Coalgebraic Semantics of Heavy-Weighted Automata

Marie Fortin[1,3], Marcello M. Bonsangue[2,3(✉)], and Jan Rutten[3,4]

[1] École Normale Supérieure de Cachan, Cachan, France
[2] LIACS – Leiden University, Leiden, The Netherlands
m.m.bonsangue@liacs.leidenuniv.nl
[3] Centrum Wiskunde & Informatica, Amsterdam, The Netherlands
[4] ICIS – Radboud University Nijmegen, Nijmegen, The Netherlands

Abstract. In this paper we study *heavy-weighted automata*, a generalization of weighted automata in which the weights of the transitions can be formal power series. As for ordinary weighted automata, the behaviour of heavy-weighted automata is expressed in terms of formal power series. We propose several equivalent definitions for their semantics, including a system of *behavioural differential equations* (following the approach of *coinductive calculus*), or an embedding into a coalgebra for the functor $S \times (-)^A$, for which the set of formal power series is a final coalgebra. Using techniques based on bisimulations and coinductive calculus, we study how ordinary weighted automata can be transformed into more compact heavy-weighted ones.

1 Introduction

Weighted automata are a generalization of non-deterministic automata in which each transition carries a weight [5]. This weight is an element of a semiring, representing, for example, the cost or probability of taking the transition. Weighted automata have many different areas of application. Recently, for example, they have been used to solve counting problems, first in [10] with a procedure called *coinductive counting*, then in [4] with the *counting automata methodology*.

Whereas non-deterministic automata either accept or reject a word, weighted automata associate with each word the cost of its execution. Their semantics is thus defined in terms of *weighted languages*, also called *formal power series*, which are functions mapping words to weights.

Formal power series form themselves a semiring, and thus can be used as weights of transitions, yielding what we call *heavy-weighted automata*. In [4], such automata are used to give a compact representation of some combinatorial problems. One of our motivation here is to extend the coalgebraic setting existing for ordinary weighted automata [1,10] to study equivalence between heavy-weighted automata. In particular, we are interested in producing a more compact representation of some well-shaped infinite weighted automata, generalizing the examples given in [4].

M. Codescu et al. (Eds.): WADT 2014, LNCS 9463, pp. 48–68, 2015.
DOI: 10.1007/978-3-319-28114-8_4

A second motivation for the introduction of heavy-weighted automata is to provide something similar to what generalized automata (where transitions are labeled by regular expressions) are to ordinary automata. In particular, we will see that Brzozowski-McCluskey's *state elimination method* [3,14] to compute the regular expression associated with a finite automaton also works for weighted automata. The method is not new, but is a nice application of our definition of heavy-weighted automata.

Though heavy-weighted automata can be seen as classic weighted automata over the semiring of weighted languages, their standard semantics as such (given in terms of power series over the semiring of power series) is not so interesting. Instead, we define the semantics of heavy-weighted automata in terms of (ordinary) power series, in three equivalent ways:

- by a system of equations linking the semantics of the different states.
- in terms of the final homomorphism. Here the set of weighted languages is the final coalgebra for the functor $S \times (-)^A$, which means that from any $S \times (-)^A$-coalgebra, there is a unique coalgebra homomorphism to the set of all weighted languages. Heavy-weighted automata are not themselves $S \times (-)^A$-coalgebras, but they can be embedded into one, using some kind of determinization procedure (as introduced in [13]).
- by giving a procedure that transforms a heavy-weighted automaton into an ordinary weighted automaton. This is done by composing ordinary weighted automata that recognize the weighted languages labeling the transitions of the heavy-weighted automaton.

We proceed as follows. First we briefly discuss some related work. Then, in Sect. 2 we recall the basic notions on formal power series, coinductive calculus and (ordinary) weighted automata. In Sect. 3, we define heavy-weighted automata and their behaviour, first in terms of behavioural differential equations and then in terms of a final homomorphism of coalgebras. In Sect. 4, we give a new interpretation to the behaviour of heavy-weighted automata, by giving a procedure that transforms a heavy-weighted automata into an ordinary weighted automata. In Sect. 5.1, we recall the state elimination method, and in Sect. 5.2, we show how heavy-weighted automata can be used to give a more compact representation for certain well-shaped infinite weighted automata.

Missing proof details can be found in the extended technical report [6].

Related Work

Our study of heavy-weighted automata is motivated by the work done in [4]. *Counting automata*, which are also automata having power series as weights, are used to model combinatorial problems. Some examples of reductions of infinite weighted automata to finite ones are given; yet there is no general format for such reductions. In Sect. 5.2 we describe a generalization of these reductions to a particular class of well-shaped, infinite weighted automata.

Automata in which transitions are labeled by power series were already present in [12]. The *state elimination method* (see Sect. 5.1) is also mentioned,

though rather as the resolution of a system of equations. Apart from that, our work has no real intersection with what is done in [12]. A first difference is that we are interested in both finite and infinite atomata, whereas [12] focuses mostly on the finite case. Furthermore, no real separation is made in [12] between automata with ordinary weights and automata with power series as weights; as a consequence, the question of the transformation of a heavy-weighted automaton into an ordinary weighted automaton (see Sect. 4) is not raised.

Our definition of heavy-weighted automata present some differences with both [4,12]. In particular, we choose to define heavy-weighted automata in such a way that their behaviour is always defined, by requiring finite branching and not allowing ε-transitions. Thus we do not investigate the convergence issues that are given some importance in [4,12].

Our coalgebraic approach to heavy-weighted automata is new, and follows previous work on (ordinary) weighted automata, see e.g. [1,9,10]. Our construction in Sect. 3 can be seen as an instance of the generalized determinization construction described in [13]. Finally, our work relies largely on *coinductive calculus* for streams and power series, see e.g. [9,11].

2 Preliminaries

2.1 Coalgebras

We recall some basic definitions about coalgebras. Given a functor $\mathcal{F} : \mathrm{Set} \to \mathrm{Set}$, an *$\mathcal{F}$-coalgebra* is a pair (X, f) consisting of a set X and a function $f : X \to \mathcal{F}X$.

An *$\mathcal{F}$-homomorphism* from an $\mathcal{F}$-coalgebra (X, f) to another $\mathcal{F}$-coalgebra (Y, g) is a function $h : X \to Y$ such that diagram to the right commutes, i.e. such that $g \circ h = \mathcal{F}h \circ f$. An $\mathcal{F}$-coalgebra (Y, g) is called *final* if for any $\mathcal{F}$-coalgebra (X, f) there exists a unique $\mathcal{F}$-homomorphism $[\![-]\!] : X \to Y$.

$$\begin{array}{ccc} X & \xrightarrow{\;h\;} & Y \\ {\scriptstyle f}\downarrow & & \downarrow{\scriptstyle g} \\ \mathcal{F}X & \xrightarrow[\mathcal{F}h]{} & \mathcal{F}Y \end{array}$$

2.2 Formal Power Series

Throughout the paper, A denotes a nonempty finite alphabet.

Weighted automata associate to each input word a certain weight: their behaviour is defined in terms of *weighted languages*, also called *formal power series*, which are functions mapping words to elements of a semiring.

A *semiring* $(S, +, \times, 0, 1)$ consists of a set S together with two binary operations $+$ and $\times$ and two constants $0, 1 \in S$, such that:

(1) $(S, +, 0)$ is a commutative monoid and $(S, \times, 1)$ is a monoid;
(2) $\times$ distributes over $+$: $\forall x, y, z \in S,\ x \times (y + z) = x \times y + x \times z$ and $(x + y) \times z = x \times z + y \times z$;
(3) 0 is an annihilator for $\times$: $\forall x \in S, x \times 0 = 0 \times x = 0$.

Given a semiring S and a finite alphabet A, a *formal power series with coefficients in S and variables in A* is a function $\sigma : A^* \to S$. We denote by $S\langle\langle A\rangle\rangle$ the set of all formal power series with coefficients in S and variables in A.

Examples:

(1) Taking $S = \mathbb{B}$, where $\mathbb{B}$ denotes the boolean semiring, $\mathbb{B}\langle\langle A\rangle\rangle$ is isomorphic to the set of all formal languages over the alphabet A.
(2) $S\langle\langle\{X\}\rangle\rangle$ is isomorphic to the set of all streams with values in S (that is, the set of all functions $\mathbb{N} \to S$). We denote this set by S^ω, and we sometimes write $(s_0, s_1, \ldots)$ for the stream $i \mapsto s_i$.

The *support* of a formal power series σ is the set $\{w \in A^* \mid \sigma(w) \neq 0\}$. A *polynomial* is a power series with finite support. The set of all polynomials with coefficients in S and variables in A is denoted by $S\langle A\rangle$.

Finally, a power series $\sigma \in S\langle\langle A\rangle\rangle$ is called *proper* when $\sigma(\varepsilon) = 0$ (where ε is the empty word). We denote by $S\langle\langle A\rangle\rangle_p$ the set of all proper power series.

$S\langle\langle A\rangle\rangle$ can be given a $S \times (-)^A$-coalgebra structure using a generalization of the notion of *Brzozowski derivatives*. Given $a \in A$, the *a-derivative* $\sigma_a \in S\langle\langle A\rangle\rangle$ of a power series σ is defined by $\sigma_a(w) = \sigma(aw)$ and the *output* of σ is defined as $O(\sigma) = \sigma(\varepsilon)$. When A is a singleton, we write σ' for the derivative of σ.

We define $\Delta : S\langle\langle A\rangle\rangle \to S\langle\langle A\rangle\rangle^A$ by $\Delta(\sigma)(a) = \sigma_a$, and $\langle O, \Delta\rangle : S\langle\langle A\rangle\rangle \to S \times S\langle\langle A\rangle\rangle^A$ as the function $\sigma \mapsto (O(\sigma), \Delta(\sigma))$. Then $(S\langle\langle A\rangle\rangle, \langle O, \Delta\rangle)$ is a coalgebra for the functor $S \times (-)^A$. Moreover, we have the following theorem [9].

Theorem 1. *$(S\langle\langle A\rangle\rangle, \langle O, \Delta\rangle)$ is a final coalgebra for the functor $S \times (-)^A$.*

2.3 A Coinductive Calculus for Power Series

We now present some basic facts from the coinductive calculus for streams and power series developed in [9,11].

First we recall the coinduction proof principle, which will be one of our main proof techniques. A *bisimulation* on formal power series is a relation $\mathcal{R} \subseteq S\langle\langle A\rangle\rangle \times S\langle\langle A\rangle\rangle$ such that, for all σ and τ in $S\langle\langle A\rangle\rangle$, if $\sigma \mathcal{R} \tau$ then

(1) $O(\sigma) = O(\tau)$;
(2) for all $a \in A$, $\sigma_a \mathcal{R} \tau_a$.

The union of all bisimulation relations is called *bisimilarity*, and is denoted by $\sim$. A relation $\mathcal{R} \subseteq S\langle\langle A\rangle\rangle \times S\langle\langle A\rangle\rangle$ is a *bisimulation-up-to* if its closure under linear combination is a bisimulation relation [8].

Theorem 2 (Coinduction). *For all $\sigma, \tau \in S\langle\langle A\rangle\rangle$, if $\sigma \sim \tau$ then $\sigma = \tau$.*

Note that the converse trivially holds, since $\{(\sigma, \sigma) \mid \sigma \in S\langle\langle A\rangle\rangle\}$ is a bisimulation. The consequence of Theorem 2 is that to prove the equality of two power series σ and τ, it is sufficient to establish the existence of a bisimulation $\mathcal{R}$ such that $\sigma \mathcal{R} \tau$.

Next, various operators on power series are defined coinductively. Coinductive definitions are given as *behavioural differential equations*, which have a unique solution. In particular, there exist a unique binary operator $+$, a unique binary operator $\times$, and for all $s \in S$ and $b \in A$, a unique $[s] \in S\langle\langle A \rangle\rangle$ and $[b] \in S\langle\langle A \rangle\rangle$ satisfying the following system of behavioural differential equations:

a-derivative (for all $a \in A$)	Initial value
$[s]_a = [0]$	$O([s]) = s$
$[b]_b = [1]$, $[b]_a = [0]$ for $a \neq b$	$O([b]) = 0$
$(\sigma + \tau)_a = \sigma_a + \tau_a$	$O(\sigma + \tau) = O(\sigma) + O(\tau)$
$(\sigma \times \tau)_a = (\sigma_a \times \tau) + ([O(\sigma)] \times \tau_a)$	$O(\sigma \times \tau) = O(\sigma) \times O(\tau)$

We then have for all $s \in S$, $[s](\varepsilon) = s$ and $[s](w) = 0$ if $w \neq \varepsilon$. For $b \in A$, $[b](b) = 1$ and $[b](w) = 0$ if $w \neq b$. The coinductive definitions given for the sum and convolution product coincide with the classic pointwise definitions: for all $\sigma, \tau \in S\langle\langle A \rangle\rangle$ and $w \in A^*$,

$$(\sigma + \tau)(w) = \sigma(w) + \tau(w) \quad \text{and} \quad (\sigma \times \tau)(w) = \sum_{uv=w} \sigma(u)\tau(v)\,.$$

When S is a ring, we also define the inverse σ^{-1} of series σ such that $O(\sigma)$ is invertible in S, as the unique solution to $(\sigma^{-1})_a = -[O(\sigma)^{-1}] \times \sigma_a \times \sigma^{-1}$ and $O(\sigma^{-1}) = O(\sigma)^{-1}$. We then have $\sigma \times \sigma^{-1} = [1] = \sigma^{-1} \times \sigma$.

Theorem 3 (Fundamental Theorem). *For all $\sigma \in S\langle\langle A \rangle\rangle$,*

$$\sigma = [O(\sigma)] + \sum_{a \in A} [a] \times \sigma_a.$$

As a notational convenience, we will write s for $[s]$ and b for $[b]$ whenever it is clear from the context whether we intend elements of S and A or formal power series. Similarly, we will identify a word $w = a_1 \dots a_n \in A^*$ with the product $[a_1] \times \dots \times [a_n]$.

With these conventions, for all $\sigma \in S\langle\langle A \rangle\rangle$, $\sigma = \sum_{w \in A^*} \sigma(w) \times w$.

2.4 Rational Power Series

A family $\{\sigma_i \mid i \in I\}$ of power series is called *locally finite* when for all $w \in A^*$, the set $I_w = \{i \mid \sigma_i(w) \neq 0\}$ is finite. In this case, we define the sum $\sum_{i \in I} \sigma_i$ by $\left(\sum_{i \in I} \sigma_i\right)(w) = \sum_{i \in I_w} \sigma_i(w)$.

Let σ be a *proper* power series. For all $n \in \mathbb{N}$, we denote by σ^n the n-fold product of σ with itself: $\sigma^0 = 1$, and $\sigma^{n+1} = \sigma \times \sigma^n$. Then for all $w \in A^*$ and $n > |w|$, $\sigma^n(w) = 0$. Hence $\{\sigma^n \mid n \in \mathbb{N}\}$ is locally finite. We can thus define the *star* of a proper power series σ as the sum $\sigma^* = \sum_{n \in \mathbb{N}} \sigma^n$.

We define the set $\mathsf{RatE}_S(A)$ of all *rational S-expressions* E as follows:

$$E ::= s \in S \mid a \in A \mid (E + E) \mid (E \times E) \mid E^* .$$

We then define simultaneously the set of *valid* rational S-expressions, and the power series $\mathsf{val}(E)$ denoted by a valid expression, by induction:

- for all $s \in S$, s is valid and $\mathsf{val}(s) = s$;
- For all $a \in A$, a is valid and $\mathsf{val}(a) = a$;
- if E_1 and E_2 are valid, $(E_1 + E_2)$ is valid and $\mathsf{val}(E_1 + E_2) = \mathsf{val}(E_1) + \mathsf{val}(E_2)$;
- if E_1 and E_2 are valid, $(E_1 \times E_2)$ is valid and $\mathsf{val}(E_1 \times E_2) = \mathsf{val}(E_1) \times \mathsf{val}(E_2)$;
- if E is valid and $\mathsf{val}(E)$ is proper, E^* is valid and $\mathsf{val}(E^*) = \mathsf{val}(E)^*$.

A power series $\sigma \in S\langle\langle A \rangle\rangle$ is called *rational* if there exists a valid rational S-expression E such that $\mathsf{val}(E) = \sigma$. We denote by $S_{\mathrm{rat}}\langle\langle A \rangle\rangle$ the set of all rational power series.

2.5 Weighted Automata

Weighted automata are a generalisation of automata, where each transition has a weight in addition to the input letter. We associate a weight with each path in the automaton by multiplying the weights of all taken transitions; and we associate a weight with each word by adding the weights of all paths accepting it.

Let S be a semiring and A a finite alphabet. For any set X, we denote by $X \to_f S$ the set of all functions $g : X \to S$ such that $\{x \in X \mid g(x) \neq 0\}$ is finite.

Formally, a *weighted automaton* (or WA, for short) with input alphabet A and weights in the semiring S consists of a pair $(Q, \langle o, t \rangle)$, where:

- Q is a set of *states*.
- $o : Q \to S$ is the *output function*.
- $t : Q \to (Q \to_f S)^A$ is the *transition function*.

We will write $p \xrightarrow{a,s} q$ for $t(p)(a)(q) = s$, and $p \xrightarrow{s}$ for $o(p) = s$. A state $q \in Q$ is called *final* when $o(q) \neq 0$.

The *behaviour* $\mathcal{S}(q)$ of a state $q \in Q$, or weighted language recognized by state q, is classically defined as follows: for all $w = a_1 \dots a_n \in A^*$,

$$\mathcal{S}(q)(w) = \sum_{q_1,\dots,q_n \in Q} t(q)(a_1)(q_1) \times t(q_1)(a_2)(q_2) \times \cdots \times t(q_{n-1})(a_n)(q_n) \times o(q_n).$$

Note that this is a finite sum, since for all $q_i \in Q$ there are only finitely many $q_{i+1} \in Q$ such that $t(q_i)(a_{i+1})(q_{i+1}) \neq 0$. This follows the intuitive definition we gave before: an accepting path for w starting at state q is of the form $q = q_0 \xrightarrow{a_1,s_1} q_1 \xrightarrow{a_2,s_2} \cdots \xrightarrow{a_n,s_n} q_n \xrightarrow{s}$, with $s_i \neq 0$ and $s \neq 0$. Taking the sum of $s_1 \dots s_n s$ for all such paths, we obtain the given expression. However, we will mostly use the following equivalent coinductive definition: $\mathcal{S} : Q \to S\langle\langle A \rangle\rangle$ is

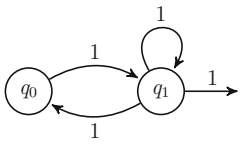
defined as the unique solution to the following system of behavioural differential equations: for all $q \in Q$ and $a \in A$,

$$\mathcal{S}(q)_a = \sum_{r \in Q} t(q)(a)(r) \times \mathcal{S}(r) \qquad O(\mathcal{S}(q)) = o(q)\,.$$

Using the fundamental theorem of coinductive calculus, this is equivalent to

$$\mathcal{S}(q) = o(q) + \sum_{r \in Q} \left(\sum_{a \in A} a \times t(q)(a)(r) \right) \times \mathcal{S}(r)\,.$$

Example. Take $A = \{X\}$, $S = (\mathbb{R}, +, \times)$ and the automaton defined to the right. We have $\mathcal{S}(q_0) = X \times \mathcal{S}(q_1)$ and $\mathcal{S}(q_1) = 1 + X \times \mathcal{S}(q_0) + X \times \mathcal{S}(q_1)$. Recall that, as defined in Sect. 2.3, X is the stream $(0, 1, 0, 0, \ldots)$. These equations lead to $S(q_0) = X \times (1 - X - X^2)^{-1} = (0, 1, 1, 2, 3, \ldots)$, which corresponds to the Fibonacci sequence.



A weighted automaton is called *finite* when its set of states is finite. A power series $\sigma \in S\langle\langle A \rangle\rangle$ is *recognizable* when there exist a finite weighted automaton $\mathcal{A} = (Q, \langle o, t \rangle)$ and $q_0 \in Q$ such that $\mathcal{S}(q_0) = \sigma$. We denote by $S_{\mathrm{rec}}\langle\langle A \rangle\rangle$ the set of all recognizable power series.

Theorem 4 (Kleene-Schutzenberger). *$S_{rat}\langle\langle A \rangle\rangle = S_{rec}\langle\langle A \rangle\rangle$.*

Note that when we don't require the set of states to be finite, for any power series $\sigma \in S\langle\langle A \rangle\rangle$, there exist a weighted automaton $(Q, \langle o, t \rangle)$ and $q_0 \in Q$ such that $\mathcal{S}(q_0) = \sigma$. For instance, take $Q = A^*$, $o = \sigma$, $t(w)(a)(aw) = 1$, and $t(w)(a)(v) = 0$ in all other cases. Then $\mathcal{S}(\varepsilon) = \sigma$.

3 Heavy-Weighted Automata

We generalize weighted automata to *heavy-weighted automata*, by allowing the weights of the transitions to be any power series rather than an element of the semiring S.

name	output function	transition function
Weighted automaton (WA)	$o : Q \to S$	$t : Q \to (Q \to_f S)^A$
Heavy-weighted automaton (HWA)	$o : Q \to S$	$t : Q \to (Q \to_f S\langle\langle A \rangle\rangle)^A$

Fig. 1. Definitions of WAs and HWAs

A *heavy-weighted automaton* (or HWA, for short) over the semiring S and the alphabet A consists of a pair $(Q, \langle o, t \rangle)$, where:

– Q is a set of *states*.
– $o : Q \to S$ is the *output function*.
– $t : Q \to (Q \to_f S\langle\langle A\rangle\rangle)^A$ is the *transition function*.

For any $p, q \in Q$, we also define the *cumulated weight between p and q* as

$$w(p)(q) = \sum_{a \in A} a \times t(p)(a)(q).$$

Note that for any $p, q \in Q$, $a \in A$, $w(p)(q)$ is proper and $t(p)(a)(q) = w(p)(q)_a$. As for ordinary weighted automata, we write $p \xrightarrow{a,\sigma} q$ for $t(p)(a)(q) = \sigma$, or $p \xrightarrow{\tau} q$ for $w(p)(q) = \tau$.

Remark. The transition function t is uniquely determined by w. Thus a HWA can equivalently be defined by giving its set of states Q, its output function $o : Q \to S$, and its cumulated weights, that is, a function $w : Q \to (Q \to_f S\langle\langle A\rangle\rangle_p)$. (Recall that here $S\langle\langle A\rangle\rangle_p$ denotes the set of all proper series.) Indeed, given any $w : Q \to (Q \to_f S\langle\langle A\rangle\rangle_p)$, define $t : Q \to (Q \to_f S\langle\langle A\rangle\rangle)^A$ by $t(p)(a)(q) = w(p)(q)_a$. The fundamental theorem then gives $w(p)(q) = \sum_{a \in A} a \times t(p)(a)(q)$.

Let $\mathcal{A} = (Q, \langle o, t\rangle)$ a HWA. The *behaviour* of a state $q \in Q$ is defined as a power series $\mathcal{S}(q) \in S\langle\langle A\rangle\rangle$, and satisfies the same equations as we had for WAs. More precisely, $\mathcal{S} : Q \to S\langle\langle A\rangle\rangle$ is defined as the unique solution to the following system of behavioural differential equations: for all $q \in Q$ and $a \in A$,

$$\mathcal{S}(q)_a = \sum_{r \in Q} t(q)(a)(r) \times \mathcal{S}(r) \qquad\qquad O(\mathcal{S}(q)) = o(q)\,.$$

HWAs are indeed a generalization of WAs, in the sense that any WA can be seen as a HWA, by identifying the weights in S with power series in $S\langle\langle A\rangle\rangle$. Since the behaviour of WAs and HWAs are defined by the same system of equations, the behaviour of a WA is unchanged when we consider it as a HWA.

Final Semantics for Heavy-Weighted Automata

In coalgebra theory, the behaviour of a system is usually defined in terms of final homorphism: given a functor $\mathcal{F}$ with a final coalgebra (Ω, ω), every element of an $\mathcal{F}$-coalgebra (X, f) is associated to a canonical representative in Ω by the final $\mathcal{F}$-homomorphism $[\![-]\!] : X \to \Omega$.

Here however, HWAs are coalgebras for the functor $X \mapsto S \times (X \to_f S\langle\langle A\rangle\rangle)^A$, whereas their semantics is defined in terms of formal power series, which is the final coalgebra for the functor $X \mapsto S \times X^A$.

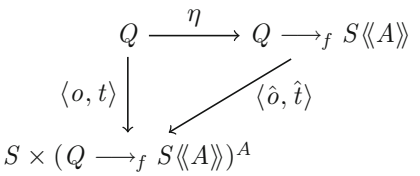
The objective of this subsection is to propose another definition for the semantics of HWAs, equivalent to the previous one, but expressed in terms of final homomorphisms. For that, we will associate each HWA $(Q, \langle o, t\rangle)$ to an $S \times (-)^A$-coalgebra, in a construction similar to the determinization procedure for automata.

For the remainder of this subsection, we fix a HWA $\mathcal{A} = (Q, \langle o, t\rangle)$. Similarly to the powerset construction, we define a map $\eta : Q \to (Q \to_f S\langle\langle A\rangle\rangle)$ by

$$\eta(p)(q) = \begin{cases} 1 & \text{if } p = q \\ 0 & \text{if } p \neq q. \end{cases}$$

Note that every $\alpha : Q \to_f S\langle\langle A\rangle\rangle$ can be expressed as $\alpha = \sum_{q \in Q} \alpha(q) \cdot \eta(q)$ where, for all $\sigma \in S\langle\langle A\rangle\rangle$ and $\beta : Q \to_f S\langle\langle A\rangle\rangle$, $\sigma \cdot \beta : Q \to_f S\langle\langle A\rangle\rangle$ is defined as $q \mapsto \sigma \times \beta(q)$.

We want to define an $S \times (-)^A$ coalgebra structure $\langle \hat{o}, \hat{t} \rangle$ for $(Q \to_f S\langle\langle A\rangle\rangle)$ compatible with $\langle o, t \rangle$, meaning that $\hat{o} \circ \eta = o$ and $\hat{t} \circ \eta = t$. Moreover, $\hat{o}$ and $\hat{t}$ should behave as output and derivative functions, that is, the following equalities should hold for every $\alpha, \beta \in Q \to_f S\langle\langle A\rangle\rangle$, $\sigma \in S\langle\langle A\rangle\rangle$ and $a \in A$:

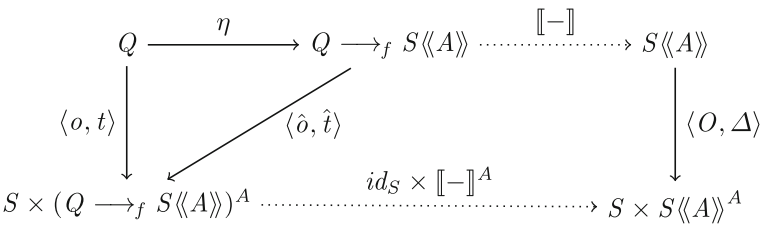


$$\begin{aligned} \hat{o}(\alpha + \beta) &= \hat{o}(\alpha) + \hat{o}(\beta) & \qquad \hat{t}(\alpha + \beta) &= \hat{t}(\alpha) + \hat{t}(\beta) \\ \hat{o}(\sigma \cdot \alpha) &= O(\sigma) \cdot \hat{o}(\alpha) & \qquad \hat{t}(\sigma \cdot \alpha)(a) &= \sigma_a \cdot \alpha + O(\sigma) \cdot \hat{t}(\alpha) \end{aligned}$$

This leads to the following definitions for $\hat{o}$ and $\hat{t}$:

$$\hat{o}(\alpha) = \sum_{q \in Q} O(\alpha(q)) \cdot o(q) \qquad \hat{t}(\alpha)(a) = \sum_{q \in Q} (\alpha(q)_a \cdot \eta(q) + O(\alpha(q)) \cdot t(q)(a))$$

We are now going to exploit the fact that $S\langle\langle A\rangle\rangle$ is a final $S \times (-)^A$-coalgebra to define the semantics of $\mathcal{A}$. Denote by $[\![-]\!]$ the unique $S \times (-)^A$-homomorphism from $(Q \to_f S\langle\langle A\rangle\rangle)$ to $S\langle\langle A\rangle\rangle$.



We now define the *behaviour* of state $q \in Q$ as the power series $[\![\eta(q)]\!]$, and show that it is indeed the same as the behaviour $\mathcal{S}(q)$ defined in Fig. 1.

Lemma 1. *For all* $\alpha, \beta \in (Q \to_f S\langle\langle A\rangle\rangle)$ *and* $\sigma \in S\langle\langle A\rangle\rangle$,

$$[\![\alpha + \beta]\!] = [\![\alpha]\!] + [\![\beta]\!] \quad \textit{and} \quad [\![\sigma \cdot \alpha]\!] = \sigma \times [\![\alpha]\!].$$

Proof. It is enough to show that $\mathcal{R}_1 = \{([\![\alpha + \beta]\!], [\![\alpha]\!] + [\![\beta]\!]) \mid \alpha, \beta : Q \to_f S\langle\langle A\rangle\rangle\}$ and $\mathcal{R}_2 = \{([\![\sigma \cdot \alpha]\!], \sigma \times [\![\alpha]\!]) \mid \sigma \in S\langle\langle A\rangle\rangle, \alpha : Q \to_f S\langle\langle A\rangle\rangle\}$ are a bisimulation and a bisimulation-up-to, using the fact that $O([\![\alpha]\!]) = \hat{o}(\alpha)$ and $[\![\alpha]\!]_a = [\![\hat{t}(\alpha)(a)]\!]$. □

Proposition 1. *For all* $q \in Q$, $[\![\eta(q)]\!] = \mathcal{S}(q)$.

Proof. We show that $q \mapsto [\![\eta(q)]\!]$ satisfies the system of equations defining $\mathcal{S}$:

– for all $q \in Q$, $O([\![\eta(q)]\!]) = \hat{o}(\eta(q)) = o(q)$.
– for all $q \in Q$ and $a \in A$,

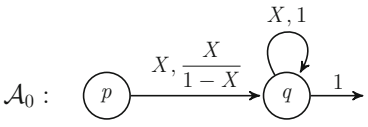
$$[\![\eta(q)]\!]_a = \left[\!\left[\hat{t}(\eta(q))(a)\right]\!\right] = [\![t(q)(a)]\!]$$
$$= \left[\!\left[\sum_{r\in Q} t(q)(a)(r)\cdot\eta(r)\right]\!\right] = \sum_{r\in Q} t(q)(a)(r)\times[\![\eta(r)]\!]. \qquad \square$$

4 From Heavy-Weighted Automata to Weighted Automata

As we saw in Sect. 3, there is a trivial injection from the set of WAs to the set of HWAs. Reciprocally, we want to be able to transform any HWA $\mathcal{A} = (Q, \langle o, t\rangle)$ into a WA $\hat{\mathcal{A}} = \left(\hat{Q}, \left\langle \hat{o}, \hat{t}\right\rangle\right)$ with weights in S and input alphabet A, such that for all $q \in Q$, there exists $\hat{q} \in \hat{Q}$ such that q and $\hat{q}$ have the same behaviour. (Intuitively, this means that whatever state q we choose as "initial" in $\mathcal{A}$, we can find a state $\hat{q}$ in $\hat{\mathcal{A}}$ having the same behaviour.) There are two motivations for this construction:

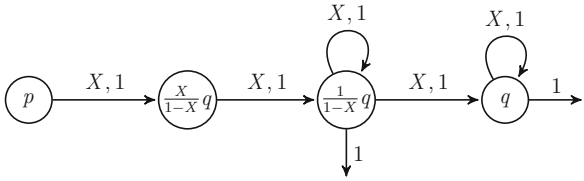
– giving a new interpretation to the semantics of HWAs.
– giving a constructive proof that HWAs and WAs have the same expressivity. In the general case this is trivial since any power series can be recognized both by a HWA, and a (possibly infinite) WA, as shown in Sect. 2.5. Yet there are other interesting cases; for instance we can require the set of states to be finite, and the "heavy-weights" to be rational power series.

Let us first look at an example. We take $A = \{X\}$ and $S = \mathbb{R}$, and consider the automaton $\mathcal{A}_0$ on the right:

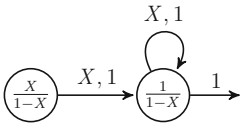


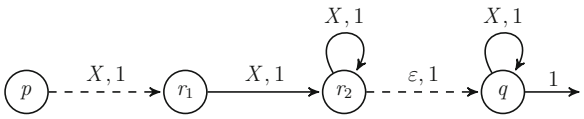
First Idea. We compute directly an equivalent WA by the method of "splitting the derivatives" [11]: we compute the successive derivatives of $\mathcal{S}(p)$, and add corresponding states at each step.

We have $\mathcal{S}(p)' = \frac{X}{1-X}\mathcal{S}(q)$, so we add a state with behaviour $\frac{X}{1-X}\mathcal{S}(q)$. Then $\left(\frac{X}{1-X}\mathcal{S}(q)\right)' = \frac{1}{1-X}\mathcal{S}(q)$, and we add a state with behaviour $\frac{1}{1-X}\mathcal{S}(q)$. Finally, $\left(\frac{1}{1-X}\mathcal{S}(q)\right)' = \frac{1}{1-X}\mathcal{S}(q) + \mathcal{S}(q)$, so we get the following automaton:



Second Idea. In the automaton on the right, the leftmost state recognize the stream $\frac{X}{1-X}$. We can plug it into $\mathcal{A}_0$, in place of the heavy-weighted transition, as follows:





We did not allow ε-transitions in our definition of weighted automata, because the behaviour of a weighted automaton with ε-transitions is not always well-defined: for instance if we have a cycle of ε-transitions, we could have infinitely many paths labeled by the same word, which would lead to an infinite sum when computing the behaviour of a state. Yet here we don't add any infinite path labeled by ε, and in this particular case it is easy to adapt all the definitions.

Removing the ε-transitions, we get precisely the same automaton as with the first method. This is not surprising, since in the first method, computing the derivative of $\mathcal{S}(q)$ amounted to computing the derivative of $\frac{X}{1-X}$.

We come back to the general case. Both methods can be generalized, and lead again to the same definition of the equivalent WA.

Let $\mathcal{A} = (Q, \langle o, t\rangle)$ be a HWA. Assume that for all $p, q \in Q$ and $a \in A$, we have a WA $\mathcal{A}_{p,a,q} = (Q_{p,a,q}, \langle o_{p,a,q}, t_{p,a,q}\rangle)$ and a state $i_{p,a,q} \in Q_{p,a,q}$ with behaviour $t(p)(a)(q)$.

We define a WA $\hat{\mathcal{A}} = \left(\hat{Q}, \langle \hat{o}, \hat{t}\rangle\right)$ by setting:

- $\hat{Q} = Q \uplus \biguplus_{\substack{p,q\in Q\\ a\in A}} Q_{p,a,q}$
- $\forall q \in Q,\ \hat{o}(q) = o(q)$, and $\forall p, q \in Q, a \in A, r \in Q_{p,a,q},\ \hat{o}(r) = o_{p,a,q}(r)o(q)$
- $\forall p, q \in Q, a \in A,$

$$\hat{t}(p)(a)(i_{p,a,q}) = \begin{cases} 1 & \text{if } t(p)(a)(q) \neq 0 \\ 0 & \text{otherwise} \end{cases}$$

$\forall p, q \in Q, a, b \in A, r, s \in Q_{p,b,q},$

$$\hat{t}(r)(a)(s) = \begin{cases} t_{p,b,q}(r)(a)(s) + o_{p,b,q}(r) & \text{if } p = q, a = b \text{ and } s = i_{p,a,p} \\ t_{p,b,q}(r)(a)(s) & \text{otherwise} \end{cases}$$

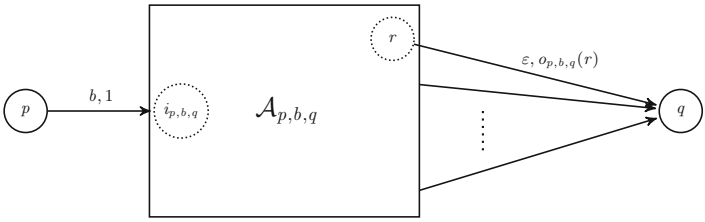
$\forall a, b \in A, p, q, q' \in Q$ s.t. $(p, b, q) \neq (q, a, q'), \forall r \in Q_{p,b,q},$

$$\hat{t}(r)(a)(i_{q,a,q'}) = o_{p,b,q}(r)$$

In all other cases, $\hat{t}(r)(a)(s) = 0$.

This construction corresponds to the intuition we gave before (expressed as in the "second idea", though both methods lead to the same automaton). In fact, consider a transition $p \xrightarrow{b,\sigma} q$ in $\mathcal{A}$. We replace it by connecting $\mathcal{A}_{p,b,q}$

between p and q as we did in the example, using ε-transitions: we set a transition $p \xrightarrow{b,1} i_{p,b,q}$ and for each final state $r \in Q_{p,b,q}$ we set a transition $r \xrightarrow{\varepsilon, o_{p,b,q}(r)} q$, and set the output of r to 0.



(There are no paths labeled by ε of length > 1, hence the semantics of the new automaton is well-defined.) We can then proceed to the removal of the ε-transitions : for every transition $r \xrightarrow{\varepsilon, o_{p,b,q}(r)} q$, we do the following : we remove the transition, we set $o(r) := o(r) + o_{p,b,q}(r)o(q)$, and for every $q \xrightarrow{a,s} q'$ we add a transition $r \xrightarrow{a, o_{p,b,q}(r)\cdot s} q'$. Note that the only such transitions $q \xrightarrow{a,s} q'$ are transitions of the form $q \xrightarrow{a,1} i_{q,a,q'}$.

Theorem 5. *With the above notations, denote by $\mathcal{S}$ the semantics of the automaton $\mathcal{A}$, and by $\hat{\mathcal{S}}$ the semantics of $\hat{\mathcal{A}}$. Then for all $q \in Q, \mathcal{S}(q) = \hat{\mathcal{S}}(q)$.*

Proof. It is enough to show that

$$\mathcal{R} = \left\{ \left(\hat{\mathcal{S}}(q), \mathcal{S}(q) \right) \mid q \in Q \right\} \cup \left\{ \left(\hat{\mathcal{S}}(r), \mathcal{S}_{p,a,q}(r) S(q) \right) \mid p, q \in Q, r \in Q_{p,a,q} \right\}$$

is a bisimulation-up-to. (For all p, q, $\mathcal{S}_{p,a,q}$ denotes the semantics of $\mathcal{A}_{p,a,q}$.) □

Remark. Theorem 5 holds without any restriction on $\mathcal{A}$. Now consider the case where $\mathcal{A}$ is finite, and for all $p, q \in Q$, $t(p)(a)(q)$ is a rational power series. Then we can suppose that all $\mathcal{A}_{p,a,q}$ are also finite, and we obtain for $\hat{\mathcal{A}}$ a finite automaton as well. In particular, for all q, $\hat{\mathcal{S}}(q)$ is rational, i.e. $\mathcal{S}(q)$ is rational. This gives us a proof that (not surprisingly) finite HWAs with rational weights have the same expressivity as WAs. Yet there are other ways to prove this, for instance by directly computing $\mathcal{S}(q)$ as in Sect. 5.1.

5 Some Applications of Heavy-Weighted Automata

Heavy-weighted automata provide a more compact way of representing power series than ordinary weighted automata. We give two examples of how this can be used. First, there is the *state elimination method*, that describes a way to remove a state in a weighted automaton. In the case of finite automata, it also leads to an algorithm to compute a rational expression for the power series recognised by some state of the automaton. Secondly, we consider the case of infinite weighted automata representing algebraic power series. Under precise conditions on the shape of the automaton, we can formulate some contraction rules that lead to an equivalent, possibly finite, HWA.

5.1 State Elimination Method

Brzozowski and McCluskey's state elimination method for computing the rational expression associated to an (ordinary) finite automaton can easily be adapted to weighted automata. The only thing new with weighted automata is that we need to update also the outputs of the remaining states when we remove a state.

For practical reasons, we adopt in this subsection a slightly different definition of HWAs than in the rest of the paper. We now allow not only the weight of the transitions, but also the outputs of the states to be power series. Furthermore, we choose to define HWAs by giving their cumulated weight function w rather than t (see Sect. 3). Formally, a *heavy-weighted automaton* now is a pair $(Q, \langle o, w\rangle)$, where Q is a set of states, $o : Q \to S\langle\langle A\rangle\rangle$ is the output function, and $w : Q \to (Q \to_f S\langle\langle A\rangle\rangle_p)$.

The *behaviours* $\mathcal{S}(q)$ of each state $q \in Q$ are again defined as the unique solutions of a system of equations: for all $q \in Q$,

$$\mathcal{S}(q) = o(q) + \sum_{r \in Q} w(p)(q) \times \mathcal{S}(r).$$

Note that such an automaton can always be transformed into an automaton in which the output of all states are elements of S: we add one state f, with no outgoing transitions and output 1. For each other state $q \in Q$, we decompose $o(q)$ into $o(q) = s + \sigma$, with $s = O(o(q))$ and $\sigma = \sum_{a \in A} a \times o(q)_a$. Then we replace the ouptut of q by s, and we add a transition $q \xrightarrow{\sigma} f$. (Fig. 2).

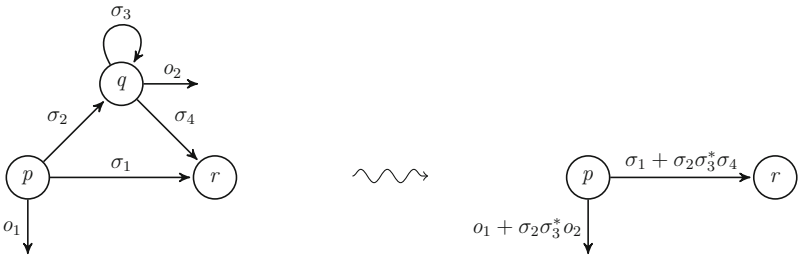


Fig. 2. Elimination of state q

State Elimination. Let $\mathcal{A} = (Q, \langle o, w\rangle)$ be a HWA with at least two states, and $q \in \mathcal{A}$. We define the automaton $\mathsf{elimination}(q, \mathcal{A})$ resulting from the elimination of state q in $\mathcal{A}$ as $\mathsf{elimination}(q, \mathcal{A}) = (Q \setminus \{q\}, \langle \hat{o}, \hat{w}\rangle)$, with for all $p, r \in Q$,

$$\hat{o}(p) = o(p) + w(p)(q) \times w(q)(q)^* \times o(q)$$
$$\hat{w}(p)(r) = w(p)(r) + w(p)(q) \times w(q)(q)^* \times w(q)(r).$$

We denote by $\mathcal{S}(p)$ the behaviour of a state $p \in Q$ in the automaton $\mathcal{A}$, and for $p \neq q$, we denote by $\hat{\mathcal{S}}(p)$ its behaviour in the automaton $\mathsf{elimination}(q, \mathcal{A})$.

Proposition 2. *For all $p \in Q \setminus \{q\}$, $\hat{\mathcal{S}}(p) = \mathcal{S}(p)$.*

Proof. $\mathcal{S}$ is defined by the following system of equations:

$$\forall p \in Q \qquad \mathcal{S}(p) = o(p) + \sum_{r \in Q} w(p)(r) \times \mathcal{S}(r)$$

The equation for q is equivalent to

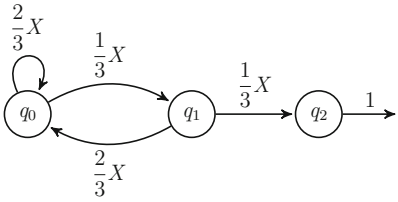
$$\mathcal{S}(q) = w(q)(q)^* \left(o(q) + \sum_{r \in Q \setminus \{q\}} w(q)(r)\mathcal{S}(r) \right).$$

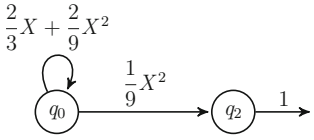
Substituting into the other equations, we get exactly the system defining $\hat{\mathcal{S}}$. □

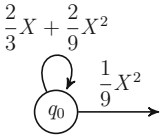
Computing Rational Expressions for Finite Weighted Automata. The previous result holds without further restriction on $\mathcal{A}$. We now consider the case where Q is finite, and where all weights and outputs in $\mathcal{A}$ are given by rational expressions. Clearly, the weights and outputs in $\mathsf{eliminate}(q, \mathcal{A})$ can again be given by rational expressions, so these assumptions are preserved at each elimination of a state. In what follows, we identify rational expressions and the power series they denote.

Suppose we start from a finite ordinary weighted automaton. To compute the behaviour of a state q, we can eliminate successively all the other states of the automaton. We get an automaton with only one state q, the behaviour of which is given by the equation $\mathcal{S}(q) = o(q) + w(q)(q)\mathcal{S}(q)$, which leads to $\mathcal{S}(q) = w(q)(q)^* \times o(q)$. Since $o(q)$ and $w(q)(q)$ are given by rational expressions, this gives us a rational expression for $\mathcal{S}(q)$.

Example. Consider the automaton on the right. Removing successively q_1 and q_2 leads to:







Finally, we get $\mathcal{S}(q_0) = \left(\frac{2}{3}X + \frac{2}{9}X^2\right)^* \times \frac{1}{9}X^2$.

5.2 Reduction of Well-Shaped Infinite Weighted Automata

Most of what is known about weighted automata concerns finite automata. Yet some situations can conveniently be represented by an infinite weighted automaton, in a rather natural way.

For instance, in [4,10], infinite weighted automata are extensively used to model counting problems. The idea is to give a weighted automata recognizing

the generating function of the family of combinatorial objects studied, that is, the power series $\sum_{n\in\mathbb{N}} f_n X^n$, where f_n denotes the number of objects of size n. Typically, such a generating function is represented by an infinite weighted automaton in which most transitions have weight 1, and which is constructed in such a way that each object of size n correspond precisely to one accepting path of length n in the automaton.

Example. One of the main examples given in [4] is the study of Motzkin paths. A Motzkin path of length n is a lattice path of $\mathbb{Z}\times\mathbb{Z}$ going from $(0,0)$ to $(n,0)$, that never passes below the x-axis and whose permitted steps are the up diagonal step $(1,1)$, the down diagonal step $(1,-1)$ and the level step $(1,0)$.

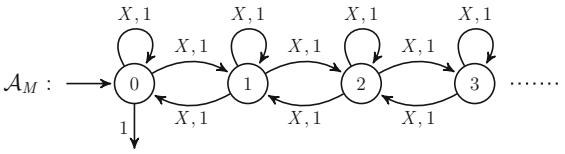


Fig. 3. Weighted automaton for Motzkin paths.

The number Motzkin paths of length n, called the n-th Motzkin number, is given by the behaviour of the automaton $\mathcal{A}_M$ (see Fig. 3); more precisely, it is equal to $\mathcal{S}(0)(n)$. Intuitively, a transition $i \to (i+1)$ corresponds to a step $(1,1)$, a transition $(i+1) \to i$ to a step $(1,-1)$, and a transition $i \to i$ to a step $(1,0)$. A Motzkin path starts and ends at level $y=0$, which is why 0 is both the only initial and final state.

In this example as in many others, the automaton that interests us is constructed by repeating the same pattern infinitely often. Our aim is to use these regularities to contract some part of the automaton, and compute an equivalent finite HWA when it is possible.

Example. The automaton in Fig. 3 is equivalent to the automaton to the right. The idea is the following. Consider an accepting path starting in state 1, that is, a path $1 = q_0 \to q_1 \to \cdots \to q_n = 0$. Let k be the smallest index strictly greater than 0 such that $q_k = 0$. Necessarily, $q_{k-1} = 1$, and $q_0 \to \cdots \to q_{k-1}$ is a path from 1 to 1 that never passes through 0: there are as many such paths as there are paths from 0 to 0 of length $k-1$, i.e. $\mathcal{S}(0)(k-1) = X \times \mathcal{S}(0)(k)$. This is what is expressed by $1 \xrightarrow{X,\mathcal{S}(0)} 0$ in $\hat{\mathcal{A}}_M$.

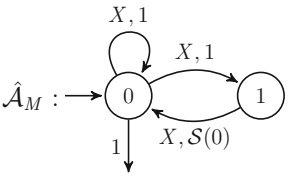


Fig. 4. Simplified automaton for Motzkin paths.

Formally, we can prove that the two automata are equivalent as follows. Denote by $\hat{\mathcal{S}}(i)$ the semantics of state i in $\hat{\mathcal{A}}_M$. We can check that the closure under linear combinations of the relation

$$\mathcal{R} = \left\{ \left(\mathcal{S}(0), \hat{\mathcal{S}}(0)\right), \left(\mathcal{S}(1), \hat{\mathcal{S}}(1)\right) \right\} \cup \left\{ \left(\mathcal{S}(i), X \times \mathcal{S}(i-1) \times \hat{\mathcal{S}}(0)\right) \mid i \geq 1 \right\}$$

is a bisimulation; hence $\mathcal{S}(0) = \hat{\mathcal{S}}(0)$ and $\mathcal{S}(1) = \hat{\mathcal{S}}(1)$.

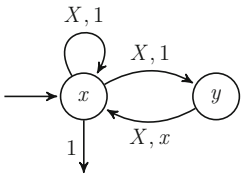
From the automaton $\hat{\mathcal{A}}_M$, we then get the following equation for $\mathcal{S}(0)$:

$$\mathcal{S}(0) = 1 + X \times \mathcal{S}(0) + X^2 \times \mathcal{S}(0)^2.$$

A few other examples of such reductions are given in [4]. Our aim here is to identify specific classes for which such reductions are possible, and to give a few general rules applicable in these situations. The key point in the reduction we did for $\mathcal{A}_M$ was the fact that the sub-automaton consisting of the states $\{1, 2, \ldots\}$ is isomorphic to $\mathcal{A}_M$. Similarly, the rules we will give apply to automata where some sub-automaton is isomorphic to the whole automaton.

Syntactic Heavy-Weighted Automata. In the automaton $\hat{\mathcal{A}}_M$ (see Fig. 4), the stream $\mathcal{S}(0)$ appearing in the transition from 1 to 0 refers to the behaviour of the state 0 of the automaton $\mathcal{A}_M$ (see Fig. 3), and not $\hat{\mathcal{A}}_M$ itself. A priori, we can't refer to the behaviour of an automaton in the definition of its transition function, since its behaviour is itself defined using the transition function.

Yet $\hat{\mathcal{A}}_M$ is precisely constructed so as to have $\mathcal{S}(0) = \hat{\mathcal{S}}(0)$, and we would like to be able to define $\hat{\mathcal{A}}_M$ without refering to $\mathcal{A}_M$. To allow such definitions, we introduce a new kind of automata: *syntactic heavy-weighted automata*. To simplify notations, we will identify the behaviour of a state with this state itself. For instance, the syntactic heavy-weighted automaton corresponding to $\hat{\mathcal{A}}_M$ will be the one on the right.



A *syntactic heavy-weighted automata* (or SHWA, for short) over a semiring S and an alphabet A consists of a pair $(Q, \langle o, t\rangle)$, where Q is the set of states, $o : Q \to S$ is the output function, and $t : Q \to (Q \to_f S\langle A \cup Q\rangle)^A$ is the transition function.

The *behaviour* $\mathcal{S}(x)$ of a state $x \in Q$ is defined as the unique solutions to the following system of behavioural differential equations: for all $x \in Q$ and $a \in A$,

$$O(x) = o(x) \qquad x_a = \sum_{y \in Q} t(x)(a)(y) \times y. \tag{1}$$

We extend $\mathcal{S} : Q \to S\langle\langle A\rangle\rangle$ to polynomials and power series as follows. We first define $\mathcal{S}$ inductively on words over $(A \cup Q)$: for all $a \in A$, $x \in Q$ and $u \in A^*$,

$$\mathcal{S}(\varepsilon) = 1 \qquad \mathcal{S}(a \cdot u) = a \times \mathcal{S}(u) \qquad \mathcal{S}(x \cdot u) = \mathcal{S}(x) \times \mathcal{S}(u).$$

For all $\sigma \in S\langle\langle A \cup Q\rangle\rangle$ and $w \in A^*$, we then define $\mathcal{S}(\sigma) = \sum_{w \in A^*} \sigma(w)\mathcal{S}(w)$. For instance, we have $\mathcal{S}(3aq + qr) = 3a\mathcal{S}(q) + \mathcal{S}(q)\mathcal{S}(r)$. The fact that $\{\mathcal{S}(x) \mid x \in Q\}$ is the solution to the system of behavioural differential equations in (1) can then be written as follows: for all $x \in Q$ and $a \in A$,

$$O(\mathcal{S}(x)) = o(x) \qquad \mathcal{S}(x)_a = \sum_{y \in Q} \mathcal{S}(t(x)(a)(y)) \times \mathcal{S}(y).$$

First Reduction Rule. We describe a method to remove a (possibly infinite) set of states from a SHWA $\mathcal{A}$, under precise assumptions. We proceed in two steps: first, we show how to disconnect a subset Q' of the states of $\mathcal{A}$ from the rest

of the states (Proposition 3), and then we show that when the sub-automaton induced by Q' is isomorphic to $\mathcal{A}$, we can remove it entirely (Corollary 1).

Let $\mathcal{A} = (Q, \langle o, t\rangle)$ be a SHWA, and $Q' \subsetneq Q$. Let

$$F' = \{q \in Q' \mid \exists r \in Q \setminus Q',\ \exists a \in A,\ t(q)(a)(r) \neq 0\},$$

and

$$I' = \{q \in Q' \mid \exists r \in Q \setminus Q',\ \exists a \in A,\ t(r)(a)(q) \neq 0\}.$$

We assume that Q' satisfies the following conditions:

(1) For all $q \in Q'$, $o(q) = 0$.
(2) For all $p, q \in Q$ and $a \in A$, $t(p)(a)(q) \in S\langle A \cup (Q \setminus Q')\rangle$.
(3) There exists $o' : Q \to S$ and $f : A \to S\langle A \cup (Q \setminus Q')\rangle^{Q \setminus Q'}$ such that:
   (a) for all $q \in Q'$, $(o'(q) \neq 0 \iff q' \in F')$.
   (b) for all $q \in Q'$ and $r \in Q \setminus Q'$, $t(q)(a)(r) = o'(q) \times f(a)(r)$.

If F' contains only one state q, condition (3) becomes useless, and we simply set $o'(q) = 1$ and $f(a)(r) = t(q)(a)(r)$.

Define $\hat{\mathcal{A}} = \left(Q, \langle \hat{o}, \hat{t}\rangle\right)$ as follows:

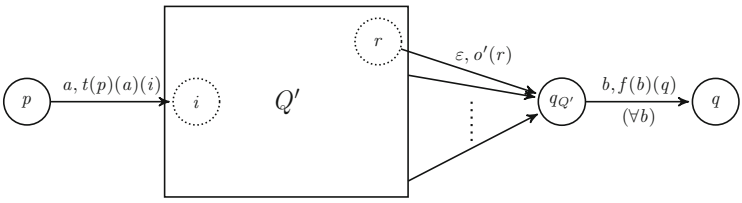
– $\hat{o}(q) = \begin{cases} o(q) & \text{if } q \in Q \setminus Q' \\ o'(q) & \text{if } q \in Q' \end{cases}$

– $\hat{t}(p)(a)(q) = \begin{cases} t(p)(a)(q) + \sum_{r \in Q'} t(p)(a)(r) \times r \\ \quad \times \sum_{b \in A} b \times f(b)(q) & \text{if } p, q \in Q \setminus Q' \\ t(p)(a)(q) & \text{if } p, q \in Q' \\ 0 & \text{otherwise.} \end{cases}$

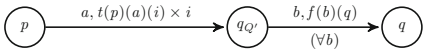
We denote by $\hat{\mathcal{S}}(q)$ the behaviour of a state $q \in Q$ in automaton $\hat{\mathcal{A}}$, and by $\mathcal{S}(q)$ its behaviour in automaton $\mathcal{A}$.

Proposition 3. *Under the above assumptions* $\hat{\mathcal{S}}(q) = \mathcal{S}(q)$ *for all* $q \in Q \setminus Q'$.

The idea is that this construction is the opposite of the construction that we did in Sect. 4. We recognize something of the form



and we contract it into the one on the right which, after removal of $q_{Q'}$ (as in Sect. 5.1), leads for all $a \in A$ to a transition $p \xrightarrow{a, t(p)(a)(i) \times i \times \sum_b b f(b)(q)} q$.



Proof (of Proposition 3). Let $C = \sum_{a\in A}\sum_{r\in Q\setminus Q'} a \times \mathcal{S}(f(a)(r)) \times \mathcal{S}(r)$. Then $\{(\mathcal{S}(q), \hat{\mathcal{S}}(q)) \mid q \in Q \setminus Q'\} \cup \{(\mathcal{S}(q), \hat{\mathcal{S}}(q) \times C) \mid q \in Q'\}$ is a bisimulation-up-to. □

Proposition 3 allows us to isolate the sub-automaton of $\mathcal{A}$ obtained by keeping only the states in Q', and setting as final those states that have an outgoing transition leaving Q'. More interestingly, when this sub-automaton is isomorphic to $\mathcal{A}$ itself, we can remove all the states in Q', as follows.

Corollary 1. *Assume that there exists a bijection $\varphi : Q' \to Q$ such that:*

- $\forall q \in Q',\ o'(q) = o(\varphi(q))$
- $\forall p, q \in Q,\ t(\varphi(p))(a)(\varphi(q)) = t(p)(a)(q)$
- $\forall q \in I',\ \varphi(q) \in Q \setminus Q'$

We define $\bar{\mathcal{A}} = (Q \setminus Q', \langle \bar{o}, \bar{t}\rangle)$ as follows: $\bar{o}$ is the restriction of o to $Q \setminus Q'$, and for all $p, q \in Q \setminus Q'$,

$$\bar{t}(p)(a)(q) = t(p)(a)(q) + \sum_{r\in Q'} t(p)(a)(r) \times \varphi(r) \times \sum_{b\in A} bf(b)(q).$$

Then $\bar{\mathcal{S}}(q) = \mathcal{S}(q)$ for all $q \in Q \setminus Q'$.

Proof. $\bar{t}$ is well-defined, because of condition (2) and the assumption that for all $q \in I', \varphi(q) \in Q \setminus Q'$. To prove the equality, we show successively that $\mathcal{R}_1 = \left\{\left(\mathcal{S}(\varphi(q)), \hat{\mathcal{S}}(q)\right) \mid q \in Q'\right\}$ and $\mathcal{R}_2 = \left\{\left(\bar{\mathcal{S}}(q), \hat{\mathcal{S}}(q)\right) \mid q \in Q \setminus Q'\right\}$ are bisimulations-up-to. Hence $\bar{\mathcal{S}}(q) = \hat{\mathcal{S}}(q) = \mathcal{S}(q)$ for all $q \in Q \setminus Q'$. □

Example. Consider again the automaton $\mathcal{A}_M$. Taking $Q' = \{q_1, q_2, \ldots\}$ and $\varphi(q_i) = q_{i-1}$, we obtain the automaton $\bar{\mathcal{A}}_M$, which can be also be obtained from $\hat{\mathcal{A}}_M$ in Fig. 4 by removing state 1.

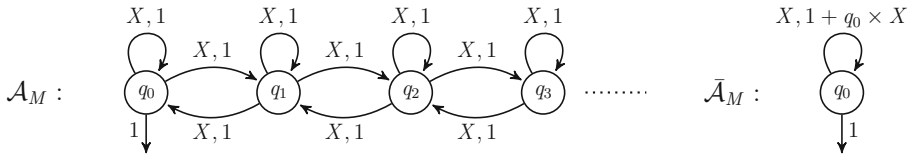


Fig. 5. Application of the first reduction rule to $\mathcal{A}_M$

Second Reduction Rule. Condition (1) in the previous reduction rule is quite restrictive, even when being interested only in specific, well-shaped automata. To gain a little more generality, we present a second reduction rule, which consists not in removing states, but in transforming some final states into non final states.

Let $\mathcal{A} = (q, \langle o, t\rangle)$ be a SHWA, and $Q' \subsetneq Q$. We define as before $I' = \{q \in Q' \mid \exists r \in Q,\ \exists a \in A,\ t(r)(a)(q) \neq 0\}$, but this time we set $F' = \{q \in Q' \mid o(q) = 0\}$.

Suppose that there exists a bijection $\psi : Q' \to Q$ such that:

(1) For all $q, r \in Q'$, $a \in A$, $t(p)(a)(q) = t(\psi(p))(a)(\psi(q))$ and $o(q) = o(\psi(q))$
(2) For all $i \in I'$, $\psi(i) \in Q \setminus Q'$.

and that

(3) For all $p, q \in Q$ and $a \in A$, $t(p)(a)(q) \in S\langle A \cup (Q \setminus Q')\rangle$

We define $\hat{\mathcal{A}} = \big(Q, \langle \hat{o}, \hat{t}\rangle\big)$ as follows: for all $p, q \in Q$ and $a \in A$,

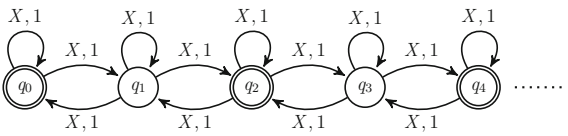
$$\hat{o}(q) = \begin{cases} o(q) & \text{if } q \in Q \setminus Q' \\ 0 & \text{if } q \in Q' \end{cases}$$

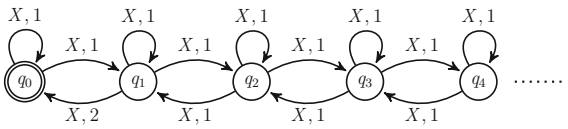
$$\hat{t}(p)(a)(q) = \begin{cases} t(p)(a)(q) + t(p)(a)(\psi^{-1}(q)) & \text{if } p, q \in Q \setminus Q' \\ t(p)(a)(q) & \text{otherwise.} \end{cases}$$

Proposition 4. *Under the above assumptions, denote by $\mathcal{S}$ the behaviour of $\mathcal{A}$, and by $\hat{\mathcal{S}}$ the behaviour of $\hat{\mathcal{A}}$. Then $\hat{\mathcal{S}}(q) = \mathcal{S}(q)$ for all $q \in Q \setminus Q'$.*

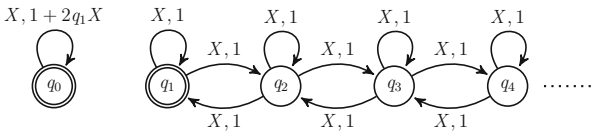
Proof. $\big\{(\mathcal{S}(q), \hat{\mathcal{S}}(q)) \mid q \in Q \setminus Q'\big\} \cup \big\{(\mathcal{S}(q), \hat{\mathcal{S}}(q) + \mathcal{S}(\psi(q))) \mid q \in Q'\big\}$ is a bisimulation-up-to. □

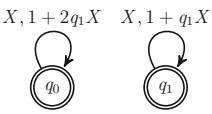
Example. Consider the automaton on the right (taken from [4]), where all final states have output 1. Taking $Q' = \{q_i \mid i \geq 2\}$ and $\psi(q_i) = q_{i-2}$, we obtain:





Using Proposition 3, q_0 has the same behaviour in the automaton on the right. And combining this with the reduction shown in Fig. 5, we get:





The behaviour of q_0 is then described by:

$$O(\mathcal{S}(q_0)) = 1 \qquad \mathcal{S}(q_0)' = \mathcal{S}(q_0) + 2\mathcal{S}(q_1)X\mathcal{S}(q_0)$$
$$O(\mathcal{S}(q_1) = 1 \qquad \mathcal{S}(q_1)' = \mathcal{S}(q_1) + \mathcal{S}(q_1)X\mathcal{S}(q_1).$$

Link with Algebraic Power Series. All the examples we treated in this section are *algebraic* (or *context-free*) power series. More generally, SHWAs can be seen as a representation of polynomial systems of equations. In the finite case, the solution to such a system of equations is an algebraic power series.

A *polynomial system of behavioural differential equations* over a semiring S, an alphabet A, and a set of variables $\mathcal{X}$ consists of a set of equations (one for each $x \in \mathcal{X}$) of the form

$$x_a = t \qquad O(x) = s$$

where $t \in S\langle A \cup \mathcal{X}\rangle$ and $s \in S$. A polynomial system of behavioural differential equations is *finite* if $\mathcal{X}$ is finite.

A polynomial system of behavioural differential equations always has a unique solution. A formal power series $\sigma \in S\langle\langle A\rangle\rangle$ is called *algebraic* when it is part of the solution of a *finite* polynomial system of behavioural differential equations.

This coinductive characterization of algebraic power series is equivalent to other notions of algebraic or context-free power series [7], as shown in [2].

Proposition 5. *Let $\sigma \in S\langle\langle A\rangle\rangle$. Then σ is algebraic if and only if there exists a finite* SHWA $\mathcal{A} = (Q, \langle o, t\rangle)$ *and $q_0 \in Q$ such that $\mathcal{S}(q_0) = \sigma$.*

6 Conclusion

We studied an extension of weighted automata that allows the weights of the transitions to be any power series in $S\langle\langle A\rangle\rangle$, rather than elements of S. The semantics of a heavy weighted automaton can be given by a system of behavioural differential equations linking the behaviours of the different states, or by transforming the automaton into an $S \times (-)^A$-coalgebra and applying the final $S \times (-)^A$-homomorphism. Moreover, any heavy weighted automaton can be transformed into a weighted automaton in a canonical way.

Heavy weighted automata often provide a more compact representation of a power series than weighted automata. In particular, they can be used to compute a regular expression associated with a finite weighted automata, or in some cases to give a finite representation of an infinite weighted automata. The state elimination method can be used to remove one state at a time, and in some special cases, see Sect. 5.2, allow to remove an infinite subset of states.

References

1. Bonchi, F., Bonsangue, M.M., Boreale, M., Rutten, J.J.M.M., Silva, A.: A coalgebraic perspective on linear weighted automata. Inf. Comput. **211**, 77–105 (2012)
2. Bonsangue, M.M., Rutten, J., Winter, J.: Defining context-free power series coalgebraically. In: Pattinson, D., Schröder, L. (eds.) CMCS 2012. LNCS, vol. 7399, pp. 20–39. Springer, Heidelberg (2012)
3. Brzozowski, J., Mccluskey, E.J.: Signal flow graph techniques for sequential circuit state diagrams. IEEE Trans. Electron. Comput. **12**(2), 67–76 (1963)
4. Castro, R.D., Ramírez, A., Ramírez, J.L.: Applications in enumerative combinatorics of infinite weighted automata and graphs. Sci. Annal. Comput. Sci. **24**(1), 137–171 (2014)

5. Droste, M., Kuich, W., Vogler, H. (eds.): Handbook of Weighted Automata. Monographs in Theoretical Computer Science. An EATCS Series, 1st edn. Springer, Heidelberg (2009)
6. Fortin, M., Bonsangue, M.M., Rutten, J.J.M.M.: Coalgebraic semantics of heavy-weighted automata. Technical report FM-1405, CWI - Amsterdam (2014). http://oai.cwi.nl/oai/asset/22603/22603D.pdf
7. Petre, I., Salomaa, A.: Algebraic systems and pushdown automata. In: Droste, M., Kuich, W., Vogler, H. (eds.) Handbook of Weighted Automata [5]. Monographs in Theoretical Computer Science. An EATCS Series, pp. 257–289. Springer, Heidelberg (2009)
8. Rot, J., Bonsangue, M., Rutten, J.: Coalgebraic bisimulation-up-to. In: van Emde Boas, P., Groen, F.C.A., Italiano, G.F., Nawrocki, J., Sack, H. (eds.) SOFSEM 2013. LNCS, vol. 7741, pp. 369–381. Springer, Heidelberg (2013)
9. Rutten, J.J.M.M.: Behavioural differential equations: a coinductive calculus of streams, automata, and power series. Theoret. Comput. Sci. **308**(1–3), 1–53 (2003)
10. Rutten, J.J.M.M.: Coinductive counting with weighted automata. J. Automata Lang. Comb. **8**(2), 319–352 (2003)
11. Rutten, J.J.M.M.: A coinductive calculus of streams. Math. Struct. Comput. Sci. **15**(1), 93–147 (2005)
12. Sakarovitch, J.: Elements of Automata Theory. Cambridge University Press, New York (2009)
13. Silva, A., Bonchi, F., Bonsangue, M.M., Rutten, J.J.M.M.: Generalizing determinization from automata to coalgebras. Log. Methods Comput. Sci. **9**(1) (2013)
14. Wood, D.: Theory of Computation. Harper & Row, New York (1987)