

An Application of Stream Calculus to Signal Flow Graphs

J.J.M.M. Rutten

CWI and VUA, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

1 Summary

The present paper can be seen as an exercise in the author's stream calculus [Rut01] and gives a new proof for an existing result about stream circuits. Such circuits are also known under the name of signal flow graphs, and are built from (scalar) multipliers, copiers (fan-out), adders (element-wise sum), and registers (here: one-element memory cells, aka delays). Because of the presence of memory, the input-output behaviour of these circuits is best described in terms of functions from streams to streams (of real numbers). The main statement of this paper (Theorem 6), gives a characterization of the input-output behaviour of finite stream circuits in terms of so-called rational streams. It is well-known in the world of signal processing, where it is formulated and proved in terms of the Z-transform (a discrete version of the Laplace transform) and transfer functions (see for instance [Lah98, p.694]). These transforms are used as *representations* of streams of (real or complex) numbers. As a consequence, one has to deal with two different worlds, and some care is required when moving from the one to the other. In contrast, we use stream calculus to formulate and obtain essentially the same result. What is somewhat different and new here is that we use only streams and nothing else. In particular, expressions for streams such as $\frac{1}{(1-X)^2} = (1, 2, 3, \dots)$, are not mere representations but should be read as formal identities. Technically, the formalism of stream calculus is simple, because it uses the *constant* stream $X = (0, 1, 0, 0, 0, \dots)$ as were it a formal variable (cf. work on formal power series such as [BR88]).

We find it worthwhile to present this elementary treatment of signal flow graphs for a number of reasons:

- It explains in very basic terms two fundamental phenomena in the theory of computation: memory (in the form of register or delay elements) and infinite behaviour (in the form of feedback).
- Although Theorem 6 is well-known to electrical engineers, computer scientists do not seem to know it. Also, the result as such is not so easy to isolate in the relevant literature on (discrete-time, linear) system theory.
- Although not worked out here, there is a very close connection between Theorem 6, and a well-known result from theoretical computer science: Kleene's theorem on rational (regular) languages and deterministic finite state automata.

- The present methodology is relevant for the area of component-based software engineering: it has recently been generalised to model software composition by means of so-called component connectors, in terms of relations on the streams of ingoing and outgoing messages (or data elements) at the various communication ports [AR03]. A similar remark applies to our ongoing work on stream-based models of sequential digital circuits.

Stream calculus has been mainly developed as a playground for the use of coinduction definition and proof principles (see [Rut01]). In particular, streams and stream functions can be elegantly defined by so-called stream differential equations. For the elementary operations on streams that are used in this paper (sum, convolution product and its inverse), the more traditional definitions of the necessary operations on streams suffice and therefore, coinduction is not discussed here.

Acknowledgements. Thanks are due to the referees for their critical yet constructive remarks.

2 Basic Stream Calculus

In this section, we study the set $\mathbb{R}^\omega = \{\sigma \mid \sigma : \{0, 1, 2, \dots\} \rightarrow \mathbb{R}\}$ of streams of real numbers. We shall introduce a number of constants and shall define the operations of sum, product, and inverse of streams. These constants and operations make of \mathbb{R}^ω a calculus with many pleasant properties. In particular, it will be possible to compute solutions of linear systems of equations.

We denote streams $\sigma \in \mathbb{R}^\omega$ by $\sigma = (\sigma_0, \sigma_1, \sigma_2, \dots)$. We define the *sum* $\sigma + \tau$ of $\sigma, \tau \in \mathbb{R}^\omega$ by

$$\sigma + \tau = (\sigma_0 + \tau_0, \sigma_1 + \tau_1, \sigma_2 + \tau_2, \dots)$$

(Note that we use the same symbol $+$ for both the sum of two streams and the sum of two real numbers.) We define the *convolution product* $\sigma \times \tau$ by

$$\sigma \times \tau = (\sigma_0 \cdot \tau_0, (\sigma_0 \cdot \tau_1) + (\sigma_1 \cdot \tau_0), (\sigma_0 \cdot \tau_2) + (\sigma_1 \cdot \tau_1) + (\sigma_2 \cdot \tau_0), \dots)$$

That is, for any $n \geq 0$,

$$(\sigma \times \tau)_n = \sum_{k=0}^n \sigma_k \cdot \tau_{n-k}$$

In general, we shall simply say ‘product’ rather than ‘convolution product’. Note that we use the symbol \times for the multiplication of streams and the symbol \cdot for the multiplication of real numbers. Similar to the notation for the multiplication of real numbers (and functions), we shall write $\sigma^0 \equiv 1$ and $\sigma^{n+1} \equiv \sigma \times \sigma^n$. It will be convenient to define the operations of sum and product also for the combination of a real number r and a stream σ . This will allow us, for instance, to write $3 \times \sigma$ for $\sigma + \sigma + \sigma$. In order to define this formally, it will be convenient

to view real numbers as streams in the following manner. We define for every $r \in \mathbb{R}$ a stream $[r] \in \mathbb{R}^\omega$ by

$$[r] = (r, 0, 0, 0, \dots)$$

Note that this defines in fact a function

$$[\] : \mathbb{R} \rightarrow \mathbb{R}^\omega, \quad r \mapsto [r]$$

which embeds the set of real numbers into the set of streams. This definition allows us to add and multiply real numbers r with streams σ , yielding:

$$\begin{aligned} [r] + \sigma &= (r, 0, 0, 0, \dots) + \sigma \\ &= (r + \sigma_0, \sigma_1, \sigma_2, \sigma_3 \dots) \\ [r] \times \sigma &= (r, 0, 0, 0, \dots) \times \sigma \\ &= (r \cdot \sigma_0, r \cdot \sigma_1, r \cdot \sigma_2, \dots) \end{aligned}$$

For notational convenience, we shall usually write simply $r + \sigma$ for $[r] + \sigma$, and similarly $r \times \sigma$ for $[r] \times \sigma$. The context will always make clear whether the notation r has to be interpreted as the real number r or as the stream $[r]$. For multiplication, this difference is moreover made explicit by the use of two different symbols: $r \times \sigma$ always denotes the multiplication of streams (and hence r should be read as the stream $[r]$) and $r \cdot s$ always denotes the multiplication of real numbers. We shall also use the following convention:

$$\begin{aligned} -\sigma &\equiv [-1] \times \sigma \\ &= (-\sigma_0, -\sigma_1, -\sigma_2, \dots) \end{aligned}$$

Here are a few basic properties of our operators.

Proposition 1. *For all $r, s \in \mathbb{R}$ and $\sigma, \tau, \rho \in \mathbb{R}^\omega$,*

$$\begin{aligned} [r] + [s] &= [r + s] \\ \sigma + \mathbf{0} &= \sigma \\ \sigma + \tau &= \tau + \sigma \\ \sigma + (\tau + \rho) &= (\sigma + \tau) + \rho \\ [r] \times [s] &= [r \cdot s] \\ \mathbf{0} \times \sigma &= \mathbf{0} \\ \mathbf{1} \times \sigma &= \sigma \\ \sigma \times \tau &= \tau \times \sigma \\ \sigma \times (\tau + \rho) &= (\sigma \times \tau) + (\sigma \times \rho) \\ \sigma \times (\tau \times \rho) &= (\sigma \times \tau) \times \rho \end{aligned}$$

Particularly simple are those streams that from a certain point onwards are constant zero:

$$\sigma = (r_0, r_1, r_2, \dots, r_n, 0, 0, 0, \dots)$$

for $n \geq 0$ and $r_0, \dots, r_n \in \mathbb{R}$. Using the following constant, we shall see that there is a very convenient way of denoting such streams: we define

$$X = (0, 1, 0, 0, 0, \dots)$$

It satisfies, for all $r \in \mathbb{R}$, $\sigma \in \mathbb{R}^\omega$, and $n \geq 0$:

$$\begin{aligned} r \times X &= (0, r, 0, 0, 0, \dots) \\ X \times \sigma &= (0, \sigma_0, \sigma_1, \sigma_2, \dots) \\ X^n &= (\underbrace{0, \dots, 0}_{n \text{ times}}, 1, 0, 0, 0, \dots) \end{aligned}$$

For instance, $2 + 3X - 8X^3 = (2, 3, 0, -8, 0, 0, 0, \dots)$. More generally, we have, for all $n \geq 0$ and all $r_0, \dots, r_n \in \mathbb{R}$:

$$r_0 + r_1X + r_2X^2 + \dots + r_nX^n = (r_0, r_1, r_2, \dots, r_n, 0, 0, 0, \dots)$$

Such streams are called *polynomial streams*. Note that although a polynomial stream such as $2 + 3X - 8X^3$ looks like a (polynomial) *function* $f(x) = 2 + 3x - 8x^3$, for which x is a variable, it really is a *stream*, built from constant streams (2, 3, 8, and X), and the operations of sum and product. At the same time, it is true that we can calculate with polynomial streams in precisely the same way as we are used to compute with (polynomial) functions, as is illustrated by the following example (here we use the basic properties of sum and product listed in Proposition 1): $(2 - X) + (1 + 3X) = 3 + 2X$ and $(2 - X) \times (1 + 3X) = 2 + 5X - 3X^2$.

We shall need to solve linear equations in one unknown τ , such as

$$\tau = 1 + (X \times \tau) \tag{1}$$

(where, recall, $1 = (1, 0, 0, 0, \dots)$). Ideally, we would like to solve (1) by reasoning as follows:

$$\begin{aligned} \tau &= 1 + (X \times \tau) \\ \Rightarrow \tau - (X \times \tau) &= 1 \\ \Rightarrow (1 - X) \times \tau &= 1 \\ \Rightarrow \tau &= \frac{1}{1 - X} \end{aligned}$$

Recall that we are not dealing with functions but with streams. Therefore it is not immediately obvious what we mean by the ‘inverse’ of a stream $1 - X = (1, -1, 0, 0, 0, \dots)$. There is however the following fact: for any stream σ such that $\sigma_0 \neq 0$, there exists a unique stream τ such that $\sigma \times \tau = 1$. A proof of this fact can be given in various ways:

- (a) Using the definition of convolution product, one can easily derive the following recurrence relation

$$\tau_n = \frac{1}{\sigma_0} \cdot \sum_{k=0}^{n-1} \sigma_{n-k} \cdot \tau_k$$

by which the elements of τ can be constructed one by one.

- (b) Alternatively and equivalently, one can use the algorithm of long division to obtain τ out of σ .
- (c) Our personal favourite is a method described in [Rut01], where we have introduced the operation of inverse by means of so-called stream differential equations, formulated in terms of the notions of *stream derivative* $\sigma' = (\sigma_1, \sigma_2, \sigma_3, \dots)$ and *initial value* σ_0 for $\sigma \in \mathbb{R}^\omega$. (In fact, all the operations on \mathbb{R}^ω are there introduced by means of such equations.)

Now we can simply define the *inverse* $\frac{1}{\sigma}$ of a stream σ with $\sigma_0 \neq 0$ as the unique stream τ such that $\sigma \times \tau = 1$. Here are a few examples that can be easily computed using any of the methods (a)-(c) above:

$$\begin{aligned} \frac{1}{1 - X} &= (1, 1, 1, \dots) \\ \frac{1}{1 - X^2} &= (1, 0, 1, 0, 1, 0, \dots) \\ \frac{1}{(1 - X)^2} &= (1, 2, 3, \dots) \end{aligned}$$

As with sum and product, we can calculate with the operation of inverse in the same way as we compute with functions: For all $\sigma, \tau \in \mathbb{R}^\omega$ with $\sigma_0 \neq 0 \neq \tau_0$,

$$\begin{aligned} \sigma \times \frac{1}{\sigma} &= 1 \\ \frac{1}{\sigma} \times \frac{1}{\tau} &= \frac{1}{\sigma \times \tau} \\ \frac{1}{\frac{1}{\sigma}} &= \sigma \end{aligned}$$

Using the various properties of our operators, it is straightforward to see that in the calculus of streams, we can solve linear equations as usual. Consider for instance the following system of equations:

$$\begin{aligned} \sigma &= 1 + (X \times \tau) \\ \tau &= X \times \sigma \end{aligned}$$

In order to find σ and τ , we compute as follows: $\sigma = 1 + (X \times \tau) = 1 + (X \times X \times \sigma) = 1 + (X^2 \times \sigma)$. This implies $\sigma - (X^2 \times \sigma) = 1$ and $(1 - X^2) \times \sigma = 1$. Thus $\sigma = \frac{1}{1 - X^2}$ and $\tau = \frac{X}{1 - X^2}$.

We conclude this section with the following definition, which is of central importance for the formulation of the main result of this paper.

Definition 2. *The product of a polynomial stream and the inverse of a polynomial stream is called a rational stream. Equivalently, a stream σ is rational if there exist $n, m \geq 0$ and coefficients $r_0, \dots, r_n, s_0, \dots, s_m \in \mathbb{R}$ with $s_0 \neq 0$, such that*

$$\sigma = \frac{r_0 + r_1X + r_2X^2 + \dots + r_nX^n}{s_0 + s_1X + s_2X^2 + \dots + s_mX^m}$$

□

3 Stream Circuits

Certain functions from \mathbb{R}^ω to \mathbb{R}^ω can be represented by means of graphical networks that are built from a small number of basic ingredients. Such networks can be viewed as implementations of stream functions. We call them stream circuits; in the literature, they are also referred to as (signal) flow graphs. Using the basic stream calculus from Section 2, we shall give a formal but simple answer to the question precisely which stream functions can be implemented by such stream circuits.

3.1 Basic Circuits

The circuits that we are about to describe, will generally have a number of *input ends* and a number of *output ends*. Here is an example of a simple circuit, consisting of one input and one output end:



The input end is denoted by the arrow shaft \vdash and the output end is denoted by the arrow head \longrightarrow . For streams $\sigma, \tau \in \mathbb{R}^\omega$, we shall write

$$\sigma \vdash \longrightarrow \tau$$

and say that the circuit *inputs* the stream σ and *outputs* the stream τ . Writing the elements of these streams explicitly, this notation is equivalent to

$$(\sigma_0, \sigma_1, \sigma_2, \dots) \vdash \longrightarrow (\tau_0, \tau_1, \tau_2, \dots)$$

which better expresses the intended operational behaviour of the circuit: It consists of an infinite sequence of actions, at time moments $0, 1, 2, \dots$. At each moment $n \geq 0$, the circuit simultaneously inputs the value $\sigma_n \in \mathbb{R}$ at its input end and outputs the value $\tau_n \in \mathbb{R}$ at its output end. In general, this value τ_n depends both on the value σ_n and on the values σ_i that have been taken as inputs at earlier time moments $i < n$. Note that this implies that circuits have memory.

Next we present the four basic types of circuits, out of which all other circuits in this section will be constructed.

- (a) For every $a \in \mathbb{R}$, we define a circuit with one input and one output end, called an *a-multiplier*, for all $\sigma, \tau \in \mathbb{R}^\omega$, by

$$\begin{aligned} \sigma \vdash \xrightarrow{a} \tau &\iff \tau_n = a \cdot \sigma_n, \text{ all } n \geq 0 \\ &\iff \tau = a \times \sigma \end{aligned}$$

This circuit takes, at any moment $n \geq 0$, a value σ_n at its input end, multiplies it with the constant a , and outputs the result $\tau_n = a \cdot \sigma_n$ at its output end. It defines, in other words, a function that assigns to an input stream σ the output stream $\tau = a \times \sigma$.

Occasionally, it will be more convenient to write the multiplying factor a as a super- or subscript of the arrow:

$$\text{---} \overset{a}{\rightarrow} \equiv \text{---} \overset{a}{\rightarrow} \equiv \text{---} \underset{a}{\rightarrow}$$

- (b) The *adder* circuit has two input and one output ends, and is defined, for all $\sigma, \tau, \rho \in \mathbb{R}^\omega$ by

$$\begin{array}{ccc} \begin{array}{c} \sigma \curvearrowright \\ + \\ \tau \curvearrowleft \end{array} \longrightarrow \rho & \iff & \rho_n = \sigma_n + \tau_n, \text{ all } n \geq 0 \\ & & \iff \rho = \sigma + \tau \end{array}$$

At moment $n \geq 0$, the adder simultaneously inputs the values σ_n and τ_n at its input ends, and outputs their sum $\rho_n = \sigma_n + \tau_n$ at its output end.

- (c) The *copier* circuit has one input and two output ends and is defined, for all $\sigma, \tau, \rho \in \mathbb{R}^\omega$, by

$$\begin{array}{ccc} \sigma \text{---} c \begin{array}{l} \curvearrowright \\ \curvearrowleft \end{array} & \iff & \tau_n = \sigma_n = \rho_n, \text{ all } n \geq 0 \\ & & \iff \tau = \sigma = \rho \end{array}$$

At any moment $n \geq 0$, the copier inputs the value σ_n at its input end, and outputs two identical copies τ_n and ρ_n at its output ends.

- (d) A *register* circuit has one input and one output end and is defined, for all $\sigma, \tau \in \mathbb{R}^\omega$, by

$$\begin{array}{ccc} \sigma \text{---} R \rightarrow \tau & \iff & \tau_0 = 0 \text{ and } \tau_n = \sigma_{n-1}, \text{ all } n \geq 1 \\ & & \iff \tau = (0, \sigma_0, \sigma_1, \sigma_2, \dots) \end{array}$$

The register circuit can be viewed as consisting of a one-place memory cell that initially contains the value 0. The register starts its activity, at time moment 0, by outputting its value $\tau_0 = 0$ at its output end, while it simultaneously inputs the value σ_0 at its input end, which is stored in the memory cell. At any future time moment $n \geq 1$, the value $\tau_n = \sigma_{n-1}$ is output and the value σ_n is input and stored. (For obvious reasons, the register circuit is sometimes also called a *unit delay*.) Recalling that for the constant stream $X = (0, 1, 0, 0, \dots)$, we have $X \times \sigma = (0, \sigma_0, \sigma_1, \sigma_2, \dots)$, it follows that for all $\sigma, \tau \in \mathbb{R}^\omega$,

$$\sigma \text{---} R \rightarrow \tau \iff \tau = X \times \sigma$$

3.2 Circuit Composition

We can construct a larger circuit out of two smaller ones by connecting output ends of the first to input ends of the second. For instance, for the composition of a 2-multiplier and a 3-multiplier, we shall write

$$\text{---} \overset{2}{\rightarrow} \circ \text{---} \overset{3}{\rightarrow}$$

We call the connection point \circ an (internal) *node* of the composed circuit. A computation step of this circuit, at any moment in time, consists of the simultaneous occurrence of the following actions: a value is input at the input end of the 2-register; it is multiplied by 2 and output at the output end of the 2-register; the result is input at the input end of the 3-register, is multiplied by 3 and is output at the output end of the 3-multiplier. More formally, and fortunately also more succinctly, we define the behaviour of the composed circuit, for all $\sigma, \tau \in \mathbb{R}^\omega$, by

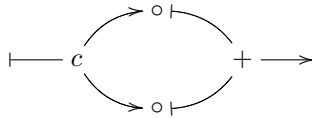
$$\begin{aligned} \sigma \vdash 2 \rightarrow \circ \vdash 3 \rightarrow \tau \\ \iff \sigma \vdash 2 \rightarrow \exists \rho \vdash 3 \rightarrow \tau \\ \iff \exists \rho \in \mathbb{R}^\omega : \sigma \vdash 2 \rightarrow \rho \text{ and } \rho \vdash 3 \rightarrow \tau \end{aligned}$$

We shall consider all three of the above notations as equivalent. Combining the definitions of a 2- and 3-multiplier, we can in the above example easily compute how the output stream τ depends on the input stream σ :

$$\begin{aligned} \sigma \vdash 2 \rightarrow \circ \vdash 3 \rightarrow \tau \\ \iff \exists \rho \in \mathbb{R}^\omega : \sigma \vdash 2 \rightarrow \rho \text{ and } \rho \vdash 3 \rightarrow \tau \\ \iff \exists \rho \in \mathbb{R}^\omega : \rho = 2 \times \sigma \text{ and } \tau = 3 \times \rho \\ \iff \tau = 6 \times \sigma \end{aligned}$$

Note that the stream ρ is uniquely determined by the stream σ . The motivation for our notation “ $\exists \rho$ ” is not so much to suggest that there might be more possible candidate streams for ρ , but rather to emphasise the fact that in order to express the output stream τ in terms of σ , we have to compute the value of the stream ρ in the middle.

We can compose circuits, more generally, with several output ends with circuits having a corresponding number of input ends, as in the following example:



In this example, the behaviour of the resulting circuit is defined, for all $\sigma, \tau \in \mathbb{R}^\omega$, by

$$\begin{aligned} \sigma \vdash c \rightarrow \circ \vdash + \rightarrow \tau \\ \iff \sigma \vdash c \rightarrow \begin{matrix} \exists \gamma \vdash \\ \exists \delta \vdash \end{matrix} \circ \vdash + \rightarrow \tau \end{aligned}$$

(To save space, we have omitted the symbol \times for multiplication.) We can now express the output stream τ in terms of the input stream σ as follows:

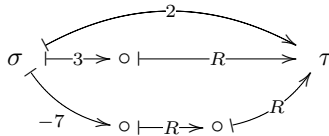
$$\begin{aligned} \tau &= (2 \times \sigma) + (3X \times \sigma) + (-7X^2 \times \sigma) \\ &= (2 + 3X - 7X^2) \times \sigma \end{aligned}$$

The circuit above computes — we shall also say *implements* — the following function on streams:

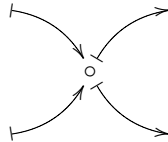
$$f : \mathbb{R}^\omega \rightarrow \mathbb{R}^\omega, \quad f(\sigma) = (2 + 3X - 7X^2) \times \sigma$$

If we supply the circuit with the input stream $\sigma = 1$ ($= (1, 0, 0, 0, \dots)$) then the output stream is $\tau = f(1) = 2 + 3X - 7X^2$. We call this the stream *generated* by the circuit.

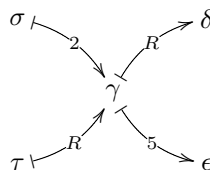
Convention 3. *In order to reduce the size of the diagrams with which we depict stream circuits, it will often be convenient to leave the operations of copying and addition implicit. In this manner, we can, for instance, draw the circuit above as follows:*



The (respective elements of the) stream σ gets copied along each of the three outgoing arrows. Similarly, the stream τ will be equal to the sum of the output streams of the three incoming arrows. This convention saves a lot of writing. Moreover, if we want to express τ in terms of σ , we now have only three internal streams to compute. If a node has both incoming and outgoing arrows, such as



then first the values of the output streams of the incoming arrows have to be added; then the resulting sum is copied and given as an input stream to each of the outgoing arrows. Consider for instance the circuit below. It has input streams σ and τ , an intermediate stream γ , and output streams δ and ϵ in \mathbb{R}^ω :

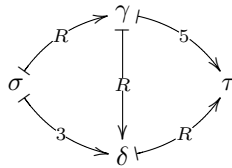


satisfying

$$\begin{aligned}
 \gamma &= 2\sigma + (X \times \tau) \\
 \delta &= X \times \gamma \\
 &= (2X \times \sigma) + (X^2 \times \tau) \\
 \epsilon &= 5\gamma \\
 &= 10\sigma + (5X \times \tau)
 \end{aligned}$$

□

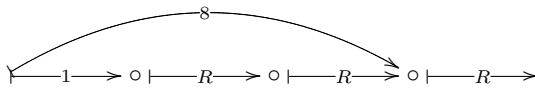
As an example, we compute the stream function implemented by the following circuit, with input stream σ , output stream τ , and intermediate streams γ and δ :



We have:

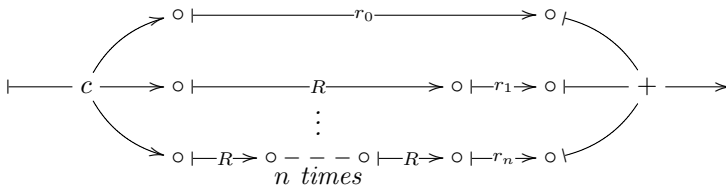
$$\begin{aligned}
 \gamma &= X \times \sigma \\
 \delta &= (3 \times \sigma) + (X^2 \times \sigma) \\
 \tau &= (5 \times \gamma) + (X \times \delta) \\
 &= (8X + X^3) \times \sigma
 \end{aligned}$$

Thus the stream function implemented by this circuit is $f : \mathbb{R}^\omega \rightarrow \mathbb{R}^\omega$ with $f(\sigma) = (8X + X^3) \times \sigma$, for all $\sigma \in \mathbb{R}^\omega$. An equivalent circuit, implementing the same stream function, is given by:

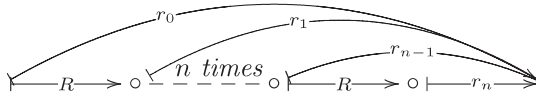


The following proposition, of which we have omitted the easy proof, characterizes which stream functions can be implemented by the type of circuits that we have been considering so far.

Proposition 4. For all $n \geq 0$ and $r_0, \dots, r_n \in \mathbb{R}$, each of the following two circuits:



and



implements the stream function $f : \mathbb{R}^\omega \rightarrow \mathbb{R}^\omega$ given, for all $\sigma \in \mathbb{R}^\omega$, by

$$f(\sigma) = \rho \times \sigma$$

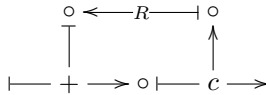
where the stream ρ (generated by these circuits) is the polynomial

$$\rho = r_0 + r_1X + r_2X^2 + \dots + r_{n-1}X^{n-1} + r_nX^n$$

□

3.3 Circuits with Feedback Loops

The use of feedback loops in stream circuits increases their expressive power substantially. We shall start with a elementary example and then give a simple and precise characterization of all stream functions that can be implemented by circuits with feedback loops. Consider the following circuit:

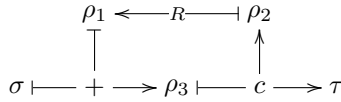


In spite of its simplicity, this circuit is already quite interesting. Before we give a formal computation of the stream function that this circuit implements, we give an informal description of its behaviour first. Assuming that we have an input stream $\sigma = (\sigma_0, \sigma_1, \sigma_2, \dots)$, we compute the respective elements of the output stream $\tau = (\tau_0, \tau_1, \tau_2, \dots)$. Recall that a register can be viewed as a one-place memory cell with initial value 0. At moment 0, our circuit begins its activity by inputting the first value σ_0 at its input end. The present value of the register, 0, is added to this and the result $\tau_0 = \sigma_0 + 0 = \sigma_0$ is the first value to be output. At the same time, this value σ_0 is copied and stored as the new value of the register. The next step consists of inputting the value σ_1 , adding the present value of the register, σ_0 , to it, and outputting the resulting value $\tau_1 = \sigma_0 + \sigma_1$. At the same time, this value $\sigma_0 + \sigma_1$ is copied and stored as the new value of the register. The next step will input σ_2 and output the value $\tau_2 = \sigma_0 + \sigma_1 + \sigma_2$. And so on. We find:

$$\tau = (\sigma_0, \sigma_0 + \sigma_1, \sigma_0 + \sigma_1 + \sigma_2, \dots)$$

Next we show how the same answer can be obtained, more formally and more systematically, by applying a bit of basic stream calculus. As before, we try to express the output stream τ in terms of the input stream σ by computing

the values of intermediate streams $\rho_1, \rho_2, \rho_3 \in \mathbb{R}^\omega$, corresponding to the three internal nodes of the circuit, such that



Note that the values of ρ_1, ρ_2, ρ_3 are mutually dependent because of the presence of the feedback loop: ρ_3 depends on ρ_1 which depends on ρ_2 which depends on ρ_3 . The stream calculus developed in Section 2 is precisely fit to deal with this type of circularity. Unfolding the definitions of the basic circuits of which the above circuit is composed (one adder, one register, and one copier), we find the following system of equations:

$$\begin{aligned}
 \rho_1 &= X \times \rho_2 \\
 \rho_3 &= \sigma + \rho_1 \\
 \rho_2 &= \rho_3 \\
 \tau &= \rho_3
 \end{aligned}$$

We have seen in the previous section how to solve such systems of equations. The right way to start, which will work in general, is to compute first the output stream of the register:

$$\begin{aligned}
 \rho_1 &= X \times \rho_2 \\
 &= X \times \rho_3 \\
 &= X \times (\sigma + \rho_1) \\
 &= (X \times \sigma) + (X \times \rho_1)
 \end{aligned}$$

This implies $\rho_1 - (X \times \rho_1) = X \times \sigma$, which is equivalent to $\rho_1 = \frac{X}{1-X} \times \sigma$. As a consequence, $\tau = \rho_3 = \rho_2 = \sigma + \rho_1 = \sigma + (\frac{X}{1-X} \times \sigma) = \frac{1}{1-X} \times \sigma$. Thus the stream function $f : \mathbb{R}^\omega \rightarrow \mathbb{R}^\omega$ that is implemented by the feedback circuit is given, for all $\sigma \in \mathbb{R}^\omega$, by

$$f(\sigma) = \frac{1}{1-X} \times \sigma$$

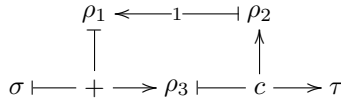
We see that this function consists again of the convolution product of the argument σ and a constant stream $\frac{1}{1-X}$. The main difference with the examples in the previous subsections is that now this constant stream is no longer polynomial but rational.

We still have to check that the first informal and the second formal computation of the function implemented by the feedback circuit coincide. But this follows from the fact that, for all $\sigma \in \mathbb{R}^\omega$,

$$\frac{1}{1-X} \times \sigma = (\sigma_0, \sigma_0 + \sigma_1, \sigma_0 + \sigma_1 + \sigma_2, \dots)$$

which is an immediate consequence of the definition of the convolution product and the fact that $\frac{1}{1-X} = (1, 1, 1, \dots)$.

Not every feedback loop gives rise to a circuit with a well-defined behaviour. Consider for instance the following circuit, with input stream σ , output stream τ , and internal streams ρ_1, ρ_2, ρ_3 :



In this circuit, we have replaced the register feedback loop of the example above by a 1-multiplier. If we try to compute the stream function of this circuit as before, we find the following system of equations:

$$\begin{aligned}
 \rho_1 &= 1 \times \rho_2 \\
 \rho_3 &= \sigma + \rho_1 \\
 \rho_2 &= \rho_3 \\
 \tau &= \rho_3
 \end{aligned}$$

This leads to $\rho_3 = \sigma + \rho_3$, which implies $\sigma = 0$. But σ is supposed to be an arbitrary input stream, so this does not make sense.

Problems like this can be avoided by assuming that circuits have the following property.

Assumption 5. *From now on, we shall consider only circuits in which every feedback loop passes through at least one register.* □

Note that this condition is equivalent to requiring that the circuit has no infinite paths passing through only multipliers, adders, and copiers.

Next we present the main result of the present paper. It is a characterization of which stream functions can be implemented by finite stream circuits. We formulate it for finite circuits that have one input and one output end, but it can be easily generalised to circuits with many inputs and outputs.

Theorem 6. (a) *Let C be any finite stream circuit, possibly containing feedback loops (that always pass through at least one register). The stream function $f : \mathbb{R}^\omega \rightarrow \mathbb{R}^\omega$ implemented by C is always of the form:*

$$f(\sigma) = \rho \times \sigma$$

for all $\sigma \in \mathbb{R}^\omega$ and for some fixed rational stream

$$\rho = \frac{r_0 + r_1X + r_2X^2 + \dots + r_nX^n}{s_0 + s_1X + s_2X^2 + \dots + s_mX^m}$$

with $n, m \geq 0, r_0, \dots, r_n, s_0, \dots, s_m \in \mathbb{R}$, and $s_0 \neq 0$.

(b) Let $f : \mathbb{R}^\omega \rightarrow \mathbb{R}^\omega$ be a stream function of the form, for all $\sigma \in \mathbb{R}^\omega$:

$$f(\sigma) = \rho \times \sigma$$

for some fixed rational stream ρ . Then there exists a finite stream circuit C that implements f .

Proof. (a) Consider a finite circuit C containing $k \geq 1$ registers. We associate with the input end of C a stream σ and with the output end of C a stream τ . With the output end of each register R_i , we associate a stream α_i . For the input end of each register R_i , we look at all incoming paths that: (i) start in either an output end of any of the registers or the input end of C , (ii) lead via adders, copiers, and multipliers, (iii) to the input end of R_i . Because of Assumption 5, there are only finitely many of such paths. This leads to an equation of the form

$$\alpha_i = (a_i^1 \times X \times \alpha^1) + \dots + (a_i^k \times X \times \alpha^k) + (a_i \times X \times \sigma)$$

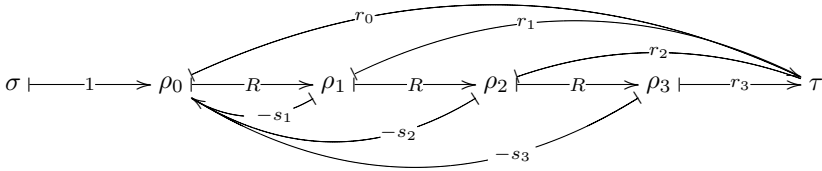
for some $a_i, a_i^j \in \mathbb{R}$. We have one such equation for each $1 \leq i \leq k$. Solving this system of k equations in stream calculus as before, yields for each register an expression $\alpha_i = \rho_i \times \sigma$, for some rational stream ρ_i . Finally, we play the same game for τ , at the output end of C , as we did for each of the registers. This will yield the following type of expression for τ :

$$\begin{aligned} \tau &= (b_1 \times \alpha_1) + \dots + (b_k \times \alpha_k) + (b \times \sigma) \\ &= ((b_1 \times \rho_1) + \dots + (b_k \times \rho_k) + b) \times \sigma \end{aligned}$$

for some $b, b_i \in \mathbb{R}$, which proves (a). For (b), we treat only the special case that

$$\rho = \frac{r_0 + r_1X + r_2X^2 + r_3X^3}{1 + s_1X + s_2X^2 + s_3X^3}$$

where we have taken $n = m = 3$ and $s_0 = 1$. The general case is not more difficult, just more writing. We claim that the following circuit implements the function $f(\sigma) = \rho \times \sigma$ (all $\sigma \in \mathbb{R}^\omega$):



where we have denoted input and output streams by σ and τ , and intermediate streams by $\rho_0, \rho_1, \rho_2, \rho_3$. They satisfy the following equations:

$$\begin{aligned} \rho_0 &= \sigma - (s_1 \times \rho_1) - (s_2 \times \rho_2) - (s_3 \times \rho_3) \\ \rho_1 &= X \times \rho_0 \\ \rho_2 &= X \times \rho_1 \\ \rho_3 &= X \times \rho_2 \\ \tau &= (r_0 \times \rho_0) + (r_1 \times \rho_1) + (r_2 \times \rho_2) + (r_3 \times \rho_3) \end{aligned}$$

It follows that

$$\rho_0 = \sigma - (s_1 X \times \rho_0) - (s_2 X^2 \times \rho_0) - (s_3 X^3 \times \rho_0)$$

As a consequence, we have, for $i = 0, 1, 2, 3$, that

$$\rho_i = \frac{X^i}{1 + s_1 X + s_2 X^2 + s_3 X^3} \times \sigma$$

This implies

$$\tau = \frac{r_0 + r_1 X + r_2 X^2 + r_3 X^3}{1 + s_1 X + s_2 X^2 + s_3 X^3} \times \sigma$$

whereby the claim above is proved. \square

Taking $\sigma = 1$ in Theorem 6 gives the following corollary.

Corollary 7. *A stream $\rho \in \mathbb{R}^\omega$ is rational if and only if it is generated by a (finite) stream circuit.* \square

References

- [AR03] F. Arbab and J.J.M.M. Rutten. A coinductive calculus of component connectors. In M. Wirsing, D. Pattinson, and R. Hennicker, editors, *Proceedings of WADT 2002*, volume 2755 of *LNCS*, pages 35–56. Springer, 2003.
- [BR88] J. Berstel and C. Reutenauer. *Rational series and their languages*, volume 12 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, 1988.
- [Lah98] B.P. Lahti. *Signal Processing & Linear Systems*. Oxford University Press, 1998.
- [Rut01] J.J.M.M. Rutten. Elements of stream calculus (an extensive exercise in coinduction). In S. Brooks and M. Mislove, editors, *Proceedings of MFPS 2001*, volume 45 of *ENTCS*, pages 1–66. Elsevier Science Publishers, 2001. To appear in *MSCS*.