Extending Faust's Block-Diagram Algebra

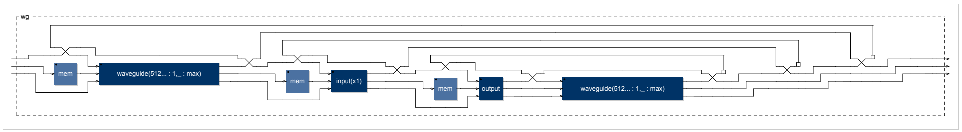
Yann Orlarey[1], Romain Michon[2]

[1]GRAME, Lyon, France
[2]CCRMA, Stanford University, USA

June 26, 2016

In this note we present an extension of Faust's block-diagram algebra aimed at providing a better support for physical modeling. While it is perfectly possible to use the current version of Faust for that purpose, the resulting block-diagrams are difficult to read due to their inherent left to right orientation.

The following diagram from `physicalModeling/basicString.dsp` illustrates the problem:



In order to better describe and represent bi-directional connections, the proposed extension will allow inputs and outputs on all four sides of a diagram[1]. It will also introduce a new vertical composition operation, the possibility to rotate expressions and a series of new route primitives.

1 Conventions

To distinguish the inputs and outputs of the different sides we must number them and name the sides.

Definition 1 (Side names)**.** The sides are named: *West*, *North*, *East*, *South* and are always enumerated in that order.

[1]Here *(block-)diagram* must be understood as the graphical representation of a Faust expression

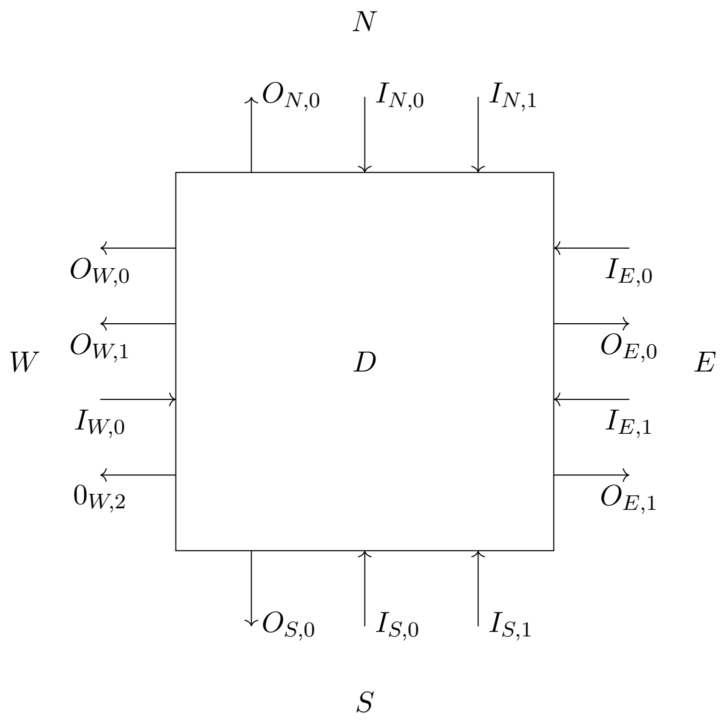


Figure 1: Extended diagrams can have inputs and outputs on all sides

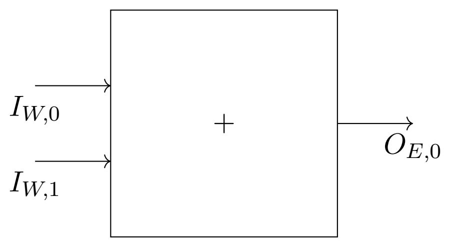


Figure 2: The $+$ primitive

Definition 2 (number of inputs and outputs)**.** We introduce 8 functions : $I_{W,N,E,S}$ and $O_{W,N,E,S}$ to denote the number of inputs and outputs of each side of a diagram. For example Figure 1, we have: $I_W(D) = 1$, $O_W(D) = 3$, $I_N(D) = 2$, etc.

Definition 3 (numbering of inputs and outputs)**.** Inputs (resp. outputs) of the North and South sides are numbered from left to right starting from 0. Inputs (resp. outputs) of the West and East sides are numbered from top to down starting from 0.

For example will notate $I_{E,0}(D)$ the first input on the east side of the diagram D and $O_{W,2}(D)$ the third output of the west side. All other inputs and outputs are notated accordingly.

Definition 4 (topological types)**.** The *topological type* of a diagram describes its connectivity. It indicates the number of inputs and the number of outputs of the *West*, *North*, *East* and *South* sides. For example the diagram D of Figure 1 has type :

$$D : (1, 2, 2, 2) \rightarrow (3, 1, 2, 1)$$

For any diagram D we have:

$$D : (I_W(D), I_N(D), I_E(D), I_S(D)) \rightarrow (O_W(D), O_N(D), O_E(D), O_S(D))$$

With this new convention Faust primitive `+` (see Figure 9) has type $(2, 0, 0, 0) \rightarrow (0, 0, 1, 0)$.

2 Horizontal composition

Sequential composition `A:B`, split composition `A<:B`, and merge composition `A:>B`, are subsumed under a more general *horizontal composition* operation also notated `A:B`. Horizontal composition connects the East side of `A` to the West side of `B`. The East outputs of `A` are connected to the West inputs of `B`, and the West outputs

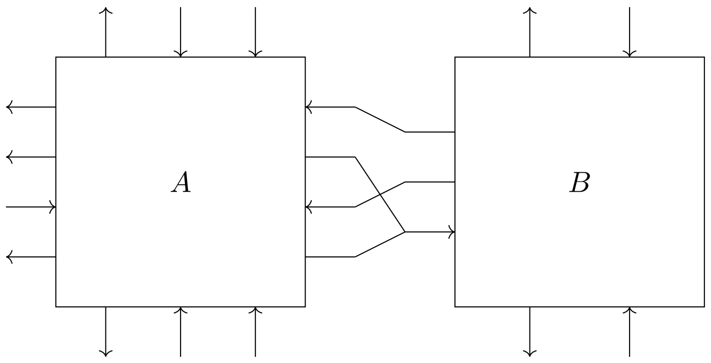


Figure 3: Horizontal composition with implicit merge: $O_E(A) = 2 \times I_W(B)$

of B are connected to the East inputs of A. The number of inputs and outputs must be either equals or one must be an integer multiple of the other. In this case the split or merge rule apply.

In other words the *horizontal composition* A:B is possible if we have $n, m, p, q \in \mathbf{N}^*$ such that:

$$\begin{aligned} n = 1 \vee m = 1 \\ p = 1 \vee q = 1 \\ n.I_E(A) = m.O_W(B) \\ p.O_E(A) = q.I_W(B) \end{aligned}$$

In the example of Figure 3 we have:

$$\begin{aligned} I_E(A) = 2 \\ O_W(B) = 2 \\ O_E(A) = 2 \\ I_W(B) = 1 \end{aligned}$$

Therefore the *horizontal composition* is possible. Moreover we have a merge of the East outputs of A into the West input of B because $O_E(A) = 2 \times I_E(B)$.

Horizontal composition A:B is such that:

$$\begin{aligned} I_W(A, B) = I_W(A) \\ I_N(A, B) = I_N(A) + I_N(B) \\ I_E(A, B) = I_E(B) \\ I_S(A, B) = I_S(A) + I_S(B) \end{aligned}$$

and

$$
\begin{aligned}
O_W(A,B) &= O_W(A) \\
O_N(A,B) &= O_N(A) + O_N(B) \\
O_E(A,B) &= O_E(B) \\
O_S(A,B) &= O_S(A) + O_S(B)
\end{aligned}
$$

3 Vertical composition

Vertical composition `A||B`, is the equivalent of *horizontal composition* in the vertical direction. It connects the south side of A to the north side of B.

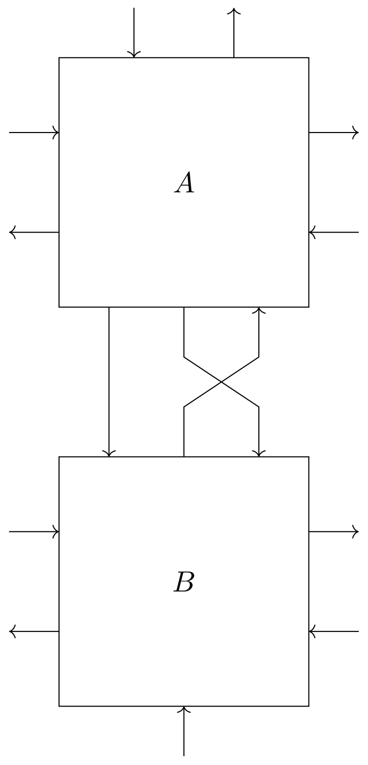


Figure 4: Vertical composition `A||B`

The composition is possible if and only if we have $n, m, p, q \in \mathbf{N}^*$ such that:

$$\begin{aligned}
n = 1 \vee m = 1 \\
p = 1 \vee q = 1 \\
n.I_S(A) = m.O_N(B) \\
p.O_S(A) = q.I_N(B)
\end{aligned}$$

Vertical composition `A||B` is such that:

$$\begin{aligned}
I_W(A, B) &= I_W(A) + I_W(B) \\
I_N(A, B) &= I_N(A) \\
I_E(A, B) &= I_E(A) + I_E(B) \\
I_S(A, B) &= I_S(B)
\end{aligned}$$

and

$$\begin{aligned}
O_W(A, B) &= O_W(A) + O_W(B) \\
O_N(A, B) &= O_N(A) \\
O_E(A, B) &= O_E(A) + O_E(B) \\
O_S(A, B) &= O_S(B)
\end{aligned}$$

4 Parallel composition

Parallel composition `A,B` is always possible. It is such that:

$$\begin{aligned}
I_W(A, B) &= I_W(A) + I_W(B) \\
I_N(A, B) &= I_N(A) + I_N(B) \\
I_E(A, B) &= I_E(A) + I_E(B) \\
I_S(A, B) &= I_S(A) + I_S(B)
\end{aligned}$$

and

$$\begin{aligned}
O_W(A, B) &= O_W(A) + O_W(B) \\
O_N(A, B) &= O_N(A) + O_N(B) \\
O_E(A, B) &= O_E(A) + O_E(B) \\
O_S(A, B) &= O_S(A) + O_S(B)
\end{aligned}$$

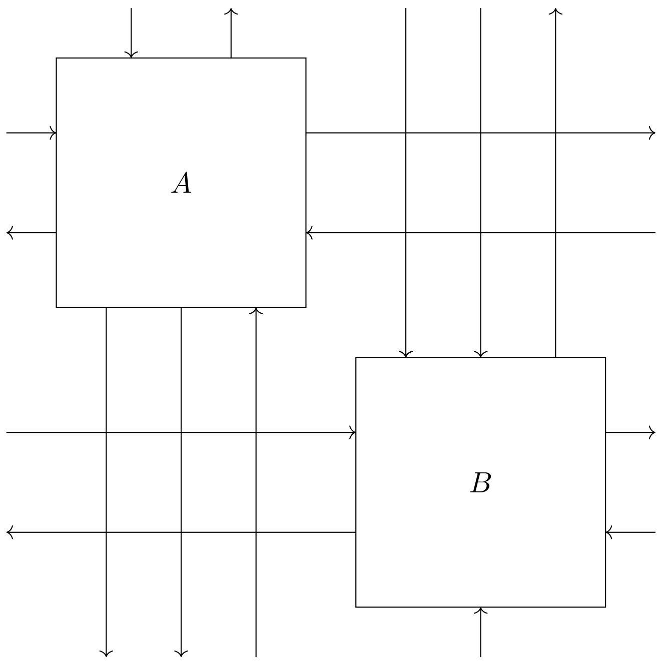


Figure 5: Parallel composition A,B

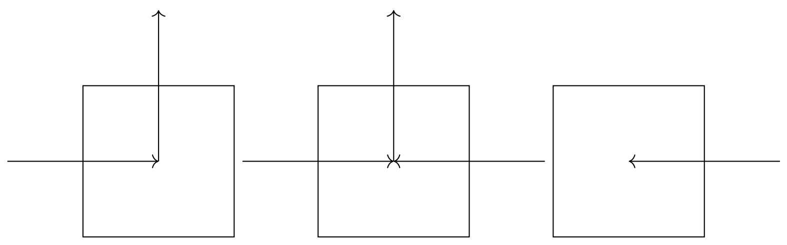


Figure 6: `route("io..")`, `route("ioi.")` and `route(".i..")`

5 Route

The `route()` primitive is used to describe small connection routes likes those of Figure 6.

Each route is defined a 4-character string that indicates if a side contains an input (letter 'i'), an output (letter 'o') or nothing (letter '.').

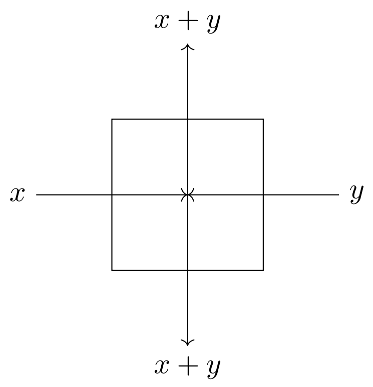


Figure 7: `route("ioio")`

For example `route("ioio")` correspond to Figure 7 and has type

$$(1,0,1,0) \rightarrow (0,1,0,1)$$

Please note that all input signals are added together and the resulting signal is delivered to all outputs. A route with no inputs delivers a 0 signal to its outputs.

We have the following equivalences (see Figure 8):

```
route("..o.")  = 0
route("i.o.")  = _
route("i...")  = !
```

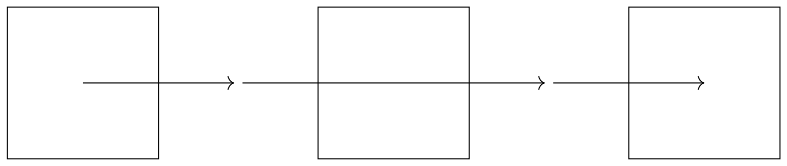


Figure 8: `route("..o."),route("i.o."),route("i...")`

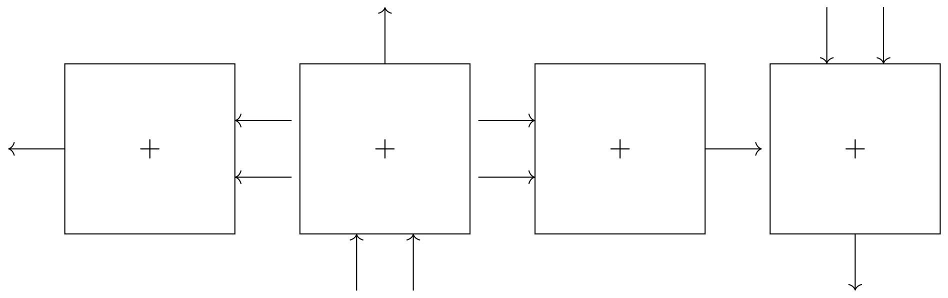


Figure 9: `<*<*+`, `<*+`, `+` and `+*>`

6 Rotation

A diagram `D` can be rotated by -90 using the unary prefix operator `<*` or by +90 using the unary postfix operator `*>`.

If we have `D` with type $(a, b, c, d) \to (i, j, k, l)$ then `<* D` has type $(b, c, d, a) \to (j, k, l, i)$, and `D *>` has type $(d, a, b, c) \to (l, i, j, k)$.

```
<*(A:B)   =  (<*B)||(<*A)
<*(A||B)  =  (<*A):(<*B)

(A:B)*>   =  (A*>)||(B*>)
(A||B)*>  =  (B*>):(A*>)

           <*<*<*<*(A:B)   =  <*<*<*(<*B)||(<*A)
        <*<*<*(<*B)||(<*A)  =  <*<*((<*<*B):(<*<*A))
      <*<*((<*<*B):(<*<*A))  =  <*((<*<*<*A)||(<*<*<*B))
   <*((<*<*<*A)||(<*<*<*B))  =  (<*<*<*<*A):(<*<*<*<*B)
    (<*<*<*<*A):(<*<*<*<*B)  =  A:B
```

7 Examples

Here are some examples illustrating the extended block-diagram algebra in action.

7.1 General Cases

```
// +~_ is equivalent to
feedback = route("..io") || route("iio.")
         : (_' : route("o..i")) || route("ioo.");
```

7.2 Physical Modeling

```

// where "l" is length
waveguide(l) = <*<*(@(l)),@(l);

// where "a" is any 1 input / 1 output element
terminationUp(a) =
    route("o..i") || <*a || route("io..");

// where "a" is any 1 input / 1 output element
terminationUpOutput(a) =
    route("o..i") || <*a || route("ioi.");

// where "a" is any 1 input / 1 output element
terminationDown(a) =
    route("..io") || a*> || route(".io");

// where "a" is any 1 input / 1 output element
terminationDownInput(a) =
    route("i.io") || a*> || route(".io");

rigidTerminationUp = terminationUp(*(-1));

rigidTerminationUpOutput = terminationUpOutput(*(-1));

rigidTerminationDown = terminationDown(*(-1));
```

```
// cross means that voices are crossing
output = route("o.io") || route(".ioi") || route("ioo.")
       : _,cross;

input = _,cross : route("o.ii") || route("io.o")
      || route("iio.");

string(freq,pos) = rigidTerminationDown
                 : waveguide(n1) : input : waveguide(n2)
                 : rigidTerminationUpOutput
    with {
        l = SR/freq/2;
        n1 = l*pos;
        n2 = l*(1-pos);
    };
```

7.3 Implementation of Figure C.40

Implementation of figure C.40 of
https://ccrma.stanford.edu/~jos/pasp/Second_Order_Waveguide_Filter.html

```
wgOsc(g,c) =
    (route("..oi") || <*(_') || route(".oi."))
    : ((route("..oi") : route("i..o")) // upper feed forward
    || ( *(g) : route("ioo.") : route("i.oi")
    : *(c) : route("i.oo") : route("iio."))
    || (route("oii.") : route("ooi.")))
    : (route("i..o") || (_')*> || route("oi.."));
```

8 Implications

The adoption of this extension will have two implications:

1. A new block-diagram drawing algorithm
2. A new propagation algorithm

Signal expressions and code generation remain unchanged.