/SW partitioning and mapping

OVERVIEW

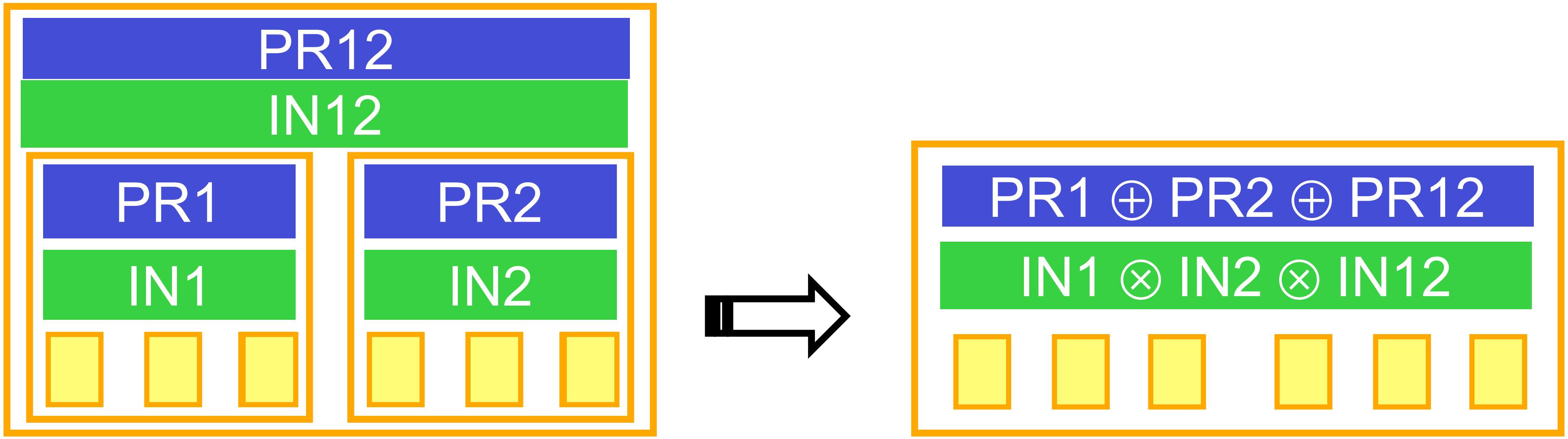
- BIP – Global state model
- Distributed centralized implementation
- Decentralized implementations
- Architecture transformations
- Discussion

BIP – Basic Concepts

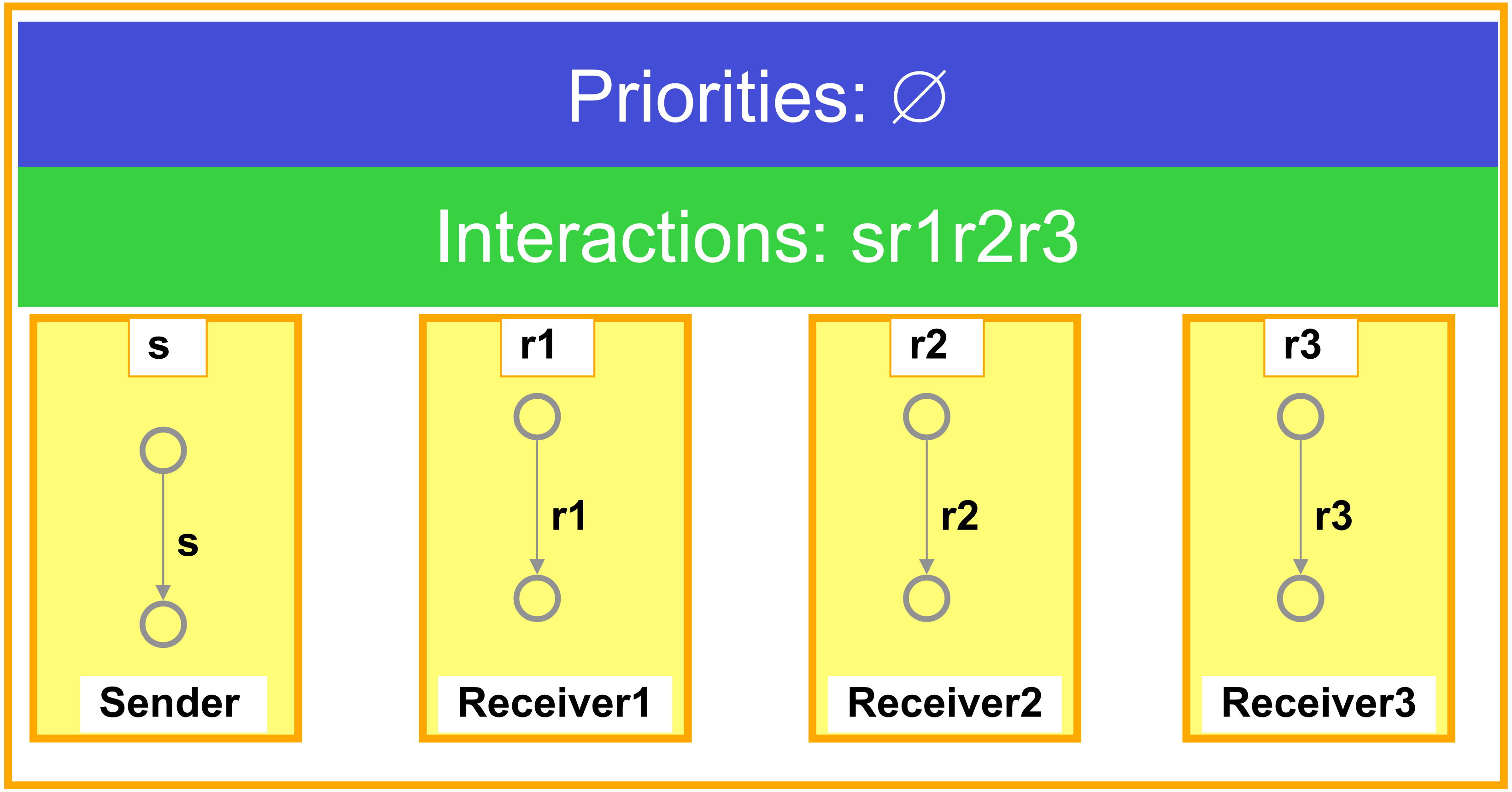
Layered component model



Composition operation parameterized by glue IN12, PR12

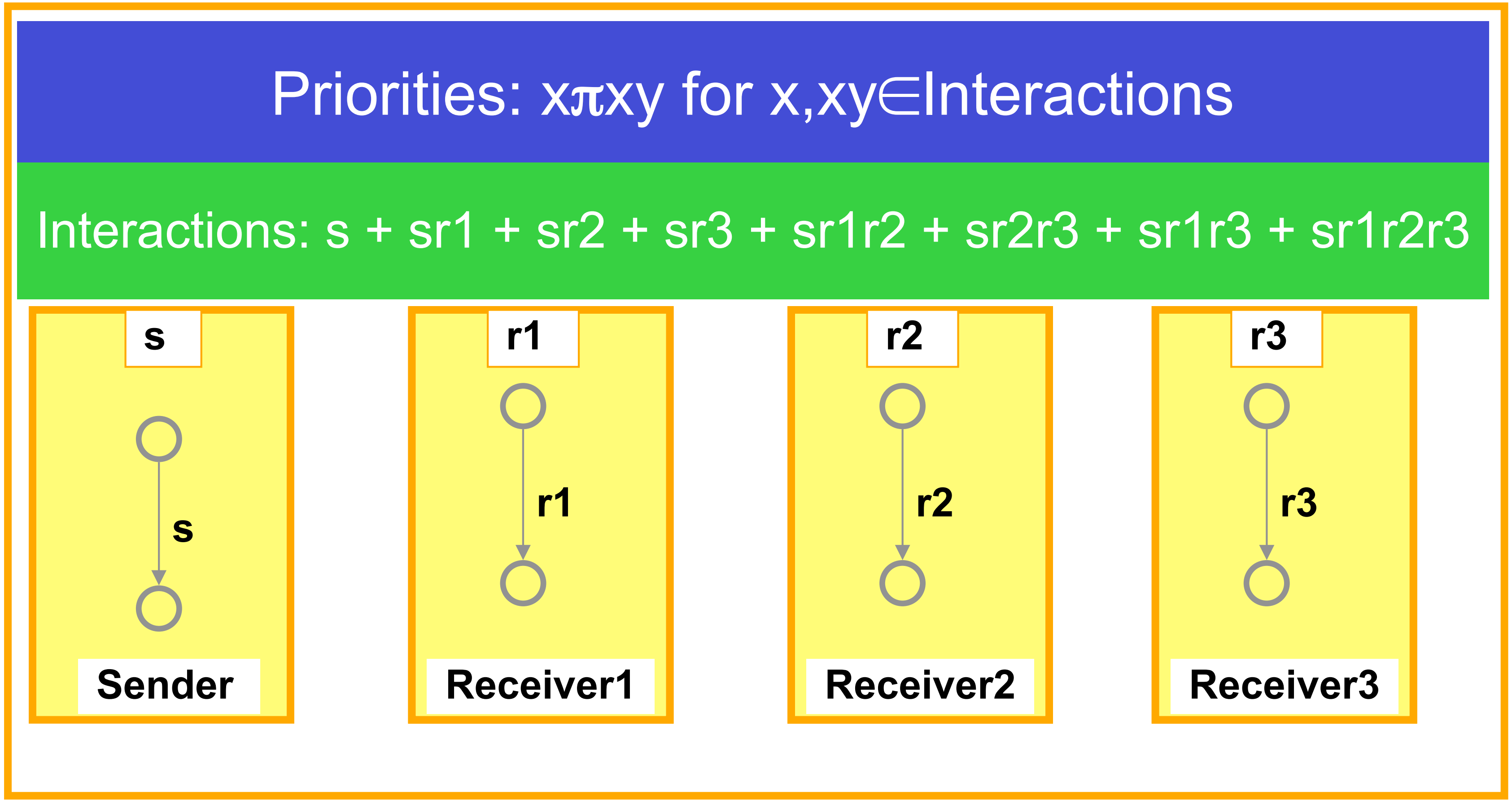


BIP – Basic Concepts



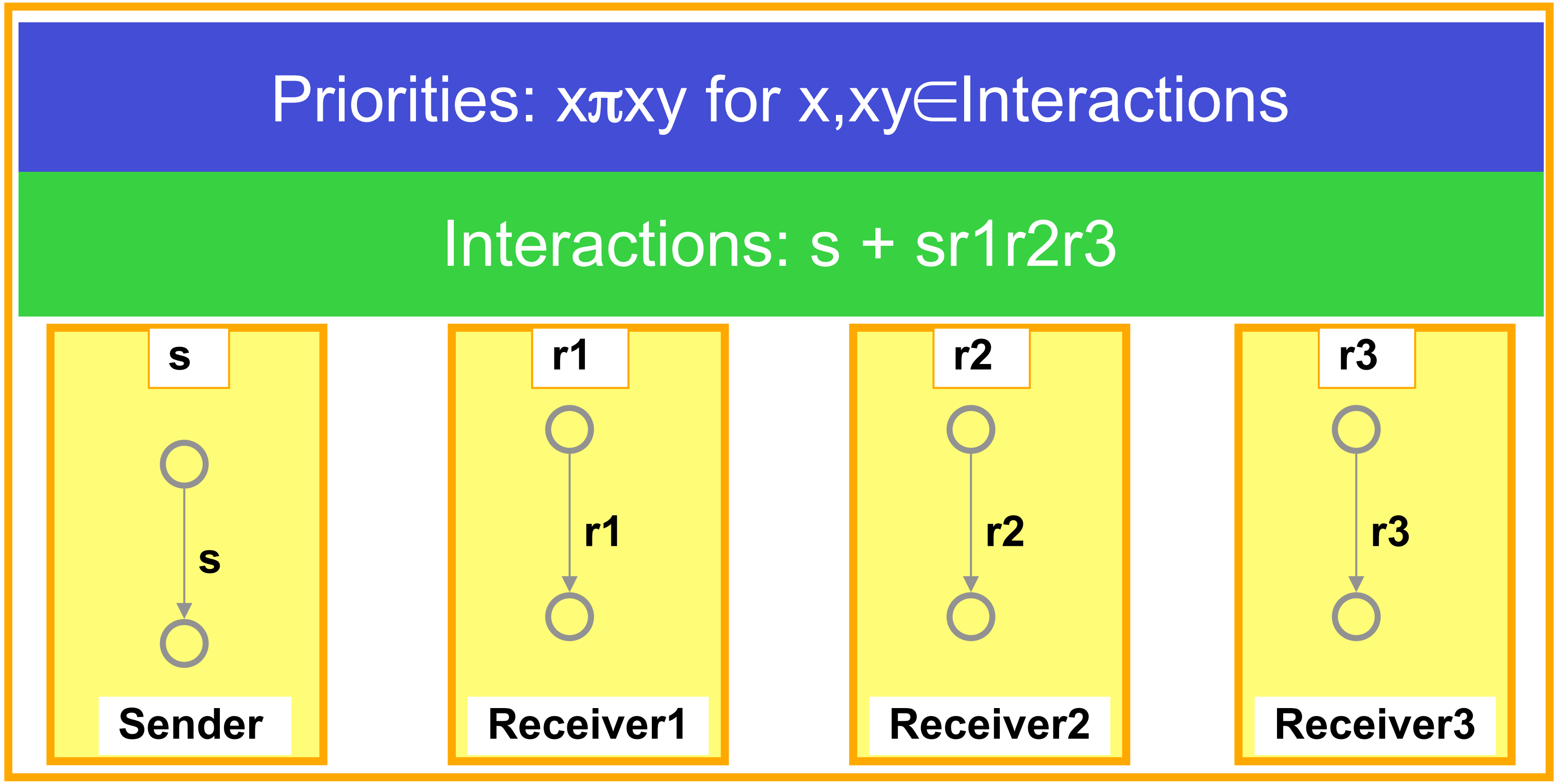
Rendezvous

BIP – Basic Concepts

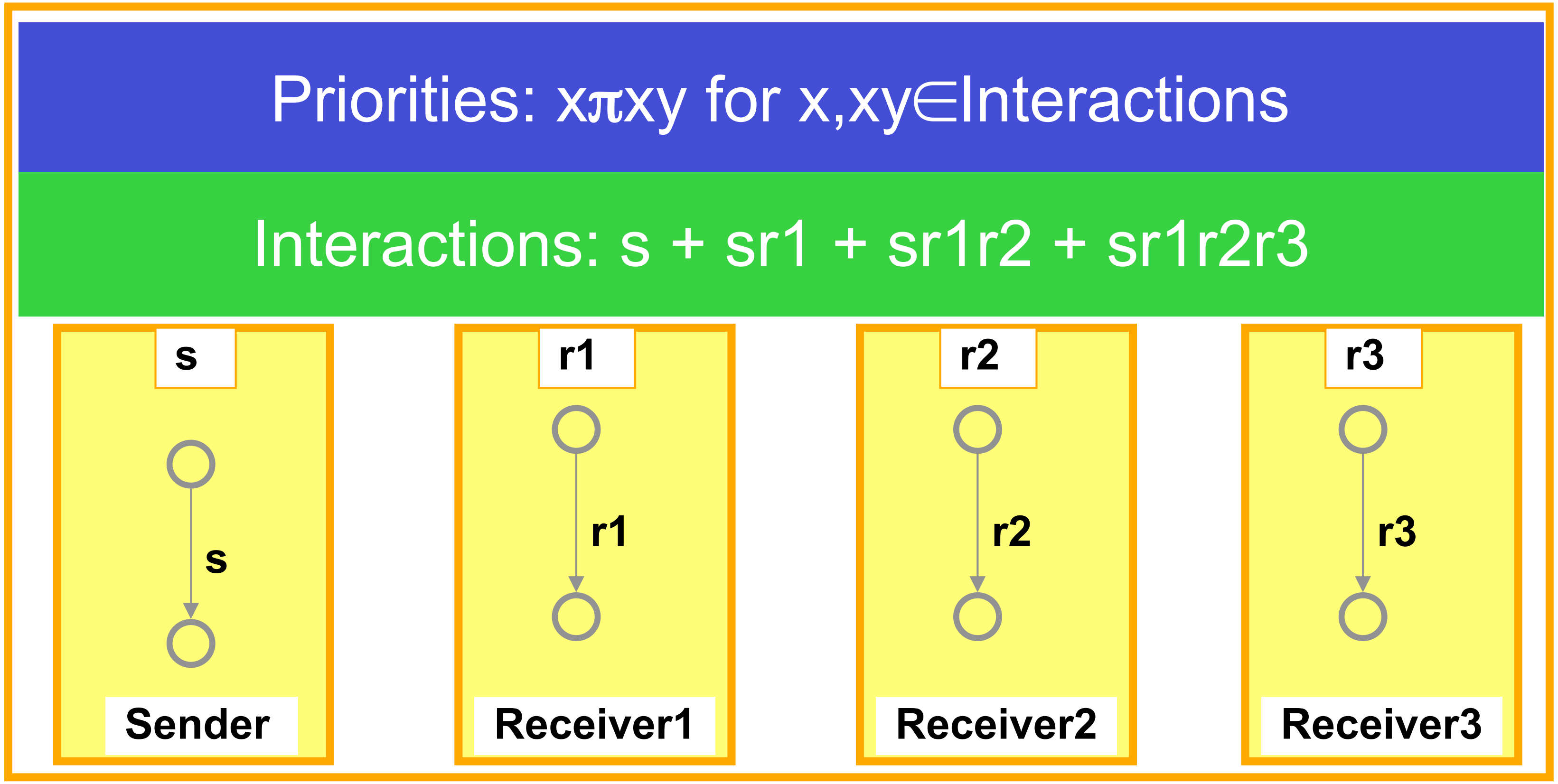


Broadcast

BIP – Basic Concepts



Atomic Broadcast



Causal Chain

BIP – Basic Concepts: Semantics

$$\left.\begin{array}{l}\text{▪ a set of atomic components } \{B_i\}_{i=1..n} \\ \quad \text{where } B_i = (Q_i,\ 2^{P_i},\ \rightarrow_i) \\ \text{▪ a set of interactions } \gamma \\ \text{▪ priorities } \pi \subseteq \gamma \times (\otimes Q_i) \times \gamma\end{array}\right\} \pi\,\gamma\,(B_1,.., B_n)$$

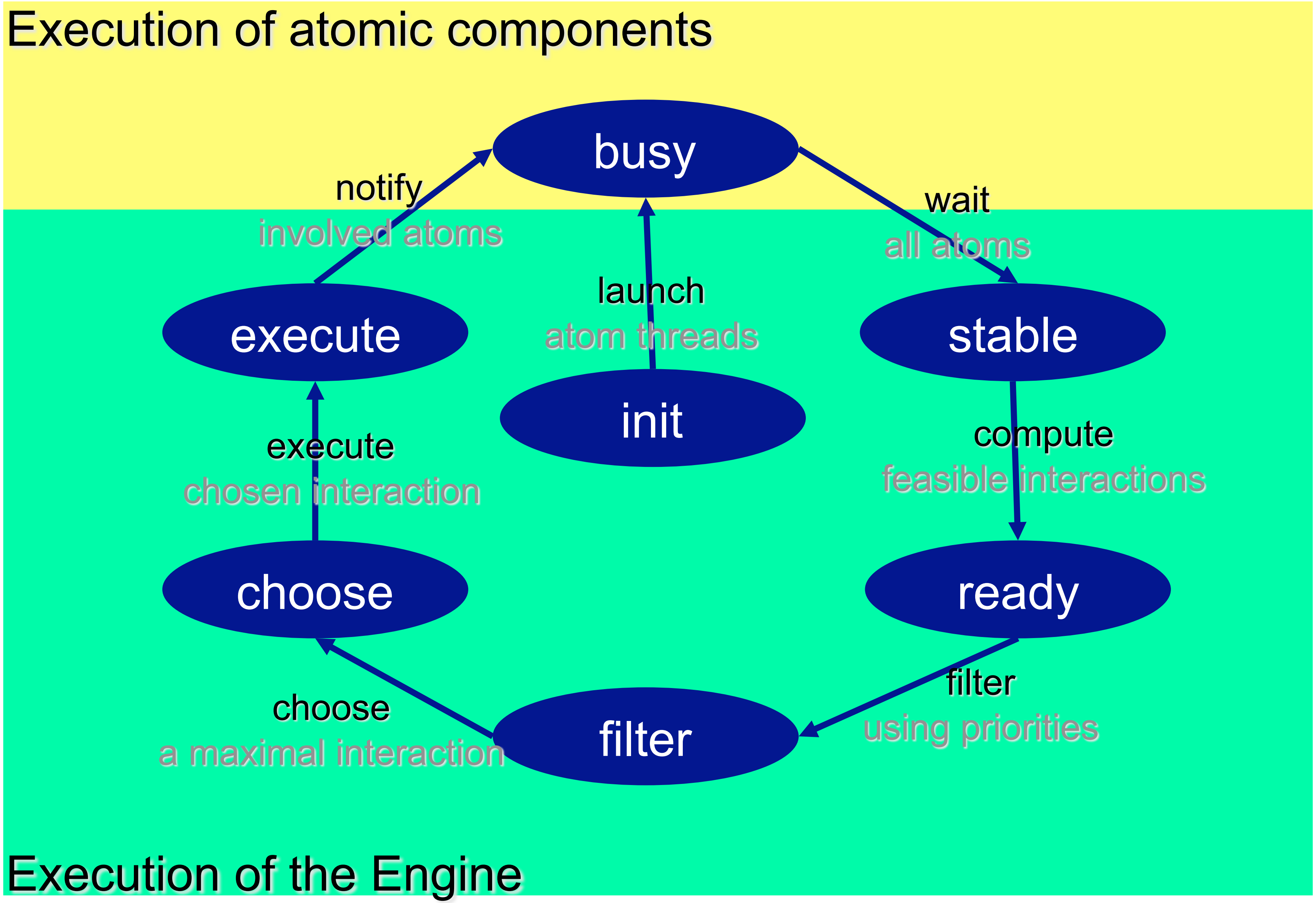
Interactions

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

Priorities

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

BIP – The execution Engine



OVERVIEW

- ❑ BIP – Global state model
- ❑ Distributed centralized implementation
- ❑ Decentralized implementations
- ❑ Architecture transformations
- ❑ Discussion

Distributed Implementation

BIP is based on:

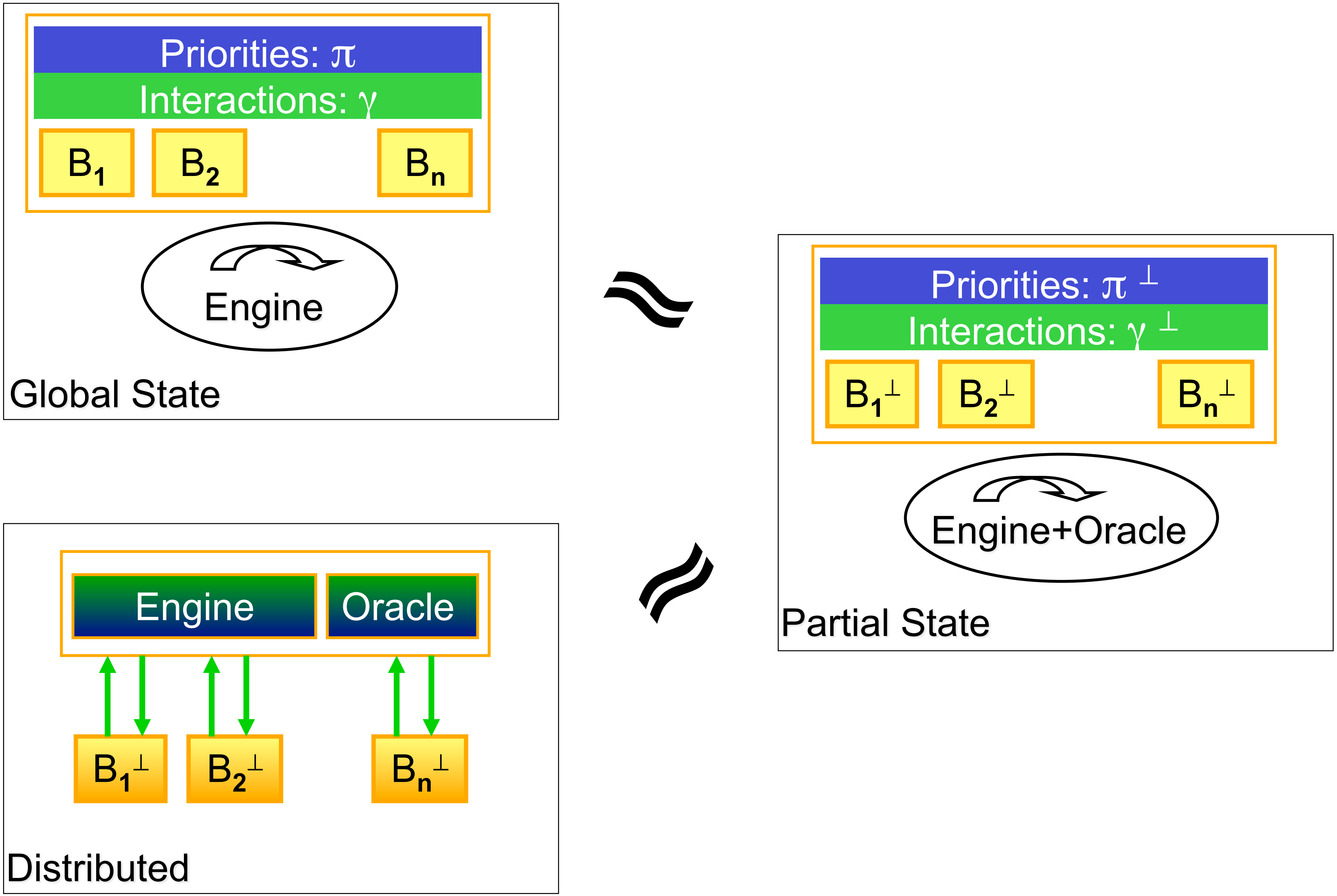
- ❑ Global state semantics, defined by operational semantics rules, implemented by the Engine
- ❑ Atomic multiparty interactions, e.g. by rendezvous or broadcast

Translate BIP models into distributed models

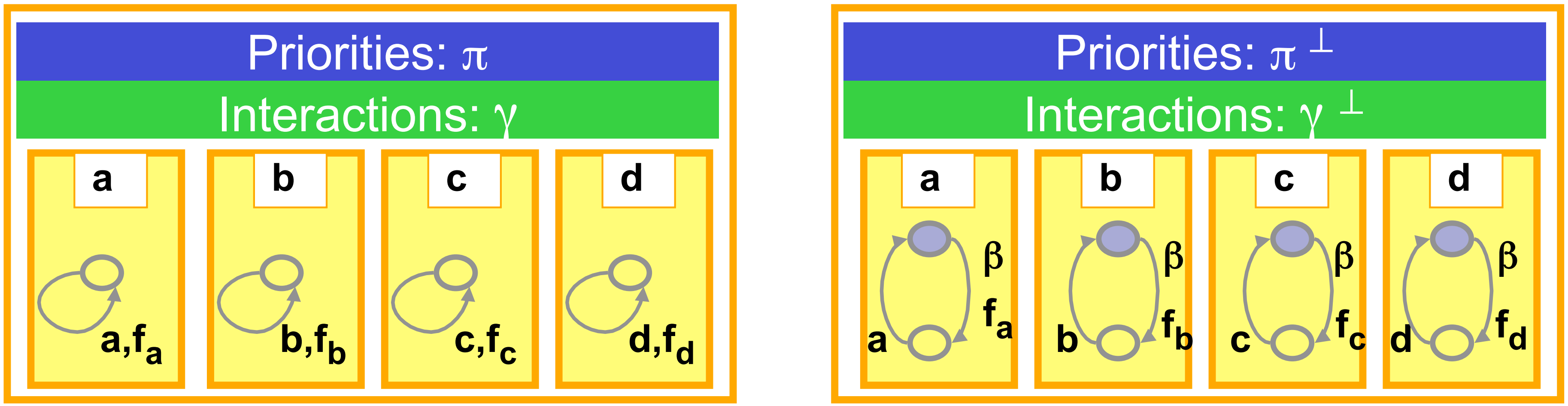
- ❑ Collection of independent components intrinsically concurrent - No global state
- ❑ Separate interaction from internal computation of components
- ❑ Point to point communication by asynchronous message passing
- ❑ Correctness by construction that is, the initial BIP model is observationally equivalent to the implementation

Approach: BIP $\Rightarrow$ Partial state BIP $\Rightarrow$ Distributed BIP

Centralized Distributed Implementation – The Principle



Distributed Implementation – Global vs. Partial State Models



(a) Global State Model

(b) Partial State Model

Broadcast γ = a+ab+ac+ad+abc+abd+acd+abcd, with maximal progress.

(a): only abcd is possible.

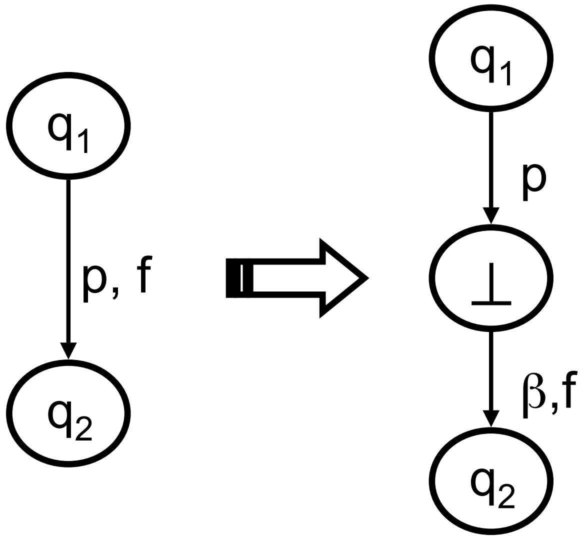
(b): arbitrary desynchronization may occur.

Rendezvous γ = ab+bc+cd and priority abπbc, cdπbc .

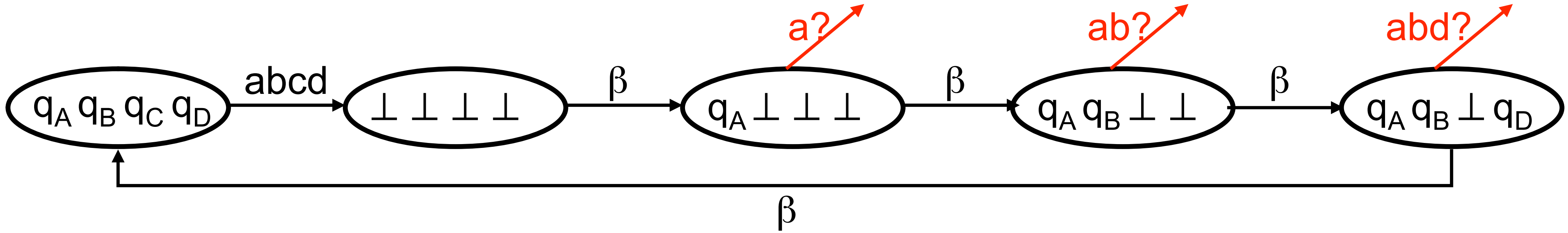
(a): only $(bc)^\omega$ is possible

(b): it is possible to reach a state from which bc never occurs e.g. ab(f_a cd f_b f_d ab f_c)$^\omega$.

Distributed Implementation – Partial state semantics



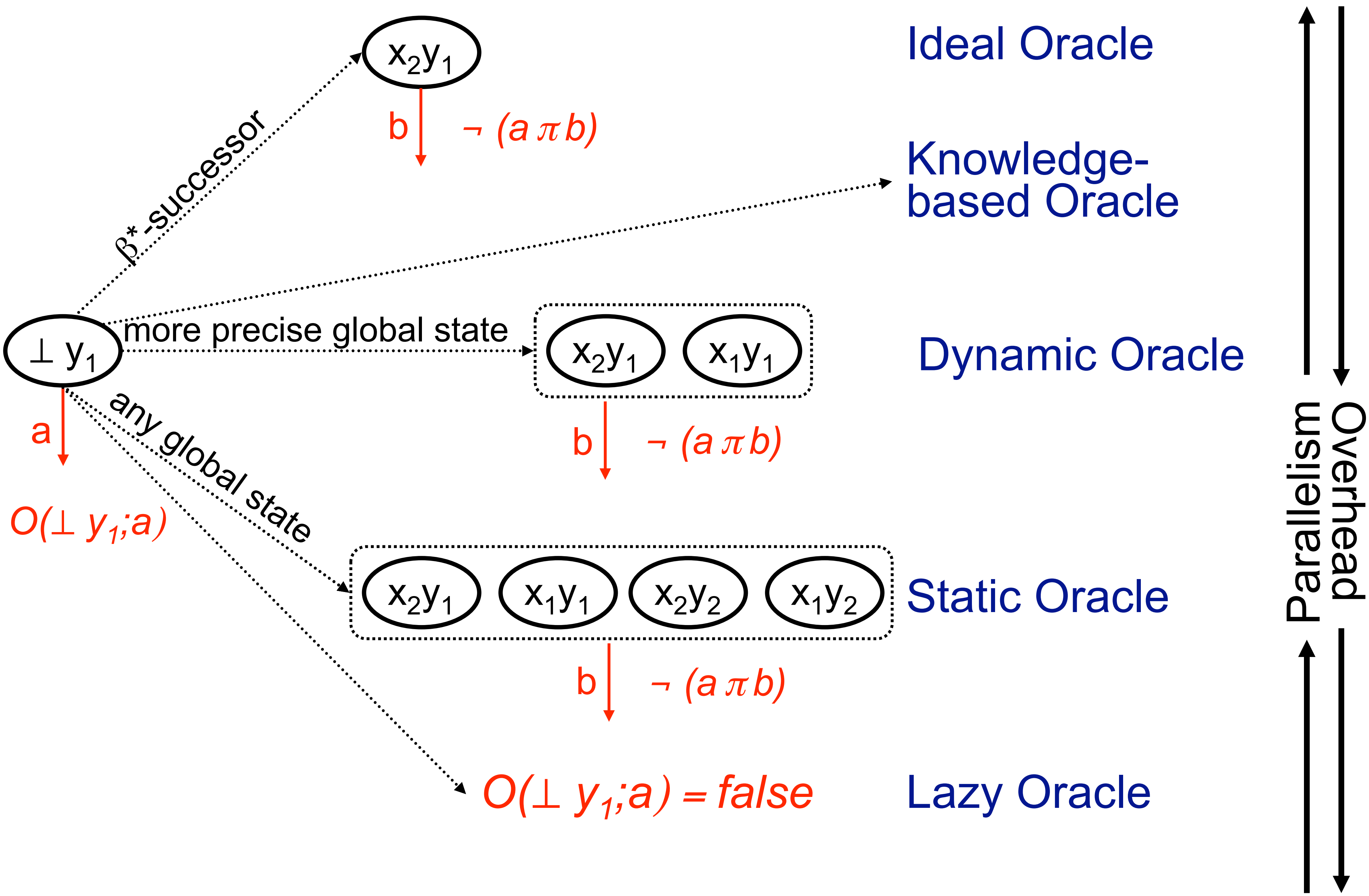
- States are global or partial
- β-transitions interleave
- From any state q a unique global state is reached by application of β-transitions



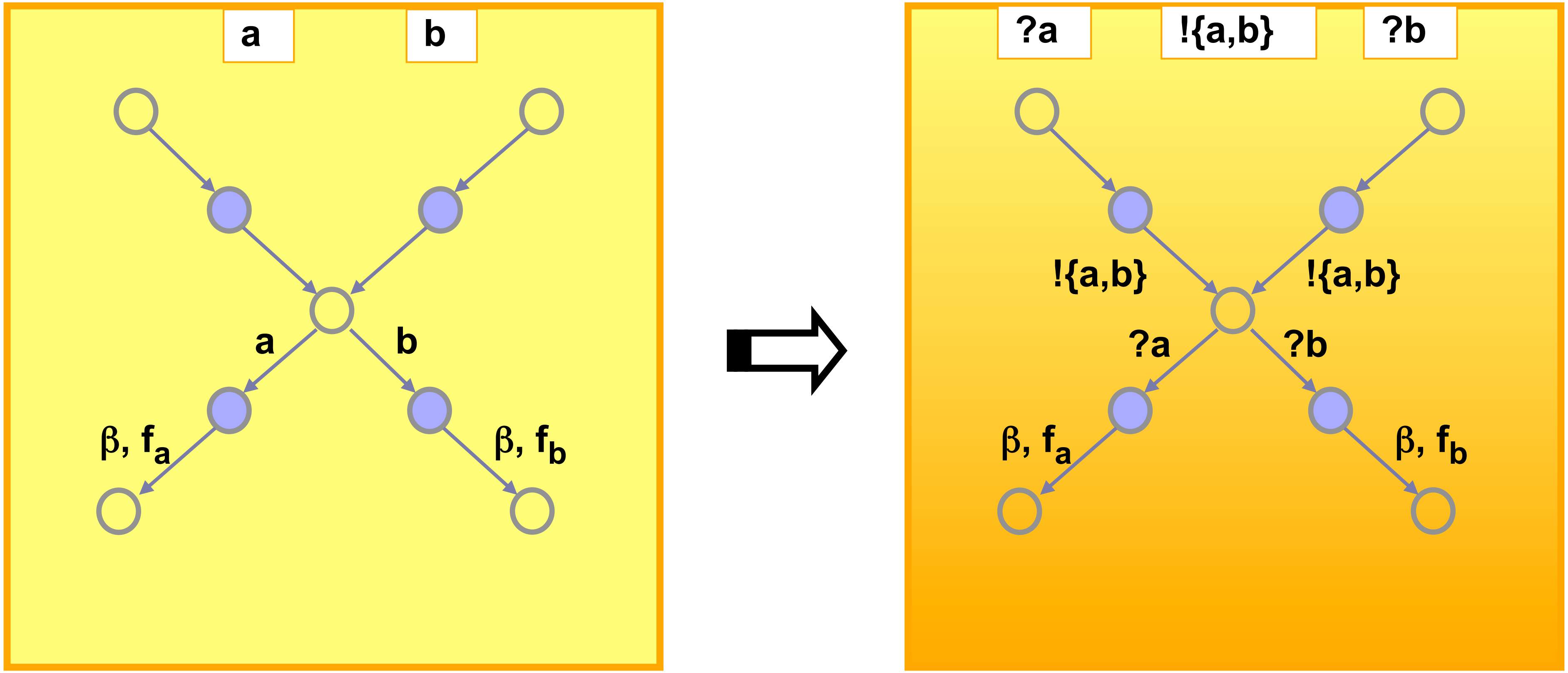
Objective: Safe and efficient execution from partial states

$$\frac{a\in\gamma \qquad \forall i\in[1,n]\ \ q_i \xrightarrow{a\cap P_i}_i q'_i \qquad O(q_1,..,q_n;a)}{(q_1,..,q_n) \xrightarrow{a}_\gamma (q'_1,..,q'_n)\ \ \text{where } q'_i=q_i \text{ if } a\cap P_i=\emptyset}$$

Distributed Implementation – Oracles



Distributed Implementation – Asynchronous MP Model

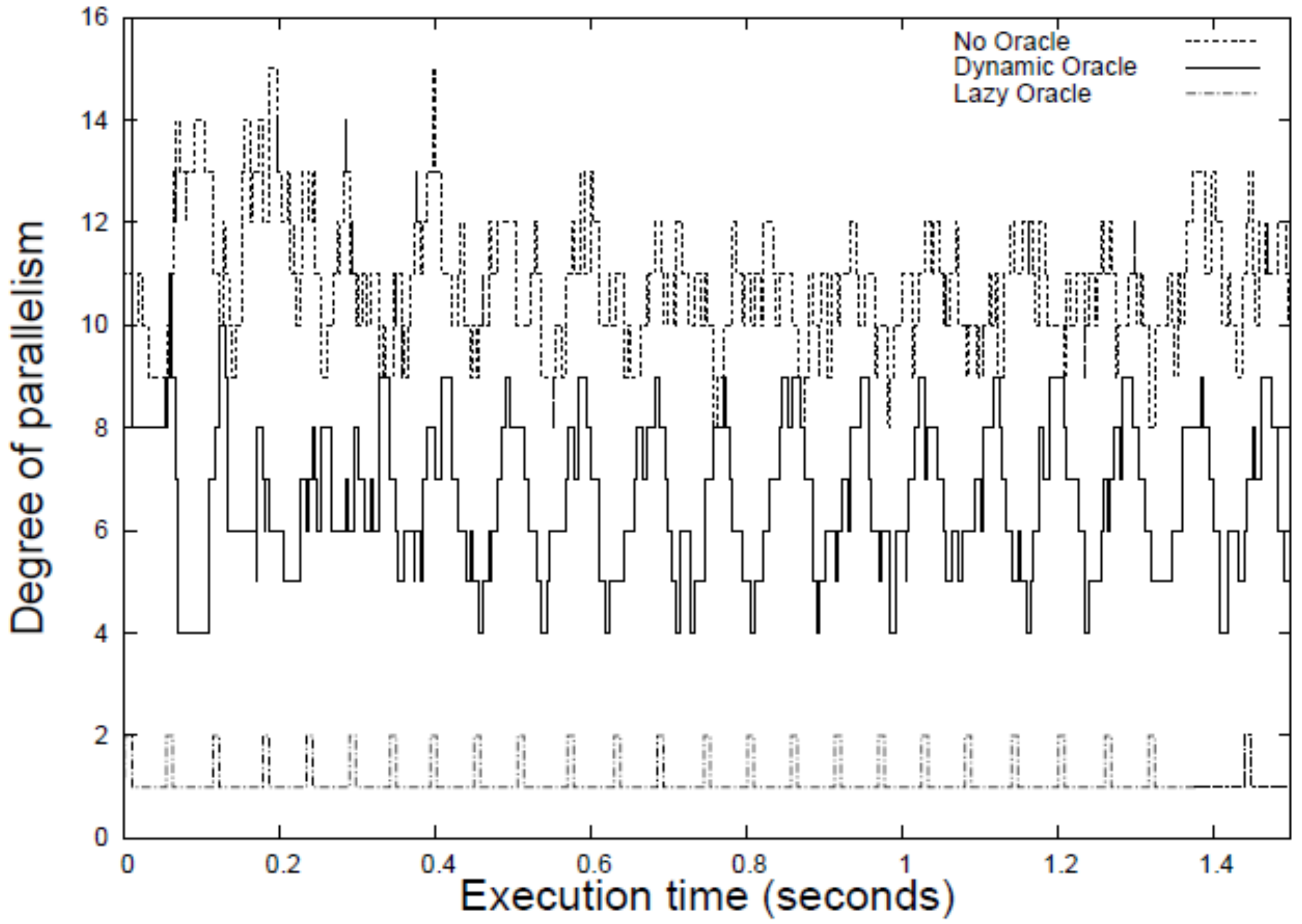


Partial State Model

Message Passing Model

- Before reaching a ready state, the set of the enabled ports is sent to the Engine
- From a ready state, await notification from the Engine indicating the selected port

Distributed Implementation – Example (Forte 08)

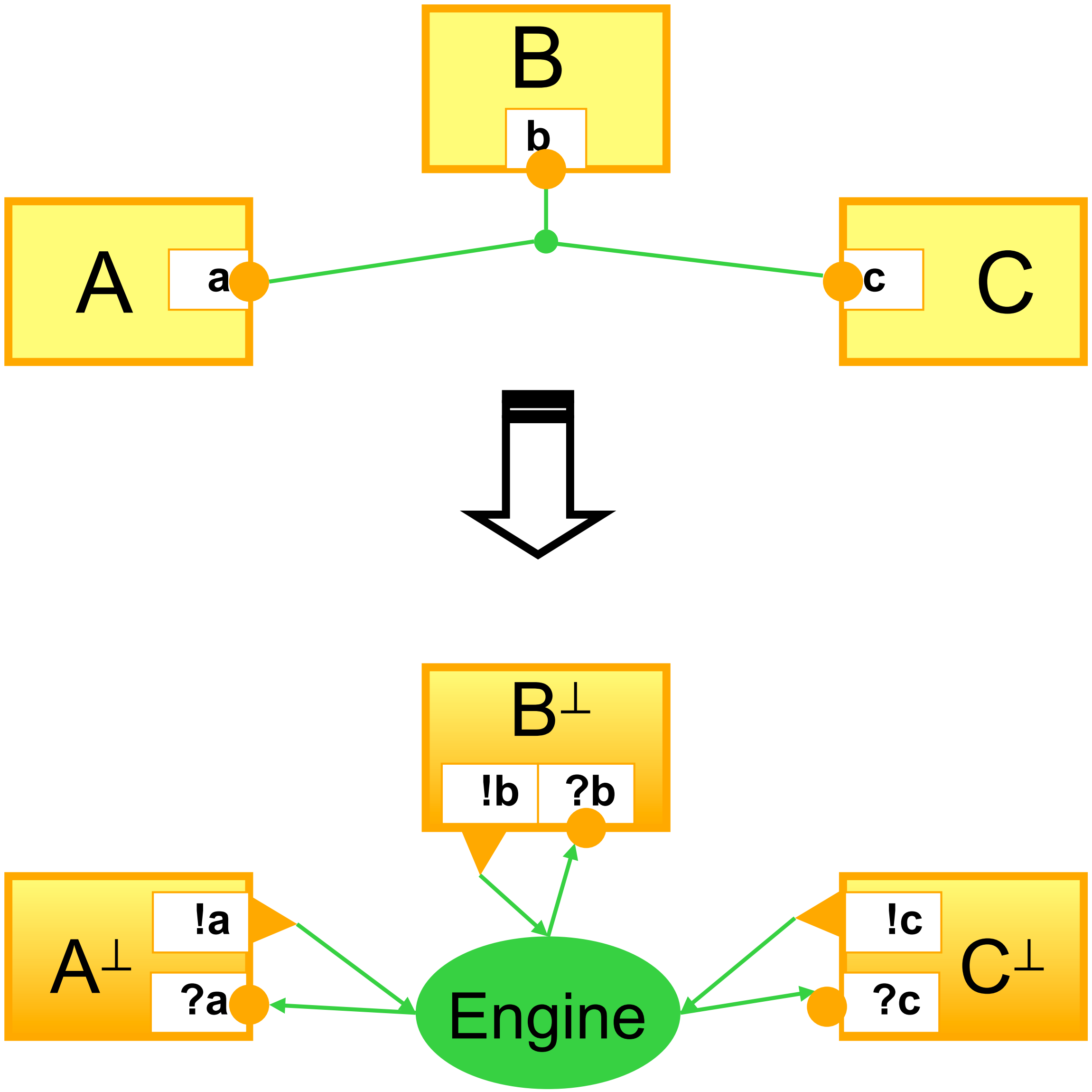


- No oracle: average parallelism 10, but incorrect functional behavior.
- Lazy oracle: maximal parallelism 2.
- Dynamic oracle: average parallelism 7.

O
V
E
R
V
I
E
W

- BIP – Global state model
- Distributed centralized implementation
- Decentralized implementations
- Architecture transformations
- Discussion

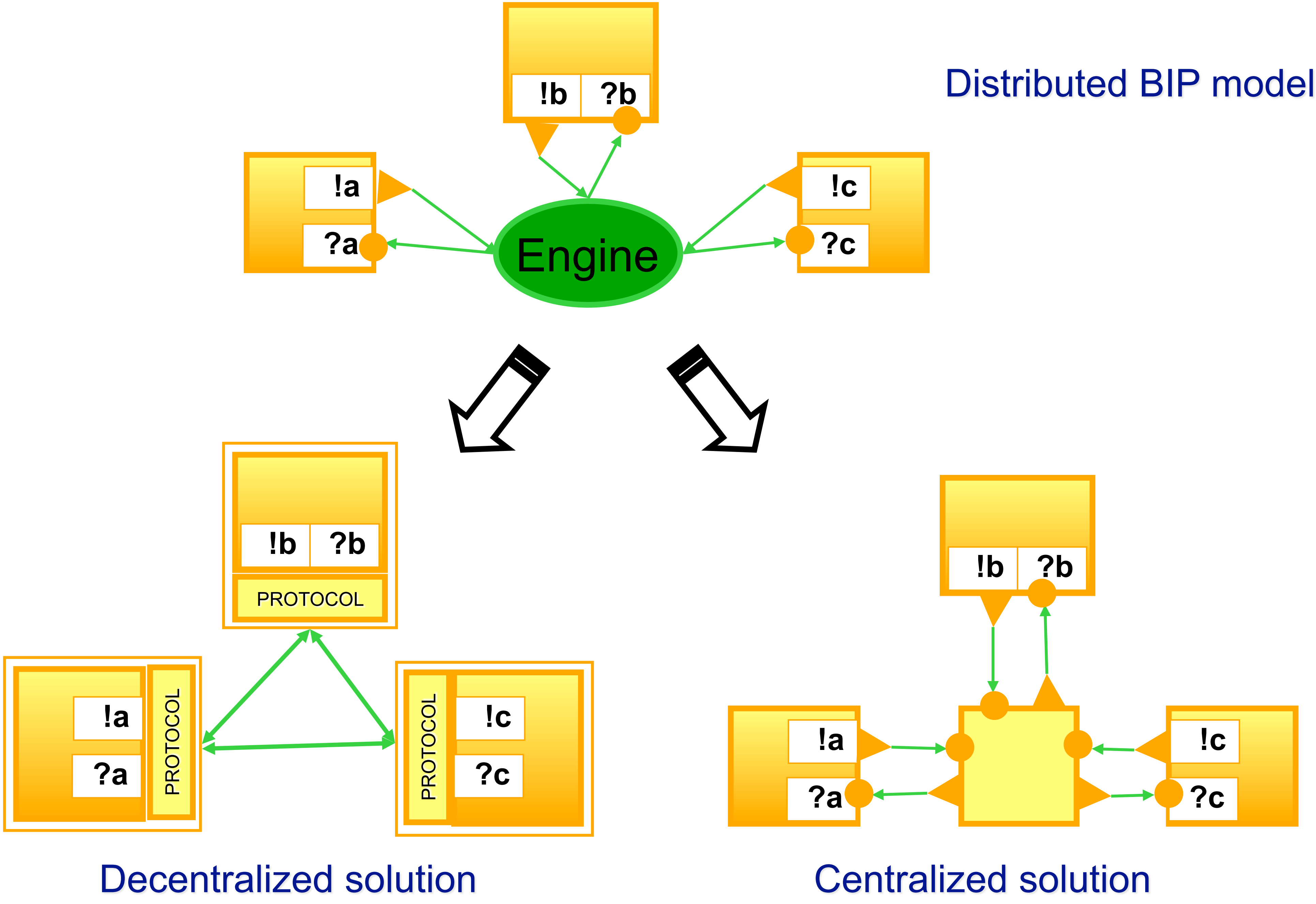
Decentralized Distributed Implementation – The principle



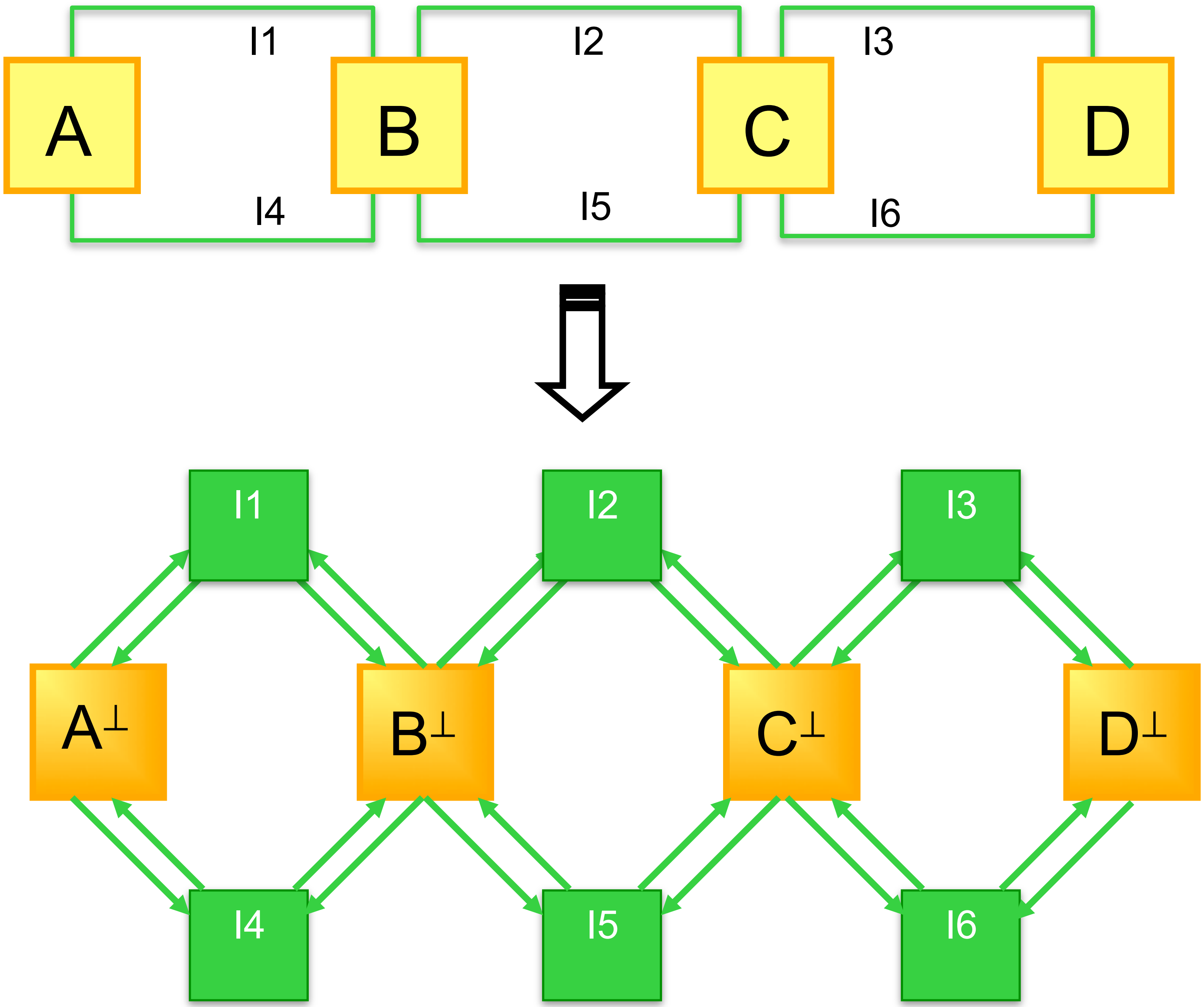
Initial BIP model

Distributed BIP model

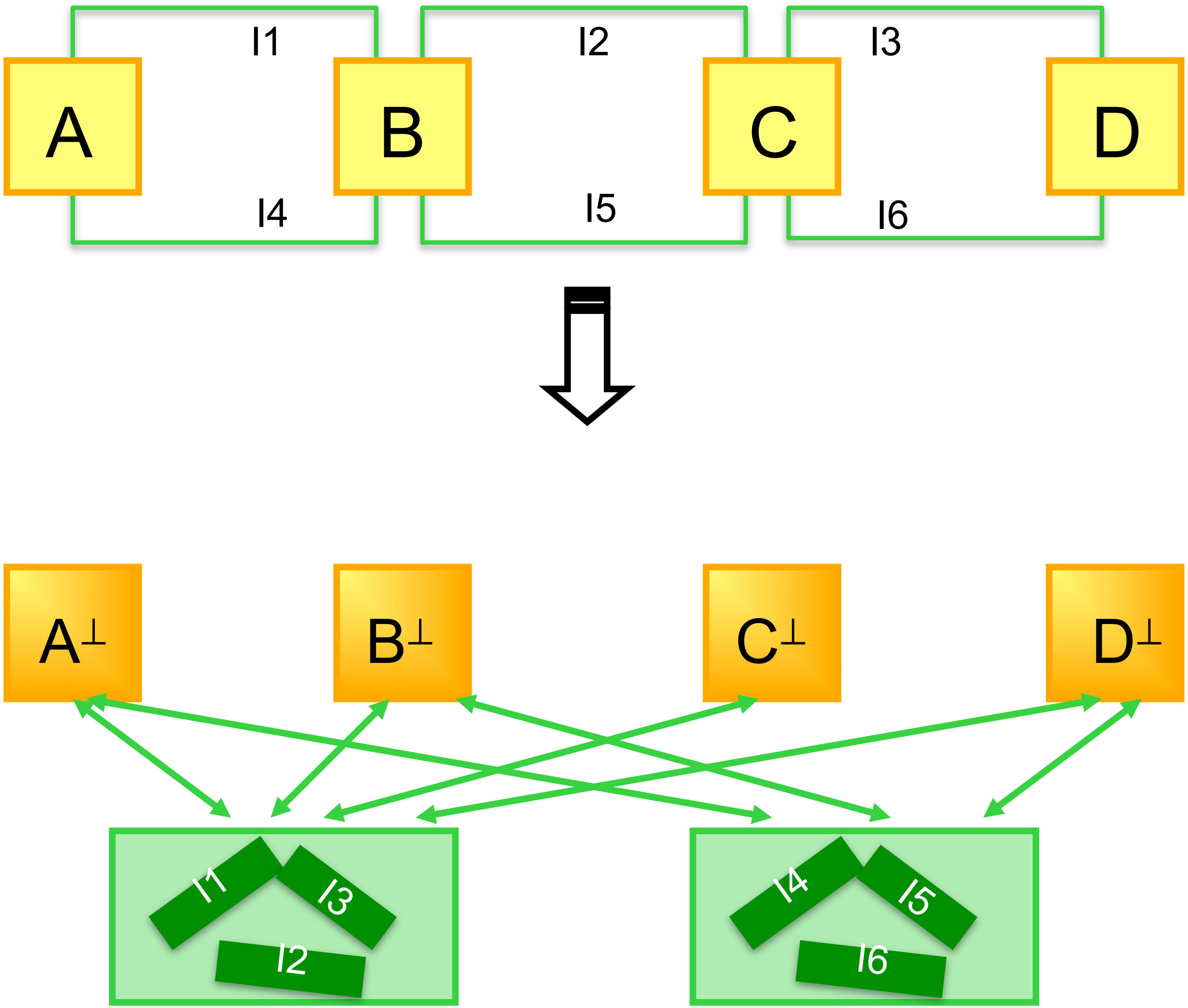
Distributed Implementation – Implementing Connectors



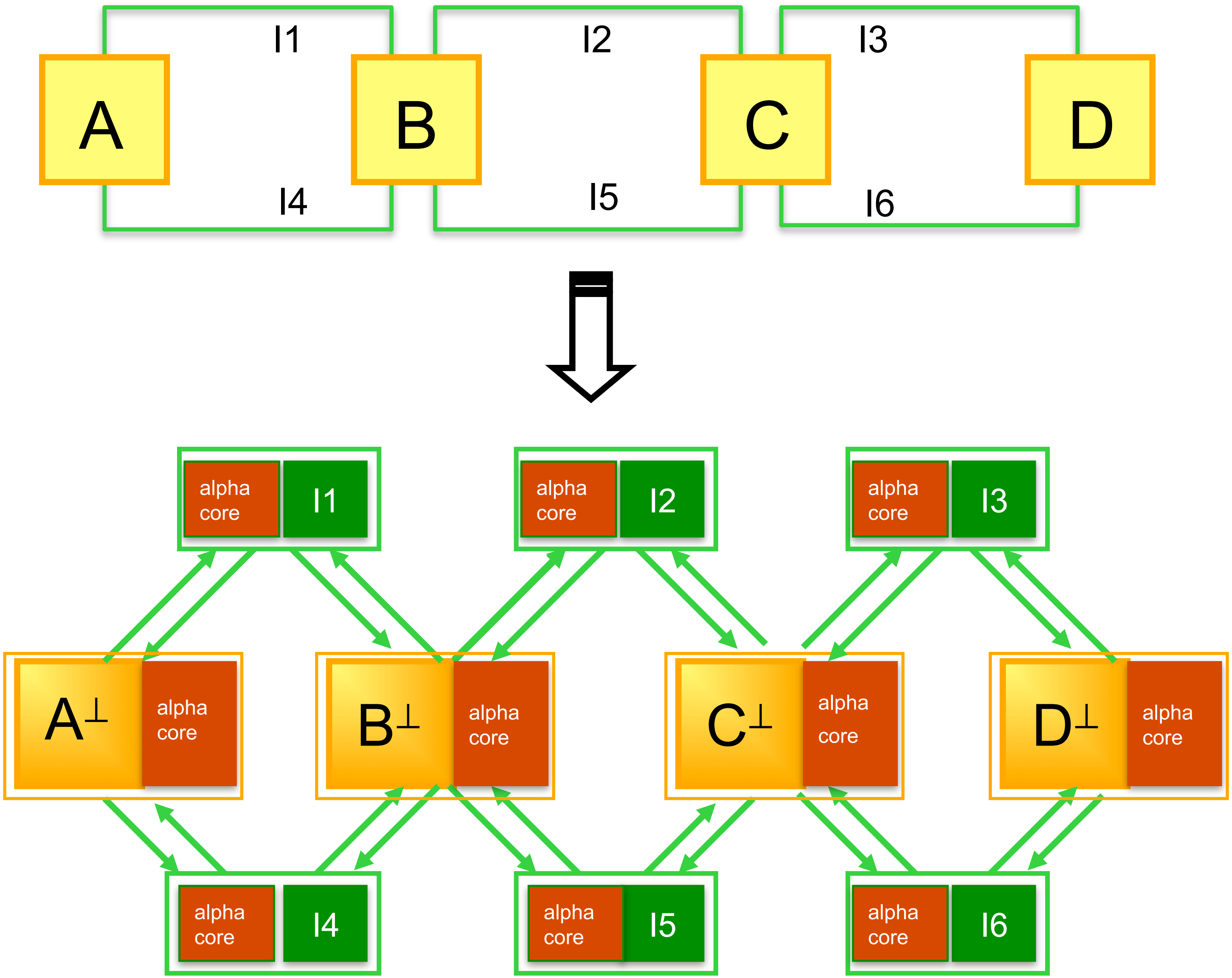
Centralized solution – Conflict Resolution



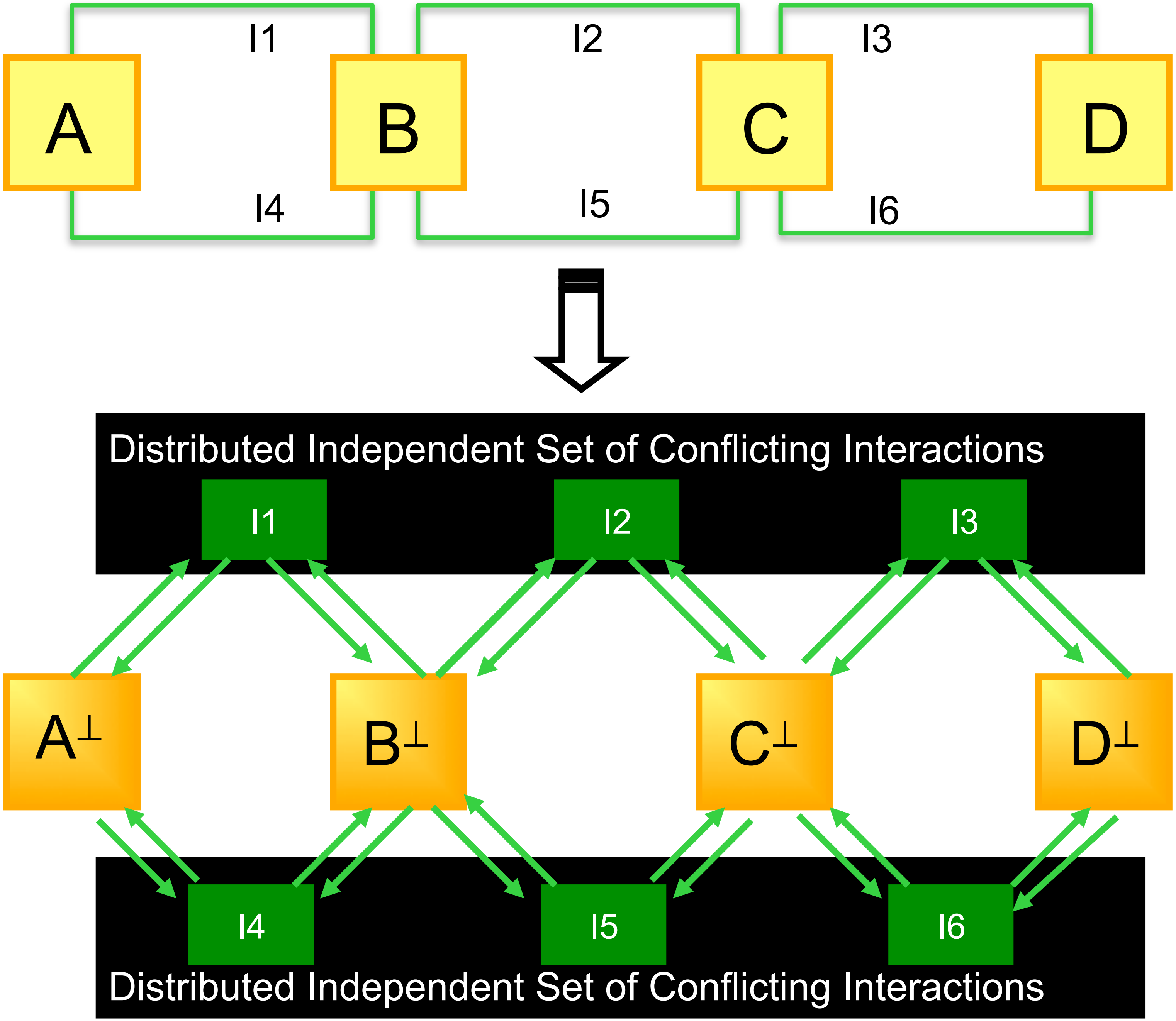
Centralized solution – Conflict Resolution



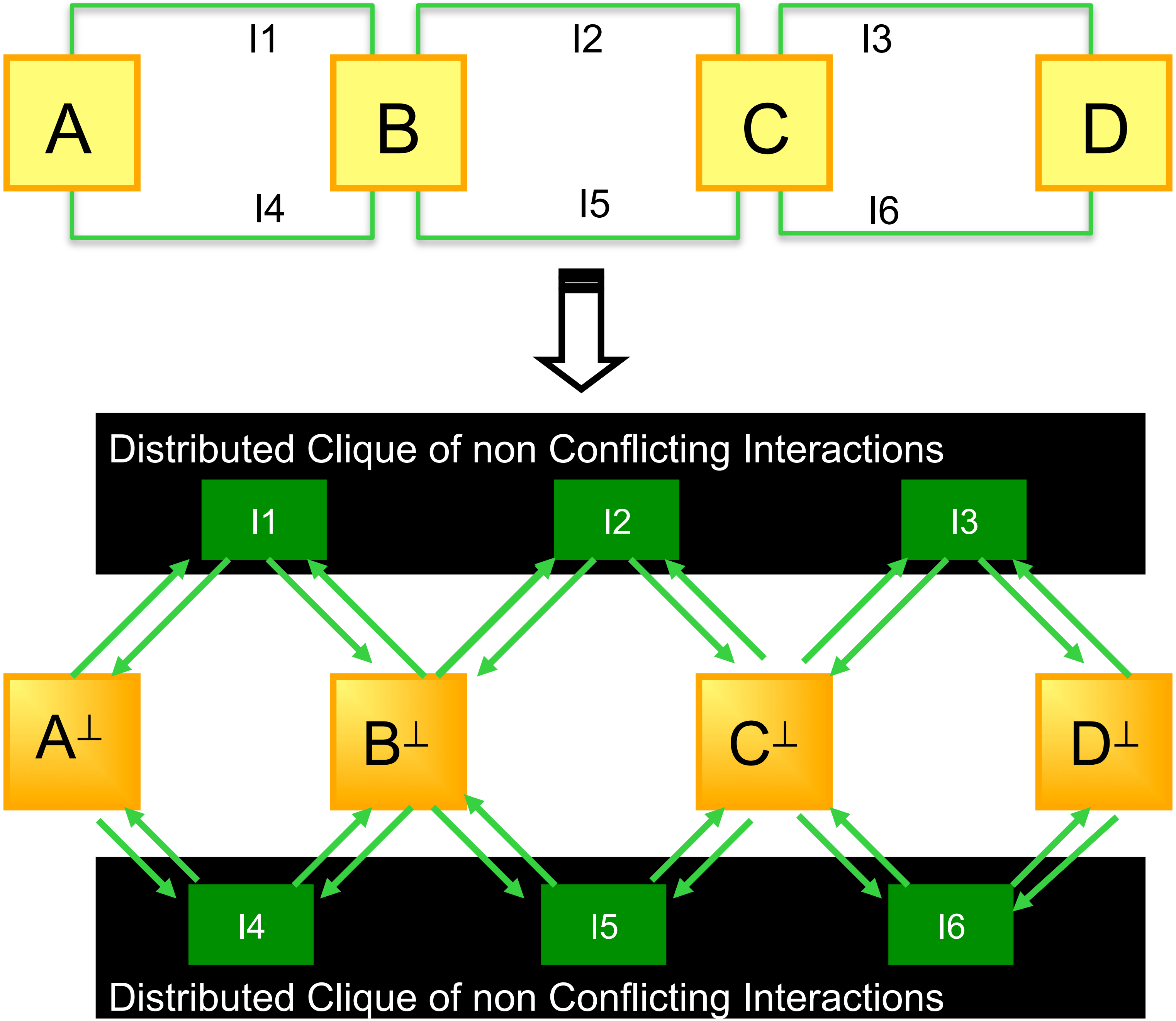
Centralized solution – Conflict Resolution



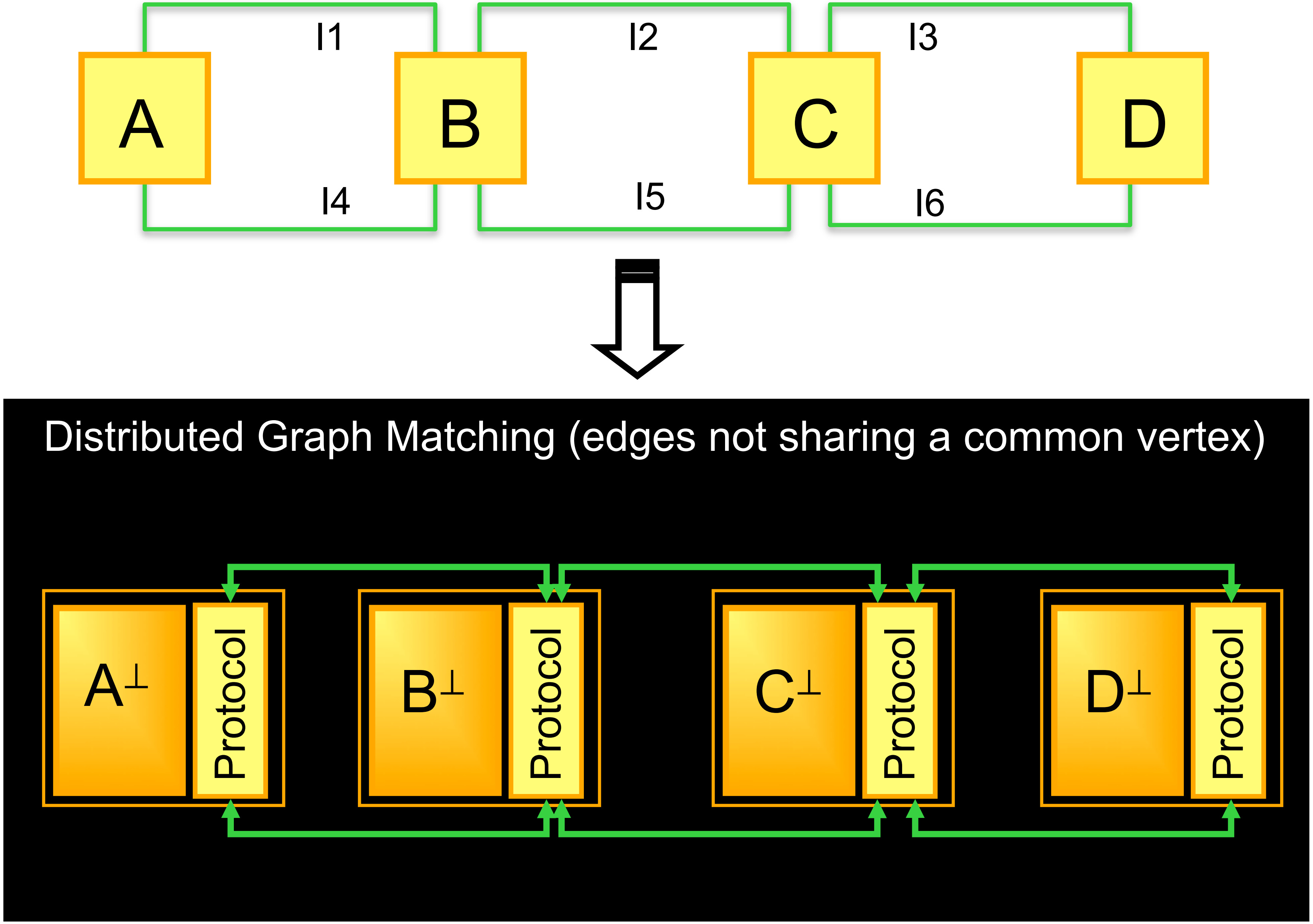
Centralized solution – Conflict Resolution



Centralized solution – Conflict Resolution



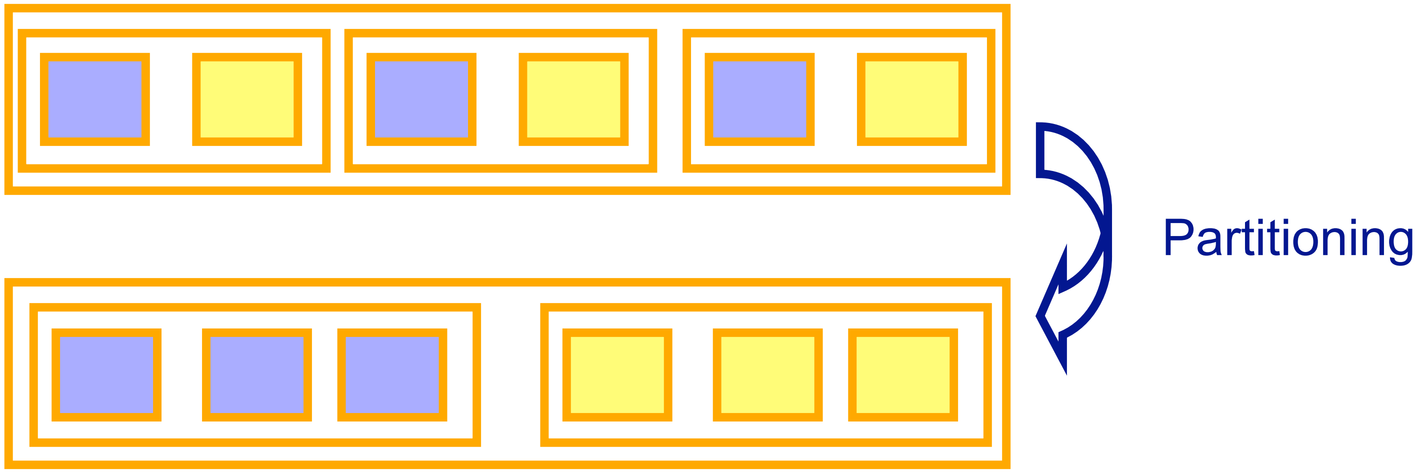
Decentralized Solution

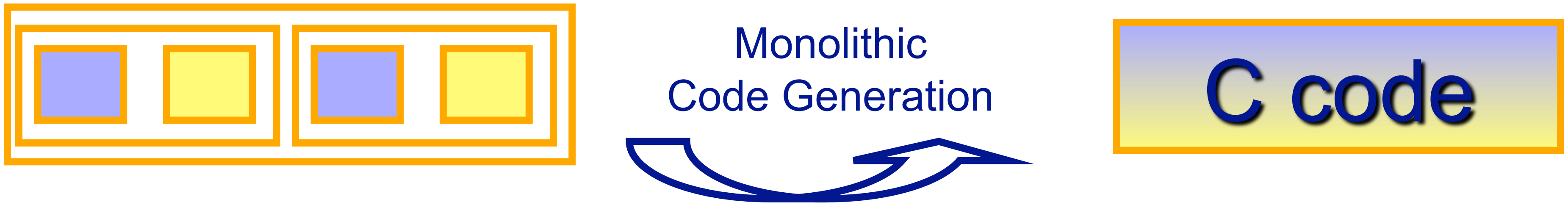


OVERVIEW

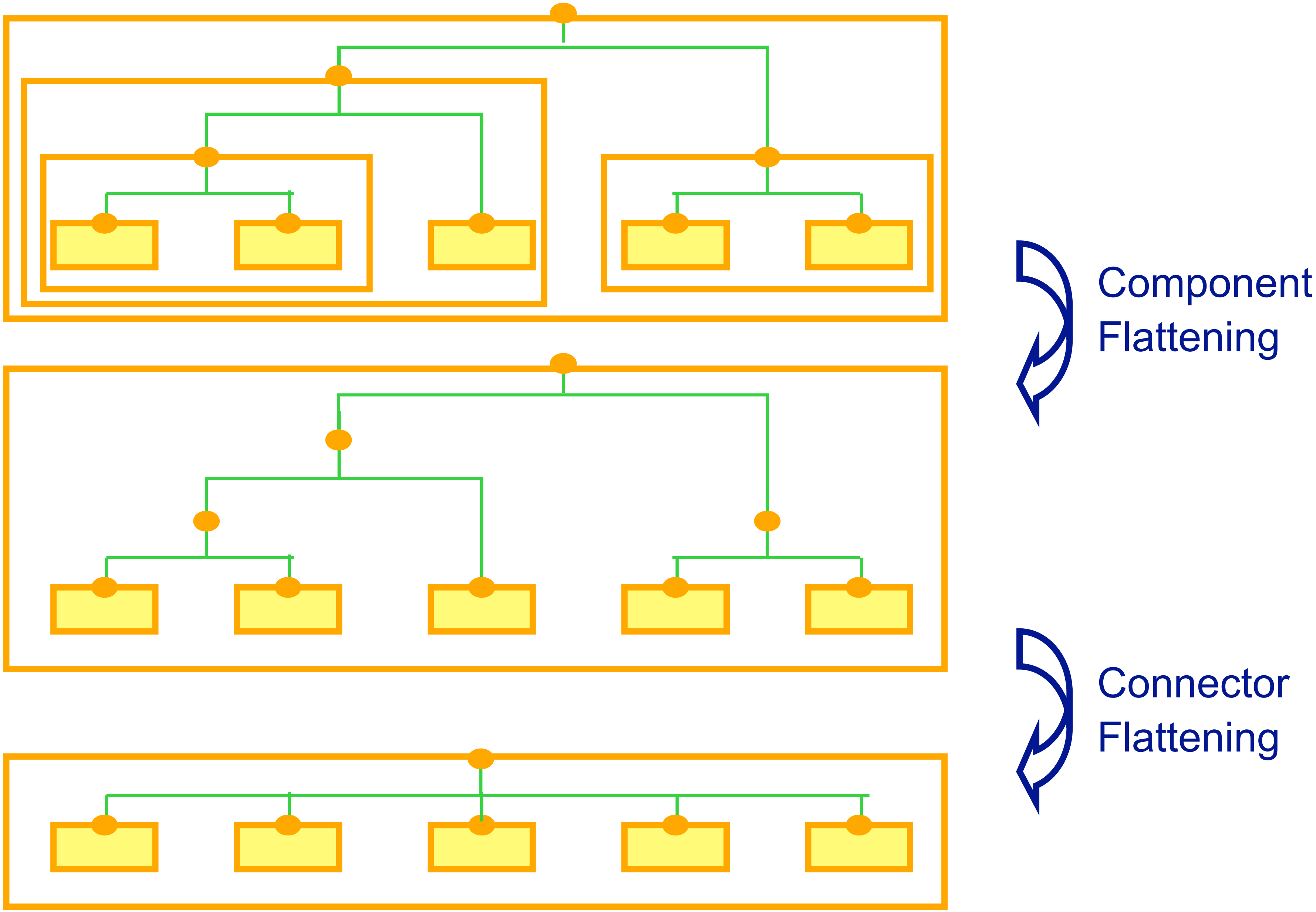
- BIP – Global state model
- Distributed centralized implementation
- Decentralized implementations
- Architecture transformations
- Discussion

Architecture Transformations

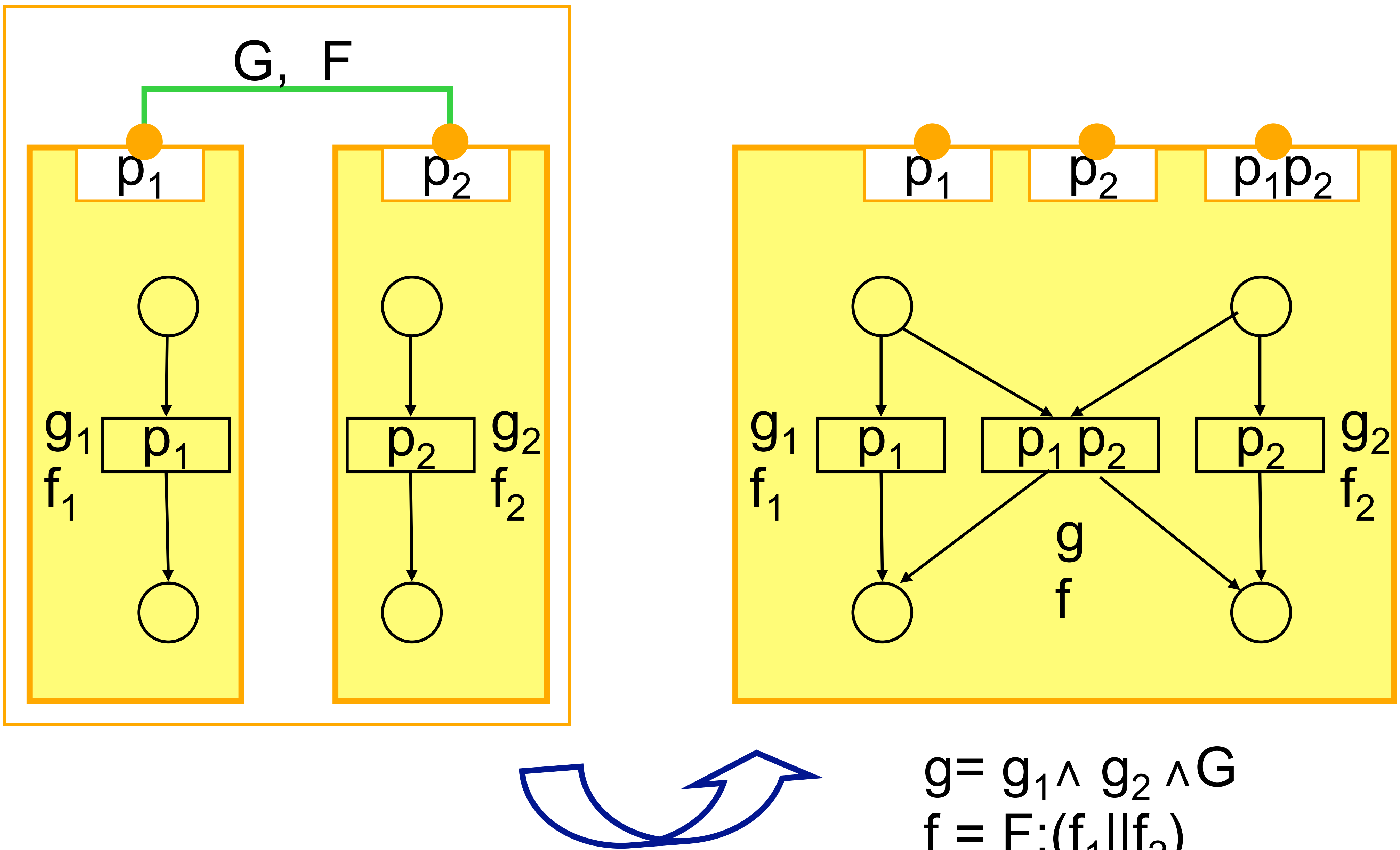




Architecture Transformations – Composite to Monolithic



Architecture Transformations – Composite to Monolithic



$g = g_1 \wedge g_2 \wedge G$

$f = F;(f_1 || f_2)$

Example – MPEG4 Video Encoder

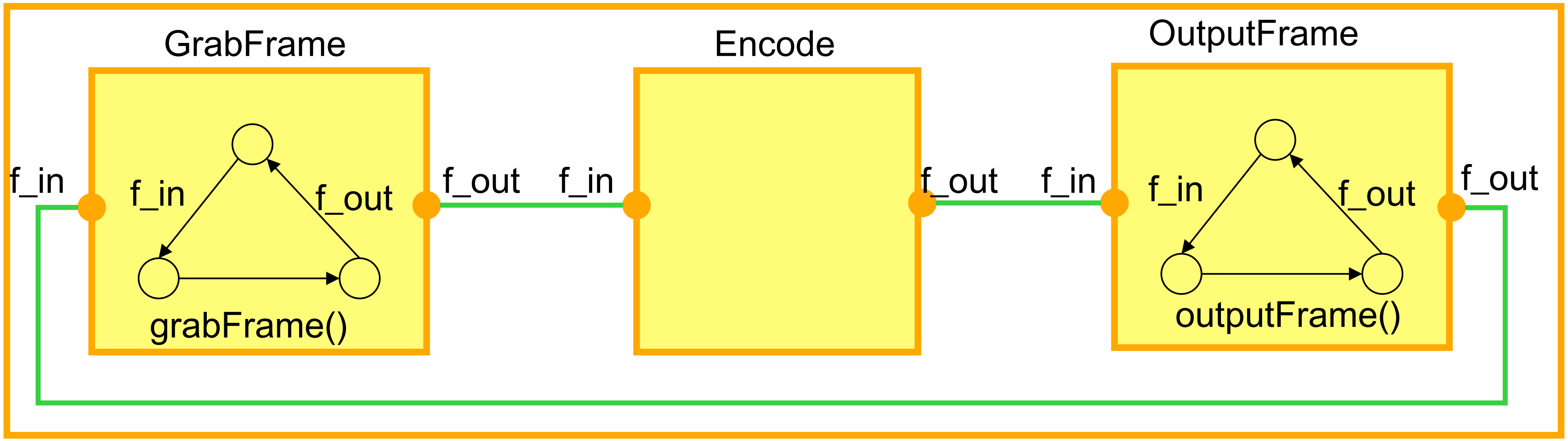
Transform the monolithic sequential program (12000 lines of C code) into a componentized one:

++ reusability, schedulability analysis, reconfigurability

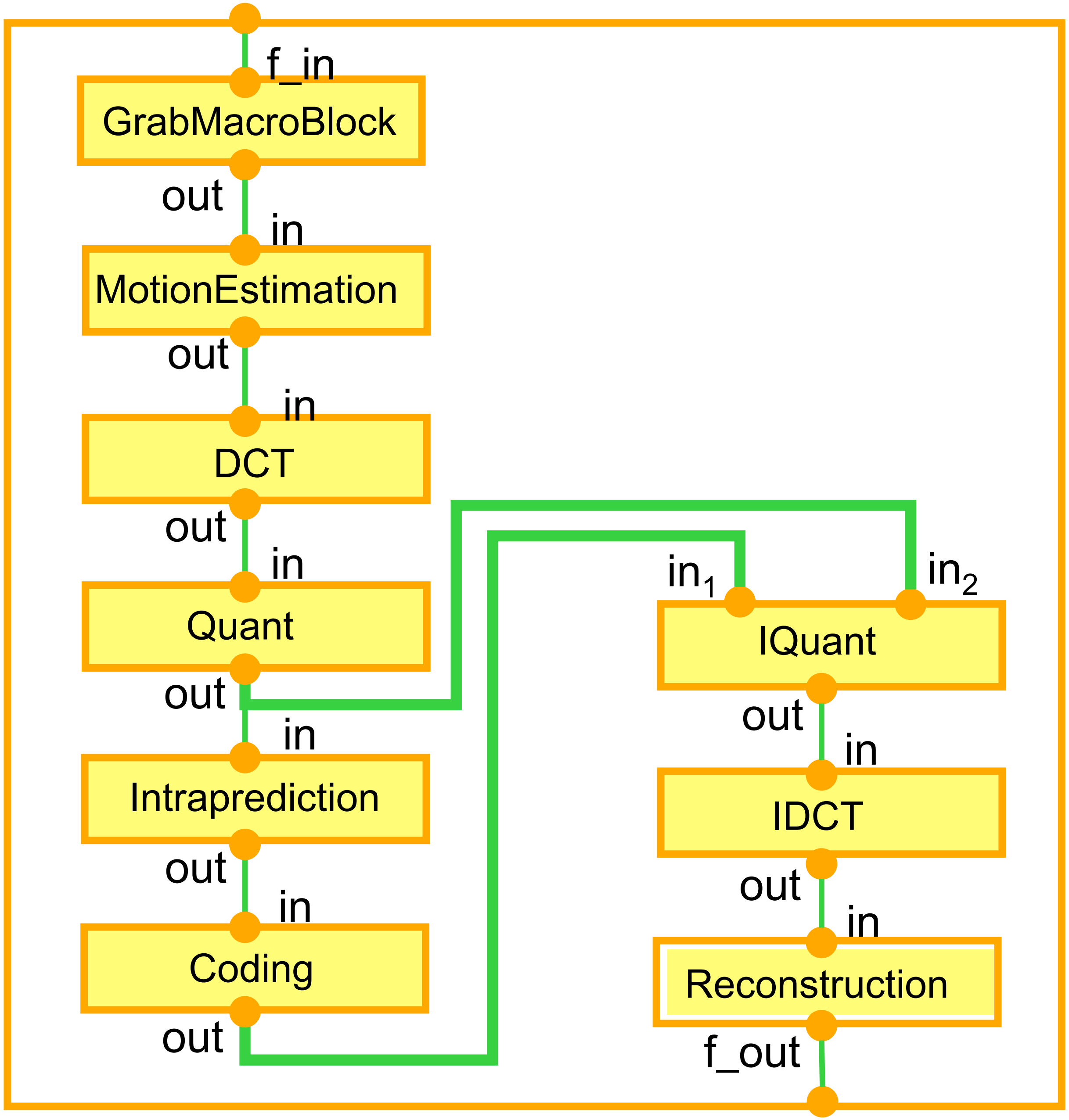
– – overhead in memory and execution time

Decomposition:

- GrabFrame: gets a frame and produces macroblocks
- OutputFrame: produces an encoded frame
- Encode: encodes macroblocks



Example – MPEG4 Video Encoder

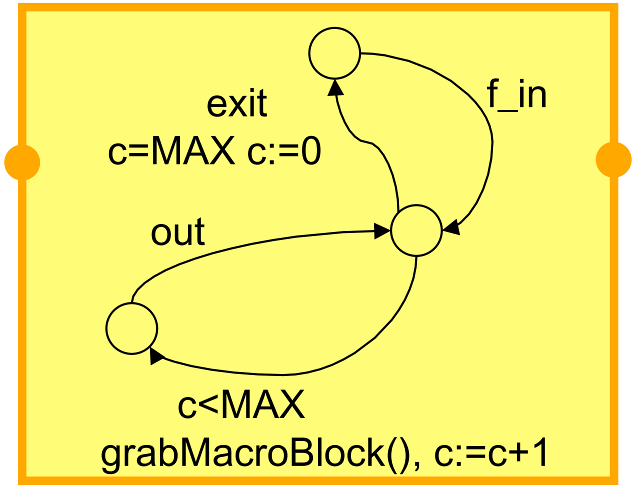


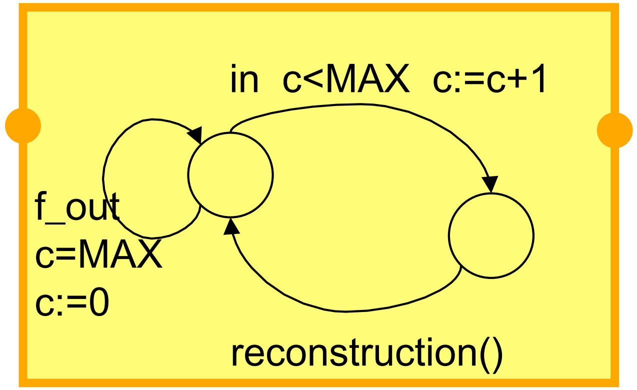
GrabMacroBlock: splits a frame in (W*H)/256 macro blocks, outputs one at a time

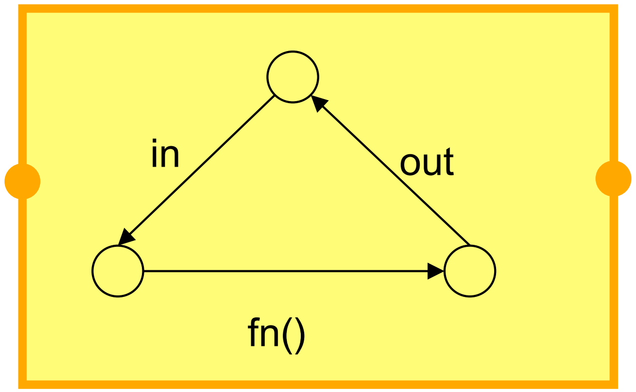
Reconstruction: regenerates the encoded frame from the encoded macro blocks.

: buffered connections

Example – MPEG4 Video Encoder







MAX=(W*H)/256
W=width of frame
H=height of frame

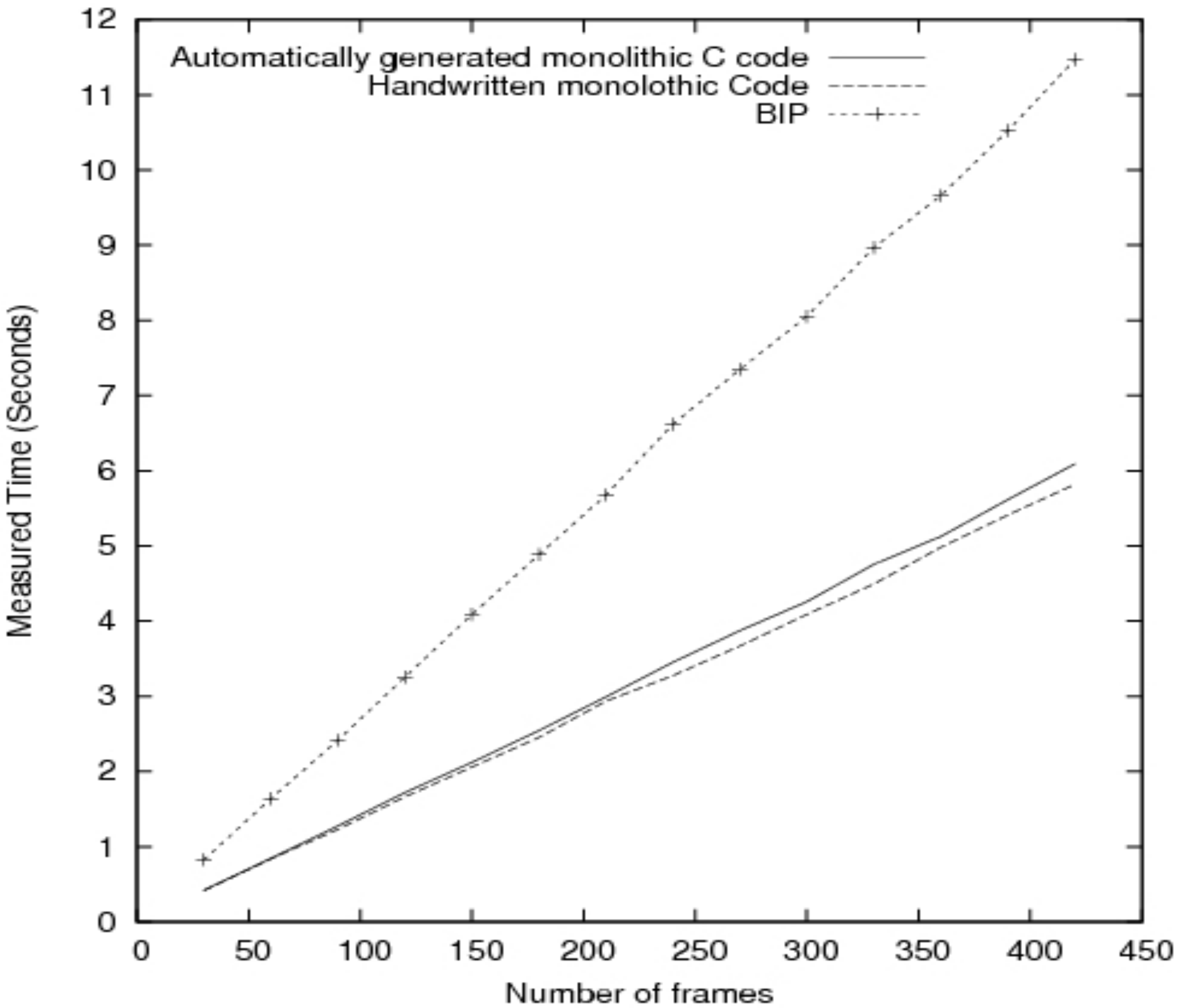
Axample – MPEG4 Video Encoder: Results

- ~ 500 lines of BIP code
  - Consists of 20 atomic components and 34 connectors
  - Components call routines from the encoder library
- The generated C++ code from BIP is ~ 2,000 lines
- BIP binary is 288 KB compared to 172 KB of monolithic binary

100% overhead in execution time wrt monolithic code

- ~66% due to computation of interactions (can be reduced by composing components)
- ~34% due to evaluation of priorities (can be reduced by applying priorities to atomic components)

Source-to-Source – MPEG4 Video Encoder: Results



OVERVIEW

- ❑ BIP – Global state model
- ❑ Distributed centralized implementation
- ❑ Decentralized implementations
- ❑ Architecture transformations
- ❑ Discussion

Results

Papers

- A. Basu, Ph. Bidinger, M. Bozga and J. Sifakis. Distributed Semantics and Implementation for Systems with Interaction and Priority FORTE, 2008, pp. 116-133.
- Marius Bozga, Mohamad Jaber, Joseph Sifakis: Source-to-source architecture transformation for performance optimization in BIP. IEEE Fourth International Symposium on Industrial Embedded Systems 2009, pp 152-160
- Ananda Basu, Saddek Bensalem, Doron Peled, Joseph Sifakis: Priority Scheduling of Distributed Systems Based on Model Checking. CAV 2009: 79-93
- Borzoo Bonakdarpour, Marius Bozga, Mohamad Jaber, Joseph Sifakis. Incremental Component-based Modeling, Verification, and Performnce Evaluation of Distributed Reset, DISC 09.

Distributed Implementations

- Centralized BIP Engine (FORTE08)
- Weakly decentralized (one Engine per set of conflicting interactions) over Linux using TCP sockets
- Weakly decentralized (one Engine per interaction + Conflict resolution with alpha-core) over Linux using TCP sockets

Discussion

- Study different distributed implementations from fully decentralized to fully centralized ones
- Use existing distributed algorithms for multiparty interaction and conflict resolution e.g. maximal matching algorithm
- Prove correctness by using composability techniques - non interference of features of the composed algorithms
- Performance evaluation – tradeoffs wrt two criteria:
  - degree of parallelism
  - overhead for coordination
- Implementation tools and case studies.

Thank You