Sampling, Splitting and Merging in Coinductive Stream Calculus

Milad Niqui[1,⋆] and Jan Rutten[1,2]

[1] Centrum Wiskunde & Informatica (CWI), The Netherlands
{M.Niqui,janr}@cwi.nl
[2] Radboud University Nijmegen, The Netherlands

Abstract. We study various operations for partitioning, projecting and merging streams of data. These operations are motivated by their use in dataflow programming and the stream processing languages. We use the framework of *stream calculus* and *stream circuits* for defining and proving properties of such operations using behavioural differential equations and coinduction proof principles. We study the invariance of certain well patterned classes of streams, namely rational and algebraic streams, under splitting and merging. Finally we show that stream circuits extended with gates for dyadic split and merge are expressive enough to realise some non-rational algebraic streams, thereby going beyond ordinary stream circuits.

Keywords: stream calculus, dataflow programming, coinduction, rational stream, algebraic stream, stream circuit.

1 Introduction

In this paper, we study various operations for splitting, partitioning, projecting and merging streams (infinite sequences of data). These operations are motivated by their use in dataflow programming and stream processing languages (e.g., [BŞ01]).

Our perspective on streams and stream operations will be essentially coalgebraic. More specifically, we use the framework of *stream calculus* [Rut05a] and *stream circuits* [Rut05b] for defining and proving properties of such operations. Definitions are typically given using behavioural stream differential equations. Proofs will mostly be given by coinduction, with which two streams can be shown to be equal by the construction of a suitable stream bisimulation relation between them.

The use of stream calculus and coinduction leads to new and simpler definitions and proofs of several existing notions and properties, some of which are taken from [Mak08]. To mention already one example here (see Sections 3 and 4 for more): a periodic stream sampler S is a stream operation that produces a

⋆ Supported by a VENI grant from the Netherlands Organisation for Scientific Research (NWO).

C. Bolduc, J. Desharnais, and B. Ktari (Eds.): MPC 2010, LNCS 6120, pp. 310–330, 2010.

substream of a given stream σ by taking out of each block of $l \geq 0$ elements a subset of $k \leq l$ elements (at fixed positions). Periodic stream samplers can be defined by the following stream differential equation:

$$S(\sigma)^{(k)} = S(\sigma^{(l)})$$

(plus the specification of k initial values). Here $(-)^{(i)}$ denotes the i-th stream derivative, which is defined as the operation *tail* applied i times. This differential equation is elementary, almost trivial. Yet it allows for proofs of basic facts (such as: composing two periodic steam samplers yields again a periodic stream sampler) that are much simpler than those in the literature.

Using stream calculus and stream circuits, we obtain also a number of new results. More specifically, we prove (in Sections 5 and 6) the invariance of certain well patterned classes of streams, namely rational and algebraic streams, under the operations of splitting and merging. Furthermore, we show (in Section 7) that stream circuits extended with gates for dyadic split and merge are expressive enough to realise some non-rational algebraic streams (such as the Prouhet–Thue–Morse stream), thereby going beyond ordinary stream circuits.

As mentioned above, this paper attempts to give a new perspective on existing notions and results, and also obtains some modest new results. The presented new outlook gives rise to a host of further questions and research directions. Section 8 discusses related work and future research.

2 Preliminaries

We define the set of *streams* over a set A by $A^\omega = \{\sigma \mid \sigma : \mathbb{N} \to A\}$. We denote elements $\sigma \in A^\omega$ by $\sigma = (\sigma(0), \sigma(1), \sigma(2), \ldots)$. The *stream derivative* of a stream σ is $\sigma' = (\sigma(1), \sigma(2), \sigma(3), \ldots)$ and the *initial value* of σ is $\sigma(0)$. For $n \geq 0$ and $\sigma \in A^\omega$, we define higher-order derivatives by $\sigma^{(0)} = \sigma$ and $\sigma^{(n+1)} = (\sigma^{(n)})'$. We have $\sigma(n) = \sigma^{(n)}(0)$.

A *stream bisimulation relation* is a set $R \subseteq A^\omega \times A^\omega$ such that, for all $(\sigma, \tau) \in R$,

$$\sigma(0) = \tau(0) \quad \text{and} \quad (\sigma', \tau') \in R \ .$$

We write $\sigma \sim \tau$ if there exists a bisimulation R with $(\sigma, \tau) \in R$. The *coinduction proof principle* allows us to prove the equality of two streams by establishing the existence of an appropriate bisimulation relation:

$$\sigma \sim \tau \ \Rightarrow \ \sigma = \tau \ .$$

If A has some algebraic structure, A^ω inherits (parts of) this structure. Assume $\langle A, +, \cdot, -, 0, 1\rangle$ is a ring[1]. For $r \in A$, we define the constant stream $[r] = (r, 0, 0, 0, \ldots)$, which we often denote again by r. Another constant stream

[1] In fact many of the operations on A^ω only need a semiring structure on A [BR88, Rut08].

is $X = (0, 1, 0, 0, 0, \ldots)$. For $\sigma, \tau \in A^\omega$ and $n \geq 0$, the operations of *sum* and (convolution) *product* are given by

$$(\sigma + \tau)(n) = \sigma(n) + \tau(n) \ , \qquad (\sigma \times \tau)(n) = \sum_{i=0}^{n} \sigma(i) \cdot \tau(n-i)$$

(where $\cdot$ denotes ring multiplication).

We call a stream $\pi \in A^\omega$ *polynomial* if there are $k \geq 0$ and $a_i \in A_0$ such that

$$\pi = a_0 + a_1 X + a_2 X^2 + \cdots + a_k X^k \ = \ (a_0, a_1, a_2, \ldots, a_k, 0, 0, 0, \ldots)$$

where we write $a_i X^i$ for $[a_i] \times X^i$ with X^i the i-fold product of X with itself.

One can compute a stream from its initial value and derivative by the so-called *fundamental theorem* of stream calculus [Rut05a]: for all $\sigma \in A^\omega$,

$$\sigma = \sigma(0) + (X \times \sigma')$$

(writing $\sigma(0)$ for $[\sigma(0)]$).

Next assume A is a field, i.e., every nonzero element has a unique multiplicative inverse. Then this multiplicative inverse operation may be carried over to A^ω: if $\sigma(0) \neq 0$ then the stream σ has a (unique) multiplicative inverse σ^{-1} in A^ω, satisfying $\sigma^{-1} \times \sigma = [1]$. As usual, we shall often write $1/\sigma$ for σ^{-1} and σ/τ for $\sigma \times \tau^{-1}$. Note that the initial value of the sum, product and inverse of streams is given by the sum, product and inverse of their initial values.

If A is a field, a stream $\rho \in A^\omega$ is *rational* if it is the quotient $\rho = \sigma/\tau$ of two polynomial streams σ and τ with $\tau(0) \neq 0$.

The fundamental theorem of stream calculus allows us to solve *stream differential equations* such as $\sigma' = 2 \times \sigma$ with initial value $\sigma(0) = 1$ by computing $\sigma = \sigma(0) + (X \times \sigma') = 1 + (X \times 2 \times \sigma)$, which leads to the solution $\sigma = 1/(1 - 2X)$. Together with the basic fact that $(X \times \sigma)' = \sigma$, the fundamental theorem also leads to an easy calculation rule for the computation of derivatives: $\sigma' = (\sigma - \sigma(0))'$. This identity makes the computation of stream derivatives often surprisingly simple. For instance, for $\sigma = 1/(1 - X)^2$, we have

$$\sigma' = \ (\frac{1}{(1-X)^2} - 1)' = \ (\frac{2X - X^2}{(1-X)^2})' = \ (\, X \times \frac{2 - X}{(1-X)^2})' = \ \frac{2 - X}{(1-X)^2} \ .$$

For more stream calculations we refer the reader to [Rut05a].

In the remainder of the article we assume A is a field. Strictly speaking, this is not always necessary as some of the constructs, e.g. the stream samplers, do not presume any algebraic structure on A. Nevertheless, in order to be able to freely use the stream calculus we make this assumption. In Section 6 we work in the special case where $A := \mathbb{F}_q$ is a finite field.

3 Periodic Stream Samplers

Traditionally, a substream of an infinite stream $\sigma : \mathbb{N} \to A$ is defined by means of a (strictly) monotone function $f : \mathbb{N} \to \mathbb{N}$: if $n < m$ then $f(n) < f(m)$. Such an *index function* determines an (infinite) substream $S_f(\sigma)$ by

$$S_f(\sigma)(n) = \sigma(f(n))$$

and conversely, any substream of σ determines a unique such monotone function. Assigning to any stream the substream determined by a given monotone function f defines a *stream sampler*

$$S_f : A^\omega \to A^\omega \ , \qquad \sigma \mapsto S_f(\sigma) \ .$$

Periodic stream samplers are such that they produce a substream of a given input stream by repeatedly choosing certain elements and ignoring all others. For instance, the function $even : A^\omega \to A^\omega$ given by

$$even(\sigma) = (\sigma(0), \sigma(2), \sigma(4), \ldots)$$

takes of each incoming two elements the first and ignores the second. We say that *even* has *(input) period* 2 and *(output) block size* 1. Another example is the drop operator $D_4^2 : A^\omega \to A^\omega$ given by

$$D_4^2(\sigma) = (\sigma(0), \sigma(1), \sigma(3), \sigma(4), \sigma(5), \sigma(7), \ldots)$$

which drops from each four incoming elements the third and keeps all the others. Note we always start counting at zero hence $\sigma(2)$, $\sigma(6)$ etc. are dropped. The operator D_4^2 has period 4 and block size 3.

As it turns out, it is somewhat cumbersome to define these and similar such periodic stream samplers by means of monotone index functions. Moreover, it is surprisingly difficult to prove simple general facts such as: The composition of two periodic stream samplers is again a period stream sampler. Therefore, we prefer the following coinductive definition which uses a stream differential equation.

Definition 1. *Let $k, l \in \mathbb{N}$ with $l > 1$ and $1 \leq k \leq l$. Any sequence of k numbers $0 \leq n_0 < n_1 < \cdots < n_{k-1} < l$ determines a* periodic stream sampler $S : A^\omega \to A^\omega$ *of (input) period l and (output) block size k defined by the following stream differential equation:*

$$S(\sigma)^{(k)} = S(\sigma^{(l)})$$

with initial values

$$S(\sigma)(j) = \sigma(n_j) \quad (0 \leq j < k) \ .$$

We do not require period and block size to be minimal. If a stream sampler has period l and block size k then it also has period $2l$ with block size $2k$, etc.

The functions *even* and D_4^2 above are given by

$$even(\sigma)' = even(\sigma'') \ , \qquad even(\sigma)(0) = \sigma(0) \ ,$$

$$D_4^2(\sigma)^{(3)} = D_4^2(\sigma^{(4)}) \ , \quad D_4^2(\sigma)(0) = \sigma(0) \ , \ D_4^2(\sigma)(1) = \sigma(1) \ , \ D_4^2(\sigma)(2) = \sigma(3) \ .$$

Proposition 2. *If $S, T : A^\omega \to A^\omega$ are two periodic stream samplers then so is $T \circ S$.*

Proof. Let S and T satisfy

$$S(\sigma)^{(k)} = S(\sigma^{(l)}) \ , \qquad S(\sigma)(j) = \sigma(n_j) \quad (0 \leq j < k) \ ,$$

$$T(\sigma)^{(p)} = T(\sigma^{(q)}) \ , \qquad T(\sigma)(j) = \sigma(m_j) \quad (0 \leq j < p) \ .$$

We claim that $T \circ S$ is a periodic stream sampler with period $l \times q$ and block size $k \times p$. We define a sequence $i_0, i_1, \ldots, i_{q \times k - 1}$ by

$$i_{(x \times k) + y} = \ (x \times l) + n_y \quad (\text{all} \ \ x, y \ \ \text{with} \ \ 0 \leq x < q, \ \ 0 \leq y < k) \ .$$

Next we define a sequence $0 \leq h_0 < h_1 < \cdots < h_{(k \times p) - 1} < q \times k$ by

$$h_{(x \times p) + y} = \ (x \times q) + m_y \quad (\text{all} \ \ x, y \ \ \text{with} \ \ 0 \leq x < k, \ \ 0 \leq y < p) \ .$$

One readily shows that $T \circ S$ satisfies

$$T \circ S(\sigma)^{(k \times p)} = T \circ S(\sigma^{(l \times q)}) \ , \qquad T \circ S(\sigma)(j) = \sigma(i_{h_j}) \quad (0 \leq j < (k \times p) - 1) \ .$$

□

Next we provide some examples by introducing the family of all drop operators.

Definition 3. *For $l \geq 2$ and $0 \leq i < l$ we define the drop operator*

$$D_l^i : A^\omega \to A^\omega$$

which drops from each input block of size l the i-th element, by the following system of stream differential equations:

$$D_l^{i+1}(\sigma)' = D_l^i(\sigma') \ , \qquad D_l^{i+1}(\sigma)(0) = \sigma(0) \quad (\textit{all} \ \ l \geq 2, \ \ 0 \leq i < l - 1) \ ,$$

$$D_l^0(\sigma)' = D_l^{l-2}(\sigma'') \ , \qquad D_l^0(\sigma)(0) = \sigma(1) \quad (\textit{all} \ \ l \geq 2) \ .$$

Note that for D_4^2, this definition is equivalent with our earlier definition above; also note that $\mathit{even} = D_2^1$.

One of the benefits of coinductive definitions is that they support coinductive proofs. As an example, we prove the so-called *Drop exchange rule* from [Mak08]: for all $l \geq 1$, $0 \leq k \leq h \leq l$,

$$D_{l+1}^h \circ D_{l+2}^k = \ D_{l+1}^k \circ D_{l+2}^{h+1} \ .$$

In order to prove this equality, we define a relation $R \subseteq A^\omega \times A^\omega$ by

$$R = \{ \langle D_{l+1}^h \circ D_{l+2}^k(\sigma), \ D_{l+1}^k \circ D_{l+2}^{h+1}(\sigma) \rangle \mid \ \sigma \in A^\omega \} \ .$$

The equality now follows by coinduction from the fact that $R \cup R^{-1}$ is a stream bisimulation.

Here is another example. It is a basic instance of a *Drop expansion rule* in [Mak08]:

$$D_2^0 = D_4^0 \circ D_5^2 \circ D_6^4 \ .$$

For a proof, we define a relation $R \subseteq A^\omega \times A^\omega$ by

$$\begin{aligned} R = &\{\langle D_2^0(\sigma),\ D_4^0 \circ D_5^2 \circ D_6^4(\sigma)\rangle \mid \sigma \in A^\omega\} \\ \cup &\{\langle D_2^0(\sigma),\ D_4^2 \circ D_5^0 \circ D_6^2(\sigma)\rangle \mid \sigma \in A^\omega\} \\ \cup &\{\langle D_2^0(\sigma),\ D_4^1 \circ D_5^3 \circ D_6^0(\sigma)\rangle \mid \sigma \in A^\omega\}\ . \end{aligned}$$

The equality follows by coinduction from the fact that R is a stream bisimulation.

Returning to the general question of how to define substreams out of a given stream, we present yet another alternative to the use of monotone index functions, which is also well suited for a coinductive approach. Let $2 = \{0, 1\}$ and let 2^ω be the set of bitstreams. Note that there is a trivial field structure on 2 and hence we can apply stream calculus to 2^ω. Consider a bitstream $\alpha \in 2^\omega$ that is not eventually constant 0, i.e., there is no n such that $\alpha^{(n)} = [0]$. Then for any stream $\sigma \in A^\omega$, α defines a substream $S_\alpha(\sigma)$ consisting of those elements $\sigma(n)$ for which $\alpha(n) = 1$. (Note that the condition on α ensures that $S_\alpha(\sigma)$ is again an infinite stream.) Such a stream α acts as an oracle that tells us of any element of σ whether or not it should be included in the substream we are defining.

More formally, we first note that a stream $\alpha \in 2^\omega$ is eventually constant 0 if it is a polynomial. If α is non-polynomial, it is of the form

$$\alpha = X^n \times (1 + X \times \beta)$$

for some $n \geq 0$ and some $\beta \in 2^\omega$ that is again non-polynomial. Now we define $S_\alpha(\sigma)$ by the following system of differential equations, for arbitrary $\sigma \in A^\omega$ and non-polynomials $\alpha \in 2^\omega$:

$$S_\alpha(\sigma)' = S_\beta(\sigma^{(n+1)})\ , \quad S_\alpha(\sigma)(0) = \sigma(n) \qquad (\alpha = X^n \times (1 + X \times \beta))\ .$$

In this manner, any non-polynomial bitstream determines a substream and, conversely, any substream determines a non-polynomial bitstream.

It is now extremely simple to characterise *periodic* stream samplers:

$$S_\alpha \text{ is periodic with period } l \textit{ iff } \quad \alpha^{(l)} = \alpha\ .$$

The (output) block size is determined by the number of 1's in the set $\{\alpha(0), \ldots, \alpha(l-1)\}$.

Composition of stream samplers can be described in terms of composition of the corresponding oracle bitstreams, which we define as follows.

Definition 4. *For all $\alpha, \beta \in 2^\omega$, we define $\beta * \alpha \in 2^\omega$ by the following system of differential equations:*

$$(\beta * \alpha)' = \begin{cases} \beta' * \alpha' & \textit{if } \alpha(0) = 1 \\ \beta * \alpha' & \textit{if } \alpha(0) = 0 \end{cases} \qquad (\beta * \alpha)(0) = \beta(0) \cdot \alpha(0)$$

This composition operator is associative but not commutative and has $1/(1-X)$ as a neutral element: $\sigma * 1/(1-X) = 1/(1-X) * \sigma = \sigma$. It is not difficult to show that

$$S_\beta \circ S_\alpha = S_{\beta * \alpha}\ .$$

An alternative proof of Proposition 2 is now extremely easy: it follows from the fact that $\alpha^{(n)} = \alpha$ and $\beta^{(m)} = \beta$ imply $(\beta * \alpha)^{(n \times m)} = \beta * \alpha$.

Let us conclude this section with an example illustrating how one can reason about stream sampler composition in terms of stream calculus applied to the corresponding oracle streams. Periodic oracle bitstreams are always of the form

$$\frac{a_0 + a_1 X + a_2 X^2 + \cdots + a_{l-1} X^{l-1}}{1 - X^l}$$

for $a_0, a_1, a_2, \ldots, a_{l-1} \in 2$, not all 0. For our drop operators, for instance, one has

$$D^i_l = S_{\alpha^i_l} \quad \text{with} \quad \alpha^i_l = (1 + X + \cdots + X^{i-1} + X^{i+1} + \cdots + X^{l-1})/(1 - X^l)$$

The equality $D^0_2 = D^0_4 \circ D^2_5 \circ D^4_6$, which we proved above by coinduction, can also be deduced from the following computation in stream calculus on the corresponding oracle bitstreams:

$$\begin{aligned} \alpha^0_4 * \alpha^2_5 * \alpha^4_6 &= \frac{X + X^2 + X^3}{1 - X^4} * \frac{1 + X + X^3 + X^4}{1 - X^5} * \frac{1 + X + X^2 + X^3 + X^5}{1 - X^6} \\ &= \frac{X + X^3 + X^4}{1 - X^5} * \frac{1 + X + X^2 + X^3 + X^5}{1 - X^6} \\ &= \frac{X}{1 - X^2} = \alpha^0_2 \ . \end{aligned}$$

The work goes in the computation of the stream compositions, using the differential equation of Definition 4. This may be bothersome by hand but can easily be automated.

4 Splitting and Merging

All periodic stream samplers and, more generally, many periodic stream transformers that not necessarily preserve the order of the elements in a stream, can be obtained by splitting and merging streams. In this section, we introduce the operators of take and zip, with which streams can be split and merged, and we present a few basic laws about them.

Definition 5. *i) For $l \geq 2$ and $0 \leq i < l$, the* take operator $T^i_l : A^\omega \to A^\omega$ *is defined by the following stream differential equation:*

$$T^i_l(\sigma)' = T^i_l(\sigma^{(l)}) \ , \qquad T^i_l(\sigma)(0) = \sigma(i) \ .$$

ii) For $k \geq 1$ and streams $\sigma_0, \ldots \sigma_{k-1} \in A^\omega$, the zip operator $Z_k : (A^\omega)^k \to A^\omega$ *is defined by the stream differential equation*

$$Z_k(\sigma_0, \ldots, \sigma_{k-1})' = Z_k(\sigma_1, \ldots, \sigma_{k-1}, \sigma'_0) \ , \quad Z_k(\sigma_0, \ldots, \sigma_{k-1})(0) = \sigma_0(0) \ .$$

(Note that $\sigma_0, \ldots, \sigma_{k-1}$ above are *streams*, not *elements* of streams, which for a stream σ we denote by $\sigma(0)$, $\sigma(1)$, etc.) Examples are

$$T_3^2(\sigma) = (\sigma(2), \sigma(5), \sigma(8), \ldots) \ ,$$

$$Z_2(\sigma, \tau) = (\sigma(0), \tau(0), \sigma(1), \tau(1), \sigma(2), \tau(2), \ldots) \ .$$

As suggested by the latter, it is easy to see (by induction) that in general if $0 \leq r \leq k-1$ then

$$Z_k(\sigma_0, \ldots, \sigma_{k-1})(kn + r) = \sigma_r(n) \ . \tag{4.1}$$

Any periodic stream sampler can be expressed in terms of take and zip. With S as in Definition 1, we have

$$S(\sigma) = Z_k(T_l^{n_0}(\sigma),\, T_l^{n_1}(\sigma),\, \ldots,\, T_l^{n_{k-1}}(\sigma)) \ .$$

More generally, we can define with take and zip periodic stream transformers that not merely produce substreams but that can change also the order of the elements. For instance, we can define the operation $Rev_k : A^\omega \to A^\omega$ of stream *reverse*, for any $k \geq 1$, by

$$Rev_k(\sigma) = \ Z_k(T_k^{k-1}(\sigma),\, T_k^{k-2}(\sigma),\, \ldots,\, T_k^0(\sigma)) \ .$$

For instance,

$$Rev_3(\sigma) = (\sigma(2), \sigma(1), \sigma(0), \sigma(5), \sigma(4), \sigma(3), \ldots) \ .$$

Next we present a few basic laws for take and zip that will allow us to prove elementary properties on stream transformers by equational reasoning. All of the identities below can easily be proved by coinduction.

Proposition 6. *For all $k \geq 1$, $l \geq 2$, $0 \leq i < l$,*

$$Z_k(T_k^0(\sigma), \ldots, T_k^{k-1}(\sigma)) = \ \sigma \ ,$$

$$T_l^i(Z_l(\sigma_0, \ldots, \sigma_{l-1})) = \ \sigma_i \ ,$$

$$T_l^i(\sigma) = \ Z_k(T_{k\times l}^i(\sigma),\, T_{k\times l}^{l+i}(\sigma),\, \ldots,\, T_{k\times l}^{(k-1)\times l+i}(\sigma)) \ .$$

Let us illustrate these identities with an equational proof of our earlier example, the Drop expansion rule: for all $\sigma \in A^\omega$,

$$D_2^0(\sigma) = D_4^0 \circ D_5^2 \circ D_6^4(\sigma) \ .$$

Let $\tau = D_6^4(\sigma)$. We have

$$\tau = \ D_6^4(\sigma) = \ Z_5(\, T_6^0(\sigma),\, T_6^1(\sigma),\, T_6^2(\sigma),\, T_6^3(\sigma),\, T_6^5(\sigma)) \ .$$

Next let $\rho = D_5^2 \circ D_6^4(\sigma)$; it satisfies

$$\begin{aligned} \rho = D_5^2(\tau) &= \ Z_4(\, T_5^0(\tau),\, T_5^1(\tau),\, T_5^3(\tau),\, T_5^4(\tau)) \\ &= Z_4(\, T_6^0(\sigma),\, T_6^1(\sigma),\, T_6^3(\sigma),\, T_6^5(\sigma)) \ . \end{aligned}$$

Finally, we compute

$$\begin{aligned} D_4^0 \circ D_5^2 \circ D_6^4(\sigma) = D_4^0(\rho) &= Z_3(\, T_4^1(\rho),\, T_4^2(\rho),\, T_4^3(\rho)\,) \\ &= Z_3(\, T_6^1(\sigma),\, T_6^3(\sigma),\, T_6^5(\sigma)\,) \\ &= T_2^1(\sigma) = D_2^0(\sigma) \ . \end{aligned}$$

As a second example, we prove $Rev_3 \circ Rev_3(\sigma) = \sigma$. Putting $\tau = Rev_3(\sigma)$,

$$\tau = Rev_3(\sigma) = Z_3(\, T_3^2(\sigma),\, T_3^1(\sigma),\, T_3^0(\sigma)\,) \ .$$

It follows that

$$\begin{aligned} Rev_3 \circ Rev_3(\sigma) = Rev_3(\tau) &= Z_3(\, T_3^2(\tau),\, T_3^1(\tau),\, T_3^0(\tau)\,) \\ &= Z_3(\, T_3^0(\sigma),\, T_3^1(\sigma),\, T_3^2(\sigma)\,) = \sigma \ . \end{aligned}$$

In the above, we have illustrated that the operators of take and zip are interesting because they can express all periodic stream samplers and because they can moreover be used to define stream transformers that have a periodic behaviour but that are not stream samplers. We have not given a general definition of periodic stream transformer. We shall come back to this point later.

5 Preserving Rationality

In this section, we show that the result of applying the operators of take and zip to rational streams in A^ω is again rational. We shall use the following definition from [Rut05a, p.109].

Definition 7. *For $\sigma \in A^\omega$ and $\rho \in A^\omega$ with $\rho(0) = 0$, we define the stream σ* applied to *ρ, written as $\sigma(\rho)$, by the following system of differential equations:*

$$\sigma(\rho)' = \sigma'(\rho) \times \rho' \ , \qquad \sigma(\rho)(0) = \sigma(0) \ .$$

Recall from [Rut05a] that every stream $\sigma \in A^\omega$ can be written as an infinite sum

$$\sigma = \sigma(0) + (\sigma(1) \times X) + (\sigma(2) \times X^2) + \cdots \ .$$

We may now think of $\sigma(\rho)$ as the stream that results from the above infinite sum by replacing every X by ρ (the condition $\rho(0) = 0$ will ensure that the resulting infinite sum is well-defined). In fact, there is the following identity:

$$\sigma(\rho) = \sigma(0) + (\sigma(1) \times \rho) + (\sigma(2) \times \rho^2) + \cdots \ .$$

This reminds one of formal power series and (generating) function application (cf. [GKP94]); note that the definition and identities above all live in stream calculus, where X is a constant stream and not a function variable.

If σ is polynomial and ρ is rational (with $\rho(0) = 0$) then $\sigma(\rho)$ is rational. Since for polynomials π and τ with $\tau(0) \neq 0$, one can easily show that

$$\frac{\pi}{\tau}(\rho) = \frac{\pi(\rho)}{\tau(\rho)} \ ,$$

it follows that if σ and ρ are rational then so is $\sigma(\rho)$. We shall be using the above mostly for the case that $\rho = X^n$, for some $n \geq 1$. For instance, we have

$$\frac{X}{(1-X)^2}(X^3) = \frac{X^3}{(1-X^3)^2} \ .$$

Since $X/(1-X)^2 = (0, 1, 2, \ldots)$ it follows that

$$\frac{X^3}{(1-X^3)^2} = (0, 0, 0, 1, 0, 0, 2, 0, 0, \ldots) \ .$$

We are now ready to formulate our first preservation result. We remark that Propositions 8 and 10 below can be found in [BR88]. Our proofs are different: in Proposition 8 the novelty lies in our use of coinduction proof principle; regarding Proposition 10 we give a rather elementary proof while the proof in [BR88] is based on Kleene–Schützenberger theorem.

Proposition 8. *The function zip preserves rationality: if $\sigma_0, \ldots, \sigma_{k-1} \in A^\omega$ are rational, for $k \geq 1$, then so is $Z_k(\sigma_0, \ldots, \sigma_{k-1})$.*

Proof. The proposition follows from the identity

$$Z_k(\sigma_0, \ldots, \sigma_{k-1}) = \sigma_0(X^k) + (X \times \sigma_1(X^k)) + \cdots + (X^{k-1} \times \sigma_{k-1}(X^k))$$

which can easily be proved by coinduction. □

Next we show that the take operators preserve rationality as well. We shall use the following lemma; it has an easy proof by coinduction which we omit here.

Lemma 9. *Let $l \geq 2$ and $0 \leq i < l$.*

(a) T_l^i is linear: for all $r, s \in A$, $\sigma, \tau \in A^\omega$,

$$T_l^i((s \times \sigma) + (t \times \tau)) = (s \times T_l^i(\sigma)) + (t \times T_l^i(\tau)) \ .$$

(b) For $1 \leq i \leq l$ and $\sigma \in A^\omega$,

$$T_l^i(X \times \sigma) = T_l^{i-1}(\sigma) \ , \qquad T_l^0(X \times \sigma) = X \times T_l^{l-1}(\sigma) \ .$$

Proposition 10. *The function take preserves rationality: if $\sigma \in A^\omega$ is rational then so is $T_l^i(\sigma)$, for all $l \geq 2$ and $0 \leq i < l$.*

Proof. By Lemma 9, it is sufficient to prove the proposition for streams of the form $1/\sigma$, with σ polynomial and $\sigma(0) \neq 0$. So let $\sigma = s_0 + s_1 X + \cdots + s_d X^d$ be a polynomial stream, for $d \geq 0$ and $s_0, s_1, \ldots, s_d \in A$ with $s_0 \neq 0$. One can prove by induction that for any $l \geq 0$, the l-th stream derivative of $1/\sigma$ is of the form

$$(1/\sigma)^{(l)} = (r_0 + r_1 X + \cdots + r_{d-1} X^{d-1}) \times 1/\sigma$$

for certain $r_0, \ldots, r_{d-1} \in A$. Now for $l \geq 2$ and $0 \leq i < l$, we have

$$\begin{aligned} T_l^i(1/\sigma)' &= T_l^i(\,(1/\sigma)^{(l)}\,) \quad \text{[by definition]} \\ &= T_l^i(\,(r_0 + \cdots + r_{d-1}X^{d-1}) \times 1/\sigma\,) \quad \text{[by the equality above]} \\ &= (\rho_0 \times T_l^0(1/\sigma)\,) + \cdots + (\rho_{l-1} \times T_l^{l-1}(1/\sigma)\,) \end{aligned}$$

for certain rational streams $\rho_0, \ldots, \rho_{l-1} \in A^\omega$, where the last equality follows from Lemma 9. Multiplying the equation by X and adding $(1/\sigma)(i)$ to both sides gives

$$\begin{aligned} &T_l^i(1/\sigma) \\ &= T_l^i(1/\sigma)(0) + (X \times T_l^i(1/\sigma)') \quad \text{[by the fundamental theorem, Section 2]} \\ &= (1/\sigma)(i) + \big(X \times \big((\rho_0 \times T_l^0(1/\sigma)\,) + \cdots + (\rho_{l-1} \times T_l^{l-1}(1/\sigma))\big)\big) \\ &= (1/\sigma)(i) + (X \times \rho_0 \times T_l^0(1/\sigma)\,) + \cdots + (X \times \rho_{l-1} \times T_l^{l-1}(1/\sigma)\,)\ . \end{aligned}$$

We have an equation of this form for all i with $0 \leq i < l$. Thus we have obtained a system of l equations in l unknowns: $T_l^0(1/\sigma), \ldots, T_l^{l-1}(1/\sigma)$, where all the occurrences of the unknowns on the right are multiplied by a rational stream of the form $X \times \rho$. Such a system of what could be called *guarded* equations can easily be seen to have rational streams as solutions, essentially by standard linear algebraic reasoning. □

Corollary 11. *If an operator is built by function composition from: constant streams $[r]$ (for $r \in A$), X, sum $+$, convolution product $\times$, convolution inverse $(-)^{-1}$, and the zip and take operators Z_k and T_l^i, then it preserves rationality.*

Proof. For the constants, sum, product and inverse, this is trivial and for zip and take, we have Propositions 8 and 10. □

Here are some examples. Let $\sigma = 1/(1-X)^2 = (1, 2, 3, \ldots)$. We will compute

$$\alpha = T_3^0(\sigma)\ , \qquad \beta = T_3^1(\sigma)\ , \qquad \gamma = T_3^2(\sigma)\ .$$

In the computation below, we shall be using the following equalities:

$$\sigma^{(3)} = \frac{4-3X}{(1-X)^2}\ , \qquad T_3^0(X\times\sigma) = X\times\gamma\ , \qquad T_3^1(X\times\sigma) = \alpha\ , \qquad T_3^2(X\times\sigma) = \beta\ .$$

For α, we compute as follows:

$$\alpha' = T_3^0(\sigma^{(3)}) = T_3^0\Big(\frac{4-3X}{(1-X)^2}\Big) = 4\alpha - (3X \times \gamma)\ .$$

Using the fundamental theorem and $\alpha(0) = 1$ gives

$$\alpha = 1 + (4X \times \alpha) - (3X^2 \times \gamma)\ .$$

Similar computations lead to equations for β and γ:

$$\beta = 2 + (4X \times \beta) - (3X \times \alpha)\ ,$$

$$\gamma = 3 + (4X \times \gamma) - (3X \times \beta) \ .$$

Solving this system of three equations gives

$$\alpha = \frac{1+2X}{(1-X)^2} \ , \qquad \beta = \frac{2+X}{(1-X)^2} \ , \qquad \gamma = \frac{3}{(1-X)^2} \ .$$

As a next example, we will compute $Rev_3(\sigma)$, as follows:

$$\begin{aligned} Rev_3(\sigma) &= Z_3(T_3^2(\sigma),\, T_3^1(\sigma),\, T_3^0(\sigma)) \quad [\text{definition } Rev_3] \\ &= Z_3\left(\frac{3}{(1-X)^2},\, \frac{2+X}{(1-X)^2},\, \frac{1+2X}{(1-X)^2}\right) \\ &= \frac{3}{(1-X^3)^2} + X \times \frac{2+X^3}{(1-X^3)^2} + X^2 \times \frac{1+2X^3}{(1-X^3)^2} \quad [\text{Proposition 8}] \\ &= \frac{3-X-X^2+2X^3}{(1-X)^2(1+X+X^2)} \ . \end{aligned}$$

6 Preserving Algebraicity

Corollary 11 shows that starting with a rational stream and applying some 'basic' operations we stay in the realm of rational streams. But there is a somewhat larger class of streams that is preserved under some of these operations, namely the class of algebraic streams defined below.

Algebraicity is a notion that should be defined over other algebraic structures. In this section we study algebraicity over finite fields. For $q \geq 1$ let $\mathbb{F}_q$ be the finite field with q elements (note that $\mathbb{F}_q$ has cardinality p^n for some prime p [Hun80]). A univariate polynomial in X is a polynomial of the form $a_0 + a_1X + \cdots + a_kX^k$ where $a_i \in \mathbb{F}_q, a_k \neq 0$. Subsequently by $\mathbb{F}_q(X)$ we denote the *field of fractions of polynomials in* X, i.e., $\pi(X) \in \mathbb{F}_q(X)$ means there are univariate polynomials $\pi_1(X), \pi_2(X)$ with coefficients in $\mathbb{F}_q$ such that $\pi(X) = \pi_1(X)/\pi_2(X)$.

Definition 12. *A stream* $\sigma \in \mathbb{F}_q^\omega$ *is* algebraic *over* $\mathbb{F}_q(X)$ *if there are* $A_i \in \mathbb{F}_q(X), A_k \neq 0$ *such that* $A_0 + A_1\sigma + \ldots + A_k\sigma^k = 0$.[2]

As an example, the stream $\sigma \in \mathbb{F}_2^\omega$ for which

$$X^3 + \frac{1}{1-X}\sigma + \frac{X+1}{1-X^2}\sigma^2 = 0 \ ,$$

is algebraic over $\mathbb{F}_2(X)$.

This definition is borrowed from the theory of formal power series [Fog02] and is motivated by the fact that σ can be considered as the sequence of coefficients of a formal power series. Following Section 2, by taking $A := \mathbb{F}_q$ we can obtain the stream calculus on $\mathbb{F}_q^\omega$. As a consequence the left hand side of expression

[2] In fact we can restrict the coefficients A_i to *univariate polynomials* instead of fractions.

above can be interpreted in two ways: as a stream in the stream calculus where $X = (0, 1, 0, \ldots)$ as in Section 2 or as a formal power series in the ring of formal power series with one variable X. It can easily be observed that each rational stream in $\mathbb{F}_q^\omega$ is algebraic. The converse does not always hold. In next section we give an example of an algebraic stream that is not rational, namely the *Prouhet–Thue–Morse* sequence. There are also streams that are not algebraic, a simple example being the Fibonacci sequence [Fog02, § 1.2.2]. But in general, the so called *automatic* streams, i.e., streams that are 'computable' by a class of transducers similar to Mealy machines[3], can be shown to be algebraic [Fog02].

We state a useful criterion, originally from [Chr79], that is usually used as an intermediate step in relating algebraic and automatic sequences but here we will use it on its own. Our formulation follows [Fog02, Theorem 3.2.1].

Definition 13. *Let $\sigma \in \mathbb{F}_q^\omega$. Then the* q-kernel *of σ is the set of subsequences of σ defined as*

$$N_q(\sigma) = \{\lambda n.\sigma(q^s n + r) \mid s \geq 0 \ , \ 0 \leq r \leq q^s - 1\} \ . \tag{6.1}$$

Here $\lambda n.f(n)$ is the notation for the sequence whose nth element is $f(n)$.

Theorem 14 (Christol). *A stream $\sigma \in \mathbb{F}_q^\omega$ is algebraic over $\mathbb{F}_q(X)$ if and only if the q-kernel $N_q(\sigma)$ of σ is finite.*

By applying this theorem we can obtain what can be considered as counterparts of Propositions 8 and 10 above. First, we have the following which resembles Proposition 10. This one is an easy consequence and is also mentioned in [Fog02], so we skip the proof.

Proposition 15. *The function take preserves algebraicity for streams over a finite alphabet: if $\sigma \in \mathbb{F}_q^\omega$ is algebraic over $\mathbb{F}_q(X)$ then so is $T_l^i(\sigma)$, for all $l \geq 2$ and $0 \leq i < l$.*

For *zip* we first need to define a notion based on q-kernels.

Definition 16. *Let $\sigma_0, \ldots, \sigma_{h-1} \in \mathbb{F}_q^\omega$ (where $h > 0$). Then* h-fold q-kernel *of $\sigma_0, \ldots, \sigma_{h-1}$ is the set of sequences defined as*

$$N_q^{(h)}(\sigma_0, \ldots, \sigma_{h-1}) = \{Z_h(\tau_0, \ldots, \tau_{h-1}) \mid \forall i \exists j, \ \tau_i \in N_q(\sigma_i)\} \ . \tag{6.2}$$

Note that we have the following trivial properties.

Proposition 17

i) If $\varsigma_0, \ldots, \varsigma_{h-1}$ is a possibly repetitive sequence such that $\varsigma_i \in \{\sigma_0, \ldots, \sigma_{h-1}\}$, then $N_q^{(h)}(\varsigma_0, \ldots, \varsigma_{h-1}) \subseteq N_q^{(h)}(\sigma_0, \ldots, \sigma_{h-1})$.

ii) If q-kernel of each of $\sigma_0, \ldots, \sigma_{h-1}$ is finite then the h-fold q-kernel of them is finite.

[3] This is a very informal description. The precise definition of automatic sequences can be found in [AS03].

We use these facts for proving that *zip* preserves algebraicity. To the best of our knowledge this result is new.

Proposition 18. *The function zip preserves algebraicity for streams over a finite alphabet: if $\sigma_0, \ldots, \sigma_{h-1} \in \mathbb{F}_q^\omega$ (where $h > 0$) are algebraic over $\mathbb{F}_q(X)$, then so is $Z_h(\sigma_0, \ldots, \sigma_{h-1})$.*

Proof. Let $\tau := Z_h(\sigma_0, \ldots, \sigma_{h-1})$. We show that

$$N_q(\tau) \subset N_q^{(h)}(\sigma_0, \ldots, \sigma_{h-1}) \ . \tag{6.3}$$

The result then will follow from Theorem 14, since the right hand side is finite.

To prove (6.3) assume $\alpha \in N_q(\tau)$. Then $\alpha \equiv \lambda n.\tau(q^s n + r)$ for some s, r as in (6.1). Assume, using division algorithm, that $q = d_0 h + r_0$ and $r = d_1 h + r_1$. Furthermore by applying (4.1) it can easily be seen that

$$\alpha \equiv Z_h(\lambda n.\tau(hnq^s + r), \lambda n.\tau((hn+1)q^s + r), \cdots, \lambda n.\tau((hn + (h-1))q^s + r)) \ .$$

So α is the *zip* of h streams each of which of the form $\tau((hn+k)q^s + r)$ where $k \leq h-1$. Again using the division algorithm assume $kr_0^s + r_1 = d_k h + r_k$. Then

$$\begin{aligned}(hn+k)q^s + r &= hnq^s + k(d_0 h + r_0)^s + d_1 h + r_1 \\ &= hnq^s + k(d_0^s h^s + s d_0^{s-1} h^{s-1} r_0 + \cdots + s d_0 h r_0^{s-1} + r_0^s) + d_1 h + r_1 \\ &= h(nq^s + k d_0^s h^{s-1} + s k d_0^{s-1} h^{s-2} r_0 + \cdots + s k d_0 r^{s-1} + d_1) + d_k h + r_k \\ &= h(nq^s + U_k) + r_k \ ,\end{aligned}$$

where

$$U_k = k d_0^s h^{s-1} + s k d_0^{s-1} h^{s-2} r_0 + \cdots + s k d_0 r^{s-1} + d_1 + d_k \ .$$

From this and using the property of zip in (4.1) we get

$$\lambda n.\tau((hn+k)q^s + r) \equiv \lambda n.\tau(h(nq^s + U_k) + r_k) \equiv \lambda n.\sigma_{r_k}(nq^s + U_k) \ .$$

It remains to be checked whether $U_k < q^s$. But this is evident because

$$\begin{aligned} hU_k &= k(q^s - r_0^s) + d_1 h + d_k h \\ &= kq^s + r - r_k \\ &\leq (h-1)q^s + r \\ &< hq^s \ .\end{aligned}$$

Therefore defining $\upsilon_k := \lambda n.\sigma_{r_k}(nq^s + U)$ we obtain $\upsilon_0 \in N_q(\sigma_{r_0}), \ldots, \upsilon_{h-1} \in N_q(\sigma_{r_{h-1}})$ such that

$$\alpha \equiv Z_h(\upsilon_0, \ldots, \upsilon_{h-1}) \ .$$

Hence, by (6.2) and Proposition 17 we have $\alpha \in N_q^{(h)}(\sigma_0, \ldots, \sigma_{h-1})$. □

In general, the *zip* of algebraic sequence need not be algebraic over a field whose cardinality is the number of arguments of *zip*. This is a consequence of the following result in [Cob69] where it is stated in terms of automatic sequences. Here we rephrase it in terms of algebraicity over finite fields.

Theorem 19 (Cobham). *Let* $\sigma \in \mathbb{F}^\omega_{q_0} \cap \mathbb{F}^\omega_{q_1}$ *be algebraic over* two *fields* $\mathbb{F}_{q_0}(X)$ *and* $\mathbb{F}_{q_1}(X)$. *Then either* σ *is rational or* q_0 *and* q_1 *are powers of the same prime number.*

According to this theorem if $\sigma_0, \sigma_1, \sigma_2 \in \mathbb{F}^\omega_2$ are non-rational binary streams that are algebraic over $\mathbb{F}_2(X)$ (e.g. the sequence Ψ defined in next section) then $Z_3(\sigma_0, \sigma_1, \sigma_2)$ cannot be algebraic over $\mathbb{F}_3(X)$.

Finally, we remark that the sum of two algebraic streams is algebraic. The proof is a straightforward application of Theorem 14, together with a similar construct to the one in (6.2).

7 Stream Circuits

We briefly recall the correspondence between rational streams (of real numbers) and so-called stream circuits built from adder, copier, register and multiplier gates. Then we propose to look at stream circuits built from this set of gates extended with basic gates for splitting and merging. We study their behaviour by describing how they act on input streams of real numbers. For circuits without feedback, it will be immediate that they preserve rationality. For feedback circuits, the situation turns out to be more complicated.

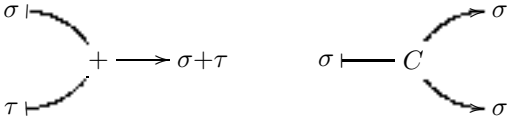
Stream circuits [Rut05b] are data flow networks that act on streams of inputs (here real numbers) and produce streams of outputs. They are built out of four types of basic gates by means of composition, which amounts simply to connecting (single) output ends to (single) input ends. Below we describe the basic gates and their input-output behaviour. An *r-multiplier*, for $r \in A$, transforms an input stream $\sigma \in A^\omega$ into $[r]\times\sigma$:

$$\sigma \longmapsto \xrightarrow{\ r\ } [r]\times\sigma$$

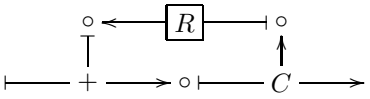
which amounts to the element-wise multiplication of the input values with r. A *register* (with initial value 0) takes an input stream σ

$$\sigma \longmapsto \boxed{R} \longrightarrow (X \times \sigma)$$

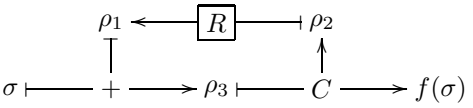
and outputs it with one step delay, after having output the initial value 0 first. An *adder* takes two input streams σ and τ and outputs the stream consisting of their element-wise addition; and a *copier* simply copies input streams into output streams:



Stream circuits are then built by composing various basic gates. Here is a simple example of a circuit with feedback:



For an input stream $\sigma \in A^\omega$, we can compute the output stream as a function $f(\sigma)$ of σ as follows. With the three internal composition nodes of the circuit, we associate streams $\rho_1, \rho_2, \rho_3 \in A^\omega$:



For each of the three basic gates used in this circuit, we have an equation:

$$\rho_1 = X \times \rho_2 \ , \qquad \rho_3 = \sigma + \rho_1 \ , \qquad \rho_2 = \rho_3 = f(\sigma) \ .$$

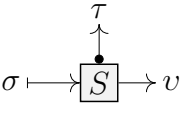
Eliminating the streams ρ_1, ρ_2 and ρ_3 from this system of equations, we find

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

In [Rut05b, Theorem 4.25], it is shown that every (finite) circuit possibly with feedback loops (which always have to pass through at least one register), compute stream functions $f : A^\omega \to A^\omega$ of the form: $f(\sigma) = \rho \times \sigma$, for all σ and some fixed rational stream ρ; conversely, every such function is implemented by some finite circuit.

Next we introduce new basic gates for the splitting and merging of streams.

A *splitter* gate in our setting is a gate with one input and two output ends:



It transforms an input stream $\sigma \in \mathbb{R}^\omega$ to streams τ, υ such that

$$\tau = D_2^1(\sigma) = T_2^0(\sigma) \ , \qquad \upsilon = D_2^0(\sigma) = T_2^1(\sigma) \ .$$

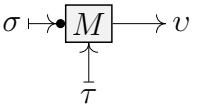
Note that $\tau = \mathit{even}(\sigma)$ and $\upsilon = \mathit{even}(\sigma')$ (where *even* is defined in Section 3). We define

$$\mathit{odd}(\sigma) := \mathit{even}(\sigma') \ .$$

Hence the splitters transforms σ to $\mathit{even}(\sigma)$ and $\mathit{odd}(\sigma)$.

The splitter is different from the previous ports (in particular copier) in that only one of its outgoing ports is active at any time. This means when a data element belonging to τ is being output, the port outputting υ is pending. Moreover, the active output port alternates with each data consumed from σ. The bullet on one of the output ports denotes the port that activates in the very beginning. This confirms the fact that $\tau = \mathit{even}(\sigma)$.

A *merger* gate is a gate with two inputs and one output end.



It transforms two input streams $\sigma, \tau \in \mathbb{R}^\omega$ to a stream υ such that

$$\upsilon = Z_2(\sigma, \tau) \ .$$

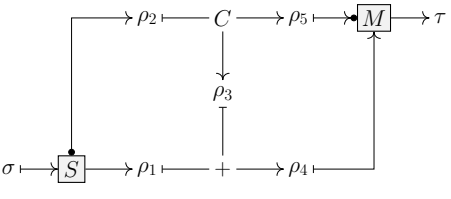
In contrast with the splitter gate, in a merger only one of the inputs is activated at a time. The active input port alternates with each data output. Again the bullet denotes the port that is activated in the very beginning, i.e., the one that contributes to υ_0.

It is clear that merger and splitter can be composed with each other and with the previously defined gates to form compound circuits. We call such a circuit an *extended stream circuit*. The functions $f(\sigma) = \rho \times \sigma$, for constant stream ρ, that are realisable by well-formed stream circuits are instances of *causal* functions on streams [Rut05b]. These are functions that output a data item after each input. Since each gate of stream circuit is causal their composition is causal too. However, introducing splitter and merger into the extended stream circuits leads to *overconsumption* (splitter) or *overproduction* (merger). So there will be data queues behind causal gates. Hence we need to assume the following important rule:

> *The connecting lines in extended stream circuits behave like unbounded FIFO buffers.*

This is similar to the framework of Kahn Networks [Kah74].

Simple feed-forward extended stream circuits can easily be analysed using the same method used for stream circuit. As an example consider the following circuit [Mak08, § 4].



First note that,

$$\begin{aligned}
\rho_1 &= \mathit{odd}(\sigma) \ , \\
\rho_2 = \rho_3 = \rho_5 &= \mathit{even}(\sigma) \ , \\
\rho_4 &= \mathit{odd}(\sigma) + \mathit{even}(\sigma) \ , \\
\tau &= Z_2(\mathit{even}(\sigma), \mathit{odd}(\sigma) + \mathit{even}(\sigma)) \ .
\end{aligned}$$

Assume we input the stream $\sigma = X/(1-X)^2 = (0, 1, 2, \cdots)$ to the above circuit. It can easily be shown that (cf. the example at the end of Section 5),

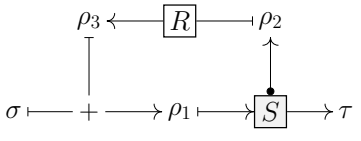
$$\mathit{even}(\sigma) = \frac{2X}{(1-X)^2} \ , \qquad \mathit{odd}(\sigma) = \frac{1+X}{(1-X)^2} \ .$$

Subsequently we derive

$$\begin{aligned}\tau &= Z_2(\frac{2X}{(1-X)^2}, \frac{1+3X}{(1-X)^2}) \\ &= \frac{2X^2}{(1-X^2)^2} + \frac{X+3X^3}{(1-X^2)^2} = \frac{X(1+2X+3X^2)}{(1-X)^2(1+X)^2} \ .\end{aligned}$$

Evidently, by sequencing splitters and mergers one can synthesise feed-forward circuits for calculating dyadic (2^n-ary) *take* and *zip* and functions. I.e., we can build circuits for calculating $T^l_{2^n}$ Z_{2^n}. This suggests that by adding new splitter and merger gates with p input and output ports, where p is a prime number, we can synthesise circuits for calculating general *take* and *zip* functions T^l_n and Z_n. We do not consider this issue in the present paper.

While feed-forward extended stream circuits are relatively easy to analyse, allowing feed-back will complicate the matter. First of all we need to formulate well-formedness rules with respect to the topology of the circuit, whose purpose would be to prevent overconsumption from happening (overproduction is not a problem, since we assume that connecting lines are buffers). Intuitively this means that for any possible path in the circuit, splitters should be directly connected to the global input or be preceded by appropriate number of mergers. In future work we plan to make such rules more formal. For now we give an example a non well-formed circuit demonstrating the problem of overconsumption.



In the circuit above, assuming there is a flow, one can take the second derivative of the behavioural equations for ρ_1 and obtain the contradiction in the form of following identity.

$$\rho_1(2) = \sigma(2) + \rho_1(2) \ .$$

We conclude this section by giving an example of a non-rational stream that can be calculated using the extended stream circuits. This will demonstrate that adding splitter and merger will indeed extend the class of definable streams with respect to those of the ordinary stream calculus. Our example is the *Prouhet–Thue–Morse* sequence which is an algebraic non-rational[4] stream over $\mathbb{F}_2(X)$. The stream, which we denote by Ψ is given by the following behavioural differential equations.

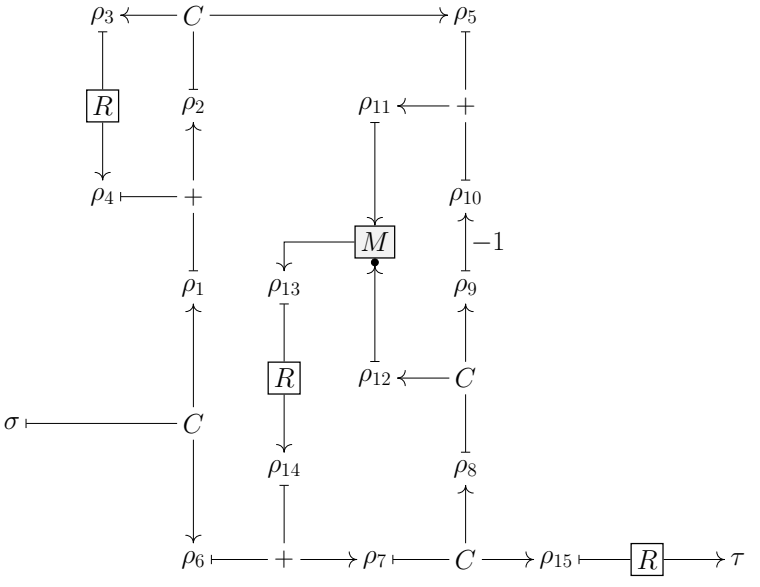
$$\begin{aligned}&\Psi(0) = 0 \ , \qquad \Psi'(0) = 1 \ , \\ &\Psi'' = Z_2(\Psi', \overline{\Psi'}) \ ;\end{aligned}$$

where $\overline{\sigma}$ is the bit-wise negation of σ itself defined as

$$\overline{\sigma}(0) = \neg\sigma(0) \ , \qquad \overline{\sigma}' = \overline{\sigma'} \ .$$

[4] Proof of this fact can be found in [Fog02].

Consider following extended circuit which contains only one merger.



Note that the -1-multiplier is meaningful since we are working in a field. Then by calculating the intermediate values ρ_i one observes that:

$$
\begin{aligned}
&\rho_1 = \rho_6 = \sigma \ , \\
&\rho_2 = \rho_3 = \rho_5 = \sigma + X \times \rho_2 = \frac{1}{1-X} \times \sigma \ , \\
&\rho_4 = \frac{X}{1-X} \times \sigma \ , \\
&\rho_7 = \rho_8 = \rho_9 = \rho_{12} = \rho_{15} = \sigma + \rho_{14} \ , \\
&\rho_{10} = -\rho_7 \ , \\
&\rho_{11} = \frac{1}{1-X} \times \sigma - \rho_7 \ , \\
&\rho_{13} = Z_2(\rho_7, \frac{1}{1-X} \times \sigma - \rho_7) \ , \\
&\rho_{14} = X \times Z_2(\rho_7, \frac{1}{1-X} \times \sigma - \rho_7) \ , \\
&\tau = X \times \rho_7 \ .
\end{aligned}
$$

Form here we can obtain

$$\rho_7 = \sigma + X \times Z_2(\rho_7, \frac{1}{1-X} \times \sigma - \rho_7) \ .$$

Hence if $\sigma = [1] = (1, 0, 0, \cdots)$ is input to this circuit then $\tau = \Psi$.

8 Discussion and Future Work

We have studied various data independent operations for partitioning, projecting or merging streams. These operations are usually studied in the context of dataflow programming, while we showed that the operations and many of their properties can be defined using elements of *stream calculus*, namely behavioural differential equations for definitions and coinduction proof principle for proofs. Furthermore we focused on *take* and *zip* operations, for merging and splitting of data that are widely used elements in dataflow programming [BŞ01, Mak08] and models of concurrency [Arb04]. We dealt with the fact that splitting and merging preserves well behaved and well patterned class of streams namely rational and algebraic streams. While some of those results were known in the literature, we present them in the framework of stream calculus. Finally we showed how adding two new gates, namely dyadic merger and splitter will enlarge the class of streams that are realisable using stream circuits to beyond rational streams and into the realm of algebraic streams.

There are several issues and directions for future work.

Automated coinduction proofs. In Section 3 we showed how to use coinduction to prove the Drop exchange rule by finding a bisimulation. There are in fact tools for automatically finding bisimulation, e.g. the CIRC tool [LR07]. We applied CIRC and it could drive the rule $D_2^0 = D_4^0 \circ D_5^2 \circ D_6^4$. The CIRC tool uses a special technique called *circular coinduction*, a partial decision procedure, whose success depends on the type of bisimulation to be found. Our goal is to further investigate the different types of bisimulation that will arise in *Periodic Drop Take Calculus (PDTCS)* of Mak [Mak08] and examine the applicability of circular coinduction to them.

Extended stream circuits. We plan to investigate precisely which class of streams are realisable using extended stream circuits of Section 7. For this we will also study extended circuits with p-adic merger and splitter where p is a prime number. Moreover the question of well-formedness with respect to the topological properties of the circuits needs to be investigated. As a related problem we are interested in finding a closed formula for *even* and *odd* (and their n-ary counterparts). Intuitively these functions correspond to the roots of unity (cf. [Wil94, § 2.4], and Lemma 9 on periodicity of *take*). This implies that one could use hyperbolic functions (e.g. cosh) to represent the effect of *even* in the stream calculus. We plan to make this connection more formal.

Coalgebraic semantics. Earlier work on stream calculus has led to a coalgebraic treatment of rational power series [Rut08]. Advantage of the coalgebraic modelling is that it present a unified way for dealing with stream circuits, stream functions and transducers. Above all it helps in dealing with various types of bisimulations. We intend to study the material of Section 7 in a coalgebraic setting, by looking into the systems based on causal functions and beyond [Rut06, UV08, Kim08].

Acknowledgements. We thank the anonymous referees for their comments.

References

[Arb04] Arbab, F.: Reo: a channel-based coordination model for component composition. Mathematical Structures in Computer Science 14, 329–366 (2004)

[AS03] Allouche, J.-P., Shallit, J.: Automatic sequences: theory, applications, generalizations. Cambridge University Press, Cambridge (2003)

[BR88] Berstel, J., Reutenauer, C.: Rational series and their languages. EATCS Monographs on Theoretical Computer Science, vol. 12. Springer, Heidelberg (1988)

[BŞ01] Broy, M., Ştefănescu, G.: The algebra of stream processing functions. Theoret. Comput. Sci. 258(1-2), 99–129 (2001)

[Chr79] Christol, G.: Ensembles presque periodiques k-reconnaissables. Theoret. Comput. Sci. 9(1), 141–145 (1979)

[Cob69] Cobham, A.: On the base-dependence of sets of numbers recognizable by finite automata. Math. Systems Theory 3, 186–192 (1969)

[Fog02] Pytheas Fogg, N.: Substitutions in dynamics, arithmetics and combinatorics. In: Berthé, V., Ferenczi, S., Mauduit, C., Siegel, A. (eds.). Lecture Notes in Math., vol. 1794. Springer, Berlin (2002)

[GKP94] Graham, R.L., Knuth, D.E., Patashnik, O.: Concrete mathematics, 2nd edn. Addison-Wesley, Reading (1994)

[Hun80] Hungerford, T.W.: Algebra. Graduate Texts in Mathematics, vol. 73, Springer, New York (1980); Reprint of the 1974 original

[Kah74] Kahn, G.: The semantics of a simple language for parallel programming. In: Information Processing 74: Proceedings of IFIP Congress 74, Stockholm, August 1974, vol. 74, pp. 471–475. North Holland Publishing Co., Amsterdam (1974)

[Kim08] Kim, J.: Coinductive properties of causal maps. In: Meseguer, J., Roşu, G. (eds.) AMAST 2008. LNCS, vol. 5140, pp. 253–267. Springer, Heidelberg (2008)

[LR07] Lucanu, D., Roşu, G.: CIRC: A circular coinductive prover. In: Mossakowski, T., Montanari, U., Haveraaen, M. (eds.) CALCO 2007. LNCS, vol. 4624, pp. 372–378. Springer, Heidelberg (2007)

[Mak08] Mak, R.H.: Design and Performance Analysis of Data-independent Stream Processing Systems. PhD thesis, Technische Universiteit Eindhoven (2008)

[Rut05a] Rutten, J.J.M.M.: A coinductive calculus of streams. Mathematical Structures in Computer Science 15, 93–147 (2005)

[Rut05b] Rutten, J.J.M.M.: A tutorial on coinductive stream calculus and signal flow graphs. Theoretical Computer Science 343(3), 443–481 (2005)

[Rut06] Rutten, J.J.M.M.: Algebraic specification and coalgebraic synthesis of Mealy automata. In: Proceedings of FACS 2005. ENTCS, vol. 160, pp. 305–319. Elsevier Science Publishers, Amsterdam (2006)

[Rut08] Rutten, J.J.M.M.: Rational streams coalgebraically. Logic. Methods in Comput. Sci. 4(3:9), 1–22 (2008)

[UV08] Uustalu, T., Vene, V.: Comonadic notions of computation. In: Adámek, J., Kupke, C. (eds.) Proc. of CMCS 2008. ENTCS, vol. 203(5), pp. 263–284. Elsevier, Amsterdam (June 2008)

[Wil94] Wilf, H.S.: Generatingfunctionology. Academic Press, London (1994)