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

\[ A.x := \max(B.y, C.z) \]
Unbuffered synchronous communication

(Not to confuse with synchronous \textit{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

- Synthesis of glue for certain safety properties
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: \( \{p, pqr, pqrst, pqrstu\} \).

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

Priorities: ∅

Interactions: sr₁r₂r₃

Broadcast

Priorities: x ≺ xy for xy interactions

Interactions: s, sr₁, sr₂, ..., sr₁r₂r₃
Interaction modelling: Hierarchical connectors

Atomic broadcast

Causality chain

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

Interactions: $s, sr_1 r_2 r_3$

The Algebra of Connectors, $AC(P)$, introduced in [EMSOFT’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:

\[
p_1 p_2 \iff \begin{pmatrix} p_1 & p_2 & p_3 & p_4 \\ 1 & 1 & 0 & 0 \end{pmatrix}
\]

- Based on the connectors, priorities, and behaviour of components, compute a boolean formula \( \varphi \) 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

Rendezvous

Broadcast

Atomic broadcast

Causal chain

Transformations are straightforward in both ways.

The Algebra of Causal Trees, $CT(P)$, introduced in [FMCO’07].
Boolean representation of connectors

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

\[
\begin{align*}
 & p \\
 & q \\
 & r \\
 & q \quad qs
\end{align*}
\]

Causal trees

Causal rules

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

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

The corresponding boolean formula is

\[
(\text{true} \Rightarrow p) \land (q \Rightarrow p) \land (r \Rightarrow pq) \land (s \Rightarrow pq) \equiv pq \lor p\overline{r}s.
\]
**Boolean representation of atomic components**

\[
 f_{B_1} = l_1 \overline{l_2} p \overline{q} \lor \overline{l_1} l_2 pq \lor \overline{p} \overline{q}
\]

\[
 f_B = (l_1 \overline{l_2} p \overline{q} \lor \overline{l_1} l_2 pq \lor \overline{p} \overline{q}) \land (l_3 \overline{l_4} r \overline{s} \lor \overline{l_3} l_4 rs \lor \overline{r} \overline{s}) \land (l_5 \overline{l_6} t \overline{u} \lor \overline{l_5} l_6 tu \lor \overline{t} \overline{u})
\]

\[
 f_C = (p \lor qr \lor st \lor u) \land (q = r) \land (s = t)
\]
Implementation: Symbolic engine*

Initialization:

0. The engine precomputes functions
\[ f_B \in \mathbb{B}[\bigcup_{i=1}^n Q_i, P] \quad \text{— behaviour,} \]
\[ f_C \in \mathbb{B}[P] \quad \text{— connectors,} \]
\[ f_S = f_B \land f_C. \]

Engine protocol:

1. Atoms notify the engine of their current states \( q_i \).
2. The engine picks any valuation \( a \) on \( P \), such that
\[ (a, q) \models f_S \land \bigwedge_{i=1}^n q_i \]
3. Notifies the atoms.

* Connectors only (for priorities see in the paper).
Complexity of the protocol:

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

- BIP
- Connectors
- Boolean encoding
- Benchmarks
Benchmarks: Bus

- \( N \) clusters
- “Sparse” connectors: 5\( N \)
- Favours enumerative engine
Benchmarks: Preemptable tasks

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

![Simulation Results Graph]

- **Tasks:** boolean enumerative
- **Bus:** boolean enumerative

**Measured Time (Seconds)** vs **Number of Components**

ICE’09, Bologna, August 31, 2009

M. Jaber, A. Basu, S. Bliudze “Symbolic Implementation...”
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!
Syntax: \[ x ::= 0 \mid 1 \mid p \mid x \cdot x \mid x + x \mid (x), \quad \text{with } p \in P \]

Axioms:

- union \quad \text{idempotent, associative, commutative, identity } 0
- synchronisation \quad \text{idempotent, associative, commutative, identity } 1, \text{ absorbing } 0
  \text{distributes over union}

Examples:

\[
\begin{align*}
  s + s r_1 + s r_2 + s r_1 r_2 &= s(1 + r_1)(1 + r_2) \quad \text{broadcast} \\
  s + s r_1 + s r_1 r_2 &= s(1 + r_1(1 + r_2)) \quad \text{causality chain}
\end{align*}
\]
The Algebra of Interactions $AI(P)$

**Semantics:** defined by the function $\| \cdot \| : AI(P) \to 2^P$

\[
\begin{align*}
\|0\| &= \emptyset, \\
\|1\| &= \{\emptyset\}, \\
\|p\| &= \{\{p\}\}, \text{ for any } p \in P, \\
\|x_1 + x_2\| &= \|x_1\| \cup \|x_2\|, \text{ for any } x_1, x_2 \in AI(P), \\
\|x_1 \cdot x_2\| &= \left\{ a_1 \cup a_2 \mid a_1 \in \|x_1\|, a_2 \in \|x_2\| \right\}, \text{ for any } x_1, x_2 \in AI(P).
\end{align*}
\]
The Algebra of Connectors \( AC(P) \)

Syntax:

\[
\begin{align*}
    s &::= [0] | [1] | [p] | [x] \quad (\text{synchrons}) \\
    t &::= [0]' | [1]' | [p]' | [x]' \quad (\text{triggers}) \\
    x &::= s | t | x \cdot x | x + x | (x)
\end{align*}
\]

Operators:

\[
\begin{align*}
    + & \quad \text{union} \quad \text{idempotent, associative, commutative, identity} \ [0] \\
    \cdot & \quad \text{fusion} \quad \text{idempotent, associative, commutative, identity} \ [1] \\
    & \quad \text{distributes over union} \ ([0] \text{ is not absorbing})
\end{align*}
\]

\[
\begin{align*}
    [:], [:]' & \quad \text{typing} \quad \text{(often denoted} [\cdot]^{\alpha} \text{ for some trigger/synchron typing } \alpha)
\end{align*}
\]

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

\[
|p'qr| \overset{def}{=} p(1 + q)(1 + r)
\]
The Algebra of Causal Interaction Trees

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

Essential axioms:

$$(t_1 \rightarrow t_2) \rightarrow t_3 = t_1 \rightarrow (t_2 \oplus t_3),$$

$$t_1 \rightarrow (t_2 \oplus t_3) = t_1 \rightarrow t_2 \oplus t_1 \rightarrow t_3,$$

$$(t_1 \oplus t_2) \rightarrow t_3 = t_1 \rightarrow t_3 \oplus t_2 \rightarrow t_3.$$