Alternative Syntaxes for Ptolemy Models

Chris Shaver
University of California, Berkeley

Block Combinator Syntaxes

- Ptolemy models can be represented in a combinator syntax rather than a point-to-point syntax with named ports.
- Models can be composed, edited, or reasoned about in such a syntax.

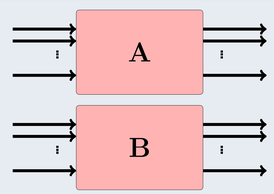
Blocks

$\mathbf{B}_M^N$ signifies the type of a block in the combinator language with N *inputs* and M *outputs*.

Operators

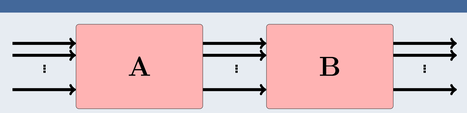
Parallel Composition: $A \otimes B$

$$\otimes : \mathbf{B}_M^N \times \mathbf{B}_P^K \to \mathbf{B}_{M+P}^{N+K}$$



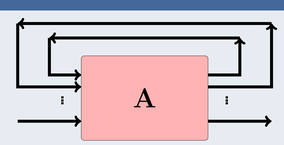
Serial Composition: $A \gg B$

$$\gg : \mathbf{B}_M^N \times \mathbf{B}_N^K \to \mathbf{B}_M^K$$



Feedback Contraction: $\langle A \rangle_n$

$$\langle\rangle_n : \mathbf{B}_M^N \to \mathbf{B}_{M-n}^{N-n}$$



Influences

- The *Faust* signal processing language.
- Diagrammatic Linear Algebra.
- Milner's Calculus of Communicating Systems.

Abstract Syntax

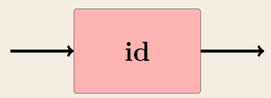
Syntax of block expression (both visual and textual):

- $E =_{syn} \ Name \mid E \otimes E \mid E \gg E \mid \langle E \rangle_{\mathbf{n}}$
- $Bind =_{syn} \ Name \leftarrow E$

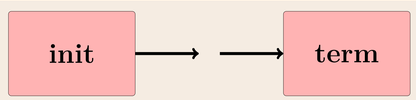
where a $Name$ can be a primitive block or bound to an E.

Primitives

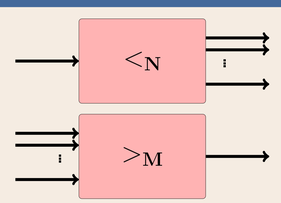
Identity: id



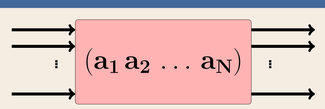
Initiator/Terminator: $init, term$



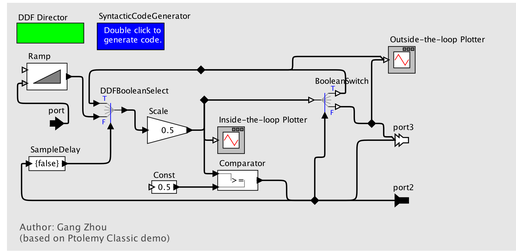
Split/Merge: $<_N, >_M$



Permutation: $(a_1\, a_2\, a_N)$



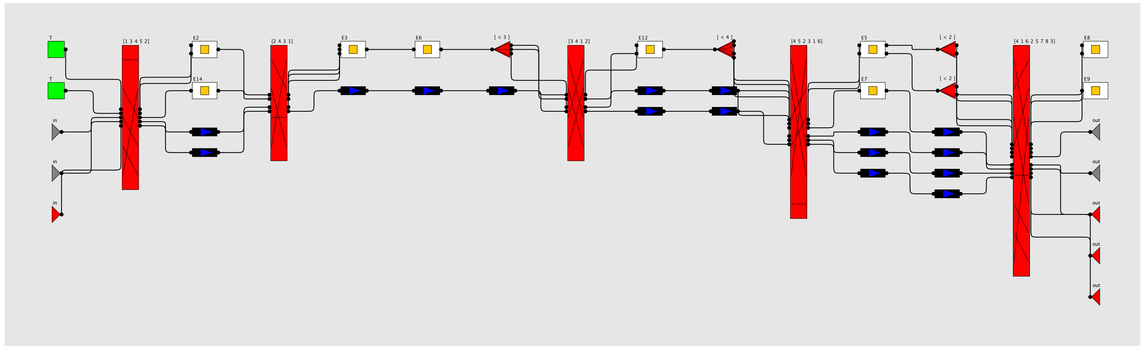
Typical Ptolemy Model



Textual Combinator Form

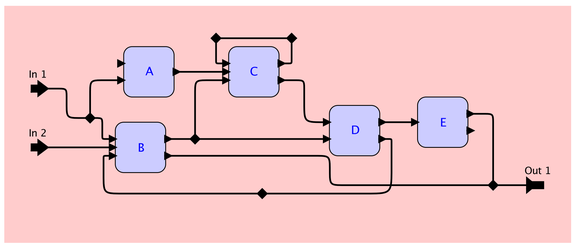
$$Expr_1 = \langle Expr_2 \rangle_2$$
$$Expr_2 = init \otimes init$$
$$\gg (1\,3\,4\,5\,2) \gg E2 \otimes E14$$
$$\gg (2\,4\,3\,1) \gg E3 \gg E6 \gg <_3$$
$$\gg (3\,4\,1\,2) \gg E12 \gg <_4$$
$$\gg (4\,5\,2\,3\,1\,6) \gg E5 \otimes E7 \gg <_2 \otimes <_2$$
$$\gg (4\,1\,6\,2\,5\,7\,8\,3) \gg E8 \otimes E9$$

Visual Combinator Form

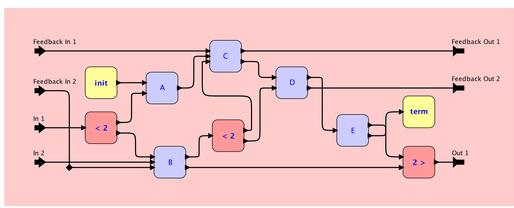


Conversion from Ptolemy to Combinator Form

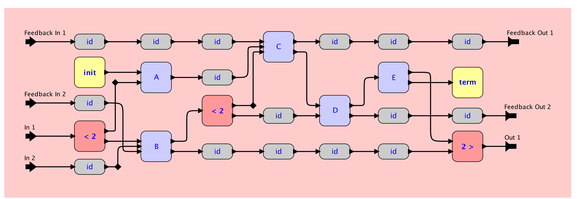
Starting with a Ptolemy model, the point-to-point syntax can be converted into a combinatorial visual syntax.



1. Split/Merge primitives replace multiply connected relations.
2. Initiator/Terminator primitives are added to unconnected ports.
3. Feedback edges are cut and drawn out to input/output pairs.



4. Organize blocks into columns dependent on predecessor columns.
5. Insert identity operators to make columns opaque.



6. Order identities to the bottom of columns.
7. Insert permutation primitives between columns.