Math. Struct. in Comp. Science (2019), *vol.* 29, *pp.* 38–66. © Cambridge University Press 2017
doi:10.1017/S0960129517000159 First published online 7 June 2017

Newton series, coinductively: a comparative study of composition

HENNING BASOLD[†], HELLE HVID HANSEN[‡],
JEAN-ÉRIC PIN[§] and JAN RUTTEN[¶]

[†]*Radboud University Nijmegen, P.O. Box 9010, 6500GL Nijmegen, The Netherlands, and CWI Amsterdam, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands.*
Email: h.basold@cs.ru.nl
[‡]*Delft University of Technology, P.O. Box 5015, 2600 GA Delft, The Netherlands.*
Email: h.h.hansen@tudelft.nl
[§]*Université Paris Denis Diderot and CNRS, 75205 Paris Cedex 13, France.*
Email: Jean-Eric.Pin@liafa.univ-paris-diderot.fr
[¶]*CWI Amsterdam, P.O. Box 94079, 1090 GB Amsterdam, The Netherlands, and Radboud University Nijmegen, P.O. Box 9010, 6500GL Nijmegen, The Netherlands.*
Email: jjmmrutten@gmail.com

Received 4 May 2016; revised 3 April 2017

We present a comparative study of four product operators on weighted languages: (i) the *convolution*, (ii) the *shuffle*, (iii) the *infiltration* and (iv) the *Hadamard* product. Exploiting the fact that the set of weighted languages is a final coalgebra, we use coinduction to prove that an operator of the classical difference calculus, the *Newton transform*, generalises from infinite sequences to weighted languages. We show that the Newton transform is an isomorphism of rings that transforms the Hadamard product of two weighted languages into their infiltration product, and we develop various representations for the Newton transform of a language, together with concrete calculation rules for computing them.

1. Introduction

Formal languages are a well-established formalism for the modelling of the behaviour of systems, typically represented by automata, cf. Eilenberg (1974). *Weighted* languages – aka formal power series (Berstel and Reutenauer 1988) – are a common generalisation of both formal languages (sets of words) and streams (infinite sequences). Formally, a weighted language is an assignment from words over an alphabet A to values in a set k of *weights*. Such weights can represent various things such as the multiplicity of the occurrence of a word, or its duration, or probability, etc. In order to be able to add and multiply, and even subtract such weights, k is typically assumed to be a semiring (e.g., the Booleans) or a ring (e.g., the integers).

We present a comparative study of four product operators on weighted languages, which give us four different ways of composing the behaviour of systems. The operators under study are (i) the *convolution*, (ii) the *shuffle*, (iii) the *infiltration* and (iv) the *Hadamard* product, representing, respectively: (i) the concatenation or sequential composition, (ii) the interleaving without synchronisation, (iii) the interleaving *with* synchronisation and (iv) the fully synchronised interleaving of systems. The set of weighted languages, together

https://doi.org/10.1017/S0960129517000159 Published online by Cambridge University Press

with the operation of *sum* and combined with any of these four product operators, is a ring itself, assuming that k is a ring. This means that in all four cases, we have a well-behaved calculus of behaviours.

Main contributions: (1) We show that a classical operator from difference calculus in mathematics: the *Newton transform*, generalises from infinite sequences to weighted languages, and we characterise it in terms of the shuffle product. (2) Next, we show that the Newton transform is an isomorphism of rings that transforms the Hadamard product of two weighted languages into an infiltration product. This allows us to switch back and forth between a fully synchronised composition of behaviours, and a shuffled, partially synchronised one. (3) We develop various representations for the Newton transform of a language, together with concrete calculation rules for computing them.

Approach: We exploit the fact that the set of weighted languages is a *final coalgebra* (Rutten 2000, 2003b). This allows us to use *coinduction* as the guiding methodology for both our definitions and proofs. More specifically, we define our operators in terms of *behavioural differential equations*, which yields, for instance, a uniform and thereby easily comparable presentation of all four product operators. Moreover, we construct *bisimulation* relations in order to prove various identities.

As the set of weighted languages over a one-letter alphabet is isomorphic to the set of streams, it turns out to be convenient to prove our results first for the special case of streams and then to generalise them to weighted languages.

Related work: The present paper fits in the coalgebraic outlook on systems behaviour, as in, for instance, Barbosa (2001) and Rutten (2003b). The definition of Newton series for weighted languages was introduced in Pin and Silva (2014), where Mahler's theorem (which is a p-adic version of the classical Stone–Weierstrass theorem) is generalised to weighted languages. The Newton transform for streams already occurs in Pavlović and Escardó (1998), where it is called the discrete Taylor transform, but not its characterisation using the shuffle product, which for streams goes back to Rutten (2005), and which for weighted languages is new. Related to that, we present elimination rules for certain uses of the shuffle product, which were known for streams (Rutten 2005) and are new for languages. The proof that the Newton transform for weighted languages is a ring *isomorphism* that exchanges the Hadamard product into the infiltration product, is new. In Lothaire (1997, Chapter 6), an operation was defined that does the reverse; it follows from our work that this operation is the inverse of the Newton transform. The infiltration product was introduced in Chen et al. (1958); as we already mentioned, Lothaire (1997, Chapter 6) studies some of its properties, using a notion of binomial coefficients for words that generalises the classical notions for numbers. The present paper introduces a new notion of binomial coefficients for words, which refines the definition of Lothaire (1997, Chapter 6).

This paper is an extended version of the earlier conference paper (Basold et al. 2015).

2. Preliminaries: stream calculus

We present basic facts from coinductive stream calculus (Rutten 2005). In the following, we assume k to be a ring, unless stated otherwise. Let then the set of streams over k be given by $k^\omega = \{\, \sigma \mid \sigma : \mathbb{N} \to k \,\}$. We define the *initial value* of a stream σ by $\sigma(0)$ and

https://doi.org/10.1017/S0960129517000159 Published online by Cambridge University Press

its *stream derivative* by $\sigma' = (\sigma(1), \sigma(2), \sigma(3), \dots)$. In order to conclude that two streams σ and τ are equal, it suffices to prove $\sigma(n) = \tau(n)$, for all $n \geqslant 0$. Sometimes this can be proved by *induction* on the natural number n but, more often than not, we will not have a succinct description or formula for $\sigma(n)$ and $\tau(n)$, and induction will be of no help. Instead, we take here a coalgebraic perspective on k^ω, and most of our proofs will use the proof principle of *coinduction*, which is based on the following notion.

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

$$\sigma(0) = \tau(0) \qquad \text{and} \qquad (\sigma', \tau') \in R. \tag{1}$$

The following theorem is easily proved by induction.

Theorem 2.1 (Coinduction proof principle). If there exists a bisimulation relation containing (σ, τ), then $\sigma = \tau$.

Coinductive *definitions* are phrased in terms of stream derivatives and initial values, and are called *stream differential equations*; see Rutten (2003b), Rutten (2005), Hansen et al. (2014) for examples and details.

Definition 2.2 (Basic operators). The following system of *stream differential equations* defines our first set of constants and operators, which are explained below:

Derivative	***Initial value***	***Name***
$[r]' = [0]$	$[r](0) = r$	$r \in k$
$X' = [1]$	$X(0) = 0$	
$(\sigma + \tau)' = \sigma' + \tau'$	$(\sigma + \tau)(0) = \sigma(0) + \tau(0)$	sum
$(\Sigma_{i\in I}\sigma_i)' = \Sigma_{i\in I}\sigma_i'$	$(\Sigma_{i\in I}\sigma_i)(0) = \sum_{i\in I}\sigma_i(0)$	infinite sum
$(-\sigma)' = -(\sigma')$	$(-\sigma)(0) = -\sigma(0)$	minus
$(\sigma \times \tau)' = (\sigma' \times \tau) + ([\sigma(0)] \times \tau')$	$(\sigma \times \tau)(0) = \sigma(0)\tau(0)$	convolution product
$(\sigma^{-1})' = -[\sigma(0)^{-1}] \times \sigma' \times \sigma^{-1}$	$(\sigma^{-1})(0) = \sigma(0)^{-1}$	convolution inverse

The unique existence of constants and operators satisfying the equations above is ultimately due to the fact that k^ω, together with the operations of initial value and stream derivative, is a final coalgebra.

For $r \in k$, we have the constant stream $[r] = (r, 0, 0, 0, \dots)$ which we often denote again by r. Then we have the constant stream $X = (0, 1, 0, 0, 0, \dots)$. The sum of two streams is simply elementwise addition. The infinite sum $\Sigma_{i\in I}\sigma_i$ is defined only when the family $\{\sigma_i\}_{i\in I}$ is *summable*, that is, if for all $n \in \mathbb{N}$ the set $\{i \in I \mid \sigma_i(n) \neq 0\}$ is finite. If $I = \mathbb{N}$, we denote $\Sigma_{i\in I}\sigma_i$ by $\sum_{i=0}^{\infty}\sigma_i$. Note that $(\tau_i \times X^i)_i$ is summable for any sequence of streams $(\tau_i)_i$. Minus is defined only if k is a ring. We give a closed formula for the convolution product in Proposition 2.14. For now, we just observe that the first terms are given by

$$\sigma \times \tau = (\sigma(0)\tau(0), \sigma(1)\tau(0) + \sigma(1)\tau(0), \sigma(2)\tau(0) + \sigma(1)\tau(1) + \sigma(0)\tau(2), \dots).$$

In spite of its non-symmetrical definition, convolution product on streams is commutative (assuming that the product on k is), see Remark 2.6 below. Convolution inverse is defined

https://doi.org/10.1017/S0960129517000159 Published online by Cambridge University Press

for those streams σ for which the initial value $\sigma(0)$ is invertible. We will often write $r\sigma$ for $[r] \times \sigma$, $1/\sigma$ for σ^{-1} and τ/σ for $\tau \times (1/\sigma)$, which – for streams – is equal to $(1/\sigma) \times \tau$.

The following analogue of the fundamental theorem of calculus, tells us how to compute a stream σ from its initial value $\sigma(0)$ and derivative σ'.

Theorem 2.3. We have $\sigma = [\sigma(0)] + (X \times \sigma')$, for every $\sigma \in k^\omega$.

We will also use coinduction-up-to, cf. Rutten (2005), Rot et al. (2013), a strengthening of the coinduction proof principle based on the following notion. A relation $R \subseteq k^\omega \times k^\omega$ is a *(stream) bisimulation-up-to* if, for all $(\sigma, \tau) \in R$,

$$\sigma(0) = \tau(0) \qquad \text{and} \qquad (\sigma', \tau') \in \bar{R}, \tag{2}$$

where $\bar{R}$ is the smallest reflexive relation on k^ω that contains R and is closed under the element-wise application of the operators in Definition 2.2. For instance, if $(\alpha, \beta), (\gamma, \delta) \in \bar{R}$, then $(\alpha + \gamma, \beta + \delta) \in \bar{R}$, etc.

Theorem 2.4 (Coinduction-up-to). If R is a bisimulation-up-to and $(\sigma, \tau) \in R$, then $\sigma = \tau$.

Proof. One shows with a straightforward induction on the definition of $\bar{R}$ that if R is a bisimulation-up-to, then $\bar{R}$ is a bisimulation. Now apply Theorem 2.1. This fact follows also from more general results. Namely, our notion of bisimulation-up-to is an instance of *bisimulation-up-to-context*, see e.g. Rot (2015, Section. 4.4.2) or Bartels (2004, Corollary 4.4.9), which follows from the fact that the operations in Definition 2.2 are defined in the so-called GSOS format, see also Hansen et al. (2014). □

Using coinduction (up-to), one can prove the following.

Proposition 2.5 (Semiring of streams – with convolution product). If k is a semiring, then the set of streams with sum and convolution product forms a semiring as well: $(k^\omega, +, [0], \times, [1])$. If k is a (commutative) ring, then so is k^ω.

Proof. We prove only a few ring identities in detail. The others are proved basically in the same way, by using both sides of the identity in question to give a relation and prove that this relation is a bismulation (up-to). For the purpose of demonstration, we will prove all necessary identities that lead up to commutativity of the convolution product.

— Associativity of $+$. We define a relation R by

$$R := \{((\sigma + \tau) + \gamma, \sigma + (\tau + \gamma) \mid \sigma, \tau, \gamma \in k^\omega\}$$

and show that it is a bisimulation. So let $((\sigma + \tau) + \gamma, \sigma + (\tau + \gamma) \in R$, we need to show that equation (1) is fulfilled. First, $((\sigma + \tau) + \gamma)(0) = (\sigma + (\tau + \gamma))(0)$ follows directly from associativity $+$ in k. Second, by spelling out the definitions, we have

$$\begin{aligned}((\sigma + \tau) + \gamma)' &= (\sigma + \tau)' + \gamma' \\ &= (\sigma' + \tau') + \gamma' \\ &\; R\; \sigma' + (\tau' + \gamma') \\ &= \sigma' + (\tau + \gamma)' \\ &= (\sigma + (\tau + \gamma))'.\end{aligned}$$

https://doi.org/10.1017/S0960129517000159 Published online by Cambridge University Press

Thus, by Theorem 2.1, all pairs in R are equal and so stream addition is associative.

— Commutativity of $+$ is proved in the same way.

— Convolution distributes from the right over addition: $(\sigma + \tau) \times \gamma = \sigma \times \gamma + \tau \times \gamma$. This is the first time we actually use a bisimulation up-to. We define

$$R := \{((\sigma + \tau) \times \gamma, \sigma \times \gamma + \tau \times \gamma) \mid \sigma, \tau, \gamma \in k^\omega\}$$

and show that R is a bisimulation up-to. For the initial value, this is just distributivity of the ring k. To prove the case for the derivative, we reason as follows:

$$\begin{aligned}
((\sigma + \tau) \times \gamma)' &= (\sigma + \tau)' \times \gamma + [(\sigma + \tau)(0)] \times \gamma' && \text{(def. } \times\text{)} \\
&= (\sigma' + \tau') \times \gamma + [\sigma(0) + \tau(0)] \times \gamma' && \text{(def. } +\text{)} \\
&\;\bar{R}\; (\sigma' \times \gamma + \tau' \times \gamma) + [\sigma(0) + \tau(0)] \times \gamma' \\
&\qquad\qquad \text{(up-to in context } (-) + [\sigma(0) + \tau(0)] \times \gamma'\text{)} \\
&\;\bar{R}\; (\sigma' \times \gamma + \tau' \times \gamma) + ([\sigma(0)] \times \gamma' + [\tau(0)] \times \gamma') \\
&\qquad\qquad \text{(up-to in context } (\sigma' \times \gamma + \tau' \times \gamma) + (-)\text{)} \\
&= (\sigma' \times \gamma + [\sigma(0)] \times \gamma') + (\tau' \times \gamma + [\tau(0)] \times \gamma') && \text{(assoc. and commut. } +\text{)} \\
&= (\sigma \times \gamma)' + (\tau \times \gamma)' && \text{(def. } \times\text{)}
\end{aligned}$$

Thus, R is a bisimulation up-to and distributivity follows from Theorem 2.4.

— The other semiring identities are proved in a similar way.

To show that k^ω is a ring if k is, we show that $R := \{(\sigma + (-\sigma), [0]) \mid \sigma \in k^\omega\}$ is a bisimulation. For the initial value, $(\sigma + (-\sigma))(0) = [0](0)$ follows from the inverse in k. The derivative case is also dealt with easily $((\sigma + (-\sigma))', [0]') = (\sigma' + (-\sigma'), [0]) \in R$. Hence, R is a bisimulation and $-\sigma$ is the inverse for σ.

Before we continue to prove commutativity of the convolution product, note that the definition of $\times$ is asymmetric. This makes it hard to prove commutativity, which is symmetric in the arguments. However, we can use Theorem 2.3 together with distributivity and associativity of $\times$ to obtain

$$\begin{aligned}
(\sigma \times \tau)' &= \sigma' \times \tau + \sigma(0) \times \tau' \\
&= \sigma' \times (\tau(0) + X \times \tau') + \sigma(0) \times \tau' \\
&= \sigma' \times \tau(0) + \sigma' \times X \times \tau' + \sigma(0) \times \tau' \\
&= \sigma' \times \tau(0) + \sigma(0) \times \tau' + \sigma' \times X \times \tau'.
\end{aligned} \tag{3}$$

This is almost symmetric in σ and τ and will make the following proof much easier.

We use the obvious relation to prove commutativity:

$$R := \{(\sigma \times \tau, \tau \times \sigma) \mid \sigma, \tau \in k^\omega\}.$$

Again, the case for the initial value follows from commutativity in k. For the derivatives, we have

$$\begin{aligned}
(\sigma \times \tau)' &= \sigma' \times \tau(0) + \sigma(0) \times \tau' + \sigma' \times X \times \tau' \\
&\;\bar{R}\; \tau(0) \times \sigma' + \tau' \times \sigma(0) + \tau' \times X \times \sigma' \\
&= \tau' \times \sigma(0) + \tau(0) \times \sigma' + \tau' \times X \times \sigma' \\
&= (\tau \times \sigma)'
\end{aligned}$$

https://doi.org/10.1017/S0960129517000159 Published online by Cambridge University Press

and so R is a bisimulation up-to. Hence, commutativity follows from Theorem 2.4. □

Remark 2.6. The reason we define convolution product as in Definition 2.2 (instead of with the almost symmetric definition in (3)) is that the shape of this equation generalises straightforwardly to a definition of the convolution product on weighted languages, in Definition 7.2, and this product is *not* commutative.

We note that if k has an additive inverse, then we can give a truly symmetric definition of the convolution product on streams by using $\sigma(0) = \sigma - X \times \sigma'$:

$$\begin{aligned}(\sigma \times \tau)' &= \sigma' \times \tau + \sigma(0) \times \tau' \\ &= \sigma' \times \tau + (\sigma - X \times \sigma') \times \tau' \\ &= \sigma' \times \tau + \sigma \times \tau' - X \times \sigma' \times \tau'.\end{aligned}$$

◄

Polynomial and rational streams are defined as usual, cf. Rutten (2003b).

Definition 2.7 (Polynomial, rational streams). We call a stream $\sigma \in k^\omega$ *polynomial* if it is of the form $\sigma = a_0 + a_1X + a_2X^2 + \cdots + a_nX^n$, for $n \geqslant 0$ and $a_i \in k$. We call σ *rational* if it is of the form

$$\sigma = \frac{a_0 + a_1X + a_2X^2 + \cdots + a_nX^n}{b_0 + b_1X + b_2X^2 + \cdots + b_mX^m}$$

with $n, m \geqslant 0$, $a_i, b_j \in k$ and b_0 is invertible.

Example 2.8. Here are a few concrete examples of streams (over the natural numbers):

$$\begin{aligned}1 + 2X + 3X^2 &= (1, 2, 3, 0, 0, 0, \ldots), \\ \frac{1}{1-2X} &= (2^0, 2^1, 2^2, \ldots), \\ \frac{1}{(1-X)^2} &= (1, 2, 3, \ldots), \\ \frac{X}{1-X-X^2} &= (0, 1, 1, 2, 3, 5, 8, \ldots).\end{aligned}$$

We note that convolution product behaves naturally, as in the following example:

$$(1 + 2X^2) \times (3 - X) = 3 - X + 6X^2 - 2X^3.$$

◄

Remark 2.9. The relation of the stream derivative to the analytic derivative (of polynomials) is nicely illustrated by the identities

$$\left(X^{n+1}\right)' = X^n \qquad \text{and} \qquad (X \times \sigma)' = \sigma,$$

where $\sigma \in k^\omega$ and k is a semiring. The first identity follows from the second, which in turn follows easily from Definition 2.2 and Proposition 2.5:

$$\begin{aligned}(X \times \sigma)' &= (X' \times \sigma) + ([0] \times \sigma') \\ &= ([1] \times \sigma) + ([0] \times \sigma') \\ &= \sigma\end{aligned}$$

https://doi.org/10.1017/S0960129517000159 Published online by Cambridge University Press

The following rule, which is immediate by Theorem 2.3 and the second identity above, is (surprisingly) helpful when computing derivatives of fractions: for all $\sigma \in k^\omega$,

$$\sigma' = (\sigma - \sigma(0))'.$$

These identities allow us to obtain, for instance,

$$\begin{aligned}\left(\frac{1}{1-X-X^2}\right)' &= \left(\frac{1}{1-X-X^2} - 1\right)' \\ &= \left(\frac{1}{1-X-X^2} - \frac{1-X-X^2}{1-X-X^2}\right)' \\ &= \left(\frac{X+X^2}{1-X-X^2}\right)' \\ &= \left(X \times \frac{1+X}{1-X-X^2}\right)' \\ &= \frac{1+X}{1-X-X^2}\end{aligned}$$

◄

We shall be using yet another operation on streams.

Definition 2.10 (Stream composition). We define the composition of streams by the following stream differential equation:

Derivative	***Initial value***	***Name***
$(\sigma \circ \tau)' = \tau' \times (\sigma' \circ \tau)$	$(\sigma \circ \tau)(0) = \sigma(0)$	Stream composition

The first terms of $\sigma \circ \tau$ are

$$\sigma \circ \tau = (\sigma(0), \tau(1)\sigma(1), \tau(2)\sigma(1) + \tau(1)\tau(1)\sigma(2), \ldots)$$

Composition enjoys the following properties.

Proposition 2.11 (Properties of composition). For all $r \in k$ and all $\rho, \sigma, \tau \in k^\omega$, we have

$$\begin{aligned}[r] \circ \tau &= [r], & (\rho + \sigma) \circ \tau &= (\rho \circ \tau) + (\sigma \circ \tau), \\ \sigma^{-1} \circ \tau &= (\sigma \circ \tau)^{-1} & (\rho \times \sigma) \circ \tau &= (\rho \circ \tau) \times (\sigma \circ \tau),\end{aligned}$$

and similarly for the infinite sum. Moreover, for all τ with $\tau(0) = 0$, we have

$$X \circ \tau = \tau.$$

Proof. All identities follow by coinduction up-to (Theorem 2.4). We give a proof for the identity $(\rho + \sigma) \circ \tau = (\rho \circ \tau) + (\sigma \circ \tau)$ by showing that the following relation is a bisimulation-up-to:

$$R = \{\, (\,(\rho + \sigma) \circ \tau,\ (\rho \circ \tau) + (\sigma \circ \tau)\,) \mid \rho, \sigma, \tau \in k^\omega,\ \tau(0) = 0 \,\}.$$

https://doi.org/10.1017/S0960129517000159 Published online by Cambridge University Press

We easily obtain $((\rho + \sigma) \circ \tau)(0) = ((\rho \circ \tau) + (\sigma \circ \tau))(0)$ from the definitions of $+$ and $\circ$. For the derivative, we have

$$\begin{aligned}
((\rho + \sigma) \circ \tau)' &= \tau' \times ((\rho + \sigma)' \circ \tau) && \text{Def. } \circ \\
&= \tau' \times ((\rho' + \sigma') \circ \tau) && \text{Def. } + \\
\bar{R}\ &\tau' \times (\rho' \circ \tau + \sigma' \circ \tau) && \text{Up-to in context } \tau' \times (-) \\
&= \tau' \times (\rho' \circ \tau) + \tau' \times (\sigma' \circ \tau) && \times \text{ distributes over } + \\
&= ((\rho \circ \tau) + (\sigma \circ \tau))' && \text{Def. of } \circ \text{ and } +.
\end{aligned}$$

Hence, R is a bisimulation up-to. The other identities are proved similarly. □

Proposition 2.11 shows that composing with τ distributes over sum, product and inverse, and on the 'atomic' streams, it acts as identity on $[r]$, but it replaces X with τ when $\tau(0) = 0$. As a consequence, for rational σ, τ with $\tau(0) = 0$, the composition $\sigma \circ \tau$ is obtained by replacing every X in σ with τ.

Example 2.12. From Proposition 2.11, we obtain

$$\frac{X}{1 - X - X^2} \circ \frac{X}{1 + X} = \frac{X(1 + X)}{1 + X - X^2}$$

◄

Defining $\sigma^{(0)} = \sigma$ and $\sigma^{(n+1)} = (\sigma^{(n)})'$, for any stream $\sigma \in k^\omega$, we have $\sigma^{(n)}(0) = \sigma(n)$. Thus, $\sigma = (\sigma(0), \sigma(1), \sigma(2), \ldots) = (\sigma^{(0)}(0), \sigma^{(1)}(0), \sigma^{(2)}(0), \ldots)$. Hence, every stream is equal to the stream of its *Taylor coefficients* (with respect to stream derivation). There is also the corresponding notion of *Taylor series* for streams. The *Taylor series* of a stream σ is defined as the infinite sum

$$\sum_{i=0}^{\infty} [\sigma^{(i)}(0)] \times X^i$$

According to Definition 2.2, the infinite sum is well defined, since for all $n \in \mathbb{N}$, the set $\{i \in \mathbb{N} \mid ([\sigma(i)] \times X^i)(n) \neq 0\} = \{n\}$ is finite. We can now state the corresponding *Taylor series* representation for streams.

Theorem 2.13 (Taylor series). For every $\sigma \in k^\omega$,

$$\sigma = \sum_{i=0}^{\infty} [\sigma^{(i)}(0)] \times X^i = \sum_{i=0}^{\infty} [\sigma(i)] \times X^i.$$

For some of the operations on streams, we have explicit formulae for the nth Taylor coefficient, that is, for their value in n.

Proposition 2.14. For all $\sigma, \tau \in k^\omega$, for all $n \geqslant 0$,

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

https://doi.org/10.1017/S0960129517000159 Published online by Cambridge University Press

3. Four product operators

In addition to convolution product, we shall discuss also the following product operators (repeating below the definitions of convolution product and inverse).

Definition 3.1 (Product operators). We define four product operators by the following system of stream differential equations:

Derivative	*Initial value*	*Name*
$(\sigma \times \tau)' = (\sigma' \times \tau) + (\sigma(0) \times \tau')$	$(\sigma \times \tau)(0) = \sigma(0)\tau(0)$	convolution
$(\sigma \otimes \tau)' = (\sigma' \otimes \tau) + (\sigma \otimes \tau')$	$(\sigma \otimes \tau)(0) = \sigma(0)\tau(0)$	shuffle
$(\sigma \odot \tau)' = \sigma' \odot \tau'$	$(\sigma \odot \tau)(0) = \sigma(0)\tau(0)$	Hadamard
$(\sigma \uparrow \tau)' = (\sigma' \uparrow \tau) + (\sigma \uparrow \tau') + (\sigma' \uparrow \tau')$	$(\sigma \uparrow \tau)(0) = \sigma(0)\tau(0)$	infiltration

For streams σ with invertible initial value $\sigma(0)$, we can define both convolution and shuffle inverse, as follows:

Derivative	*Initial value*	*Name*
$(\sigma^{-1})' = -[\sigma(0)^{-1}] \times \sigma' \times \sigma^{-1}$	$(\sigma^{-1})(0) = \sigma(0)^{-1}$	convolution inverse
$(\sigma^{-\underline{1}})' = -\sigma' \otimes \sigma^{-\underline{1}} \otimes \sigma^{-\underline{1}}$	$(\sigma^{-\underline{1}})(0) = \sigma(0)^{-1}$	shuffle inverse

(We will not need the inverse of the other two products.) Convolution and Hadamard product are standard operators in mathematics. Shuffle and infiltration product are, for streams, less well-known, and are better explained and understood when generalised to weighted languages, which we shall do in Section 7. Closed forms for shuffle and Hadamard are given in Proposition 3.3 below. In the present section and the next, we shall relate convolution product and Hadamard product to, respectively, shuffle product and infiltration product, using the so-called *Laplace* and *Newton* transforms.

Example 3.2. Here are a few simple examples of streams (over the natural numbers), illustrating the differences between these four products.

$$\frac{1}{1-X} \times \frac{1}{1-X} = \frac{1}{(1-X)^2} = (1, 2, 3, \ldots)$$
$$\frac{1}{1-X} \otimes \frac{1}{1-X} = \frac{1}{1-2X} = (2^0, 2^1, 2^2, \ldots)$$
$$\frac{1}{1-X} \odot \frac{1}{1-X} = \frac{1}{1-X}$$
$$\frac{1}{1-X} \uparrow \frac{1}{1-X} = \frac{1}{1-3X} = (3^0, 3^1, 3^2, \ldots)$$
$$(1-X)^{-\underline{1}} = (0!, 1!, 2!, \ldots) \tag{4}$$

◀

We have the following closed formulae for the shuffle and Hadamard product. Recall Proposition 2.14 for the closed form of convolution product. In Proposition 4.6 below, we derive a closed formula for the infiltration product as well.

https://doi.org/10.1017/S0960129517000159 Published online by Cambridge University Press

Proposition 3.3.

$$(\sigma \otimes \tau)(n) = \sum_{i=0}^{n} \binom{n}{i} \sigma(i)\tau(n-i), \tag{5}$$

$$(\sigma \odot \tau)(n) = \sigma(n)\tau(n). \tag{6}$$

Next, we consider the set of streams k^ω together with sum and, respectively, each of the four product operators.

Proposition 3.4 (Four (semi)rings of streams). If k is a (semi)ring, then each of the four product operators defines a corresponding (semi)ring structure on k^ω, as follows:

$$\mathcal{R}_c = (k^\omega, +, [0], \times, [1]), \qquad \mathcal{R}_s = (k^\omega, +, [0], \otimes, [1]),$$
$$\mathcal{R}_H = (k^\omega, +, [0], \odot, \mathsf{ones}), \qquad \mathcal{R}_i = (k^\omega, +, [0], \uparrow, [1]),$$

where ones denotes $(1, 1, 1, \ldots)$.

Proof. A proof is easy by coinduction-up-to, once we have adapted Theorem 2.4 by requiring $\bar{R}$ to be also closed under element-wise application of the product operators above. This is possible, as the multiplication operators again fit the GSOS format (Hansen et al. 2014). The arguments are then very similar to the ones used in the proof of Proposition 2.5. □

For the case $k = \mathbb{R}$, it was shown in Pavlović and Escardó (1998) and Rutten (2005, Theorem 10.1) that there is a ring isomorphism between $\mathcal{R}_c$ and $\mathcal{R}_s$. This result can be slightly generalised to the case where k is any field of characteristic 0.

Theorem 3.5 (Laplace for streams). Let the Laplace transform $\Lambda : k^\omega \to k^\omega$ be given by the following stream differential equation:

Derivative	***Initial value***	***Name***
$(\Lambda(\sigma))' = \Lambda(d/dX(\sigma))$	$\Lambda(\sigma)(0) = \sigma(0)$	Laplace

where $d/dX(\sigma) = (X \otimes \sigma')' = (\sigma(1), 2\sigma(2), 3\sigma(3), \ldots)$. Then, $\Lambda : \mathcal{R}_c \to \mathcal{R}_s$ is a homomorphism of rings; notably, for all $\sigma, \tau \in k^\omega$, $\Lambda(\sigma \times \tau) = \Lambda(\sigma) \otimes \Lambda(\tau)$. If k is a field of characteristic 0, then Λ is bijective and hence an isomorphism of rings.

The Laplace transform is also known as the Laplace–Carson transform, cf. Bergeron et al. (1998, p. 350) and Comtet (1974, p. 48).

Proof. For a sketch of the proof of Theorem 3.5, note that

$$\Lambda(\sigma) = (0!\sigma(0), 1!\sigma(1), 2!\sigma(2), \ldots)$$

as one can readily prove. Note that here the natural number n denotes the element $1 + \cdots + 1$ (n times) in k. It follows that Λ is bijective, with inverse

$$\Lambda^{-1}(\sigma) = \left(\frac{\sigma(0)}{0!}, \frac{\sigma(1)}{1!}, \frac{\sigma(2)}{2!}, \ldots\right),$$

https://doi.org/10.1017/S0960129517000159 Published online by Cambridge University Press

where the assumption that k has characteristic 0 ensures that $n! \neq 0$, for all $n \geqslant 0$. Coalgebraically, Λ arises as the unique final coalgebra homomorphism between two different coalgebra structures on k^ω:

$$\begin{array}{ccc} k^\omega & \xrightarrow{\Lambda} & k^\omega \\ {\scriptstyle \langle (-)(0),\, d/dX\rangle}\downarrow & & \downarrow{\scriptstyle \langle (-)(0),\, (-)'\rangle} \\ k \times k^\omega & \xrightarrow{1\times\Lambda} & k \times k^\omega \end{array}$$

On the right, we have the standard (final) coalgebra structure on streams, given by $\sigma \mapsto (\sigma(0), \sigma')$, whereas on the left, the operator d/dX is used instead of stream derivative $\sigma \mapsto (\sigma(0), d/dX(\sigma))$. The commutativity of the diagram above is precisely expressed by the stream differential equation defining Λ above. It is this definition, in terms of stream derivatives, that enables us to give an easy proof of Theorem 3.5, by coinduction-up-to. Since one can easily show that also Λ^{-1} is a homomorphism, it follows that both coalgebras above are isomorphic (and, as a consequence, both are final). The inverse Λ^{-1} can therefore be defined coinductively by

$$(\Lambda^{-1}(\sigma))(0) = \sigma(0) \qquad \text{and} \qquad d/dX(\Lambda^{-1}(\sigma)) = \Lambda^{-1}(\sigma').$$

Alternatively, Λ^{-1} can be defined inductively by

$$(\Lambda^{-1}(\sigma))(0) = \sigma(0) \qquad \text{and} \qquad (\Lambda^{-1}(\sigma))(n+1) = \frac{1}{n+1}\Lambda^{-1}(\sigma'))(n).$$

□

As we shall see, there exists also a ring isomorphism between $\mathcal{R}_H$ and $\mathcal{R}_i$. It will be given by the *Newton* transform, which we will consider next.

4. Newton transform

Assuming that k is a ring, let the *difference operator* on a stream $\sigma \in k^\omega$ be defined by

$$\Delta\sigma = \sigma' - \sigma = (\,\sigma(1) - \sigma(0), \sigma(2) - \sigma(1), \sigma(3) - \sigma(2), \dots).$$

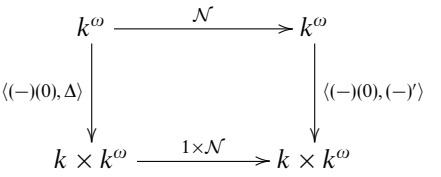
Definition 4.1 (Newton transform). We define the *Newton transform* $\mathcal{N} : k^\omega \to k^\omega$ by the following stream differential equation:

Derivative	***Initial value***	***Name***
$(\mathcal{N}(\sigma))' = \mathcal{N}(\Delta\sigma)$	$\mathcal{N}(\sigma)(0) = \sigma(0)$	Newton transform

It follows that $\mathcal{N}(\sigma) = (\,(\Delta^0\sigma)(0), (\Delta^1\sigma)(0), (\Delta^2\sigma)(0), \dots)$, where $\Delta^0\sigma = \sigma$ and $\Delta^{n+1}\sigma = \Delta(\Delta^n\sigma)$. We call $\mathcal{N}(\sigma)$ the stream of the *Newton coefficients* of σ. Coalgebraically, $\mathcal{N}$ arises as the unique mediating homomorphism – in fact, as we shall see below, an isomorphism

https://doi.org/10.1017/S0960129517000159 Published online by Cambridge University Press

– between the following two coalgebras:



On the right, we have as before the standard (final) coalgebra structure on streams, whereas on the left, the difference operator is used instead: $\sigma \mapsto (\sigma(0), \Delta\sigma)$. We note that the term Newton transform is used in mathematical analysis (Burns and Palmore 1989) for an operational method for the transformation of differentiable functions. In Pavlović and Escardó (1998), where the diagram above is discussed, our present Newton transform $\mathcal{N}$ is called the discrete Taylor transformation.

The fact that $\mathcal{N}$ is bijective follows from Theorem 4.3 below, which characterises $\mathcal{N}$ in terms of the shuffle product. Its proof uses the following lemma.

Lemma 4.2. We have $\dfrac{1}{1-X} \otimes \dfrac{1}{1+X} = 1.$

Proof. Noting that $\frac{1}{1-X} = (1, 1, 1, \ldots)$ and $\frac{1}{1+X} = (1, -1, 1, -1, 1, -1, \ldots)$, the lemma is immediate by Equation (5). Alternatively, a proof by coinduction-up-to is straightforward. And last, the lemma is a special instance of Theorem 5.1. □

Note that this formula combines the convolution inverse with the shuffle product. The function $\mathcal{N}$, and its inverse, can be characterised by the following formulae.

Theorem 4.3 (Rutten 2005). The function $\mathcal{N}$ is bijective and satisfies, for all $\sigma \in k^\omega$,

$$\mathcal{N}(\sigma) = \frac{1}{1+X} \otimes \sigma, \qquad \mathcal{N}^{-1}(\sigma) = \frac{1}{1-X} \otimes \sigma.$$

Proof. We show that

$$R = \left\{ \left(\mathcal{N}(\sigma), \frac{1}{1+X} \otimes \sigma \right) \;\middle|\; \sigma \in k^\omega \right\}$$

is a bisimulation. So let $\sigma \in k^\omega$ be any stream. We have $\mathcal{N}(\sigma)(0) = (\frac{1}{1+X} \otimes \sigma)(0)$, since $\left(\frac{1}{1+X}\right)(0) = 1$. As for the derivatives, we first note that

$$\begin{aligned} \left(\frac{1}{1+X} \otimes \sigma \right)' &= \left(-\frac{1}{1+X} \otimes \sigma \right) + \left(\frac{1}{1+X} \otimes \sigma' \right) \\ &= \frac{1}{1+X} \otimes (\sigma' - \sigma) \\ &= \frac{1}{1+X} \otimes \Delta\sigma. \end{aligned}$$

Hence, $(\mathcal{N}(\sigma))' = \mathcal{N}(\Delta\sigma)\ R\ \left(\frac{1}{1+X} \otimes \Delta\sigma\right) = \left(\frac{1}{1+X} \otimes \sigma\right)'$. It follows that R is a bisimulation, and the first identity holds by coinduction. It follows from Lemma 4.2 that $\mathcal{N}^{-1}(\mathcal{N}(\sigma)) = \sigma$ and $\mathcal{N}(\mathcal{N}^{-1}(\sigma)) = \sigma$. □

https://doi.org/10.1017/S0960129517000159 Published online by Cambridge University Press

At this point, we observe the following structural parallel between the Laplace transform from Theorem 3.5 and the Newton transform: For all $\sigma \in k^\omega$,

$$\Lambda(\sigma) = (1 - X)^{-\underline{1}} \odot \sigma, \tag{7}$$

$$\mathcal{N}(\sigma) = (1 + X)^{-1} \otimes \sigma. \tag{8}$$

Equation (7) is immediate by recalling from Equation (4) that $(1-X)^{-\underline{1}} = (0!, 1!, 2!, \ldots)$. Equation (8) is Theorem 4.3.

The Newton transform is also an isomorphism of *rings*, as follows.

Theorem 4.4 (Newton transform as ring isomorphism). The map $\mathcal{N} : \mathcal{R}_H \to \mathcal{R}_i$ is an isomorphism of rings; notably, $\mathcal{N}(\sigma \odot \tau) = \mathcal{N}(\sigma) \uparrow \mathcal{N}(\tau)$, for all $\sigma, \tau \in k^\omega$.

Proof. We have that $\mathcal{N}([0]) = [0]$ and that $\mathcal{N}(\mathsf{ones}) = [1]$. The identity $\mathcal{N}(-\sigma) = -\mathcal{N}(\sigma)$ follows using the ring identities for $\mathcal{R}_s$. Using the ring properties and the fact that $\mathcal{N}(\sigma)' = \mathcal{N}(\Delta\sigma)$ as we saw in the proof of Theorem 4.3, one easily proves that

$$\{(\mathcal{N}(\sigma + \tau),\ \mathcal{N}(\sigma) + \mathcal{N}(\tau)) \mid \sigma, \tau \in k^\omega\} \ \cup\ \{(\mathcal{N}(\sigma \odot \tau),\ \mathcal{N}(\sigma) \uparrow \mathcal{N}(\tau)) \mid \sigma, \tau \in k^\omega\}$$

is a bisimulation-up-to, and the result follows from Theorem 2.4. To illustrate, we compute derivatives of the pairs involving Hadamard and infiltration product:

$$\begin{aligned}\mathcal{N}(\sigma \odot \tau)' &= \mathcal{N}(\Delta(\sigma \odot \tau)) \\ &= \mathcal{N}(\sigma' \odot \tau') - \mathcal{N}(\sigma \odot \tau)\end{aligned}$$

$$\begin{aligned}(\mathcal{N}(\sigma) \uparrow \mathcal{N}(\tau))' &= \mathcal{N}(\sigma)' \uparrow \mathcal{N}(\tau) + \mathcal{N}(\sigma) \uparrow \mathcal{N}(\tau)' + \mathcal{N}(\sigma)' \uparrow \mathcal{N}(\tau)' \\ &= \mathcal{N}(\Delta\sigma) \uparrow \mathcal{N}(\tau) + \mathcal{N}(\sigma) \uparrow \mathcal{N}(\Delta\tau) + \mathcal{N}(\Delta\sigma) \uparrow \mathcal{N}(\Delta\tau) \\ &= (\mathcal{N}(\sigma') - \mathcal{N}(\sigma)) \uparrow \mathcal{N}(\tau) + \mathcal{N}(\sigma) \uparrow (\mathcal{N}(\tau') - \mathcal{N}(\tau)) \\ &\quad + \ (\mathcal{N}(\sigma') - \mathcal{N}(\sigma)) \uparrow (\mathcal{N}(\tau') - \mathcal{N}(\tau)) \\ &= \mathcal{N}(\sigma') \uparrow \mathcal{N}(\tau) - \mathcal{N}(\sigma) \uparrow \mathcal{N}(\tau) + \mathcal{N}(\sigma) \uparrow \mathcal{N}(\tau') - \mathcal{N}(\sigma) \uparrow \mathcal{N}(\tau) \\ &\quad + \mathcal{N}(\sigma') \uparrow \mathcal{N}(\tau') - \mathcal{N}(\sigma') \uparrow \mathcal{N}(\tau) - \mathcal{N}(\sigma') \uparrow \mathcal{N}(\tau) + \mathcal{N}(\sigma) \uparrow \mathcal{N}(\tau) \\ &= \mathcal{N}(\sigma') \uparrow \mathcal{N}(\tau') - \mathcal{N}(\sigma) \uparrow \mathcal{N}(\tau)\end{aligned}$$

□

Expanding the formulae in Theorem 4.3 and applying Proposition 3.3, we obtain the following closed formulae.

Proposition 4.5. For all $\sigma \in k^\omega$ and $n \geqslant 0$,

$$\mathcal{N}(\sigma)(n) = \sum_{i=0}^{n} \binom{n}{i} (-1)^{n-i} \sigma(i), \qquad \mathcal{N}^{-1}(\sigma)(n) = \sum_{i=0}^{n} \binom{n}{i} \sigma(i).$$

From these, we can derive the announced closed formula for the infiltration product.

https://doi.org/10.1017/S0960129517000159 Published online by Cambridge University Press

Proposition 4.6. For all $\sigma, \tau \in k^\omega$,

$$(\sigma \uparrow \tau)(n) = \sum_{i=0}^{n} \binom{n}{i} (-1)^{n-i} \left(\sum_{j=0}^{i} \binom{i}{j} \sigma(j) \right) \left(\sum_{l=0}^{i} \binom{i}{l} \tau(l) \right).$$

Proof. By Theorems 4.3 and 4.4, we have

$$\sigma \uparrow \tau = \mathcal{N}(\mathcal{N}^{-1}(\sigma \uparrow \tau)) = \mathcal{N}(\mathcal{N}^{-1}(\sigma) \odot \mathcal{N}^{-1}(\tau)).$$

The proposition follows using Equation (6) and the two identities from Proposition 4.5. □

5. Calculating Newton coefficients

The Newton coefficients of a stream can be computed using the following theorem (Rutten 2005, Theorem 10.2(68)). Note that the right-hand side of Equation (9) no longer contains the shuffle product.

Theorem 5.1 (Shuffle product elimination). For all $\sigma \in k^\omega$, $r \in k$,

$$\frac{1}{1-rX} \otimes \sigma = \frac{1}{1-rX} \times \left(\sigma \circ \frac{X}{1-rX} \right). \tag{9}$$

Let us calculate the Newton transform for a few interesting examples.

Example 5.2. For the Fibonacci numbers

$$\frac{X}{1-X-X^2} = (0, 1, 1, 2, 3, 5, 8, \ldots),$$

we have

$$\begin{aligned}
\mathcal{N}\left(\frac{X}{1-X-X^2}\right) &\overset{\text{Theorem 4.3}}{=} \frac{1}{1+X} \otimes \frac{X}{1-X-X^2} \\
&\overset{\text{Theorem 5.1}}{=} \frac{1}{1+X} \times \left(\frac{X}{1-X-X^2} \circ \frac{X}{1+X} \right) \\
&= \frac{1}{1+X} \times \left(\frac{\frac{X}{1+X}}{1 - \frac{X}{1+X} - (\frac{X}{1+X})^2} \right) \\
&= \frac{X}{1+X-X^2}.
\end{aligned}$$

For $r \in k$, the stream of powers of r is transformed as follows:

$$\begin{aligned}
\mathcal{N}(1, r, r^2, \ldots) &= \mathcal{N}\left(\frac{1}{1-rX}\right) \\
&\overset{\text{Theorem 4.3}}{=} \frac{1}{1+X} \otimes \frac{1}{1-rX} \\
&\overset{\text{Theorem 5.1}}{=} \frac{1}{1+X} \times \left(\frac{1}{1-rX} \circ \frac{X}{1+X} \right)
\end{aligned}$$

https://doi.org/10.1017/S0960129517000159 Published online by Cambridge University Press

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

Similarly, we can calculate the transform of the stream that alternates between 0 and 1:

$$\begin{aligned} \mathcal{N}(0,1,0,1,0,1,\dots) &= \mathcal{N}\left(\frac{X}{1-X^2}\right) \\ &\overset{\text{Theorem 4.3}}{=} \frac{1}{1+X} \otimes \frac{X}{1-X^2} \\ &\overset{\text{Theorem 5.1}}{=} \frac{1}{1+X} \times \left(\frac{X}{1-X^2} \circ \frac{X}{1+X}\right) \\ &= \frac{X}{1+2X} \\ &= (0,-2,2^2,-2^3,\dots). \end{aligned}$$

◀

It is immediate by Theorems 4.3 and 5.1 and Example 2.12 that the Newton transform preserves rationality.

Corollary 5.3. A stream $\sigma \in k^\omega$ is rational iff its Newton transform $\mathcal{N}(\sigma)$ is rational.

Example 5.4. There are also non-rational streams to which we can apply the method above. The stream $\phi = (0!, 1!, 2!, \dots)$ of the factorial numbers can be expressed as $\phi = (1-X)^{-1}$. It can also be written as a continued fraction (all in stream calculus, see Rutten (2003a)):

$$\phi = \cfrac{1}{1 - X - \cfrac{1^2X^2}{1 - 3X - \cfrac{2^2X^2}{1 - 5X - \cfrac{3^2X^2}{\ddots}}}}$$

Calculating with infinite patience, we find

$$\begin{aligned} \mathcal{N}(0!,1!,2!,\dots) &= \mathcal{N}(\phi) \\ &\overset{\text{Theorem 4.3}}{=} \frac{1}{1+X} \otimes \phi \\ &\overset{\text{Theorem 5.1}}{=} \frac{1}{1+X} \times \left(\phi \circ \frac{X}{1+X}\right) \\ &= \frac{1}{1+X} \times \cfrac{1}{1 - \frac{X}{1+X} - \cfrac{1^2\left(\frac{X}{1+X}\right)^2}{1 - 3\left(\frac{X}{1+X}\right) - \cfrac{2^2\left(\frac{X}{1+X}\right)^2}{1 - 5\left(\frac{X}{1+X}\right) - \cfrac{3^2\left(\frac{X}{1+X}\right)^2}{\ddots}}}} \end{aligned}$$

https://doi.org/10.1017/S0960129517000159 Published online by Cambridge University Press

$$= \cfrac{1}{1 - \cfrac{1^2X^2}{1 - 2X - \cfrac{2^2X^2}{1 - 4X - \cfrac{3^2X^2}{\ddots}}}}$$
$$= (1, 0, 1, 2, 9, 44, 265, 1854, \ldots).$$

The value of $\mathcal{N}(\phi)(k)$ is the number of *derangements*[†]. As an aside, let us remark that the above computation can also be nicely described in terms of a simple transformation on infinite weighted stream automata (again, in the style of Rutten (2003a)).

There is also the following closed formula for the stream of derangements:

$$\begin{aligned}
\mathcal{N}(\phi) &= \mathcal{N}((1-X)^{-\underline{1}}) \\
&\overset{\text{Theorem 4.3}}{=} \frac{1}{1+X} \otimes (1-X)^{-\underline{1}} \\
&\overset{\text{Theorem 5.1}}{=} \frac{1}{1+X} \times \left((1-X)^{-\underline{1}} \circ \frac{X}{1+X}\right) \\
&= \frac{1}{1+X} \times \left(\frac{1}{1+X}\right)^{-\underline{1}}.
\end{aligned}$$

Note that the latter expression combines convolution inverse, convolution product and shuffle inverse. ◀

6. Newton series

Theorem 4.3 tells us how to compute for a given stream σ the stream of its Newton coefficients $\mathcal{N}(\sigma)$, using the shuffle product. Conversely, the following Newton series representation tells us how to express a stream σ in terms of its Newton coefficients.

Theorem 6.1 (Newton series for streams). For all $\sigma \in k^\omega$, $n \geqslant 0$,

$$\sigma(n) = \sum_{i=0}^{n} (\Delta^i\sigma)(0) \binom{n}{i}.$$

Proof.

$$\begin{aligned}
\sigma(n) &\overset{\text{Theorem 4.3}}{=} \left(\frac{1}{1-X} \otimes \mathcal{N}(\sigma)\right)(n) \\
&\overset{\text{Proposition 2.14, (5)}}{=} \sum_{i=0}^{n} (\Delta^i\sigma)(0) \binom{n}{i}.
\end{aligned}$$

□

[†] See Sloane's *On-Line Encyclopedia of Integer Sequences*, `http://oeis.org/A000166A000166`.

https://doi.org/10.1017/S0960129517000159 Published online by Cambridge University Press

Using $\binom{n}{i} = n!/i!(n-i)!$ and writing $n^{\underline{i}} = n(n-1)(n-2)\cdots(n-i+1)$ (not to be confused with our notation for the shuffle inverse), Newton series can be also denoted as

$$\sigma(n) = \sum_{i=0}^{n} \frac{(\Delta^i\sigma)(0)}{i!} n^{\underline{i}},$$

thus emphasising the structural analogy with Taylor series, cf. Graham et al. (1994, Equation (5.45)).

Combining Theorem 4.3 with Theorem 5.1 leads to yet another, and less familar expansion theorem (see Scheid (1968) for a *finitary* version thereof).

Theorem 6.2 (Euler expansion for streams). For all $\sigma \in k^\omega$,

$$\sigma = \sum_{i=0}^{\infty} (\Delta^i\sigma)(0) \times \frac{X^i}{(1-X)^{i+1}}.$$

Proof. The proof below is a minor variation of that of Rutten (2005, Theorem11.1):

$$\begin{aligned}
\sigma &\overset{\text{Theorem 4.3}}{=} \frac{1}{1-X} \otimes \mathcal{N}(\sigma) \\
&\overset{\text{Theorem 2.13}}{=} \frac{1}{1-X} \otimes \left(\sum_{i=0}^{\infty} (\Delta^i\sigma)(0) \times X^i \right) \\
&= \sum_{i=0}^{\infty} (\Delta^i\sigma)(0) \times \left(\frac{1}{1-X} \otimes X^i \right) \\
&\overset{\text{Theorem 5.1}}{=} \sum_{i=0}^{\infty} (\Delta^i\sigma)(0) \times \frac{X^i}{(1-X)^{i+1}},
\end{aligned}$$

where in the last but one equality, we use the fact that $r \times \tau = r \otimes \tau$, for all $r \in k$ and $\tau \in k^\omega$, together with the ring properties of $\mathcal{R}_s$. □

Example 6.3. Theorem 6.2 leads, for instance, to an easy derivation of a rational expression for the stream of cubes, namely

$$(1^3, 2^3, 3^3, \ldots) = \frac{1 + 4X + X^2}{(1-X)^4}.$$

To this end, let $\mathsf{ones} = (1, 1, 1, \ldots)$ and $\mathsf{nat} = (1, 2, 3, \ldots)$. We shall write $\sigma^{\langle 0 \rangle} = \mathsf{ones}$ and $\sigma^{\langle n+1 \rangle} = \sigma^{\langle n \rangle} \odot \sigma$. First, we note that $\mathsf{nat}' = \mathsf{nat} + \mathsf{ones}$. Using this together with the ring properties of $\mathcal{R}_H$, we can compute the respective values of $\Delta^n(\mathsf{nat}^{\langle 3 \rangle})$:

$$\begin{aligned}
\Delta^0(\mathsf{nat}^{\langle 3 \rangle}) &= \mathsf{nat}^{\langle 3 \rangle} \\
\Delta^1(\mathsf{nat}^{\langle 3 \rangle}) &= 3\mathsf{nat}^{\langle 2 \rangle} + 3\mathsf{nat} + \mathsf{ones} \\
\Delta^2(\mathsf{nat}^{\langle 3 \rangle}) &= 6\mathsf{nat} + 6\mathsf{ones} \\
\Delta^3(\mathsf{nat}^{\langle 3 \rangle}) &= 6\mathsf{ones} \\
\Delta^{4+i}(\mathsf{nat}^{\langle 3 \rangle}) &= 0.
\end{aligned}$$

https://doi.org/10.1017/S0960129517000159 Published online by Cambridge University Press

By Theorem 6.2, we obtain the following rational expression:

$$\begin{aligned}\mathsf{nat}^{\langle 3\rangle} &= \frac{1}{1-X} + \frac{7X}{(1-X)^2} + \frac{12X^2}{(1-X)^3} + \frac{6X^3}{(1-X)^4} \\ &= \frac{1+4X+X^2}{(1-X)^4}.\end{aligned}$$

More generally, one can prove by induction that, for all $n \geqslant 1$ and for all $i > n$,

$$\Delta^i\left(\mathsf{nat}^{\langle n\rangle}\right) = 0$$

and that, cf. Niqui and Rutten (2011),

$$\mathsf{nat}^{\langle n\rangle} = \frac{\sum_{m=0}^{n-1} A(n,m) \times X^m}{1 - X^{n+1}}.$$

Here, $A(n,m)$ are the so-called *Eulerian numbers*, which are defined, for every $n \geqslant 0$ and $0 \leqslant m \leqslant n-1$, by the following recurrence relation:

$$A(n,m) = (n-m)A(n-1,m-1) + (m+1)A(n-1,m). \qquad \blacktriangleleft$$

7. Weighted languages

Let k again be a ring or semiring and let A be a set. We consider the elements of A as *letters* and call A the *alphabet*. Let A^* denote the set of all finite sequences or *words* over A. We define the set of *languages over A with weights in k* by

$$k^{A^*} = \{\, \sigma \mid \sigma : A^* \to k \,\}.$$

Weighted languages are also known as *formal power series* (over A with coefficients in k), cf. Berstel and Reutenauer (1988). If k is the Boolean semiring $\{0,1\}$, then weighted languages are just sets of words. If k is arbitrary again, but we restrict our alphabet to a singleton set $A = \{X\}$, then k^{A^*} is isomorphic to k^ω, the set of streams with values in k. In other words, by moving from a one-letter alphabet to an arbitrary one, streams generalise to weighted languages.

From a coalgebraic perspective, much about streams holds for weighted languages as well, and typically with an almost identical formulation. This structural similarity between streams and weighted languages is due to the fact that weighted languages carry a final coalgebra structure that is very similar to that of streams, as follows. We define the *initial value* of a (weighted) language σ by $\sigma(\varepsilon)$, that is, σ applied to the *empty word* ε. Next, we define for every $a \in A$ the *a-derivative* of σ by $\sigma_a(w) = \sigma(aw)$, for every $w \in A^*$. Initial value and derivatives together define a final coalgebra structure on weighted languages, given by

$$k^{A^*} \to k \times (k^{A^*})^A \qquad \sigma \mapsto (\sigma(\varepsilon), (\sigma_a)_{a\in A})),$$

where $(k^{A^*})^A = \{f \mid f : A \to k^{A^*}\}$. For the case that $A = \{X\}$, the coalgebra structure on the set of streams is a special case of the one above, since under the isomorphism between k^{A^*} and k^ω, we have that $\sigma(\varepsilon)$ corresponds to $\sigma(0)$, and σ_X corresponds to σ'.

https://doi.org/10.1017/S0960129517000159 Published online by Cambridge University Press

We can now summarise the remainder of this paper, roughly and succinctly, as follows: if we replace in the previous sections $\sigma(0)$ by $\sigma(\varepsilon)$, and σ' by σ_a (for $a \in A$), everywhere, then most of the previous definitions and properties for streams generalise to weighted languages. Notably, we will again have a set of basic operators for weighted languages, four different product operators, four corresponding ring structures and the Newton transform between the rings of Hadamard and infiltration product. (An exception to this optimistic program of translation sketched above, however, is the Laplace transform: there does not seem to exist an obvious generalisation of the Laplace transform for streams – transforming the convolution product into the shuffle product – to the corresponding rings of weighted languages.)

Let us now be more precise and discuss all of this in some detail. For a start, there is again the proof principle of coinduction, now for weighted languages.
A relation $R \subseteq k^{A^*} \times k^{A^*}$ is a *(language) bisimulation* if for all $(\sigma, \tau) \in R$:

$$\sigma(\varepsilon) = \tau(\varepsilon) \quad \text{and} \quad (\sigma_a, \tau_a) \in R, \text{ for all } a \in A. \tag{10}$$

We have the following *coinduction proof principle*, similar to Theorem 2.1:

Theorem 7.1 (Coinduction for languages). If there exists a (language) bisimulation relation containing (σ, τ), then $\sigma = \tau$.

Coinductive *definitions* are given again by differential equations, now called *behavioural differential equations* (Rutten 2003b, 2005).

Definition 7.2 (Basic operators for languages). The following system of *behavioural differential equations* defines the basic constants and operators for languages:

Derivative	***Initial value***	***Name***
$[r]_a = [0]$	$[r](\varepsilon) = r$	$r \in k$
$b_a = [0]$	$b(\varepsilon) = 0$	$b \in A$, $b \neq a$
$b_a = [1]$	$b(\varepsilon) = 0$	$b \in A$, $b = a$
$(\sigma + \tau)_a = (\sigma_a + \tau_a)$	$(\sigma + \tau)(\varepsilon) = \sigma(\varepsilon) + \tau(\varepsilon)$	sum
$(\Sigma_{i\in I}\sigma_i)_a = \Sigma_{i\in I}(\sigma_i)_a$	$(\Sigma_{i\in I}\sigma_i)(\varepsilon) = \sum_{i\in I}\sigma_i(\varepsilon)$	infinite sum
$(-\sigma)_a = -(\sigma_a)$	$(-\sigma)(\varepsilon) = -\sigma(\varepsilon)$	minus
$(\sigma \times \tau)_a = (\sigma_a \times \tau) + ([\sigma(\varepsilon)] \times \tau_a)$	$(\sigma \times \tau)(\varepsilon) = \sigma(\varepsilon)\tau(\varepsilon)$	convolution product
$(\sigma^{-1})_a = -[\sigma(\varepsilon)^{-1}] \times \sigma_a \times \sigma^{-1}$	$(\sigma^{-1})(\varepsilon) = \sigma(\varepsilon)^{-1}$	convolution inverse

The convolution inverse is again defined only for σ with $\sigma(\varepsilon)$ invertible in k. We will write a both for an element of A and for the corresponding constant weighted language. We shall often use shorthands like $ab = a \times b$, where the context will determine whether a word or a language is intended. Also, we will sometimes write A for $\Sigma_{a\in A}a$. The infinite sum $\Sigma_{i\in I}\sigma_i$ is, again, only defined if the family $\{\sigma_i\}_{i\in I}$ is *summable*, i.e., if for all $w \in A^*$ the set $\{i \in I \mid \sigma_i(w) \neq 0\}$ is finite. As before, we shall often write $1/\sigma$ for σ^{-1}. Note that convolution product is weighted concatenation and is no longer commutative. As a consequence, τ/σ is now generally ambiguous as it could mean either $\tau \times \sigma^{-1}$ or $\sigma^{-1} \times \tau$.

https://doi.org/10.1017/S0960129517000159 Published online by Cambridge University Press

Only when the latter are equal, we shall sometimes write τ/σ. An example is $A/(1-A)$, which is A^+, the set of all non-empty words.

Theorem 7.3 (Fundamental theorem, for languages). For every $\sigma \in k^{A^*}$,

$$\sigma = \sigma(\varepsilon) + \sum_{a \in A} a \times \sigma_a$$

(cf. Conway 1971; Rutten 2003b).

We can now extend Theorem 2.4 to languages. Given a relation R on k^{A^*}, we denote by $\bar{R}$ the smallest reflexive relation on k^{A^*} that contains R and is closed under the element-wise application of the operators in Definition 7.2. For instance, if $(\alpha, \beta), (\gamma, \delta) \in \bar{R}$, then $(\alpha+\gamma, \beta+\delta) \in \bar{R}$, etc. A relation $R \subseteq k^{A^*} \times k^{A^*}$ is a *(weighted language) bisimulation-up-to* if for all $(\sigma, \tau) \in R$:

$$\sigma(\varepsilon) = \tau(\varepsilon) \text{ and } \text{ for all } a \in A, (\sigma_a, \tau_a) \in \bar{R}. \tag{11}$$

Theorem 7.4 (Coinduction-up-to for languages). If $(\sigma, \tau) \in R$ for some bisimulation-up-to, then $\sigma = \tau$.

Composition of languages is defined by the following differential equation:

Derivative	***Initial value***	***Name***
$(\sigma \circ \tau)_a = \tau_a \times (\sigma_a \circ \tau)$	$(\sigma \circ \tau)(\varepsilon) = \sigma(\varepsilon)$	composition

Language composition $\sigma \circ \tau$ has similar distribution properties as stream composition.

Proposition 7.5 (Composition of languages). For all $r \in k$ and all $\rho, \sigma, \tau \in k^{A^*}$, we have

$$\begin{aligned} [r] \circ \tau &= [r], & (\rho + \sigma) \circ \tau &= (\rho \circ \tau) + (\sigma \circ \tau), \\ a \circ \tau &= a \times \tau_a, & (\rho \times \sigma) \circ \tau &= (\rho \circ \tau) \times (\sigma \circ \tau), \\ & & \sigma^{-1} \circ \tau &= (\sigma \circ \tau)^{-1}. \end{aligned}$$

For $\tau \in k^{A^*}$ with $\tau(\varepsilon) = 0$, we have $A \circ \tau = \tau$.

Definition 7.6 (Polynomial, rational languages). We call $\sigma \in k^{A^*}$ *polynomial* if it can be constructed using constants ($r \in k$ and $a \in A$) and the operations of finite sum and convolution product. We call $\sigma \in k^{A^*}$ *rational* if it can be constructed using constants and the operations of finite sum, convolution product and convolution inverse.

As a consequence of Proposition 7.5, for every rational σ, $\sigma \circ \tau$ is obtained by replacing every occurrence of a in σ by $a \times \tau_a$, for every $a \in A$.

Defining $\sigma_\varepsilon = \sigma$ and $\sigma_{wa} = (\sigma_w)_a$, for any language $\sigma \in k^{A^*}$, we have $\sigma_w(\varepsilon) = \sigma(w)$. This leads to a *Taylor series* representation for languages.

Theorem 7.7 (Taylor series, for languages). For every $\sigma \in k^{A^*}$,

$$\sigma = \sum_{w \in A^*} \sigma_w(\varepsilon) \times w = \sum_{w \in A^*} \sigma(w) \times w.$$

https://doi.org/10.1017/S0960129517000159 Published online by Cambridge University Press

Example 7.8. Here are a few concrete examples of weighted languages:

$$\frac{1}{1-A} = \sum_{w \in A^*} w = A^*$$

$$\frac{1}{1+A} = \sum_{w \in A^*} (-1)^{|w|} \times w, \qquad \frac{1}{1-2ab} = \sum_{i \geqslant 0} 2^i \times (ab)^i.$$

◄

8. Four rings of weighted languages

The definitions of the four product operators for streams generalise straightforwardly to languages, giving rise to four different ring structures on languages.

Definition 8.1 (Product operators for languages). We define four product operators by the following system of behavioural differential equations:

Derivative	*Initial value*	*Name*
$(\sigma \times \tau)_a = (\sigma_a \times \tau) + ([\sigma(\varepsilon)] \times \tau_a)$	$(\sigma \times \tau)(\varepsilon) = \sigma(\varepsilon)\tau(\varepsilon)$	convolution
$(\sigma \otimes \tau)_a = (\sigma_a \otimes \tau) + (\sigma \otimes \tau_a)$	$(\sigma \otimes \tau)(\varepsilon) = \sigma(\varepsilon)\tau(\varepsilon)$	shuffle
$(\sigma \odot \tau)_a = \sigma_a \odot \tau_a$	$(\sigma \odot \tau)(\varepsilon) = \sigma(\varepsilon)\tau(\varepsilon)$	Hadamard
$(\sigma \uparrow \tau)_a = (\sigma_a \uparrow \tau) + (\sigma \uparrow \tau_a) + (\sigma_a \uparrow \tau_a)$	$(\sigma \uparrow \tau)(\varepsilon) = \sigma(\varepsilon)\tau(\varepsilon)$	infiltration

For languages σ with invertible initial value $\sigma(\varepsilon)$, we can define both convolution and shuffle inverse, as follows:

Derivative	*Initial value*	*Name*
$(\sigma^{-1})_a = -[\sigma(0)^{-1}] \times \sigma_a \times \sigma^{-1}$	$(\sigma^{-1})(0) = \sigma(0)^{-1}$	convolution inverse
$(\sigma^{-\underline{1}})_a = -\sigma_a \otimes \sigma^{-\underline{1}} \otimes \sigma^{-\underline{1}}$	$(\sigma^{-\underline{1}})(0) = \sigma(0)^{-1}$	shuffle inverse

Convolution product is concatenation of (weighted) languages and Hadamard product is the fully synchronised product, which corresponds to the intersection of weighted languages. The shuffle product generalises the definition of the shuffle operator on classical languages (over the Boolean semiring), and can be, equivalently, defined by induction. The following definition is from Lothaire (1997, p.126) (where shuffle product is denoted by the symbol $\circ$): for all $v, w \in A^*$, $\sigma, \tau \in k^{A^*}$,

$$\begin{aligned} v \otimes \varepsilon &= \varepsilon \otimes v = v, \\ va \otimes wb &= (v \otimes wb)a + (va \otimes w)b, \qquad (12) \\ \sigma \otimes \tau &= \sum_{v,w \in A^*} \sigma(v) \times \tau(w) \times (v \otimes w). \qquad (13) \end{aligned}$$

The infiltration product, originally introduced in Chen et al. (1958), can be considered as a variation on the shuffle product that not only interleaves words but also synchronises them on identical letters. In the differential equation for the infiltration product above,

https://doi.org/10.1017/S0960129517000159 Published online by Cambridge University Press

this is apparent from the presence of the additional term $\sigma_a \uparrow \tau_a$. There is also an inductive definition of the infiltration product, in Lothaire (1997, p.128). It is a variant of Equation (12) above that for the case that $a = b$ looks like

$$va \uparrow wa \;=\; (v \uparrow wa)a + (va \uparrow w)a + (v \uparrow w)a.$$

However, we shall be using the coinductive definitions, as these allow us to give proofs by coinduction.

Example 8.2. Here are a few simple examples of weighted languages, illustrating the differences between these four products. Recall that $1/1-A = A^*$, that is, $(1/1-A)(w) = 1$, for all $w \in A^*$. Indicating the length of a word $w \in A^*$ by $|w|$, we have the following identities:

$$\left(\frac{1}{1-A} \times \frac{1}{1-A}\right)(w) = |w| + 1, \qquad \left(\frac{1}{1-A} \otimes \frac{1}{1-A}\right)(w) = 2^{|w|}$$

$$\frac{1}{1-A} \odot \frac{1}{1-A} = \frac{1}{1-A}, \qquad \left(\frac{1}{1-A} \uparrow \frac{1}{1-A}\right)(w) = 3^{|w|}$$

$$\left((1-A)^{-\underline{1}}\right)(w) \;=\; |w|! \tag{14}$$

If we restrict the above identities to streams, that is, if the alphabet $A = \{X\}$, then we obtain the identities on streams from Example 3.2. ◀

Next, we consider the set of weighted languages together with sum and each of the four product operators.

Proposition 8.3 (Four rings of weighted languages). If k is a ring, then each of the four product operators defines a corresponding ring structure on k^{A^*}, as follows:

$$\mathcal{L}_c = \left(k^{A^*}, +, [0], \times, [1]\right), \qquad \mathcal{L}_s = \left(k^{A^*}, +, [0], \otimes, [1]\right)$$

$$\mathcal{L}_H = \left(k^{A^*}, +, [0], \odot, \frac{1}{1-A}\right), \qquad \mathcal{L}_i = \left(k^{A^*}, +, [0], \uparrow, [1]\right).$$

Proof. A proof is again straightforward by coinduction-up-to, once we have adapted Theorem 7.4 by requiring $\bar{R}$ to be also closed under the element-wise application of all four product operators above. □

We conclude the present section with closed formulae for the Taylor coefficients of the above product operators, thus generalising Propositions 2.14 and 3.3 to languages. We first introduce the following notion.

Definition 8.4 (Binomial coefficients on words). For all $u, v, w \in A^*$, we define $\binom{w}{u|v}$ as the number of different ways in which u can be taken out of w as a subword, *leaving* v; or equivalently – and more formally – as the number of ways in which w can be obtained by shuffling u and v; that is,

$$\binom{w}{u \mid v} = (u \otimes v)(w). \tag{15}$$

https://doi.org/10.1017/S0960129517000159 Published online by Cambridge University Press

The above definition generalises the notion of binomial coefficient for words from (Lothaire 1997, p.121), where one defines $\binom{w}{u}$ as the number of ways in which u can be taken as a subword of w. The two notions of binomial coefficient are related by the following formula:

$$\binom{w}{u} = \sum_{v \in A^*} \binom{w}{u \mid v}. \tag{16}$$

As an immediate consequence of the defining Equation (15), we find the following recurrence.

Proposition 8.5. For all $a \in A$ and $u, v, w \in A^*$,

$$\binom{aw}{u \mid v} = \binom{w}{u_a \mid v} + \binom{w}{u \mid v_a}. \tag{17}$$

Proof. We have

$$\begin{aligned} \binom{aw}{u \mid v} &= (u \otimes v)(aw) \\ &= (u \otimes v)_a(w) \\ &\overset{\text{Definition 8.1}}{=} (u_a \otimes v)(w) + (u \otimes v_a)(w) \\ &\overset{(15)}{=} \binom{w}{u_a \mid v} + \binom{w}{u \mid v_a}, \end{aligned}$$

proving the required identity. □

Note that for the case of streams, Equation (17) gives us Pascal's formula for classical binomial coefficients (by taking $a = X$, $w = X^n$, $u = X^k$ and $v = X^{n+1-k}$):

$$\binom{n+1}{k} = \binom{n}{k-1} + \binom{n}{k}.$$

Proposition 8.6 gives another property, the easy proof of which illustrates the convenience of the new definition of binomial coeficient. (It is also given in Lothaire (1997, Proposition 6.3.13), where $1/1 - A$ is written as A^* and convolution product as $\circ$.)

Proposition 8.6. For all $u, w \in A^*$,

$$\left(u \otimes \frac{1}{1-A}\right)(w) = \binom{w}{u}.$$

Proof.

$$\left(u \otimes \frac{1}{1-A}\right)(w) = \left(u \otimes \sum_{v \in A^*} v\right)(w) = \sum_{v \in A^*} (u \otimes v)(w)$$

$$= \sum_{v \in A^*} \binom{w}{u \mid v} = \binom{w}{u}.$$

□

https://doi.org/10.1017/S0960129517000159 Published online by Cambridge University Press

Example 8.7. We have

$$\binom{abab}{ab} = \binom{abab}{ab \mid ab} + \binom{abab}{ab \mid ba} = 2 + 1 = 3.$$

◄

We have the following closed formulae for three of our product operators.

Proposition 8.8. For all $\sigma, \tau \in k^{A^*}$, $w \in A^*$,

$$(\sigma \times \tau)(w) = \sum_{u,v \in A^* \text{ s.t. } u \cdot v = w} \sigma(u)\tau(v),$$

$$(\sigma \otimes \tau)(w) = \sum_{u,v \in A^*} \binom{w}{u \mid v} \sigma(u)\tau(v), \tag{18}$$

$$(\sigma \odot \tau)(w) = \sigma(w)\tau(w). \tag{19}$$

A closed formula for the infiltration product can be derived later, once we have introduced the Newton transform for weighted languages.

9. Newton transform for languages

Assuming again that k is a ring, we define the *difference operator* (with respect to $a \in A$) by

$$(\Delta^a \sigma)(w) = \sigma_a(w) - \sigma(w) = \sigma(aw) - \sigma(w), \tag{20}$$

for all $\sigma \in k^{A^*}$ and $w \in A^*$.

Definition 9.1 (Newton transform for languages). We define the *Newton transform* $\mathcal{N} : k^{A^*} \to k^{A^*}$ by the following behavioural differential equation:

Derivative	***Initial value***	***Name***
$(\mathcal{N}(\sigma))_a = \mathcal{N}(\Delta^a \sigma)$	$\mathcal{N}(\sigma)(\varepsilon) = \sigma(\varepsilon)$	Newton transform

(using again the symbol $\mathcal{N}$, now for weighted languages instead of streams). ◄

It follows that $\mathcal{N}(\sigma)(w) = (\Delta^w \sigma)(\varepsilon)$, for all $w \in A^*$, where we define

$$\Delta^\varepsilon \sigma = \sigma \quad \text{and} \quad \Delta^{wa} \sigma = \Delta^a(\Delta^w \sigma).$$

We point out that our definition in Equation (20) of Δ^a is based on left derivative whereas the definition in Pin and Silva (2014) is based on right derivative, and uses the inductive definition $\Delta^{aw} \sigma = \Delta^a(\Delta^w \sigma)$.

https://doi.org/10.1017/S0960129517000159 Published online by Cambridge University Press

Coalgebraically, $\mathcal{N}$ arises again as a unique mediating isomorphism between two final coalgebras:

$$\begin{array}{ccc} k^{A^*} & \xrightarrow{\quad\mathcal{N}\quad} & k^{A^*} \\ \Big\downarrow{\scriptstyle \sigma\mapsto(\sigma(\varepsilon),(\Delta^a\sigma)_{a\in A})} & & \Big\downarrow{\scriptstyle \sigma\mapsto(\sigma(\varepsilon),(\sigma_a)_{a\in A})} \\ k\times(k^{A^*})^A & \xrightarrow{\quad 1\times\mathcal{N}\quad} & k\times(k^{A^*})^A \end{array}$$

On the right, we have the standard (final) coalgebra structure on weighted languages, given by $\sigma \mapsto (\sigma(\varepsilon), (\sigma_a)_{a\in A})$, whereas on the left, the difference operator is used instead of the stream derivative $\sigma \mapsto (\sigma(\varepsilon), (\Delta^a\sigma)_{a\in A})$.

Theorem 9.2. The function $\mathcal{N}$ is bijective and satisfies, for all $\sigma \in k^{A^*}$,

$$\mathcal{N}(\sigma) = \frac{1}{1+A} \otimes \sigma, \qquad \mathcal{N}^{-1}(\sigma) = \frac{1}{1-A} \otimes \sigma.$$

Note again that these formulae combine the convolution inverse with the shuffle product.

Proof. By coinduction-up-to for languages (Theorem 7.4) and the fact that

$$\frac{1}{1-A} \otimes \frac{1}{1+A} = 1.$$

This identity is easily proved by coinduction, but is also an instance of Theorem 9.4. □

The Newton transform is again an isomorphism of *rings*.

Theorem 9.3 (Newton transform as ring isomorphism for languages). The Newton transform $\mathcal{N} : \mathcal{L}_H \to \mathcal{L}_i$ is an isomorphism of rings; notably, we have for all $\sigma, \tau \in k^\omega$, $\mathcal{N}(\sigma \odot \tau) = \mathcal{N}(\sigma) \uparrow \mathcal{N}(\tau)$.

Noting that $\mathcal{N}(\frac{1}{1-A}) = [1]$, a proof of the theorem by coinduction-up to is straightforward. Part of this theorem is already known in the literature: Lothaire (1997, Theorem 6.3.18) expresses (for the case that $k = \mathbb{Z}$) that $\frac{1}{1-A} \otimes (-)$ transforms the infiltration product of two words into a Hadamard product.

Propositions 4.5 and 4.6 for streams straightforwardly generalise to weighted languages. Also, Theorem 5.1 generalises to weighted languages, as follows.

Theorem 9.4 (Shuffle product elimination for languages). For all $\sigma \in k^{A^*}$, $r \in k$,

$$\frac{1}{1-(r\times A)} \otimes \sigma = \frac{1}{1-(r\times A)} \times \left(\sigma \circ \frac{A}{1-(r\times A)}\right). \tag{21}$$

Proof. One readily shows that the relation

$$\left\{ \left\langle \frac{1}{1-(r\times A)} \otimes \sigma\,,\ \frac{1}{1-(r\times A)} \times \left(\sigma \circ \frac{A}{1-(r\times A)}\right) \right\rangle \;\middle|\; r \in k,\ \sigma \in k^{A^*} \right\}$$

is a bisimulation-up-to. The theorem then follows by Theorem 7.4. □

Corollary 9.5. For all $\sigma \in k^{A^*}$, σ is rational iff $\mathcal{N}(\sigma)$ is rational. For all $\sigma, \tau \in k^{A^*}$, if both $\mathcal{N}(\sigma)$ and $\mathcal{N}(\tau)$ are polynomial resp. rational, then so is $\mathcal{N}(\sigma \odot \tau)$.

https://doi.org/10.1017/S0960129517000159 Published online by Cambridge University Press

Example 9.6. We illustrate the use of Theorem 9.4 in the calculation of the Newton transform with an example stemming from Pin and Silva (2014, Example 2.1).

This example is concerned with defining a map β that assigns to a bitstring its value as binary number. To this end, let the alphabet be $A = \{\hat{0}, \hat{1}\}$, where we use the little festive hats to distinguish the alphabet symbols $\hat{0}, \hat{1} \in A$ from the values $0, 1 \in k$. Since our definition of Δ^w is based on left derivative, our bitstrings have the least significant bit left whereas in Pin and Silva (2014, Example 2.1) bitstrings are read with most significant bit left.

We define $\beta \in k^{A^*}$ by the following behavioural differential equation:

$$\beta(\varepsilon) = 0, \qquad \beta_{\hat{0}} = 2 \times \beta, \qquad \beta_{\hat{1}} = (2 \times \beta) + \frac{1}{1 - A},$$

where $2 = 1 + 1$. Using Theorem 7.3, we can substitute the right-hand side expressions for the derivatives above and solve for β to obtain the following expression (using that $A = \hat{0} + \hat{1}$):

$$\beta = \frac{1}{1 - 2A} \times \hat{1} \times \frac{1}{1 - A}.$$

We have, for instance, that

$$\beta(\hat{0}\hat{1}\hat{1}) = \beta_{\hat{0}\hat{1}\hat{1}}(\varepsilon) = \left((8 \times \beta) + \frac{6}{1 - A}\right)(\varepsilon) = 6.$$

Applying the Newton transform, we find that $\mathcal{N}(\beta) = \frac{1}{1-A} \times \hat{1}$:

$$\begin{aligned}
\mathcal{N}(\beta) &\overset{\text{Theorem 9.2}}{=} \frac{1}{1 + A} \otimes \beta \\
&\overset{\text{Theorem 9.4}}{=} \frac{1}{1 + A} \times \left(\beta \circ \frac{A}{1 + A}\right) \\
&= \frac{1}{1 + A} \times \left(\left(\frac{1}{1 - 2A} \times \hat{1} \times \frac{1}{1 - A}\right) \circ \frac{A}{1 + A}\right) \\
&\overset{\text{Proposition 7.5}}{=} \frac{1}{1 + A} \times \left(\frac{1}{1 - 2A} \circ \frac{A}{1 + A}\right) \times \left(\hat{1} \circ \frac{A}{1 + A}\right) \times \left(\frac{1}{1 - A} \circ \frac{A}{1 + A}\right) \\
&\overset{\text{Proposition 7.5}}{=} \frac{1}{1 + A} \times \left(\frac{1}{1 - 2\frac{A}{1+A}}\right) \times \left(\hat{1} \times \frac{1}{1 + A}\right) \times \left(\frac{1}{1 - \frac{A}{1+A}}\right) \\
&= \frac{1}{1 - A} \times \hat{1}.
\end{aligned}$$

Thus, $\mathcal{N}(\beta)(w) = 1$, for all w ending in $\hat{1}$. ◄

10. Newton series for languages

Theorem 6.1 generalises to weighted languages as follows.

Theorem 10.1 (Newton series for languages). For all $\sigma \in k^{A^*}$, $w \in A^*$,

$$\sigma(w) = \sum_u \binom{w}{u} (\Delta^u \sigma)(\varepsilon).$$

https://doi.org/10.1017/S0960129517000159 Published online by Cambridge University Press

Proof.

$$\begin{aligned}
\sigma(w) &= (1 \otimes \sigma)(w) \\
&= \left(\frac{1}{1-A} \otimes \frac{1}{1+A} \otimes \sigma \right)(w) \\
&= \left(\frac{1}{1-A} \otimes \mathcal{N}(\sigma) \right)(w) \\
&\overset{(18)}{=} \sum_{u,v} \binom{w}{u \mid v} \left(\frac{1}{1-A} \right)(v) \cdot \mathcal{N}(\sigma)(u) \\
&= \sum_{u,v} \binom{w}{u \mid v} (\Delta^u \sigma)(\varepsilon) \\
&\overset{(16)}{=} \sum_{u} \binom{w}{u} (\Delta^u \sigma)(\varepsilon).
\end{aligned}$$

□

Also, Theorem 6.2 generalises to weighted languages.

Theorem 10.2 (Euler expansion for languages). For all $\sigma \in k^{A^*}$,

$$\sigma = \sum_{a_1 \cdots a_n \in A^*} (\Delta^{a_1 \cdots a_n} \sigma)(\varepsilon) \times \frac{1}{1-A} \times a_1 \times \frac{1}{1-A} \times \cdots \times a_n \times \frac{1}{1-A},$$

where we understand this sum to include $\sigma(\varepsilon) \times \frac{1}{1-A}$, corresponding to $\varepsilon \in A^*$.

Proof.

$$\begin{aligned}
\sigma &\overset{\text{Theorem 9.2}}{=} \frac{1}{1-A} \otimes \mathcal{N}(\sigma) \\
&\overset{\text{Theorem 7.3}}{=} \frac{1}{1-A} \otimes \left(\sum_{w \in A^*} \mathcal{N}(\sigma)_w(\varepsilon) \times w \right) \\
&\overset{\text{Def. } \mathcal{N}}{=} \frac{1}{1-A} \otimes \left(\sum_{w \in A^*} (\Delta^w \sigma)(\varepsilon) \times w \right) \\
&= \sum_{w \in A^*} (\Delta^w \sigma)(\varepsilon) \times \left(\frac{1}{1-A} \otimes w \right) \\
&\overset{\text{Theorem 9.4}}{=} \sum_{w \in A^*} (\Delta^w \sigma)(\varepsilon) \times \frac{1}{1-A} \times \left(w \circ \frac{A}{1-A} \right) \\
&\overset{\text{Proposition 7.5}}{=} \sum_{a_1 \cdots a_n \in A^*} (\Delta^{a_1 \cdots a_n} \sigma)(\varepsilon) \times \frac{1}{1-A} \times a_1 \times \frac{1}{1-A} \times \cdots \times a_n \times \frac{1}{1-A}.
\end{aligned}$$

□

https://doi.org/10.1017/S0960129517000159 Published online by Cambridge University Press

11. Discussion

All our definitions are coinductive, given by behavioural differential equations, allowing all our proofs to be coinductive as well, that is, based on constructions of bisimulation (up-to) relations. This makes all proofs uniform and transparent. Moreover, coinductive proofs can be easily automated and often lead to efficient algorithms, for instance, as in Bonchi and Pous (2015). There are several topics for further research: (i) *Theorems 10.1 and 10.2* are pretty but are they also useful? We should like to investigate possible applications. (ii) The *infiltration product* deserves further study (including its restriction to streams, which seems to be new). It is reminiscent of certain versions of synchronised merge in process algebra (cf. Bergstra and Klop 1984), but it does not seem to have ever been studied there. (iii) *Theorem 9.2* characterises the Newton transform in terms of the shuffle product, from which many subsequent results follow. Recently, in Pin (2015), Newton series have been defined for functions from words to words. We are interested in seeing whether our present approach could be extended to those as well. (iv) *Behavioural differential equations* give rise to weighted automata (by what could be called the 'splitting' of derivatives into their summands, cf. Hansen et al. (2014)). We should like to investigate whether our representation results for Newton series could be made relevant for weighted automata as well. (v) Our new *Definition 8.4* of binomial coefficients for words, which seems to offer a precise generalisation of the standard notion for numbers, and, e.g., Pascal's formula, deserves further study.

We thank the anonymous referees for their constructive comments, and Joost Winter for pointing out a minor mistake in the conference version. H. Basold was supported by project 612.001.021 of the Netherlands Organisation for Scientific Research (NWO). H. H. Hansen was supported by Veni grant 639.021.231 of the Netherlands Organisation for Scientific Research (NWO). J.-E. Pin was supported by the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 670624).

References

Barbosa, L. (2001). *Components as Coalgebras*. PhD thesis, Universidade do Minho, Braga, Portugal.

Bartels, F. (2004). *On Generalised Coinduction and Probabilistic Specification Formats*. PhD thesis, Vrije Universiteit, Amsterdam.

Basold, H., Hansen, H.H., Pin, J.-E. and Rutten, J.J.M.M. (2015). Newton series, coinductively. In: Leucker, M., Rueda, C. and Valencia, F.D. (eds.) *Theoretical Aspects of Computing – ICTAC 2015*, Lecture Notes in Computer Science, vol. 9399, Springer, 91–109.

Bergeron, F., Labelle, G. and Leroux, P. (1998). *Combinatorial Species and Tree-like Structures*, Encyclopedia of Mathematics and its Applications, vol. 67. Cambridge University Press.

Bergstra, J. and Klop, J.W. (1984). Process algebra for synchronous communication. *Information and Control* **60** (1) 109–137.

Berstel, J. and Reutenauer, C. (1988). *Rational Series and their Languages*, EATCS Monographs on Theoretical Computer Science, vol. 12, Springer-Verlag.

Bonchi, F. and Pous, D. (2015). Hacking nondeterminism with induction and coinduction. *Communications of the ACM* **58** (2) 87–95.

https://doi.org/10.1017/S0960129517000159 Published online by Cambridge University Press

Burns, S.A. and Palmore, J.I. (1989). The newton transform: An operational method for constructing integral of dynamical systems. *Physica D: Nonlinear Phenomena* **37** (13) 83–90. ISSN 0167-2789.

Chen, K., Fox, R. and Lyndon, R. (1958). Free differential calculus, IV – The quotient groups of the lower series. *Annals of Mathemathics. Second Series* **68** (1) 81–95.

Comtet, L. (1974). *Advanced Combinatorics*, D. Reidel Publishing Company.

Conway, J.H. (1971). *Regular Algebra and Finite Machines*, Chapman and Hall.

Eilenberg, S. (1974). *Automata, Languages and Machines (Vol. A)*. Pure and Applied Mathematics. Academic Press.

Graham, R.L., Knuth, D.E. and Patashnik, O. (1994). *Concrete Mathematics*, 2nd edition, Addison-Wesley.

Hansen, H.H., Kupke, C. and Rutten, J.J.M.M. (2014). Stream differential equations: Specification formats and solution methods. Report FM-1404, CWI. Available at: www.cwi.nl.

Lothaire, M. (1997). *Combinatorics on Words*, Cambridge Mathematical Library, Cambridge University Press.

Niqui, M. and Rutten, J.J.M.M. (2011). A proof of Moessner's theorem by coinduction. *Higher-Order and Symbolic Computation* **24** (3) 191–206.

Pavlović, D. and Escardó, M. (1998). Calculus in coinductive form. In: *Proceedings of the 13th Annual IEEE Symposium on Logic in Computer Science*, IEEE Computer Society Press, 408–417.

Pin, J.-E. (2015). Newton's forward difference equation for functions from words to words. In: *Proceedings of the Evolving Computability – 11th Conference on Computability in Europe, CiE 2015*, Bucharest, Romania, 71–82. Available at: `http://dx.doi.org/10.1007/978-3-319-20028-6_8`.

Pin, J.E. and Silva, P.V. (2014). A noncommutative extension of Mahler's theorem on interpolation series. *European Journal of Combinatorics* **36** 564–578.

Rot, J. (2015). *Enhanced Coinduction*. Phd, University Leiden, Leiden.

Rot, J., Bonsangue, M.M. and Rutten, J.J.M.M. (2013). Coalgebraic bisimulation-up-to. In: *SOFSEM*, Lecture Notes in Computer Science, vol. 7741, Springer, 369–381.

Rutten, J.J.M.M. (2000). Universal coalgebra: A theory of systems. *Theoretical Computer Science* **249** (1) 3–80.

Rutten, J.J.M.M. (2003a). Coinductive counting with weighted automata. *Journal of Automata, Languages and Combinatorics* **8** (2) 319–352.

Rutten, J.J.M.M. (2003b). Behavioural differential equations: A coinductive calculus of streams, automata, and power series. *Theoretical Computer Science* **308** (1) 1–53.

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

Scheid, F. (1968). *Theory and Problems of Numerical Analysis (Schaum's outline series)*, McGraw-Hill.

https://doi.org/10.1017/S0960129517000159 Published online by Cambridge University Press