Architectural design: the coordination perspective

José Proença
HASLab - INESC TEC & UM
Arquitectura e Cálculo 2015-16
Reo semantics

Jongmans and Arbab 2012

Overview of Thirty Semantic Formalisms for Reo
Reo semantics

- **Coalgebraic models**
  - Timed data streams
  - Record streams

- **Coloring models**
  - Two colors
  - Three colors
  - Tile models

- **Other models**
  - Process algebra
  - Constraints
  - Petri nets & intuitionistic logic
  - Unifying theories of programming
  - Structural operational semantics

- **Operational models**
  - Constraint automata
  - Variants of constraint automata
    - Port automata
    - Timed Probabilistic
    - Continuous-time
    - Quantitative
    - Resource-sensitive timed
    - Transactional
  - Context-sensitive automata
    - Büchi automata
    - Reo automata
    - Intentional automata
    - Action constraint automata
    - Behavioral automata
2CM : Coloring models with two colors [28, 29, 33]
3CM : Coloring models with three colors [28, 29, 33]
ABAR : Augmented BAR [39, 40]
ACA : Action CA [46]
BA : Behavioral automata [61]
BAR : Büchi automata of records [38, 40]
CA : Constraint automata [10, 17]
CASM : CA with state memory [60]
CCA : Continuous-time CA [18]
Constr. : Propositional constraints [30, 31, 32]
GA : Guarded automata [20, 21]
IA : Intentional automata [33]
ITLL : Intuitionistic temporal linear logic [27]
LCA : Labeled CA [44]
mCRL2 : Process algebra [47, 48, 49]

PA : Port automata [45]
PCA : Probabilistic CA [15]
QCA : Quantitative CA [12, 53]
QIA : Quantitative IA [13]
RS : Record streams [38, 40]
RSTCA : Resource-sensitive timed CA [51]
SGA : Stochastic GA [56, 57]
SOS : Structural operational semantics [58]
SPCA : Simple PCA [15]
TCA : Timed CA [8, 9]
TDS : Timed data streams [4, 5, 14, 62]
Tiles : Tile models [11]
TNCA : Transactional CA [54]
UTP : Unifying theories of programming [55, 52]
ZSN : Zero-safe nets [27]
# Outline

<table>
<thead>
<tr>
<th>Formalism</th>
<th>Synchr.</th>
<th>Data</th>
<th>Context</th>
<th>Partial</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connector Colouring</td>
<td>CC2</td>
<td>-</td>
<td>CC3</td>
<td>-</td>
</tr>
<tr>
<td>Automata</td>
<td>Port Automata</td>
<td>Constraint Automata</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Constraints</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
Outline

- Intuitive & visual
- Compilation & verification
- Runtime / scalability
Reo Connector Colouring

Dave Clarke, David Costa, and Farhad Arbab. Connector colouring I: Synchronisation and context dependency
Behaviour?

merger: data flows from one of the source ends to the sink end

lossy-sync: either data flows from the source to the sink end, OR it is lost

FIFO-1: data flows from the source end to the buffer, becoming a FIFOFull-1

FIFOFull-1: data flows from the buffer to the sink buffer, becoming a FIFO-1
Colourings to describe synchronous dataflow
Colouring composition

Colours match in nodes
Colouring semantics (CC2)

- **Colouring**: End $\rightarrow$ \{Flow, NoFlow\}
- **Colouring table**: Set(Colouring)
- **Composition** = matching colours
- More visual (intuitive)
- Used for generating animations
Colouring semantics (CC2)

- **Colouring**: \( \text{End} \rightarrow \{\text{Flow, NoFlow}\} \)

- **Colouring table**: \( \text{Set(Colouring)} \)

- **Composition** = matching colours

\[
CT_1 \Join CT_2 = \{ cl_1 \Join cl_2 \mid cl_1 \in CT_1, cl_2 \in CT_2, cl_1 \prec cl_2 \}
\]

\[
cl_1 \prec cl_2 = \forall e \in \text{dom}(cl_1) \cap \text{dom}(cl_2) \cdot cl_1(e) = cl_2(e)
\]

\[
cl_1 \Join cl_2 = cl_1 \cup cl_2
\]
Exercise: compose colouring tables

[Diagram with arrows and labels indicating composition of tables]
Reo Connector Colouring

Dave Clarke, David Costa, and Farhad Arbab. Connector colouring I: Synchronisation and context dependency

(on connector colouring)
Port and Constraint Automata

Christel Baier, Marjan Sirjani, Farhad Arbab, Jan Rutten. Modeling Component Connectors in Reo by Constraint Automata. 2004

Christian Koehler and Dave Clarke. Decomposing Port Automata. 2009
Connector behaviour (stateful)

• Dataflow behaviour is discrete in time: it can be observed and snapshots taken at a pace fast enough to obtain (at least) a snapshot as often as the configuration of the connector changes.

• At each time unit the connector performs an evaluation step: it evaluates its configuration and according to its interaction constraints changes to another (possibly different) configuration.

• A connector can fire multiple ports in the same evaluation step.
Port Automata

\[ A = (Q, N, \rightarrow, Q_0) \]

- \( Q \) set of states
- \( N \) a set of ports \( N \)
- \( \rightarrow \subseteq Q \times 2^N \times Q \) a transition relation
- \( Q_0 \subseteq Q \) a set of initial states

Transitions must have a non-empty set of ports!

Examples:

\[ q_e \xrightarrow{a} q_f \]
\[ q_L \xrightarrow{ab} \]
\[ a \rightarrow b \]
\[ a \rightarrow b \]
Composing steps

\[
\begin{align*}
\text{Composing steps} & \\
\text{Diagram showing transitions and states.} & \\
\text{Examples include:} & \\
\text{ac, bc} & \\
\text{acd, bcd} & \\
\text{e, ace, bce} & \\
\end{align*}
\]
Composing steps

\[ ac \otimes cd \otimes d = acd \]
\[ ac \otimes c \otimes d = \perp \]
**Composition - formally**

*Definition 2.* The product of two port automata $A_1 = (Q_1, N_1, \rightarrow_1, Q_0, 1)$ and $A_2 = (Q_2, N_2, \rightarrow_2, Q_0, 2)$ is defined by

$$A_1 \Join A_2 = (Q_1 \times Q_2, N_1 \cup N_2, \rightarrow, Q_0, 1 \times Q_0, 2)$$

where $\rightarrow$ is defined by the rule

$$\begin{align*}
q_1 \xrightarrow{N_1} p_1 & \quad q_2 \xrightarrow{N_2} p_2 \\
\langle q_1, q_2 \rangle \xrightarrow{N_1 \cup N_2} \langle p_1, p_2 \rangle
\end{align*}$$

and the following and its symmetric rule

$$\begin{align*}
q_1 \xrightarrow{N_1} p_1 & \quad N_1 \cap N_2 = \emptyset \\
\langle q_1, q_2 \rangle \xrightarrow{N_1} \langle p_1, q_2 \rangle
\end{align*}$$
Formalize and compose

\[ q_1 \xrightarrow{N_1} p_1 \quad q_2 \xrightarrow{N_2} p_2 \quad N_1 \cap N_2 = N_2 \cap N_1 \]
\[ \langle q_1, q_2 \rangle \xrightarrow{N_1 \cup N_2} \langle p_1, p_2 \rangle \]

\[ A = (Q, N, \rightarrow, Q_0) \]

\[ q_1 \xrightarrow{N_1} p_1 \quad N_1 \cap N_2 = \emptyset \]
\[ \langle q_1, q_2 \rangle \xrightarrow{N_1} \langle p_1, q_2 \rangle \]
Examples I

Flow regulator

“b” controls flow from “a” to “c”

data flows from “a” to “b” ONLY if either “c” or “d” have data
Examples II

Synchronising barrier
data flows “a” → “b”
iff
data flows “c” → “d”

Alternator
data flows from “a” and from “b” to “z”, alternating (+ extra synch constraints)
Examples III

N-Alternator

data flows from “a”, “b”, “c”, and “d” to “z”, alternating (+ extra synch constraints)
Examples IV

Data flows from “a” to “d”, “b” to “e”, and “c” to “f” alternating.

Sequencer
Reo in mCRL2

Lossy = (c|d + c).Lossy

Merger = (a|c + b|c).Merger
Reo in mCRL2

Conn = hide(\{c,d\},
    block(\{c_1,c_2,d_1,d_2\},
    comm(\{c_1|c_2 -> c, d_1|d_2 -> d\},
    Merger || Lossy || FIFO1)))
Can you prove?

**colours and port automata provide equivalent semantics**

\[ A(C_1) = (Q_1, \mathcal{N}_1, \rightarrow_1, q_{0,1}) \]

\[ A(C_2) = (Q_2, \mathcal{N}_2, \rightarrow_2, q_{0,2}) \]

\[ \mathcal{T}(C) \text{ – colouring table of } C \]

\[ \text{col}(q \xrightarrow{P} q') \text{ – colouring associated to a transition} \]

\[ (\langle q_{0,1}, q_{0,2} \rangle \xrightarrow{P} \langle q_1, q_2 \rangle) \in A(C_1) \bowtie A(C_2) \]

\[ \Rightarrow \]

\[ \text{col}(\langle q_{0,1}, q_{0,2} \rangle \xrightarrow{P} \langle q_1, q_2 \rangle) \in \mathcal{T}(C_1) \bowtie \mathcal{T}(C_2) \]
Can you prove?
(more generically)

*colourings and port automata provide equivalent semantics*

\[
\mathcal{A} = (Q, \mathcal{N}, \rightarrow, \{q_0\}) \\
(q_0 \xrightarrow{P} q) \in \mathcal{A}(C) \\
\Rightarrow \\
col(P, \mathcal{N}) \in \mathcal{CT}(C)
\]
Constraint Automata

Automata labelled by

• a **data constraint** which represents a set of data assignments to port names

\[ g ::= \text{true} \mid d_A = v \mid g_1 \lor g_2 \mid \neg g \]

**Note:** other constraints, such as

\[ d_A = d_B \overset{\text{abv}}{=} \lor_{d \in \text{Data}}(d_A = d \land d_B = d) \]

are derived.

• a **name set** which represents the set of port names at which IO can occur

States represent the configurations of the corresponding connector, while transitions encode its maximally-parallel stepwise behaviour.
Constraint Automata

Example: FIFO

Figure 11: Constraint automaton for a FIFO

Also it is used for Model checker tool [7], which checks properties like deadlock-freeness and behavioral equivalence of connectors. For Performance analysis Quantitative Intentional Automata are used. It is an extension of Constraint Automata with quantitative properties, such as arrival rates at ports and average delays of data-flows between ports.

6.4 Connector coloring

If we abstract away from which way the data flows and what transformations are done on data, then the semantics of a Reo circuit become the set of all of its dataflow alternatives. Where under “dataflow” we understand mapping of all elements to set of two colors: solid line and dashed line (it means “data flowing” and “no data flowing” respectively).

An example of all possible dataflows in Exclusive router connector is depicted in Figure 12.

Figure 12: Possible data flow behaviour. The solid line marks the part of the connector where data flows synchronously. In unmarked parts no data flows.

The CC model is also used in the implementation of a visualization tool that produces Flash animations depicting the behavior of a connector.

7 Conclusion

In this paper we reviewed basic concepts of Reo coordination language. Basic definitions were provided. Many examples of channels and connectors were also reviewed so the reader gets an impression how easy it is to construct elements with complex behavior in the Reo model. After the practical examples we also looked very briefly at the main semantics models (ABT, Constraint automata...
Constraint Automata - Definition

\[ A = (Q, \mathcal{N}, \rightarrow, Q_0) \]

- \( Q \) is the set of states
- \( \mathcal{N} \) is a set of ports \( \mathcal{N} \)
- \( Q_0 \subseteq Q \) is a set of initial states
- \( \rightarrow \subseteq Q \times 2^{\mathcal{N}} \times DC \times Q \) is a transition relation such that \( \stackrel{P, g}{\rightarrow} \) iff
  1. \( P \neq \emptyset \)
  2. \( g \in DC(P, Data) \)

\( (DC(P, Data) \) is the set of data constraints over Data and P)
Constraint Automata - Definition

\[ s \xrightarrow{P,g} s' \text{ iff} \]
1. \( P \neq \emptyset \)
2. \( g \in DC(P, Data) \)

In configuration \( s \), ports in \( P \) can perform 10 operations which meet guard \( g \) and lead to \( s' \).

Transitions fire only if data occurs at a (set of) ports \( P \).

Behaviour depends only on observed data (not on future evolution).
Constraint Automata as a semantics for Reo

- cannot capture context-awareness [Baier, Sirjani, Arbab, Rutten 2006], but forms the basis for more elaborated models (eg, Reo automata)
- captures all behaviour alternatives of a connector; useful to generate a state-machine implementing the connector’s behaviour
- basis for several tools, including the model checker Vereofy [Kluppelholz, Baier 2007]
Constraint Automata - Reo connectors

Composition as coordination

Introduction to Reo

Examples

Semantics

Further examples

Constraint automata as a semantics for Reo

Examples

4.2

Constraint automata for the basic channels

Figure 7 shows the constraint automata for some of the standard basic channel types: synchronous channels with source \( A \) and sink \( B \) (or vice versa), (asynchronous drain with the sources \( A \), \( B \), (asynchronous spout with the sinks \( A \), \( B \) and lossy synchronous channels with source \( A \) and sink \( B \).

In every case, one single state is sufficient. Moreover, the automata are deterministic.

\( \{A,B\} \quad d_A = d_B \)

synchronous channel

synchronous drain or synchronous spout

\( \{A,B\} \)

lossy synchronous channel

asynchronous drain or asynchronous spout

\( \{A,B\} \quad d_A = d_B \)

\( \{A\} \quad \{B\} \)
Parameterised constraint automata

States are parametric on data values ... therefore capturing complex constraint automata emerging form data-dependencies

Example: 1 bounded FIFO

\[
\begin{aligned}
q_0 & \quad \xrightarrow{\{A\}} \quad q(x) \\
\{A\} & \\
x & := d_A \\
\{B\} & \\
d_B = x 
\end{aligned}
\]
Composing constraint automata

**Definition 4.1** [Product-automaton] The product-automaton of the two constraint automata $\mathcal{A}_1 = (Q_1, \mathcal{N}ames_1, \rightarrow_1, Q_{0,1})$ and $\mathcal{A}_2 = (Q_2, \mathcal{N}ames_2, \rightarrow_2, Q_{0,2})$, is:

$$\mathcal{A}_1 \bowtie \mathcal{A}_2 = (Q_1 \times Q_2, \mathcal{N}ames_1 \cup \mathcal{N}ames_2, \rightarrow, Q_{0,1} \times Q_{0,2})$$

where $\rightarrow$ is defined by the following rules:

$$\frac{\langle q_1, q_2 \rangle \quad N_1 \cup N_2 \cdot g_1 \wedge g_2}{\langle p_1, p_2 \rangle}$$

and

$$\frac{\langle q_1, q_2 \rangle \quad N \cdot g \cdot \mathcal{N}ames_2 \equiv \emptyset}{\langle p_1, q_2 \rangle}$$
<table>
<thead>
<tr>
<th>Formalism</th>
<th>Synchr.</th>
<th>Data</th>
<th>Context</th>
<th>Partial</th>
</tr>
</thead>
<tbody>
<tr>
<td>Connector Colouring</td>
<td>CC2</td>
<td>-</td>
<td>CC3</td>
<td>PCC</td>
</tr>
<tr>
<td>(Process Algebra)</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Automata</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Port Automata</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Constraint Automata</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
</tbody>
</table>
2 reasons for context

1 - avoid data loss when the context (FIFO) can receive the data.
2 reasons for context

2 - give priority based on the context (writer)
Context = 3 colours

- Colouring:
  
  \[\text{End} \rightarrow \{\text{Flow, GiveReason, GetReason}\}\]

- Composition = matching colours:

[Diagram showing matching colour combinations]
Context = 3 colours

• **Colour**

  \[ \text{End} = \{e_1, \ldots, e_n\} \cup \{\overline{e_1}, \ldots, \overline{e_n}\} \]

  \[ \text{End} \rightarrow \{\text{Flow, GiveReason, GetReason}\} \]

• **Composition** = matching colours:

  \[ CT_1 \bowtie CT_2 = \{cl_1 \bowtie cl_2 \mid cl_1 \in CT_1, cl_2 \in CT_2, cl_1 \sim cl_2\} \]

  \[ cl_1 \sim cl_2 = \forall e_1 \in \text{dom}(cl_1) \cdot \forall e_2 \text{dom}(cl_2) \cdot \]

  \[ e_1 = \overline{e_2} \Rightarrow \]

  \[ (cl_1(e), cl_2(e)) \in \{(\text{ dạ, dạ}), (\text{ sáng, sáng}), (\text{ dạ, sáng}), \} \]

  \[ cl_1 \bowtie cl_2 = cl_1 \cup cl_2 \]
Composition
Priority with 3 colours
• **Compositional** – composition operation is associative, commutative, and does not require post-processing.

• *Reasons* for the absence of flow are *propagated*.

• Expresses *priority*.

• 2 colours $\Leftrightarrow$ constraint automata (without data)

• 3 colours: + expressive ($\Leftrightarrow$ intentional automata)
Build a connector

prefer fast FIFO
Build connectors

- $a, b, c, d, e, \ldots$
- $a, b, c, \ldots$
- $d, e, f, \ldots$
- $a, b, c, d, e, \ldots$
- stop

- $a, c, e, \ldots$
- $a, d, b, e, c, f, \ldots$
- $a, b, c, d$
Outline

1. Visual semantics for Reo
   - Connector colouring (CC)\(^1\)

2. Locality (concurrency)
   - Partial connector colouring (PCC)\(^2\)

3. Constraints
   - SAT solving with data for Reo\(^3\)

---

\(^1\) Dave Clarke, David Costa, and Farhad Arbab. Connector colouring I: Synchronisation and context dependency

\(^2\) Dave Clarke and José Proença. Partial connector colouring

\(^3\) Dave Clarke, José Proença, Alexander Lazovik, and Farhad Arbab, Channel-based coordination via constraint satisfaction
José Proença, Dave Clarke, Interactive interaction constraints
Locality (concurrency)
Motivation

• Connector colouring is not optimal for distributed systems.

• All-or-nothing – all channels are needed to decide where data goes.

• Need to identify local flows that are not composed with the full connector.

• Model context dependency
Problems

• 2 colours (or constraint automata):
  ➢ assume primitives can make a no-flow step

• 3 colours:
  ➢ cannot assume primitives have a no-flow colour – which direction would it be?
  ➢ Idea?: add another no-flow colour, without direction, and assume all primitives have it...
Example

CC 2

Partial CC

CC 3
Synchronous regions

Static regions: boundaries = FIFO’s

Dynamic regions: boundaries = GiveReason
Partial connector colouring

• Colouring:
  \[ \text{End} \xrightarrow{\text{Flow, GiveReason, GetReason}} \]

• Composition = matching colours:
In practice

Shall I search now for a colouring?