

Symbolic Implementation of Connectors in BIP



Ananda Basu, Mohamad Jaber

Simon Bliudze

Presentation info: Simon.Bliudze@cea.fr

ICE'09, Bologna, August 31, 2009

Presentation outline

- BIP
- Connectors
- Boolean encoding
- Benchmarks

Motivation

Context: *Component-based* modelling, design and validation of embedded real-time systems.

Presently:

- A number of *coordination mechanisms* for concurrent systems: shared variables, semaphores, regions, etc.
- Ad-hoc use and analysis methodologies.

Our goal: *Unified framework* for component-based modelling and design

- allowing incremental description & correctness by construction
- encompassing heterogeneity
 - synchronous and asynchronous execution
 - event and data driven computation
 - centralised and distributed implementation



Component design by refinement

Three layers:

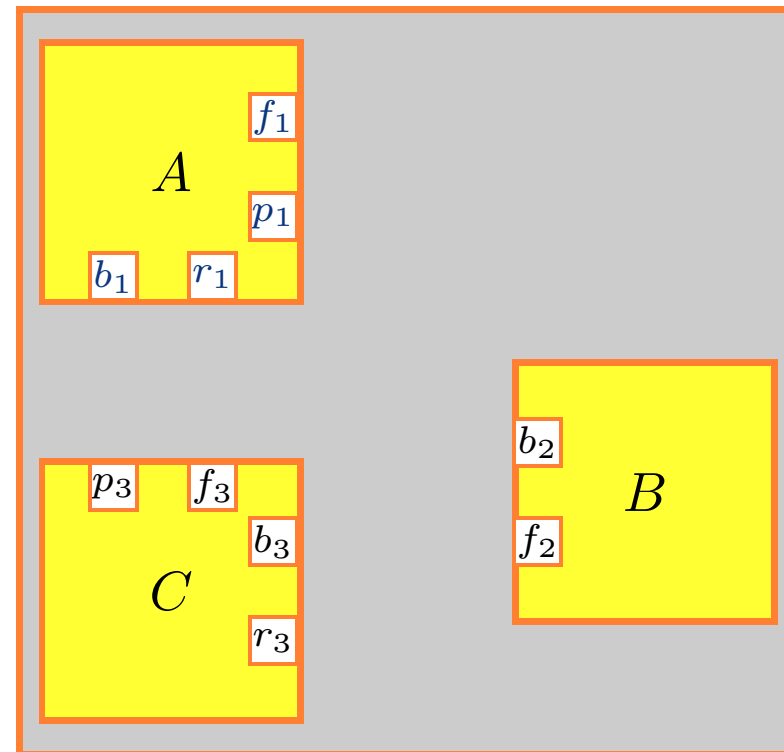
1. Component behaviour
2. Coordination
3. Data transfer



Component design by refinement

Three layers:

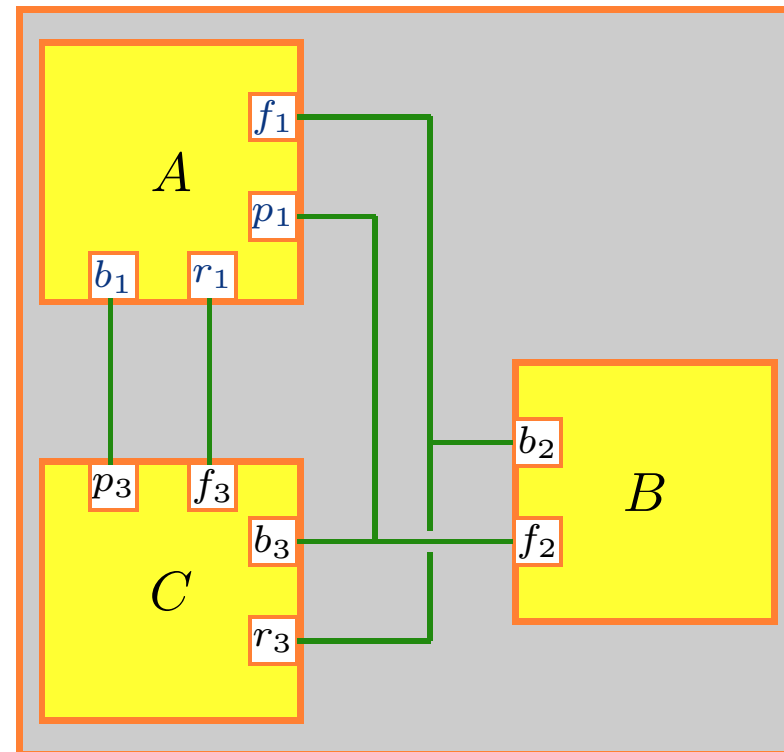
1. Component behaviour
2. Coordination
3. Data transfer



Component design by refinement

Three layers:

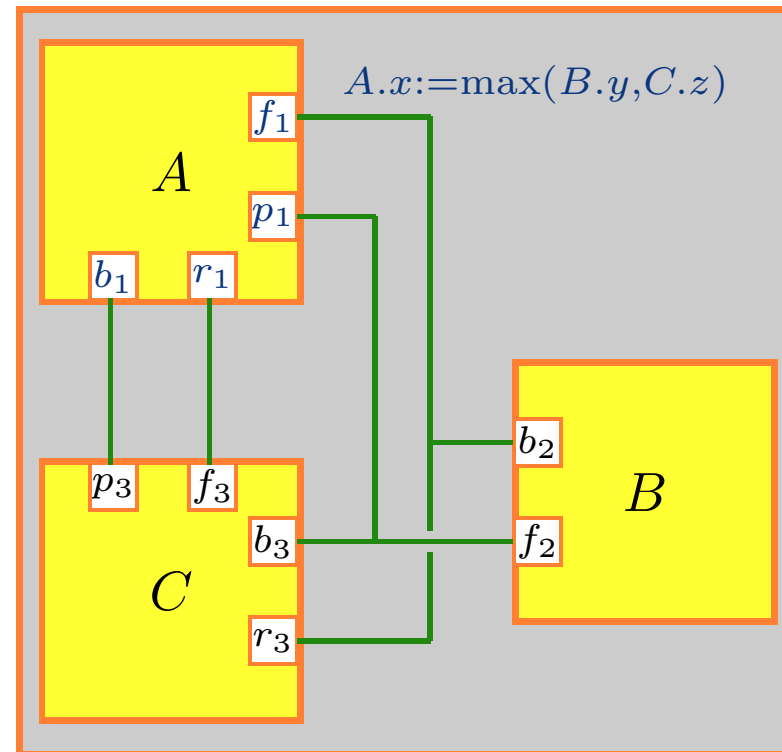
1. Component behaviour
2. Coordination
3. Data transfer



Component design by refinement

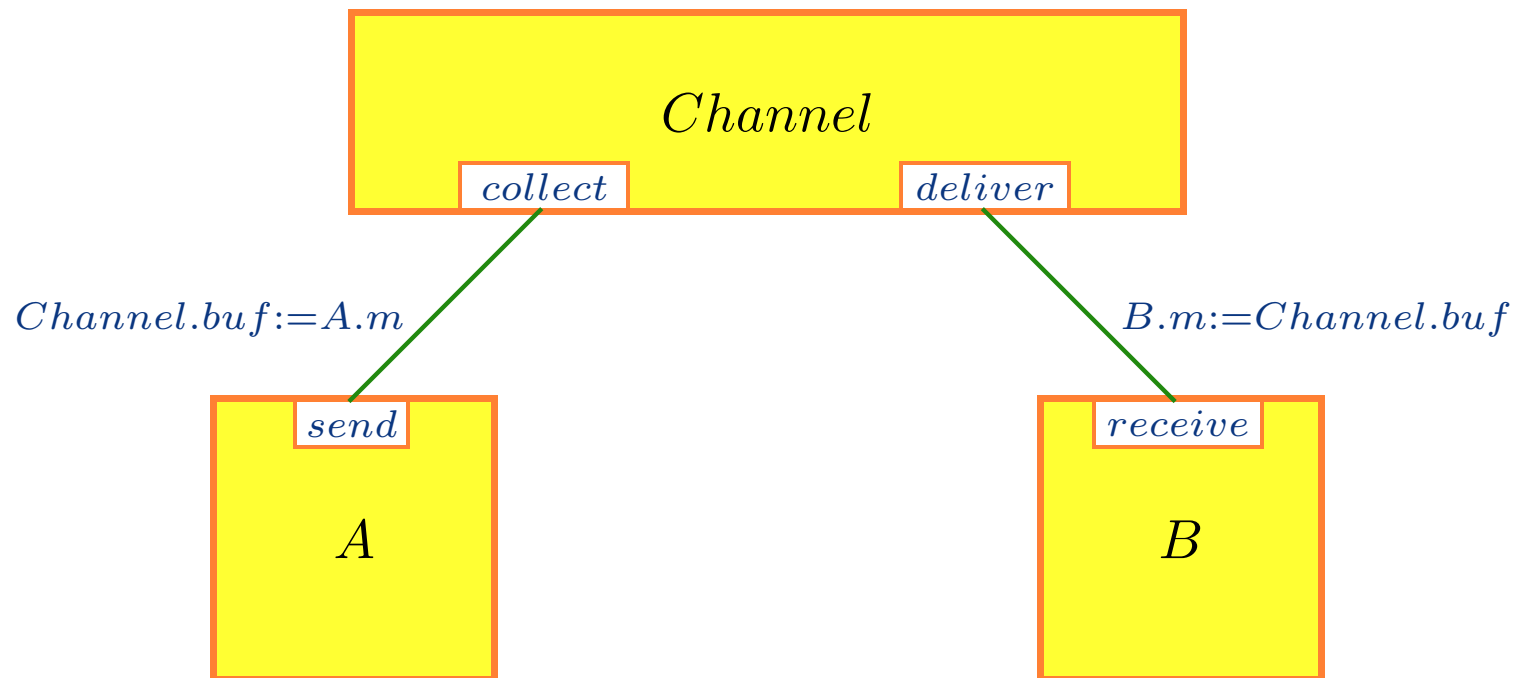
Three layers:

1. Component behaviour
2. Coordination
3. Data transfer



Unbuffered synchronous communication

(Not to confuse with synchronous *execution*!)



A sends a message *m* to *B*:

- Two synchronisations with the channel
- Each synchronisation allows a data transfer
- An explicit model of the channel behaviour

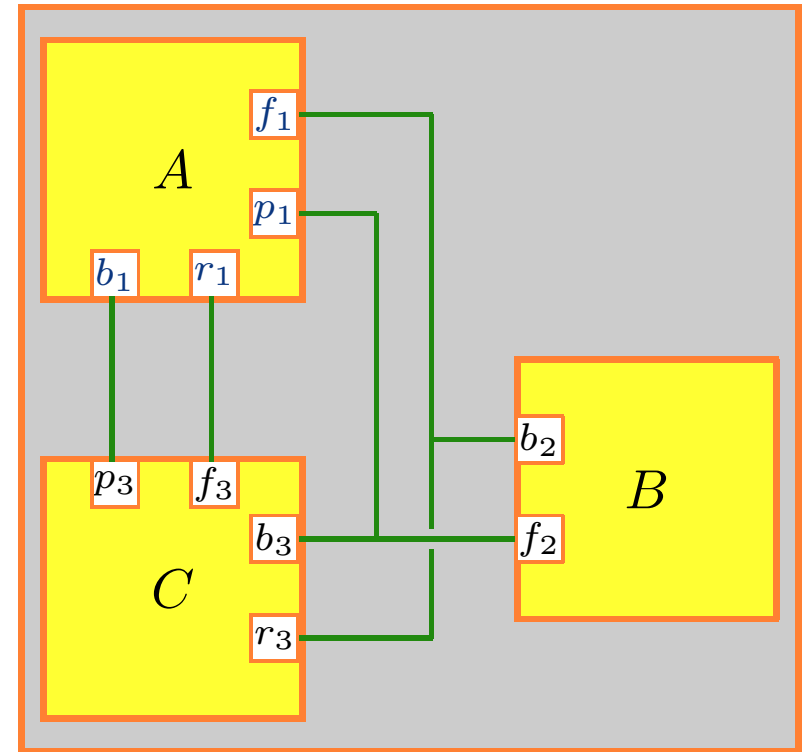
Scope of the basic model

Three layers:

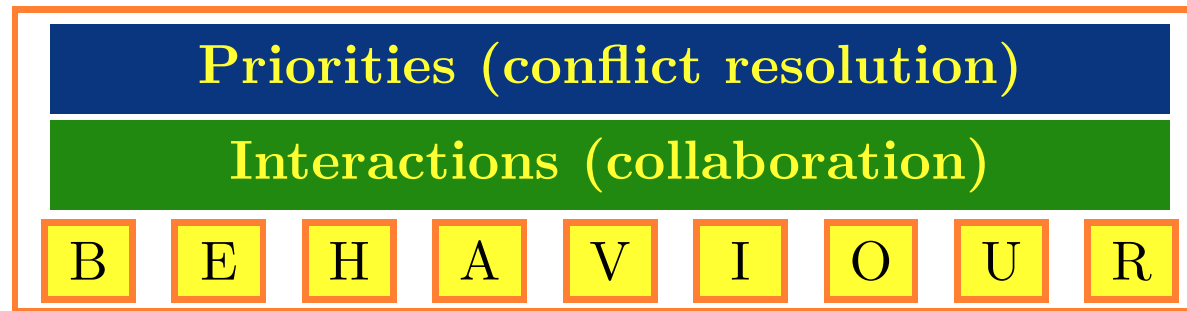
1. Component behaviour
2. Coordination
3. Data transfer

Interesting results already at this level, e.g.

- Analysis of synchronisation deadlocks
 - S. Bensalem, M. Bozga, J. Sifakis, T.-H. Nguyen. *D-Finder: A Tool for Compositional Deadlock Detection and Verification*. [CAV'09]
- Synthesis of glue for certain safety properties
 - S. Bliudze, J. Sifakis. *Synthesizing Glue Operators from Glue Constraints for the Construction of Component-Based Systems*. [Submitted to POPL'10]



Basic model of BIP

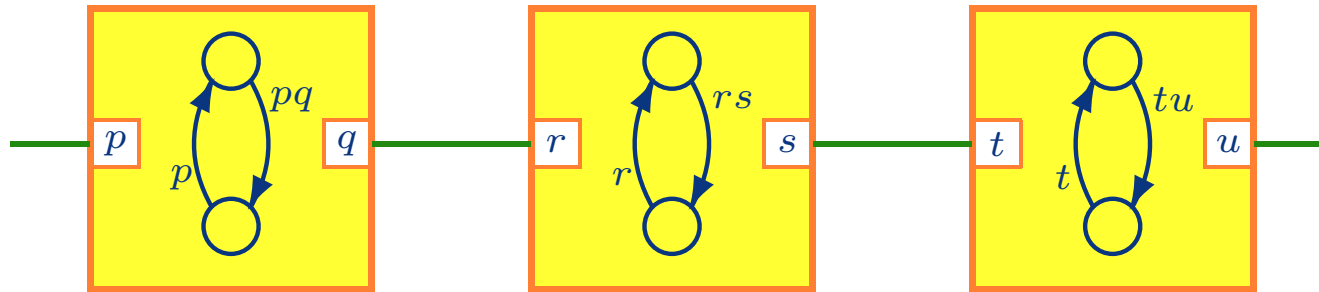


Layered component model

- **Behaviour** — labelled transition systems with **disjoint** sets of ports
- **Interaction** — set of interactions (interaction = set of ports)
- **Priorities** — strict partial order on interactions

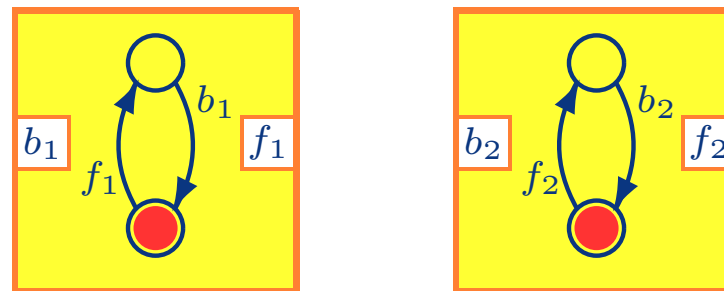
BIP: Examples

Modulo-8 counter:



Interactions: $\{p, pqr, pqr\ st, pqr\ stu\}$.

Mutual exclusion:



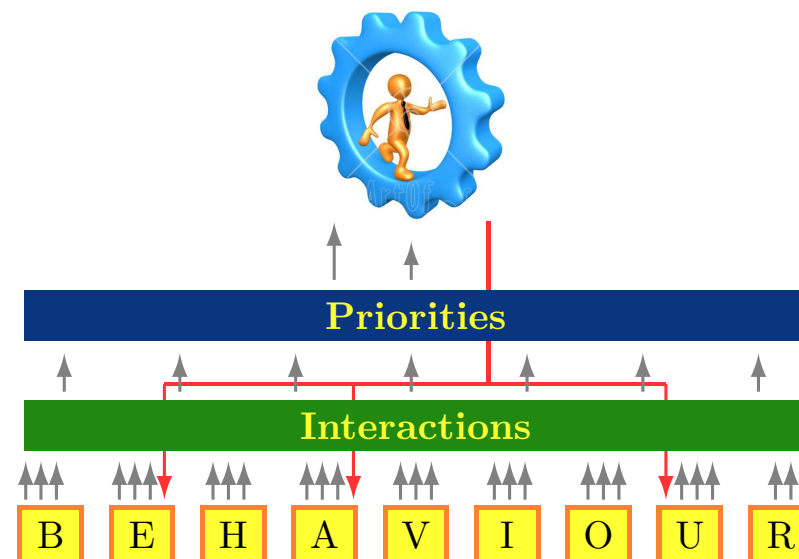
Interactions: $\{b_1, f_1, b_2, f_2\}$

Priority: $b_1 \prec f_2, b_2 \prec f_1$.

Implementation: Enumerative engine

Engine protocol:

1. Atoms notify the engine of their active ports.
2. The engine enumerates the allowed interactions;
3. Filters out low priority ones;
4. Picks one among those left;
5. Notifies the atoms.



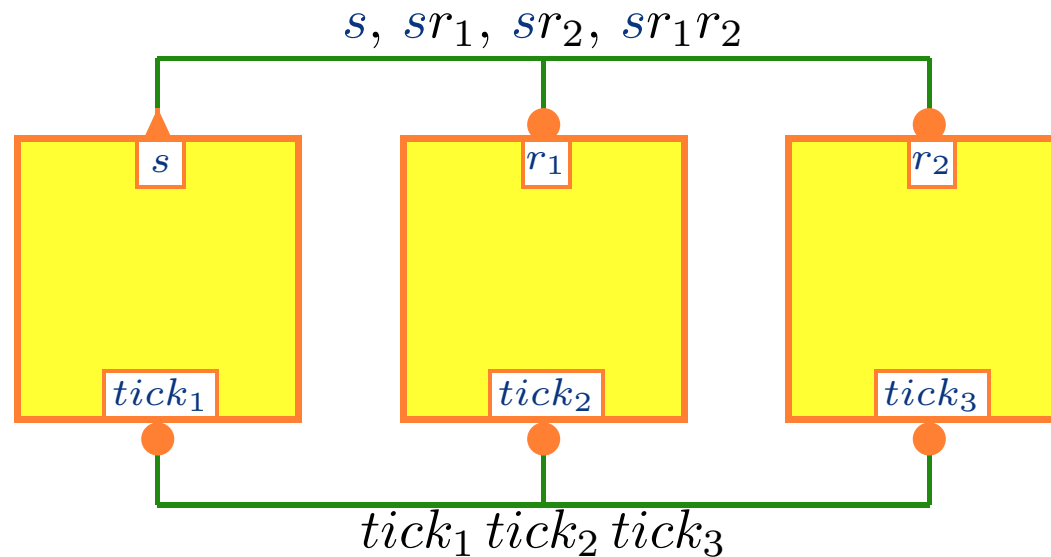
Interaction model is a **set of sets** of ports \Rightarrow exponential complexity.

Presentation outline

- BIP
- Connectors
- Boolean encoding
- Benchmarks

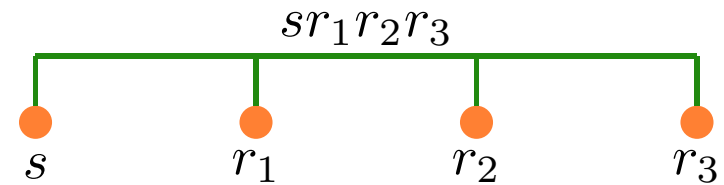
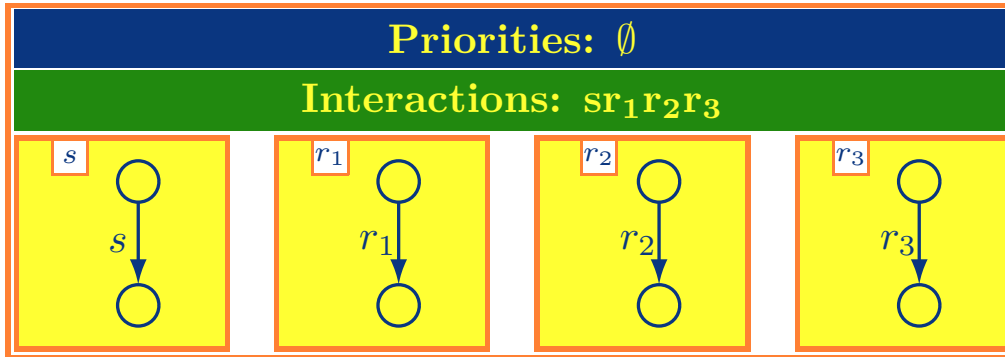
Basic connectors

- A **connector** is a set of ports which can be involved in an interaction.
- Port attributes (**trigger** ▲, **synchron** ●) determine the synchronisation type.
- An **interaction** in a connector is a subset of ports such that either it contains a trigger or it is maximal.

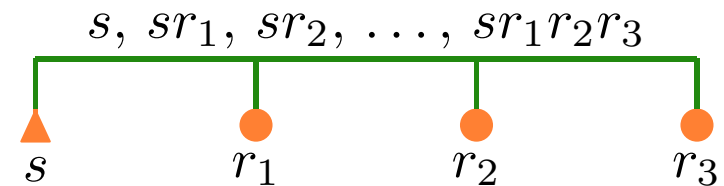
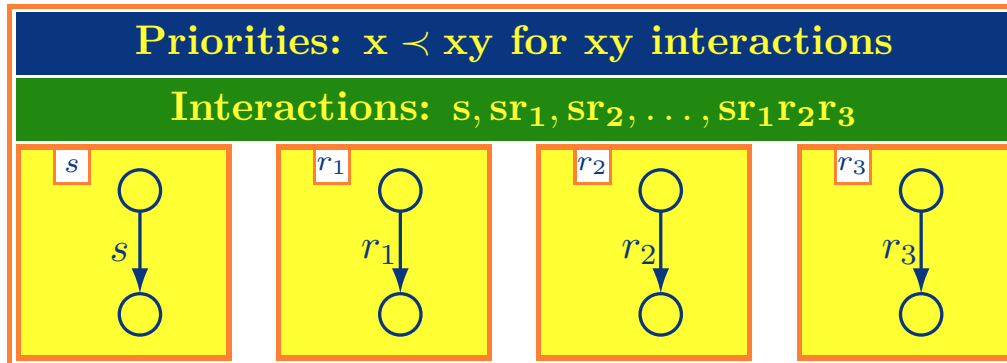


Interaction modelling: Flat connectors

Rendezvous

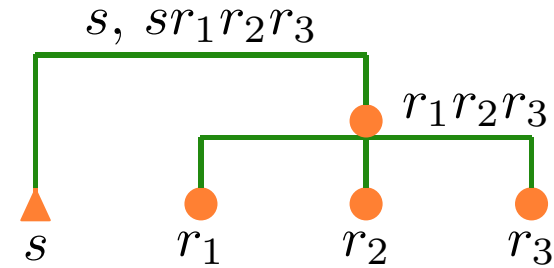
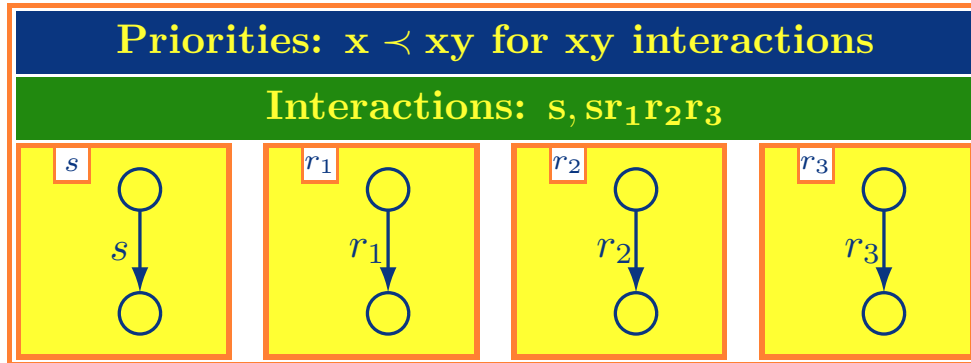


Broadcast

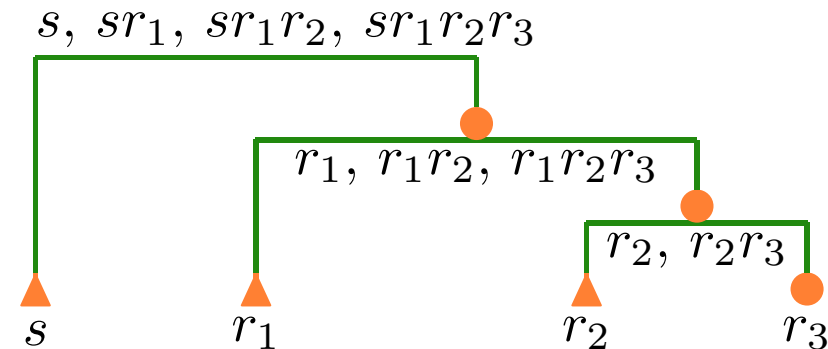
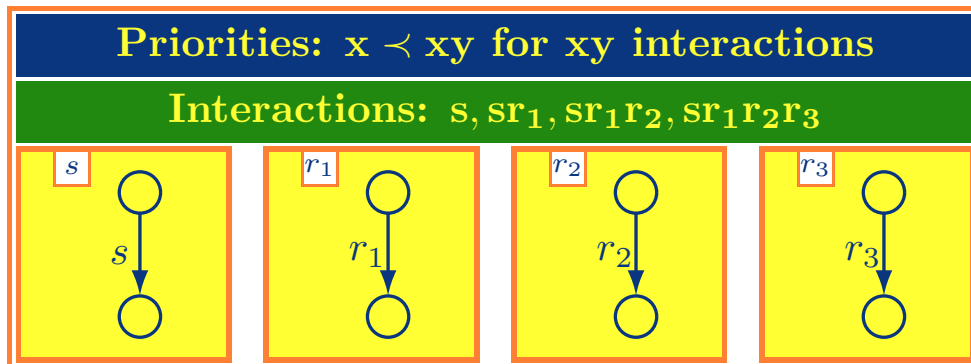


Interaction modelling: Hierarchical connectors

Atomic broadcast



Causality chain



The Algebra of Connectors, $\mathcal{AC}(P)$, introduced in [EMSOF T'07].

Presentation outline

- BIP
- Connectors
- Boolean encoding
- Benchmarks

Symbolic implementation

Motivation: Reduce Component/Engine protocol overhead.

Approach: Use boolean functions to pick the interaction to be executed.

- An interaction can be seen as a boolean valuation on ports:

$$p_1 p_2 \longleftrightarrow \begin{pmatrix} p_1 & p_2 & p_3 & p_4 \\ 1 & 1 & 0 & 0 \end{pmatrix}$$

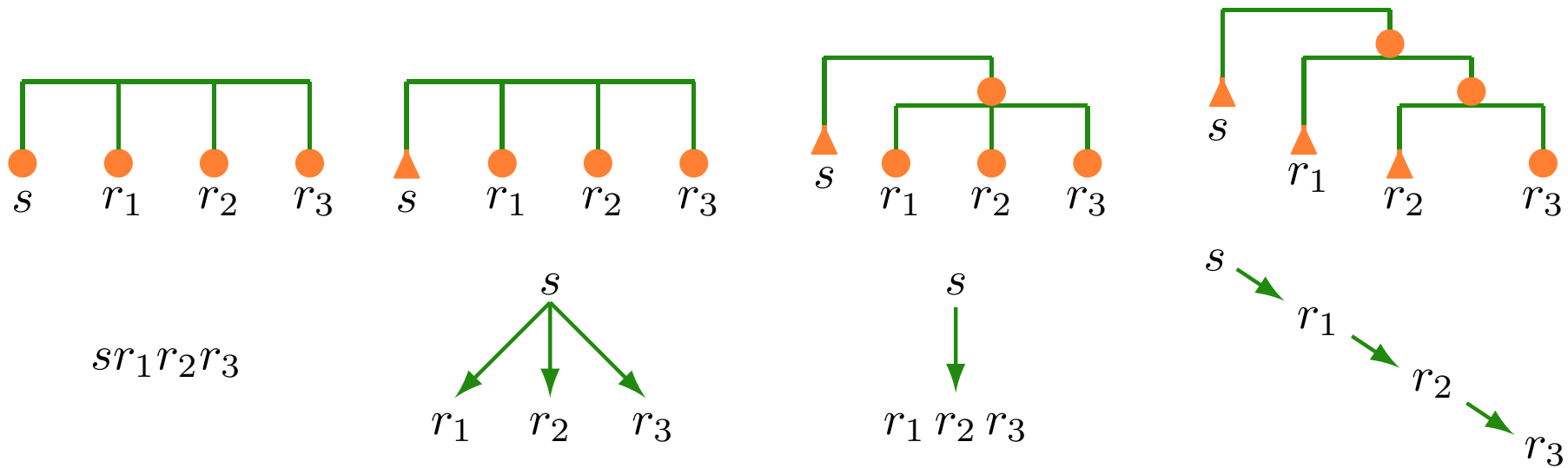
- Based on the connectors, priorities, and behaviour of components, compute a boolean formula φ on $P \cup Q$, such that

$$(a, q) \models \varphi \iff a \text{ is enabled in } q.$$

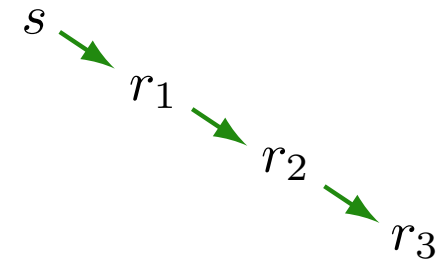
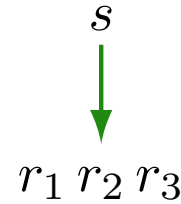
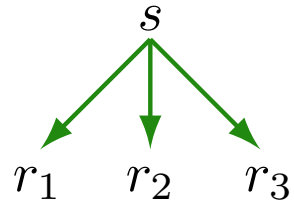
- Implementation using existing BDD package.



Causal interaction trees



$sr_1r_2r_3$



Rendezvous

$sr_1r_2r_3$

Broadcast

s, sr_1, sr_2, \dots

$sr_2r_3, sr_1r_2r_3$

Atomic broadcast

$s, sr_1r_2r_3$

Causal chain

$s, sr_1, sr_1r_2,$

$sr_1r_2r_3$

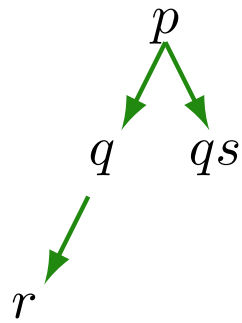
Transformations are straightforward in both ways.

The Algebra of Causal Trees, $CT(P)$, introduced in [FMCO'07].



Boolean representation of connectors

$p'[q'r][qs]$:



Causal trees

$true \Rightarrow p,$
 $q \Rightarrow p,$
 $r \Rightarrow pq,$
 $s \Rightarrow pq$

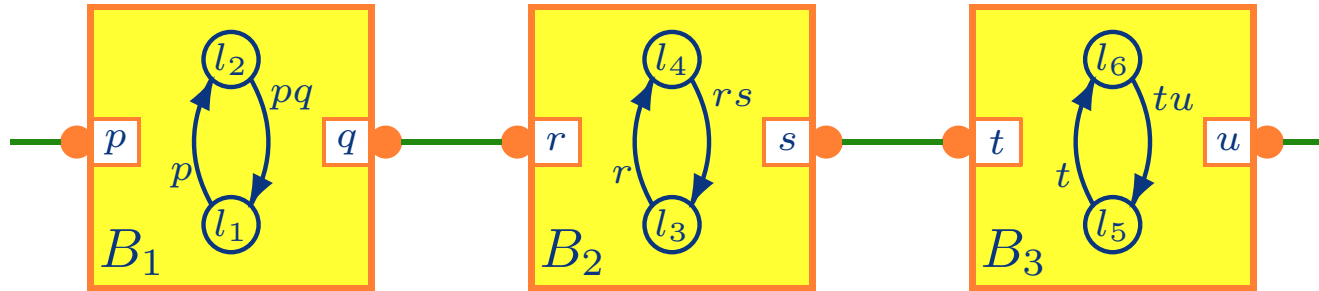
Causal rules

Notice that: $(q \Rightarrow p \vee ps) \equiv (q \Rightarrow p)$.

The corresponding boolean formula is

$$(true \Rightarrow p) \wedge (q \Rightarrow p) \wedge (r \Rightarrow pq) \wedge (s \Rightarrow pq) \equiv pq \vee p\bar{r}\bar{s}.$$

Boolean representation of atomic components



$$f_{B_1} = l_1 \bar{l}_2 p \bar{q} \vee \bar{l}_1 l_2 p q \vee \bar{p} \bar{q}$$

$$f_B = (l_1 \bar{l}_2 p \bar{q} \vee \bar{l}_1 l_2 p q \vee \bar{p} \bar{q}) \wedge (l_3 \bar{l}_4 r \bar{s} \vee \bar{l}_3 l_4 r s \vee \bar{r} \bar{s}) \\ \wedge (l_5 \bar{l}_6 t \bar{u} \vee \bar{l}_5 l_6 t u \vee \bar{t} \bar{u})$$

$$f_C = (p \vee q r \vee s t \vee u) \wedge (q = r) \wedge (s = t)$$

Implementation: Symbolic engine*

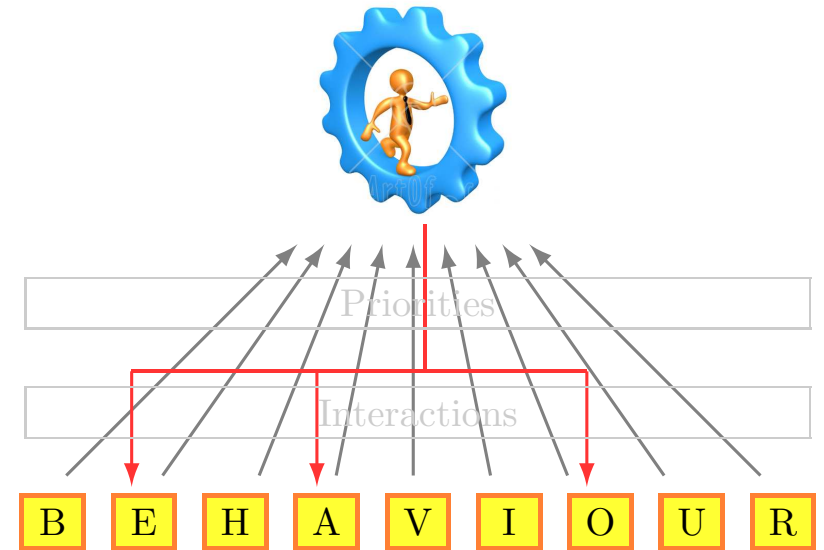
Initialization:

0. The engine precomputes functions

$f_B \in \mathbb{B}[\bigcup_{i=1}^n Q_i, P]$ — behaviour,

$f_C \in \mathbb{B}[P]$ — connectors,

$f_S = f_B \wedge f_C$.



Engine protocol:

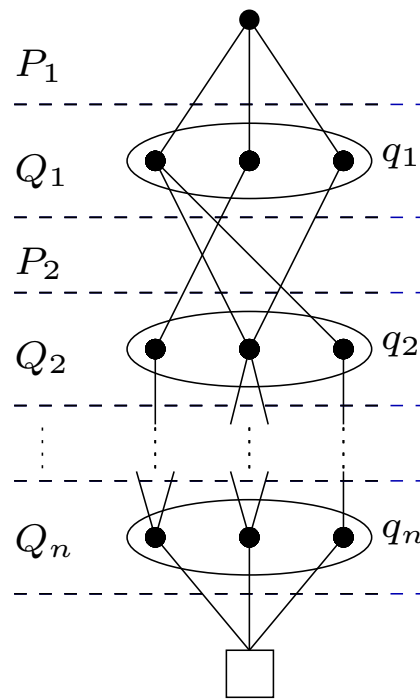
1. Atoms notify the engine of their current states q_i .
2. The engine picks any valuation a on P , such that

$$(a, q) \models f_S \wedge \bigwedge_{i=1}^n q_i$$

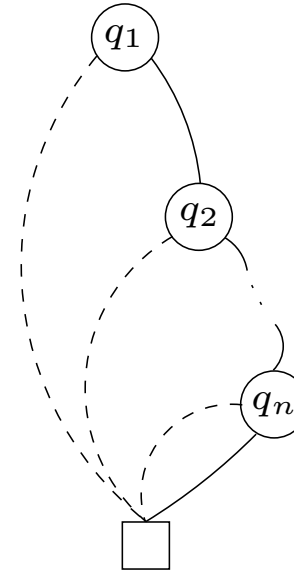
3. Notifies the atoms.

* Connectors only (for priorities see in the paper).

Complexity of the symbolic engine



$$f_S = f_B \wedge f_C$$



$$\bigwedge_{i=1}^n q_i$$

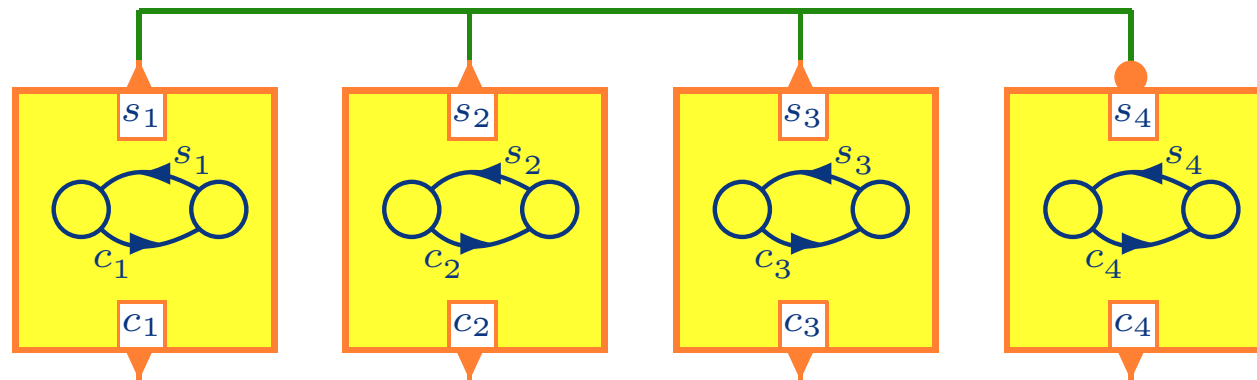
Complexity of the protocol:

- Conjunction $f_S \wedge \bigwedge_{i=1}^n q_i : O(|f_S|)$ (sic!)
- Selecting $a \models \dots : O(n)$, where $n = |P \cup Q|$.

Presentation outline

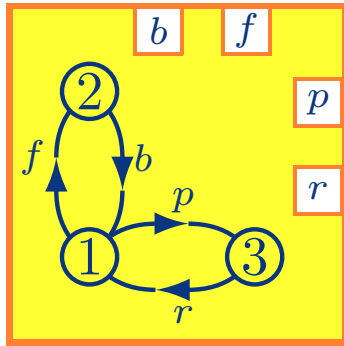
- BIP
- Connectors
- Boolean encoding
- Benchmarks

Benchmarks: Bus

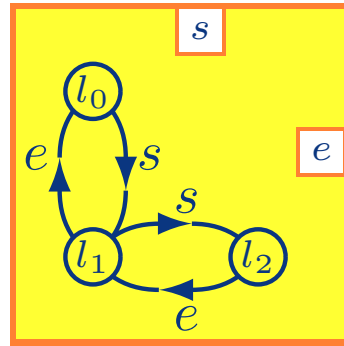


- N clusters
- “Sparse” connectors: $5N$
- Favours enumerative engine

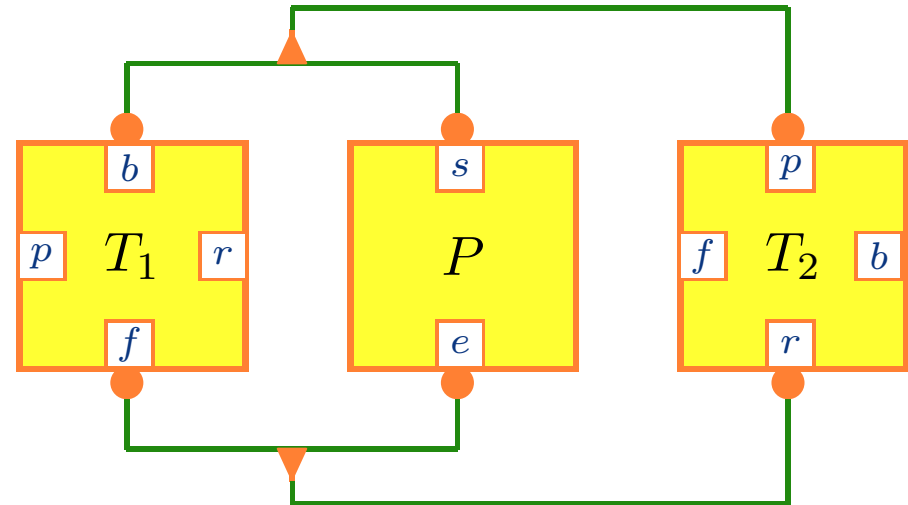
Benchmarks: Preemptable tasks



Task



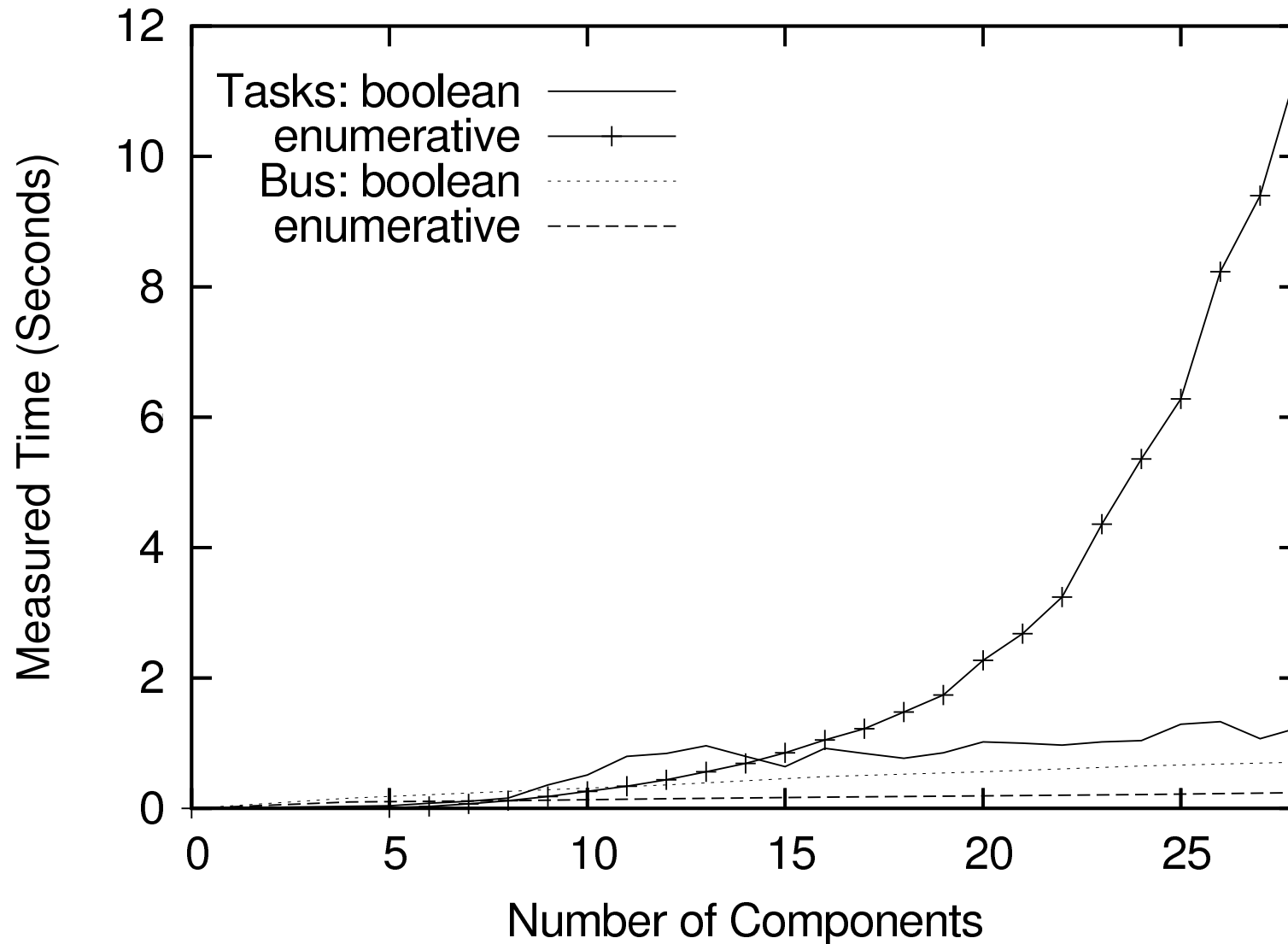
Processor



Connectors

- N Tasks, 4 Processors
- “Dense” connectors: $8N(N - 1)$
- Favours symbolic engine

Benchmarks: Simulation results



Conclusion

- Symbolic engine implementation outperforms the enumerative one for “densly” connected systems.
 - How to characterize “dense”?
- BDD size linear in the number of components for both benchmarks.
 - Why? What about the general case?
 - Further optimizations, e.g. reordering components?
- Only one of the applications of the boolean representation:
 - Connector manipulation
 - Glue (connector & priority) synthesis



Thank you for your attention!

The Algebra of Interactions $AI(P)$

Syntax: $x ::= 0 \mid 1 \mid p \mid x \cdot x \mid x + x \mid (x)$, with $p \in P$

Axioms:

- + union idempotent, associative, commutative, identity 0
- synchronisation idempotent, associative, commutative, identity 1, absorbing 0
 distributes over union

Examples: $s + s r_1 + s r_2 + s r_1 r_2 = s(1 + r_1)(1 + r_2)$ broadcast
 $s + s r_1 + s r_1 r_2 = s(1 + r_1(1 + r_2))$ causality chain



The Algebra of Interactions $\mathcal{AI}(P)$

Semantics: defined by the function $\| \cdot \| : \mathcal{AI}(P) \rightarrow 2^{2^P}$

$$\|0\| = \emptyset,$$

$$\|1\| = \{\emptyset\},$$

$$\|p\| = \left\{ \{p\} \right\}, \text{ for any } p \in P,$$

$$\|x_1 + x_2\| = \|x_1\| \cup \|x_2\|, \text{ for any } x_1, x_2 \in \mathcal{AI}(P),$$

$$\|x_1 \cdot x_2\| = \left\{ a_1 \cup a_2 \mid a_1 \in \|x_1\|, a_2 \in \|x_2\| \right\}, \text{ for any } x_1, x_2 \in \mathcal{AI}(P).$$

The Algebra of Connectors $AC(P)$

Syntax:

$s ::= [0] \mid [1] \mid [p] \mid [x] \quad (\textit{synchrons})$

$t ::= [0]' \mid [1]' \mid [p]' \mid [x]' \quad (\textit{triggers})$

$x ::= s \mid t \mid x \cdot x \mid x + x \mid (x)$

Operators:

$+$ union idempotent, associative, commutative, identity $[0]$

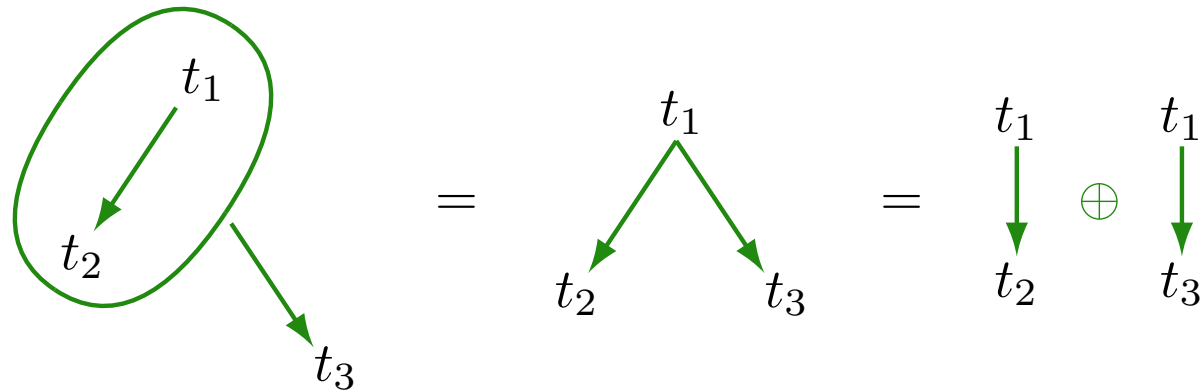
\cdot fusion idempotent, associative, commutative, identity $[1]$
distributes over union ($[0]$ is **not** absorbing)

$[\cdot], [\cdot]'$ typing (often denoted $[\cdot]^\alpha$ for some trigger/synchron typing α)

Semantics: is given by a function $|\cdot| : AC(P) \rightarrow AI(P)$.

$$|p'qr| \stackrel{def}{=} p(1+q)(1+r)$$

The Algebra of Causal Interaction Trees



Syntax: $t ::= a \mid t \rightarrow t \mid t \oplus t.$

Essential axioms:

$$\begin{aligned} (t_1 \rightarrow t_2) \rightarrow t_3 &= t_1 \rightarrow (t_2 \oplus t_3), \\ t_1 \rightarrow (t_2 \oplus t_3) &= t_1 \rightarrow t_2 \oplus t_1 \rightarrow t_3, \\ (t_1 \oplus t_2) \rightarrow t_3 &= t_1 \rightarrow t_3 \oplus t_2 \rightarrow t_3. \end{aligned}$$

Semantics: $|\cdot| : CT(P) \rightarrow AI(P)$

$$\begin{aligned} |a| &= a, \\ |a \rightarrow t| &= a(1 + |t|), \\ |t_1 \oplus t_2| &= |t_1| + |t_2| + |t_1||t_2| \end{aligned}$$

