



Centrum voor Wiskunde en Informatica

REPORT*RAPPORT*

SEN

Software Engineering



Software ENgineering

Coinductive counting with weighted automata

J.J.M.M. Rutten

REPORT SEN-R0224 DECEMBER 31, 2002

CWI is the National Research Institute for Mathematics and Computer Science. It is sponsored by the Netherlands Organization for Scientific Research (NWO).

CWI is a founding member of ERCIM, the European Research Consortium for Informatics and Mathematics.

CWI's research has a theme-oriented structure and is grouped into four clusters. Listed below are the names of the clusters and in parentheses their acronyms.

Probability, Networks and Algorithms (PNA)

Software Engineering (SEN)

Modelling, Analysis and Simulation (MAS)

Information Systems (INS)

Copyright © 2001, Stichting Centrum voor Wiskunde en Informatica

P.O. Box 94079, 1090 GB Amsterdam (NL)

Kruislaan 413, 1098 SJ Amsterdam (NL)

Telephone +31 20 592 9333

Telefax +31 20 592 4199

ISSN 1386-369X

Coinductive Counting with Weighted Automata

J.J.M.M. Rutten*

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

Email: janr@cwi.nl, URL: www.cwi.nl/~janr

ABSTRACT

A general methodology is developed to compute the solution of a wide variety of basic counting problems in a uniform way: (1) the objects to be counted are enumerated by means of an infinite weighted automaton; (2) the automaton is reduced by means of the quantitative notion of stream bisimulation; (3) the reduced automaton is used to compute an expression (in terms of stream constants and operators) that represents the stream of all counts.

2000 Mathematics Subject Classification: 05A15, 68Q10, 68Q55, 68Q85

1998 ACM Computing Classification System: F.1, F.3

Keywords & Phrases: Coinduction, coalgebra, stream, derivative, differential equation, enumeration, combinatorics, counting, weighted automaton

Note: An extended abstract of this report appeared as Technical Report SEN-R0129. The present report has been accepted for publication in the Journal of Automata, Languages and Combinatorics.

*Also: Faculty of Mathematics and Computer Science, Vrije Universiteit, De Boelelaan 1081a, 1081 HV Amsterdam.

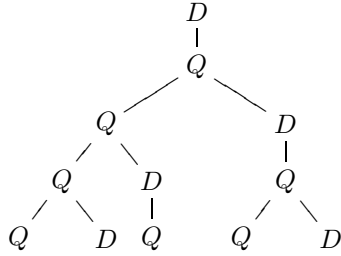
Contents

1	Motivation	3
2	Basic facts from stream calculus	5
3	Streams and weighted automata	9
4	An aside: splitting derivatives	10
5	Compositions of natural numbers	11
6	Surjections	12
7	Counting with probabilities	13
8	Well-bracketed words	14
9	Streams and continued fractions	15
10	More on well-bracketed words	17
11	Permutations	19
12	Set partitions	24
13	Special numbers	27
14	Discussion	27

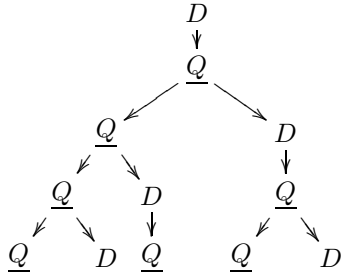
1 Motivation

There is a trend in combinatorial analysis started by several schools around 1980, where much emphasis is placed on the relations between combinatorial structure and the algebraic structure of generating functions. Over the years, this has led to several formalizations (addressing the foundational question: what *is* a combinatorial structure? How do we “specify” it? What is the relation of such specifications to counting?) being introduced by Goulden-Jackson [GJ83], Flajolet-Sedgewick [FS93, FS01], Joyal [BLL98], Stanley [Sta97, Sta99], and several others. Here we add one more formal system to the list, called the method of coinductive counting. From the enumerative point of view, it makes it possible to derive existing counting results in a new perspective. We first illustrate the method of coinductive counting by means of an example, and then summarize motivation and contents of the paper.

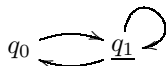
The following counting problem is taken from [GKP94, p.291]. Male bees are called drones and female bees are called queens. Drones are born out of a queen and have no father; a queen is born out of a father drone and a mother queen. The first few levels of the pedigree of a drone (drawn upside-down) look as follows:



We see that each drone has one mother, one grandmother, two great-grandmothers, three great-great-grandmothers, and so on. What is, for any $k \geq 0$, the number s_k of female ancestors at level k ? The key idea of coinductive counting is to use the very tree that enumerates all the (female) ancestors of a drone, as the basis for a representation of the infinite *stream* $\sigma = (s_0, s_1, s_2, \dots)$ containing all the answers. To this end, the tree is turned into a automaton, in which the arrows indicate transitions and in which all the queen states are output states:



(Formally, we shall be dealing with weighted automata over a one letter alphabet and with weights in the reals.) The *stream behaviour* of such automata will be defined coinductively, and can be expressed in terms of transition sequences. In the present automaton, the number s_k is encoded as the number of paths of length k leading from the initial (topmost) drone state to an output queen state, since there are as many such sequences as there are queens at level k . Thus we have translated the original counting problem into a question about streams and their representation by automata, thereby entering the coinductive world of *stream calculus* [Rut01]. A crucial ingredient of stream calculus is the notion of stream bisimulation, with the help of which the above automaton can be simplified by identifying all equivalent states as follows. Every drone state is bisimulation equivalent to the state q_0 and every queen state is equivalent to the state q_1 of the following two state automaton:



Intuitively, any queen state and the state q_1 have the same transition behaviour: both can take two transitions to states that are again, respectively, equivalent. Similarly for drone states and q_0 , which have only one transition. As a consequence, (the state q_0 of) this new automaton is an equivalent representation of our stream of answers σ : s_k corresponds again to the number of paths of length k leading from q_0 to the (in this case only) output state q_1 . A classical and convenient way to capture infinite sequences by means of one single expression, is the use of generating functions or, more generally, formal power series. In stream calculus, such a ‘closed expression’ for the infinite stream σ of answers represented by the state q_0 can be easily defined coinductively (see Section 2 for the formal computation), yielding

$$\sigma = \frac{X}{1 - X - X^2}$$

As with generating functions, this stream expression encodes the numbers s_k for all $k \geq 0$ (which turn out to be the Fibonacci numbers). In the present paper, we leave aside the computation of an explicit formula for s_k , and consider the above fraction, which is formulated in terms of stream constants and operators, as a satisfactory answer to our question.

Summarizing the above, we distinguish three phases in the procedure of coinductive counting:

1. *Enumeration* of the objects to be counted in an infinite, tree-shaped weighted automaton.
2. *Identification* of equivalent states using bisimulation.
3. *Expression* of the resulting stream of counts in terms of stream constants and operators.

As we shall see shortly, the entire approach is essentially quantitative: both transitions and output states will generally be labelled with *weights* (here: real numbers), which are taken into account by the notion of stream bisimulation (and bisimulation-up-to).

Although from the enumerative point of view, no new results are contained in the present paper, we feel that the method of coinductive counting has a number of contributions to make:

- Coinductive counting proves to be a very general and flexible method, by which many totally different structures can be counted in a uniform and simple way. This is to be contrasted with the use of many different methodologies in the discipline of enumerative combinatorics, such as context-free languages, tournament trees, symbolic counting, the transfer-matrix method, and many more. Moreover, coinductive counting leads in a number of cases to new representations of existing solutions.
- The heart of the method consists of the reduction of infinite weighted automata to much better structured (and often finite) ones, using bisimulation relations to make the necessary identification of states. Therewith, the method provides yet another illustration of the fundamental nature of bisimulation in mathematics, a notion originally stemming from the world of modal logic and the semantics of parallel programming languages.
- The method contains a number of elements that seem to be new to the theory of weighted automata. Notably, infinite weighted automata, which are usually not given much attention, play a crucial role. Furthermore, extensive use is made of the notion of stream derivative, based on a generalisation of Brzozowski’s notion of input derivative, both to go from weighted automata to streams, and vice versa. Finally, the method of coinductive counting yields a number of rather beautifully structured infinite weighted automata for many sequences of so-called special numbers.

Overview of the paper: Sections 2 through 4 summarize that part of stream calculus that is needed for the present paper. Then some first basic exercises in coinductive counting are treated in Sections 5 through 7, dealing with compositions of natural numbers, surjections, and counting with probabilities, respectively. The counting problem of Section 8 deals with well-bracketed words, and causes the need to develop some further theory. In Section 9, certain well-structured infinite weighted automata are studied, which have a stream behaviour that is expressible by

means of (generalisations of) continued fractions. The results of Section 9 are next used in Section 10, on well-bracketed words, in Section 11, on permutations, and in Section 12, on set partitions. The paper will be concluded with a discussion of related (and future) work.

2 Basic facts from stream calculus

We present the basic facts of a coinductive calculus of streams, by repeating parts of earlier work on formal power series and streams ([Rut00a] and [Rut01]). Since the concepts of bisimulation and coinduction may be new to some readers, a number of examples will be treated as well.

The set of all streams is defined by $\mathbb{R}^\omega = \{\sigma \mid \sigma : \{0, 1, 2, \dots\} \rightarrow \mathbb{R}\}$. Individual streams will be denoted by $\sigma = (\sigma(0), \sigma(1), \sigma(2), \dots) = (s_0, s_1, s_2, \dots)$. We shall call $\sigma(0) = s_0$ the *initial value* of σ . The *derivative* of a stream σ is defined by $\sigma' = (s_1, s_2, s_3, \dots)$.

In order to conclude that two streams σ and τ are equal, it is both necessary and sufficient to prove $\sigma(n) = \tau(n)$, for all $n \geq 0$. Sometimes this can be proved by a straightforward *induction* on the natural number n (prove $\sigma(0) = \tau(0)$ and show that $\sigma(n) = \tau(n)$ implies $\sigma(n+1) = \tau(n+1)$). More often than not, however, a succinct description or formula for $\sigma(n)$ and $\tau(n)$ will not exist, so that, consequently, induction simply cannot be applied.

Instead, we shall take a coalgebraic perspective on \mathbb{R}^ω , and use almost exclusively the proof principle of *coinduction*, which is formulated in terms of the following notion from the world of universal coalgebra [JR97, Rut00b]. A *stream bisimulation*, or bisimulation for short, is a relation $R \subseteq \mathbb{R}^\omega \times \mathbb{R}^\omega$ such that, for all σ and τ in \mathbb{R}^ω : if $\sigma R \tau$ then $\sigma(0) = \tau(0)$ and $\sigma' R \tau'$. (We shall use both $\sigma R \tau$ and $\langle \sigma, \tau \rangle \in R$ as equivalent notations.) The union of all bisimulation relations is called *bisimilarity*, and is denoted by \sim .

Theorem 2.1 [*Coinduction*] For all streams σ and τ in \mathbb{R}^ω , if $\sigma \sim \tau$ then $\sigma = \tau$.

Proof: Consider two streams σ and τ and let $R \subseteq \mathbb{R}^\omega \times \mathbb{R}^\omega$ be a bisimulation on \mathbb{R}^ω containing the pair $\langle \sigma, \tau \rangle$. Because R is a bisimulation, it follows by induction on n that $\langle \sigma^{(n)}, \tau^{(n)} \rangle \in R$, for all $n \geq 0$. Here $\sigma^{(n)}$ (and likewise $\tau^{(n)}$) denotes the n -th derivative of σ defined by $\sigma^{(0)} = \sigma$ and $\sigma^{(n+1)} = (\sigma^{(n)})'$. This implies, again because R is a bisimulation, that $\sigma^{(n)}(0) = \tau^{(n)}(0)$, for all $n \geq 0$. Because $\sigma^{(n)}(0) = \sigma(n)$ and $\tau^{(n)}(0) = \tau(n)$, this proves $\sigma = \tau$. \square

(Note that the converse trivially holds, since $\{\langle \sigma, \sigma \rangle \mid \sigma \in \mathbb{R}^\omega\}$ is a bisimulation relation on \mathbb{R}^ω .) Theorem 2.1 gives rise to the following, surprisingly powerful proof principle, called *coinduction*: in order to prove the equality of two streams σ and τ , it is sufficient to establish the existence of a bisimulation relation $R \subseteq \mathbb{R}^\omega \times \mathbb{R}^\omega$ with $\langle \sigma, \tau \rangle \in R$. We shall see some examples of coinductive proofs in this paper (a first one follows in a moment), but [Rut01] contains many more (and in much more detail).

Coinductive *definitions* are phrased in terms of derivatives and initial values, and are called *behavioural differential equations*. For any $r \in \mathbb{R}$ we denote the constant stream $(r, 0, 0, 0, \dots)$ by r again. The context will always make clear whether the real number or the stream r is intended. Another constant stream is $X = (0, 1, 0, 0, 0, \dots)$, which plays the role of a formal variable. Note that $r' = 0$ and that $X' = 1$. We shall use the following operators on streams, all of which are defined by means of a behavioural differential equation:

behavioural differential equation	initial value	name
$(\sigma + \tau)' = \sigma' + \tau'$	$(\sigma + \tau)(0) = \sigma(0) + \tau(0)$	sum
$(\sigma \times \tau)' = (\sigma' \times \tau) + (\sigma(0) \times \tau')$	$(\sigma \times \tau)(0) = \sigma(0) \times \tau(0)$	(convolution) product
$(\sigma^{-1})' = -\sigma(0)^{-1} \times (\sigma' \times \sigma^{-1})$	$(\sigma^{-1})(0) = \sigma(0)^{-1}$	inverse ($\sigma(0) \neq 0$)
$(\sigma \otimes \tau)' = (\sigma' \otimes \tau) + (\sigma \otimes \tau')$	$(\sigma \otimes \tau)(0) = \sigma(0) \times \tau(0)$	shuffle product
$(\sigma^{-1})' = -\sigma' \otimes (\sigma^{-1} \otimes \sigma^{-1})$	$\sigma^{-1}(0) = \sigma(0)^{-1}$	shuffle inverse ($\sigma(0) \neq 0$)

Whenever writing σ^{-1} or σ^{-1} , we shall silently assume that $\sigma(0) \neq 0$. Note that in the definition of $(\sigma \times \tau)'$, $\sigma(0)$ should be read as the stream $(\sigma(0), 0, 0, 0, \dots)$; and similarly, in the definition of

$(\sigma^{-1})', \sigma(0)^{-1}$ should be read as the stream $(\sigma(0)^{-1}, 0, 0, 0, \dots)$. As usual, we shall write

$$\sigma^0 \equiv 1, \quad \sigma^{n+1} \equiv \sigma \times \sigma^n, \quad \sigma^{-n} \equiv (\sigma^{-1})^n, \quad \frac{\sigma}{\tau} \equiv \sigma \times \tau^{-1}$$

for all $n \geq 0$. Moreover, the following conventions will be used:

$$\sigma^{\underline{0}} \equiv 1, \quad \sigma^{\underline{n+1}} \equiv \sigma \otimes \sigma^n, \quad \sigma^{-\underline{n}} \equiv (\sigma^{-1})^{\underline{n}}$$

The unique existence of a solution to the above equations is ultimately due to the fact that the combination of the operations of initial value and stream derivative:

$$\langle -(0), (-)' \rangle : \mathbb{R}^\omega \rightarrow \mathbb{R} \times \mathbb{R}^\omega, \quad \sigma \mapsto \langle \sigma(0), \sigma' \rangle$$

constitutes a *final coalgebra* structure on the set of streams. Formally, the existence of a unique solution follows from [Rut01, Thm 19.1], which deals with a rather large family of systems of behavioural differential equations, generalising the one above. In fact, all of the systems of behavioural differential equations that we shall encounter in this paper, can be shown to have a unique solution by an application of the same theorem.

The format of the above definitions is rather non-standard, and the reader is invited to consult [Rut01] for a comparison between these definitions and the (equivalent) traditional ones such as, for $n \geq 0$,

$$(\sigma \otimes \tau)(n) = \sum_{k=0}^n \binom{n}{k} \times \sigma(n-k) \times \tau(k) \tag{1}$$

for the shuffle product. The bottom line is that definitions by behavioural differential equations are formulated in terms of initial values and derivatives, which enable us to give proofs by coinduction, since these amount to the construction of bisimulation relations, which themselves are characterized in terms of initial values and derivatives.

We shall freely use various basic properties of the stream operators of sum, product, inverse, shuffle, and so on, such as $\sigma \times \tau = \tau \times \sigma$ and $(\sigma^{-1})^{-1} = \sigma$. Note that the derivatives of convolution product and its inverse are different from what we are used to in classical analysis. At the same time, the derivatives of shuffle product and inverse *are* given by the familiar formulae. Even if we do not always refer to it explicitly, [Rut01] is the source for all identities on streams and their operators that are not proved in the present paper.

Although the proof of the coinduction proof principle (Theorem 2.1 above) uses induction, the construction of a bisimulation for a pair $\langle \sigma, \tau \rangle$ is in general not by any use of induction. (Often, we even have no clue as to how the n -th element of the streams involved looks like.) Instead, bisimulation relations are constructed by simply closing the set $\{\langle \sigma, \tau \rangle\}$ under the taking of derivatives (while checking that initial values agree). The point of the use of bisimulation relations R is that instead of trying to prove (only) $\sigma = \tau$, we prove the stronger statement: $\alpha = \beta$, for all pairs $\langle \alpha, \beta \rangle \in R$. Here are a few examples:

- The identity $(1, 1, 1, \dots) = \frac{1}{1-X}$ follows by coinduction from the fact that

$$\{ \langle (1, 1, 1, \dots), \frac{1}{1-X} \rangle \}$$

is a bisimulation on \mathbb{R}^ω . To show that this singleton set is a bisimulation, one proves

$$\left(\frac{1}{1-X} \right)' = \frac{1}{1-X}$$

using the behavioural differential equations above. Similarly, in order to prove

$$(1, -1, 1, -1, \dots) = \frac{1}{1+X}$$

it suffices to show that

$$\{ \langle (1, -1, 1, -1, \dots), \frac{1}{1+X} \rangle, \langle (-1, 1, -1, 1, \dots), \frac{-1}{1+X} \rangle \}$$

is a bisimulation on \mathbb{R}^ω , which follows from $(\frac{1}{1+X})' = \frac{-1}{1+X}$.

- In order to appreciate the strength of coinduction in the next example, the reader is invited first to prove the following identity without coinduction, possibly using formula (1) above:

$$\frac{1}{1-X} \otimes \frac{1}{1+X} = 1$$

(Recall that $1 = (1, 0, 0, 0, \dots)$.) A proof by coinduction follows from the fact that

$$\{ \langle \frac{1}{1-X} \otimes \frac{1}{1+X}, 1 \rangle, \langle 0, 0 \rangle \}$$

is a bisimulation, which is immediate consequence of the basic properties of the operators involved, notably

$$\begin{aligned} (\frac{1}{1-X} \otimes \frac{1}{1+X})' &= (\frac{1}{1-X} \otimes \frac{1}{1+X}) + (\frac{1}{1-X} \otimes \frac{-1}{1+X}) \\ &= (\frac{1}{1-X} \otimes \frac{1}{1+X}) - (\frac{1}{1-X} \otimes \frac{1}{1+X}) \\ &= 0 \end{aligned}$$

- For a function $f : \mathbb{R} \rightarrow \mathbb{R}$ that is analytical in 0, the stream $T(f)$ of Taylor coefficients of $f(x)$ can be defined, coinductively, by the following behavioural differential equation over \mathbb{R}^ω (cf. [PE98]):

behavioural differential equation	initial value	name
$T(f)' = T(df/dx)$	$T(f)(0) = f(0)$	Taylor series

(Here df/dx denotes the analytical derivative of the function $f(x)$.) One can readily check that this defines $T(f)$ to be the stream $(f(0), df/dx(0), d^2f/dx^2(0), \dots)$. We have the following identity:

$$T(\sin(x)) = \frac{X}{1+X^2}$$

A proof by coinduction simply consists of the observation that the relation

$$\begin{aligned} &\{ \langle T(\sin(x)), \frac{X}{1+X^2} \rangle, \langle T(\cos(x)), \frac{1}{1+X^2} \rangle, \\ &\langle T(-\sin(x)), \frac{-X}{1+X^2} \rangle, \langle T(-\cos(x)), \frac{-1}{1+X^2} \rangle \} \end{aligned}$$

is a bisimulation relation on \mathbb{R}^ω . This observation can be readily proved, using the facts that

$$(\frac{X}{1+X^2})' = \frac{1}{1+X^2}, \quad (\frac{1}{1+X^2})' = \frac{-X}{1+X^2}$$

which are immediate consequences of the defining behavioural differential equations for convolution product and inverse.

As was mentioned above, for large classes of behavioural differential equations the existence of a unique solution is an immediate consequence of the fact that \mathbb{R}^ω is a final coalgebra. This insight does not necessarily provide us with much information on how that solution looks like. Fortunately, one can compute for many, relatively simple, behavioural differential equations, an explicit so-called closed form for the solution, that is, an expression that is build from (the above) stream constants and operators. The main tool for obtaining such a closed form is the following simple but useful ‘Fundamental Theorem’ [Rut01]: for all streams $\sigma \in \mathbb{R}^\omega$,

$$\sigma = \sigma(0) + (X \times \sigma') \quad (2)$$

The theorem expresses how σ can be obtained from its initial value $\sigma(0)$ and its derivative σ' . (Note that multiplication with X can be considered as a type of stream *integration*.) In order to illustrate the use of this theorem, we consider a stream σ that is defined by the following behavioural differential equation: $\sigma(0) = 1$ and $\sigma' = 2 \times \sigma$. Clearly, this defines σ as the stream $(2^0, 2^1, 2^2, \dots)$. Using (2), we can also compute a more succinct expression, since $\sigma = \sigma(0) + (X \times \sigma') = 1 + (X \times 2 \times \sigma)$ implies $(1 - (2 \times X)) \times \sigma = 1$, whence

$$\sigma = \frac{1}{1 - (2 \times X)}$$

For any stream σ and $n \geq 0$, the number $\sigma(n)$ can be obtained by repeatedly computing derivatives, according to the following identity

$$\sigma(n) = \sigma^{(n)}(0)$$

Writing, as usual, s_n for $\sigma(n)$, one can easily prove the following Taylor-like expansion for any stream σ :

$$\sigma = s_0 + (s_1 \times X) + (s_2 \times X^2) + \dots$$

but note the absence of the factorial coefficients. (Here we use an operation of infinite sum, which can be defined by generalising the definition of binary sum above.) For the example above, this yields

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

It is worthwhile to note that in stream calculus, all of these equalities are formal identities. This is to be contrasted to the use of, for instance, generating functions, which are used to *represent* or *encode* streams.

There is the following mild generalisation of the notion of bisimulation, which often makes it easier to give coinductive proofs. A *bisimulation-up-to* is a relation $R \subseteq \mathbb{R}^\omega \times \mathbb{R}^\omega$ such that, for all $\sigma, \tau \in \mathbb{R}^\omega$: if $\sigma R \tau$ then $\sigma(0) = \tau(0)$ and there exist $n \geq 0$ and streams $\alpha_0, \dots, \alpha_n$ and β_0, \dots, β_n , such that $\sigma' = \alpha_0 + \dots + \alpha_n$ and $\tau' = \beta_0 + \dots + \beta_n$ and, for all $0 \leq i \leq n$: either $\alpha_i = \beta_i$ or $\alpha_i R \beta_i$. The notion of bisimulation-up-to is used in the following strengthening of the coinduction proof principle, called *coinduction-up-to*: if $\sigma R \tau$ and R is a bisimulation-up-to then $\sigma = \tau$. For a simple but typical example, consider streams σ, τ and ρ such that $\sigma(0) = \tau(0) = \rho(0) = 1$, $\sigma' = 2 \times \sigma$, and $\tau' = \rho' = \tau + \rho$. Then $\{\langle \sigma, \tau \rangle, \langle \sigma, \rho \rangle\}$ is a bisimulation-up-to (note that $\sigma' = 2 \times \sigma = \sigma + \sigma$), and $\sigma = \tau$ follows by coinduction-up-to.

We shall also encounter the following notion. A stream is called *rational* if it can be defined using constant streams (r and X), sum, convolution product and convolution inverse. A typical example is $(1 + X)/(3 - X + 7X^2)$. Rational streams are closed under applications of the shuffle product, but they are *not* closed under shuffle inverse. A typical example of a non-rational stream is $(1 - X)^{-1} = (0!, 1!, 2!, \dots)$.

3 Streams and weighted automata

A stream can often be conveniently represented by means of a *weighted automaton* $Q = (Q, \langle o, t \rangle)$ consisting of a (generally infinite) set Q of states, together with an output function $o : Q \rightarrow \mathbb{R}$, and a transition function $t : Q \rightarrow (Q \rightarrow_f \mathbb{R})$ (the latter set contains functions $\phi : Q \rightarrow \mathbb{R}$ of finite *support*, that is, such that $\{q \in Q \mid \phi(q) \neq 0\}$ is finite). The output function o assigns to each state q in Q a real number $o(q)$ in \mathbb{R} . The transition function t assigns to a state q in Q a function $t(q) : Q \rightarrow_f \mathbb{R}$, which specifies for any state q' in Q a real number $t(q)(q')$ in \mathbb{R} . This number can be thought of as the *weight* or *multiplicity* with which the transition from q to q' occurs. The following notation will be used: $q \xrightarrow{r} q'$ denotes $t(q)(q') = r$ and \underline{q}_r denotes $o(q) = r$. Moreover, for $t(q)(q') = 1$ we shall often simply write $q \rightarrow q'$, and we write \underline{q} when $o(q) = 1$, in which case q is called an *output state*. In pictures, we usually include only non-zero output values and only arrows with a non-zero label.

The *stream behaviour* $S(q) \in \mathbb{R}^\omega$ of a state q in a weighted automaton $(Q, \langle o, t \rangle)$, can be defined in two equivalent ways. First, there is the following formula, for any $k \geq 0$:

$$S(q)(k) = \sum \{l_1 \times \cdots \times l_k \times l \mid \exists q_0, \dots, q_k : q = q_0 \xrightarrow{l_1} \cdots \xrightarrow{l_k} q_k \text{ and } o(q_k) = l\} \quad (3)$$

This formula computes the k -th element $S(q)(k)$ of the stream $S(q)$ as a weighted count of the number of paths (transition sequences) of length k that start in the state q (and end in a state with a non-zero output). It is precisely this aspect of counting paths that makes weighted automata very suitable for the representation of counting problems in general.

At the same time, formula (3) does not yield any compact representations for $S(q)$ and is thereby not very suited for actual reasoning. And that is where we shall benefit from the coinductive calculus of streams, sketched above. More specifically, consider again a state q in a weighted automaton $(Q, \langle o, t \rangle)$, and let $\{q_1, \dots, q_n\}$ be the set of all states q' for which $t(q)(q') \neq 0$. Then the stream behaviour $S(q) \in \mathbb{R}^\omega$ of the state q can also be defined by the following behavioural differential equation (or rather, system of behavioural differential *equations*, one for each state q):

differential equation	initial value
$S(q)' = t(q)(q_1) \times S(q_1) + \cdots + t(q)(q_n) \times S(q_n)$	$S(q)(0) = o(q)$

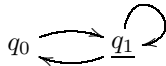
(As before, the unique existence of a solution for this system of equations is an immediate consequence of [Rut01, Thm 19.1].) It follows from the ‘Fundamental Theorem’ (2) in Section 2 that this system of behavioural differential equations is equivalent to the following system of equations:

$$S(q) = o(q) + (t(q)(q_1) \times X \times S(q_1)) + \cdots + (t(q)(q_n) \times X \times S(q_n))$$

If the automaton (that is, the state space Q) is finite, this is a finite system of equations, which can be solved in the familiar algebraic way, yielding rational streams as solution.

It is not very difficult to prove the equivalence of the definition of $S(q)$ by formula (3), on the one hand, and the definition of $S(q)$ by means of behavioural differential equations, on the other. See [Rut01, Prop. 13.1] for a formal proof.

As an example of the stream behaviour of a weighted automaton, consider the automaton defined by the following picture

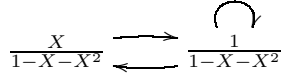


which occurred in the counting example of Section 1. The stream $S(q_0)$ represented by the state q_0 can be computed in two ways. According to formula (3), one finds $S(q_0)(k)$ by simply counting (since all weights are 1) the number of paths of length k from q_0 to q_1 (which is the only state with a non-zero output), yielding $S(q_0) = (0, 1, 1, 2, 3, 5, 8, \dots)$. By the second definition, one has $S(q_0) = X \times S(q_1)$, and $S(q_1) = 1 + (X \times S(q_0)) + (X \times S(q_1))$, yielding $S(q_0) = X / (1 - X - X^2)$.

4 An aside: splitting derivatives

We have explained now most of what is needed for the formulation and solution of combinatorial counting problems in a coinductive manner. There still is yet another point we would like to make before turning to the actual subject of this paper. Technically speaking, it will not play a role in what follows, but it does provide a further explanation of the close relationship between streams and weighted automata, and of the fundamental importance of the notion of stream derivative.

In Section 3, we have explained how to go from automata to streams. The converse is also possible, by a procedure that we have called ‘splitting of derivatives’, and which we shall explain next by means of a small example. Consider the stream $X/1 - X - X^2$ that was obtained as the behaviour of the state q_0 of the automaton at the end of Section 3. Computing its (repeated) derivatives, we find $(X/1 - X - X^2)' = 1/1 - X - X^2$, and $(1/1 - X - X^2)' = (1/1 - X - X^2) + (X/1 - X - X^2)$. We can now take the expressions $1/1 - X - X^2$ and $X/1 - X - X^2$ as the states of the following automaton, where the transitions are determined by the derivatives:



So we use the fact that the derivative of $1/1 - X - X^2$ consists of a sum, to define two transitions, to each of the summands. The fact that $(1/1 - X - X^2)(0) = 1$ determines $1/1 - X - X^2$ to be an output state. Using the coinductive definition of the stream behaviour of weighted automata of the previous section, we find that the behaviour of the state $X/1 - X - X^2$ in the obtained automaton is exactly the stream $X/1 - X - X^2$ (and similarly for $1/1 - X - X^2$).

Of course, there are in general many different ways of splitting a derivative, yielding different but equivalent weighted automata as representations of the same stream. For certain classes of streams, however, this procedure can be described and used in a systematic way (cf. [Rut99, Rut00a]). An example is the class of rational streams, for which the splitting of derivatives gives rise to (minimal) finite weighted automata representations.

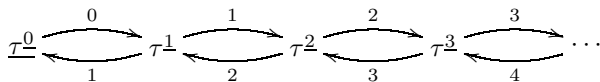
Interestingly, splitting derivatives can also be applied to streams that are not rational, and often gives rise to infinite but very well structured weighted automata representations. This latter point is illustrated by the following example, repeated here from the afore mentioned references. Consider the stream τ defined by the following behavioural differential equation:

differential equation	initial value
$\tau' = 1 + (\tau \otimes \tau)$	$\tau(0) = 0$

It is actually the stream of the so-called tangent numbers, which are the Taylor coefficients of the tangent function $\tan(x)$. Recall that τ^n was defined as the shuffle product of τ with itself, n times. Using a little bit of basic stream calculus, the derivative of τ^n in stream can be computed as follows:

$$\begin{aligned}
 (\tau^n)' &= n \times (\tau' \otimes \tau^{n-1}) \\
 &= n \times (1 + (\tau \otimes \tau)) \otimes \tau^{n-1} \quad [\text{using the differential equation for } \tau] \\
 &= (n \times \tau^{n-1}) + (n \times \tau^{n+1})
 \end{aligned}$$

This gives rise to the following automaton, in which every state τ^n has two n -labelled transitions, by splitting its derivative into its two summands τ^{n-1} and τ^{n+1} (the transition from $\tau^0 = 1$ to τ^1 has been included for symmetry only):

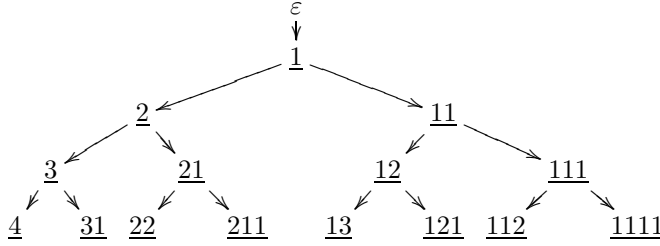


(The only output state is $\tau^0 = 1$, since it has initial value 1.) This infinite but pretty automaton representation for the stream of the tangent numbers was constructed by computing and splitting stream derivatives (using the defining behavioural differential equation for τ above). We shall encounter the same automaton later again, as the outcome of one of our coinductive counting exercises.

5 Compositions of natural numbers

A *composition* of a natural number $k \geq 0$ is a sequence of natural numbers $n_1 \cdots n_l$ such that $k = n_1 + \cdots + n_l$. What is, for any $k \geq 0$, the number s_k of compositions of k ? Or, equivalently, what is the stream of all counts $\sigma = (s_0, s_1, s_2, \dots)$? The problem will be solved coinductively by going through the three phases mentioned in the introduction.

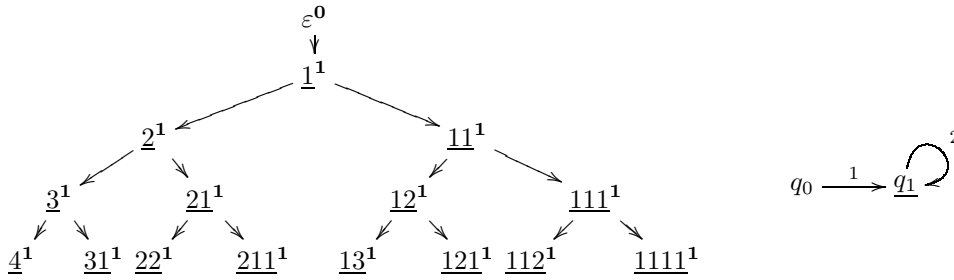
Enumeration: The following automaton enumerates all compositions for all natural numbers (here and in what follows, pictures show only the first few levels of what is understood to be an infinite automaton):



The automaton of this picture can be formally described by defining a state set $Q = \{w \mid w \in \mathbb{N}^*\}$; an output function $o : Q \rightarrow \mathbb{R}$ given by $o(\varepsilon) = 0$ and $o(w) = 1$ for all $w \neq \varepsilon$; and a transition function $t : Q \rightarrow (Q \rightarrow_f \mathbb{R})$, defined by $t(v)(w) = 1$ iff either $w = v \cdot 1$ or $(v = u \cdot k$ and $w = u \cdot (k + 1)$, for some $u \in Q$ and $k \in \mathbb{N}$), and by $t(v)(w) = 0$ otherwise. (Here $u \cdot k$ denotes the concatenation of the word u and the one letter word k .) In what follows, however, we shall present automata usually by their pictures, becoming more formal only when strictly necessary.

Let $k \geq 0$ be any natural number. There is a one-to-one correspondence between paths of length k starting in the (topmost) initial state ε , on the one hand, and compositions of the natural number k (all of which are situated at the k -th level of the automaton), on the other. As a consequence, the total number s_k of all compositions of k is equal to the total number of paths of length k starting in the initial state ε . Therefore, it follows from formula (3) of Section 2 that the stream behaviour $S(\varepsilon)$ of the initial state ε equals the stream $\sigma = (s_0, s_1, s_2, \dots)$ of answers we are after: $S(\varepsilon) = \sigma$.

Identification: Next we identify as many states as possible by defining a bisimulation-up-to between (the streams represented by) our infinite weighted automaton, repeated below on the left, and the tiny 2 state automaton on the right:



The superscripts that we have added to the states of our automaton on the left, indicate to which state in the automaton on the right they are related. More explicitly and precisely, the above picture suggests the definition of a relation $R \subseteq \mathbb{R}^\omega \times \mathbb{R}^\omega$ as follows:

$$R = \{\langle S(\varepsilon), S(q_0) \rangle\} \cup \{\langle S(w), S(q_1) \rangle \mid w \in \mathbb{N}^*, w \neq \varepsilon\}$$

relating the streams represented by (that is, the stream behaviour of) the states in the automaton on the left, with the streams represented by the states q_0 and q_1 on the right. In order to show that R is a bisimulation-up-to, first note that the initial values of all related pairs match: $S(\varepsilon)(0) = 0 = S(q_0)(0)$ and, for all words $v \in \mathbb{N}^*$ with $v \neq \varepsilon$, $S(v)(0) = 1 = S(q_1)(0)$. Next

we check derivatives: $S(\varepsilon)' = S(1)$, which is related to $S(q_1) = S(q_0)'$; and for all words $v \in \mathbb{N}^*$ and natural numbers n , we have $S(v \cdot n)' = S(v \cdot (n+1)) + S(v \cdot n \cdot 1)$, each component of which is related to $S(q_1)$, thus matching $S(q_1)' = 2 \times S(q_1) = S(q_1) + S(q_1)$. This proves that R is a bisimulation-up-to. Now it follows by coinduction-up-to that $S(\varepsilon) = S(q_0)$.

Expression: We can now easily compute a closed expression for $\sigma = S(q_0)$ by solving the system of equations, consisting of $S(q_0) = X \times S(q_1)$ and $S(q_1) = 1 + (2 \times X \times S(q_1))$, yielding

$$\sigma = S(q_0) = \frac{X}{1 - 2X}$$

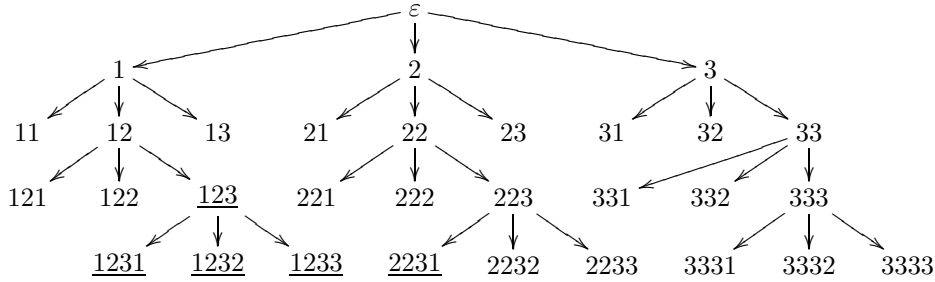
(The latter stream can be readily proved to be equal to the stream $(0, 2^0, 2^1, 2^2, \dots)$, whence $s_k = 2^{k-1}$, but as we already announced in the introduction, in this paper we shall not deal with the analysis of finding closed formulae for the individual numbers s_k .)

It is worthwhile emphasizing the quantitative aspect of the notion of bisimulation (up-to): the fact that any state of the original weighted automaton labelled by a non-empty word w can take *two* transitions to similar such states, is reflected by a *2-labelled* transition from q_1 to itself.

6 Surjections

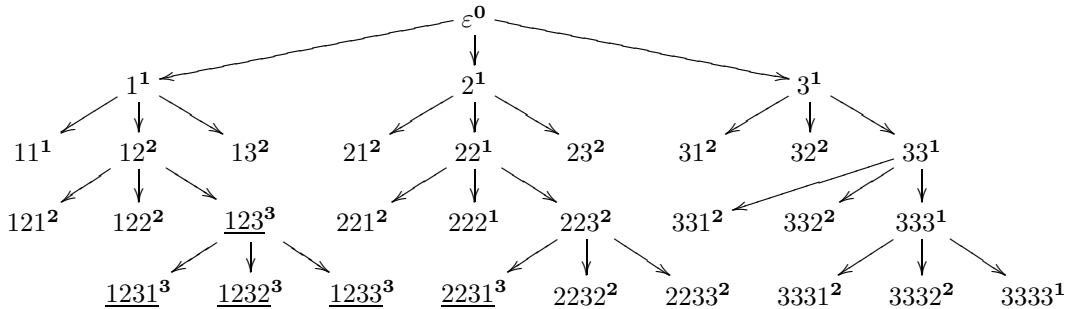
What is, for any natural number $k \geq 0$, the number s_k of surjections from the set $\{1, \dots, k\}$ onto the set $\{1, 2, 3\}$ (defining s_0 to be 0)? Below we shall see how the answer can be generalized to surjections onto the set $\{1, \dots, n\}$, for a fixed but arbitrary $n \geq 1$.

Enumeration: Let us denote a function $f : \{1, \dots, k\} \rightarrow \{1, 2, 3\}$ by means of the word $f(1) \dots f(k)$. The following automaton enumerates at each level k all such functions:

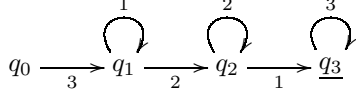


Note that all states labelled by a word representing a surjection (that is, containing at least one 1, one 2, and one 3), have been defined as output states. Also note that we have not only restricted the picture to the first five levels of the automaton, but that moreover not all transitions have been included, for lack of space. As before, it follows from (3) that the initial state ε represents the stream $\sigma = (s_0, s_1, s_2, \dots)$ of answers we are interested in.

Identification: The automaton can be simplified by identifying all states (labelled by words) that have the same number of different symbols, as indicated by the superscripts below:



More precisely, (the streams represented by) the i -superscripted states above can be related with (the stream represented by) the state q_i in the automaton below,



by means of a relation $R \subseteq \mathbb{R}^\omega \times \mathbb{R}^\omega$ that is defined as follows:

$$R = \{ \langle S(w), S(q_i) \rangle \mid w \in \{1, 2, 3\}^*, i \in \{0, 1, 2, 3\} : \sharp(w) = i \}$$

where, for this occasion, $\sharp(w)$ denotes the number of different symbols contained in w . It is straightforward to check that R is a bisimulation-up-to, from which $S(\varepsilon) = S(q_0)$ follows by coinduction-up-to.

Expression: The stream behaviour of our 4-state automaton is determined by the following system of equations:

$$\begin{aligned} S(q_0) &= 3 \times X \times S(q_1) \\ S(q_1) &= (1 \times X \times S(q_1)) + (2 \times X \times S(q_2)) \\ S(q_2) &= (2 \times X \times S(q_2)) + (1 \times X \times S(q_3)) \\ S(q_3) &= 1 + (3 \times X \times S(q_3)) \end{aligned}$$

Solving this system of equations, one finds:

$$\sigma = S(q_0) = \frac{3!X^3}{(1-X)(1-2X)(1-3X)}$$

The above can be easily generalised. Let $n \geq 1$ be arbitrary but fixed, and let t_k , for any $k \geq 0$, be the number of all surjections from the set $\{1, \dots, k\}$ onto the set $\{1, \dots, n\}$. Then the stream of counts $\tau = (t_0, t_1, t_2, \dots)$ is given by

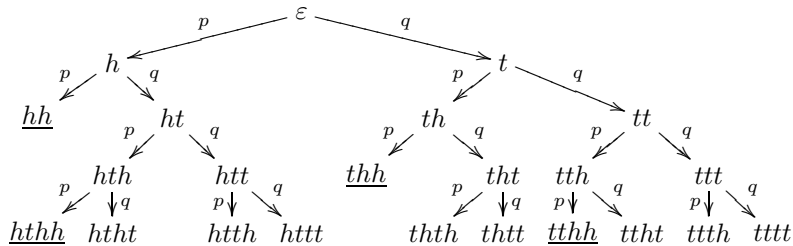
$$\begin{aligned} \tau &= \frac{n!X^n}{(1-X)(1-2X) \cdots (1-nX)} \\ &= \left(\frac{1}{1-X} - 1 \right)^n \end{aligned}$$

where the latter equality can be proved using some basic stream calculus.

7 Counting with probabilities

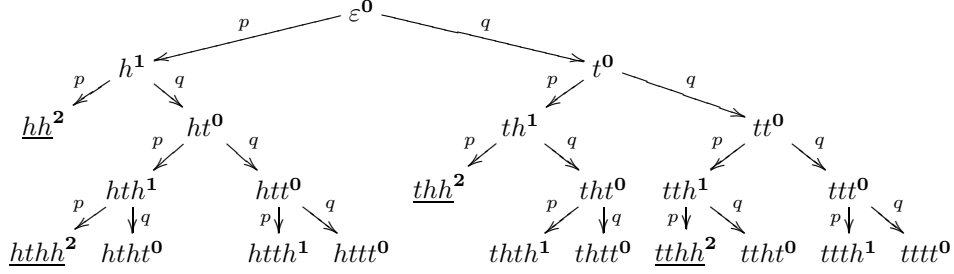
Consider a not necessarily fair coin with probability p of producing a head and probability $q = 1 - p$ of producing a tail. What is, for any $k \geq 0$, the probability s_k of getting, by flipping the coin k times, a sequence of heads and tails (of length k) without the occurrence of two consecutive heads but for the two very last outcomes, which are required to be heads? We hope the reader is by now already sufficiently experienced with coinductive counting, to be able to supply the details that have been left out in the succinct presentation of the solution below.

Enumeration: Here is a weighted automaton describing all possible scenarios:

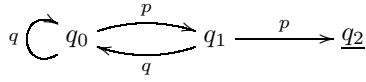


All states that are (labelled with a sequence) ending in two heads are output states, and have no further transitions. The stream $\sigma = (s_0, s_1, s_2, \dots)$ of all counts is represented by the initial state ε , that is, $\sigma = S(\varepsilon)$.

Identification: States can be identified according to the number of final heads:



yielding the following automaton:



More precisely, one can prove by coinduction-up-to that $S(\varepsilon) = S(q_0)$.

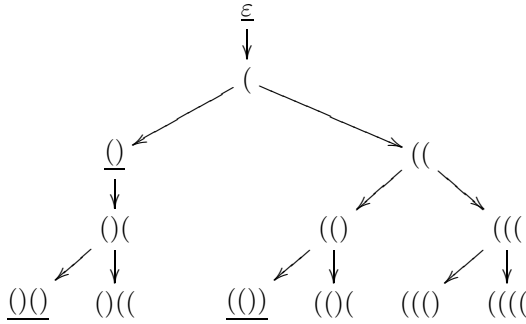
Expression: The stream behaviour of $S(q_0)$ can be easily computed:

$$\sigma = S(q_0) = \frac{p^2 X^2}{1 - qX - pqX^2}$$

8 Well-bracketed words

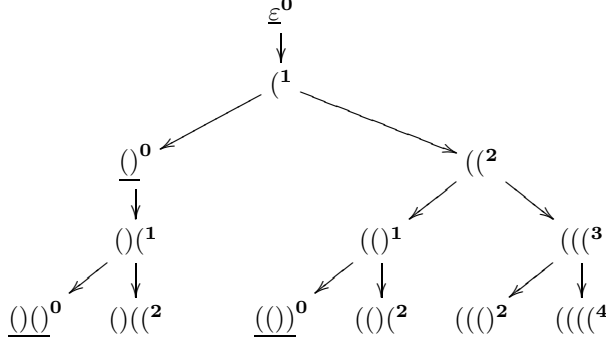
So far the identification phase of our coinductive counting exercises, has always resulted in a finite automaton yielding a rational stream. Here is an example for which the stream of counts is no longer rational (but *algebraic*). Consider a two letter alphabet $\{ (,) \}$ consisting of a left and a right bracket. What is, for any $k \geq 0$, the number s_k of well-bracketed words over this alphabet, of length k ?

Enumeration: The output states at level k of the following automaton correspond precisely to all such words:

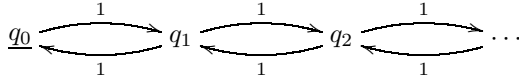


so the stream $\sigma = (s_0, s_1, s_2, \dots)$ of all counts is represented by the initial state: $\sigma = S(\varepsilon)$.

Identification: We identify states according to the number of left brackets they contain without a matching right bracket:



This yields the following well-structured, but still infinite automaton:



As before, one can prove $S(\varepsilon) = S(q_0)$ by coinduction-up-to.

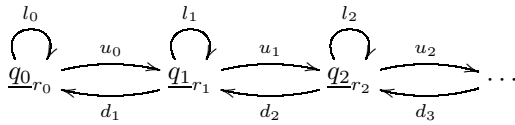
Expression: The stream behaviour of the above automaton is given by the following infinite system of equations: $S(q_0) = 1 + (X \times S(q_1))$ and

$$\boxed{S(q_i) = (X \times S(q_{i-1})) + (X \times S(q_{i+1})) \mid i \geq 1}$$

but it is not immediately obvious how this infinite system of equations can be solved. Rather than trying to solve it, we shall in Section 9 develop some more general results directly on infinite automata of the type above. The discussion of the solution of the present problem is therefore postponed until Section 10.

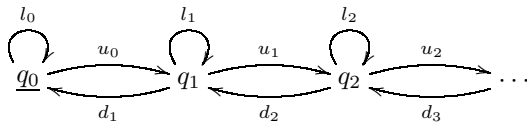
9 Streams and continued fractions

In this section, we study the stream behaviour of infinite weighted automata of the following type:



All of the labels and output values are arbitrary real numbers. The labels l_i and u_i , for $i \geq 0$, and d_i , for $i \geq 1$, could be pronounced as *level*, *up*, and *down*. The stream behaviour of this automaton will be expressed by means of a (generalised) continued fraction, but before we deal with it in its full generality, we first treat two special cases that are particularly relevant for many of the counting problems to come.

The first case concerns the situation that all output values r_i are 0 but for the output value r_0 of the first state q_0 , which (may be arbitrary but for convenience) has been chosen to be 1:



Theorem 9.1 *The stream behaviour $S(q_0)$ of the state q_0 in the automaton above is given by the following continued fraction:*

$$S(q_0) = \frac{1}{1 - (l_0 \times X) - \frac{(u_0 \times X) \times (d_1 \times X)}{1 - (l_1 \times X) - \frac{(u_1 \times X) \times (d_2 \times X)}{1 - (l_2 \times X) - \frac{(u_2 \times X) \times (d_3 \times X)}{\ddots}}}} \quad (4)$$

Proof: Let us first make sense of the continued fraction, which is denoted by what seems to be an infinite expression. Formally, it can be defined as follows. Let the streams $\sigma_0, \sigma_1, \dots$ be the unique solutions of the following system of behavioural differential equations:

differential equation	initial value
$\sigma'_i = (l_i \times \sigma_i) + (u_i \times \sigma_{i+1} \times d_i \times X \times \sigma_i)$	$\sigma_i(0) = 1$

By the ‘Fundamental Theorem’ (2), it follows that

$$\sigma_i = \frac{1}{1 - (l_i \times X) - (u_i \times X) \times \sigma_{i+1} \times (d_{i+1} \times X)}$$

for all $i \geq 0$. The infinite continued fraction in the theorem can now simply be read as a suggestive notation for σ_0 . Next we show that $S(q_0) = \sigma_0$. Consider the following relation on streams:

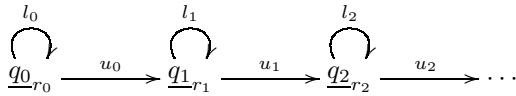
$$R = \{ \langle r \times S(q_i), r \times \tau_i \rangle \mid r \in \mathbb{R}, i \geq 0 \}$$

where the streams τ_i are defined by

$$\tau_i = \sigma_i \times (d_i \times X) \times \sigma_{i-1} \times (d_{i-1} \times X) \times \dots \times \sigma_1 \times (d_1 \times X) \times \sigma_0$$

Because R is a bisimulation relation on \mathbb{R}^ω , $S(q_0) = \sigma_0$ follows by coinduction. \square

For the second special case of the automaton presented at the beginning of this section, we assume that $d_i = 0$ for all $i \geq 1$, that is, there are no downward transitions. In contrast to the automaton of Theorem 4, however, all states may have again a non-trivial output value $r_i \in \mathbb{R}$:

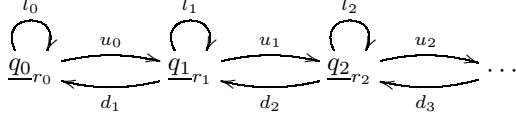


Theorem 9.2 *The stream behaviour $S(q_0)$ of the state q_0 is given by the following ‘upward’ continued fraction:*

$$S(q_0) = \frac{r_0 + (u_0 \times X) \times \frac{r_1 + (u_1 \times X) \times \frac{r_2 + (u_2 \times X) \times \frac{\vdots}{1 - (l_3 \times X)}}{1 - (l_2 \times X)}}{1 - (l_1 \times X)}}{1 - (l_0 \times X)} \quad (5)$$

The proof is omitted. It can be given along the same lines as the proof of Theorem 4.

Finally, we return to the automaton at the beginning of this section, which is the most general of all:



Theorem 9.3 *The stream behaviour $S(q_0)$ of the state q_0 is given by the following crazy expression, which consists of (nested) continued fractions growing both upward and downward:*

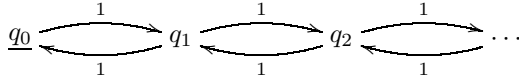
$$S(q_0) = \frac{r_0 + (u_0 \times X) \times \frac{r_1 + (u_1 \times X) \times (\dots)}{1 - (l_1 \times X) - (u_1 \times X) \times (\dots) \times (d_2 \times X)}}{1 - (l_0 \times X) - (u_0 \times X) \times \frac{r_1 + (u_1 \times X) \times (\dots)}{1 - (l_1 \times X) - (u_1 \times X) \times (\dots) \times (d_2 \times X)} \times (d_1 \times X)}$$

Also this theorem can be proved in the same manner as the previous two theorems.

10 More on well-bracketed words

We return to the problem of Section 8, and compute a closed expression for the stream $\sigma = (s_0, s_1, s_2, \dots)$ of numbers s_k of well-bracketed words of length k (over the alphabet $\{(\cdot, \cdot)\}$). In addition, we shall also look at a variation of this problem, by counting well-bracketed words consisting of k *pairs* of matching brackets.

At the end of Section 8, the following automaton representation for the stream σ of counts was found:



An application of Theorem 9.1 yields the following continued fraction for $\sigma = S(q_0)$:

$$\sigma = S(q_0) = \frac{1}{1 - \frac{X^2}{1 - \frac{X^2}{\ddots}}}$$

Because of the regular structure of this fraction, it is possible to come up with a more succinct expression for σ . It is easy to see that

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

(cf. the proof of Theorem 9.1). This implies that σ is a solution of the following quadratic equation:

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

In order to solve it, we need a square root operator on streams. As usual in stream calculus, its definition can be given by means of a behavioural differential equation. Let $\sqrt{\sigma}$, for all streams σ with $\sigma(0) > 0$, be defined as the unique stream satisfying the following differential equation:

behavioural differential equation	initial value
$(\sqrt{\sigma})' = \frac{\sigma'}{\sqrt{\sigma(0)} + \sqrt{\sigma}}$	$\sqrt{\sigma}(0) = \sqrt{\sigma(0)}$

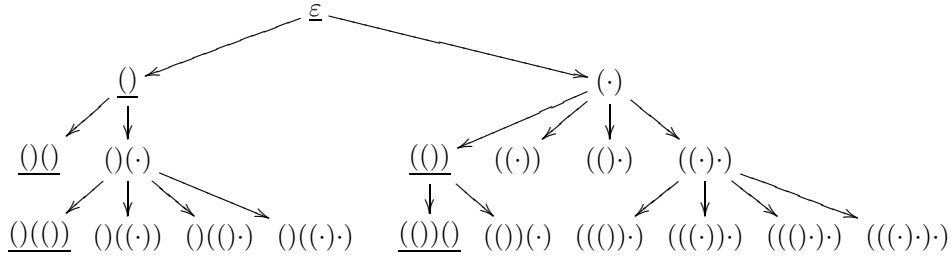
(Here $\sqrt{\sigma(0)}$ is the square root of the real number $\sigma(0)$.) With this operator, quadratic equations like the one above can be solved in much the same way we are used to in analysis (see [Rut01] for details). This yields as the final outcome for our stream of counts of well-bracketed words:

$$\sigma = \frac{2}{1 + \sqrt{1 - 4X^2}}$$

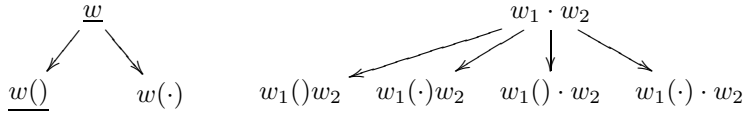
(which equals the stream $(1, 0, 1, 0, 2, 0, 5, 0, 14, 0, \dots)$ having the Catalan numbers at the even positions).

Sometimes there are various ways to enumerate the structures to be counted, by means of different automata, and often this leads to new ways of expressing the stream of counts. As an example, we tackle the counting problem of well-bracketed words again, in a slightly different form: What is, for any $k \geq 0$, the number s_k of well-bracketed words consisting of k matching *pairs* of an opening and a closing bracket?

Enumeration: Here is one way of enumerating all such words:

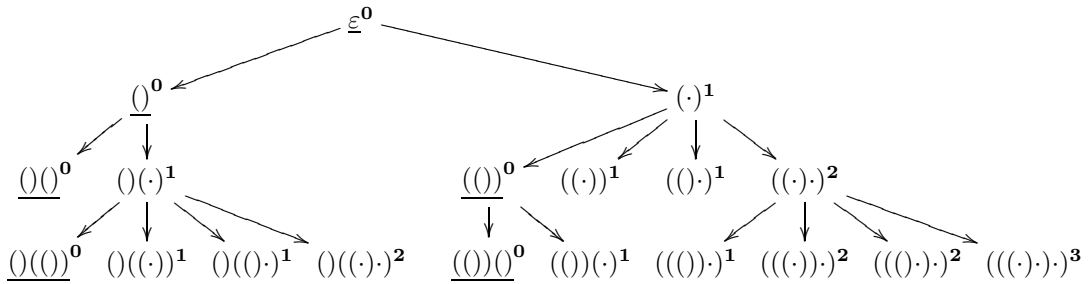


Any (state labelled with a) word w without dots is considered an output state (thus underlined), and has two children; and any word with one or more dots has four children, which arise by replacing the left-most dot in four different ways. Here is a kind of grammar for the ‘growth’ of our automaton, describing for any state what its children look like:

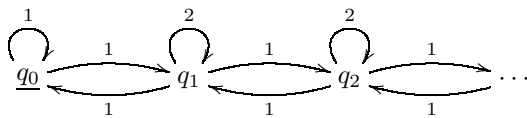


where w and w_1 do not contain any dots.

Identification: States can next be identified according to the number of dots they contain:



yielding the following automaton:



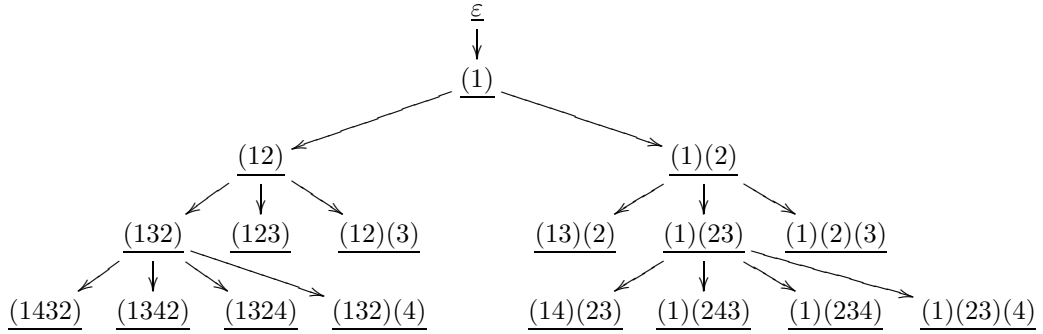
Expression: We find the following expressions for the stream of answers $S(q_0)$:

$$S(q_0) = \frac{1}{1 - X - \frac{X^2}{1 - 2X - \frac{X^2}{1 - 2X - \frac{X^2}{\ddots}}}} = \frac{2}{1 + \sqrt{1 - 4X}} = \frac{1}{1 - \frac{X}{1 - \frac{X}{1 - \frac{X}{\ddots}}}}$$

(= (1, 1, 2, 5, 14, ...), the Catalan numbers). The first equality follows from formula (4). The second and third equalities can be obtained similarly to our first problem on well-bracketed words, at the beginning of the present section.

11 Permutations

A permutation (bijection) $p : \{1, \dots, k\} \rightarrow \{1, \dots, k\}$ can be represented by the corresponding sequence of images $p = (p(1) \dots p(k))$. Another very common and equivalent representation, which is the one that we shall be using, describes a permutation by the (unique) sequence of cycles of which the permutation is composed. For instance, the permutation $p = (532461)$, with $p(1) = 5$, $p(2) = 3$, $p(3) = 2$, $p(4) = 4$, $p(5) = 6$, and $p(6) = 1$, can also be represented by the following sequence: $p = (156)(23)(4)$. Here a cycle $c = (x_1 \dots x_k)$ denotes a permutation of $\{x_1, \dots, x_k\}$ defined by: $c(x_i) = x_{i+1}$, for all $1 \leq i \leq k-1$, and $c(x_k) = x_1$. We start with a trivial question, for any $k \geq 0$: what is the number s_k of permutations of the set $\{1, \dots, k\}$ (defining $s_0 = 1$)? The following automaton enumerates all permutations by listing all possible sequences of cycles:



Any state at level k represents a permutation of the set $\{1, \dots, k\}$ and is therefore an output state. It can make a transition to a state at the next level, either by adding the number $k+1$ to one of the existing cycles or by adding the new cycle $(k+1)$. There are (for all states at level k) precisely k transitions of the first, and one transition of the second type, $k+1$ transitions in total. This explains the structure of the automaton above, and at the same time indicates that all states of every single level can be identified, yielding the following automaton:

$$\underline{q_0} \xrightarrow{1} \underline{q_1} \xrightarrow{2} \underline{q_2} \xrightarrow{3} \dots$$

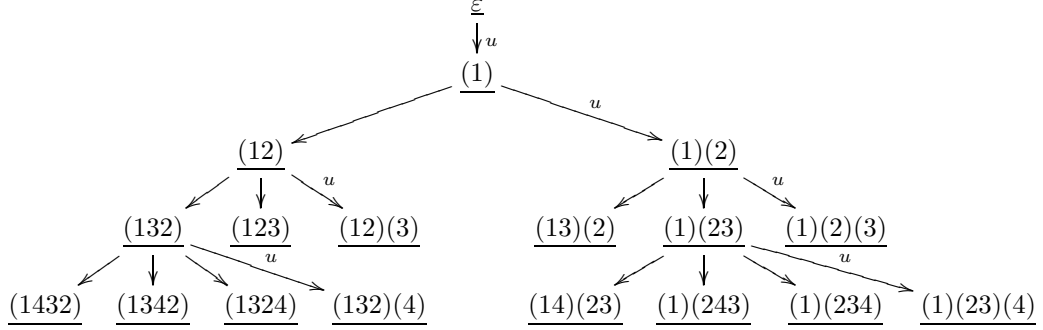
Applying formula (5), with $l_k = 0$ and $u_k = k+1$ for all $k \geq 0$, we obtain

$$S(q_0) = 1 + 1!X + 2!X^2 + 3!X^3 + \dots$$

for the stream (s_0, s_1, s_2, \dots) of counts we are after. In other words, there are $k!$ different permutations of the set $\{1, \dots, k\}$, which comes as no surprise.

We have chosen to represent permutations by sequences of cycles in the automaton above, because it can be easily adapted to deal with various related counting problems. A first and straightforward variation is to keep track of the total number of cycles in each permutation. This we can do by fixing any real number (variable) u , which we use as a label for all transitions that

represent the addition of a new cycle (recall that all other transitions have label 1, which is as usual omitted):



Identifying again all states of the same level gives the following equivalent automaton:

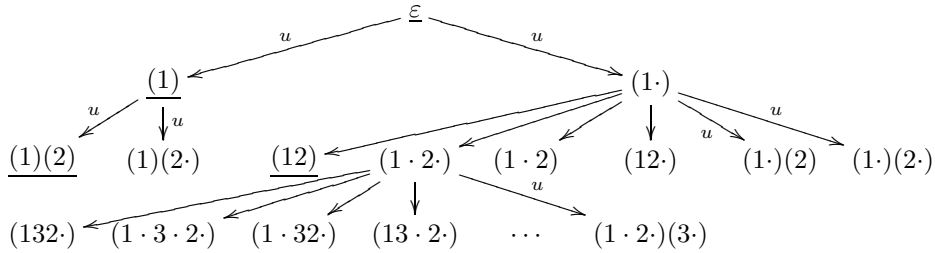
$$\underline{q_0} \xrightarrow{u} \underline{q_1} \xrightarrow{u+1} \underline{q_2} \xrightarrow{u+2} \dots$$

to which, as before, formula (5) can be applied, now yielding

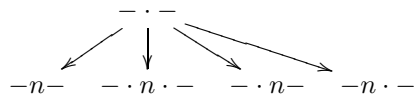
$$\begin{aligned} S(q_0) &= 1 + u \times X + u(u+1) \times X^2 + u(u+1)(u+2) \times X^3 + \dots \\ &= (1 - X)^{-u} \end{aligned}$$

where the latter equality can be proved using some elementary stream calculus. (Note that this formula generalises the solution of our previous counting, which is obtained by taking $u = 1$.) The above stream can be considered as having u as a parameter. It encodes all numbers $s_{k,n}$, for $k, n \geq 0$, counting all permutations of $\{1, \dots, k\}$ consisting of n cycles. These numbers are known as the Stirling numbers of the first kind, and can be computed from the stream above as the (Taylor) coefficients of the powers $u^n \times X^k$. (One can also treat both X and u as formal variables; this would bring us to multivariate stream calculus, which is omitted here.)

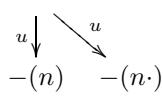
Next we present yet another way of counting permutations while keeping track of the number of cycles, yielding another representation of the Stirling numbers of the first kind. More specifically, permutations can, alternatively, be enumerated as follows:



The general pattern in which this tree-like weighted automaton is grown, is as follows. In any state (node) at level $n - 1$, all of the numbers $\{1, \dots, n - 1\}$ occur, possibly together with a number of dots. All states without dots are output states, representing completed permutations. States with dots represent permutations that have not yet been completed, and the dots indicate the places where further growth is possible. More specifically, for each dot, there are four successor states:

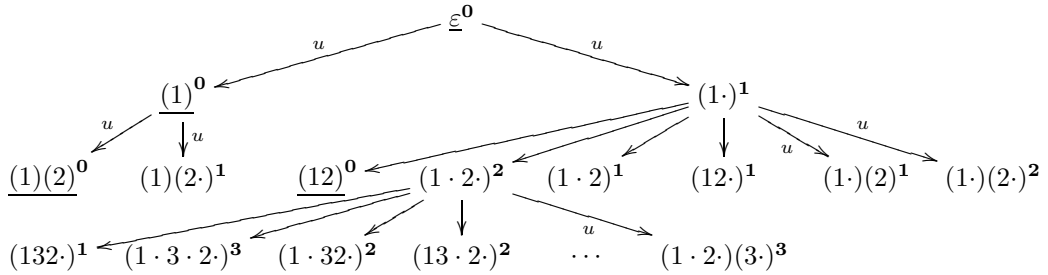


in which the dot has been replaced by n , $\cdot n$, $\cdot n$, and $n \cdot$, respectively. In addition, any state at level $n - 1$ has two more successors, corresponding to the opening of a new cycle:

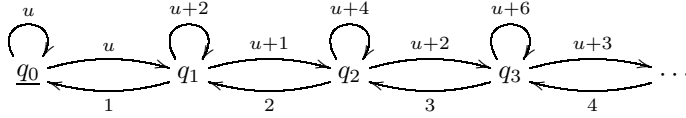


As before, the transitions in which a new cycle is opened, are labelled by u (instead of 1).

Similarly to the second counting problem about well-bracketed words in Section 10, states can be identified according to the number of dots they contain, as is indicated by the superscripts below:



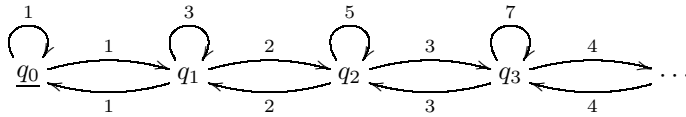
Identification of all equivalent states yields the following well-structured automaton:



to which formula (4) can be applied, resulting in the following continued fraction:

$$\begin{aligned} S(q_0) &= \frac{1}{1 - uX - \frac{1uX^2}{1 - (u+2)X - \frac{2(u+1)X^2}{1 - (u+4)X - \frac{3(u+2)X^2}{\ddots}}}} \\ &= (1 - X)^{-u} \end{aligned}$$

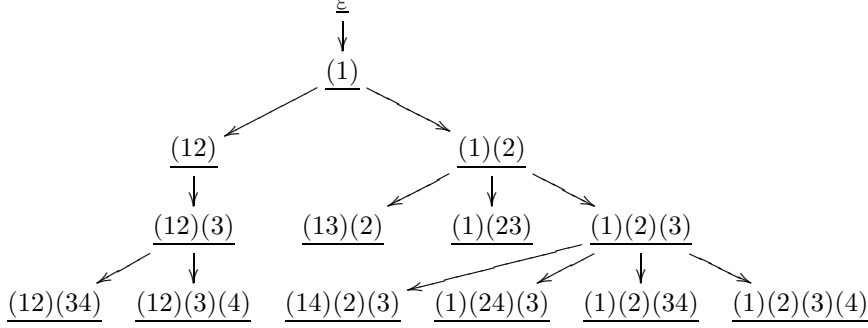
The second equality is obtained by combining our first and our second way of counting permutations and cycles. Thus we have obtained two different expressions for the Stirling numbers of the first kind. Forgetting about the number of cycles again, that is, taking $u = 1$, we obtain



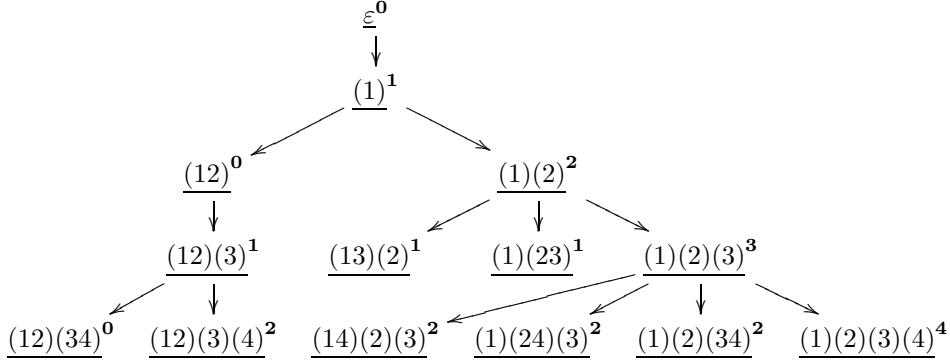
which leads to the following equalities:

$$\begin{aligned} S(q_0) &= \frac{1}{1 - X - \frac{1^2X^2}{1 - 3X - \frac{2^2X^2}{1 - 5X - \frac{3^2X^2}{\ddots}}}} \\ &= (1 - X)^{-1} \\ &= (0!, 1!, 2!, 3!, \dots) \end{aligned}$$

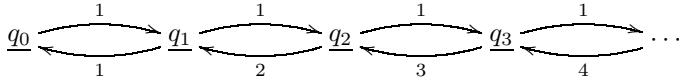
What is, moving to a next question, for any $k \geq 0$ the number s_k of permutations p of $\{1, \dots, k\}$ such that $p \circ p = 1$ (the so-called involutions)? For this we return to the first automaton of this section, from which we now remove all states but the ones consisting of 1- and 2-cycles only:



States can be identified according to the number of 1-cycles, which can still become 2-cycles, they contain, as indicated by the superscripts below:

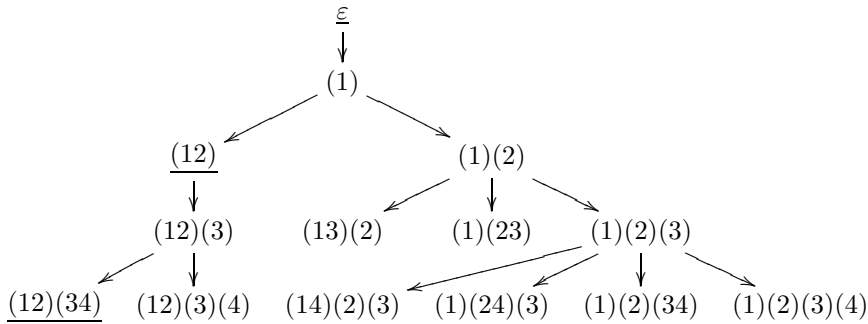


yielding the following automaton



Applying formula (9.3) yields an expression for $S(q_0)$, but not a very pretty one (which is therefore omitted).

For the special case of permutations consisting of 2-cycles only, the same enumeration as for the involutions can be used, the only difference being that states that contain 1-cycles are no longer output states:



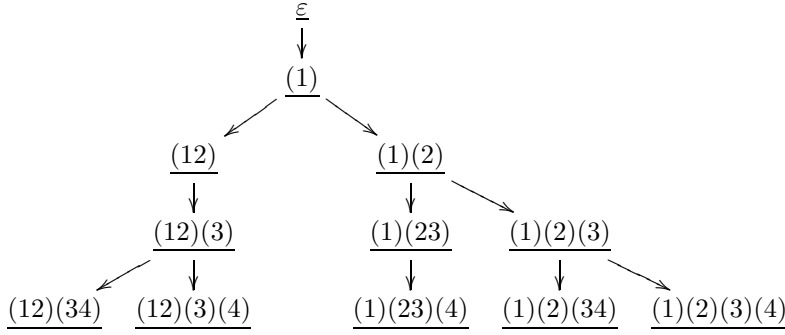
The corresponding reduced automaton now looks like



with, according to formula (4), a pleasant expression for the stream of answers:

$$S(q_0) = \frac{1}{1 - \frac{1 \cdot X^2}{1 - \frac{2 \cdot X^2}{1 - \frac{3 \cdot X^2}{\ddots}}}}$$

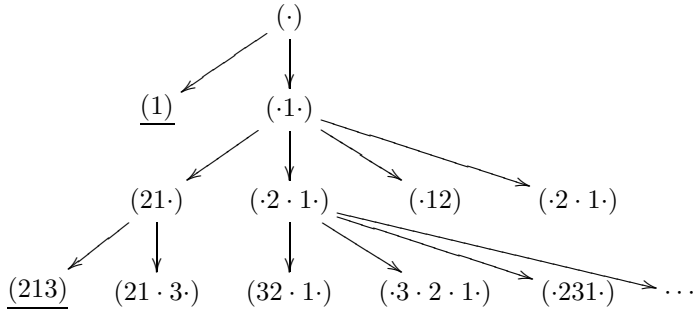
Yet another question: what is, for any $k \geq 0$, the number s_k of permutations $p : \{1, \dots, k\} \rightarrow \{1, \dots, k\}$ such that $|p(i) - i| \leq 1$, for all $1 \leq i \leq k$? These are precisely all permutations consisting of 1-cycles, and of 2-cycles (xy) with $y = x + 1$. The following automaton, which happens to be a further pruning of the last tree-like automaton above, lists them all:



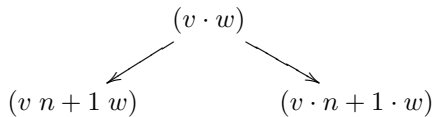
(Note that now all states are output states again.) All states ending with a 1-cycle and all states ending with a 2-cycle can be identified, respectively, leading to the following automaton and corresponding expression for the stream of answers:

$$\begin{array}{c} q_0 \xrightarrow{\quad} q_1 \xrightarrow{\quad} q_0 \\ \text{(loop)} \end{array} \quad S(q_0) = \frac{1}{1 - X - X^2}$$

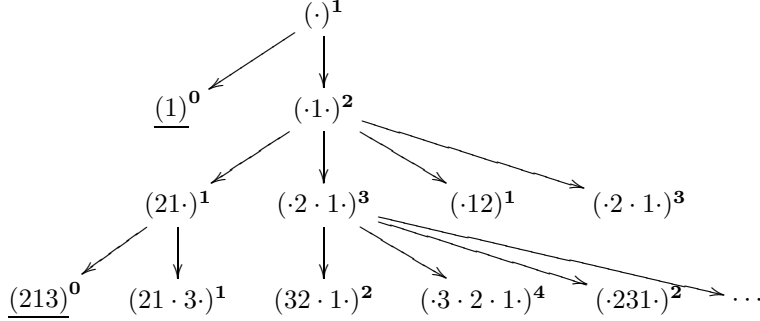
For the last example of this section, we return to the representation of permutations mentioned at the beginning of this section. That is, a permutation (bijection) $p : \{1, \dots, k\} \rightarrow \{1, \dots, k\}$ will be represented below by the corresponding sequence of images $p = (p(1) \cdots p(k))$. Referring to this representation, a permutation is called *alternating* whenever either $p(0) > p(1) < p(2) > \cdots > p(n)$ or $p(0) < p(1) > p(2) < \cdots < p(n)$. What is, we ask ourselves (because we already know that the question will have a pretty answer), for any *odd* natural number $k \geq 0$, the total number s_k of alternating permutations of the set $\{1, \dots, k\}$ (putting $s_k = 0$ for all even k)? Here is an enumeration of all such odd alternating permutations:



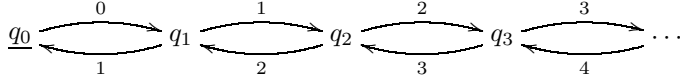
The general pattern according to which this tree is grown is as follows. As before, dots indicate places where further growth is possible. And any dot at level n has two children:



in which the dot has been replaced by $n + 1$ or by $n + 1$ with dots on both sides. Also as before, states can be identified according to the number of dots they contain:



After identification of all equivalent states, the following automaton is obtained:



Note that in this automaton, the stream of answers is represented by q_1 and not by q_0 . Using (a minor variation of) formula (4), we get

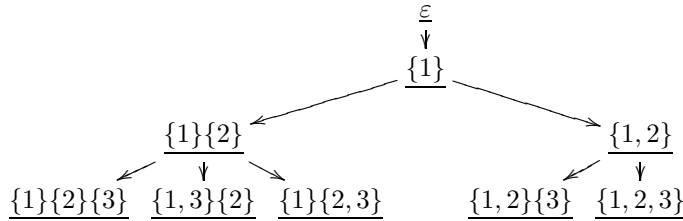
$$S(q_1) = \frac{X}{1 - \frac{1 \cdot 2 \cdot X^2}{1 - \frac{2 \cdot 3 \cdot X^2}{1 - \frac{3 \cdot 4 \cdot X^2}{\ddots}}}} = \text{Taylor}(\tan(x))$$

where the latter equality follows from the fact that the reduced automaton above is precisely the same as the automaton that we found at the end of Section 4 by splitting the derivatives of the stream τ of the Taylor coefficients of the function $\tan(x)$.

12 Set partitions

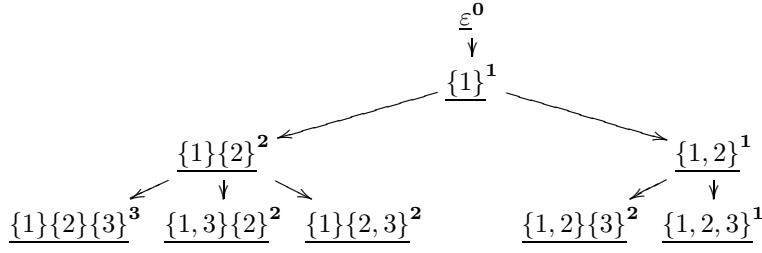
Our last series of counting problems deals with set partitions, counted in various ways.

What is, for any $k \geq 0$, the number s_k of ways in which the set $\{1, \dots, k\}$ can be partitioned into subsets (the so-called Bell numbers)? The following automaton enumerates at level k all such partitions:

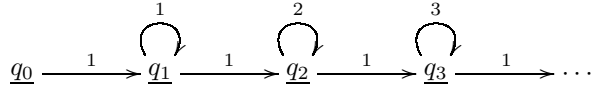


States can be identified according to the number of subsets used in the corresponding partition,

as indicated by the superscripts below:



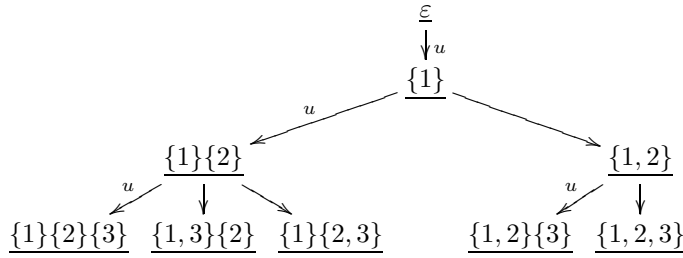
Identifying as usual all equivalent states gives the following reduced automaton:



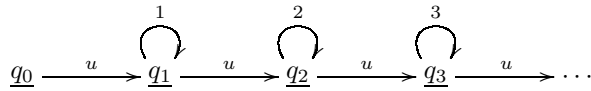
And by formula (5), the following expression is obtained for our stream of answers:

$$S(q_0) = 1 + \frac{X}{1-X} + \frac{X^2}{(1-X)(1-2X)} + \frac{X^3}{(1-X)(1-2X)(1-3X)} + \dots$$

It is easy to keep track explicitly of the number of subsets used in any partition, by labelling all transitions in which a new subset is added to the partition by a fixed real number (variable) u :



Identification can be done as before, yielding the following reduced automaton



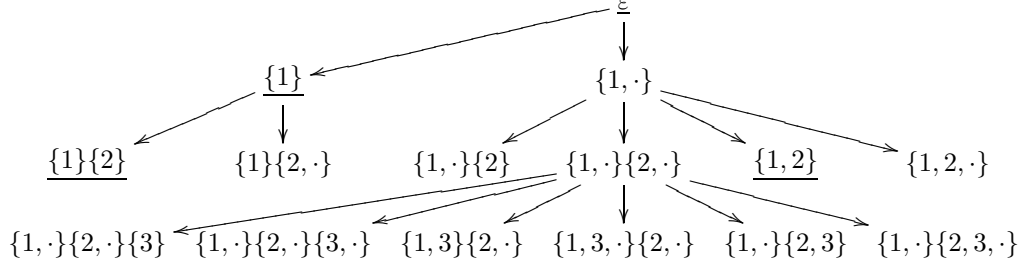
and corresponding expression for the stream of answers:

$$S(q_0) = 1 + \frac{uX}{1-X} + \frac{u^2X^2}{(1-X)(1-2X)} + \frac{u^3X^3}{(1-X)(1-2X)(1-3X)} + \dots$$

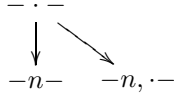
This stream encodes, for all $k, n \geq 0$, all numbers $s_{k,n}$ of partitioning the set $\{1, \dots, k\}$ into n subsets. These numbers are called the Stirling numbers of the second kind. Note that by taking $u = 1$, the previous case is found back.

We have seen various examples where a different way of enumeration, using a different representation of the structures to be counted, leads to a different but equivalent expression for the solution of the counting problem. This phenomenon is here further illustrated by the following, alternative solution for the problem of counting set partitions. For any $k \geq 0$, the following automaton contains at level k all partitions of the set $\{1, \dots, k\}$ as output states; in addition, each

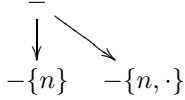
level contains various intermediate states containing dots that indicate possibilities for further growth:



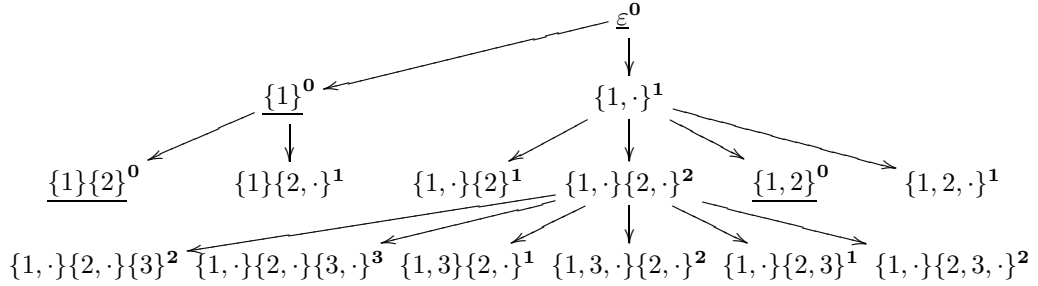
The general pattern is as follows: For any dot occurring in a node at level $n - 1$, there are two children, obtained by replacing the dot by either n or by n and a dot (allowing still further growth):



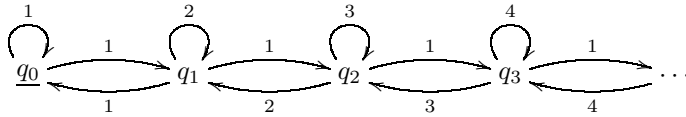
In addition, any node at level $n - 1$ has two more children, obtained by opening a new subset containing n , again with or without the presence of a dot:



As in all the previous examples of representations involving dots, states can be identified according to the number of dots they contain:



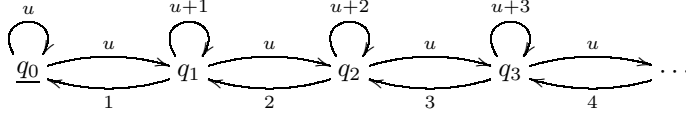
yielding the following reduced automaton



and, by an application of formula (4), the following corresponding expression for the stream of answers:

$$\begin{aligned}
 S(q_0) &= \frac{1}{1 - X - \frac{X^2}{1 - 2X - \frac{2X^2}{1 - 3X - \frac{3X^2}{\ddots}}}} \\
 &= 1 + \frac{X}{1 - X} + \frac{X^2}{(1 - X)(1 - 2X)} + \frac{X^3}{(1 - X)(1 - 2X)(1 - 3X)} + \cdots
 \end{aligned}$$

where the latter equality is obtained by identifying the first and the second solution of this problem. Again, it is possible to keep track of the number of subsets used in each partition. Omitting the enumerating automaton, we only present the resulting reduced automaton:



and the corresponding expression for the stream of answers:

$$S(q_0) = \frac{1}{1 - (u \times X) - \frac{uX^2}{1 - ((u+1) \times X) - \frac{2uX^2}{1 - ((u+2) \times X) - \frac{3uX^2}{\ddots}}}}$$

13 Special numbers

The theorem below summarizes the representations by means of a reduced weighted automaton, of the various sequences of so-called special numbers that we have encountered in the preceding sections. It clearly shows that these representations have a great structural similarity.

Theorem 13.1 *In the table in Figure 1, the stream of numbers mentioned on the right is represented by the state q_0 of the automaton on the left, except for the stream of the tangent numbers, which is represented by the state q_1 .*

14 Discussion

For each and every single counting problem that we have dealt with in this paper, there exist in the literature far more detailed and complete treatments (see the references mentioned in the introduction). The contribution of the method of coinductive counting lies in the fact that it solves all these different counting problems, in the very same way: (1) The structures to be counted are always enumerated by infinite tree-shaped weighted automata. (2) These automata are reduced by means of bisimulations. (3) From the resulting reduced automata, an expression for the stream of answers is derived by means of elementary stream calculus. Traditionally, the situation with respect to these points is as follows: (1) Many different mathematical structures are used as representations of combinatorial objects, including weighted graphs, sets of walks, various kinds of trees, formal languages (regular and context-free), automata, (transfer-) matrices, and so on. (2) We know of no *systematic* way, in the sense of being based on one specific notion, of reducing such representations to smaller or better structured ones. (3) Stream calculus [Rut00a, Rut01] can be viewed as a formalisation and generalisation of the use of both generating functions and formal power series (cf. [Rut01] for a detailed comparison).

Technically, the novelty of the coinductive counting method is based on (i) the systematic use of *infinite* weighted automata, which have received in the literature on automata theory so far no attention; (ii) the use of the *quantitative* notion of *stream bisimulation*; and (iii) the consequent use of a *coinductive* calculus of streams in the search for a closed expression for the outcome.

The presently proposed method of coinductive counting is far more restricted in scope than the references mentioned in the introduction. Notably, no (classical) analytical methods are being used here. Such methods might be used, though, at the point where our methods stops: many of the obtained stream expressions are suited for further analytical treatment. The use of continued fractions has been inspired by [Fla80]; see also [GJ83, Chapter 5.2]. The formal treatment of such continued fractions seems somewhat easier in the present setting of coinductive stream calculus. Also, some of the combinatorial interpretations of the various fractions discussed here, seem to be

<i>automaton</i>	<i>represents</i>
	<i>Fibonacci numbers</i>
	<i>Catalan numbers</i>
	<i>involutions</i>
	<i>tangent numbers</i>
	<i>Stirling numbers (1st)</i>
	<i>factorial numbers (u = 1)</i>
	<i>Stirling numbers (1st)</i>
	<i>factorial numbers (u = 1)</i>
	<i>Stirling numbers (2nd)</i>
	<i>Bell numbers (u = 1)</i>
	<i>Stirling numbers (2nd)</i>
	<i>Bell numbers (u = 1)</i>

Figure 1: Representations of special numbers

simpler and more uniform. A basic reference on weighted automata is [BR88]. Most of the Sections 2 through 4 can be fairly straightforwardly generalised from streams to so-called multivariate streams or, more generally, to formal power series in many non-commutative variables and with coefficients in an arbitrary semiring. (One of the obvious things to be done is replacing inverse and shuffle inverse by star and shuffle star.) Much of this has been worked out in [Rut99, Rut00a].

We mention very briefly a few points for further research. (i) The strength of the presented method of counting through enumeration and minimization is its generality and its simplicity. Its weakness is the sometimes more and sometimes less ad-hoc character of the way the counted structures are enumerated. This raises the question whether there exists a more systematic way of enumeration, possibly in terms of some kind of grammars for tree-growing. Such grammars will no doubt be closely related to an alternative approach to counting, which is based on structural properties expressed as a kind of domain equation (such as $T = 1 + T \times X \times T$ for binary trees), as present in for instance [FS93] and also [BLL98]. Also the work of the Florence school of Pinzani and others [BLPP99] is related to this point. (ii) The issue of minimization of weighted automata has of course only been touched upon. In the presented examples, there usually was an obvious minimized candidate, but a more systematic and algorithmic analysis would be welcome. (iii) We have dealt with the univariate case (in X) only. It is worthwhile to develop also the multivariate case in some detail. (iv) In one case (Section 8), we have distinguished between rational and algebraic streams. Notably in the work of Flajolet, such as [FS01], much more has already been said about the classification of streams (there rather: generating functions) in analytical terms. Looking at the various weighted automata that we have encountered so far, we distinguish at least three types: Finite automata correspond to rational streams. Infinite automata that are ‘regular’ in the sense of having only finitely many states that locally have a different transition behaviour; these correspond to algebraic streams (see the examples on well-bracketed words). And infinite automata that are not regular in the afore-mentioned sense, but might still be considered regular according to some other criterion (for instance, involving nicely increasing sequences of labels, such as in most of our examples). The latter type of automata and the streams they represent seem to deserve further study. (v) In some of the counting exercises not reported on here, the use of what we would like to call ‘heavy-weighted automata’ turned out to be extremely practical. What we have in mind are automata that have transitions labelled by complete stream expressions (such as X^3 or $X + X^2$) rather than by real numbers (and X) only.

Acknowledgements

Many thanks are due to Manfred Droste, Heiko Vogler, and the anonymous referee for corrections and suggestions.

References

- [BLL98] F. Bergeron, G. Labelle, and P. Leroux. *Combinatorial species and tree-like structures*, volume 67 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, 1998.
- [BLPP99] E. Barcucci, A. Lungu, E. Pergola, and R. Pinzani. ECO: a methodology for the enumeration of combinatorial objects. *Journal of Difference Equations and Applications*, 5:435–490, 1999.
- [BR88] J. Berstel and C. Reutenauer. *Rational series and their languages*, volume 12 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, 1988.
- [Fla80] P. Flajolet. Combinatorial aspects of continued fractions. *Discrete Mathematics*, 32:125–161, 1980.
- [FS93] P. Flajolet and R. Sedgewick. The average case analysis of algorithms: Counting and generating functions. Research Report 1888, INRIA Rocquencourt, 1993. 116 pages.

- [FS01] P. Flajolet and R. Sedgewick. Analytical combinatorics: Functional equations, rational and algebraic functions. Research Report 4103, INRIA Rocquencourt, 2001. 98 pages.
- [GJ83] I.P. Goulden and D.M. Jackson. *Combinatorial enumeration*. John Wiley and Sons, 1983.
- [GKP94] R.L. Graham, D.E. Knuth, and O. Patashnik. *Concrete mathematics, second edition*. Addison-Wesley, 1994.
- [JR97] Bart Jacobs and Jan Rutten. A tutorial on (co)algebras and (co)induction. *Bulletin of the EATCS*, 62:222–259, 1997. Available at URL: www.cwi.nl/~janr.
- [PE98] D. Pavlović and M. Escardó. Calculus in coinductive form. In *Proceedings of the 13th Annual IEEE Symposium on Logic in Computer Science*, pages 408–417. IEEE Computer Society Press, 1998.
- [Rut99] J.J.M.M. Rutten. Automata, power series, and coinduction: taking input derivatives seriously (extended abstract). Report SEN-R9901, CWI, 1999. Available at URL: www.cwi.nl. Also in the proceedings of ICALP '99, LNCS 1644, 1999, pp. 645–654.
- [Rut00a] J.J.M.M. Rutten. Behavioural differential equations: a coinductive calculus of streams, automata, and power series. Report SEN-R0023, CWI, 2000. Available at URL: www.cwi.nl. To appear in Theoretical Computer Science.
- [Rut00b] J.J.M.M. Rutten. Universal coalgebra: a theory of systems. *Theoretical Computer Science*, 249(1):3–80, 2000.
- [Rut01] J.J.M.M. Rutten. Elements of stream calculus (an extensive exercise in coinduction). In Stephen Brooks and Michael Mislove, editors, *Proceedings of MFPS 2001: Seventeenth Conference on the Mathematical Foundations of Programming Semantics*, volume 45 of *Electronic Notes in Theoretical Computer Science*, pages 1–66. Elsevier Science Publishers, 2001.
- [Sta97] R.P. Stanley. *Enumerative Combinatorics I*, volume 49 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, 1997.
- [Sta99] R.P. Stanley. *Enumerative Combinatorics II*, volume 62 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, 1999.