Scientific Annals of Computer Science vol. 27 (2), 2017, pp. 137–176
doi: 10.7561/SACS.2017.2.137

Derived-Term Automata of Multitape Expressions with Composition*

Akim DEMAILLE[1]

Abstract

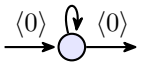
Rational expressions are powerful tools to define automata, but often restricted to single-tape automata. Our goal is to unleash their expressive power for transducers, and more generally, any multitape automaton; for instance $(a^+|x+b^+|y)^*$. We generalize the construction of the derived-term automaton by using *expansions*. This approach generates small automata, and even allows us to support a composition operator.

Keywords: Antimirov, automaton, derivatives, derived term, expansion, extended rational expressions, transducer, multitape, composition

1 Introduction

To compute the edit-distance between words and/or (rational) languages, Mohri [20, Figure 4] introduces the following two-tape automaton (aka transducer) $\mathcal{A}$, whose weights, written in angle brackets, are in $\langle \mathbb{N}, \min, + \rangle$:

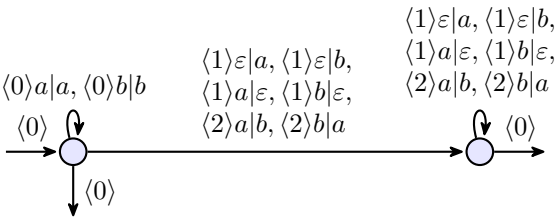
$$\langle 0\rangle a|a, \langle 0\rangle b|b, \langle 1\rangle \varepsilon|a, \langle 1\rangle \varepsilon|b, \langle 1\rangle a|\varepsilon, \langle 1\rangle b|\varepsilon$$



Ng [21, Figure 2] focuses on the *prefix* distance and introduces $\mathcal{A}'$:

*Extended version of *Derived-term automata of multitape rational expressions*, presented at CIAA'16 [9].

[1]EPITA Research and Development Laboratory (LRDE), 14-16, rue Voltaire, 94276 Le Kremlin-Bicêtre, France, E-mail: akim@lrde.epita.fr. (2017-11-27 19:48:50 +0100 6e2f2c2)



Such automata are tedious to type in. Rational expressions have already proved being extremely concise and handy tools to define automata, but rational expressions for multitape automata have received little attention. Yet an expression such as $(a|a + b|b + \langle 1\rangle(1|a + 1|b + a|1 + b|1))^*$ clearly denotes the behavior of $\mathcal{A}$.[1] Provided that operators such as $+$ can be used below the tupling operator $|$, an more concise expression is $\mathsf{E} := (a|a + b|b + \langle 1\rangle(1|(a + b) + (a + b)|1))^*$.[2] Similarly $\mathsf{E}' := (a|a + b|b)^*(\langle 1\rangle(1|(a + b) + (a + b)|1) + \langle 2\rangle(a|b + b|a))^*$ denotes the behavior of $\mathcal{A}'$.[3]

Our purpose is to define multitape rational expressions such as E and E' (Section 2) and to introduce an algorithm that computes precisely automata $\mathcal{A}$ and $\mathcal{A}'$ from them (Section 4). To this end, we rely on an intermediary structure, *expansions*, studied in Section 3.

Mohri [20] also shows (still in Figure 4) that $\mathcal{A}$ is equivalent to the composition of the following two simpler transducers, using two new symbols to denote *I*nsertion and *S*uppression[4]:

In Section 5 we introduce a composition operator @, such that E is equivalent to $(a|a + b|b + \langle 1\rangle(1|I + [ab]|S))^* \,@\, (a|a + b|b + I|[ab] + S|1)^*$, and extend our procedure to support it. It builds $\mathcal{A}$ from this expression.

Various aspects of our proposal are discussed in Section 6, and present related works in Section 7.

The contributions of this paper are:

[1] We use ε for the empty word, 1 for the expression that denotes the empty word, and we now leave the unit weight implicit; 0 in the current case.

[2] Linguists often simplify (partial) identities such as $(a|a) + (b|b)(c|c)^*$ as $a + bc^*$. With such conventions, we could even write $([a\text{–}z] + \langle 1\rangle(1|[a\text{–}z] + [a\text{–}z]|1))^*$.

[3] As a matter of fact, the $\langle 2\rangle(a|b + b|a)$ part is useless, as substitution is already scored as 1 + 1, one suppression, one insertion.

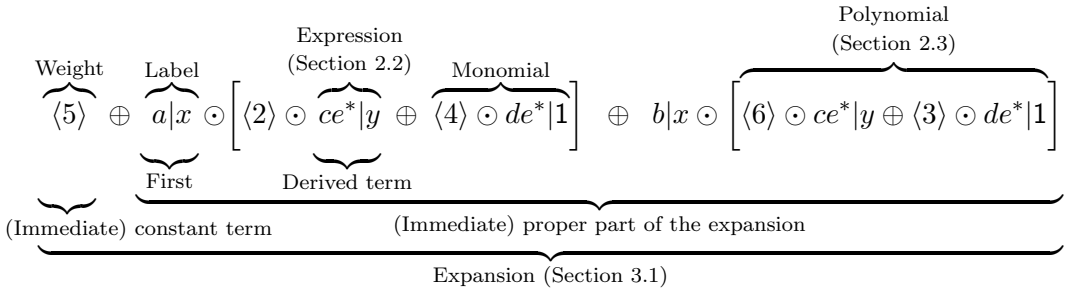
[4] In the paper, the second automaton shows transitions for $\langle 0\rangle S|a$ and $\langle 0\rangle S|b$ which were clearly not meant by the author.

- we define (weighted) multitape rational expressions featuring operators for tupling, $|$, and for composition, @;
- we provide an algorithm to build a concise automaton equivalent to such an expression. This algorithm is a generalization of the derived-term based algorithms, freed from the requirement that the monoid is free.

The constructs exposed in this paper are implemented in Vcsn[5]. Vcsn is a free-software platform dedicated to weighted automata and rational expressions [10]; its lowest layer is a C++ library, on top of which Python/IPython bindings provide an interactive graphical environment. The examples in this paper are demonstrated at `http://vcsn.lrde.epita.fr/dload/2.6/notebooks/SACS-2017.html`. They can be executed, modified, extended, directly from any web browser at `http://vcsn-sandbox.lrde.epita.fr`.

2 Notations

Our purpose is to define (weighted) multitape rational expressions, such as $\mathsf{E}_1 \coloneqq \langle 5\rangle 1|1 + \langle 4\rangle a\,d\,e^*|x + \langle 3\rangle b\,d\,e^*|x + \langle 2\rangle a\,c\,e^*|x\,y + \langle 6\rangle b\,c\,e^*|x\,y$ (weights are written in angle brackets). It relates ade with x, with weight 4. We introduce an algorithm to build a multitape automaton (aka *transducer*) from such an expression, e.g., Fig. 1. This algorithm relies on *rational expansions*. They are to the derivatives of rational expressions what differential forms are to the derivatives of functions. Defining expansions requires several concepts, defined bottom-up in this section. The following figure presents these different entities, how they relate to each other, and where we are heading to: given a weighted multitape rational expression such as E_1, compute *its* expansion:



from which we build its derived-term automaton (Fig. 1).

[5] Vcsn `http://vcsn.lrde.epita.fr`.

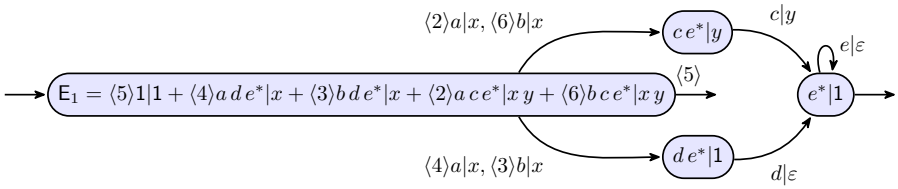


Figure 1: The derived-term automaton of E_1 (see Examples 1 to 4) with $\mathsf{E}_1 \coloneqq \langle 5\rangle 1|1 + \langle 4\rangle a\,d\,e^*|x + \langle 3\rangle b\,d\,e^*|x + \langle 2\rangle a\,c\,e^*|x\,y + \langle 6\rangle b\,c\,e^*|x\,y$.

It is helpful to think of expansions as a normal form for expressions.

2.1 Rational Series

Series will be used to define the semantics of the forthcoming structures: they are to weighted automata what languages are to Boolean automata. Not all languages are rational (denoted by an expression), and similarly, not all series are rational (denoted by a weighted expression). We follow Sakarovitch [24, Chap. III].

In order to cope with (possibly) several tapes, we cannot rely on the traditional definitions based on the free monoid A^* for some alphabet A.

2.1.1 Labels

Let M be a monoid (e.g., A^* or $A^* \times B^*$), whose neutral element is denoted ε_M —or ε when clear from the context— and named *the empty word*. For consistency with the way transducers are usually represented, we use $m \mid n$ rather than (m, n) to denote the pair of m and n. For instance $\varepsilon_{A^*\times B^*} = \varepsilon_{A^*} \mid \varepsilon_{B^*}$, and $\varepsilon_M \mid a \in M \times \{a\}^*$.

A *set of generators* G of M is a subset of M such that $G^* = M$. By $G^?$ we denote $\{\varepsilon\} \cup G$. A monoid M is of *finite type* (or *finitely generated*) if it admits a finite set of generators.

A monoid M is *graded* if it admits a *gradation* function $|\cdot| \in M \to \mathbb{N}$ such that $\forall m, n \in M$, $|m| = 0$ iff $m = \varepsilon$, and $|mn| = |m| + |n|$. Cartesian products of graded monoids are graded, and Cartesian products of finitely generated monoids are finitely generated. Free monoids and Cartesian products of free monoids are graded and finitely generated.

2.1.2 Weights

Let $\langle \mathbb{K}, +, \cdot, 0_{\mathbb{K}}, 1_{\mathbb{K}} \rangle$ (or $\mathbb{K}$ for short) be a semiring whose (possibly non commutative) multiplication will be denoted by juxtaposition. $\mathbb{K}$ is *commutative* if its multiplication is. $\mathbb{K}$ is a *topological semiring* if it is equipped with a topology, and both addition and multiplication are continuous. It is *strong* if the product of two summable families is summable.

2.1.3 Series

A (formal power) *series* over M with *weights* (or *multiplicities*) in $\mathbb{K}$ is a map from M to $\mathbb{K}$. The weight of $m \in M$ in a series s is denoted $s(m)$. The *null* series, $m \mapsto 0_{\mathbb{K}}$, is denoted 0; for any $m \in M$ (including ε_M), m denotes the series $u \mapsto 1_{\mathbb{K}}$ if $u = m, 0_{\mathbb{K}}$ otherwise. If M is of finite type, then we can define the Cauchy product of series. $s \cdot t \coloneqq m \mapsto \sum_{u,v \in M | uv = m} s(u) \cdot t(v)$. Equipped with the pointwise addition ($s + t \coloneqq m \mapsto s(m) + t(m)$) and $\cdot$ as multiplication, the set of these series forms a semiring denoted $\langle \mathbb{K}\langle\langle M \rangle\rangle, +, \cdot, 0, \varepsilon \rangle$.

The *constant term* of a series s, denoted s_ε, is $s(\varepsilon)$, the weight of the empty word. A series s is *proper* if $s_\varepsilon = 0_{\mathbb{K}}$. The *proper part* of s is the proper series s_p such that $s = s_\varepsilon + s_p$.

2.1.4 Star

The *star* of a series is an infinite sum: $s^* \coloneqq \sum_{n \in \mathbb{N}} s^n$. To ensure semantic soundness, we need M to be graded monoid and $\mathbb{K}$ to be a strong topological semiring.

We will need a property of star based on the following result. In various forms it is named the 'denesting rule' [15, p. 57], the 'property **S**' [24, Propositions III.2.5 and III.2.6], or the 'sum-star equation' [11, p. 188]. Proofs can be found for the axiomatic approach of star (based on Conway semirings), but we followed the topology-based one, for which we did not find a published version.

Proposition 1 (Super S) *Let $\mathbb{K}$ be a strong topological semiring. For any series $s, t \in \mathbb{K}\langle\langle A^* \rangle\rangle$, if s_ε^*, $(t_\varepsilon s_\varepsilon^*)^*$, and $(s_\varepsilon + t_\varepsilon)^*$ are defined and $(s_\varepsilon + t_\varepsilon)^* = s_\varepsilon^*(t_\varepsilon s_\varepsilon^*)^*$, then $(s + t)^* = s^*(ts^*)^*$.*

Proof: This proof goes in several steps, with different constraints over s and t. From a formal point of view, it is actually 'trivial': a simple look at

the proof of Sakarovitch [24, Proposition III.2.6] shows that both expressions are *formally* equivalent. The real technical difficulty is semantic: ensuring that all the (infinite) sums are properly defined.

We actually only need Item 4 below to establish Proposition 2.

1. *When s and t are proper.* This is a well-known consequence of Arden's lemma [24, Proposition III.2.5].

2. *When $s \in \mathbb{K}$, and t is proper.* This property holds when $\mathbb{K}$ is a strong topological semiring, and when s^* is defined [24, Proposition III.2.6].

3. *When $s, t \in \mathbb{K}$.* This result follows directly from the hypothesis of this property. Note however that $s^*(ts^*)^* = (s+t)^*$ is verified in all the 'usual' semirings.

   - If $\mathbb{K}$ is a usual numerical semiring (i.e., $\mathbb{Q}, \mathbb{R}$, or more generally, a subring of $\mathbb{C}^n$), then s^* is the inverse of $1-s$, i.e., $(1-s)s^* = s^*(1-s) = 1$. To establish the result, we show that $s^*(ts^*)^*$ is the inverse of $1-(s+t)$. By hypothesis, s^* and $(ts^*)^*$ are defined. $(1-(s+t))s^*(ts^*)^* = (1-s)s^*(ts^*)^* - ts^*(ts^*)^* = (ts^*)^* - ts^*(ts^*)^* = (1-ts^*)(ts^*)^* = 1$, which shows that $(s+t)^*$ is defined.
   - If $\mathbb{K}$ is a tropical semiring, say, $\langle \mathbb{Z} \cup \{\infty\}, \min, +, \infty, 0 \rangle$, then s^* is defined iff $s \geq 0$, and then $s^* = 0$, hence the result trivially follows.
   - If $\mathbb{K}$ is the Log semiring, $\langle \mathbb{R}^+ \cup \{\infty\}, +_{\text{Log}}, +, \infty, 0 \rangle$ where $+_{\text{Log}} \coloneqq x, y \mapsto -\log(\exp(-x) + \exp(-y))$. Then we get $x^* = \log(1 - \exp(-x))$. Again, one can verify the identity.

4. *When $s \in \mathbb{K}$ and t is any series.* By hypothesis, $(ts^*)^*$ is defined, i.e., $(t_\varepsilon s^*)^*$ is defined, so by Item 3, $(s+t_\varepsilon)^*$ is defined.

$$
\begin{aligned}
(s+t)^* &= (s + t_\varepsilon + t_p)^* \\
&= (s+t_\varepsilon)^*(t_p(s+t_\varepsilon)^*)^* && \text{by Item 2, } t_p \text{ proper, } (s+t_\varepsilon)^* \text{ defined} \\
&= s^*(t_\varepsilon s^*)^*(t_p s^*(t_\varepsilon s^*)^*)^* && \text{by Item 3} \\
&= s^*(t_\varepsilon s^* + t_p s^*)^* && \text{by Item 2, } t_p s^* \text{ proper, } (t_\varepsilon s^*)^* \text{ defined} \\
&= s^*((t_\varepsilon + t_p)s^*)^* \\
&= s^*(ts^*)^*
\end{aligned}
$$

5. *When s is any series and t is proper.* By hypothesis, s^* is defined, so s_ε^* is defined.

$$\begin{aligned}
(s+t)^* &= (s_\varepsilon + (s_p + t))^* \\
&= s_\varepsilon^*((s_p + t)s_\varepsilon^*)^* && \text{by Item 2, } s_p + t \text{ proper} \\
&= s_\varepsilon^*(s_p s_\varepsilon^* + ts_\varepsilon^*)^* \\
&= s_\varepsilon^*(s_p s_\varepsilon^*)^*(ts_\varepsilon^*(s_p s_\varepsilon^*)^*)^* && \text{by Item 1, } s_p s_\varepsilon^* \text{ and } ts_\varepsilon^* \text{ proper} \\
&= (s_\varepsilon + s_p)^*(t(s_\varepsilon + s_p)^*)^* && \text{by Item 2 } s_\varepsilon^* \text{ defined, } s_p \text{ proper} \\
&= s^*(ts^*)^*
\end{aligned}$$

6. *When s and t are any series.* By hypothesis, s^* is defined.

$$\begin{aligned}
(s+t)^* &= (s + t_\varepsilon + t_p)^* \\
&= (s + t_\varepsilon)^*(t_p(s + t_\varepsilon)^*)^* && \text{by Item 5, } t_p \text{ proper} \\
&= s^*(t_\varepsilon s^*)(t_p s^*(t_\varepsilon s^*)^*)^* && \text{by Item 4, } t_\varepsilon \in \mathbb{K} \\
&= s^*(t_\varepsilon s^* + t_p s^*)^* && \text{by Item 5, } t_p s^* \text{ proper} \\
&= s^*(ts^*)^* && \square
\end{aligned}$$

All the usual semirings ($\mathbb{B}, \mathbb{F}_2, \mathbb{Q}, \mathbb{R}, \mathbb{R}_{\min}, \mathrm{Log}$, etc.) are strong topological semirings, in which if s_ε^*, $(t_\varepsilon s_\varepsilon^*)^*$, and $(s_\varepsilon + t_\varepsilon)^*$ are defined then $(s_\varepsilon + t_\varepsilon)^* = s_\varepsilon^*(t_\varepsilon s_\varepsilon^*)^*$.

Proposition 2 *Let $\mathbb{K}$ be a strong topological semiring. Let $s \in \mathbb{K}, t \in \mathbb{K}\langle\langle A^* \rangle\rangle$, if s^*, $(t_\varepsilon s^*)^*$, and $(s + t_\varepsilon)^*$ are defined and $(s + t_\varepsilon)^* = s^*(t_\varepsilon s^*)^*$ then $(s + t)^* = s^* + s^* t(s + t)^*$.*

Proof: The result follows from Proposition 1, and from $(ts^*)^* = \varepsilon + (ts^*)(ts^*)^*$: $(s + t)^* = s^*(ts^*)^* = s^*(\varepsilon + (ts^*)(ts^*)^*) = s^* + s^* t(s^*(ts^*)^*) = s^* + s^* t(s + t)^*$. $\square$

2.1.5 Tuple

We suppose $\mathbb{K}$ is commutative. Let M and N be two monoids. The *tupling* of two series $s \in \mathbb{K}\langle\langle M \rangle\rangle, t \in \mathbb{K}\langle\langle N \rangle\rangle$, is the series $s \mid t \coloneqq m \mid n \in M \times N \mapsto s(m)t(n)$. It is a member of $\mathbb{K}\langle\langle M \times N \rangle\rangle$.

Proposition 3 (Series Tupling is Bilinear)
For all series $s, s' \in \mathbb{K}\langle\langle M \rangle\rangle$, $t, t' \in \mathbb{K}\langle\langle N \rangle\rangle$, and all weights $k \in \mathbb{K}$,

$$(s + s') \mid t = s \mid t + s' \mid t \qquad s \mid (t + t') = s \mid t + s \mid t'$$
$$(ks) \mid t = k(s \mid t) \qquad s \mid (kt) = k(s \mid t)$$

Proof: We prove the first equality. Let $m \mid n \in M \times N$. $((s+s') \mid t)(m \mid n) = (s + s')(m) \cdot t(n) = (s(m) + s'(m)) \cdot t(n) = s(m) \cdot t(n) + s'(m) \cdot t(n) = (s \mid t)(m \mid n) + (s' \mid t)(m \mid n) = (s \mid t + s' \mid t)(m \mid n)$. □

From now on, M is a graded monoid of finite type, and $\mathbb{K}$ a commutative strong topological semiring.

2.2 Weighted Rational Expressions

Contrary to the usual definition, we do not require a finite alphabet: any set of generators $G \subseteq M$ will do. For expressions with more than one tape, we require $\mathbb{K}$ to be commutative; however, for single tape expressions, our results apply to non-commutative semirings, hence there are two exterior products.

Definition 1 (Expression) *A* rational expression E *over G is a term built from the following grammar, where $a \in G$ denotes any non empty label, and $k \in \mathbb{K}$ any weight:*

$$\mathsf{E} ::= 0 \mid 1 \mid a \mid \mathsf{E} + \mathsf{E} \mid \langle k \rangle \mathsf{E} \mid \mathsf{E}\langle k \rangle \mid \mathsf{E} \cdot \mathsf{E} \mid \mathsf{E}^* \mid \mathsf{E} \mid \mathsf{E}$$

.

Expressions are syntactic; they are finite notations for (some) series.

Definition 2 (Series Denoted by an Expression) *Let E be an expression. The series denoted by E, noted $[\![\mathsf{E}]\!]$, is defined by induction on E:*

$$[\![0]\!] := 0 \qquad [\![1]\!] := \varepsilon \qquad [\![a]\!] := a \qquad [\![\mathsf{E} + \mathsf{F}]\!] := [\![\mathsf{E}]\!] + [\![\mathsf{F}]\!] \qquad [\![\langle k \rangle \mathsf{E}]\!] := k[\![\mathsf{E}]\!]$$
$$[\![\mathsf{E}\langle k \rangle]\!] := [\![\mathsf{E}]\!]k \quad [\![\mathsf{E} \cdot \mathsf{F}]\!] := [\![\mathsf{E}]\!] \cdot [\![\mathsf{F}]\!] \quad [\![\mathsf{E}^*]\!] := [\![\mathsf{E}]\!]^* \quad [\![\mathsf{E} \mid \mathsf{F}]\!] := [\![\mathsf{E}]\!] \mid [\![\mathsf{F}]\!]$$

An expression is *valid* if it denotes a series. More specifically, there are two requirements. First, the expression must be well-formed, i.e., concatenation and disjunction must be applied to expressions of appropriate number of tapes.

For instance, $a + b|c$ and $a(b|c|d)$ are ill-formed[6], $(a|b)^*|c + a|(b|c)^*$ is well-formed. Second, to ensure that $[\![\mathsf{F}]\!]^*$ is well defined for each subexpression of the form F^*, the constant term of $[\![\mathsf{F}]\!]$ must be *starrable* in $\mathbb{K}$ (Proposition 2).

Let $[n]$ denote $\{1, \dots, n\}$. The *size* (aka *length*) of a (valid) expression E, $|\mathsf{E}|$, is its total number of symbols, not counting parenthesis; for a given tape number $i \in [k]$ the *width on tape* i, $\|\mathsf{E}\|_i$, is the number of occurrences of labels on the tape i, the *width* of E (aka *literal length*), $\|\mathsf{E}\| \coloneqq \sum_{i\in[k]} \|\mathsf{E}\|_i$ is the total number of occurrences of labels different from ε.

Two expressions E and F are *equivalent* iff $[\![\mathsf{E}]\!] = [\![\mathsf{F}]\!]$. Some expressions are 'trivially equivalent', for instance $\mathsf{E} \cdot \varepsilon$ and E. To simplify the examples, our expressions will always be simplified according to the following identities.

Definition 3 (Trivial Identities) *Any subexpression of a form listed to the left of a '$\Rightarrow$' is rewritten as indicated on the right.*

$$\mathsf{E} + 0 \Rightarrow \mathsf{E} \qquad 0 + \mathsf{E} \Rightarrow \mathsf{E}$$
$$\langle 0_{\mathbb{K}}\rangle\mathsf{E} \Rightarrow 0 \quad \langle 1_{\mathbb{K}}\rangle\mathsf{E} \Rightarrow \mathsf{E} \quad \langle k\rangle 0 \Rightarrow 0 \quad \langle k\rangle\langle h\rangle\mathsf{E} \Rightarrow \langle kh\rangle\mathsf{E}$$
$$\mathsf{E}\langle 0_{\mathbb{K}}\rangle \Rightarrow 0 \quad \mathsf{E}\langle 1_{\mathbb{K}}\rangle \Rightarrow \mathsf{E} \quad 0\langle k\rangle \Rightarrow 0 \quad \mathsf{E}\langle k\rangle\langle h\rangle \Rightarrow \mathsf{E}\langle kh\rangle$$
$$(\langle k\rangle\mathsf{E})\langle h\rangle \Rightarrow \langle k\rangle(\mathsf{E}\langle h\rangle) \qquad \ell\langle k\rangle \Rightarrow \langle k\rangle\ell$$
$$\mathsf{E} \cdot 0 \Rightarrow 0 \qquad 0 \cdot \mathsf{E} \Rightarrow 0$$
$$(\langle k\rangle^?1) \cdot \mathsf{E} \Rightarrow \langle k\rangle^?\mathsf{E} \qquad \mathsf{E} \cdot (\langle k\rangle^?1) \Rightarrow \mathsf{E}\langle k\rangle^?$$
$$0^\star \Rightarrow 1$$
$$\mathsf{E} \mid 0 \Rightarrow 0 \qquad 0 \mid \mathsf{E} \Rightarrow 0 \qquad (\langle k\rangle^?\mathsf{E}) \mid (\langle h\rangle^?\mathsf{F}) \Rightarrow \langle kh\rangle^?\mathsf{E} \mid \mathsf{F}$$

where E is a rational expression, $\ell \in G \cup \{1\}$ a label, $k, h \in \mathbb{K}$ weights, and $\langle k\rangle^?\ell$ denotes either $\langle k\rangle\ell$, or ℓ in which case $k = 1_{\mathbb{K}}$ in the right-hand side of $\Rightarrow$.

These identities are taken from Sakarovitch [24]; they are discussed in Section 6.2. Note that *linearity* ('weighted ACI': associativity, commutativity and $\langle k\rangle\mathsf{E} + \langle h\rangle\mathsf{E} \Rightarrow \langle k + h\rangle\mathsf{E}$) is not enforced; this is the role of polynomials.

2.3 Rational Polynomials

At the core of the idea of 'partial derivatives' introduced by Antimirov [3], is that of *sets* of rational expressions, later generalized in *weighted sets* by

[6] As discussed in the introduction, an implementation could accept them, as abbreviations for $a|a + b|c$ and $(a|a|a)(b|c|d)$. But what sense could be given to $a|b + c|d|e$?

Lombardy and Sakarovitch [16], i.e., functions (partial, with finite domain) from the set of rational expressions into $\mathbb{K} \setminus \{0_\mathbb{K}\}$. It proves useful to view such structures as 'polynomials of expressions'. In essence, they capture the linearity of addition.

Definition 4 (Rational Polynomial) *A* polynomial *(of rational expressions) is a finite (left) linear combination of expressions. Syntactically it is a term built from the grammar*

$$\mathsf{P} ::= 0 \mid \langle k_1 \rangle \odot \mathsf{E}_1 \oplus \cdots \oplus \langle k_n \rangle \odot \mathsf{E}_n$$

where $k_i \in \mathbb{K} \setminus \{0_\mathbb{K}\}$ *denote* non-null *weights, and* E_i *denote* non-null *expressions. Expressions may not appear more than once in a polynomial. A* monomial *is a pair* $\langle k_i \rangle \odot \mathsf{E}_i$*. The weight of* E *in* P *is written* $\mathsf{P}(\mathsf{E})$*.*

We use specific symbols ($\odot$ and $\oplus$) to clearly separate the outer polynomial layer from the inner expression layer. Let $\mathsf{P} = \bigoplus_{i\in[n]} \langle k_i \rangle \odot \mathsf{E}_i$ be a polynomial of expressions. The *projection* of P is the expression $\mathsf{expr}(\mathsf{P}) := \langle k_1 \rangle \mathsf{E}_1 + \cdots + \langle k_n \rangle \mathsf{E}_n$ (or $\mathsf{0}$ if P is null); this operation is performed on a canonical form of the polynomial (expressions are sorted in a well defined order). Polynomials denote series: $[\![\mathsf{P}]\!] := [\![\mathsf{expr}(\mathsf{P})]\!]$. The *terms* of P is the set $\mathsf{exprs}(\mathsf{P}) := \{\mathsf{E}_1, \ldots, \mathsf{E}_n\}$.

Example 1 *Let* $\mathsf{E}_1 := \langle 5 \rangle 1 | 1 + \langle 4 \rangle a\, d\, e^* | x + \langle 3 \rangle b\, d\, e^* | x + \langle 2 \rangle a\, c\, e^* | x\, y + \langle 6 \rangle b\, c\, e^* | x\, y$*. Polynomial* '$\mathsf{P}_{1,a|x} := \langle 2 \rangle \odot ce^* \mid y \oplus \langle 4 \rangle \odot de^* \mid 1$' *has two monomials:* '$\langle 2 \rangle \odot ce^* \mid y$' *and* '$\langle 4 \rangle \odot de^* \mid 1$'*. It denotes the (left) quotient of* $[\![\mathsf{E}_1]\!]$ *by* $a \mid x$*, and* '$\mathsf{P}_{1,b|x} := \langle 6 \rangle \odot ce^* \mid y \oplus \langle 3 \rangle \odot de^* \mid 1$' *the quotient by* $b \mid x$*.*

Let $\mathsf{P} = \bigoplus_{i\in[n]} \langle k_i \rangle \odot \mathsf{E}_i$, $\mathsf{Q} = \bigoplus_{j\in[m]} \langle h_i \rangle \odot \mathsf{F}_i$ be polynomials, k a weight and F an expression, all possibly null, we introduce the following operations:

$$\mathsf{P} \cdot \mathsf{F} := \bigoplus_{i\in[n]} \langle k_i \rangle \odot (\mathsf{E}_i \cdot \mathsf{F})$$

$$\langle k \rangle \mathsf{P} := \bigoplus_{i\in[n]} \langle k k_i \rangle \odot \mathsf{E}_i \qquad \mathsf{P} \langle k \rangle := \bigoplus_{i\in[n]} \langle k_i \rangle \odot (\mathsf{E}_i \langle k \rangle)$$

$$\mathsf{P} \mid 1 := \bigoplus_{i\in[n]} \langle k_i \rangle \odot \mathsf{E}_i \mid 1 \qquad 1 \mid \mathsf{P} := \bigoplus_{i\in[n]} \langle k_i \rangle \odot 1 \mid \mathsf{E}_i$$

$$\mathsf{P} \mid \mathsf{Q} := \bigoplus_{(i,j)\in[n]\times[m]} \langle k_i \cdot h_j \rangle \odot \mathsf{E}_i \mid \mathsf{F}_j$$

Trivial identities might simplify the result. Note the asymmetry between left and right exterior products. The addition of polynomials is commutative, multiplication by zero (be it an expression or a weight) evaluates to the null polynomial, and the left-multiplication by a weight is distributive.

Lemma 1 $[\![\mathsf{P}\cdot\mathsf{F}]\!] = [\![\mathsf{P}]\!]\cdot[\![\mathsf{F}]\!]$ $\quad [\![\langle k\rangle\mathsf{P}]\!] = \langle k\rangle[\![\mathsf{P}]\!]$ $\quad [\![\mathsf{P}\langle k\rangle]\!] = [\![\mathsf{P}]\!]\langle k\rangle$ $[\![\mathsf{P}\mid\mathsf{Q}]\!] = [\![\mathsf{P}]\!]\mid[\![\mathsf{Q}]\!]$.

Proof: The proofs of the first three equalities are straightforward. The last one is a direct consequence of the bilinearity of tupling.

$$
\begin{aligned}
[\![\mathsf{P}\mid\mathsf{Q}]\!] &= \Big[\!\Big[\bigoplus_{(i,j)\in[n]\times[m]} \langle k_i\cdot h_j\rangle \odot \mathsf{E}_i \mid \mathsf{F}_j \Big]\!\Big] \\
&= \sum_{(i,j)\in[n]\times[m]} \big\langle k_i\cdot h_j\big\rangle [\![\mathsf{E}_i\mid\mathsf{F}_j]\!] \\
&= \sum_{(i,j)\in[n]\times[m]} \big\langle k_i\cdot h_j\big\rangle \Big([\![\mathsf{E}_i]\!]\mid[\![\mathsf{F}_j]\!]\Big) && \text{by Definition 2} \\
&= \Big(\sum_{i\in[n]}\langle k_i\rangle[\![\mathsf{E}_i]\!]\Big)\mid\Big(\sum_{j\in[m]}\big\langle h_j\big\rangle[\![\mathsf{F}_j]\!]\Big) && \text{by Proposition 3} \\
&= \Big[\!\Big[\bigoplus_{i\in[n]}\langle k_i\rangle\odot\mathsf{E}_i\Big]\!\Big]\mid\Big[\!\Big[\bigoplus_{j\in[m]}\langle h_j\rangle\odot\mathsf{F}_j\Big]\!\Big] \\
&= [\![\mathsf{P}]\!]\mid[\![\mathsf{Q}]\!] && \square
\end{aligned}
$$

2.4 Finite Weighted Automata

Our definition is slightly unusual in its handling of the labels, because it is meant for single and multitape automata.

Definition 5 (Weighted Automaton) *A* weighted automaton $\mathcal{A}$ *is a tuple* $\langle M, G, \mathbb{K}, Q, E, I, T\rangle$ *where:*

- M *is a monoid,*
- G *(the labels) is a set of generators of* M*,*
- $\mathbb{K}$ *(the set of weights) is a semiring,*
- Q *is a finite set of states,*
- I *and* T *are the* initial *and* final *functions from* Q *into* $\mathbb{K}$*,*

- *E is a (partial) function from $Q \times G \times Q$ into $\mathbb{K} \setminus \{0_\mathbb{K}\}$;*
  its domain represents the transitions: (source, label, destination).

An automaton is proper *if no label is ε_M.*

Our automata are 'ε-NFAs': they may have spontaneous transitions, as we do not require $\varepsilon \notin G$.

The *size* of an automaton is its number of states: $|\mathcal{A}| \coloneqq |Q|$.

A *path* π is a sequence of transitions $(q_0, \ell_1, q_1)(q_1, \ell_2, q_2) \cdots (q_{n-1}, \ell_n, q_n)$ where the source of each is the destination of the previous one; its *source* is $\iota(\pi) \coloneqq q_0$, its *destination* is $\tau(\pi) \coloneqq q_n$, its *label* is the word $\ell(\pi) \coloneqq \ell_1 \cdots \ell_n$, its *weight* is $w(\pi) \coloneqq E(q_0, \ell_1, q_1) \cdot \ldots \cdot E(q_{n-1}, \ell_n, q_n)$, and its *weighted label* [17] is the monomial $wl(\pi) \coloneqq w(\pi)\ell(\pi)$. The set of paths of $\mathcal{A}$ is denoted $\mathsf{Path}(\mathcal{A})$.

A state q is *initial* if $I(q) \neq 0_\mathbb{K}$. A state q is *accessible* if there is a path from an initial state to q. The *accessible* part of an automaton $\mathcal{A}$ is the sub-automaton whose states are the accessible states of $\mathcal{A}$.

A *computation* c is a path π together with its initial and final functions at the ends: $c \coloneqq (I(\iota(\pi)), \pi, T(\tau(\pi)))$, its weight is $w(c) \coloneqq I(\iota(\pi))w(\pi)T(\tau(\pi))$. The *evaluation* of word u by an automaton $\mathcal{A}$, $\mathcal{A}(u)$, is the sum of the weights of all the computations labeled by u, or $0_\mathbb{K}$ if there are none. The *behavior* of $\mathcal{A}$ is the series $[\![\mathcal{A}]\!] \coloneqq u \mapsto \mathcal{A}(u)$.

Automata with spontaneous transitions may be *invalid*, if they have cycles of spontaneous transitions whose weight is not starrable [17].

Definition 6 (Semantics of a State) *Given a valid weighted automaton $\mathcal{A} = \langle M, G, \mathbb{K}, Q, E, I, T\rangle$, the* semantics of state q *(aka, its* future*) is the series:*

$$[\![q]\!] \coloneqq T(q) + \sum_{\pi \in \mathsf{Path}(\mathcal{A}) | q = i(\pi)} wl(\pi)T(\tau(\pi)) \tag{1}$$

Clearly, $[\![\mathcal{A}]\!] = \sum_{q \in Q} I(q)[\![q]\!]$.

Proposition 4 *For any valid automaton $\mathcal{A}$, we have:*

$$[\![q]\!] = T(q) + \sum_{\ell \in G, q' \in Q} E(q, \ell, q')\ell[\![q']\!] \tag{2}$$

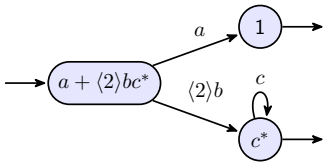
The equivalence of Eqs. (1) and (2) can be seen as two different strategies of evaluation: the first one is by depth first (follow each path individually,

then sum their weights), the second one by breadth (starting from the set of initial states, descend 'simultaneously' each transition, and repeat).

A simple proof by induction [8, Sec. 2.5] suffices in the absence of spontaneous transitions. With cycles of spontaneous transitions, we face infinite sums whose formal treatment requires arguments that go way beyond the scope of this paper, and the possibility that the automaton is invalid. This is in fact the core of the work of Lombardy and Sakarovitch [17].

3 Rational Expansions

Given an expression such as $a + \langle 2\rangle bc^*$, we want an algorithm to compute its *derived-term automaton*:



To this end, we introduce *expansions*. They are comparable to some normal form for rational expressions. They highlight the labels by which an expression may 'start' (the *first*), and then the various possible continuations (a weighted set of expressions, aka, a polynomial). The expansion of $a+\langle 2\rangle bc^*$ is $a \odot [\langle 1\rangle \odot 1] \oplus b \odot [\langle 2\rangle \odot c^*]$.

3.1 Rational Expansions

Definition 7 (Rational Expansion) *A* rational expansion X *is a term* $\mathsf{X} ::= \langle k\rangle \oplus \ell_1 \odot [\mathsf{P}_1] \oplus \cdots \oplus \ell_n \odot [\mathsf{P}_n]$ *where* k *is a weight (possibly zero),* $\ell_i \in G$ *are labels (occurring at most once), and* P_i *non-null polynomials. The* immediate constant term *of an expansion* X*, noted* $\mathsf{X}_\$$*, is* k*. The* firsts *of* X *is* $f(\mathsf{X}) \coloneqq \{\ell_1, \dots, \ell_n\}$ *(possibly empty) and its* terms *are* $\mathsf{exprs}(\mathsf{X}) \coloneqq \bigcup_{i\in[n]} \mathsf{exprs}(\mathsf{P}_i)$.

To ease reading, polynomials are written in square brackets. Contrary to expressions and polynomials, there is no specific term for the zero expansion: it is represented by $\langle 0_{\mathbb{K}}\rangle$, the zero weight. Given an expansion X, we denote by X_ℓ (or $\mathsf{X}(\ell)$) the polynomial corresponding to ℓ in X, or the null polynomial if $\ell \notin f(\mathsf{X})$. Expansions will thus be written: $\mathsf{X} = \langle \mathsf{X}_\$\rangle \oplus \bigoplus_{\ell\in f(\mathsf{X})} \ell \odot [\mathsf{X}_\ell]$.

An expansion X can be projected as a rational expression $\mathsf{expr}(\mathsf{X})$ by mapping weights, labels and polynomials to their corresponding rational

expressions, and $\oplus/\odot$ to the sum/concatenation of expressions. Again, this is performed on a canonical form of the expansion: labels are sorted. Expansions also denote series: $[\![\mathsf{X}]\!] \coloneqq [\![\mathsf{expr}(\mathsf{X})]\!]$. An expansion X is *equivalent* to an expression E iff $[\![\mathsf{X}]\!] = [\![\mathsf{E}]\!]$.

An expansion X is *immediately proper* if $\mathsf{X}_{\$} = 0_{\mathbb{K}}$; its *immediate proper part*, X_p, is the expansion which coincides with X but with a null immediate constant term; hence[7] $\mathsf{X} = \langle \mathsf{X}_{\$} \rangle \oplus \mathsf{X}_p$. An immediately proper expansion may denote an improper series: consider for instance $\mathsf{X} \coloneqq \varepsilon \odot [\langle 3 \rangle \odot a^*]$; it is immediately proper ($\mathsf{X}_{\$} = 0$), yet $[\![\mathsf{X}]\!](\varepsilon) = 3$.

Example 2 (Example 1 continued) *Let expansion* $\mathsf{X}_1 \coloneqq \langle 5 \rangle \oplus a|x \odot [\mathsf{P}_{1,a|x}] \oplus b|x \odot [\mathsf{P}_{1,b|x}]$. *Its immediate constant term is 5, and* X_1 *maps the generator* $a|x$ *(resp.* $b|x$*) to the polynomial* $\mathsf{X}_1(a|x) = \mathsf{P}_{1,a|x}$ *(resp.* $\mathsf{X}_1(b|x) = \mathsf{P}_{1,b|x}$*).* X_1 *can be proved to be equivalent to* E_1.

Let X, Y be expansions, k a weight, and E an expression (all possibly null):

$$\mathsf{X} \oplus \mathsf{Y} \coloneqq \langle \mathsf{X}_{\$} + \mathsf{Y}_{\$} \rangle \oplus \bigoplus_{\ell \in f(\mathsf{X}) \cup f(\mathsf{Y})} \ell \odot [\mathsf{X}_\ell \oplus \mathsf{Y}_\ell] \tag{3}$$

$$\langle k \rangle \mathsf{X} \coloneqq \langle k \mathsf{X}_{\$} \rangle \oplus \bigoplus_{\ell \in f(\mathsf{X})} \ell \odot [\langle k \rangle \mathsf{X}_\ell] \qquad \mathsf{X} \langle k \rangle \coloneqq \langle \mathsf{X}_{\$} k \rangle \oplus \bigoplus_{\ell \in f(\mathsf{X})} \ell \odot [\mathsf{X}_\ell \langle k \rangle] \tag{4}$$

$$\mathsf{X} \cdot \mathsf{E} \coloneqq \bigoplus_{\ell \in f(\mathsf{X})} \ell \odot [\mathsf{X}_\ell \cdot \mathsf{E}] \qquad \text{with } \mathsf{X} \text{ immediately proper} \tag{5}$$

$$\mathsf{X} \mid \mathsf{Y} \coloneqq \begin{cases} \langle \mathsf{X}_{\$} \mathsf{Y}_{\$} \rangle \\ \oplus \langle \mathsf{X}_{\$} \rangle \bigoplus_{\ell' \in f(\mathsf{Y})} (\varepsilon|\ell') \odot (1 \mid \mathsf{Y}_{\ell'}) \\ \oplus \langle \mathsf{Y}_{\$} \rangle \bigoplus_{\ell \in f(\mathsf{X})} (\ell|\varepsilon) \odot (\mathsf{X}_\ell \mid 1) \\ \oplus \bigoplus_{\ell|\ell' \in f(\mathsf{X}) \times f(\mathsf{Y})} (\ell|\ell') \odot (\mathsf{X}_\ell \mid \mathsf{Y}_{\ell'}) \end{cases} \tag{6}$$

Since by definition expansions never map to null polynomials, some firsts might be smaller that suggested by these equations. For instance in $\mathbb{Z}$ the sum of $\langle 1 \rangle \oplus a \odot [\langle 1 \rangle \odot b]$ and $\langle 1 \rangle \oplus a \odot [\langle -1 \rangle \odot b]$ is $\langle 2 \rangle$.

[7] The (straightforward) definition of addition of expansions, $\oplus$, will be given below.

The following lemma is simple to establish: lift semantic equivalences, such as Proposition 3, to syntax, using Lemma 1.

Lemma 2 $[\![\mathsf{X} \oplus \mathsf{Y}]\!] = [\![\mathsf{X}]\!] + [\![\mathsf{Y}]\!]$ $\quad [\![\langle k\rangle \mathsf{X}]\!] = \langle k\rangle [\![\mathsf{X}]\!]$ $\quad [\![\mathsf{X}\langle k\rangle]\!] = [\![\mathsf{X}]\!]\langle k\rangle$
$[\![\mathsf{X} \cdot \mathsf{E}]\!] = [\![\mathsf{X}]\!] \cdot [\![\mathsf{E}]\!]$ $\quad [\![\mathsf{X} \mid \mathsf{Y}]\!] = [\![\mathsf{X}]\!] \mid [\![\mathsf{Y}]\!]$

3.2 Computing the Expansion of an Expression

We introduce a procedure to compute an (equivalent) expansion from an expression.

Definition 8 (Expansion of a Rational Expression) *The* expansion of a rational expression E*, written* $d(\mathsf{E})$*, is defined inductively as follows:*

$$d(\mathsf{0}) \coloneqq \langle 0_{\mathbb{K}}\rangle \qquad d(\mathsf{1}) \coloneqq \langle 1_{\mathbb{K}}\rangle \qquad d(a) \coloneqq a \odot [\langle 1_{\mathbb{K}}\rangle \odot \mathsf{1}] \tag{7}$$

$$d(\mathsf{E} + \mathsf{F}) \coloneqq d(\mathsf{E}) \oplus d(\mathsf{F}) \tag{8}$$

$$d(\langle k\rangle \mathsf{E}) \coloneqq \langle k\rangle d(\mathsf{E}) \qquad d(\mathsf{E}\langle k\rangle) \coloneqq d(\mathsf{E})\langle k\rangle \tag{9}$$

$$d(\mathsf{E} \cdot \mathsf{F}) \coloneqq d_p(\mathsf{E}) \cdot \mathsf{F} \oplus \big\langle d_{\$}(\mathsf{E})\big\rangle d(\mathsf{F}) \tag{10}$$

$$d(\mathsf{E}^*) \coloneqq \big\langle d_{\$}(\mathsf{E})^*\big\rangle \oplus \big\langle d_{\$}(\mathsf{E})^*\big\rangle d_p(\mathsf{E}) \cdot \mathsf{E}^* \tag{11}$$

$$d(\mathsf{E} \mid \mathsf{F}) \coloneqq d(\mathsf{E}) \mid d(\mathsf{F}) \tag{12}$$

where $d_{\$}(\mathsf{E}) \coloneqq d(\mathsf{E})_{\$}, d_p(\mathsf{E}) \coloneqq d(\mathsf{E})_p$ *are the immediate constant term and immediate proper part of* $d(\mathsf{E})$*.*

The right-hand sides are indeed expansions. The computation trivially terminates: induction is performed on strictly smaller subexpressions. Note that the firsts are a subset of the labels of the expression.

Example 3 (Examples 1 and 2 continued)
With $\mathsf{E}_1 \coloneqq \langle 5\rangle 1|1 + \langle 4\rangle a\,d\,e^*|x + \langle 3\rangle b\,d\,e^*|x + \langle 2\rangle a\,c\,e^*|x\,y + \langle 6\rangle b\,c\,e^*|x\,y$*, one has:*

$$d(\mathsf{E}_1) = \begin{cases} \langle 5\rangle \\ \oplus\, a|x \odot [\langle 2\rangle \odot ce^*|y \oplus \langle 4\rangle \odot de^*|\varepsilon] \\ \oplus\, b|x \odot [\langle 6\rangle \odot ce^*|y \oplus \langle 3\rangle \odot de^*|\varepsilon] \end{cases}$$
$$= \mathsf{X}_1 \qquad \textit{(from Example 2)}$$

This expansion is the introductory example from Section 2.

Proposition 5 *The expansion of a rational expression is equivalent to the expression.*

Proof: We prove that $[\![d(\mathsf{E})]\!] = [\![\mathsf{E}]\!]$ by induction on the expression. The equivalence is straightforward for Eqs. (7) to (9) and (12), viz., $[\![d(\mathsf{E} \mid \mathsf{F})]\!] = [\![d(\mathsf{E}) \mid d(\mathsf{F})]\!]$ (by Eq. (12)) $= [\![d(\mathsf{E})]\!] \mid [\![d(\mathsf{F})]\!]$ (by Lemma 2) $= [\![\mathsf{E}]\!] \mid [\![\mathsf{F}]\!]$ (by induction hypothesis) $= [\![\mathsf{E} \mid \mathsf{F}]\!]$ (by Lemma 2) . The case of multiplication, Eq. (10), follows from:

$$
\begin{aligned}
[\![d(\mathsf{E}\cdot\mathsf{F})]\!] &= \Big[\!\Big[d_p(\mathsf{E})\cdot\mathsf{F} \oplus \big\langle d_{\$}(\mathsf{E})\big\rangle \cdot d(\mathsf{F}) \Big]\!\Big] && = [\![d_p(\mathsf{E})]\!]\cdot[\![\mathsf{F}]\!] + \big\langle d_{\$}(\mathsf{E})\big\rangle \cdot [\![d(\mathsf{F})]\!] \\
&= [\![d_p(\mathsf{E})]\!]\cdot[\![\mathsf{F}]\!] + \big\langle d_{\$}(\mathsf{E})\big\rangle \cdot [\![\mathsf{F}]\!] && = \Big([\![\langle d_{\$}(\mathsf{E})\rangle]\!] + [\![d_p(\mathsf{E})]\!] \Big)\cdot[\![\mathsf{F}]\!] \\
&= \Big[\!\Big[\big\langle d_{\$}(\mathsf{E})\big\rangle + d_p(\mathsf{E}) \Big]\!\Big] \cdot [\![\mathsf{F}]\!] && = [\![d(\mathsf{E})]\!]\cdot[\![\mathsf{F}]\!] \\
&= [\![\mathsf{E}]\!]\cdot[\![\mathsf{F}]\!] && = [\![\mathsf{E}\cdot\mathsf{F}]\!]
\end{aligned}
$$

The case of Kleene star, Eq. (11), follows from Proposition 2. □

An expansion X is *normal* if $\varepsilon \notin f(\mathsf{X}_p)$. For instance $\langle 1\rangle + a \odot [\langle \frac{1}{2}\rangle \odot (\langle \frac{1}{2}\rangle a)^*]$ is normal, $\varepsilon \odot [\langle 1_{\mathbb{K}}\rangle \odot (\langle \frac{1}{2}\rangle a)^*]$ is not. Both denote $(\langle \frac{1}{2}\rangle a)^*$.

Lemma 3 *The expansion of an expression is normal.*

This is easily established by a simple verification on Definition 8. However, with the composition operator, normality is no longer guaranteed (Section 5).

3.3 Derived Terms of an Expression

In this section, we prove that repeated computations of expansions are 'generated' by a finite number of expressions, called the *derived terms*. The result, Lemma 7, will prove that our construct builds *finite* automata.

For any sets of expressions S, T, let $S \mid T \coloneqq \{\mathsf{E} \mid \mathsf{F}\}_{\mathsf{E}\in S, \mathsf{F}\in T}$.

Definition 9 (Derived Terms) *The* proper derived terms *of an expression* E *is* $\mathrm{PD}(\mathsf{E})$*, the set of expressions defined inductively below:*

$$
\begin{aligned}
&\mathrm{PD}(\mathsf{0}) \coloneqq \emptyset \\
&\mathrm{PD}(\mathsf{1}) \coloneqq \{\mathsf{1}\} \\
&\mathrm{PD}(\ell) \coloneqq \{\mathsf{1}\} \qquad \forall \ell \in G
\end{aligned}
$$

$$
\begin{aligned}
\mathrm{PD}(\mathsf{E}+\mathsf{F}) &\coloneqq \mathrm{PD}(\mathsf{E}) \cup \mathrm{PD}(\mathsf{F}) \\
\mathrm{PD}(\langle k\rangle \mathsf{E}) &\coloneqq \mathrm{PD}(\mathsf{E}) \qquad \forall k \in \mathbb{K} \\
\mathrm{PD}(\mathsf{E}\langle k\rangle) &\coloneqq \{\mathsf{E}_i\langle k\rangle \mid \mathsf{E}_i \in \mathrm{PD}(\mathsf{E})\} \qquad \forall k \in \mathbb{K} \\
\mathrm{PD}(\mathsf{E}\cdot\mathsf{F}) &\coloneqq \{\mathsf{E}_i \cdot \mathsf{F} \mid \mathsf{E}_i \in \mathrm{PD}(\mathsf{E})\} \cup \mathrm{PD}(\mathsf{F}) \\
\mathrm{PD}(\mathsf{E}^*) &\coloneqq \{\mathsf{E}_i \cdot \mathsf{E}^* \mid \mathsf{E}_i \in \mathrm{PD}(\mathsf{E})\} \\
\mathrm{PD}(\mathsf{E} \mid \mathsf{F}) &\coloneqq (\mathrm{PD}(\mathsf{E}) \mid \mathrm{PD}(\mathsf{F})) \cup (\{\mathsf{1}\} \mid \mathrm{PD}(\mathsf{F})) \cup (\mathrm{PD}(\mathsf{E}) \mid \{\mathsf{1}\})
\end{aligned}
$$

where G is the set of generators.

The derived terms *of an expression* E *is* $\mathrm{D}(\mathsf{E}) \coloneqq \mathrm{PD}(\mathsf{E}) \cup \{\mathsf{E}\}$.

This simple inductive definition is similar to Def. 3 of Lombardy and Sakarovitch [16]. Later, the authors changed their definition of derived term to rely on derivatives with respect to words [2, Def. 3]. The original definition denotes the set of *potential* derived terms, the second one denotes the *true* derived terms (i.e., the actual states of the derived-term automaton). While the two concepts coincide in the case of basic operators, they differ in ours: tupling (and later composition) introduce many potential derived terms, most of them not appearing in the resulting derived-term automaton. This is why we do not use the name *true derived term* and the notation TD.

Lemma 4 (Number of Derived Terms) *For any k-tape expression* E*,*

$$|\mathrm{PD}(\mathsf{E})| \leq \prod_{i\in[k]} (\|\mathsf{E}\|_i + 1) \quad .$$

Proof: It is simple to check by induction on E that for all cases, except tuple, $\mathrm{PD}(\mathsf{E}) \leq \|\mathsf{E}\|$ (which is the classical result for single-tape expressions, see for example Lombardy and Sakarovitch [16, Theorem 2] or Angrand et al. [2, Theorem 3]). In the case of $|$, it is clear that $|\mathrm{PD}(\mathsf{E} \mid \mathsf{F})| \leq (|\mathrm{PD}(\mathsf{E})| + 1) \cdot (|\mathrm{PD}(\mathsf{F})| + 1)$, hence the result.□

Lemma 5 (Proper Derived Terms and Single Expansion) *For any expression* E*,* $\mathsf{exprs}\big(d(\mathsf{E})\big) \subseteq \mathrm{PD}(\mathsf{E})$.

Proof: Established by a simple verification of Definition 8. □

The derived terms of derived terms of E are derived terms of E. In other words, repeated expansions never 'escape' the set of derived terms.

Lemma 6 (Proper Derived Terms and Repeated Expansions) *Let* E *be an expression. For all* $\mathsf{F} \in \mathrm{PD}(\mathsf{E})$*,* $\mathsf{exprs}\big(d(\mathsf{F})\big) \subseteq \mathrm{PD}(\mathsf{E})$.

Proof: This will be proved by induction over E.

Case $\mathsf{E} = 0$ or $\mathsf{E} = 1$. Impossible, as then $\mathrm{PD}(\mathsf{E}) = \emptyset$.

Case $\mathsf{E} = a$. Then $\mathrm{PD}(\mathsf{E}) = \{1\}$, hence $\mathsf{F} = 1$ and therefore $d(\mathsf{F}) = d(1) = \langle 0_{\mathbb{K}} \rangle$, so $\mathsf{exprs}\big(d(\mathsf{F})\big) = \emptyset \subseteq \mathrm{PD}(\mathsf{E})$.

Case $\mathsf{E} = \mathsf{G} + \mathsf{H}$. Then $\mathrm{PD}(\mathsf{E}) = \mathrm{PD}(\mathsf{G}) \cup \mathrm{PD}(\mathsf{H})$. Suppose, without loss of generality, that $\mathsf{F} \in \mathrm{PD}(\mathsf{G})$. Then, by induction hypothesis, $\mathsf{exprs}\big(d(\mathsf{F})\big) \subseteq \mathrm{PD}(\mathsf{G}) \subseteq \mathrm{PD}(\mathsf{E})$.

Case $\mathsf{E} = \langle k \rangle \mathsf{G}$. Then if $\mathsf{F} \in \mathrm{PD}(\langle k \rangle \mathsf{G}) = \mathrm{PD}(\mathsf{G})$, so by induction hypothesis $\mathsf{exprs}\big(d(\mathsf{F})\big) \subseteq \mathrm{PD}(\mathsf{G}) = \mathrm{PD}(\langle k \rangle \mathsf{G}) = \mathrm{PD}(\mathsf{E})$.

Case $\mathsf{E} = \mathsf{G}\langle k \rangle$. Then $\forall \mathsf{F} \in \mathrm{PD}(\mathsf{G}\langle k \rangle) = \{\mathsf{G}_i \langle k \rangle \mid \mathsf{G}_i \in \mathrm{PD}(\mathsf{G})\}$, there exists an i such that $\mathsf{F} = \mathsf{G}_i \langle k \rangle$. Then $d(\mathsf{F}) = d(\mathsf{G}_i \langle k \rangle) = d(\mathsf{G}_i)\langle k \rangle$ hence $\mathsf{exprs}\big(d(\mathsf{F})\big) = \mathsf{exprs}\big(d(\mathsf{G}_i)\langle k \rangle\big)$.

Since $\mathsf{G}_i \in \mathrm{PD}(\mathsf{G})$, by induction hypothesis $\mathsf{exprs}\big(d(\mathsf{G}_i)\big) \subseteq \mathrm{PD}(\mathsf{G})$, so by definition of the right exterior product of expansions (and polynomials), $\mathsf{exprs}\big(d(\mathsf{G}_i)\langle k \rangle\big) \subseteq \mathrm{PD}(\mathsf{G}\langle k \rangle) = \mathrm{PD}(\mathsf{E})$.

Hence $\mathsf{exprs}\big(d(\mathsf{F})\big) \subseteq \mathrm{PD}(\mathsf{E})$.

Case $\mathsf{E} = \mathsf{G} \cdot \mathsf{H}$. Then $\mathrm{PD}(\mathsf{E}) = \{\mathsf{G}_i \cdot \mathsf{H} \mid \mathsf{G}_i \in \mathrm{PD}(\mathsf{G})\} \cup \mathrm{PD}(\mathsf{H})$.

- If $\mathsf{F} = \mathsf{G}_i \cdot \mathsf{H}$ with $\mathsf{G}_i \in \mathrm{PD}(\mathsf{G})$, then $d(\mathsf{F}) = d(\mathsf{G}_i \cdot \mathsf{H}) = d_p(\mathsf{G}_i) \cdot \mathsf{H} \oplus \big\langle d_{\$}(\mathsf{G}_i) \big\rangle d(\mathsf{H})$.
  Since $\mathsf{G}_i \in \mathrm{PD}(\mathsf{G})$ by induction hypothesis $\mathsf{exprs}\big(d_p(\mathsf{G}_i)\big) = \mathsf{exprs}\big(d(\mathsf{G}_i)\big) \subseteq \mathrm{PD}(\mathsf{G})$. By definition of the product of an expansion by an expression, $\mathsf{exprs}\big(d_p(\mathsf{G}_i) \cdot \mathsf{H}\big) \subseteq \{\mathsf{G}_j \cdot \mathsf{H} \mid \mathsf{G}_j \in \mathrm{PD}(\mathsf{G})\} \subseteq \mathrm{PD}(\mathsf{G} \cdot \mathsf{H}) = \mathrm{PD}(\mathsf{E})$.
- If $\mathsf{F} \in \mathrm{PD}(\mathsf{H})$, then by induction hypothesis $\mathsf{exprs}\big(d(\mathsf{F})\big) \subseteq \mathrm{PD}(\mathsf{H}) \subseteq \mathrm{PD}(\mathsf{E})$.

Case $\mathsf{E} = \mathsf{G}^*$. If $\mathsf{F} \in \mathrm{PD}(\mathsf{E}) = \{\mathsf{G}_i \cdot \mathsf{G}^* \mid \mathsf{G}_i \in \mathrm{PD}(\mathsf{G})\}$, i.e., if $\mathsf{F} = \mathsf{G}_i \cdot \mathsf{G}^*$ with $\mathsf{G}_i \in \mathrm{PD}(\mathsf{G})$, then $d(\mathsf{F}) = d(\mathsf{G}_i \cdot \mathsf{G}^*) = d_p(\mathsf{G}_i) \cdot \mathsf{G}^* \oplus \big\langle d_{\$}(\mathsf{G}_i) \big\rangle d(\mathsf{G}^*)$, so $\mathsf{exprs}\big(d(\mathsf{F})\big) \subseteq \mathsf{exprs}\big(d_p(\mathsf{G}_i) \cdot \mathsf{G}^*\big) \cup \mathsf{exprs}\big(d(\mathsf{G}^*)\big)$.[8] We will show that both are subsets of $\mathrm{PD}(\mathsf{E})$, which will prove the result.

[8] Given two expansions $\mathsf{X}_1, \mathsf{X}_2$, $\mathsf{exprs}(\mathsf{X}_1 \oplus \mathsf{X}_2) \subseteq \mathsf{exprs}(\mathsf{X}_1) \cup \mathsf{exprs}(\mathsf{X}_2)$, but they may be different; consider for instance $\mathsf{X}_1 = a \odot [\langle 1 \rangle \odot 1]$ and $\mathsf{X}_2 = a \odot [\langle -1 \rangle \odot 1]$ with $\mathbb{K} = \mathbb{Z}$.

Since $\mathsf{G}_i \in \mathrm{PD}(\mathsf{G})$, by induction hypothesis, $\mathsf{exprs}\big(d_p(\mathsf{G}_i)\big) = \mathsf{exprs}\big(d(\mathsf{G}_i)\big) \subseteq \mathrm{PD}(\mathsf{G})$, so by definition of a product of an expansion by an expression, $\mathsf{exprs}\big(d_p(\mathsf{G}_i) \cdot \mathsf{G}^*\big) \subseteq \{\mathsf{G}_j \cdot \mathsf{G}^* \mid \mathsf{G}_j \in \mathrm{PD}(\mathsf{G})\} = \mathrm{PD}(\mathsf{E})$.

By Lemma 5 $\mathsf{exprs}\big(d(\mathsf{G}^*)\big) \subseteq \mathrm{PD}(\mathsf{G}^*) = \mathrm{PD}(\mathsf{E})$.

Case $\mathsf{E} = \mathsf{G} \mid \mathsf{H}$. Let $\mathsf{F} \in \mathrm{PD}(\mathsf{E}) = \mathrm{PD}(\mathsf{G})|\mathrm{PD}(\mathsf{H}) \cup \{1\}|\mathrm{PD}(\mathsf{H}) \cup \mathrm{PD}(\mathsf{G})|\{1\}$.

- Suppose $\mathsf{F} \in \mathrm{PD}(\mathsf{G})|\mathrm{PD}(\mathsf{H})$, i.e., $\mathsf{F} = \mathsf{G}_i|\mathsf{H}_j$ with $\mathsf{G}_i \in \mathrm{PD}(\mathsf{G}), \mathsf{H}_j \in \mathrm{PD}(\mathsf{H})$. By induction hypothesis $\mathsf{exprs}\big(d(\mathsf{G}_i)\big) \subseteq \mathrm{PD}(\mathsf{G})$ and $\mathsf{exprs}\big(d(\mathsf{H}_j)\big) \subseteq \mathrm{PD}(\mathsf{H})$, hence by definition of the tupling of expansions (Eq. (6)) $\mathsf{exprs}\big(d(\mathsf{G}_i) \mid d(\mathsf{H}_j)\big) \subseteq (\mathrm{PD}(\mathsf{G}) \mid \mathrm{PD}(\mathsf{H})) \cup (\{1\} \mid \mathrm{PD}(\mathsf{H})) \cup (\mathrm{PD}(\mathsf{G}) \mid \{1\}) = \mathrm{PD}(\mathsf{E})$.
  We have $d(\mathsf{F}) = d(\mathsf{G}_i \mid \mathsf{H}_j) = d(\mathsf{G}_i) \mid d(\mathsf{H}_j)$, so $\mathsf{exprs}\big(d(\mathsf{F})\big) = \mathsf{exprs}\big(d(\mathsf{G}_i) \mid d(\mathsf{H}_j)\big) \subseteq \mathrm{PD}(\mathsf{E})$.
- Suppose $\mathsf{F} \in \{1\} \mid \mathrm{PD}(\mathsf{H})$, i.e., $\mathsf{F} = 1 \mid \mathsf{H}_j$ with $\mathsf{H}_j \in \mathrm{PD}(\mathsf{H})$. By induction hypothesis $\mathsf{exprs}\big(d(\mathsf{H}_j)\big) \subseteq \mathrm{PD}(\mathsf{H})$, hence by Eq. (6) $\mathsf{exprs}\big(d(1) \mid d(\mathsf{H}_j)\big) = \mathsf{exprs}\big(\langle 1_{\mathbb{K}} \rangle \mid d(\mathsf{H}_j)\big) = \{1\} \mid \mathsf{exprs}\big(d(\mathsf{H}_j)\big) \subseteq \{1\} \mid \mathrm{PD}(\mathsf{H}) \subseteq (\mathrm{PD}(1) \mid \mathrm{PD}(\mathsf{H})) \cup (\{1\} \mid \mathrm{PD}(\mathsf{H})) \cup (\mathrm{PD}(1) \mid \{1\}) = \mathrm{PD}(\mathsf{E})$.
  We have $d(\mathsf{F}) = d(1 \mid \mathsf{H}_j) = d(1) \mid d(\mathsf{H}_j)$, so $\mathsf{exprs}\big(d(\mathsf{F})\big) = \mathsf{exprs}\big(d(1) \mid d(\mathsf{H}_j)\big) \subseteq \mathrm{PD}(\mathsf{E})$.
- The case $\mathsf{F} \in \mathrm{PD}(\mathsf{G}) \mid \{1\}$ is similar. □

Lemma 7 (Derived Terms and Repeated Expansions)
Let E *be an expression. For all* $\mathsf{F} \in \mathrm{D}(\mathsf{E})$, $\mathsf{exprs}\big(d(\mathsf{F})\big) \subseteq \mathrm{PD}(\mathsf{E})$.

Proof: Since $\mathrm{D}(\mathsf{E}) = \mathrm{PD}(\mathsf{E}) \cup \{\mathsf{E}\}$, this is an immediate consequence of Lemmas 5 and 6.□

4 Expansion-Based Derived-Term Automaton

The repeated computations of expansions build an automaton.

Definition 10 (Derived-Term Automaton)
The derived-term automaton *of an expression* E *over* G *is the* accessible part *of the automaton* $\mathcal{A}_{\mathsf{E}} \coloneqq \langle M, G^?, \mathbb{K}, Q, E, I, T \rangle$ *defined as follows:*

- Q *is the set of rational expressions over* G *with weights in* $\mathbb{K}$,
- $I = \mathsf{E} \mapsto 1_{\mathbb{K}}$,

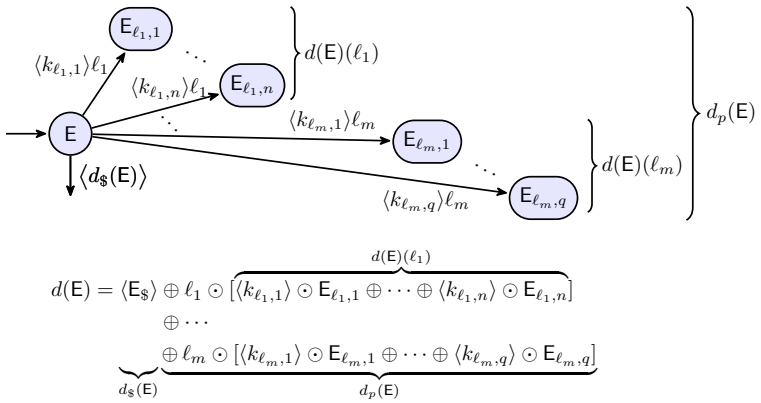


Figure 2: Initial part of the derived-term automaton of E. This figure is somewhat misleading: some $\mathsf{E}_{\ell,i}$ might be equal to an $\mathsf{E}_{\ell',j}$ with $\ell \neq \ell'$, or E — but never another $\mathsf{E}_{\ell,j}$. In other words, from a given state, transitions with different labels may reach common states.

- $E(\mathsf{F}, \ell, \mathsf{F}') = k$ *iff* $\ell \in f(d(\mathsf{F}))$ and $\langle k\rangle \odot \mathsf{F}' \in d_p(\mathsf{F})(\ell)$,
- $T(\mathsf{F}) = d_\$(\mathsf{F})$.

The Fig. 2 illustrates the process.

Even if $\varepsilon \notin G$, the derived-term automaton *may* have spontaneous transitions ($G \coloneqq \{\varepsilon\} \cup G$). These provisions will be used in Section 5.

Example 4 (Examples 1 to 3 continued) *Fig. 1 shows the derived-term automaton of* E_1 *from the introductory example (Section 2 and Example 1).*

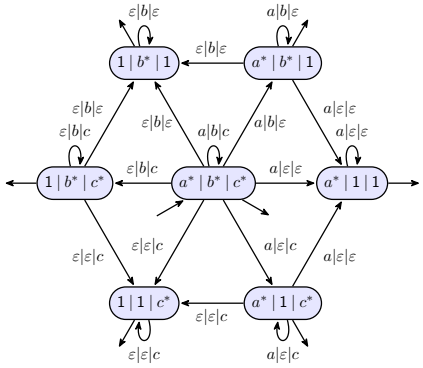
We must justify Definition 10 by proving that this automaton is finite.

Theorem 1 *For any k-tape expression* E, $|\mathcal{A}_\mathsf{E}| \leq \prod_{i\in[k]}(\|\mathsf{E}\|_i + 1) + 1$.

Proof: First observe that the states of $\mathcal{A}_\mathsf{E}$ are members of $\mathrm{D}(\mathsf{E})$ (this follows from a simple examination of the repeated computations of expansions in Definition 10). Then Lemma 7 allows to conclude. □

Example 5 *Let $\mathcal{A}_k$ be the derived-term automaton of the k-tape expression $a_1^* \mid \cdots \mid a_k^*$. The states of $\mathcal{A}_k$ are all the possible expressions where the tape i features 1 or a_i^*, except $1 \mid \cdots \mid 1$. Therefore $|\mathcal{A}_k| = 2^k - 1$, and $\prod_{i\in[k]}(\|\mathsf{E}\|_i + 1) = 2^k$.*

$\mathcal{A}_3$, the derived-term automaton of $a^* \mid b^* \mid c^*$, is depicted on the right.



Theorem 2 *If valid, any expression E and its expansion-based derived-term automaton $\mathcal{A}_\mathsf{E}$ denote the same series, i.e., $[\![\mathcal{A}_\mathsf{E}]\!] = [\![\mathsf{E}]\!]$.*

Since the expansions are normal (Lemma 3), the firsts of the immediate proper part exclude ε, this automaton is therefore proper. As a consequence, the proof of Demaille [9, Theorem 2] would suffice to establish this result. However, with the introduction of the composition operator in Section 5, expansions may no longer be normal and automata proper. We need a more powerful proof.

Proof: We show that the semantics of the states of $\mathcal{A}_\mathsf{E}$ Eq. (2) and of the expressions in $\mathrm{D}(\mathsf{E})$ define the same system of linear equations.

The Definition 10 shows that each state q_F of the $\mathcal{A}_\mathsf{E}$ has the following semantics:

$$[\![q_\mathsf{F}]\!] = \sum_{\substack{\ell\in f(d(\mathsf{F}))\\ \langle k\rangle\odot\mathsf{F}'\in d(\mathsf{F})(\ell)}} k_{\ell,\mathsf{F}'}\,\ell\,[\![q_{\mathsf{F}'}]\!] \tag{13}$$

Besides:

$$\begin{aligned}
[\![\mathsf{F}]\!] &= [\![d(\mathsf{F})]\!] && \text{(by Proposition 5)}\\
&= \Big[\!\Big[\bigoplus_{\ell\in f(d(\mathsf{F}))} \ell\odot d(\mathsf{F})(\ell)\Big]\!\Big]\\
&= \sum_{\ell\in f(d(\mathsf{F}))} \ell\,[\![d(\mathsf{F})(\ell)]\!]\\
&= \sum_{\ell\in f(d(\mathsf{F}))} \ell\Big[\!\Big[\Big[\bigoplus_{\langle k_{\ell,i}\rangle\odot\mathsf{F}_{\ell,i}\in d(\mathsf{F})(\ell)} \langle k_{\ell,i}\rangle\odot\mathsf{F}_{\ell,i}\Big]\Big]\!\Big]
\end{aligned}$$

$$= \sum_{\ell \in f(d(\mathsf{F}))} \ell \sum_{\langle k_{\ell,i}\rangle \odot \mathsf{F}_{\ell,i} \in d(\mathsf{F})(\ell)} k_{\ell,i} \llbracket \mathsf{F}_{\ell,i} \rrbracket$$

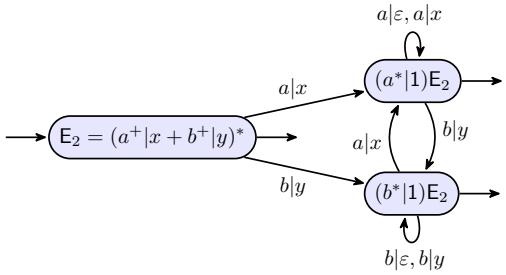
$$= \sum_{\substack{\ell \in f(d(\mathsf{F})) \\ \langle k_{\ell,i}\rangle \odot \mathsf{F}_{\ell,i} \in d(\mathsf{F})(\ell)}} k_{\ell,i}\, \ell \, \llbracket \mathsf{F}_{\ell,i} \rrbracket \tag{14}$$

One can then verify that Eqs. (13) and (14) define the same system of linear equations, hence $\llbracket \mathcal{A}_{\mathsf{E}} \rrbracket = \llbracket \mathsf{E} \rrbracket$.□

Example 6 *Let* $\mathsf{E}_2 \coloneqq (a^+ \mid x + b^+ \mid y)^*$*, where* $\mathsf{E}^+ \coloneqq \mathsf{E}\mathsf{E}^*$*. Its expansion is*

$$\begin{aligned} d(\mathsf{E}_2) = \quad & \varepsilon|\varepsilon \odot [\langle 1\rangle \odot 1] \\ & \oplus\ a|x \odot [(a^* \mid 1)(a^+ \mid x + b^+ \mid y)^*] \\ & \oplus\ b|y \odot [(b^* \mid 1)(a^+ \mid x + b^+ \mid y)^*] \\ = {} & \varepsilon|\varepsilon \odot [\langle 1\rangle \odot 1] \oplus a|x \odot [(a^* \mid 1)\mathsf{E}_2] \oplus b|y \odot [(b^* \mid 1)\mathsf{E}_2] \end{aligned}$$

Its derived-term automaton is:



It is straightforward to extract an algorithm from Definition 10, using a work-list of states whose outgoing transitions to compute (see Algorithm 1). This approach admits a natural lazy implementation: the whole automaton is not computed at once, but rather, states and transitions are computed on-the-fly, on demand, for instance when evaluating a word, or during a composition or a shortest path traversal, etc. One can apply transformations on the expansion *before* extracting transitions from it, for instance to generate deterministic/sequential automata [8, Section 4.2].

5 Support for Composition

Our goal is to introduce a composition operator in rational expressions, so that, for instance, $a|x \mathbin{@} x|c$ is equivalent to $a|c$. The construct is not limited

Input : E, a rational expression
Output : $\langle E, I, T\rangle$ an automaton (simplified notation)

```
I(E) := 1_K ;                                  // Unique initial state
Q := Queue(E) ;                                // A work list loaded with E
while Q is not empty do
    E := pop(Q) ;                              // A new state/expression to complete
    X := d(E) ;                                // Compute the expansion of E
    T(E) := X_$ ;                              // Final weight: the constant term
    foreach a ⊙ [X(a)] ∈ X do                  // For each (first, polynomial) in X
        foreach ⟨k⟩ ⊙ F ∈ X(a) do               // For each monomial of X(a)
            E(E, a, F) := k ;                  // New transition
            if F ∉ Q then                      // F is a new state...
                push(Q, F) ;                   // ... to complete later
```

Algorithm 1: Building the derived-term automaton. The set of states is implicitly grown when transitions are added.

to two-tape automata, and the 'zipping' could be performed on any tape, so for instance $a|x|c(1 \mathbin{@} 2)A|B|x$ would denote $a|A|B|c$ (or $a|c|A|B$?). To avoid useless complications, we limit the presentation to the simple case of two-tape expressions, where composition 'zips' the last tape of the left with the first of the right. We also require both tapes to have the same type. As a consequence, composition is an internal law.

Let A be an alphabet. By $A^?$ we denote $\{\varepsilon\} \cup A$. We use $a, b, \ldots$ to denote letters of A, and ℓ, ℓ' to denote labels of $A^?$.

5.1 Composition of Rational Series

Let A be an alphabet, and $s, t \in \mathbb{K}\langle\langle A^* \times A^*\rangle\rangle$ two series. The composition of s with t is the series $s \mathbin{@} t := m|n \mapsto \sum_{x \in M} s(m|x) \cdot t(x|n)$, which also belongs to $\mathbb{K}\langle\langle A^* \times A^*\rangle\rangle$.

Proposition 6 (Series Composition is Bilinear)
For all series $s, s', t, t' \in \mathbb{K}\langle\langle A^* \times A^*\rangle\rangle$, and all weights $k \in \mathbb{K}$,

$$(s + s') \mathbin{@} t = s \mathbin{@} t + s' \mathbin{@} t \qquad s \mathbin{@} (t + t') = s \mathbin{@} t + s \mathbin{@} t'$$
$$(ks) \mathbin{@} t = k(s \mathbin{@} t) \qquad s \mathbin{@} (kt) = k(s \mathbin{@} t)$$

Proposition 7 *For all series $s, t \in \mathbb{K}\langle\langle A^* \times A^* \rangle\rangle$, and labels $\ell, \ell', \ell_1, \ell_2 \in A^?$,*

$$((\ell_1|\ell)\cdot s) \mathbin{@} ((\ell'|\ell_2)\cdot t) = (\ell_1|\ell_2)\cdot \begin{cases} s \mathbin{@} t & \text{if } \ell = \ell' \\ ((\varepsilon|\ell)\cdot s) \mathbin{@} t & \text{if } \ell \neq \varepsilon, \ell' = \varepsilon \\ s \mathbin{@} ((\ell'|\varepsilon)\cdot t) & \text{if } \ell = \varepsilon, \ell' \neq \varepsilon \\ 0 & \text{otherwise} \end{cases}$$

Proof: With the convention that terms with undefined words (e.g., $a^{-1}b$) are null, we have:

$$\big(((\ell_1|\ell)\cdot s) \mathbin{@} ((\ell'|\ell_2)\cdot t)\big)(m|n) = \sum_{x\in A^*} ((\ell_1|\ell)\cdot s)(m|x)((\ell'|\ell_2)\cdot t)(x|n)$$

$$= \sum_{x\in A^*} s(\ell_1^{-1}m|\ell^{-1}x)t(\ell'^{-1}x|\ell_2^{-1}n) = (\ell_1|\ell_2)\sum_{x\in A^*} s(m|\ell^{-1}x)t(\ell'^{-1}x|n)$$

Then we reason by cases:

- if $\ell = \ell'$, then:

$$\sum_{x\in A^*} s(m|\ell^{-1}x)t(\ell^{-1}x|n) = \sum_{y\in A^*} s(m|y)t(y|n) = (s \mathbin{@} t)(m|n)$$

- if $\ell \neq \varepsilon$ and $\ell' = \varepsilon$, then:

$$\sum_{x\in A^*} s(m|\ell^{-1}x)t(\ell'^{-1}x|n) = \sum_{x\in A^*} ((\varepsilon|\ell)\cdot s)(m|x)(t)(x|n) = \big(((\varepsilon|\ell)\cdot s) \mathbin{@} t\big)(m|n)$$

- the case $\ell = \varepsilon$ and $\ell' \neq \varepsilon$ is similar.
- if $\ell \neq \ell'$ and neither is the empty word, then at least one of $\ell^{-1}x$ or $\ell'^{-1}x$ is undefined. □

5.2 Composition of Weighted Rational Expressions

To the Definition 1, we add a clause $\mathsf{E} ::= \mathsf{E} \mathbin{@} \mathsf{E}$. Its semantics is defined by $[\![\mathsf{E} \mathbin{@} \mathsf{F}]\!] := [\![\mathsf{E}]\!] @ [\![\mathsf{F}]\!]$. Its trivial identities are:

$$\mathsf{E} \mathbin{@} 0 \Rightarrow 0 \qquad 0 \mathbin{@} \mathsf{E} \Rightarrow 0 \qquad (\langle k\rangle^? 1) \mathbin{@} (\langle h\rangle^? 1) \Rightarrow \langle kh\rangle^? 1$$

where, as in Definition 3, $\langle k\rangle^? 1$ denotes either $\langle k\rangle 1$, or 1 in which case $k = 1_{\mathbb{K}}$ in the right-hand side of $\Rightarrow$.

The definition of composition of polynomials follows from the bilinearity of composition.

$$\mathsf{P} \mathbin{@} \mathsf{Q} \coloneqq \bigoplus_{(i,j)\in[n]\times[m]} \langle k_i \cdot h_j\rangle \odot \mathsf{E}_i \mathbin{@} \mathsf{F}_j$$

With a proof similar to the case of $|$ in Lemma 1, we can prove that for any polynomials P and Q, $[\![\mathsf{P} \mathbin{@} \mathsf{Q}]\!] = [\![\mathsf{P}]\!] \mathbin{@} [\![\mathsf{Q}]\!]$.

5.3 Composition of Rational Expansions

The definition of expansion composition may look straightforward: just 'zip' on the common letters, and compose the corresponding expressions. For instance $\big((a|x) \odot [\mathsf{E}] \oplus (a|b) \odot [\mathsf{E}']\big) \mathbin{@} \big((x|b) \odot [\mathsf{F}] \oplus (a|b) \odot [\mathsf{F}']\big)$ results in $(a|b) \odot [\mathsf{E} \mathbin{@} \mathsf{F}]$: only x appears both in output and input.

However, the empty word makes things more interesting. Consider for instance $\big(a|\varepsilon \odot [\varepsilon|x]\big) \mathbin{@} \big(x|b \odot [\varepsilon|b]\big)$: it denotes $(a|x) \mathbin{@} (x|b) \equiv a|b$. Therefore the empty word must be 'zippable' with any other label; this applies to the empty-word as output of the left-hand side: $(a|\varepsilon \odot [\mathsf{P}]) \mathbin{@} (x|b \odot [\mathsf{Q}]) \Rightarrow a|\varepsilon \odot [\mathsf{P} \mathbin{@} ((x|b)\mathsf{Q})]$, and as input of the right-hand side: $(a|x \odot [\mathsf{P}]) \mathbin{@} (\varepsilon|b \odot [\mathsf{Q}]) \Rightarrow \varepsilon|b \odot [((a|x)\mathsf{P}) \mathbin{@} \mathsf{Q}]$. However, we must be careful not to pair twice $a|\varepsilon$ with $\varepsilon|b$, once for ε as right output label and once for ε as left input label: that would denote $\langle 2\rangle a|b$ instead of $a|b$. Hence the following definition:

$$\mathsf{X} \mathbin{@} \mathsf{Y} \coloneqq \begin{cases} \langle \mathsf{X}_{\$} \mathsf{Y}_{\$}\rangle \\ \quad \oplus \displaystyle\bigoplus_{\varepsilon|\ell_2 \in f(\mathsf{Y})} (\varepsilon|\ell_2) \odot [\langle \mathsf{X}_{\$}\rangle (1 \mathbin{@} \mathsf{Y}_{\varepsilon|\ell_2})] \\ \quad \oplus \displaystyle\bigoplus_{\ell_1|\varepsilon \in f(\mathsf{X})} (\ell_1|\varepsilon) \odot [\langle \mathsf{Y}_{\$}\rangle (\mathsf{X}_{\ell_1|\varepsilon} \mathbin{@} 1)] \\ \quad \oplus \displaystyle\bigoplus_{\substack{\ell_1|\ell \in f(\mathsf{X}) \\ \ell'|\ell_2 \in f(\mathsf{Y})}} (\ell_1|\ell_2) \odot \begin{bmatrix} \mathsf{X}_{\ell_1|\ell} \mathbin{@} \mathsf{Y}_{\ell'|\ell_2} & \text{if } \ell = \ell' \\ \mathsf{X}_{\ell_1|\ell} \mathbin{@} (\ell'|\varepsilon)\mathsf{Y}_{\ell'|\ell_2} & \text{if } \ell = \varepsilon, \ell' \neq \varepsilon \\ (\varepsilon|\ell)\mathsf{X}_{\ell_1|\ell} \mathbin{@} \mathsf{Y}_{\ell'|\ell_2} & \text{if } \ell \neq \varepsilon, \ell' = \varepsilon \end{bmatrix} \end{cases} \tag{15}$$

where ℓ, ℓ_1, and ℓ_2, are labels.

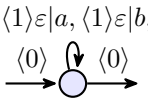
The following lemma is proved using Propositions 6 and 7.

Lemma 8 $[\![\mathsf{X} \mathbin{@} \mathsf{Y}]\!] = [\![\mathsf{X}]\!] \mathbin{@} [\![\mathsf{Y}]\!]$.

To compute the expansion of an expression with composition, Definition 8 only needs one additional case:

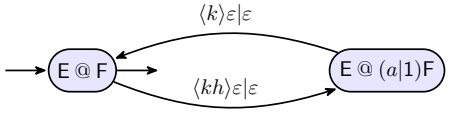
$$d(\mathsf{E} \mathbin{@} \mathsf{F}) \coloneqq d(\mathsf{E}) \mathbin{@} d(\mathsf{F})$$

Example 7 *Consider the introductory example in* $\mathbb{Z}_{\min}$. *Let* $\mathsf{F} \coloneqq (a|a+b|b+\langle 1\rangle(1|I+[ab]|S))^* \mathbin{@} (a|a+b|b+I|[ab]+S|1)^*$. *Its derived-term automaton is exactly the automaton from Mohri [20, Figure 4]:*



Expansions of expressions with composition may be not normal, which will result in derived-term automata with spontaneous transitions.

Example 8 *Let* $\mathsf{E} \coloneqq \big(\langle k\rangle 1 \mid a\big)^*$ *and* $\mathsf{F} \coloneqq \big(\langle h\rangle aa \mid 1\big)^*$. *The derived-term automaton of* $\mathsf{E} \mathbin{@} \mathsf{F}$ *is:*



Theorem 3 (Theorem 2 with Composition) *If valid, any multitape expression with compositions* E *and its expansion-based derived-term automaton* $\mathcal{A}_{\mathsf{E}}$ *denote the same series, i.e.,* $[\![\mathcal{A}_{\mathsf{E}}]\!] = [\![\mathsf{E}]\!]$.

Proof: Because it already 'supports' automata with spontaneous transitions, the proof of Theorem 2 still applies here. We must however justify that the automaton is indeed finite.

With the convention that $(\varepsilon|\varepsilon)\mathsf{E} = \mathsf{E}$, we can define the proper derived terms of $\mathsf{E} \mathbin{@} \mathsf{F}$ as: $\mathrm{PD}(\mathsf{E} \mathbin{@} \mathsf{F}) \coloneqq \{\varepsilon|\ell\}_{\ell\in A^?}\mathrm{PD}(\mathsf{E}) \mathbin{@} \{\ell|\varepsilon\}_{\ell\in A^?}\mathrm{PD}(\mathsf{F})$. With this definition, Lemmas 5 and 6 apply to expressions with compositions, which proves that the set of derived terms of an multitape expression with composition is finite.□

Contrary to expressions without composition, the procedure may successfully build an invalid automaton. For a start, consider the automaton of Example 8. This automaton is valid in $\mathbb{B}$ (as any automaton…), but might be in $\mathbb{Q}$ depending on the starrability of the weight k^2h. In all the cases, the validity of the automaton is equivalent to the validity of the expression. Unfortunately, there exists cases where this procedure builds invalid automata from valid expressions (Section 6.1).

6 Discussion

This section addresses several issues:

- not all the generated automata are valid (Section 6.1),
- the expressions may be normalized before and during the computations (Section 6.2),
- the computations can be simplified by relying more on spontaneous transitions, at the cost of creating useless states (Section 6.3),
- it is possible to keep simple computation and generate the same automata (Section 6.4),
- an efficient implementation of the procedure must pay attention to some issues (Section 6.5),
- the generated automata are small (Section 6.6),
- the tupling operator can be supported by the derivative-based computation of the derived-term automaton (Section 6.7).

6.1 On the Validity of Automata

The computation of the expansion of product and star of expressions are quite involved:

$$d(\mathsf{E}\cdot\mathsf{F}) \coloneqq d_p(\mathsf{E})\cdot\mathsf{F} \oplus \langle d_\$(\mathsf{E})\rangle d(\mathsf{F}) \qquad \text{(Eq. (10))}$$

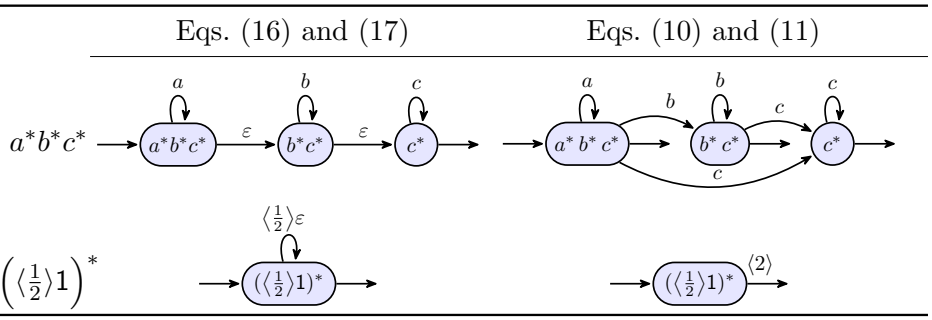
$$d(\mathsf{E}^*) \coloneqq \langle d_\$(\mathsf{E})^*\rangle \oplus \langle d_\$(\mathsf{E})^*\rangle d_p(\mathsf{E})\cdot\mathsf{E}^* \qquad \text{(Eq. (11))}$$

whereas some simpler versions enjoying the freedom to use non-normal expansions (see Section 3.2) suffice:

$$d(\mathsf{E}\cdot\mathsf{F}) \coloneqq d(\mathsf{E})\cdot\mathsf{F} \qquad (16)$$

$$d(\mathsf{E}^*) \coloneqq \langle 1_{\mathbb{K}}\rangle \oplus d(\mathsf{E})\cdot\mathsf{E}^* \qquad (17)$$

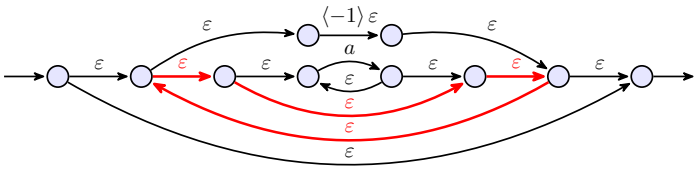
They generate arguably more natural automata, but with spontaneous transitions.



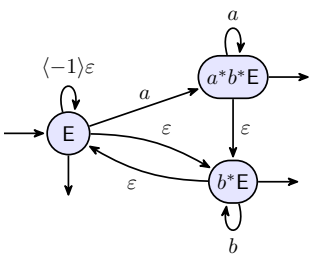
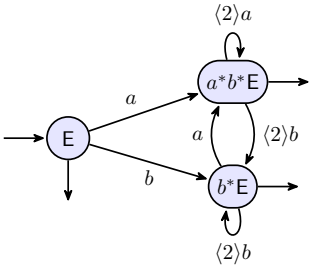
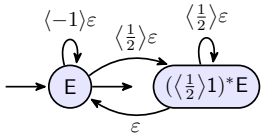
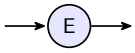
Also, a significant advantage of Eq. (17) over Eq. (11) is that its correctness is straightforward to prove (it follows from $s^* = 1 + ss^*$), while justifying Eq. (11) required the Super S property (see Propositions 1 and 2), whose proof involve topological arguments. With Eqs. (16) and (17) all these 'details' are delegated to the spontaneous transition removal procedure, as discussed by Lombardy and Sakarovitch [17] for instance.

Therefore, Eqs. (10) and (11) may appear as mere optimizations of Eqs. (16) and (17): they generate automata that have fewer spontaneous transitions.

Alas, we are then exposed to the same problems as the techniques that start from the Thompson automaton to compute different types of automata [1]: for some valid expressions, we generate invalid automata. For instance in $\mathbb{Q}$, the expression $(a^* + \langle -1\rangle 1)^*$ is valid, as $[\![a^* + \langle -1\rangle 1]\!]$ is proper, yet its Thompson automaton is invalid, as it contains a spontaneous cycle whose weight, 1, is not starrable:

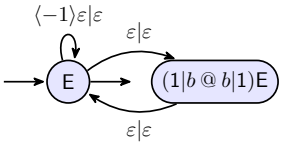


The following expressions show invalid automata built by Eqs. (16) and (17) and the corresponding valid ones using Eqs. (10) and (11).

	Eqs. (16) and (17)	Eqs. (10) and (11)
$\big(a^*b^* + \langle -1\rangle 1\big)^*$		
$\Big(\big(\langle \frac{1}{2}\rangle 1\big)^* + \langle -2\rangle 1\Big)^*$		

Since it may also generate non-normal expansions, our handling of the composition may generate invalid automata from valid expressions too; for

instance with $\mathsf{E} \coloneqq (1|ab \mathbin{@} ab|1 + \langle -1\rangle 1|1)^*$:

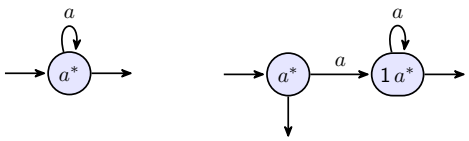


However, this can never happen in positive semirings.

6.2 Identities on Expressions

Small is beautiful. The smaller the automaton, the better. Therefore, it is natural to be eager at simplifying the expressions, and apply all the possible transformations that help reducing the size of the automaton [22].

Yet we chose relatively few identities: basically, those of Definition 3 are about the constants $(\mathsf{0}, \mathsf{1}, 0_{\mathbb{K}}, 1_{\mathbb{K}})$. The identities were chosen to avoid useless clutter in the examples. Compare for instance the derived-term automaton of a^* with, and without the trivial identity $\mathsf{1} \cdot \mathsf{E} \Rightarrow \mathsf{E}$:



Eliminating zeroes $(\mathsf{0}, 0_{\mathbb{K}})$ allows to accept expressions that contain an invalid but useless part. For instance $a + \langle 0\rangle \mathsf{1}^* + \mathsf{0}\mathsf{1}^*$ processed without identities would fail in $\mathbb{Q}$. Simplifying zeros also avoids creating non-coaccessible states (consider $abc\mathsf{0}$ for instance).

However, *none* of the trivial identities is needed for the procedure to terminate: everything is taken care of by the polynomials. Identities are not even needed to guarantee that expansions of expressions (without composition) are normal, i.e., that the derived-term automaton is proper.

6.3 Denormalized Expressions

Both theory and implementation are simpler with *denormalized expansions*, whose immediate constant term is moved into the terms of the empty word. Consider for instance the expansion of $a^*\langle 2\rangle$:

$$\langle 2\rangle \oplus a \odot [\langle 1_{\mathbb{K}}\rangle \odot a^*\langle 2\rangle] \qquad \text{normal expansion} \qquad (18)$$

$$\varepsilon \odot [\langle 2\rangle \odot 1] \oplus a \odot [\langle 1_{\mathbb{K}}\rangle \odot a^*\langle 2\rangle] \qquad \text{denormalized expansion} \qquad (19)$$

With denormalized expansions, Eqs. (3) to (6) and (15) can be simplified into:

$$\mathsf{X} \oplus \mathsf{Y} \coloneqq \bigoplus_{\ell \in f(\mathsf{X}) \cup f(\mathsf{Y})} \ell \odot [\mathsf{X}_\ell \oplus \mathsf{Y}_\ell]$$

$$\langle k \rangle \mathsf{X} \coloneqq \bigoplus_{\ell \in f(\mathsf{X})} \ell \odot [\langle k \rangle \mathsf{X}_\ell] \qquad \mathsf{X}\langle k \rangle \coloneqq \bigoplus_{\ell \in f(\mathsf{X})} \ell \odot [\mathsf{X}_\ell \langle k \rangle]$$

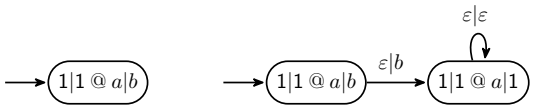
$$\mathsf{X} \cdot \mathsf{E} \coloneqq \bigoplus_{\ell \in f(\mathsf{X})} \ell \odot [\mathsf{X}_\ell \cdot \mathsf{E}]$$

$$\mathsf{X} \mid \mathsf{Y} \coloneqq \bigoplus_{\ell \in f(\mathsf{X}), \ell' \in f(\mathsf{Y})} (\ell|\ell') \odot (\mathsf{X}_\ell \mid \mathsf{Y}_{\ell'})$$

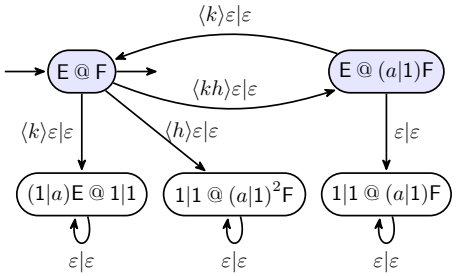
$$\mathsf{X} \mathbin{@} \mathsf{Y} \coloneqq \bigoplus_{\substack{\ell_1|\ell \in f(\mathsf{X}) \\ \ell'|\ell_2 \in f(\mathsf{Y})}} (\ell_1|\ell_2) \odot \begin{bmatrix} \mathsf{X}_{\ell_1|\ell} \mathbin{@} \mathsf{Y}_{\ell'|\ell_2} & \text{if } \ell = \ell' \\ \mathsf{X}_{\ell_1|\ell} \mathbin{@} (\ell'|\varepsilon)\mathsf{Y}_{\ell'|\ell_2} & \text{if } \ell = \varepsilon, \ell' \neq \varepsilon \\ (\varepsilon|\ell)\mathsf{X}_{\ell_1|\ell} \mathbin{@} \mathsf{Y}_{\ell'|\ell_2} & \text{if } \ell \neq \varepsilon, \ell' = \varepsilon \end{bmatrix}$$

With adjusted definitions of immediate constant term (the weight associated to $\mathsf{1}$ in the term of ε) and immediate proper part, the remainder of the automaton construction procedure remains the same.

However, this introduces new terms in the case of composition. Consider expansions $\mathsf{X} \coloneqq \langle 1 \rangle$ and $\mathsf{Y} \coloneqq a|b \odot [\mathsf{1}]$. Their (normal) composition with no identities is $\mathsf{0}$. The denormalized expansion of X is $\mathsf{X}' \coloneqq \varepsilon|b \odot [\varepsilon|\varepsilon \mathbin{@} a|\varepsilon]$. These results yield two different automata:



The second automaton includes a (useless) spontaneous loop which is invalid in $\mathbb{Q}$ for instance. In this case, stronger identities on the composition simplifies the expression $\mathsf{1}|\mathsf{1}@a|b$ into $\mathsf{0}$, which solves the problem, but in other cases these spurious transitions remain. For instance, the expression from Example 8 yields the following automaton with denormalized expansions:



With denormalized expansions, the introductory example (Example 7) yields eight such useless states, in addition to the single useful one.

6.4 The Endmarker

Both types of expansions (denormalized or not) have different pros and cons.

With (non denormalized) expansions (Eq. (18)) the constant term has a different nature from the rest of the expansion, which lacks elegance. The equations are somewhat complex.

With denormalized expansions (Eq. (19)) the immediate constant term is buried with other derived terms following the empty word, which leads to convoluted definitions of the immediate constant term and of the immediate proper part. The interpretation of the latter is somewhat clumsy: sometimes they denote final states, sometimes spontaneous transitions. Besides, hiding the (immediate) constant term in the derived terms of ε blurs the important distinction between normal expansions (that yield automata without spontaneous transitions) and non normal ones. Finally, the generated automata may have many useless states (Section 6.3).

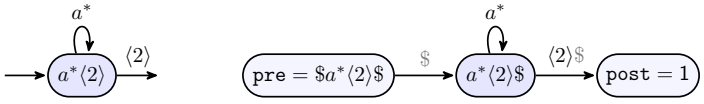
These concerns can be addressed if we introduce an *endmarker* (aka, *end of tape symbol*, or *terminator*), \$, added at the end of the expression. For instance the expansion of $a^*\langle 2\rangle\$$ is:

$$\$ \odot [\langle 2\rangle \odot 1] \oplus a \odot [\langle 1_{\mathbb{K}}\rangle \odot a^*\langle 2\rangle] \qquad \text{with endmarker}$$

to compare with Eqs. (18) and (19). (Expressions/automata with/without endmarker are equivalent [24, Proposition IV.5.1 p. 579].) We keep the simpler and more regular equations of denormalized expansions: the constant term becomes a regular weight, associated to the only possible derived term in the polynomial of \$: 1. And we also avoid the useless states, as with non-denormalized expansions.

This also allows to simplify the construction of the derived-term automaton. In Vcsn for example [10], the initial and final weights are implemented

as weights of special transitions from the unique preinitial state to the initial states, and from the final states to the unique postfinal state, all labeled with the endmarker (which therefore also servers as a beginmarker).



In the mathematical definition of an automaton, this corresponds to the replacement of the initial and final functions, I and T, by two constants: the `pre` and `post` states. Starting with an endmarker at both ends, $\mathsf{E}$$, Definition 10 can then be simplified as:

- Q is the set of rational expressions on alphabet A with weights in $\mathbb{K}$,
- $E(\mathsf{F}, \ell, \mathsf{F}') = k$ iff $\langle k \rangle \odot \mathsf{F}' \in d(\mathsf{F})(\ell)$, for all labels $\ell \in \{\$, \varepsilon\} \cup A$.

Using the endmarker, the simplified definitions of Section 6.3 can be used in place of Eqs. (3) to (6) and (15) and yield the same automata. This vastly simplifies the implementation.

6.5 Implementation Issues

In an implementation, a single recursive call to $d(\mathsf{E})$ suffices for Eqs. (10) and (11), from which $d_\$(\mathsf{E})$ and $d_p(\mathsf{E})$ are obtained; expansions are computed only when needed. So they should rather be written:

$$d(\mathsf{E} \cdot \mathsf{F}) \coloneqq \texttt{let}\ \mathsf{X} = d(\mathsf{E})\ \texttt{in}\ \texttt{if}\ \langle \mathsf{X}_\$ \rangle \neq 0_{\mathbb{K}}\ \texttt{then}\ \mathsf{X}_p \cdot \mathsf{F} \oplus \langle \mathsf{X}_\$ \rangle d(\mathsf{F})\ \texttt{else}\ \mathsf{X}_p \cdot \mathsf{F}$$
$$d(\mathsf{E}^*) \coloneqq \texttt{let}\ \mathsf{X} = d(\mathsf{E})\ \texttt{in}\ \left\langle \mathsf{X}^*_\$ \right\rangle \oplus \left\langle \mathsf{X}^*_\$ \right\rangle \mathsf{X}_p \cdot \mathsf{E}^*$$

Besides, existing expressions are referenced to, not duplicated. In the previous piece of code, E^* is not built again, the input argument is reused.

Identities that enforce right-associativity of the product are a strong optimization that saves recursive calls. Consider $((ab)c)d$; computing its expansion requires that of $(ab)c$ is needed, which requires that of (ab) which requires that of a, which is $a \odot [\langle 1_{\mathbb{K}} \rangle \odot 1]$, that we multiply by b to get $a \odot [\langle 1_{\mathbb{K}} \rangle \odot 1b]$, then multiplied by c, and finally by d, which results in $a \odot [\langle 1_{\mathbb{K}} \rangle \odot ((1b)c)d]$. Note that $((1b)c)d$ is still left-associative and will require similarly deep computations. On the contrary, the expansion of $a(b(cd))$ is computed in a single step: $a \odot [\langle 1_{\mathbb{K}} \rangle \odot 1(b(cd))]$.

At each step of the construction of the derived-term automaton we compute the expansion of an expression, extract its terms and add transitions to the states of these terms. It it therefore critical to use an efficient structure

Expression	Derived-Term #S	Derived-Term #T	Inductive #S	Inductive #T
$([ab] + \langle 1\rangle(\varepsilon \mid [ab] + [ab] \mid \varepsilon))^*$	1	6	7	42
$[ab]^*(\langle 2\rangle(a\|b + b\|a) + \langle 1\rangle(\varepsilon \mid [ab] + [ab] \mid \varepsilon))^*$	2	14	9	60
$[ab] + \langle 1\rangle(\varepsilon\|I + (a + b) \mid S))^*$	1	5	6	30
$([ab] + S\|\varepsilon + I \mid [ab])^*$	1	5	6	30
$([ab] + \langle 1\rangle(\varepsilon\|I + [ab] \mid S))^*@([ab] + S\|\varepsilon + I \mid [ab])^*$	1	6	7	42
$\langle 4\rangle ade^*\|x + \langle 3\rangle bde^*\|x + \langle 2\rangle ace^*\|xy + \langle 6\rangle bce^*\|xy$	4	7	13	16
$a + \langle 2\rangle(bc^*)$	3	3	4	4
$a^* \mid b^* \mid c^* \mid d^* \mid e^*$	31	211	32	242
$(a^+ \mid x + b^+ \mid y)^*$	3	8	5	14
$(\langle k\rangle\varepsilon \mid a)^*@(\langle h\rangle aa \mid \varepsilon)^*$	2	2	3	3

Table 1: Number of states and of transitions of the derived-term and inductive automata for the expressions used in this paper. We used traditional abbreviations, implemented in Vcsn: $[ab] \coloneqq a + b$, and single-tape expressions in multitape context denote partial identities, e.g., $a + \langle 2\rangle(bc^*)$ denotes $(a|a) + \langle 2\rangle((b|b)(c|c)^*)$.

to store and retrieve the derived terms. Hash tables are well suited for this task.

6.6 Performances

We claimed that this construction builds small automata. On single tape expressions, it is well known that the size of the standard automaton (aka Glushkov automaton) of an expression E is exactly $\|\mathsf{E}\| + 1$ [6], and that the derived-term automaton is *at most* $\|\mathsf{E}\| + 1$ but 'often' much smaller.

In Vcsn we implemented `inductive`, a generalization of the recursive implementation of the computation of the standard automaton of an expression (see Lombardy and Sakarovitch [16, pp. 163-164] for instance) with support for the | and @ operators, and compared the sizes of the automata for the expressions used as examples in this paper. The results are presented in Table 1. The derived-term automaton has never more states or transitions on these examples.

Benchmarks ran on generated expressions show similar results (see `http://vcsn.lrde.epita.fr/dload/2.6/notebooks/SACS-2017.html`). There, the speed of both implementations are also compared.

6.7 Multitape Derivatives

We reproduce here the definition of constant terms and derivatives from Lombardy et al [16, p. 148 and Def. 2], with our notations and added support for multitape expressions. To facilitate reading, weights such as the constant term are written in angle brackets, although so far this was reserved to syntactic constructs.

Definition 11 (Constant Term and Derivative)

$$
\begin{aligned}
c(\mathsf{0}) &\coloneqq \langle 0_{\mathbb{K}}\rangle, & \partial_a \mathsf{0} &\coloneqq \mathsf{0}, && (20)\\
c(\mathsf{1}) &\coloneqq \langle 1_{\mathbb{K}}\rangle, & \partial_a \mathsf{1} &\coloneqq \mathsf{0}, &&\\
c(a) &\coloneqq \langle 0_{\mathbb{K}}\rangle, \forall a \in A, & \partial_a b &\coloneqq \mathsf{1} \textit{ if } b = a, \mathsf{0} \textit{ otherwise}, && (21)\\
c(\mathsf{E}+\mathsf{F}) &\coloneqq c(\mathsf{E}) + c(\mathsf{F}), & \partial_a(\mathsf{E}+\mathsf{F}) &\coloneqq \partial_a\mathsf{E} \oplus \partial_a\mathsf{F}, && (22)\\
c(\langle k\rangle\mathsf{E}) &\coloneqq \langle k\rangle c(\mathsf{E}), & \partial_a(\langle k\rangle\mathsf{E}) &\coloneqq \langle k\rangle(\partial_a\mathsf{E}), && (23)\\
c(\mathsf{E}\cdot\mathsf{F}) &\coloneqq c(\mathsf{E})\cdot c(\mathsf{F}), & \partial_a(\mathsf{E}\cdot\mathsf{F}) &\coloneqq (\partial_a\mathsf{E})\cdot\mathsf{F} \oplus \big\langle c(\mathsf{E})\big\rangle\partial_a\mathsf{F}, && (24)\\
c(\mathsf{E}^*) &\coloneqq c(\mathsf{E})^*, & \partial_a\mathsf{E}^* &\coloneqq \big\langle c(\mathsf{E})^*\big\rangle(\partial_a\mathsf{E})\cdot\mathsf{E}^*, && (25)\\
c(\mathsf{E}\mid\mathsf{F}) &\coloneqq c(\mathsf{E})\cdot c(\mathsf{F}), & \partial_{a|b}(\mathsf{E}\mid\mathsf{F}), &\coloneqq \partial_a\mathsf{E}\mid\partial_b\mathsf{F}, && (26)\\
& & \partial_{a|\varepsilon}(\mathsf{E}\mid\mathsf{F}), &\coloneqq \big\langle c(\mathsf{F})\big\rangle(\partial_a\mathsf{E}\mid\mathsf{1}),\\
& & \partial_{\varepsilon|b}(\mathsf{E}\mid\mathsf{F}), &\coloneqq \big\langle c(\mathsf{E})\big\rangle(\mathsf{1}\mid\partial_b\mathsf{F}).
\end{aligned}
$$

where Eq. (25) *applies iff* $c(\mathsf{E})^*$ *is defined in* $\mathbb{K}$.

From an implementation point of view, Eq. (26) leads to repeated computations of $\partial_a\mathsf{E}$ and of $\partial_b\mathsf{F}$, unless one would cache them, but that's what expansions do.

Lemma 9 *For any expression* E *(without composition),* $d(\mathsf{E})(\varepsilon) = c(\mathsf{E})$*, and* $d(\mathsf{E})(a) = \partial_a\mathsf{E}$*.*

Proof: A straightforward induction on E. The cases of constants and letters are immediate consequences of Eqs. (20) and (21) on the one hand, and Eq. (7) on the other hand. Equation (8) matches Eqs. (22) and (23). Multiplication (concatenation) is again barely a change of notation between Eq. (10) and Eq. (24), and likewise for the Kleene star (Eqs. (11) and (25)) and tuple (Eqs. (12) and (26), using Eq. (6)). □

Note that, if we were to define the derivative with respect to the empty word as the constant term, i.e., $\partial_\varepsilon\mathsf{E} \coloneqq c(\mathsf{E})$, then the previous definition

would simplify, for some operators, to:

$$\partial_\ell(\mathsf{E}+\mathsf{F}) \coloneqq \partial_\ell\mathsf{E}+\partial_\ell\mathsf{F} \quad \partial_\ell(\langle k\rangle\mathsf{E}) \coloneqq \langle k\rangle(\partial_\ell\mathsf{E}) \quad \partial_{\ell|\ell'}(\mathsf{E}\mid\mathsf{F}) \coloneqq \partial_\ell(\mathsf{E})\mid\partial_{\ell'}(\mathsf{F})$$

where for any weights $k, k', k \mid k' \coloneqq k \cdot k'$.

Note that these derivatives are no longer equivalent to the left quotient of the corresponding language. Consider $\mathsf{F} \coloneqq (a^* \mid 1)(a^+ \mid x + b^+ \mid y)^*$: the language it denotes includes $ab|y$, yet $\partial_{a|y}\mathsf{F} = \langle 0_{\mathbb{K}}\rangle$. Albeit surprising, this result is nevertheless sufficient as can be observed in the derived-term automaton in Example 6: while the state $(a^* \mid 1)(a^+ \mid x + b^+ \mid y)^*$ does accept words starting with a on the first tape, and y on the second, an outgoing transition on $a|y$ would result in a more complex automaton.

7 Related Work

This paper is about an algorithm to convert an expression into automata, and more specifically about multitape expressions.

7.1 From Expression to Automaton

Automata and rational (or regular) expressions share the same expressive power [14]. This fact made rational expressions an extremely handy practical tool to specify some rational languages in a concise way, from which acceptors (automata) are built [25].

There are numerous algorithms to build an automaton from an expression starting with Glushkov [12], McNaughton and Yamada [19]. Brzozowski [4] introduced the idea of derivatives of expressions as a means to construct an equivalent automaton. The method applies to extended (unweighted) rational expressions, and constructs a deterministic automaton. Antimirov [3] modified the computation to rely on parts of the derivatives ('partial derivatives'), which results in nondeterministic automata.

Lombardy and Sakarovitch [16] extended this approach to support weighted expressions; independently; with different foundations, Rutten [23] proposed a similar construction. Caron et al. [5] introduced support for (unweighted) extended expressions. Demaille [8] provides support for weighted extended expressions; expansions, originally mentioned by Brzozowski [4], are placed at the center of the construct, replacing derivatives, to gain independence with respect to the size of the alphabet, and efficiency.

We are particularly interested in the derivative-based family of algorithms, because they offer a very natural interpretation to states (they are labeled by an expression that denotes the future of the states, i.e., the language/series accepted from this state), and provide easy support for on-the-fly conversion.

7.2 Multitape Expressions

Multitape automata, including transducers, share many properties with 'single-tape' automata, in particular the Fundamental Theorem [24, Theorem 2.1, p. 409]: under appropriate conditions, multitape automata and rational (multitape) series share the same expressive power.

Multitape rational expressions have been considered early [18], but "an n-way regular expression is simply a regular expression whose terms are n-tuples of alphabetic symbols or ε" [13], e.g., $(\varepsilon|a+\varepsilon|b)^*$, but not $(\varepsilon|(a+b))^*$. Kaplan and Kay [13] do consider the full generality of the semantics of operations on rational languages and rational relations, including $\times$, the Cartesian product of languages, and even use rational expressions more general than their definition. They do not, however, provide an explicit automaton construction algorithm, apparently relying on the simple inductive construction (using the Cartesian product between automata). Our $|$ operator on series was defined as the *tensor product*, denoted $\otimes$, by Sakarovitch [24, Sec. III.3.2.5], but without equivalent for expressions.

Makarevskii and Stotskaya [18] define multitape derivatives, but (i) in the case of expressions over tuples of letters, and (ii) only when in so-called 'standard form', for which he notes "no method of constructing [an] n-expression in standard form for a regular n-expression is known."

We first introduced multitape expressions and their derived-term automata in Demaille [9]. This paper extends this work: the base theory is generalized to support spontaneous transitions, which we used in Section 5 to introduce support for composition.

Constructions of the derived-term automaton with completely different grounds have been discovered [1, 7]: they do not rely on derivatives at all. It is an open question whether these approaches can be adapted to support a tuple or a composition operator.

8 Conclusion

Our work is in the continuation of derivative-based computations of an automaton from an expression [3–5, 16]. However, we replaced the derivatives by expansions, which lifted the requirement for the monoid of labels to be free.

This freedom allowed us to generalize the computation of the derived-term automaton to expressions with a tupling operator $(a|b)$ and a composition operator $(a|x \mathbin{@} x|b)$. This procedure generates small automata.

Compared to the derivative-based approach, expansions allowed simpler proofs, and a more efficient implementation.

Vcsn[5] implements the techniques exposed in this paper. Our future work aims at other operators, and studying more closely the complexity of the algorithm.

Acknowledgments

The author thanks Alexandre Duret-Lutz, Sylvain Lombardy, Luca Saiu and Jacques Sakarovitch for their feedback during this work, and the anonymous reviewers for their precise and constructive comments.

This would not have been possible without Vcsn; many thanks to its contributors: Alexandre Lewkowicz, Alfred M. Szmidt, Antoine Pietri, Canh Luu, Clément Démoulins, Clément Gillard, David Moreira, Guillaume Sanchez, Lucien Boillod, Nicolas Barray, Raoul Billion, Roland Levillain, Sarasvati Moutoucomarapoulé, Sébastien Piat, Thibaud Michaud, Valentin Tolmer, Victor Marie Santet, Yann Bourgeois–Copigny, and last but not least, Younes Khoudli.

References

1. C. Allauzen and M. Mohri. A unified construction of the Glushkov, follow, and Antimirov automata. In R. Královič and P. Urzyczyn, editors, *Mathematical Foundations of Computer Science*, volume 4162 of *Lecture Notes in Computer Science*, pages 110–121. Springer Berlin Heidelberg, 2006. doi:10.1007/11821069_10.

2. P.-Y. Angrand, S. Lombardy, and J. Sakarovitch. On the number of broken derived terms of a rational expression. *Journal of Automata, Languages and Combinatorics*, 15(1/2):27–51, 2010.

3. V. Antimirov. Partial derivatives of regular expressions and finite automaton constructions. *Theoretical Computer Science*, 155(2):291–319, 1996. doi:10.1016/0304-3975(95)00182-4.

4. J.A. Brzozowski. Derivatives of regular expressions. *Journal of the ACM*, 11(4):481–494, 1964. doi:10.1145/321239.321249.

5. P. Caron, J.-M. Champarnaud, and Ludovic Mignot. Partial derivatives of an extended regular expression. In A.-H. Dediu, S. Inenaga, and C. Martín-Vide, editors, *Language and Automata Theory and Applications*, volume 6638 of *Lecture Notes in Computer Science*, pages 179–191. Springer Berlin Heidelberg, 2011. doi:10.1007/978-3-642-21254-3_13.

6. P. Caron and M. Flouret. Glushkov construction for multiplicities. In S. Yu and A. Păun, editors, *Implementation and Application of Automata: 5th International Conference, CIAA 2000 London, Ontario, Canada, July 24–25, 2000 Revised Papers*, pages 67–79, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg. doi:10.1007/3-540-44674-5_5.

7. J.-M. Champarnaud, F. Ouardi, and D. Ziadi. An efficient computation of the equation $\mathbb{K}$-automaton of a regular $\mathbb{K}$-expression. In T. Harju, J. Karhumäki, and A. Lepistö, editors, *Developments in Language Theory*, volume 4588 of *Lecture Notes in Computer Science*, pages 145–156. Springer Berlin Heidelberg, 2007. doi:10.1007/978-3-540-73208-2_16.

8. A. Demaille. Derived-term automata for extended weighted rational expressions. In A. Sampaio and F. Wang, editors, *Theoretical Aspects of Computing - ICTAC 2016 - 13th International Colloquium, Taipei, Taiwan, ROC, October 24-31, 2016, Proceedings*, volume 9965 of *Lecture Notes in Computer Science*, pages 351–369, 2016. doi:10.1007/978-3-319-46750-4_20.

9. A. Demaille. Derived-term automata of multitape rational expressions. In Y.-S. Han and K. Salomaa, editors, *Proceedings of Implementation and Application of Automata, 21st International Conference (CIAA'16)*, volume 9705 of *Lecture Notes in Computer Science*, pages 51–63, Seoul, South Korea, July 2016. Springer. doi:10.1007/978-3-319-40946-7_5.

10. A. Demaille, A. Duret-Lutz, S. Lombardy, and J. Sakarovitch. Implementation concepts in Vaucanson 2. In S. Konstantinidis, editor, *Proceedings of Implementation and Application of Automata, 18th International Conference (CIAA'13)*, volume 7982 of *Lecture Notes in Computer Science*, pages 122–133, Halifax, NS, Canada, July 2013. Springer. doi:10.1007/978-3-642-39274-0_12.

11. Z. Ésik and W. Kuich. Equational axioms for a theory of automata. In C. Martín-Vide, V. Mitrana, and G. Păun, editors, *Formal Languages and Applications*, pages 183–196. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004. doi:10.1007/978-3-540-39886-8_10.

12. V. M. Glushkov. The abstract theory of automata. *Russian Math. Surveys*, 16:1–53, 1961. doi:10.1070/RM1961v016n05ABEH004112.

13. Ronald M. Kaplan and Martin Kay. Regular models of phonological rule systems. *Computational Linguistics - Special issue on computational phonology*, 20(3):331–378, September 1994.

14. S. C. Kleene. Representation of events in nerve nets and finite automata. In Claude Shannon and John McCarthy, editors, *Automata Studies*, pages 3–41. Princeton University Press, Princeton, NJ, 1956.

15. Dexter C. Kozen. *Automata and Computability*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1st edition, 1997. doi:10.1007/978-1-4612-1844-9.

16. S. Lombardy and J. Sakarovitch. Derivatives of rational expressions with multiplicity. *Theoretical Computer Science*, 332(1-3):141–177, 2005. doi:10.1016/j.tcs.2004.10.016.

17. S. Lombardy and J. Sakarovitch. The validity of weighted automata. *Int. J. of Algebra and Computation*, 23(4):863–914, 2013. doi:10.1142/S0218196713400146.

18. A. Ya. Makarevskii and E. D. Stotskaya. Representability in deterministic multi-tape automata. *Cybernetics and System Analysis*, 5(4):390–399, 1969. doi:10.1007/BF01073050.

19. R. McNaughton and H. Yamada. Regular expressions and state graphs for automata. *IEEE Transactions on Electronic Computers*, 9:39–47, 1960. doi:10.1109/TEC.1960.5221603.

20. M. Mohri. Edit-distance of weighted automata: General definitions and algorithms. *International Journal of Foundations of Computer Science*, 14(6):957–982, 2003. doi:10.1142/S0129054103002114.

21. T. Ng. Prefix distance between regular languages. In Y.S. Han and K. Salomaa, editors, *Implementation and Application of Automata - 21st International Conference, CIAA 2016, Seoul, South Korea, July 19-22, 2016, Proceedings*, pages 224–235, 2016. doi:10.1007/978-3-319-40946-7_19.

22. S. Owens, J. Reppy, and A. Turon. Regular-expression derivatives re-examined. *Journal of Functional Programming*, 19(2):173–190, March 2009. doi:10.1017/S0956796808007090.

23. J.J.M.M. Rutten. Behavioural differential equations: a coinductive calculus of streams, automata, and power series. *Theoretical Computer Science*, 308(1-3):1–53, 2003. doi:10.1016/S0304-3975(02)00895-2.

24. J. Sakarovitch. *Elements of Automata Theory*. Cambridge University Press, 2009. Corrected English translation of *Éléments de théorie des automates*, Vuibert, 2003. doi:10.1017/CBO9781139195218.

25. K. Thompson. Programming techniques: Regular expression search algorithm. *Communications of the ACM*, 11(6):419–422, 1968. doi:10.1145/363347.363387.

© Scientific Annals of Computer Science 2017