The Algebra of Connectors — Structuring Interaction in BIP

Simon Bliudze, Joseph Sifakis  
{bliudze, sifakis}@imag.fr  
VERIMAG, Grenoble
Presentation outline

• Overview of BIP
• Interactions and Connectors
• The Algebra of Connectors
• Applications
• Conclusion
Motivation

• Develop a unified compositional framework for describing and analysing the interaction between components
  – interaction and system architecture — first class entities
  – minimal set of constructs and principles
  – tangible, well-founded, and organised concepts
    (instead of using dispersed mechanisms such as semaphores, monitors, message passing, remote call etc.)
  – full separation of concerns: computation and coordination

• Encompass different kinds of heterogeneity
  – strong and weak synchronisation,
  – synchronous and asynchronous execution.

• Provide automated support for component integration and generation of glue code.
Basic model of BIP

Layered component model

- **Behaviour** — labelled transition systems with communication ports
- **Interaction** — set of interactions (interaction = set of ports)
- **Priorities** — order on interactions
Rendezvous

Priorities: $\emptyset$
Interactions: $sr_1r_2r_3$
Broadcast

Priorities: $x \prec xy$ for $xy$ interactions

Interactions: $s, sr_1, sr_2, sr_3, sr_1r_2, sr_1r_3, sr_2r_3, sr_1r_2r_3$
Atomic broadcast

Priorities: $x \prec xy$ for $xy$ interactions

Interactions: $s, sr_1r_2r_3$
Causality chain

Priorities: $x \prec xy$ for $xy$ interactions

Interactions: $s, sr_1, sr_1r_2, sr_1r_2r_3$
Modulo-8 counter

Priorities: $x \prec xy$ for $xy$ interactions

Interactions: $a, abc, abcde, abcdef$
A family of atomic components
\( \{B_i\}_{i=1}^n \) with \( B_i = (Q_i, 2^{P_i}, \rightarrow_i) \)

A set of interactions \( \gamma \in 2^{2^P} \)
\( \sim \leadsto \pi \gamma(B_1, \ldots, B_n) \) — product automaton

A strict partial order \( \pi \) on \( 2^P \)
\( \mathcal{P} = \bigcup_{i=1}^n P_i \)

**Interactions** (\( n \)-ary strong synchronisation)

\[
a \in \gamma \quad \land \quad \forall i \in I, \quad q_i \xrightarrow{a \cap P_i}_i q'_i
\]

\[
(q_1, \ldots, q_n) \xrightarrow{a}_\gamma (q'_1, \ldots, q'_n)
\]

**Priorities**

\[
q \xrightarrow{a}_\gamma q' \quad \land \quad \exists a' : (a \prec a' \land q \xrightarrow{a'}\gamma)
\]

\[
q \xrightarrow{a}_\pi q'
\]

Other parallel composition operators (e.g. CCS, CSP) can be expressed in BIP.
Presentation outline

- Overview of BIP
- Interactions and Connectors
- The Algebra of Connectors
- Applications
- Conclusion
The Algebra of Interactions $\mathcal{A}\mathcal{I}(P)$

**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 $0$
- $\cdot$ 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

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

\[
\begin{align*}
\|0\| & = \emptyset, \\
\|1\| & = \{\emptyset\}, \\
\|p\| & = \{\{p\}\}, \text{ for any } p \in P, \\
\|x_1 + x_2\| & = \|x_1\| \cup \|x_2\|, \text{ for any } x_1, x_2 \in \mathcal{A}\mathcal{I}(P), \\
\|x_1 \cdot x_2\| & = \left\{ a_1 \cup a_2 \mid a_1 \in \|x_1\|, a_2 \in \|x_2\| \right\}, \text{ for any } x_1, x_2 \in \mathcal{A}\mathcal{I}(P).
\end{align*}
\]
Correspondence with boolean functions

<table>
<thead>
<tr>
<th>$\mathcal{AI}(P)$</th>
<th>$\mathbb{B}[P]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td><em>false</em></td>
</tr>
<tr>
<td>1</td>
<td>$\overline{p}\overline{q}$</td>
</tr>
<tr>
<td>$p+1$</td>
<td>$q+1$</td>
</tr>
<tr>
<td>$p+q+1$</td>
<td>$pq+p+1$</td>
</tr>
<tr>
<td>$p\overline{q}$</td>
<td>$\overline{p}\overline{q}\lor p\overline{q}$</td>
</tr>
<tr>
<td>$\overline{p}\lor \overline{q}$</td>
<td>$p\lor \overline{q}$</td>
</tr>
<tr>
<td>$true$</td>
<td></td>
</tr>
</tbody>
</table>

Boolean function representation depends on the set $P$:

$\mathcal{AI}(P) : \quad pq$, if $P = \{p, q\}$

$\quad pq\overline{r}\overline{s}$, if $P = \{p, q, r, s\}$

Synchronisation in $\mathcal{AI}(P)$ is represented by simple concatenation
Interaction modelling: Basic connectors

- A **connector** is a set of ports which can be involved in an interaction.
- Port attributes (trigger $\triangle$, synchron $\circledcirc$) 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.

![Diagram](image-url)
Interaction modelling: Flat connectors

Rendezvous

Priorities: $\emptyset$

Interactions: $sr_1r_2r_3$

Broadcast

Priorities: $x \prec xy$ for $xy$ interactions

Interactions: $s, sr_1, sr_2, sr_3, sr_1r_2, sr_1r_3, sr_2r_3, sr_1r_2r_3$

\[
\left/ s(1 + r_1)(1 + r_2)(1 + r_3) \right/
\]
Interaction modelling: Hierarchical connectors

Atomic broadcast

Priorities: \( x \prec xy \) for \( xy \) interactions

Interactions: \( s, sr_1 r_2 r_3 \)

\[
\text{\( s \) \hspace{1cm} \( r_1 \) \hspace{1cm} \( r_2 \) \hspace{1cm} \( r_3 \)}
\]

\[
\frac{1}{s(1 + r_1 r_2 r_3)}
\]

Causality chain

Priorities: \( x \prec xy \) for \( xy \) interactions

Interactions: \( s, sr_1, sr_1 r_2, sr_1 r_2 r_3 \)

\[
\text{\( s \) \hspace{1cm} \( r_1 \) \hspace{1cm} \( r_2 \) \hspace{1cm} \( r_3 \)}
\]

\[
\frac{1}{s \left(1 + r_1 \left(1 + r_2 \left(1 + r_3\right)\right)\right)}
\]
Presentation outline

- Overview of BIP
- Interactions and Connectors
- The Algebra of Connectors
- Applications
- Conclusion
The Algebra of Connectors $AC(P)$

Syntax:

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

$t ::= [0]' \mid [1]' \mid [p]' \mid [x]'$ \hspace{1cm} (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 $\alpha$)

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

$$|p'qr|^{def} = p(1 + q)(1 + r)$$
Examples

Rendezvous

Broadcast

Atomic broadcast

Causality chain
Examples: Fusion vs. Typing

Consider two connectors: $x = p'_1 p_2$ and $y = p'_3 p_4$.

Fusion

```
xy = p'_1 p_2 p'_3 p_4
```

“Typed fusion”

```
[x]'[y] = [p'_1 p_2]' [p'_3 p_4]
```

\[
xy \quad \leadsto \quad p_1 + p_1 p_2 + p_1 p_3 + p_1 p_4 + p_1 p_2 p_3 + p_1 p_2 p_4 + p_1 p_3 p_4 + p_3 + p_2 p_3 + p_3 p_4 + p_2 p_3 p_4 + p_1 p_2 p_3 p_4
\]

\[
[x]'[y] \quad \leadsto \quad p_1 + p_1 p_2 + p_1 p_3 + p_1 p_2 p_3 + p_1 p_3 p_4 + p_1 p_2 p_3 p_4
\]
The axioms of $\mathcal{AC}(P)$: Typing

Axioms for **typing** (for arbitrary typing $\alpha, \beta$):

1. $[0]' = [0]$,
2. $[[x]^{\alpha}]^{\beta} = [x]^\beta$,
3. $[x+y]^\alpha = [y]^\alpha + [x]^\alpha$,
4. $[x]' \cdot [y]' = [x]' \cdot [y] + [x] \cdot [y]'$.

Fusion of typed connectors is not associative, e.g.

$$[p \cdot q] r \neq p [q \cdot r]$$
Equivalence of connectors

\[ x \simeq y \overset{\text{def}}{\iff} |x| = |y|, \text{ i.e. they represent the same sets of interactions.} \]

- The axiomatisation of \( AC(P) \) is semantically sound, i.e. \( x = y \Rightarrow x \simeq y \).
- Semantic equivalence is not a congruence (not preserved by fusion)

\[
p + pq \simeq p'q, \quad \text{but} \quad pr + pqr \not\simeq p'qr \overset{\simeq}{\Rightarrow} p + pq + pr + pqr,
\]

\[
p[qr] \simeq [pq]r, \quad \text{but} \quad s'p[qr] \not\simeq s'[pq]r.
\]
Congruence of connectors

$\cong$ is the largest congruence contained in $\simeq$

- A criterion to infer congruence from equivalence is available.
- Similarly typed semantically equivalent elements are congruent, i.e. for any two connectors $x, y \in AC(P)$, and any typing $\alpha$, we have
  \[ x \simeq y \implies [x]^\alpha \cong [y]^\alpha. \]
- Zero is not absorbing: $[x]'[0] \cong [x]'$.
- Congruent normal form: the union of all triggered, maximal, and potential interactions
  \[
  \begin{align*}
  p'q & \cong p' + [pq]' + 0q \simeq p + pq \\
  pqr & \cong p'r + [pq]'r + 0qr \simeq p + pq + pr + pqr \\
  pqr' & \cong p'r' + [pq]'r' + 0qr' \simeq p + pq + pr + pqr + r + qr
  \end{align*}
  \]
Presentation outline

- Overview of BIP
- Interactions and Connectors
- The Algebra of Connectors
- Applications
- Conclusion
Incrementality & transformation of connectors

Incremental construction

\[ s r_1 r_2 r_3 \cong s r_1 r_2 r_3 \cong s r_1 r_2 r_3 \]

Transformation (separate one port in a connector)

\[ s r_1 \ldots r_n \cong s r_1 \ldots r_n \cong s r_1 \ldots r_n \]

\[ p q r \cong p q r \cong p q r \]
Connector synthesis: Modulo-8 counter

**Multi-shot semantics:** several connectors can be fired simultaneously

Interactions: \( a + abc + abcde + abcdef \)

**One-shot semantics:** one connector can be fired at a time

Connector synthesis: \([a + ab]' [c + cd]' [e + ef]' \cap a'[bc'][de]'f' \simeq a'[bc]'[de]'f\).
• Overview of BIP
• Interactions and Connectors
• The Algebra of Connectors
• Applications
• Conclusion
The BIP framework: Implementation

BIP program

BIP MetaModel

model transformations

static composition
timed vs. untimed

code generation

centralised or distributed execution,
guided or exhaustive simulation

BIP C++ Code

BIP/Linux Platform

BIP C Code

BIP/Think Platform

compiler
deadlock detection
invariant generation

structural analysis

eclipse

BIP Model
Conclusion

The algebra of connectors

- Allows compact and structured description and analysis of interactions, in terms of two operators admitting a very intuitive interpretation: typing and fusion.

- Not a process algebra: these do not study interactions as such, but only as means to compose behaviour.

- Provides basis for the symbolic comparison, transformation, and synthesis of connectors, which can be directly implemented.

- Boolean representation provides powerful techniques for manipulation, implementation, and synthesis.

- Application in BIP to model interaction in non trivial case studies, e.g. TinyOS-based wireless networks, autonomous robot software.

http://www-verimag.imag.fr/~async/BIP/bip.html
Perspectives

- Causal semantics for $AC(P)$
- Temporised connectors
- Algebraic approach to constraints and priorities
- Application to other formalisms encompassing event-based interaction, in particular coordination languages
- Efficient implementation in future versions of BIP.
Annex: Correspondence with boolean functions

A mapping $\beta : \mathcal{AI}(P) \rightarrow \mathbb{B}[P]$

\[ \beta(0) = \text{false}, \]
\[ \beta(p_{i_1} \cdots p_{i_k}) = \bigwedge_{j=1}^{k} p_{i_j} \land \bigwedge_{i \not\in \{i_j\}} \overline{p_i}, \text{ for any } p_{i_1}, \ldots p_{i_k} \in P, \]
\[ \beta(x + y) = \beta(x) \lor \beta(y), \text{ for any } x, y \in \mathcal{AI}(P), \]
\[ \beta(1) = \bigwedge_{p \in P} \overline{p}, \]
Annex: Semantics of $\mathcal{AC}(P)$

The semantics of $\mathcal{AC}(P)$ is given by the function $|\cdot| : \mathcal{AC}(P) \rightarrow \mathcal{AI}(P)$, defined by the rules

$$
|[p]| = p, \text{ for } p \in P \cup \{0, 1\}
$$

$$
|x_1 + x_2| = |x_1| + |x_2|,
$$

$$
\left| \prod_{i=1}^{n} [x_i] \right| = \prod_{i=1}^{n} |x_k|,
$$

$$
\left| \prod_{i=1}^{n} [x_i] \cdot \prod_{j=1}^{m} [y_j] \right| = \sum_{i=1}^{n} |x_i| \cdot \left( \prod_{k \neq i} (1 + |x_k|) \cdot \prod_{j=1}^{m} (1 + |y_j|) \right),
$$

for $x, x_1, \ldots, x_n, y_1, \ldots, y_m \in \mathcal{AC}(P)$. 
Annex: Characterisation of $\simeq$

**Th 1** For $x, y \in AC(P)$, we have

$$x \simeq y \iff \begin{cases} x \simeq y \\ x \cdot 1' \simeq y \cdot 1' \\ \#x > 0 \iff \#y > 0. \end{cases}$$

$\#x$ is the number of top-level triggers in $x$. 
Annex: Full axiomatisation of $\simeq$

For $x, y, z \in AC(P)$ and arbitrary typing $\alpha, \beta$:

**Basic:**

1. $[0]' = [0]$,
2. $\left([x]^\alpha\right)^\beta = [x]^\beta$,
3. $[x + y]^\alpha = [y]^\alpha + [x]^\alpha$,

**Additional:**

1. $[x]' [0] = [x]'$,
2. $\left([x] [0]\right) = [0]$,
3. $[x]' + [x] = [x]'$,
4. $[x]' [y]' = [x]' [y] + [y]'$,
5. $[x]' [y] [z] = [x]' \left([y]' [z]\right)'$,
6. $[x]' [y] = \left([x]' [y]\right) + [0] [y]$,
7. $[x] [y] = \left([x] [y]\right) + [0] [x] [y]$,
8. $\left([x]' [y]\right) = [x] + \left([x] [y]\right)$,
9. $\left([x] [y] [z]\right) = \left([x] [y] [z]\right)$.