Electronic Notes in Theoretical Computer Science 65 No. 1 (2002)
URL: http://www.elsevier.nl/locate/entcs/volume65.html 19 pages

Coinductive Counting: Bisimulation in Enumerative Combinatorics (Extended Abstract)

J.J.M.M. Rutten

CWI and Free University, Amsterdam
Jan.Rutten@cwi.nl

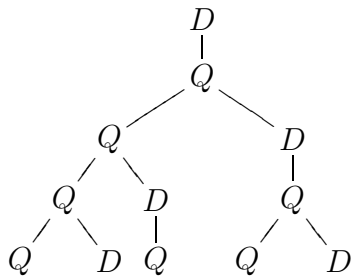
Abstract

The recently developed coinductive calculus of streams finds here a further application in enumerative combinatorics. A general methodology is developed to solve 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 minimized by means of the quantitative notion of stream bisimulation; (3) the minimized automaton is used to compute an expression (in terms of stream constants and operators) that represents the stream of all counts.

1 Motivation

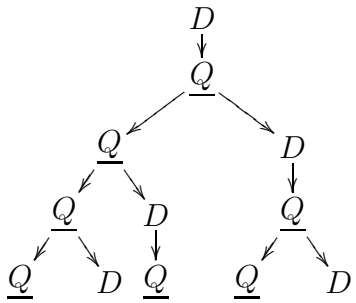
We first illustrate the method of coinductive counting by means of an example, then become more formal in Section 2, after which we continue with many more examples.

The following counting problem is taken from [6, 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:

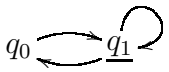


Each drone has one mother, one grandmother, two great-grandmothers, three great-great-grandmothers, and so on. What is, for any $k \geq 0$, the number

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* $\sigma = (s_0, s_1, s_2, \ldots)$ 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:



The *stream behaviour* of such automata will be defined coinductively, and can be expressed in terms of transition sequences. In the present automaton, the numbers s_k are 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* [7]. 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 equivalent to the state q_0 and every queen state is bisimilar 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

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

This expression encodes, so to speak, the numbers s_k for all $k \geq 0$, each of

which can be retrieved as the initial value of the k-th *stream derivative*:

$$s_k = (\frac{X}{1 - X - X^2})^{(k)}(0)$$

(which turns out to be the k-th Fibonacci number). In the present paper, however, 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:

(i) *Enumerate* the objects to be counted in an infinite, tree-shaped automaton.

(ii) *Identify* states that represent identical streams, using bisimulation.

(iii) *Express* 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* (real numbers), which are taken into account by the notion of stream bisimulation (and bisimulation-up-to).

The method of coinductive counting is both very simple and surprisingly general, as will be illustrated by many further examples. At the same time, it has, so far, not led to new solutions of counting problems that had not been solved before. We feel that the present coinductive perspective on counting has, nevertheless, 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.

About all our example counting problems stem from one of the following texts, from which we have learned most of what we know about enumerative combinatorics: [6], [4,5], and [9,10]. The use of continued fractions has been inspired by [3]. A basic reference on weighted automata is [2]. The present paper is an extended abstract of a full paper in preparation [8].

2 Basic facts from stream calculus

We briefly summarize those parts of [7] that are needed in the present paper. The set of all streams is defined by $\mathbb{R}^\omega = \{\sigma \mid \sigma : \{0,1,2,\ldots\} \to \mathbb{R}\}$. Individual streams will be denoted by $\sigma = (\sigma(0), \sigma(1), \sigma(2), \ldots) = (s_0, s_1, s_2, \ldots)$ We shall call $\sigma(0)$ the *initial value* of σ. The *derivative* of a stream σ is defined by $\sigma' = (s_1, s_2, s_3, \ldots)$. A *bisimulation* on $\mathbb{R}^\omega$ is a relation $R \subseteq \mathbb{R}^\omega \times \mathbb{R}^\omega$ such that, for all σ and τ in $\mathbb{R}^\omega$: if $\sigma\ R\ \tau$ then $\sigma(0) = \tau(0)$ and $\sigma'\ R\ \tau'$. Bisimilarity, which is the union of all bisimulation relations, is denoted by $\sim$. We have the usual principle of coinduction: for all $\sigma, \tau \in \mathbb{R}^\omega$, if $\sigma \sim \tau$ then $\sigma = \tau$. 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 $(r, 0, 0, 0, \ldots)$ again by r. Another constant stream is $X = (0, 1, 0, 0, 0, \ldots)$, 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
$(\sigma \times \tau)' = (\sigma' \times \tau) + (\sigma(0) \times \tau')$	$(\sigma \times \tau)(0) = \sigma(0) \times \tau(0)$	product
$(\sigma^{-1})' = -\sigma(0)^{-1} \times (\sigma' \times \sigma^{-1})$	$(\sigma^{-1})(0) = \sigma(0)^{-1}$	inverse
$(\sqrt{\sigma})' = \sigma' \times (\sqrt{\sigma(0)} + \sqrt{\sigma})^{-1}$	$\sqrt{\sigma}(0) = \sqrt{\sigma(0)}$	square root
$(\sigma \otimes \tau)' = (\sigma' \otimes \tau) + (\sigma \otimes \tau')$	$(\sigma \otimes \tau)(0) = \sigma(0) \times \tau(0)$	shuffle product
$(\sigma^{-\underline{1}})' = -\sigma' \otimes (\sigma^{-\underline{1}} \otimes \sigma^{-\underline{1}})$	$\sigma^{-\underline{1}}(0) = \sigma(0)^{-1}$	shuffle inverse
$(\sum_{i \in I} \sigma_i)' = \sum_{i \in I} (\sigma_i)'$	$(\sum_{i \in I} \sigma_i)(0) = \sum_{i \in I} \sigma_i(0)$	sum

The unique existence of a solution to the above equations is discussed in [7] and is ultimately due to the fact that the combination of the operations of initial value and stream derivative constitutes a *final coalgebra* structure on $\mathbb{R}^\omega$. We shall freely use the identities on these operators proved in [7], many of which are familiar anyway (such as $\sigma \times \tau = \tau \times \sigma$ and $\sqrt{\sigma} \times \sqrt{\sigma} = \sigma$). Simple differential equations can often be solved in an algebraic manner, using the

so-called 'Fundamental Theorem': for all streams $\sigma \in \mathbb{R}^\omega$,

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

Consider for instance $\sigma \in \mathbb{R}^\omega$ such that $\sigma(0) = 1$ and $\sigma' = 2 \times \sigma$. This implies $\sigma = \sigma(0) + (X \times \sigma') = 1 + (X \times 2 \times \sigma)$. It follows that $(1 - (2 \times X)) \times \sigma = 1$, whence

$$\sigma = \frac{1}{1 - (2 \times X)} = 2^0 + 2^1 X + 2^2 X^2 + \cdots = (2^0, 2^1, 2^2, \ldots)$$

(Note that in stream calculus, all of these are formal identities as opposed to, for instance, the use of generating functions as a mere 'representation' convention.) 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 $\alpha_0, \ldots, \alpha_n$ and $\beta_0, \ldots, \beta_n$, such that $\sigma' = \alpha_0 + \cdots + \alpha_n$ and $\tau' = \beta_0 + \cdots + \beta_n$ and, for all $0 \leq i \leq n$: either $\alpha_i = \beta_i$ or $\alpha_i \; R \; \beta_i$. There is 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.

A stream can be 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 $o : Q \to \mathbb{R}$, and a transition function $t : Q \to (Q \to_f \mathbb{R})$ (the latter set contains functions $\phi : Q \to \mathbb{R}$ 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 $t(q) : Q \to \mathbb{R}$, 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 $t(q)(q') = r$. 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 $k \geq 0$:

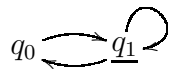
(2) $S(q)(k) = \sum \{l_0 l_1 \cdots l_{k-1} l \mid q = q_0 \xrightarrow{l_0} q_1 \xrightarrow{l_1} \cdots \xrightarrow{l_{k-1}} q_k \text{ and } o(q_k) = l\}$

It expresses the stream $S(q)$ in terms of the labels of transition sequences starting in q, and gives a clear operational intuition. At the same time, it does not yield any compact representations for $S(q)$ and is thereby not very suited for actual reasoning. Fortunately, it is equivalent to the following (coinductive) definition in terms of a system of behavioural differential equations (one for each state in Q). Let $\{q_1, \ldots, q_n\}$ be the set of states q' for which $t(q)(q') \neq 0$:

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

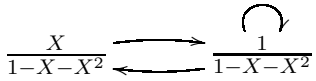
Using (1) above, such systems can often be easily solved. Before considering a small example, we introduce some further notational conventions: for

$t(q)(q') = 1$ we shall often simply write $q \to q'$. We write $\underline{q}$ for $o(q) = 1$, and call q an *output* state. And we include in pictures only non-zero output values and only arrows with a non-zero label. Now consider the automaton



which occurred in the counting example of Section 1. We have $S(q_0)(0) = 0$, $S(q_1)(0) = 1$, $S(q_0)' = S(q_1)$, and $S(q_1)' = S(q_0) + S(q_1)$. Using (1), one finds $S(q_0) = X/1 - X - X^2$.

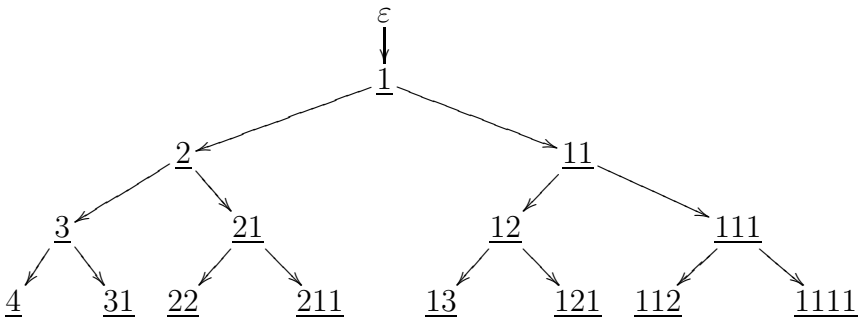
In the above, we have explained how to go from automata to streams. The converse is also possible, by a procedure that we have called 'splitting of derivatives'. Computing the respective derivatives of, for instance, the stream $X/1 - X - X^2$ that we have just obtained, we find $(X/1 - X - X^2)' = 1/1 - X - X^2$, 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:



Note that the fact that the derivative of $1/1 - X - X^2$ consists of a sum, gives rise to 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. And so from $X/1 - X - X^2$, which was the behaviour of the state q_0 in the automaton above, we find back the same automaton (up to a renaming of the states).

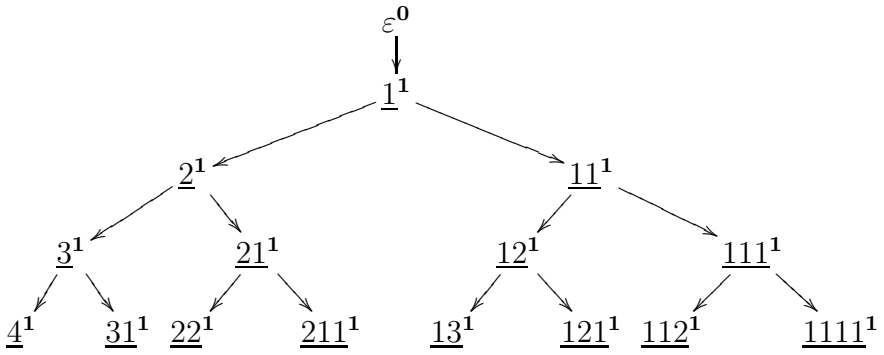
3 Compositions of natural numbers

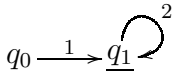
A *composition* of a natural number $k \geq 0$ is a sequence of natural numbers $n_1 \cdots n_l$ such that $k = n_1 + \cdots + n_l$. What is, for any $k \geq 0$, the number s_k of compositions of k? The following automaton enumerates all compositions for all natural numbers (here and in what follows, pictures show only the first few levels of what is understood to be an infinite automaton):



Note that we have 1-transitions only (the labels are omitted) and that all states (but the first) are output states. The k-th level of this automaton contains all compositions of the natural number k. It is an immediate consequence of formula (2), therefore, that the initial state ε represents the stream

$\sigma = (s_0, s_1, s_2, \ldots)$ of answers we are after: $S(\varepsilon) = \sigma$. Next we identify as many states as possible by defining a bisimulation-up-to between (the streams represented by) our weighted automaton, repeated below, and the tiny 2 state automaton that follows:





The superscripts that we have added to the states of our automaton indicate to which state in the little automaton they are related. Or, more explicitly, the above picture suggests the definition of a relation $R \subseteq \mathbb{R}^\omega \times \mathbb{R}^\omega$ as $R = \{\langle S(\varepsilon), S(q_0)\rangle\} \cup \{\langle S(w), S(q_1)\rangle \mid w \in \mathbb{N}^*,\ w \neq \varepsilon\}$. It is easily checked that R is indeed a bisimulation-up-to: all initial values match; $S(\varepsilon)' = S(1)$, which is related to $S(q_1) = S(q_0)'$; and for all words $v \in \mathbb{N}^*$ and natural numbers n, writing vn for the concatenation of v and n, we have $S(vn)' = S(v(n+1)) + S(vn1)$, each component of which is related to $S(q_1)$, thus matching $S(q_1)' = 2 \times S(q_1) = S(q_1) + S(q_1)$. It follows by coinduction-up-to that $\sigma = S(\varepsilon) = S(q_0)$. The latter can be easily computed:

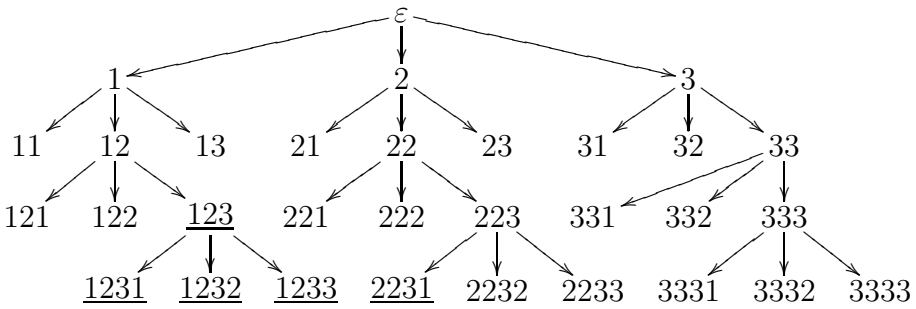
$$S(q_0) = \frac{X}{1 - 2X} \quad (= (0, 2^0, 2^1, 2^2, \ldots))$$

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-*step* from q_1 to itself.

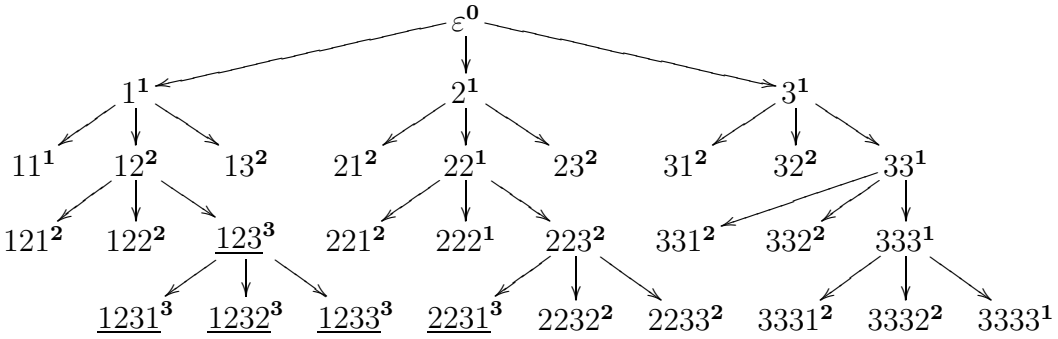
4 Surjections

What is, for any natural number $k \geq 0$, the number s_k of surjections from the set $\{1, \ldots, k\}$ 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 $\{1, \ldots, n\}$, for a fixed but arbitrary $n \geq 1$. Let us denote a function $f : \{1, \ldots, k\} \to \{1, 2, 3\}$ by means of the word $f(1) \cdots f(k)$. The following automaton enumerates at

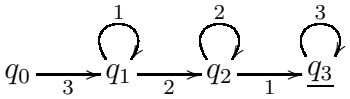
each level k all such functions:



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, but that moreover not all transitions have been included, for lack of space. As before, it follows from (2) that the initial state ε represents the stream $\sigma = (s_0, s_1, s_2, \ldots)$ of answers we are interested in. The automaton can be simplified by identifying all states (labelled with a word) containing an equal number of different symbols, as indicated by the superscripts below:



If one relates (the streams represented by) all i-superscripted states above with the state q_i in the automaton below,



one obtains a bisimulation-up-to, from which $S(\varepsilon) = S(q_0)$ follows by coinduction-up-to. The latter stream can be easily computed, yielding

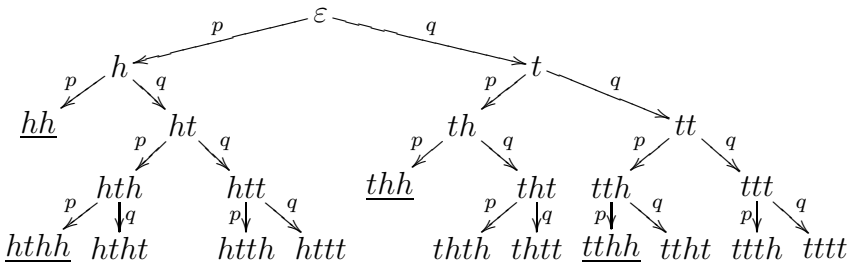
$$\sigma = S(q_0) = \frac{3!X^3}{(1-X)(1-2X)(1-3X)}$$

The formula for surjections onto the set $\{1, \ldots, n\}$, for arbitrary $n \geq 1$, can without much more work be found, too: $n!X^n/(1-X)(1-2X)\cdots(1-nX)$.

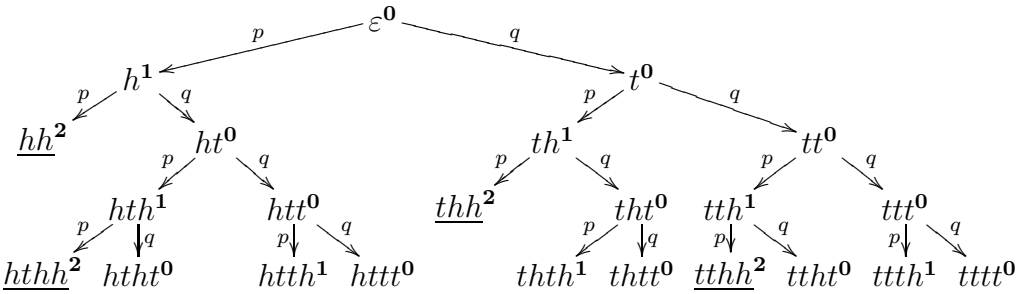
5 Counting with probabilities

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

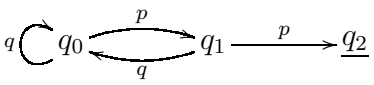
probability s_k of getting, by flipping the coin k times, a sequence of heads and tails (of length k) without the occurrence of two consecutive heads but for the two very last outcomes, which are required to be heads? Here is a weighted automaton describing all possible scenarios:



All states that are (labelled with a sequence) ending in two heads are output states, and have no further transitions. States can be identified according to the number of final heads:



yielding the following automaton and corresponding formula for our stream of answers $\sigma = (s_0, s_1, s_2, \ldots)$:

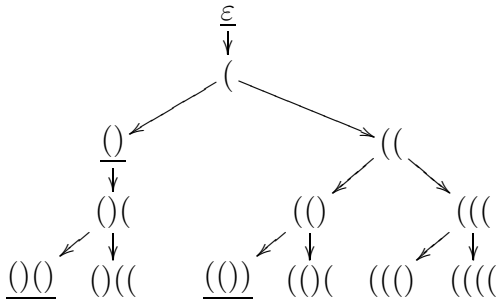


$$\sigma = S(q_0) = \frac{p^2X^2}{1 - qX - pqX^2}$$

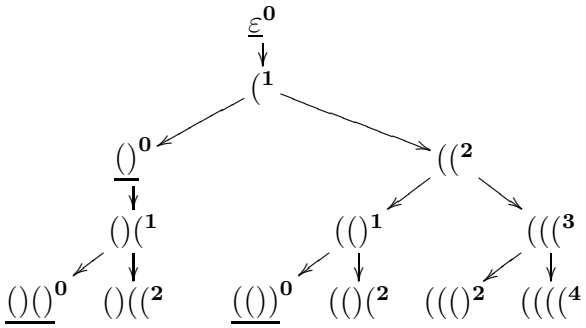
6 Well-bracketed words

So far minimization of our automata has always resulted in a finite automaton yielding, by a well-known general result (cf. [7, Thm. 13.3]), a rational stream. Here is an example for which the stream of counts in not 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? The output states at level k of the following automaton

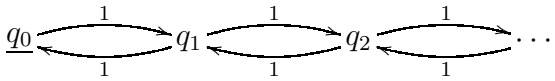
correspond precisely to all such words:



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:



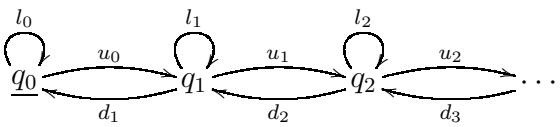
As a consequence of formula (3) in Section 7, which will be dedicated to this type of automata, the following expression for the stream $S(q_0)$ of answers is obtained:

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

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

7 Streams and continued fractions

Let l_i and u_i, for $i \geq 0$, and d_i, for $i \geq 1$, be real numbers that serve as labels, which one might want to pronounce as *level*, *up*, and *down*, of the following automaton:



There is the following continued fraction for the stream represented by the state q_0:

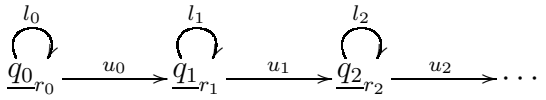
$$S(q_0) = \cfrac{1}{1-(l_0\times X)-\cfrac{(u_0\times X)\times(d_1\times X)}{1-(l_1\times X)-\cfrac{(u_1\times X)\times(d_2\times X)}{\ddots}}} \tag{3}$$

This 'infinite expression' makes perfect sense, being just a shorthand for the solution $S(q_0) = \sigma_0$ of the following infinite system of equations in streams σ_i, for $i \geq 0$:

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

A formal proof of $S(q_0) = \sigma_0$ is easy, by coinduction, see [7, Thm. 17.1]. The formula for $S(q_0)$ can intuitively be understood by combining the general fact that $1/1-\tau = 1+\tau+\tau^2+\cdots \ \ (=\tau^*)$, for any $\tau \in \mathbb{R}^\omega$ with $\tau(0) = 0$, with the observation that the state q_0 has the repeated choice between taking a l_0 step to itself, or a u_0 step to q_1, then anything q_1 can do, that is, σ_1, followed by a d_1 step back to q_0. For the automaton at the end of Section 6, the labels are $l_i = 0$, and $u_i = d_{i+1} = 1$, for all $i \geq 0$. Therefore $\sigma_0 = \sigma_1$, which leads to $\sigma_0 = 1/1-X^2\sigma_0$ or, equivalently, the algebraic equation $X^2\sigma_0^2-\sigma_0+1 = 0$. The solution of this equation is, indeed, $2/1+\sqrt{1-4X^2}$.

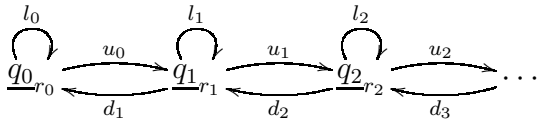
In some of our examples, we shall also encounter the following type of automaton, with no downward transitions but where all states may have a non-trivial output value $r_i \in \mathbb{R}$:



The stream that is represented by q_0 is given by the following 'upward' continued fraction:

$$S(q_0) = \cfrac{r_0+(u_0\times X)\times\cfrac{r_1+(u_1\times X)\times\cfrac{r_2+(u_2\times X)\times\cfrac{\vdots}{1-(l_3\times X)}}{1-(l_2\times X)}}{1-(l_1\times X)}}{1-(l_0\times X)} \tag{4}$$

This type of automaton was not yet dealt with in [7]. Again, the continued fraction represents an infinite system of equations, the solution of which can be shown to be equal to $S(q_0)$. The next type of automaton, finally, generalizes both of the above families of automata:



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

$$\frac{r_0 + (u_0 \times X) \times A}{1 - (l_0 \times X) - (u_0 \times X) \times A \times (d_1 \times X)} \tag{5}$$

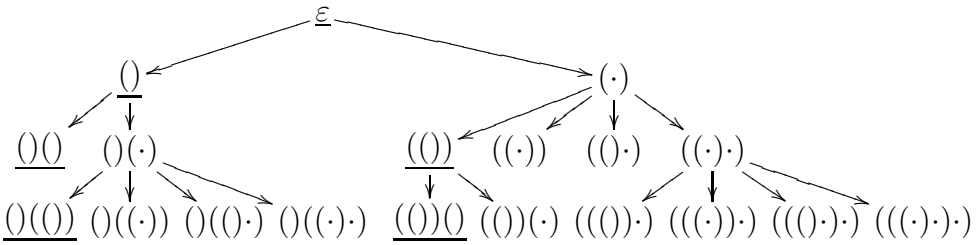
where for notational convenience, A is a shorthand for

$$A = \frac{r_1 + (u_1 \times X) \times (\cdots)}{1 - (l_1 \times X) - (u_1 \times X) \times (\cdots) \times (d_2 \times X)}$$

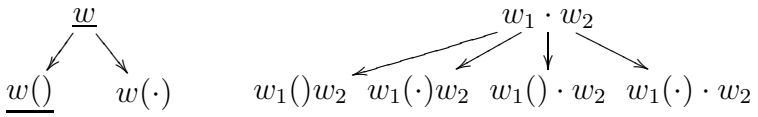
Note that all of the earlier (minimized) automata that we have seen in the present paper sofar, are special instances of this last type of automaton, and that all the corresponding expressions for $S(q_0)$ are (extremely simple) special instances of formula (5) above, such as $S(q_0) = p^2X^2/1 - qX - pqX^2$, at the end of Section 5, to mention one example.

8 More on well-bracketed words

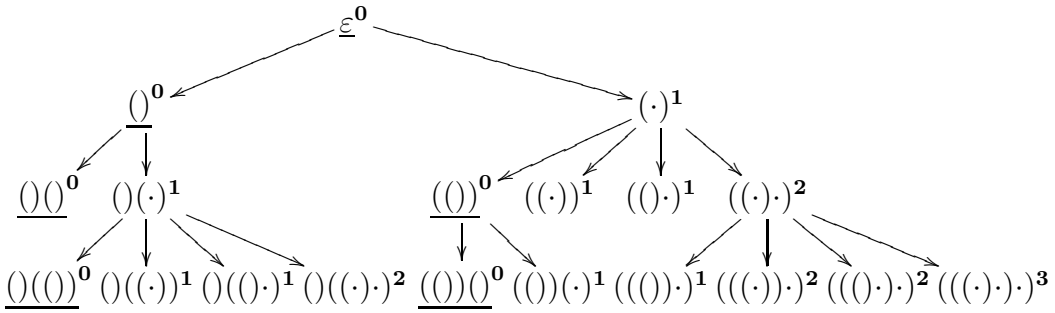
Sometimes there are various ways to enumerate the structures to be counted, by means of different automata, and often this leads to new ways of expressing the stream of counts. As an example, we tackle the counting problem of Section 6 again, in a slightly different form: What is, for any $k \geq 0$, the number s_k of well-bracketed words consisting of k matching *pairs* of an opening and a closing bracket? Here is one way of enumerating all such words:



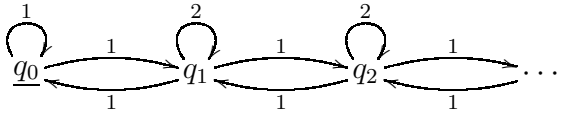
Any (state labelled with a) word w without dots is considered an output state (thus underlined), and has two children; and any word with one or more dots has four children, which arise by replacing the left-most dot in four different ways. Here is a kind of grammar for the 'growth' of our automaton,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



where w and w_1 do not contain any dots. States can next be identified according to the number of dots they contain:



yielding the following automaton:



and the following expressions for the stream of answers $S(q_0)$:

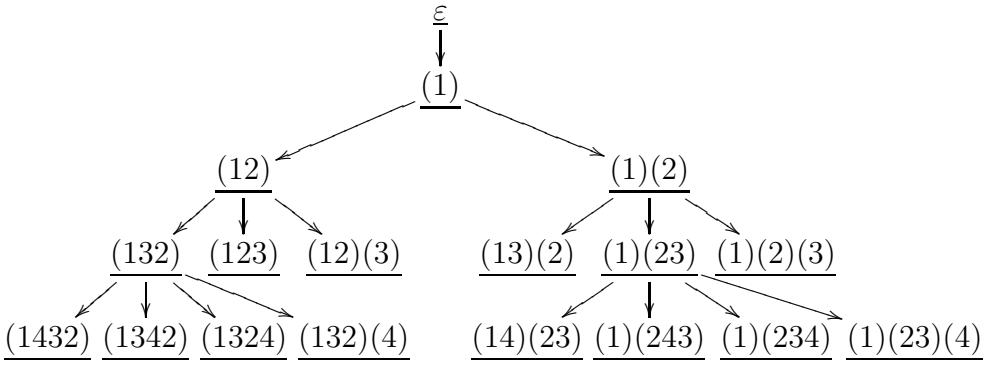
$$
\begin{aligned}
S(q_0) &= \cfrac{1}{1 - X - \cfrac{X^2}{1 - 2X - \cfrac{X^2}{1 - 2X - \cfrac{X^2}{\ddots}}}} \\
&= \frac{2}{1 + \sqrt{1 - 4X}} \\
&= \cfrac{1}{1 - \cfrac{X}{1 - \cfrac{X}{1 - \cfrac{X}{\ddots}}}}
\end{aligned}
$$

(= $(1, 1, 2, 5, 14, \ldots)$, the Catalan numbers). The first equality follows from formula (3), the second and third by some general stream calculus.

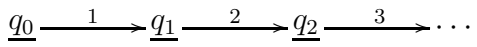
9 Permutations and cycles

A permutation (bijection) $p : \{1, \ldots, k\} \to \{1, \ldots, k\}$ can be represented by the corresponding sequence of images $p = (p(1) \cdots p(k))$. 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 $c = (x_1 \cdots x_k)$ denotes a permutation of $\{x_1, \ldots, x_k\}$ 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 $\{1, \ldots, k\}$ (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 $\{1, \ldots k\}$ 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:

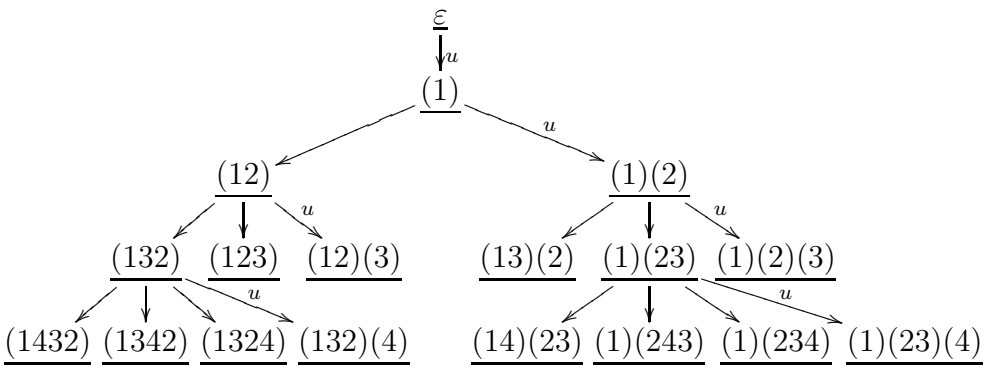


Applying formula (4), with $l_k = 0$ and $u_k = k+1$ for all $k \geq 0$, we obtain

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

for the stream $(s_0, s_1, s_2, \ldots)$ of counts we are after. In other words, there are $k!$ different permutations of the set $\{1, \ldots, k\}$, 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:

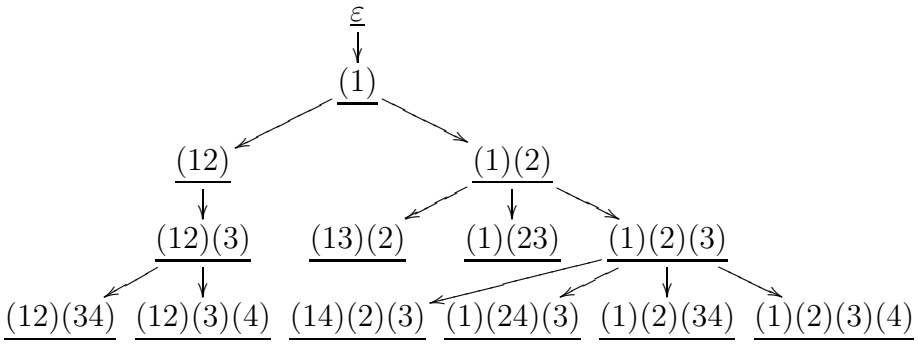
$$\underline{q_0} \xrightarrow{u} \underline{q_1} \xrightarrow{u+1} \underline{q_2} \xrightarrow{u+2} \cdots$$

to which, as before, formula (4) an be applied, now yielding

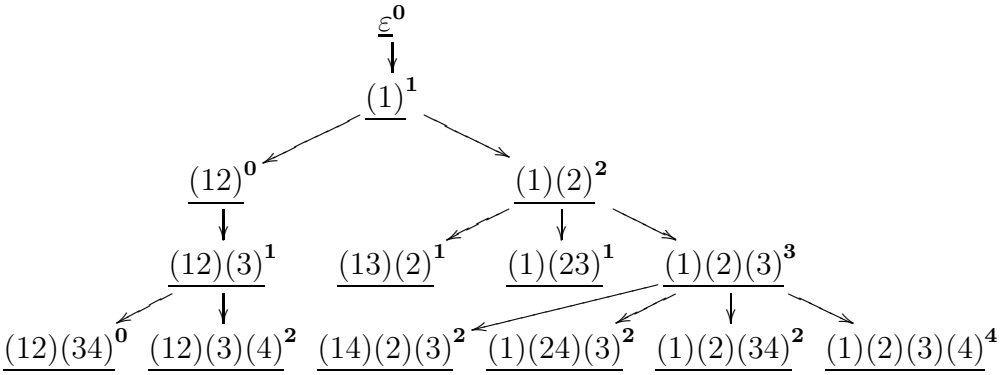
$$S(q_0) = 1 + u \times X + u(u+1) \times X^2 + u(u+1)(u+2) \times X^3 + \cdots$$

(which with some elementary stream calculus can be proved to equal $(1-X)^{-\underline{u}}$, that is, the shuffle product of $(1-X)^{-\underline{1}}$ with itself, u times). This 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 $\{1, \ldots, n\}$ consisting of k cycles (these numbers are known as the Stirling numbers of the first kind). An alternative approach would have been to treat both X and u as formal variables. This would bring us to multivariate stream calculus, which is omitted here.

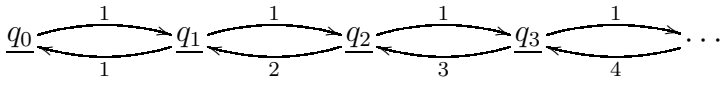
What is, moving to a next question, for any $k \geq 0$ the number s_k of permutations p of $\{1, \ldots, k\}$ such that $p \circ p = 1$ (the so-called involutions)? For this we return to the first automaton of this section, from which we now remove all states but the ones consisting of 1- and 2-cycles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

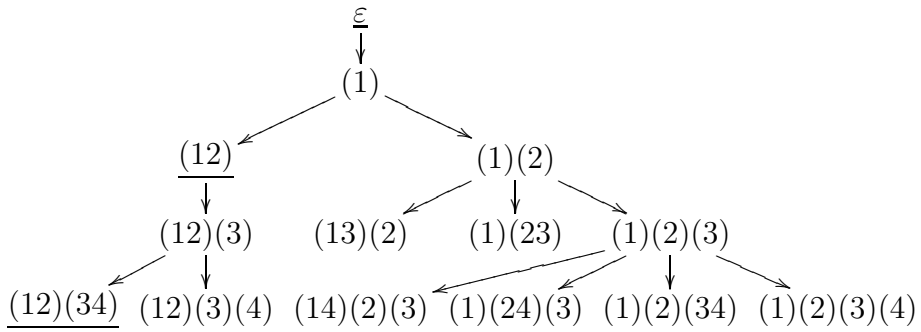


yielding the following automaton

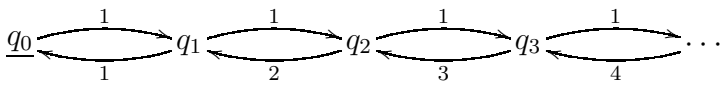


to which formula (5) can be applied. For the special case of permutations consisting of 2-cycles only, the same enumeration as for the involutions can be

used, the only difference being that states that contain 1-cycles are no longer output states:



The corresponding minimized automaton now looks like



with, according to formula (3), a pretty expression for the stream of answers:

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

10 Special numbers

We have already seen a few examples of streams of so-called special numbers, such as the Fibonacci and the Catalan numbers. In Figure 1, a table is presented listing automaton representations for a number of such streams, including the outcomes of some further counting experiments that we have been performing (on permutations, set partitionings, tilings, polyomino's, and many more), and that we have not described in full detail here for lack of space (see [8] for full details). We have also included the table simply because it gives such a pretty uniform presentation of all these different streams. In Figure 1, the names on the right refer to the stream $S(q_0)$ 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. Note that the table contains two different representations for the Stirling numbers of the first kind. The claim is that they are equivalent, which can be proved formally by coinduction. The same holds for the special case of $u = 1$, yielding two different but equivalent representations of the factorial numbers. A similar remark applies to the two representations of the Stirling numbers of the second kind, and the Bell numbers.

Let us emphasize once again the tight connection between weighted automata and stream calculus, by means of the following example. The automaton representation of the stream τ of the tangent numbers in Figure 1 arises

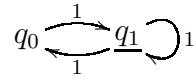
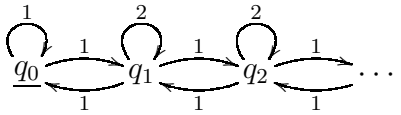
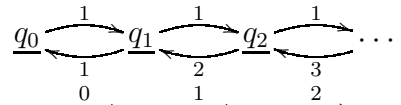
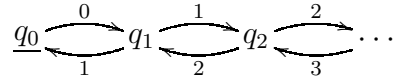
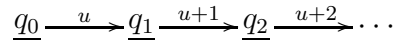
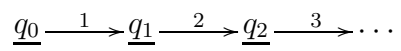
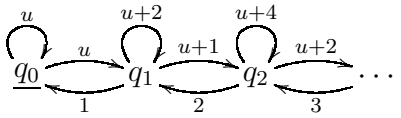
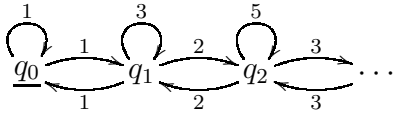
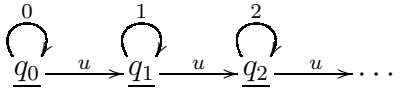
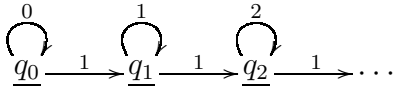
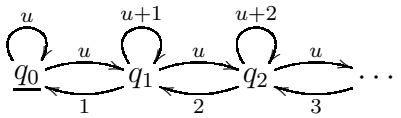
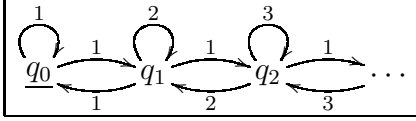
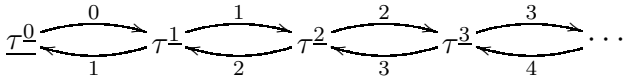
automaton	represents
	Fibonacci numbers
	Catalan numbers
	involutions
	tangent numbers
	Stirling numbers (1st)
	factorial numbers ($u = 1$)
	Stirling numbers (1st)
	factorial numbers ($u = 1$)
	Stirling numbers (2nd)
	Bell numbers ($u = 1$)
	Stirling numbers (2nd)
	Bell numbers ($u = 1$)

Fig. 1. Representations of special numbers

as the minimization of a tree-shaped automaton enumeration of all so-called alternating permutations of the odd natural numbers (the details are irrelevant here). At the same time, τ is also equal to the stream of the Taylor coefficients of the tangent function (hence the name), and satisfies, in stream calculus, the following differential equation: $\tau' = 1 + (\tau \otimes \tau)$ and $\tau(0) = 0$. Using the method of 'split derivatives', already illustrated with a small example at the end of Section 2, the automaton for τ can also be constructed in a purely formal way, not guided by any combinatorial insight whatsoever, using the differential equation for τ as follows. For any $n \geq 0$ let $\tau^{\underline{n}}$ denote the shuffle product of τ with itself, n times; as usual, $\tau^{\underline{0}} = 1$. Computing the derivative of $\tau^{\underline{n}}$ in stream calculus gives

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

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



(The only output state is $\tau^{\underline{0}} = 1$, since it has initial value 1.) Up to a renaming of states, this is precisely the automaton for the tangent numbers in the table of Figure 1.

11 Discussion

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 [4] and also [1]. (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 6), we have distinguished between rational and algebraic streams. Notably in the work of Flajolet, such as [5], 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 sofar, 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 which have transitions labelled by complete stream expressions (such as X^3 or $X + X^2$) rather than by real numbers (and X) only.

References

[1] 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.

[2] J. Berstel and C. Reutenauer. *Rational series and their languages*, volume 12 of *EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, 1988.

[3] P. Flajolet. Combinatorial aspects of continued fractions. *Discrete Mathematics*, 32:125–161, 1980.

[4] P. Flajolet and R. Sedgewick. The average case analysis of algorithms: Counting and generating functions. Research Report 1888, INRIA Rocquencourt, 1993. 116 pages.

[5] P. Flajolet and R. Sedgewick. Analytical combinatorics: Functional equations, rational and algebraic functions. Research Report 4103, INRIA Rocquencourt, 2001. 98 pages.

[6] R.L. Graham, D.E. Knuth, and O. Patashnik. *Concrete mathematics, second edition*. Addison-Wesley, 1994.

[7] 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.

[8] J.J.M.M. Rutten. Coinductive counting with weighted automata. Technical Report, CWI, 2002. To appear.

[9] R.P. Stanley. *Enumerative Combinatorics I*, volume 49 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, 1997.

[10] R.P. Stanley. *Enumerative Combinatorics II*, volume 62 of *Cambridge Studies in Advanced Mathematics*. Cambridge University Press, 1999.