Generating Connector Laws — Work in Progress —

Dave Clarke (CWI)

Goal: Axiomatize Component Connectors

What are laws of component connectors (*e.g.*, Reo)?

$$\begin{array}{rcl} \mathit{sync};\mathit{fifo1} & = & \mathit{fifo1} \\ \mathit{repl};\mathit{merge} & = & \mathbf{0};\overline{\mathbf{0}} \end{array}$$

Ideal: a finite collection of equational laws which completely characterize some class of component connectors.

A complete axiomatization in terms of equational laws tells us all we need to know about equivalence, and can form basis of verification tools and theorem provers.

Basis: Algebra of *Stateless* Connectors

- Bruni, Lanese & Montanari's *Basic algebra of stateless connectors* present a complete axiomatization of a notion of stateless connector.
- Connectors are approximately Reo connectors without filters or buffers, over a unit data type.
- Semantics are in terms of *tick-tables* which give *synchronization possibilities*.

Tick Tables

The following is a tick-table for a connector of arity $2 \to 1$, that is, two input ports and one output port:

?	0	1
00	$\surd$	
01		$\surd$
10		$\surd$
11		

Ticks denote that the given input/output combination is possible, synchronously.

The $(0, 0)$ entry is always $\surd$ — always possible to do nothing.

Primitives Connectors

Ordinary Structure		Dual Structure	
name	symbolic	name	symbolic
id	$id : 1 \to 1$		
symmetry	$\gamma : 2 \to 2$		
duplicator	$\nabla : 1 \to 2$	**coduplucator**	$\Delta : 2 \to 1$
bang	$! : 1 \to 0$	**cobang**	$¡ : 0 \to 1$
mex	$\dot{\nabla} : 1 \to 2$	**comex**	$\dot{\Delta} : 2 \to 1$
zero	$\mathbf{0} : 1 \to 0$	**cozero**	$\overline{\mathbf{0}} : 0 \to 1$

id is Reo's synchronous channel.
duplicator is the replicate behaviour of Reo's nodes.
comex is the merge behaviour of Reo's nodes.
mex is Reo's exclusive router.
bang is a half-spout. **cobang** is a half-drain.

Semantics of Primitives I

sync

id	0	1
0	$\surd$	
1		$\surd$

symmetry

γ	00	01	10	11
00	$\surd$			
01			$\surd$	
10		$\surd$		
11				$\surd$

duplicate

∇	00	01	10	11
0	$\surd$			
1				$\surd$

coduplicate

Δ	0	1
00	$\surd$	
01		
10		
11		$\surd$

Semantics of Primitives II

bang

$!$	$\emptyset$
0	$\surd$
1	$\surd$

cobang

$¡$	0	1
$\emptyset$	$\surd$	$\surd$

mex

$\dot{\nabla}$	00	01	10	11
0	$\surd$			
1		$\surd$	$\surd$	

comex

$\dot{\Delta}$	0	1
00	$\surd$	
01		$\surd$
10		$\surd$
11		

zero

$\mathbf{0}$	$\emptyset$
0	$\surd$
1	

cozero

$\overline{\mathbf{0}}$	0	1
$\emptyset$	$\surd$	

Operations

- Parallel composition $\otimes$ — shuffle product for matrices

$\overset{\bullet}{\nabla} \otimes ! : 2 \to 2$	00	01	10	11
00	√			
01	√			
10		√	√	
11		√	√	

- Sequential composition ; — matrix multiplication.

¡; $\overset{\bullet}{\Delta}$: $0 \to 1$	00	01	10	11
$\emptyset$	√	√	√	

Axiomatization I

Axioms of a strict symmetric monoidal category. Plus...

The so-called gs-monoidal axioms for ∇ and !:

$$
\begin{aligned}
\nabla; id \otimes ! &= id \\
\nabla; \gamma &= \nabla \\
\nabla; \nabla \otimes id &= \nabla; id \otimes \nabla
\end{aligned}
$$

and their duals (cogs-monoidal structure):

$$
\begin{aligned}
id \otimes \text{¡}; \Delta &= id \\
\gamma; \Delta &= \Delta \\
\Delta \otimes id; \Delta &= id \otimes \Delta; \Delta
\end{aligned}
$$

Axiomatization II

The match-share axioms:

$$\begin{array}{rcl} \nabla;\Delta & = & id \\ \Delta;\nabla & = & id \otimes \nabla;\Delta \otimes id \\ \Delta;\nabla & = & \nabla \otimes id;id \otimes \Delta \end{array}$$

Also new-bang (garbage collection):

$$¡;! \;=\; id_0$$

Axiomatization III

$$
\begin{array}{rcl}
\dot{\nabla};\dot{\Delta} & = & id \\
\nabla;\dot{\Delta} & = & \mathbf{0};\overline{\mathbf{0}} \\
\dot{\Delta};\nabla & = & \nabla_2;\dot{\Delta}\otimes\dot{\Delta} \\
\dot{\Delta};\mathbf{0} & = & \mathbf{0}\otimes\mathbf{0} \\
\nabla;id\otimes\mathbf{0} & = & \mathbf{0};\overline{\mathbf{0}} \\
\Delta;\mathbf{0} & = & \mathbf{0}\otimes\mathbf{0} \\
\dot{\nabla};!_2 & = & ! \\
\nabla;\dot{\nabla}\otimes id & = & \dot{\nabla};\nabla\otimes\nabla;id\otimes\dot{\Delta}\otimes id;id\otimes\gamma \\
\dot{\nabla};\nabla\otimes id & = & \nabla;\dot{\nabla}\otimes\dot{\nabla};id\otimes\Delta\otimes id;id\otimes\gamma
\end{array}
$$

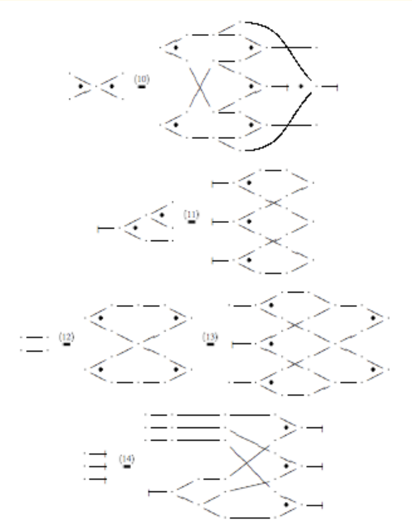
Axiomatization IV

$$\dot{\Delta}; \dot{\nabla} \;=\; \dot{\nabla}_2; \nabla \otimes \nabla \otimes \nabla \otimes \nabla;$$
$$id \otimes \dot{\Delta} \otimes (\dot{\Delta}; !) \otimes \dot{\Delta} \otimes id; \gamma \otimes \gamma; id \otimes (\dot{\Delta}; !) \otimes id$$

$$¡; \dot{\nabla}; \dot{\nabla} \otimes id \;=\; ¡_3; \dot{\nabla} \otimes \dot{\nabla} \otimes \dot{\nabla}; id \otimes \gamma \otimes \gamma \otimes id; \Delta \otimes \Delta \otimes \Delta$$
$$id_2 \;=\; \dot{\nabla} \otimes \dot{\nabla}; id \otimes \Delta \otimes id; id \otimes \nabla \otimes id; \dot{\Delta} \otimes \dot{\Delta}$$
$$id_2 \;=\; \dot{\nabla} \otimes (¡; \dot{\nabla}) \otimes \dot{\nabla}; id \otimes \gamma \otimes \gamma \otimes id;$$
$$\Delta \otimes \Delta \otimes \Delta; id \otimes \nabla \otimes id; \dot{\Delta} \otimes \dot{\Delta}$$
$$!_n \;=\; id_n \otimes ¡; id_n \otimes \nabla^n; \dot{\Delta}_n; !_n$$

Plus their duals. Complete, but not finite, due to last rule.
$\dot{\Delta}_n$ is a tree of $\dot{\Delta}$s.

Axiomatization Visually



Towards an algebra of stateful connectors

- Requires a new model—tick-automata: automata where transitions are tick-tables
- Want to add FIFO1 buffers and loops.
- Need to avoid causality problems due to synchronous loops.

Tick Automaton for a FIFO1: $1 \to 1$

$$
\begin{array}{ccc}
 & \textit{empty} & \textit{full} \\
\textit{empty} &
\begin{array}{c||c|c|}
 & 0 & 1 \\
\hline\hline
0 & \surd & \\
\hline
1 & & \\
\hline
\end{array}
&
\begin{array}{c||c|c|}
 & 0 & 1 \\
\hline\hline
0 & & \\
\hline
1 & \surd & \\
\hline
\end{array}
\\
\textit{full} &
\begin{array}{c||c|c|}
 & 0 & 1 \\
\hline\hline
0 & & \surd \\
\hline
1 & & \\
\hline
\end{array}
&
\begin{array}{c||c|c|}
 & 0 & 1 \\
\hline\hline
0 & \surd & \\
\hline
1 & & \\
\hline
\end{array}
\end{array}
$$

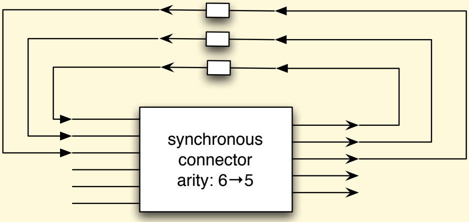
Or
$$
\left(\begin{array}{cc|cc}
1 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 \\
\hline
0 & 1 & 1 & 0 \\
0 & 0 & 0 & 0
\end{array}\right)
$$

FIFOs and Loops

Avoid causality problems: make each loop contains a buffer.

Thus connector consists of a synchronous connector, plus an array of FIFO buffers in the feedback loop.

connector arity: 3→2, with 3 buffers



Conjecture: sufficient to model all *casual connectors*

Generating Connector Laws

- Find laws by automatically generating connectors and testing equivalence.
- Examine results, manually, to find *concise* laws.
- Test independence of laws.

Problems:

- Massive number of connectors: how to generate systematically with little repetition?
- How to extract useful laws?
- Finite axiomatization? When do I stop?

Generating Tick-tables (representing connectors)

Pick 0 or 1 for each entry of the following ($1 \to 1$ and $2 \to 2$):

$$\begin{bmatrix} 0 & 1 \\ 2 & 3 \end{bmatrix} \qquad \begin{bmatrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 \\ 12 & 13 & 14 & 15 \end{bmatrix}$$

Given n inputs, m outputs, and d buffers, there are $2^{2^{n+m+2d}-1}$ combinations (entry 0 is always 1).

1-input, 1-output, 1-buffer has $2^{15} = 32,768$ possibilities.

2-input, 3-output, 0-buffer: $2^{2^5-1} = 2^{31} = 2,147,483,648$.

A Simplification

	00	01	10	11
00	a	b	c	d
01	e	f	g	h
10	i	j	k	l
11	m	n	o	p

$\Longrightarrow$

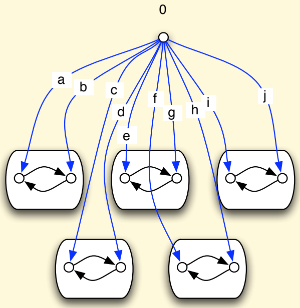
		mt 0	*mt* 1	*full* 0	*full* 1
		00		10	
mt	0	a	b	i	j
	1	e	f	m	n
		01		00	
full	0	c	d	a	b
	1	g	h	e	f

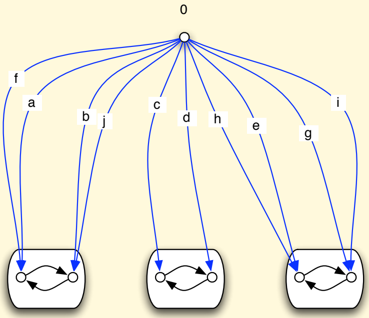
Can generate a smaller number of connectors.

Connectors which are the same except for the values of k, l, o, p form an equivalence class.

Generating and Reducing (Mega)Automata

Generated $\longrightarrow$ Reduced





Tom's Trick! (which is Simona's trick (...)).
Gives equivalence classes.

Two Spaces of Equivalences

Two spaces to explore:

- *obvious* equivalences resulting from tick-tables that differ in only in their fourth quadrant.

- equivalence classes across generated by `ltsmin`.

These equivalence classes can be explored to find laws.

Demo — Steps

1. `generate`: builds mega-automaton for *all* connectors of some given arity and number of FIFOS.
2. `ltsmin`: reduce mega-automaton modulo bisimulation.
3. `separate`: splits reduced mega-automaton into (1) equivalence classes and (2) representative elements.
4. Test candidate rules using Maude. (*manual, incomplete*).
5. If rule passes, add to Maude code.
6. Continue (step 1.) for larger connectors.

References

Bruni, Lanese, Montanari: *A basic algebra of stateless connectors*.

Arbab: *Reo: a channel-based coordination model for component composition*.

Baier, Sirjani, Arbab, Rutten: *Modeling component connectors in Reo by constraint automata.*

Bloom, Sabadini, Walters: *Matrices, Machines, Behaviour*.

Katis, Sabadini, Walters: *Bicategories of Processes*.

Lafont: *Towards an Algebraic Theory of Boolean Circuits*.

Joyal, Street, Verity: *Traced Monoidal Categories*.

μCRL: `http://www.cwi.nl/~mcrl`