Derivatives of Quantitative Regular Expressions

Rajeev Alur[1] and Konstantinos Mamouras[1] and Dogan Ulus[2]

[1] University of Pennsylvania, Philadelphia, PA, USA
[2] Verimag, Université Grenoble-Alpes, Grenoble, France

Abstract. Quantitative regular expressions (QREs) have been recently proposed as a high-level declarative language for specifying complex numerical queries over data streams in a modular way. QREs have appealing theoretical properties, and each QRE can be compiled into an efficient streaming algorithm for its evaluation. In this paper, we generalize the notion of Brzozowski derivatives for classical regular expressions to QREs. Such derivatives immediately lead to an algorithm for incremental evaluation of QREs. While this algorithm does not have better time or space complexity than the previously known evaluation technique, it has the benefit of being simpler to explain and easier to prove correct.

1 Introduction

There are numerous applications that require the real-time processing of data generated at high rates such as: analyzing stock market data, monitoring production and manufacturing using sensors, network traffic monitoring, and clickstream analysis on the web. A core computational problem that is relevant to all such applications is the incremental aggregation of a stream of data items into numerical values that are useful for real-time decision making. Due to the enormous volume of data, these applications have hard constraints regarding space usage and the time required to process each newly arriving element.

There is a large body of prior research on stream processing which focuses on algorithmic techniques, often involving approximation and randomization, for computing efficiently specific numerical quantities such as the median [18], the number of distinct elements [17], the frequency moments [2], and aggregates over sliding windows [15]. There have also been several proposals for languages and systems that integrate stream processing with the data processing capabilities of traditional relational database systems [1,8,9].

The formalism of Quantitative Regular Expressions (QREs) was recently introduced in [5] with the orthogonal goal of providing convenient high-level programming abstractions for specifying complex queries over data streams in a modular way. QREs extend regular expressions, a well-established formalism for imparting hierarchical structure to sequences of symbols, with numerical operations such as sum, difference, min, max, average, and median. A QRE thus describes both the regular parsing of the stream prefix seen so far and the hierarchical calculation of a quantitative aggregate that reflects the structure of the parse tree. This combination gives rise to a powerful declarative language, which

can express conveniently many useful queries and is amenable to space- and time-efficient evaluation in the streaming model of computation. An implementation of QREs extended with extra features for processing realistic workloads is reported in [21]. The expressiveness of QREs coincides with the class of regular functions, which can be characterized with the model of Cost Register Automata (CRAs) [4] or, equivalently, with the MSO-definable graph transformations [14].

The main computational problem for QREs is their evaluation in the streaming model of computation. An efficient algorithm has already been described in [5] and implemented in [21], but it is not based on automata-theoretic techniques, and the question remains of whether there exists a simple model of automata for the streaming evaluation of QREs. In the simpler setting of classical regular expressions, the translation into a Nondeterministic Finite Automaton (NFA) gives rise to a very efficient streaming evaluation algorithm: the state of the algorithm consists of the active states of the NFA, and upon the arrival of a new symbol the state is updated by performing all possible transitions. Another approach for the evaluation problem is based on a technique proposed by Brzozowski in 1964 [11], where he introduced the notion of *derivation* for regular expressions extended with arbitrary Boolean operations. The *derivative* of an expression e with respect to a symbol a, typically denoted as $D_a(e)$, is an expression given by a simple recursive definition on the structure of e. The crucial property of these derivatives is that a string of the form aw (starting with the symbol a) matches an expression e iff the suffix w matches the derivative $D_a(e)$. This suggests a streaming evaluation algorithm for regular expressions: the state is an expression, and upon arrival of a new symbol a the state is replaced by its derivative with respect to a. A refinement of Brzozowski's ideas was proposed by Antimirov [7] under the name of *partial derivatives*. He described a representation of derivatives as sets of partial derivatives, which correponds closely to the construction of a NFA from an expression.

Given the success of automata-based techniques for the evaluation of plain regular expressions, it is worthwhile investigating whether similar ideas can be used for QRE evaluation. The well-studied model of weighted automata over semirings (see the monograph [16] for a broad survey) seems relevant, but unfortunately it is not expressive enough to handle the complex nesting of several different quantitative operations found in QREs. In particular, by the Kleene-Schützenberger theorem [22], weighted regular expressions can be easily translated into equivalent weighted automata and evaluated efficiently. On the other hand, QREs can be translated into the model of deterministic CRAs [4], but this translation incurs a doubly exponential blowup and is therefore not conducive to efficient evaluation. A hierarchical automaton model for the streaming computation of an interesting class of quantitative queries is introduced in [6], but its precise relationship to QREs and other automata formalisms remains to be clarified. A conclusively appropriate notion of automaton for the efficient evaluation of general quantitative queries has not been proposed yet, therefore a meaningful investigation is the development of a notion of derivative. Indeed, in the present paper, we extend the notion of Brzozowski derivatives to QREs, and we show

Table 1. Complexity results under syntactic and semantic restrictions

Query language	Time-per-element & space complexity
Unrestricted, multiset semantics	Exponential in stream/query
Unrestricted, unambiguous semantics	Constant in stream, Exponential in query
Strongly typed	Constant in stream, Polynomial in query

that there is a representation of QRE derivatives that gives rise to an efficient evaluation algorithm. This result offers a simple and clean alternative proof of why QREs can be efficiently evaluated, and it strongly suggests the possibility of an automata-based formulation of the evaluation algorithm. We should note here that derivatives have already been studied in the weighted setting [20], but the case of QREs is substantially different. See also [10] for an investigation of how Brzozowski derivatives can be extended to various algebraic structures.

Outline of paper. In Sect. 2 we present the syntax and meaning of QREs. We consider two different natural semantics: (1) The *multiset semantics* allows for several output values for a given stream prefix, each of which correponds to a different parse tree. (2) The *unambiguous semantics*, on the other hand, specifies the output to be undefined when the input stream can be parsed in more than one way. Thus, the unambiguous semantics ensures a single output value for each input sequence. In Sect. 3 we define derivatives of QREs by generalizing the classical notion of Brzozowski derivatives of regular expressions, and we propose an incremental evaluation algorithm based on derivatives. We also consider in Sect. 4 a representation of QRE derivatives that is analogous to Antimirov's partial derivatives [7]. In the presence of intersection the number of distinct Antimirov derivatives (for plain regular expressions) is exponentially bounded by the size of the expression. We show how to obtain a similar bound for QRE evaluation using the unambiguous semantics. Finally, we consider in Sect. 5 a syntatic restriction for QREs [5] that guarantees unambiguous parsing. Our complexity results for the streaming evaluation problem are summarized in Table 1. Although the proposed derivative-based algorithm has the same time and space complexity as the previously known method [5], our approach here is cleaner and the analysis of the algorithm much simpler. We conclude in Sect. 6 with a summary of our results and directions for future work.

2 Quantitative Regular Expressions

The formalism of Quantitative Regular Expressions offers a declarative language for describing complex hierarchical computations on streams of data values. As an illustrative example, consider the application of patient monitoring, where the data stream is a time series of timestamped measurements produced by a sensor attached to a patient. We want to analyze the data stream by first identifying regions of interest which we call "episodes". These are maximal intervals where the

measurements are above a fixed threshold value. Every episode is summarized by recording the average measurement, the maximum measurement, and the duration of the episode. The top-level query is an aggregation over the last 30 days (e.g., by calculating the average) of the episode statistics. This query imparts a hierarchical structure on the data stream by splitting it into episodes, bucketing episodes into days, and considering the last 30 days every time the aggregate statistics are computed. To describe this computation we need a language that supports regular constructs such as iteration (to express, e.g., that an episode is a sequence of measurements exceeding the threshold) and concatenation (to express, e.g., that an episode is followed by a sequence of measurements below the threshold), extended with quantitative operations for computing numerical aggregates (e.g., the maximum measurement of the sequence of measurements that constitute an episode).

In this section we present the syntax and semantics of Quantitative Regular Expressions. A QRE is interpreted over a stream of data values, and specifies for each finite prefix of the stream an output value. We consider two different semantics: the multiset semantics which records several different output possibilities, and the unambiguous semantics which only allows the output to be defined when it is uniquely determined. Since we are interested in computing well-defined *functions* on data streams, the multiset semantics is not satisfactory. We consider it here, however, because it is a natural semantics from a mathematical perspective, and it is useful for formulating and proving our results regarding efficient evaluation. The unambiguous semantics is simply a projection of the multiset semantics, and therefore several results w.r.t. to the multiset semantics transfer essentially unchanged to the unambiguous semantics.

To define QREs, we first choose a typed signature which describes the basic data types and operations for manipulating them. We fix a collection of *basic types*, and we write $A, B, \ldots$ to range over them, as well as a collection of *basic operations* on them, e.g. $op : A_1 \times \cdots \times A_k \to B$. The identity function on D is written $id_D : D \to D$. For every basic type D, assume that we have fixed a collection of *atomic predicates*, so that the satisfiability of their Boolean combinations is decidable. We write $\phi : D \to \mathbb{B}$ to indicate that ϕ is a predicate on D, and $true_D : D \to \mathbb{B}$ for the predicate that is always true. The *unit type*, with unique inhabitant `def`, is $\mathbb{U}$ and $!_D : D \to \mathbb{U}$ is the unique function from D to $\mathbb{U}$. We also write $\pi_1 : A \times B \to A$ and $\pi_2 : A \times B \to B$ for the left and right projection respectively. We assume that the collection of basic operations contains all identities and projections, and is closed under pairing and function composition. For example, if $op : A \times B \to C$ and $a \in A$ are basic operations, then so is $(op\, a) = \lambda b.op(a, b) : B \to C$.

Every QRE is defined on a regular subset of stream prefixes, so we first introduce a variant of regular expressions with unary predicates to describe the domains of definition of QREs. For a basic type D, we define *(Symbolic) Regular Expressions (REs)* over D with the grammar $r ::= \bot \mid \varepsilon \mid \phi \mid r \sqcup r \mid r \cdot r \mid r^* \mid r \sqcap r$, where $\phi : D \to \mathbb{B}$. The expression r^n is abbreviation for $r \cdot r \cdots r$ with r repeated n times. We write $r : \mathsf{RE}\langle D \rangle$ to indicate that r is a regular expression over D.

$$\frac{}{\bot : \mathsf{QRE}\langle D, C\rangle}\ (\text{bottom}) \qquad \frac{c \in C}{\mathtt{eps}(c) : \mathsf{QRE}\langle D, C\rangle}\ (\text{empty})$$

$$\frac{\text{satisfiable } \phi : D \to \mathbb{B} \qquad op : D \to C}{\mathtt{atom}(\phi, op) : \mathsf{QRE}\langle D, C\rangle}\ (\text{single item}) \qquad \frac{\mathtt{f}, \mathtt{g} : \mathsf{QRE}\langle D, C\rangle}{\mathtt{f} \sqcup \mathtt{g} : \mathsf{QRE}\langle D, C\rangle}\ (\text{choice})$$

$$\frac{\mathtt{f} : \mathsf{QRE}\langle D, A\rangle \qquad \mathtt{g} : \mathsf{QRE}\langle D, B\rangle \qquad op : A \times B \to C}{\mathtt{split}(\mathtt{f}, \mathtt{g}, op) : \mathsf{QRE}\langle D, C\rangle}\ (\text{split})$$

$$\frac{\mathtt{init} : \mathsf{QRE}\langle D, B\rangle \quad \mathtt{body} : \mathsf{QRE}\langle D, A\rangle \quad \mathcal{E}(\mathtt{body}) = \emptyset \quad op : B \times A \to B}{\mathtt{iter}(\mathtt{init}, \mathtt{body}, op) : \mathsf{QRE}\langle D, B\rangle}\ (\text{iteration})$$

$$\frac{\mathtt{f} : \mathsf{QRE}\langle D, A\rangle \qquad op : A \to B}{op(\mathtt{f}) : \mathsf{QRE}\langle D, B\rangle}\ (\text{application})$$

$$\frac{\mathtt{f} : \mathsf{QRE}\langle D, A\rangle \qquad \mathtt{g} : \mathsf{QRE}\langle D, B\rangle \qquad op : A \times B \to C}{op(\mathtt{f}, \mathtt{g}) : \mathsf{QRE}\langle D, C\rangle}\ (\text{combination})$$

$$\begin{aligned}
\mathcal{E}(\bot) &= \emptyset & \mathcal{E}(\mathtt{f} \sqcup \mathtt{g}) &= \mathcal{E}(\mathtt{f}) \uplus \mathcal{E}(\mathtt{g}) \\
\mathcal{E}(\mathtt{eps}(c)) &= \{c\} & \mathcal{E}(\mathtt{split}(\mathtt{f}, \mathtt{g}, op)) &= \mathsf{M}(op)(\mathcal{E}(\mathtt{f}), \mathcal{E}(\mathtt{g})) \\
\mathcal{E}(\mathtt{atom}(\phi, op)) &= \emptyset & \mathcal{E}(\mathtt{iter}(\mathtt{f}, \mathtt{g}, op)) &= \mathcal{E}(\mathtt{f}) \\
\mathcal{E}(op(\mathtt{f})) &= \mathsf{M}(op)(\mathcal{E}(\mathtt{f})) & \mathcal{E}(op(\mathtt{f}, \mathtt{g})) &= \mathsf{M}(op)(\mathcal{E}(\mathtt{f}), \mathcal{E}(\mathtt{g}))
\end{aligned}$$

Fig. 1. Syntax of Quantitative Regular Expressions (QREs) without ε-cycles.

We define $[\![r]\!] : D^* \to \mathbb{N}$ to be the weighted semantics of regular expressions without ε-cycles (i.e., no ε-cycles in the corresponding ε-NFA [19]) that counts the number of different parse trees.

$$\begin{aligned}
[\![\bot]\!]\, w &= 0 & [\![\varepsilon]\!]\, w &= 0 \ (w \neq \varepsilon) & [\![r_1 \sqcup r_2]\!]\, w &= ([\![r_1]\!]\, w) + ([\![r_2]\!]\, w) \\
[\![\varepsilon]\!]\, \varepsilon &= 1 & [\![\phi]\!]\, d &= 1 \ (d \models \phi) & [\![r_1 \cdot r_2]\!]\, w &= \textstyle\sum_{w=uv} ([\![r_1]\!]\, u) \cdot ([\![r_2]\!]\, v) \\
& & [\![\phi]\!]\, d &= 0 \ (d \not\models \phi) & [\![r^*]\!]\, w &= \textstyle\sum_{w=u_1 \cdots u_n} ([\![r]\!]\, u_1) \cdots ([\![r]\!]\, u_n) \\
& & [\![\phi]\!]\, w &= 0 \ (w \notin D) & [\![r_1 \sqcap r_2]\!]\, w &= ([\![r_1]\!]\, w) \cdot ([\![r_2]\!]\, w)
\end{aligned}$$

This semantics corresponds to standard operations for formal power series [16].

In Fig. 1 we define *Quantitative Regular Expressions* (*QREs*), which we also call *queries*. The queries are typed, and we write $\mathtt{f} : \mathsf{QRE}\langle D, C\rangle$ to indicate that the query $\mathtt{f}$ has *input type* D and *output type* C. The original definition of QREs in [5] was more general in that it involved an extra sort of typed parameters, and the outputs were essentially algebraic terms built from the parameters and the operations of the signature. The definition of [5] was motivated by expressiveness considerations, i.e. so that QREs capture exactly the class of regular functions over the same signature [14,3,4]. We consider here a simpler language, where the outputs are just values. This simplification makes QREs significantly more usable for practical queries and obviates the need for a *term simplification* procedure (see section 3.2 of [5]) that depends on the nature of the data types and operations of the signature.

The language of Fig. 1 has eight core constructs: (1) $\bot$ is undefined for every input sequence; (2) $\texttt{eps}(c)$ maps the empty stream to the output value c; (3) $\texttt{atom}(\phi, op)$ maps an input stream consisting of a single item satisfying the predicate ϕ to an output computed by applying the operation op; (4) $\texttt{f} \sqcup \texttt{g}$ nondeterministically chooses either $\texttt{f}$ or $\texttt{g}$ to apply; (5) $\texttt{split}(\texttt{f}, \texttt{g}, op)$ splits the input stream into two parts, applies the queries $\texttt{f}$ and $\texttt{g}$ to the left and right parts respectively, and combines their results using the operation op; (6) $\texttt{iter}(\texttt{init}, \texttt{body}, op)$ splits the input into multiple parts $uv_1v_2 \dots v_n$, applies $\texttt{init}$ to u (which gives b) and $\texttt{body}$ to each v_i (which gives a_i), and combines the sequence of values $a_1a_2 \dots a_n$ using the initial value b and the binary aggregation operation op in the style of the *fold combinator* used in functional programming; (7) $op(\texttt{f})$ applies the query $\texttt{f}$ and transforms its output using the operation op; (8) $op(\texttt{f}, \texttt{g})$ applies both $\texttt{f}$, $\texttt{g}$ and combines their results using the operation op.

The definition of queries is by mutual induction with the function $\mathcal{E}$, which sends a query of type $\mathsf{QRE}\langle D, C\rangle$ to a finite multiset over its output type C. The $\mathcal{E}$ function is meant to give the output of a query on the empty sequence, and it is used for the assumption $\mathcal{E}(\texttt{body}) = \emptyset$ for the iteration query $\texttt{iter}(\texttt{init}, \texttt{body}, op)$. It is necessary to define the syntax of queries simultaneously with $\mathcal{E}$ in order to eliminate queries with ε-cycles, that is, queries where the body $\texttt{g}$ of the iteration query $\texttt{iter}(\texttt{f}, \texttt{g}, op)$ matches the empty sequence ε. When the semantics is defined later, we will see that ε-cycles result in having an infinity of output values. This complication of ε-cycles appears also in the context of weighted automata and expressions [16]. The operation $\uplus$ in Fig. 1 is multiset union. For a set X, we write $\mathsf{M}(X)$ to denote the set of all finite multisets over X. For a unary operation $op : A \to B$, we define the lifting $\mathsf{M}(op) : \mathsf{M}(A) \to \mathsf{M}(B)$ by $\mathsf{M}(op)(X) = \{op(a) \mid a \in X\}$. Similarly, for an operation $op : A \times B \to C$, we define the lifting $\mathsf{M}(op) : \mathsf{M}(A) \times \mathsf{M}(B) \to \mathsf{M}(C)$ by $\mathsf{M}(op)(X, Y) = \{op(a, b) \mid a \in X \text{ and } b \in Y\}$.

Multiset semantics. We give a denotational semantics of queries in terms of functions of type $D^* \to \mathsf{M}(C)$. We call this the *multiset semantics* of queries. The *domain* of f is the set of sequences for which the value of f is nonempty, i.e. $\mathrm{dom}(f) = \{w \in D^* \mid f(w) \neq \emptyset\}$. The denotation of a query $\texttt{f} : \mathsf{QRE}\langle D, C\rangle$ is the function $[\![\texttt{f}]\!] : D^* \to \mathsf{M}(C)$, where $[\![\cdot]\!]$ is called the interpretation function and is defined by induction on the structure of queries as shown in Fig. 2. To reduce the notational clutter we sometimes write $[\![\texttt{f}]\!]\, w$ instead of $[\![\texttt{f}]\!](w)$. The semantics of iteration involves the *multiset fold* combinator mfold, which is a generalization of the familiar fold combinator to multisets of values. The definitions of mfold and fold are by recursion on the length of the sequence. For example, $\mathsf{fold}(s, op, a_1a_2) = op(op(s, a_1), a_2))$ and

$$\mathsf{mfold}(\{b_1, b_2\}, op, \{a_1\}\,\{a_2, a_3\}) = \\ \{op(op(b_1, a_1), a_2), op(op(b_1, a_1), a_3), op(op(b_2, a_1), a_2), op(op(b_2, a_1), a_3)\}.$$

For an iteration query $\texttt{h} = \texttt{iter}(\texttt{f}, \texttt{g}, op)$, the typing restriction $\mathcal{E}(\texttt{g}) = \emptyset$ implies that $[\![g]\!]\, \varepsilon = \emptyset$ (formally proved later in Theorem 5). So, to calculate the value

$$\begin{array}{rl}
[\![\bot]\!]\, w = \emptyset & [\![\mathtt{f} \sqcup \mathtt{g}]\!]\, w = [\![\mathtt{f}]\!]\, w \uplus [\![\mathtt{g}]\!]\, w \\
[\![\mathtt{eps}(c)]\!]\, \varepsilon = \{c\} & [\![\mathtt{split}(\mathtt{f},\mathtt{g},op)]\!]\, w = \biguplus_{w=uv} \mathsf{M}(op)([\![\mathtt{f}]\!]\, u, [\![\mathtt{g}]\!]\, v) \\
[\![\mathtt{atom}(\phi,op)]\!]\, d = \{op(d)\},\ \text{if } d \models \phi & [\![op(\mathtt{f})]\!]\, w = \mathsf{M}(op)([\![\mathtt{f}]\!]\, w) \\
[\![\mathtt{atom}(\phi,op)]\!]\, d = \emptyset,\ \text{if } d \not\models \phi & [\![op(\mathtt{f},\mathtt{g})]\!]\, w = \mathsf{M}(op)([\![\mathtt{f}]\!]\, w, [\![\mathtt{g}]\!]\, w)
\end{array}$$

$$[\![\mathtt{iter}(\mathtt{f},\mathtt{g},op)]\!]\, w = \biguplus_{w=uv_1\cdots v_n} \mathsf{mfold}([\![\mathtt{f}]\!]\, u, op, [\![\mathtt{g}]\!]\, v_1 \cdots [\![\mathtt{g}]\!]\, v_n)$$

$$\begin{array}{ll}
\mathsf{fold} : B \times (B{\times}A \to B) \times A^* \to B & \mathsf{mfold} : \mathsf{M}(B) \times (B{\times}A \to B) \times \mathsf{M}(A)^* \to \mathsf{M}(B) \\
\mathsf{fold}(s, op, \varepsilon) = s & \mathsf{mfold}(S, op, \varepsilon) = S \\
\mathsf{fold}(s, op, wa) = op(\mathsf{fold}(s, op, w), a) & \mathsf{mfold}(S, op, WX) = \mathsf{M}(op)(\mathsf{mfold}(S, op, W), X)
\end{array}$$

Fig. 2. Finite multiset semantics of Quantitative Regular Expressions without ε-cycles.

$[\![\mathtt{h}]\!]\, w$ (see Fig. 2) we only need to consider the splittings $w = uv_1 \dots v_n$ of w where $n \geq 0$ and $v_i \neq \varepsilon$ for every $i = 1, \dots, n$.

For every input sequence w, the value $[\![\mathtt{f}]\!]\, w$ is a finite multiset whose size is at most exponential in the size of w. More precisely, (size of $[\![\mathtt{f}]\!]\, w$) $\leq 2^{|\mathtt{f}|\cdot|w|}$ for every query $\mathtt{f} : \mathsf{QRE}\langle D, C\rangle$ and every sequence $w \in D^*$.

The multiset semantics of queries induces an equivalence relation on them, written as $\equiv$. Two queries are equivalent if their denotations are equal. We can then write equations such as $\mathtt{split}(\mathtt{f}, \mathtt{eps}(b), op) \equiv op_b(\mathtt{f})$, where op_b is the unary operation given by $op_b(x) = op(x, b)$ for all x.

Example 1. The query $\mathtt{f} = \mathtt{atom}(true_{\mathbb{N}}, id_{\mathbb{N}}) : \mathsf{QRE}\langle \mathbb{N}, \mathbb{N}\rangle$ matches a single number and returns it. Using it as the body of an iteration, we write the query $\mathtt{g} = \mathtt{iter}(\mathtt{f}, \mathtt{f}, +)$ which processes a nonempty sequence of numbers and returns their sum. Now, the query $\mathtt{g}' = \mathtt{iter}(\mathtt{f}', \mathtt{f}', +)$, where $\mathtt{f}' = \mathtt{atom}(true_{\mathbb{N}}, \lambda x.1)$, processes a nonempty sequence and returns its length. If $\mathsf{div} : \mathbb{N} \times \mathbb{N} \to \mathbb{Q}$ and $\mathsf{div}(x, y)$ is the result of dividing x by y, the query $\mathtt{h} = \mathsf{div}(\mathtt{g}, \mathtt{g}') : \mathsf{QRE}\langle \mathbb{N}, \mathbb{Q}\rangle$ calculates the average of a nonempty sequence of natural numbers.

Rates. The *rate* of a query is a symbolic regular expression that denotes its domain. It is defined by induction:

$$\begin{array}{rl}
\mathsf{rate}(\bot) = \bot & \mathsf{rate}(\mathtt{f} \sqcup \mathtt{g}) = \mathsf{rate}(\mathtt{f}) \sqcup \mathsf{rate}(\mathtt{g}) \\
\mathsf{rate}(\mathtt{eps}(c)) = \varepsilon & \mathsf{rate}(\mathtt{split}(\mathtt{f},\mathtt{g},op)) = \mathsf{rate}(\mathtt{f}) \cdot \mathsf{rate}(\mathtt{g}) \\
\mathsf{rate}(\mathtt{atom}(\phi,op)) = \phi & \mathsf{rate}(\mathtt{iter}(\mathtt{f},\mathtt{g},op)) = \mathsf{rate}(\mathtt{f}) \cdot \mathsf{rate}(\mathtt{g})^* \\
\mathsf{rate}(op(\mathtt{f})) = \mathsf{rate}(\mathtt{f}) & \mathsf{rate}(op(\mathtt{f},\mathtt{g})) = \mathsf{rate}(\mathtt{f}) \sqcap \mathsf{rate}(\mathtt{g})
\end{array}$$

Notice that the value of rate is always an expression without ε-cycles.

Unambiguous semantics. We defined previously the multiset semantics of queries, which allows a query to have several (finitely many) outputs for a given input sequence. Now, we take the viewpoint that a query should specify a unique output value (or be undefined) for a given input sequence. This means that we

want to ignore output multisets of cardinality greater than one, which we do by setting the output to be undefined. This is the *unambiguous semantics* for queries. For a query $\mathtt{f} : \mathsf{QRE}\langle D, C\rangle$, this is given formally as follows:

$$\langle\!\langle \mathtt{f} \rangle\!\rangle\, w = \theta([\![\mathtt{f}]\!]\, w), \text{ where } \theta(X) = X \text{ if } |X| = 1 \text{ and } \theta(X) = \emptyset \text{ otherwise}$$

So, $\langle\!\langle \mathtt{f} \rangle\!\rangle$ is a function $D^* \to \mathsf{M}(C)$ so that each output multiset is of cardinality at most one, which we call an *unambiguous function*. This means that $\langle\!\langle \mathtt{f} \rangle\!\rangle$ can be also represented as a partial function $D^* \rightharpoonup C$. We say that a query $\mathtt{f}$ is *unambiguous* when the multiset meaning $[\![\mathtt{f}]\!]$ is an unambiguous function. This is equivalent to $[\![\mathtt{f}]\!] = \langle\!\langle \mathtt{f} \rangle\!\rangle$, which says that the multiset and unambiguous semantics coincide. See the papers [12,13] for recent surveys of unambiguity in traditional automata theory.

It is necessary that the $[\![-]\!]$ semantics records the multiplicity of each output value, otherwise $\langle\!\langle - \rangle\!\rangle$ cannot be defined as a projection or $[\![-]\!]$. Indeed, we see in the example below that we can write queries that return exactly one output value of multiplicity greater than one.

Example 2. The query $\mathtt{f} = \mathtt{iter}(\mathtt{eps}(0), \mathtt{atom}(\mathit{true}_{\mathbb{N}}, \mathit{id}_{\mathbb{N}}), +) : \mathsf{QRE}\langle \mathbb{N}, \mathbb{N}\rangle$, which processes a sequence of natural numbers and returns their sum, is unambiguous. The query $\mathtt{g} = \mathtt{iter}(\mathtt{eps}(\mathtt{def}), \mathtt{atom}(\mathit{true}_{\mathbb{N}}, !_{\mathbb{N}}), !_{\mathbb{N}\times\mathbb{N}}) : \mathsf{QRE}\langle \mathbb{N}, \mathbb{U}\rangle$, which matches a sequence of natural numbers and returns nothing, is also unambiguous. The query $\mathtt{split}(\mathtt{g}, \mathtt{f}, \pi_2) : \mathsf{QRE}\langle \mathbb{N}, \mathbb{N}\rangle$, which matches a sequence of natural numbers and returns the sum of every suffix of the sequence, is ambiguous because every sequence of length ℓ can be parsed in $\ell+1$ different ways. The query that matches sequences of length at least two and returns the sum of the last two elements is $\mathtt{split}(\mathtt{g}, \mathtt{split}(\mathtt{atom}(\mathit{true}_{\mathbb{N}}, \mathit{id}_{\mathbb{N}}), \mathtt{atom}(\mathit{true}_{\mathbb{N}}, \mathit{id}_{\mathbb{N}}), +), \pi_2) : \mathsf{QRE}\langle \mathbb{N}, \mathbb{N}\rangle$ and is unambiguous. The query $\mathtt{split}(\mathtt{f}, \mathtt{f}, +) : \mathsf{QRE}\langle \mathbb{N}, \mathbb{N}\rangle$ is ambiguous but single-valued: for a sequence of length ℓ, it returns the sum of its elements with multiplicity equal to $\ell + 1$.

The unambiguous semantics of queries induces an equivalence relation on them, written as $\approxeq$. Two queries $\mathtt{f}$ and $\mathtt{g}$ are $\approxeq$-equivalent if their denotations $\langle\!\langle \mathtt{f} \rangle\!\rangle$ and $\langle\!\langle \mathtt{g} \rangle\!\rangle$ are equal. We observe that the equivalence relation $\approxeq$ is strictly coarser than $\equiv$, that is, $\mathtt{f} \equiv \mathtt{g}$ implies $\mathtt{f} \approxeq \mathtt{g}$ and there exist queries that are $\approxeq$-equivalent but $\equiv$-inequivalent.

Observation 3. A finite multiset Q of queries of the same type can also be thought of as a query, namely the finite choice over the queries of Q. We now want to find some sufficient conditions for reducing the cardinality of Q, while preserving its meaning under the unambiguous semantics. This will turn out to be useful later in Sect. 4, where an evaluation algorithm for QREs using Antimirov derivatives is presented (Fig. 3 and Theorem 9).

Suppose Q contains the queries $\mathtt{f}_1$, $\mathtt{f}_2$, ..., $\mathtt{f}_k$ with $k \geq 2$ that have the same rate, that is $r = \mathsf{rate}(\mathtt{f}_i)$ for every i, and there is no other query in Q that has this rate. The condition on the rates implies that all functions $[\![\mathtt{f}_i]\!]$ have the same domain $D_1 = \mathrm{dom}([\![\mathtt{f}_i]\!])$, and moreover all functions $\langle\!\langle \mathtt{f}_i \rangle\!\rangle$ have the same domain $D_2 = \mathrm{dom}(\langle\!\langle \mathtt{f}_i \rangle\!\rangle) \subseteq D_1$.

(1) Remove all of $\mathtt{f}_1, \ldots, \mathtt{f}_k$ (*wrong*): We claim that $Q' = Q \setminus \{\mathtt{f}_1, \ldots, \mathtt{f}_k\}$ is not necessarily equivalent to Q. Suppose that $Q = \{\mathtt{f}_1, \ldots, \mathtt{f}_k, \mathtt{g}\}$, the queries of Q are all unambiguous, and the domains $\mathrm{dom}(\llbracket \mathtt{g} \rrbracket) = \mathrm{dom}(\langle\!\langle \mathtt{g} \rangle\!\rangle)$ and $D_1 = D_2$ intersect. So, all queries of Q are defined on some sequence w. Then, $\langle\!\langle Q \rangle\!\rangle$ is not defined on w because the cardinality of $\llbracket Q \rrbracket\, w$ is greater than one, but $\langle\!\langle Q' \rangle\!\rangle$ is defined on w and equal to $\langle\!\langle \mathtt{g} \rangle\!\rangle\, w$. So, $Q \not\approx Q'$.

(2) If $k \geq 3$ then remove $\mathtt{f}_3, \ldots, \mathtt{f}_k$ and keep $\mathtt{f}_1, \mathtt{f}_2$ (*correct*): Define the multiset $Q' = Q \setminus \{\mathtt{f}_3, \ldots, \mathtt{f}_k\}$. We claim that $Q \approx Q'$. Let w be an arbitrary sequence. If w belongs to the domain of the functions $\llbracket \mathtt{f}_i \rrbracket$, then both $\langle\!\langle Q \rangle\!\rangle$ and $\langle\!\langle Q' \rangle\!\rangle$ are undefined on w, because the cardinalities of $\llbracket Q \rrbracket\, w$ and $\llbracket Q' \rrbracket\, w$ are greater than one. Suppose now that w does not belong to the domain of the functions $\llbracket \mathtt{f}_i \rrbracket$. Then, $\llbracket Q \rrbracket\, w = \llbracket Q \setminus \{\mathtt{f}_i \mid i\} \rrbracket\, w = \llbracket Q' \rrbracket\, w$ and hence $\langle\!\langle Q \rangle\!\rangle\, w = \langle\!\langle Q' \rangle\!\rangle\, w$.

This means that we can always reduce Q so that it has at most two queries with the same rate, while preserving its unambiguous semantics.

3 Brzozowski Derivative

We introduce in this section the *Brzozowski derivative* [11] of Quantitative Regular Expressions, which is a straightforward adaption of derivatives for classical regular expressions. The main property of these derivatives is that their semantic counterpart agrees with the syntactic counterpart. This agreement property gives as a corollary the existence of an incremental algorithm for evaluating QREs on streams of data items.

Example 4. The Brzozowski derivative $\mathcal{D}^{\mathsf{B}}_a(r)$ of a regular expression r w.r.t. the letter a denotes the language that results from the language of r by removing the starting a letter from those words that start with a. For example, $r = (a \sqcup b)^* bb$ denotes the language of all strings over $\Sigma = \{a, b\}$ that end in bb, $\mathcal{D}^{\mathsf{B}}_a(r) = (a \sqcup b)^* bb = r$, $\mathcal{D}^{\mathsf{B}}_{ab}(r) = \mathcal{D}^{\mathsf{B}}_b(\mathcal{D}^{\mathsf{B}}_a(r)) = \mathcal{D}^{\mathsf{B}}_b(r) = r \sqcup b$ and $\mathcal{D}^{\mathsf{B}}_{abb}(r) = \mathcal{D}^{\mathsf{B}}_b(\mathcal{D}^{\mathsf{B}}_{ab}(r)) = \mathcal{D}^{\mathsf{B}}_b(r \sqcup b) = r \sqcup b \sqcup \varepsilon$. The string abb matches r because the empty string ε matches the derivative $\mathcal{D}^{\mathsf{B}}_{abb}(r)$.

A query of type $\mathsf{QRE}\langle \mathbb{N}, \mathbb{N} \rangle$ that is similar in form to the regex r is $\mathtt{k} = \mathtt{split}(\mathtt{g}, \mathtt{h}, \max)$, where $\mathtt{g} = \mathtt{iter}(\mathtt{eps}(0), \mathtt{f}_e \sqcup \mathtt{f}_o, +)$, $\mathtt{f}_e = \mathtt{atom}(\mathit{even}_{\mathbb{N}}, \mathit{id}_{\mathbb{N}})$, $\mathtt{f}_o = \mathtt{atom}(\mathit{odd}_{\mathbb{N}}, \mathit{id}_{\mathbb{N}})$, and $\mathtt{h} = \mathtt{split}(\mathtt{f}_o, \mathtt{f}_o, +)$. The unambiguous query $\mathtt{k}$ matches sequences that end with two odd numbers and returns the maximum of x, y where y is the sum of the last two numbers and x is the sum of the rest of the numbers. The extension of derivatives to QREs should give rise to the following calculations:

$$
\begin{aligned}
\mathcal{D}^{\mathsf{B}}_4(\mathtt{k}) &= \mathtt{split}(\mathtt{iter}((\lambda x.0 + x)(\mathtt{eps}(4)), \mathtt{f}_e \sqcup \mathtt{f}_o, +), \mathtt{h}, \max) \\
\mathcal{D}^{\mathsf{B}}_{43}(\mathtt{k}) &= \mathtt{split}(\mathtt{iter}((\lambda x.4 + x)(\mathtt{eps}(3)), \mathtt{f}_e \sqcup \mathtt{f}_o, +), \mathtt{h}, \max) \sqcup \\
&\quad (\lambda x.\max(4, x))((\lambda x.3 + x)(\mathtt{f}_o)) \\
\mathcal{D}^{\mathsf{B}}_{435}(\mathtt{k}) &= \mathtt{split}(\mathtt{iter}((\lambda x.7 + x)(\mathtt{eps}(5)), \mathtt{f}_e \sqcup \mathtt{f}_o, +), \mathtt{h}, \max) \sqcup \\
&\quad (\lambda x.\max(7, x))((\lambda x.5 + x)(\mathtt{f}_o)) \sqcup \\
&\quad (\lambda x.\max(4, x))((\lambda x.3 + x)(\mathtt{eps}(5)))
\end{aligned}
$$

$$\begin{aligned}\mathcal{D}^{\mathsf{B}}_{4351}(\mathtt{k}) = \mathtt{split}(\mathtt{iter}((\lambda x.12+x)(\mathtt{eps}(1)), \mathtt{f}_e \sqcup \mathtt{f}_o, +), \mathtt{h}, \max) \sqcup \\ (\lambda x.\max(12,x))((\lambda x.1+x)(\mathtt{f}_o)) \sqcup \\ (\lambda x.\max(7,x))((\lambda x.5+x)(\mathtt{eps}(1)))\end{aligned}$$

From the above we notice that ε matches $\mathcal{D}^{\mathsf{B}}_{435}(\mathtt{k})$ with value $\max(4, 3+5) = 8$, and it also matches $\mathcal{D}^{\mathsf{B}}_{4351}(\mathtt{k})$ with value $\max(7, 5+1) = 7$.

A simple streaming algorithm that implements the computation described by $\mathtt{h}$ can be given by maintaining the following state: the sum of all elements so far except for the last two, and the two most recent elements. The reader can observe that the derivatives calculated earlier record this information, and it can be found inside the queries that are constructed. For example, in the derivative $\mathcal{D}^{\mathsf{B}}_{4351}(\mathtt{k})$ we see (last line) the sum 7 and the elements 5, 1. The structure of the derivative also encodes how these three numbers should be combined to produce the output $max(7, 5+1) = 7$. □

For a function $f : D^* \to \mathsf{M}(C)$, we define the *semantic derivative* $\mathcal{D}_u(f) : D^* \to \mathsf{M}(C)$ with respect to the sequence $u \in D^*$ of data items as

$$\mathcal{D}_u(f)\, w = f(uw) \text{ for all } w \in D^*.$$

An immediate consequence is that $\mathcal{D}_v(\mathcal{D}_u(f)) = \mathcal{D}_{uv}(f)$ for all sequences u and v in D^*. Moreover, $f(u) = \mathcal{D}_u(f)(\varepsilon)$ for every sequence $u \in D^*$. For a query $\mathtt{f}$ of type $\mathsf{QRE}\langle D, C\rangle$ and a data item $d \in D$, the *(syntactic) Brzozowski derivative* $\mathcal{D}^{\mathsf{B}}_d(\mathtt{f})$ of $\mathtt{f}$ w.r.t. d is also a query of type $\mathsf{QRE}\langle D, C\rangle$.

$$\begin{aligned}
\mathcal{D}^{\mathsf{B}}_d(\mathtt{eps}(c)) &= \mathcal{D}^{\mathsf{B}}_d(\bot) = \bot & \mathcal{D}^{\mathsf{B}}_d(\mathtt{f} \sqcup \mathtt{g}) &= \mathcal{D}^{\mathsf{B}}_d(\mathtt{f}) \sqcup \mathcal{D}^{\mathsf{B}}_d(\mathtt{g}) \\
\mathcal{D}^{\mathsf{B}}_d(\mathtt{atom}(\phi, op)) &= \bot, \text{ if } d \not\models \phi & \mathcal{D}^{\mathsf{B}}_d(op(\mathtt{f})) &= op(\mathcal{D}^{\mathsf{B}}_d(\mathtt{f})) \\
\mathcal{D}^{\mathsf{B}}_d(\mathtt{atom}(\phi, op)) &= \mathtt{eps}(op(d)), \text{ if } d \models \phi & \mathcal{D}^{\mathsf{B}}_d(op(\mathtt{f}, \mathtt{g})) &= op(\mathcal{D}^{\mathsf{B}}_d(\mathtt{f}), \mathcal{D}^{\mathsf{B}}_d(\mathtt{g})) \\
\mathcal{D}^{\mathsf{B}}_d(\mathtt{split}(\mathtt{f}, \mathtt{g}, op)) &= \mathtt{split}(\mathcal{D}^{\mathsf{B}}_d(\mathtt{f}), \mathtt{g}, op) \sqcup \bigsqcup_{a \in \mathcal{E}(\mathtt{f})} (op\, a)(\mathcal{D}^{\mathsf{B}}_d(\mathtt{g})) \\
\mathcal{D}^{\mathsf{B}}_d(\mathtt{iter}(\mathtt{f}, \mathtt{g}, op)) &= \mathtt{iter}(\mathcal{D}^{\mathsf{B}}_d(\mathtt{f}), \mathtt{g}, op) \sqcup \bigsqcup_{b \in \mathcal{E}(\mathtt{f})} \mathtt{iter}((op\, b)(\mathcal{D}^{\mathsf{B}}_d(\mathtt{g})), \mathtt{g}, op)
\end{aligned}$$

The derivative $\mathcal{D}^{\mathsf{B}}_w(\mathtt{f})$ w.r.t. a sequence $w \in D^*$ is defined by induction on w: $\mathcal{D}^{\mathsf{B}}_\varepsilon(\mathtt{f}) = \mathtt{f}$ and $\mathcal{D}^{\mathsf{B}}_{dw}(\mathtt{f}) = \mathcal{D}^{\mathsf{B}}_w(\mathcal{D}^{\mathsf{B}}_d(\mathtt{f}))$. A crucial result is the correpondence between semantic and syntactic derivatives:

Theorem 5 (Derivative Agreement). For every query $\mathtt{f}$ of type $\mathsf{QRE}\langle D, C\rangle$ and every data item $d \in D$, we have that $\mathcal{E}(\mathtt{f}) = [\![\mathtt{f}]\!]\, \varepsilon$ and $\mathcal{D}_d([\![\mathtt{f}]\!]) = [\![\mathcal{D}^{\mathsf{B}}_d(\mathtt{f})]\!]$.

Theorem 5 suggests immediately an algorithm for evaluating queries. Given a sequence $w = d_1 d_2 \ldots d_n$ and a query $\mathtt{f}$, we notice that $[\![\mathtt{f}]\!]\, w$ is equal to

$$\mathcal{D}_w([\![\mathtt{f}]\!])\, \varepsilon = \mathcal{D}_{d_n} \cdots \mathcal{D}_{d_1}([\![\mathtt{f}]\!])\, \varepsilon = [\![\mathcal{D}^{\mathsf{B}}_{d_n} \cdots \mathcal{D}^{\mathsf{B}}_{d_1}(\mathtt{f})]\!]\, \varepsilon = \mathcal{E}(\mathcal{D}^{\mathsf{B}}_{d_n} \cdots \mathcal{D}^{\mathsf{B}}_{d_1}(\mathtt{f})).$$

So, to compute the value of a query on a given input, we iteratively calculate the derivative of the query w.r.t. each input item, and finally apply the $\mathcal{E}$ function. We suggest an optimization of this evaluation procedure by incorporating

query rewriting to eliminate subqueries that cannot contribute to the result. The equations $op(\bot) \equiv \bot$ and

$$\begin{array}{llll} \bot \sqcup \mathtt{f} \equiv \mathtt{f} & \mathtt{split}(\bot, \mathtt{f}, op) \equiv \bot & op(\bot, \mathtt{f}) \equiv \bot & \mathtt{iter}(\bot, \mathtt{f}, op) \equiv \bot \\ \mathtt{f} \sqcup \bot \equiv \mathtt{f} & \mathtt{split}(\mathtt{f}, \bot, op) \equiv \bot & op(\mathtt{f}, \bot) \equiv \bot & \mathtt{iter}(\mathtt{f}, \bot, op) \equiv \mathtt{f} \end{array}$$

are all valid, that is, they are true for every query $\mathtt{f}$. If we orient these equations from left to right, then we get a rewrite system for simplifying queries. We will assume that our Brzozowski derivative-based evaluation procedure simplifies all intermediate queries as much as possible (according to this rewrite system).

Example 6. Consider the always-true predicate $true_{\mathbb{N}} : \mathbb{N} \to \mathbb{B}$, the identity function $id_{\mathbb{N}} : \mathbb{N} \to \mathbb{N}$, the constant zero function $\lambda x.0 : \mathbb{N} \to \mathbb{N}$, and the binary sum $+ : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$. Using these operations we write the queries

$$\mathtt{f} = \mathtt{atom}(true_{\mathbb{N}}, id_{\mathbb{N}}) \quad \mathtt{g} = \mathtt{atom}(true_{\mathbb{N}}, \lambda x.0) \quad \mathtt{h} = \mathtt{iter}(\mathtt{eps}(0), \mathtt{f} \sqcup \mathtt{g}, +)$$

of type $\mathsf{QRE}\langle \mathbb{N}, \mathbb{N} \rangle$. The top-level query is $\mathtt{h}$, which maps an input sequence of natural numbers into the multiset of all possible partial sums over those numbers. First, notice that the derivatives of $\mathtt{f}$ and $\mathtt{g}$ w.r.t. the number $d \in \mathbb{N}$ are $\mathcal{D}^{\mathsf{B}}_d(\mathtt{f}) = \mathtt{eps}(d)$ and $\mathcal{D}^{\mathsf{B}}_d(\mathtt{g}) = \mathtt{eps}(0)$ respectively. Moreover, $\mathcal{E}(\mathtt{eps}(0)) = \{0\}$. We will use the derivative-based evaluation algorithm to find the value of the query $\mathtt{h}$ on the input sequence $abc \in \mathbb{N}^*$. This amounts to calculating $\mathcal{E}(\mathcal{D}^{\mathsf{B}}_c(\mathcal{D}^{\mathsf{B}}_b(\mathcal{D}^{\mathsf{B}}_a(\mathtt{h}))))$, since Theorem 5 implies that it equals $[\![\mathtt{h}]\!]\, abc$. The steps of this calculation are:

$$\begin{aligned} \mathcal{D}^{\mathsf{B}}_a(\mathtt{h}) &= \mathtt{iter}(\mathtt{eps}(a) \sqcup \mathtt{eps}(0), \mathtt{f} \sqcup \mathtt{g}, +) \\ \mathcal{E}(\mathcal{D}^{\mathsf{B}}_a(\mathtt{h})) &= \{a, 0\} \\ \mathcal{D}^{\mathsf{B}}_b(\mathcal{D}^{\mathsf{B}}_a(\mathtt{h})) &= \mathtt{iter}((\lambda x.a + x)(\mathtt{eps}(b) \sqcup \mathtt{eps}(0)), \mathtt{f} \sqcup \mathtt{g}, +) \sqcup \\ &\quad\ \mathtt{iter}((\lambda x.0 + x)(\mathtt{eps}(b) \sqcup \mathtt{eps}(0)), \mathtt{f} \sqcup \mathtt{g}, +) \\ \mathcal{E}(\mathcal{D}^{\mathsf{B}}_b(\mathcal{D}^{\mathsf{B}}_a(\mathtt{h}))) &= \{a + b, a, b, 0\} \\ \mathcal{D}^{\mathsf{B}}_c(\mathcal{D}^{\mathsf{B}}_b(\mathcal{D}^{\mathsf{B}}_a(\mathtt{h}))) &= \mathtt{iter}((\lambda x.(a + b) + x)(\mathtt{eps}(c) \sqcup \mathtt{eps}(0)), \mathtt{f} \sqcup \mathtt{g}, +) \sqcup \\ &\quad\ \mathtt{iter}((\lambda x.a + x)(\mathtt{eps}(c) \sqcup \mathtt{eps}(0)), \mathtt{f} \sqcup \mathtt{g}, +) \sqcup \\ &\quad\ \mathtt{iter}((\lambda x.b + x)(\mathtt{eps}(c) \sqcup \mathtt{eps}(0)), \mathtt{f} \sqcup \mathtt{g}, +) \sqcup \\ &\quad\ \mathtt{iter}((\lambda x.0 + x)(\mathtt{eps}(c) \sqcup \mathtt{eps}(0)), \mathtt{f} \sqcup \mathtt{g}, +) \\ \mathcal{E}(\mathcal{D}^{\mathsf{B}}_c(\mathcal{D}^{\mathsf{B}}_b(\mathcal{D}^{\mathsf{B}}_a(\mathtt{h})))) &= \{a + b + c, a + b, a + c, a, b + c, b, c, 0\} \end{aligned}$$

We have used implicitly the rewrite rules for eliminating $\bot$ as much as possible.

The previous example shows that there are queries whose evaluation requires an enormous amount of computational resources. Given that the size of the output can be exponential in the size of the input sequence and the size of the query, we have exponential time and space requirements for every evaluation algorithm.

4 Antimirov Derivative

The evaluation of the general QREs of Fig. 1 w.r.t. the multiset semantics is inherently expensive, since the output itself can be of size exponential in the size of the query and the input sequence. So, we focus here on the evaluation of QREs w.r.t. the unambiguous semantics, and our goal is to describe a streaming algorithm that uses resources that are independent of the size of the input stream. Using a variant of the partial derivatives of Antimirov [7], we show that this is indeed possible. We obtain a streaming algorithm for evaluation that uses space and time-per-element exponential in the size of the query and independent of the stream length. The crucial idea for this algorithm is that we can prune the Antimirov derivative to contain only a couple of QREs with the same rate without changing its unambiguous meaning. Antimirov-style derivatives are preferable for the results of this section, because the representation itself encodes many valid equations on queries that are useful for proving the result. This is similar to classical regular expressions where the Antimirov derivatives encode the ACI rules (associativity, commutativity and idempotence) by virtue of the set-based representation.

Example 7. We continue with Example 4 to consider Antimirov derivatives. Recall that $r = (a \sqcup b)^* bb$, and that Antimirov derivatives are a set-based representation of Brzozowski derivatives. That is, $\mathcal{D}^{\mathsf{A}}_a(r) = \{r\}$, $\mathcal{D}^{\mathsf{A}}_{ab}(r) = \mathcal{D}^{\mathsf{A}}_b(\{r\}) = \{r, b\}$ and $\mathcal{D}^{\mathsf{A}}_{abb}(r) = \mathcal{D}^{\mathsf{A}}_b(\{r, b\}) = \{r, b, \varepsilon\}$. For the query $\mathtt{k}$ of Example 4 we calculate using Antimirov-style derivatives:

$$\begin{aligned}
\mathcal{D}^{\mathsf{A}}_4(\mathtt{k}) &= \{\mathtt{split}(\mathtt{iter}((\lambda x.0 + x)(\mathtt{eps}(4)), \mathtt{f}_e \sqcup \mathtt{f}_o, +), \mathtt{h}, \max)\} \\
\mathcal{D}^{\mathsf{A}}_{43}(\mathtt{k}) &= \{\mathtt{split}(\mathtt{iter}((\lambda x.4 + x)(\mathtt{eps}(3)), \mathtt{f}_e \sqcup \mathtt{f}_o, +), \mathtt{h}, \max), \\
&\qquad (\lambda x.\max(4, x))((\lambda x.3 + x)(\mathtt{f}_o))\} \\
\mathcal{D}^{\mathsf{A}}_{435}(\mathtt{k}) &= \{\mathtt{split}(\mathtt{iter}((\lambda x.7 + x)(\mathtt{eps}(5)), \mathtt{f}_e \sqcup \mathtt{f}_o, +), \mathtt{h}, \max), \\
&\qquad (\lambda x.\max(7, x))((\lambda x.5 + x)(\mathtt{f}_o)), \\
&\qquad (\lambda x.\max(4, x))((\lambda x.3 + x)(\mathtt{eps}(5)))\}
\end{aligned}$$

Since the choice operation $\sqcup$ for queries is not idempotent, we need multisets for the representation of QRE Antimirov derivatives. □

For a regular expression $r : \mathsf{RE}\langle D\rangle$ we define $\mathcal{E}(r) \in \{0, 1\}$ (the two-element Boolean algebra with join operation $+$ and meet operation $\cdot$) and the *Antimirov derivative* $\mathcal{D}^{\mathsf{A}}_d(r)$ w.r.t. $d \in D$, which is a set of regular expressions of type $\mathsf{RE}\langle D\rangle$:

$$\begin{aligned}
\mathcal{E}(\bot) &= 0 & \mathcal{D}^{\mathsf{A}}_d(\varepsilon) &= \mathcal{D}^{\mathsf{A}}_d(\bot) = \emptyset \\
\mathcal{E}(\varepsilon) &= 1 & \mathcal{D}^{\mathsf{A}}_d(\phi) &= \emptyset, \text{ if } d \not\models \phi \\
\mathcal{E}(\phi) &= 0 & \mathcal{D}^{\mathsf{A}}_d(\phi) &= \{\varepsilon\}, \text{ if } d \models \phi \\
\mathcal{E}(r_1 \sqcup r_2) &= \mathcal{E}(r_1) + \mathcal{E}(r_2) & \mathcal{D}^{\mathsf{A}}_d(r_1 \sqcup r_2) &= \mathcal{D}^{\mathsf{A}}_d(r_1) \cup \mathcal{D}^{\mathsf{A}}_d(r_2) \\
\mathcal{E}(r_1 \sqcap r_2) &= \mathcal{E}(r_1) \cdot \mathcal{E}(r_2) & \mathcal{D}^{\mathsf{A}}_d(r_1 \sqcap r_2) &= \mathcal{D}^{\mathsf{A}}_d(r_1) \sqcap \mathcal{D}^{\mathsf{A}}_d(r_2) \\
\mathcal{E}(r_1 \cdot r_2) &= \mathcal{E}(r_1) \cdot \mathcal{E}(r_2) & \mathcal{D}^{\mathsf{A}}_d(r_1 \cdot r_2) &= \mathcal{D}^{\mathsf{A}}_d(r_1) \cdot r_2 \cup \mathcal{E}(r_1) \cdot \mathcal{D}^{\mathsf{A}}_d(r_2) \\
\mathcal{E}(r^*) &= 1 & \mathcal{D}^{\mathsf{A}}_d(r^*) &= \mathcal{D}^{\mathsf{A}}_d(r) \cdot r^*
\end{aligned}$$

State : Finite multiset S of queries of type $\mathsf{QRE}\langle D, C\rangle$.

Initialization() : Set the state S to the singleton multiset $\{\mathtt{f}\}$.

Update(d) : Compute the Antimirov derivative $S' = \mathcal{D}^{\mathsf{A}}_d(S) = \biguplus\{\mathcal{D}^{\mathsf{A}}_d(\mathtt{g}) \mid \mathtt{g} \in S\}$. Now, iterate the following conditional modification until no further changes can be made: if S' contains at least three queries that have the same rate, keep only two of them and remove the rest. Finally, set S to be equal to S'.

Output() : If $\mathcal{E}(S) = \biguplus\{\mathcal{E}(\mathtt{g}) \mid \mathtt{g} \in S\}$ is the singleton multiset $\{c\}$, then return the value c. Otherwise, the output is undefined.

Fig. 3. Streaming evaluation algorithm for an arbitrary query $\mathtt{f} : \mathsf{QRE}\langle D, C\rangle$, with respect to the unambiguous semantics of QREs.

For subsets X, Y of expressions we have used in the definition the abbreviations: $X \sqcap Y = \{r \sqcap s \mid r \in X, s \in Y\}$, $X \cdot s = \{r \cdot s \mid r \in X\}$, $0 \cdot X = \emptyset$ and $1 \cdot X = X$.

Lemma 8. For every regular expression r, the set $\bigcup_{w \in D^*} \mathcal{D}^{\mathsf{A}}_w(r)$ of all derivatives of r is of size exponential in r.

For a query $\mathtt{f}$ of type $\mathsf{QRE}\langle D, C\rangle$ and a data item $d \in D$, the *Antimirov derivative* $\mathcal{D}^{\mathsf{A}}_d(\mathtt{f})$ of $\mathtt{f}$ w.r.t. d is a finite ***multiset*** of queries of type $\mathsf{QRE}\langle D, C\rangle$.

$$\mathcal{D}^{\mathsf{A}}_d(\mathtt{eps}(c)) = \mathcal{D}^{\mathsf{A}}_d(\bot) = \emptyset \qquad \mathcal{D}^{\mathsf{A}}_d(\mathtt{f} \sqcup \mathtt{g}) = \mathcal{D}^{\mathsf{A}}_d(\mathtt{f}) \uplus \mathcal{D}^{\mathsf{A}}_d(\mathtt{g})$$

$$\mathcal{D}^{\mathsf{A}}_d(\mathtt{atom}(\phi, op)) = \emptyset, \text{ if } d \not\models \phi \qquad \mathcal{D}^{\mathsf{A}}_d(op(\mathtt{f})) = op(\mathcal{D}^{\mathsf{A}}_d(\mathtt{f}))$$

$$\mathcal{D}^{\mathsf{A}}_d(\mathtt{atom}(\phi, op)) = \{\mathtt{eps}(op(d))\}, \text{ if } d \models \phi \qquad \mathcal{D}^{\mathsf{A}}_d(op(\mathtt{f}, \mathtt{g})) = op(\mathcal{D}^{\mathsf{A}}_d(\mathtt{f}), \mathcal{D}^{\mathsf{A}}_d(\mathtt{g}))$$

$$\mathcal{D}^{\mathsf{A}}_d(\mathtt{split}(\mathtt{f}, \mathtt{g}, op)) = \mathtt{split}(\mathcal{D}^{\mathsf{A}}_d(\mathtt{f}), \mathtt{g}, op) \uplus \biguplus_{a \in \mathcal{E}(\mathtt{f})} (op\ a)(\mathcal{D}^{\mathsf{A}}_d(\mathtt{g}))$$

$$\mathcal{D}^{\mathsf{A}}_d(\mathtt{iter}(\mathtt{f}, \mathtt{g}, op)) = \mathtt{iter}(\mathcal{D}^{\mathsf{A}}_d(\mathtt{f}), \mathtt{g}, op) \uplus \biguplus_{b \in \mathcal{E}(\mathtt{f})} \mathtt{iter}((op\ b)(\mathcal{D}^{\mathsf{A}}_d(\mathtt{g})), \mathtt{g}, op)$$

We have used above convenient abbreviations like $op(X) = \{op(\mathtt{f}) \mid \mathtt{f} \in X\}$. The derivative $\mathcal{D}^{\mathsf{A}}_w(\mathtt{f})$ w.r.t. a sequence $w \in D^*$ is defined by induction on w: $\mathcal{D}^{\mathsf{A}}_\varepsilon(\mathtt{f}) = \{\mathtt{f}\}$ and $\mathcal{D}^{\mathsf{A}}_{dw}(\mathtt{f}) = \bigcup\{\mathcal{D}^{\mathsf{A}}_w(\mathtt{g}) \mid \mathtt{g} \in \mathcal{D}^{\mathsf{A}}_d(\mathtt{f})\}$.

Theorem 9. The algorithm of Fig. 3 solves the evaluation problem for QREs w.r.t. the unambiguous semantics. It requires space and time-per-element that are constant in the length of the stream, and exponential in the size of the query.

Proof. To prove that the algorithm of Fig. 3 is correct, we observe that it satisfies the following crucial invariant: after consuming the input $w \in D^*$, the multiset S is $\approx$-equivalent to the derivative $\mathcal{D}^{\mathsf{A}}_w(\mathtt{f})$. This is because the removal step of the **Update** procedure preserves the unambiguous meaning of S (recall Observation 3). It remains to see that the Antimirov derivative is a streamlined representation of the Brzozowski derivative. That is, for every query $\mathtt{f} : \mathsf{QRE}\langle D, C\rangle$

and every data item $d \in D$, it holds that $\mathcal{D}^{\mathsf{B}}_d(\mathtt{f}) \equiv \bigsqcup \mathcal{D}^{\mathsf{A}}_d(\mathtt{f})$. The equivalences

$$\begin{aligned} \mathtt{f} \sqcup \mathtt{g} &\equiv \mathtt{g} \sqcup \mathtt{f} & op(\mathtt{f} \sqcup \mathtt{g}, \mathtt{h}) &\equiv op(\mathtt{f}, \mathtt{h}) \sqcup op(\mathtt{g}, \mathtt{h}) \\ op(\mathtt{f} \sqcup \mathtt{g}) &\equiv op(\mathtt{f}) \sqcup op(\mathtt{g}) & op(\mathtt{f}, \mathtt{g} \sqcup \mathtt{h}) &\equiv op(\mathtt{f}, \mathtt{g}) \sqcup op(\mathtt{f}, \mathtt{h}) \end{aligned}$$

$$\begin{aligned} \mathtt{split}(\mathtt{f} \sqcup \mathtt{g}, \mathtt{h}) &\equiv \mathtt{split}(\mathtt{f}, \mathtt{h}) \sqcup \mathtt{split}(\mathtt{g}, \mathtt{h}) \\ \mathtt{iter}(\mathtt{f} \sqcup \mathtt{g}, \mathtt{h}, op) &\equiv \mathtt{iter}(\mathtt{f}, \mathtt{h}, op) \sqcup \mathtt{iter}(\mathtt{g}, \mathtt{h}, op) \end{aligned}$$

are used in the proof of this claim. Essentially, the above equations are encoded in the derivatives by way of their representation as multisets of queries. The output procedure returns $\langle\!\langle S \rangle\!\rangle\, \varepsilon = \langle\!\langle \mathcal{D}^{\mathsf{A}}_w(\mathtt{f}) \rangle\!\rangle\, \varepsilon = \theta([\![\mathcal{D}^{\mathsf{B}}_w(\mathtt{f})]\!]\, \varepsilon) = \theta([\![\mathtt{f}]\!]\, w) = \langle\!\langle \mathtt{f} \rangle\!\rangle\, w$.

The Antimirov derivatives on QREs correspond closely to the Antimirov derivatives on regular expressions. For every query $\mathtt{g} \in \mathcal{D}^{\mathsf{A}}_d(\mathtt{f})$, it holds that $\mathsf{rate}(\mathtt{g}) \in \mathcal{D}^{\mathsf{A}}_d(\mathsf{rate}(\mathtt{f}))$. Because of the pruning step, the state S contains at most two queries for every possible rate of the derivatives. Since there are at most exponentially many derivatives of $\mathsf{rate}(\mathtt{f})$ by Lemma 8, the cardinality of S is also at most exponential in $\mathtt{f}$ and independent of the size of the input sequence. The time to process each element is also exponential in $|\mathtt{f}|$. □

Example 10. The query $\mathtt{f} = \mathtt{atom}(\mathit{true}_{\mathbb{N}}, \mathit{id}_{\mathbb{N}})$ is of type $\mathsf{QRE}\langle \mathbb{N}, \mathbb{N} \rangle$ and it maps a single natural number to itself. The query $\mathtt{g} = \mathtt{iter}(\mathtt{eps}(0), \mathtt{f}, +)$ maps any sequence of natural numbers to their sum. We calculate the derivative of $\mathtt{g}$:

$$\begin{aligned} \mathcal{E}(\mathtt{eps}(0)) = \{0\} \quad \mathcal{D}^{\mathsf{A}}_d(\mathtt{f}) &= \{\mathtt{eps}(d)\} \quad \mathcal{D}^{\mathsf{A}}_d(\mathtt{eps}(0)) = \emptyset \\ \mathcal{D}^{\mathsf{A}}_d(\mathtt{g}) &= \mathtt{iter}(\mathcal{D}^{\mathsf{A}}_d(\mathtt{eps}(0)), \mathtt{f}, +) \uplus \mathtt{iter}((\lambda x.0 + x)(\mathcal{D}^{\mathsf{A}}_d(\mathtt{f})), \mathtt{f}, +) \\ &= \{\mathtt{iter}((\lambda x.0 + x)(\mathtt{eps}(d)), \mathtt{f}, +)\} \end{aligned}$$

For $x, y \in \mathbb{N}$ define $\mathtt{g}_{xy} = \mathtt{iter}((\lambda z.x + z)(\mathtt{eps}(y)), \mathtt{f}, +)$. The derivative of $\mathtt{g}_{xy}$ is:

$$\begin{aligned} \mathcal{D}^{\mathsf{A}}_d(\mathtt{g}_{xy}) &= \mathtt{iter}(\mathcal{D}^{\mathsf{A}}_d((\lambda z.x + z)(\mathtt{eps}(y))), \mathtt{f}, +) \uplus \\ &\qquad \mathtt{iter}((\lambda z.(x + y) + z)(\mathcal{D}^{\mathsf{A}}_d(\mathtt{f})), \mathtt{f}, +) \\ &= \{\mathtt{iter}((\lambda z.(x + y) + z)(\mathtt{eps}(d)), \mathtt{f}, +)\} = \mathtt{g}_{x+y,d} \end{aligned}$$

because $\mathcal{E}((\lambda z.x + z)(\mathtt{eps}(y))) = \{x + y\}$ and $\mathcal{D}^{\mathsf{A}}_d((\lambda z.x + z)(\mathtt{eps}(y))) = \emptyset$. □

5 Strongly typed queries and Hierarchical derivatives

Following [5], we consider a syntactic restriction of QREs that ensures unambiguity of parsing. This means that the multiset and unambiguous semantics coincide, thus this subclass of QREs inherently describes well-defined functions on data streams. Together with an additional restriction that demands in expressions of the form $op(\mathtt{f}, \mathtt{g})$ the subqueries $\mathtt{f}$ and $\mathtt{g}$ to have the same domain, it can be ensured that the evaluation of QREs can be performed efficiently, that is, using space and time-per-element that is polynomial in the size of the query

and independent of the stream. Such an algorithm is proposed in [5] and implemented in [21], but it is hard to describe and even harder to analyze. We consider here an alternative approach based on derivatives, which gives rise to an evaluation algorithm of the same complexity that is much easier to describe and prove correct. The main technical tool is a novel representation of derivatives, which we call *hierarchical derivatives*. This is a very space-efficient representation and is crucial for obtaining our complexity bounds.

Before defining the subclass of strongly typed queries, we introduce some definitions that will be used for formalizing the idea of uniqueness of parsing. The languages L_1, L_2 are said to be *unambiguously concatenable* if for every word $w \in L_1 \cdot L_2$ there are unique $w_1 \in L_1$, $w_2 \in L_2$ with $w = w_1 w_2$. The language L is said to be *unambiguously iterable* if for every word $w \in L^*$ there is a unique integer $n \geq 0$ and unique $w_i \in L$ with $w = w_1 \cdots w_n$. These definitions extend to regular expressions in the obvious way.

We say that a query is *strongly typed* if the following hold: (1) for every subquery $\mathtt{f} \sqcup \mathtt{g}$ the rates $\mathsf{rate}(\mathtt{f})$ and $\mathsf{rate}(\mathtt{g})$ are disjoint, (2) for every subquery $\mathtt{split}(\mathtt{f}, \mathtt{g}, op)$ the rates $\mathsf{rate}(\mathtt{f})$ and $\mathsf{rate}(\mathtt{g})$ are unambiguously concatenable. (3) for every subquery $\mathtt{iter}(\mathtt{f}, \mathtt{g}, op)$ the rate $\mathsf{rate}(\mathtt{g})$ is unambiguously iterable and $\mathsf{rate}(\mathtt{f})$, $\mathsf{rate}(\mathtt{g})^*$ are unambiguously concatenable, and (4) for every subquery $op(\mathtt{f}, \mathtt{g})$ the rates $\mathsf{rate}(\mathtt{f})$ and $\mathsf{rate}(\mathtt{g})$ are equivalent. It is shown in [5] that checking whether a query is strongly typed can be done in polynomial-time.

Lemma 11. If the query $\mathtt{f}$ is strongly typed, then $[\![\mathtt{f}]\!]$ is an unambiguous function. So, the multiset and unambiguous semantics coincide, i.e. $[\![\mathtt{f}]\!] = \langle\!\langle \mathtt{f} \rangle\!\rangle$.

The main problem with the Antimirov derivative of Sect. 4 is the treatment of $op(\mathtt{f}, \mathtt{g})$, where $\mathcal{D}^{\mathsf{A}}_d(op(\mathtt{f}, \mathtt{g})) = \{op(\mathtt{f}', \mathtt{g}') \mid \mathtt{f}' \in \mathcal{D}^{\mathsf{A}}_d(\mathtt{f}), \mathtt{g}' \in \mathcal{D}^{\mathsf{A}}_d(\mathtt{g})\}$. This definitions corresponds to a cartesian product of derivatives, thus causing a quadratic blowup in the number of possible derivatives. As we will see later in Example 14, this blowup can materialize even in the context of strongly typed queries. Since the output combination operation can nest with other regular constructs, the Antimirov representation can result in an exponenial blowup, which is avoidable in the strongly typed case. In order to prove this, we need a new representation that avoids this "cartesian product" problem of the Antimirov derivative by allowing sets of queries to be used as subexpressions.

We thus generalize the syntax of strongly typed queries to allow finite sets of queries as subexpressions. Intuitively, these finite sets of queries extend the choice operation $\sqcup$ to a finite number of arguments. Since the subqueries of $\mathtt{f} \sqcup \mathtt{g}$ must have disjoint domains, the $\sqcup$ constructor is associative and commutative, which means that we can represent $(\mathtt{f} \sqcup \mathtt{g}) \sqcup \mathtt{h}$ as $\{\mathtt{f}, \mathtt{g}, \mathtt{h}\}$. This finite choice constructor can be nested arbitrarily with the other query constructors. In Fig. 4 we see the formal definition, where queries and *q-sets* (finite sets of queries) are defined by mutual induction. To reduce the notational clutter, we sometimes write the query $\mathtt{f}$ instead of the q-set $\{\mathtt{f}\}$, for example $\mathtt{iter}\langle \mathtt{f}, \mathtt{g}, op \rangle$ instead of $\mathtt{iter}\langle \{\mathtt{f}\}, \{\mathtt{g}\}, op \rangle$. The rate of a generalized query is defined in the usual way, the only difference being $\mathsf{rate}(\{\mathtt{f}_1, \ldots, \mathtt{f}_k\}) = \mathsf{rate}(\mathtt{f}_1) \sqcup \cdots \sqcup \mathsf{rate}(\mathtt{f}_k)$.

$$\frac{c \in C}{\mathtt{eps}(c) : \mathsf{QRE}\langle D, C\rangle} \qquad \frac{\text{satisfiable } \phi : D \to \mathbb{B} \qquad op : D \to C}{\mathtt{atom}(\phi, op) : \mathsf{QRE}\langle D, C\rangle}$$

$$\frac{\mathtt{F} : \mathsf{QSET}\langle D, A\rangle \qquad \mathtt{G} : \mathsf{QSET}\langle D, B\rangle \qquad op : A \times B \to C \qquad \mathtt{F}, \mathtt{G} \neq \emptyset \qquad \mathsf{rate}(\mathtt{F}),\ \mathsf{rate}(\mathtt{G}) \text{ unambiguously concatenable}}{\mathtt{split}\langle \mathtt{F}, \mathtt{G}, op\rangle : \mathsf{QRE}\langle D, C\rangle}$$

$$\frac{op : B \times A \to B \qquad \mathtt{F} : \mathsf{QSET}\langle D, B\rangle \qquad \mathsf{rate}(\mathtt{F}),\ \mathsf{rate}(\mathtt{G})^* \text{ unambiguously concatenable} \qquad \mathtt{F}, \mathtt{G} \neq \emptyset \qquad \mathtt{G} : \mathsf{QSET}\langle D, A\rangle \qquad \mathsf{rate}(\mathtt{G}) \text{ unambiguously iterable}}{\mathtt{iter}\langle \mathtt{F}, \mathtt{G}, op\rangle : \mathsf{QRE}\langle D, B\rangle}$$

$$\frac{\mathtt{F} \neq \emptyset : \mathsf{QSET}\langle D, A\rangle \qquad op : A \to B}{op\langle \mathtt{F}\rangle : \mathsf{QRE}\langle D, C\rangle} \qquad \frac{\mathtt{F} : \mathsf{QSET}\langle D, A\rangle \qquad \mathtt{G} : \mathsf{QSET}\langle D, B\rangle \qquad op : A \times B \to C \qquad \mathtt{F}, \mathtt{G} \neq \emptyset \qquad \mathsf{rate}(\mathtt{F}) \text{ and } \mathsf{rate}(\mathtt{G}) \text{ are equivalent}}{op\langle \mathtt{F}, \mathtt{G}\rangle : \mathsf{QRE}\langle D, C\rangle}$$

$$\frac{\mathtt{f}_1, \ldots, \mathtt{f}_k : \mathsf{QRE}\langle D, C\rangle \qquad \mathsf{rate}(\mathtt{f}_1),\ \ldots,\ \mathsf{rate}(\mathtt{f}_k) \text{ pairwise disjoint}}{\{\mathtt{f}_1, \ldots, \mathtt{f}_k\} : \mathsf{QSET}\langle D, C\rangle}$$

Fig. 4. Generalized syntax for strongly typed QREs.

$$\mathcal{D}^{\mathsf{H}}_d(\{\mathtt{f}_i \mid i \in I\}) = \textstyle\bigcup_{i \in I} \mathcal{D}^{\mathsf{H}}_d(\mathtt{f}_i) \qquad \mathcal{D}^{\mathsf{H}}_d(\mathtt{atom}(\phi, op)) = \emptyset, \text{ if } d \not\models \phi$$
$$\mathcal{D}^{\mathsf{H}}_d(\mathtt{eps}(c)) = \emptyset \qquad \mathcal{D}^{\mathsf{H}}_d(\mathtt{atom}(\phi, op)) = \{\mathtt{eps}(op(d))\}, \text{ if } d \models \phi$$
$$\mathcal{D}^{\mathsf{H}}_d(op\langle \mathtt{F}\rangle) = \{op\langle \mathcal{D}^{\mathsf{H}}_d(\mathtt{F})\rangle\} \qquad \mathcal{D}^{\mathsf{H}}_d(op\langle \mathtt{F}, \mathtt{G}\rangle) = \{op\langle \mathcal{D}^{\mathsf{H}}_d(\mathtt{F}), \mathcal{D}^{\mathsf{H}}_d(\mathtt{G})\rangle\}$$
$$\mathcal{D}^{\mathsf{H}}_d(\mathtt{split}\langle \mathtt{F}, \mathtt{G}, op\rangle) = \{\mathtt{split}\langle \mathcal{D}^{\mathsf{H}}_d(\mathtt{F}), \mathtt{G}, op\rangle\} \cup \{(op\ a)\langle \mathcal{D}^{\mathsf{H}}_d(\mathtt{G})\rangle \mid a \in \mathcal{E}(\mathtt{f})\}$$
$$\mathcal{D}^{\mathsf{H}}_d(\mathtt{iter}\langle \mathtt{F}, \mathtt{G}, op\rangle) = \{\mathtt{iter}\langle \mathcal{D}^{\mathsf{H}}_d(\mathtt{F}), \mathtt{G}, op\rangle\} \cup \{\mathtt{iter}\langle (op\ b)\langle \mathcal{D}^{\mathsf{H}}_d(\mathtt{G})\rangle, \mathtt{G}, op\rangle \mid b \in \mathcal{E}(\mathtt{F})\}$$

$$\mathcal{E}(\mathtt{eps}(c)) = \{c\} \qquad \mathcal{E}(op\langle \mathtt{F}\rangle) = \mathsf{M}(op)(\mathcal{E}(\mathtt{F}))$$
$$\mathcal{E}(\mathtt{atom}(\phi, op)) = \emptyset \qquad \mathcal{E}(op\langle \mathtt{F}, \mathtt{G}\rangle) = \mathsf{M}(op)(\mathcal{E}(\mathtt{F}), \mathcal{E}(\mathtt{G}))$$
$$\mathcal{E}(\{\mathtt{f}_i\}_i) = \textstyle\bigcup_i \mathcal{E}(\mathtt{f}_i) \qquad \mathcal{E}(\mathtt{split}\langle \mathtt{F}, \mathtt{G}, op\rangle) = \mathsf{M}(op)(\mathcal{E}(\mathtt{F}), \mathcal{E}(\mathtt{G}))$$
$$\mathcal{E}(\mathtt{iter}\langle \mathtt{F}, \mathtt{G}, op\rangle) = \mathcal{E}(\mathtt{F})$$

Fig. 5. Hierarchical derivatives for generalized QREs.

For the expressions of Fig. 4 we define in Fig. 5 a new kind of derivative, called the *(syntactic) hierarchical derivative* $\mathcal{D}^{\mathsf{H}}_d(\cdot)$, which maps a query of type $\mathsf{QRE}\langle D, C\rangle$ or a q-set of type $\mathsf{QSET}\langle D, C\rangle$ to a q-set of type $\mathsf{QSET}\langle D, C\rangle$. The hierarchical derivative $\mathcal{D}^{\mathsf{H}}_w(\mathtt{F})$ w.r.t. a sequence $w \in D^*$ is defined by induction on w: $\mathcal{D}^{\mathsf{H}}_\varepsilon(\mathtt{F}) = \mathtt{F}$ and $\mathcal{D}^{\mathsf{H}}_{dw}(\mathtt{F}) = \mathcal{D}^{\mathsf{H}}_w(\mathcal{D}^{\mathsf{H}}_d(\mathtt{F}))$. For every sequence $u \in D^*$, the derivative $\mathcal{D}^{\mathsf{H}}_u(\mathtt{split}\langle \mathtt{F}, \mathtt{G}, op\rangle)$ is of the form:

$$\{\mathtt{split}\langle \mathcal{D}^{\mathsf{H}}_u(\mathtt{F}), \mathtt{G}, op\rangle, op\langle a_1, \mathcal{D}^{\mathsf{H}}_{v_1}(\mathtt{G})\rangle, \ldots, op\langle a_n, \mathcal{D}^{\mathsf{H}}_{v_n}(\mathtt{G})\rangle\}$$

for some $a_i \in A$ and sequences $v_i \in D^*$. Because of unambiguity, the rates of the derivatives $\mathcal{D}^{\mathsf{H}}_{v_1}(\mathtt{G}), \ldots, \mathcal{D}^{\mathsf{H}}_{v_n}(\mathtt{G})$ are pairwise disjoint. Similarly, for every sequence $u \in D^*$, the derivative $\mathcal{D}^{\mathsf{H}}_u(\mathtt{iter}\langle \mathtt{F}, \mathtt{G}, op\rangle)$ is of the form:

$$\{\mathtt{iter}\langle \mathcal{D}^{\mathsf{H}}_u(\mathtt{F}), \mathtt{G}, op\rangle, \mathtt{iter}\langle op\langle b_1, \mathcal{D}^{\mathsf{H}}_{v_1}(\mathtt{G})\rangle, \mathtt{G}, op\rangle, \ldots, \mathtt{iter}\langle op\langle b_n, \mathcal{D}^{\mathsf{H}}_{v_n}(\mathtt{G})\rangle, \mathtt{G}, op\rangle\}$$

for some $b_i \in B$ and sequences $v_i \in D^*$. Again, because of unambiguity, the rates of the derivatives $\mathcal{D}^{\mathsf{H}}_u(\mathtt{F})$, $\mathcal{D}^{\mathsf{H}}_{v_1}(\mathtt{G})$, ..., $\mathcal{D}^{\mathsf{H}}_{v_n}(\mathtt{G})$ are pairwise disjoint. A key technical lemma to obtain an efficient evaluation algorithm is that the derivatives of strongly typed queries are of polynomial size. To show this, first we define a reasonable notion of size of queries:

$$\begin{aligned}
\mathrm{size}(\{\mathtt{f}_i\}_i) &= \textstyle\sum_i \mathrm{size}(\mathtt{f}_i) & \mathrm{size}(op\langle \mathtt{F}\rangle) &= 1 + \mathrm{size}(\mathtt{F}) \\
\mathrm{size}(\mathtt{eps}(c)) &= 1 & \mathrm{size}(op\langle \mathtt{F}, \mathtt{G}\rangle) &= 1 + \mathrm{size}(\mathtt{F}) + \mathrm{size}(\mathtt{G}) \\
\mathrm{size}(\mathtt{atom}(\phi, op)) &= 2 & \mathrm{size}(\mathtt{split}\langle \mathtt{F}, \mathtt{G}, op\rangle) &= 1 + \mathrm{size}(\mathtt{F}) + \mathrm{size}(\mathtt{G}) \\
& & \mathrm{size}(\mathtt{iter}\langle \mathtt{F}, \mathtt{G}, op\rangle) &= 1 + \mathrm{size}(\mathtt{F}) + \mathrm{size}(\mathtt{G})
\end{aligned}$$

In order to obtain the desired lemma, the claim has to be strengthened:

Lemma 12. Let $\mathtt{F}$ be a q-set in $\mathsf{QSET}\langle D, C\rangle$ and $u_1, \ldots, u_n$ be sequences over D such that the rates of $\mathcal{D}^{\mathsf{H}}_{u_1}(\mathtt{F})$, ..., $\mathcal{D}^{\mathsf{H}}_{u_n}(\mathtt{F})$ are pairwise disjoint. Then:
(1) At most $\mathrm{size}(\mathtt{F})$ of the sets $\mathcal{D}^{\mathsf{H}}_{u_1}(\mathtt{F})$, ..., $\mathcal{D}^{\mathsf{H}}_{u_n}(\mathtt{F})$ are nonempty.
(2) The space needed to represent $\mathcal{D}^{\mathsf{H}}_{u_1}(\mathtt{F}) \cup \cdots \cup \mathcal{D}^{\mathsf{H}}_{u_n}(\mathtt{F})$ is bounded by $\mathrm{size}(\mathtt{F})^2$.

Proof. To prove the lemma, we must extend the same claim to queries as well. The proof then proceeds by induction on the structure of queries and q-sets. The base cases $\mathtt{eps}(c)$ and $\mathtt{atom}(\phi, op)$ and the step case $op\langle \mathtt{F}\rangle$ are easy.

For the case of a q-set $\mathtt{F} = \{\mathtt{f}_1, \ldots, \mathtt{f}_m\}$ we first notice for an arbitrary j in $\{1, \ldots, m\}$ that: $\mathsf{rate}(\mathtt{f}_j) \subseteq \mathsf{rate}(\mathtt{F})$ and therefore $\mathsf{rate}(\mathcal{D}^{\mathsf{H}}_v(\mathtt{f}_j)) \subseteq \mathsf{rate}(\mathcal{D}^{\mathsf{H}}_v(\mathtt{F}))$ for every $v \in D^*$. It follows that the rates of the derivatives $\mathcal{D}^{\mathsf{H}}_{u_1}(\mathtt{f}_j)$, ..., $\mathcal{D}^{\mathsf{H}}_{u_n}(\mathtt{f}_j)$ are pairwise disjoint. To show part (1), we observe that

$$\{i \mid \mathcal{D}^{\mathsf{H}}_{u_i}(\mathtt{F}) \neq \emptyset\} = \{i \mid \mathcal{D}^{\mathsf{H}}_{u_i}(\mathtt{f}_1) \cup \cdots \cup \mathcal{D}^{\mathsf{H}}_{u_i}(\mathtt{f}_m) \neq \emptyset\} = \textstyle\bigcup_{j=1}^m \{i \mid \mathcal{D}^{\mathsf{H}}_{u_i}(\mathtt{f}_j) \neq \emptyset\}.$$

By the induction hypothesis, the size of this set is bounded by $\sum_{j=1}^m \mathrm{size}(\mathtt{f}_j) = \mathrm{size}(\mathtt{F})$. Now,

$$\begin{aligned}
\mathsf{space}(\textstyle\bigcup_{i=1}^n \mathcal{D}^{\mathsf{H}}_{u_i}(\mathtt{F})) &= \mathsf{space}(\textstyle\bigcup_{i=1}^n \bigcup_{j=1}^n \mathcal{D}^{\mathsf{H}}_{u_i}(\mathtt{f}_j)) \\
&= \textstyle\sum_{j=1}^n \mathsf{space}(\bigcup_{i=1}^n \mathcal{D}^{\mathsf{H}}_{u_i}(\mathtt{f}_j)) \\
&\leq \textstyle\sum_{j=1}^n \mathrm{size}(\mathtt{f}_j)^2,
\end{aligned}$$

which is less than $(\mathrm{size}(\mathtt{f}_1) + \cdots + \mathrm{size}(\mathtt{f}_m))^2 = \mathrm{size}(\mathtt{F})^2$.

For the case of the query $\mathtt{h} = op\langle \mathtt{F}, \mathtt{G}\rangle$, we first recall that $\mathsf{rate}(h) = \mathsf{rate}(\mathtt{F}) \equiv \mathsf{rate}(\mathtt{G})$. So, the hypotheses of the lemma hold for both $\mathtt{F}$ and $\mathtt{G}$. This means that at most $\min(\mathrm{size}(\mathtt{F}), \mathrm{size}(\mathtt{G})) \leq \mathrm{size}(\mathtt{h})$ of the sets $\mathcal{D}^{\mathsf{H}}_{u_1}(\mathtt{h})$, ..., $\mathcal{D}^{\mathsf{H}}_{u_n}(\mathtt{h})$ are nonempty. For part (2), we have that

$$\begin{aligned}
\mathsf{space}(\textstyle\bigcup_{i=1}^n \mathcal{D}^{\mathsf{H}}_{u_i}(\mathtt{h})) &= \mathsf{space}(\{op\langle \mathcal{D}^{\mathsf{H}}_{u_i}(\mathtt{F}), \mathcal{D}^{\mathsf{H}}_{u_i}(\mathtt{G})\rangle \mid i = 1, \ldots, n\}) \\
&\leq \mathrm{size}(\mathtt{F}) + \big(\textstyle\sum_{i=1}^n \mathsf{space}(\mathcal{D}^{\mathsf{H}}_{u_i}(\mathtt{F}))\big) + \big(\textstyle\sum_{i=1}^n \mathsf{space}(\mathcal{D}^{\mathsf{H}}_{u_i}(\mathtt{G}))\big) \\
&\leq \mathrm{size}(\mathtt{F}) + \mathrm{size}(\mathtt{F})^2 + \mathrm{size}(\mathtt{G})^2,
\end{aligned}$$

which is less than $(1 + \mathrm{size}(\mathtt{F}) + \mathrm{size}(\mathtt{G}))^2 = \mathrm{size}(op\langle \mathtt{F}, \mathtt{G}\rangle)^2$.

State : Q-set $\mathtt{H}$ of type $\mathsf{QSET}\langle D, C\rangle$.

Initialization() : Set the state $\mathtt{H}$ to the singleton q-set $\{\mathtt{f}\}$.

Update(d) : Replace $\mathtt{H}$ by its hierarchical derivative $\mathcal{D}_d^{\mathsf{H}}(\mathtt{H})$.

Output() : If $\mathcal{E}(\mathtt{H})$ is the singleton multiset $\{c\}$, then return the value c. Otherwise, the output is undefined.

Fig. 6. Streaming evaluation algorithm for a strongly typed query $\mathtt{f} : \mathsf{QRE}\langle D, C\rangle$.

We handle now the case $\mathtt{h} = \mathtt{split}\langle \mathtt{F}, \mathtt{G}, op\rangle$. As discussed previously, the union $\mathcal{D}_{u_1}^{\mathsf{H}}(\mathtt{h}) \cup \cdots \cup \mathcal{D}_{u_n}^{\mathsf{H}}(\mathtt{h})$ of the derivatives is of the form

$$\mathtt{H} = \{\mathtt{split}\langle \mathcal{D}_{u_1}^{\mathsf{H}}(\mathtt{F}), \mathtt{G}, op\rangle, \ldots, \mathtt{split}\langle \mathcal{D}_{u_n}^{\mathsf{H}}(\mathtt{F}), \mathtt{G}, op\rangle, \\ op\langle a_1, \mathcal{D}_{v_1}^{\mathsf{H}}(\mathtt{G})\rangle, \ldots, op\langle a_n, \mathcal{D}_{v_m}^{\mathsf{H}}(\mathtt{G})\rangle\}$$

for some $a_1, \ldots, a_m \in A$ and $v_1, \ldots, v_m \in D^*$. Now, $\mathtt{H}$ is unambiguous and therefore the q-sets $\mathcal{D}_{u_1}^{\mathsf{H}}(\mathtt{F})$, ..., $\mathcal{D}_{u_n}^{\mathsf{H}}(\mathtt{F})$ are pairwise disjoint, and similarly the q-sets $\mathcal{D}_{v_1}^{\mathsf{H}}(\mathtt{G})$, ..., $\mathcal{D}_{v_m}^{\mathsf{H}}(\mathtt{G})$ are pairwise disjoint. From the induction hypothesis (part 1) for $\mathtt{F}$ and $\mathtt{G}$, we get that $\mathtt{H}$ is of size $\leq \mathrm{size}(\mathtt{F}) + \mathrm{size}(\mathtt{G})$, which implies part (1) for $\mathtt{h}$. To measure the space needed to represent $\mathtt{H}$, we first observe that the subquery $\mathtt{G}$ is shared by several queries in $\mathtt{H}$ and therefore we can replace every $\mathtt{G}$ occurrence with a pointer to a representation of $\mathtt{G}$. Total space:

$$\mathsf{space}(\mathtt{H}) \leq \mathrm{size}(\mathtt{F})^2 + 2 \cdot \mathrm{size}(\mathtt{F}) + \mathrm{size}(\mathtt{G})^2 + \mathrm{size}(\mathtt{G}),$$

which is less than $(1 + \mathrm{size}(\mathtt{F}) + \mathrm{size}(\mathtt{G}))^2 = \mathrm{size}(\mathtt{h})^2$.

Finally, we consider the case $\mathtt{h} = \mathtt{iter}\langle \mathtt{F}, \mathtt{G}, op\rangle$ of iteration. As discussed previously, the union $\mathcal{D}_{u_1}^{\mathsf{H}}(\mathtt{h}) \cup \cdots \cup \mathcal{D}_{u_n}^{\mathsf{H}}(\mathtt{h})$ of the derivatives is of the form

$$\mathtt{H} = \{\mathtt{iter}\langle \mathcal{D}_{u_1}^{\mathsf{H}}(\mathtt{F}), \mathtt{G}, op\rangle, \ldots, \mathtt{iter}\langle \mathcal{D}_{u_n}^{\mathsf{H}}(\mathtt{F}), \mathtt{G}, op\rangle, \\ \mathtt{iter}\langle op\langle b_1, \mathcal{D}_{v_1}^{\mathsf{H}}(\mathtt{G})\rangle, \mathtt{G}, op\rangle, \ldots, \mathtt{iter}\langle op\langle b_n, \mathcal{D}_{v_m}^{\mathsf{H}}(\mathtt{G})\rangle, \mathtt{G}, op\rangle\}$$

for some $b_1, \ldots, b_m \in B$ and $v_1, \ldots, v_m \in D^*$. The total space requirements are:

$$\mathsf{space}(\mathtt{H}) \leq \mathrm{size}(\mathtt{F})^2 + 2 \cdot \mathrm{size}(\mathtt{F}) + \mathrm{size}(\mathtt{G})^2 + 3 \cdot \mathrm{size}(\mathtt{G}),$$

which is less than $(1 + \mathrm{size}(\mathtt{F}) + \mathrm{size}(\mathtt{G}))^2 = \mathrm{size}(\mathtt{h})^2$. □

Lemma 12 establishes, in particular, that the hierarchical derivative of any strongly typed q-set $\mathtt{F}$ w.r.t. any sequence is of size at most quadratic in the size of $\mathtt{F}$. Using this fact, we can prove the main theorem of this paper:

Theorem 13. The algorithm of Fig. 6 solves the evaluation problem for strongly typed QREs. It requires space and time-per-element that are constant in the length of the stream, and polynomial in the size of the query.

Proof. First, we observe that the hierarchial derivative is simply a different representation of the Brzozowski derivative, which is streamlined for space efficiency. This implies that $\mathcal{D}_d^{\mathsf{B}}(\mathtt{f})$, $\mathcal{D}_d^{\mathsf{A}}(\mathtt{f})$ and $\mathcal{D}_d^{\mathsf{H}}(\mathtt{f})$ are all $\equiv$-equivalent when

$\mathtt{f} : \mathsf{QRE}\langle D, C\rangle$ is strongly typed. The algorithm of Fig. 6 satisfies the invariant: after consuming input $w \in D^*$, the q-set $\mathtt{H}$ is equal to the hierarchical derivative $\mathcal{D}^{\mathsf{H}}_w(\mathtt{f})$. The correctness of the algorithm then follows immediately from the semantic agreement of the hierarchical derivative with the Brzozowski derivative. At every step of the computation the state $\mathtt{H}$ is of the form $\mathcal{D}^{\mathsf{H}}_w(\mathtt{f})$, where w is the input sequence seen so far. Lemma 12 give us immediately that $\mathtt{H}$ can be represented using space that is quadratic in the size of the input query. It follows that the time to process each element is also quadratic in the query. □

Example 14. The queries $\mathtt{h}_1$, $\mathtt{h}_2$ below calculate the maximum and minimum respectively of two consecutive natural numbers, where $\mathtt{f} = \mathtt{atom}(\mathit{true}_{\mathbb{N}}, \mathit{id}_{\mathbb{N}})$. The top-level query $\mathtt{k}$ shown below is strongly typed, and it calculates the average of the maximum and minimum of the last two elements of the stream.

$$\begin{aligned} \mathtt{h}_1 &= \mathtt{split}(\mathtt{f}, \mathtt{f}, \max) & \mathtt{k}_1 &= \mathtt{split}(\mathtt{g}, \mathtt{h}_1, \pi_2) \\ \mathtt{h}_2 &= \mathtt{split}(\mathtt{f}, \mathtt{f}, \min) & \mathtt{k}_2 &= \mathtt{split}(\mathtt{g}, \mathtt{h}_2, \pi_2) \\ \mathtt{g} &= \mathtt{iter}(\mathtt{eps}(0), \mathtt{atom}(\mathit{true}_{\mathbb{N}}, \lambda x.0), \pi_2) & \mathtt{k} &= \mathrm{avg}(\mathtt{k}_1, \mathtt{k}_2) \end{aligned}$$

The Antimirov and hierarchical derivatives of $\mathtt{g}$ are the following:

$$\begin{aligned} \{\mathtt{g}'\} = \mathcal{D}^{\mathsf{A}}_w(\mathtt{g}) &= \{\mathtt{iter}((\lambda x.\pi_2(0,x))(\mathtt{eps}(0)), \mathtt{atom}(\mathit{true}_{\mathbb{N}}, \lambda x.0), \pi_2)\} \\ \mathtt{g}'' = \mathcal{D}^{\mathsf{H}}_w(\mathtt{g}) &= \{\mathtt{iter}\langle(\lambda x.\pi_2(0,x))\langle\mathtt{eps}(0)\rangle, \mathtt{atom}(\mathit{true}_{\mathbb{N}}, \lambda x.0), \pi_2\rangle\} \end{aligned}$$

for every $w \neq \varepsilon$. For the Antimirov derivative of $\mathtt{k}$ w.r.t. 3 we calculate:

$$\begin{aligned} \mathcal{D}^{\mathsf{A}}_3(\mathtt{k}_1) &= \{\mathtt{split}(\mathtt{g}', \mathtt{h}_1, \pi_2), (\lambda x.\pi_2(0,x))((\lambda x.\max(3,x))(\mathtt{f}))\} \\ \mathcal{D}^{\mathsf{A}}_3(\mathtt{k}_2) &= \{\mathtt{split}(\mathtt{g}', \mathtt{h}_2, \pi_2), (\lambda x.\pi_2(0,x))((\lambda x.\min(3,x))(\mathtt{f}))\} \\ \mathcal{D}^{\mathsf{A}}_3(\mathtt{k}) &= \{\mathrm{avg}(\mathtt{split}(\mathtt{g}', \mathtt{h}_1, \pi_2), \mathtt{split}(\mathtt{g}', \mathtt{h}_2, \pi_2)), \\ &\quad \mathrm{avg}(\mathtt{split}(\mathtt{g}', \mathtt{h}_1, \pi_2), (\lambda x.\pi_2(0,x))((\lambda x.\min(3,x))(\mathtt{f}))), \\ &\quad \mathrm{avg}((\lambda x.\pi_2(0,x))((\lambda x.\max(3,x))(\mathtt{f})), \mathtt{split}(\mathtt{g}', \mathtt{h}_2, \pi_2)), \\ &\quad \mathrm{avg}((\lambda x.\pi_2(0,x))((\lambda x.\max(3,x))(\mathtt{f})), \\ &\qquad (\lambda x.\pi_2(0,x))((\lambda x.\min(3,x))(\mathtt{f})))\} \end{aligned}$$

and then the Antimirov derivative $\mathcal{D}^{\mathsf{A}}_{35}(\mathtt{k})$ contains $3 \cdot 3 = 9$ queries. For the hierarchical derivative of $\mathtt{k}$ w.r.t. 3 we calculate:

$$\begin{aligned} \mathcal{D}^{\mathsf{H}}_3(\mathtt{k}_1) &= \{\mathtt{split}\langle\mathtt{g}'', \mathtt{h}_1, \pi_2\rangle, (\lambda x.\pi_2(0,x))\langle(\lambda x.\max(3,x))\langle\mathtt{f}\rangle\rangle\} \\ \mathcal{D}^{\mathsf{H}}_3(\mathtt{k}_2) &= \{\mathtt{split}\langle\mathtt{g}'', \mathtt{h}_2, \pi_2\rangle, (\lambda x.\pi_2(0,x))\langle(\lambda x.\min(3,x))\langle\mathtt{f}\rangle\rangle\} \\ \mathcal{D}^{\mathsf{A}}_3(\mathtt{k}) &= \{\mathrm{avg}\langle\mathcal{D}^{\mathsf{H}}_3(\mathtt{k}_1), \mathcal{D}^{\mathsf{H}}_3(\mathtt{k}_2)\rangle\} \end{aligned}$$

In the hierarchical derivative $\mathcal{D}^{\mathsf{H}}_{35}(\mathtt{k})$, on the other hand, the subexpressions $\mathcal{D}^{\mathsf{H}}_{35}(\mathtt{k}_1)$ and $\mathcal{D}^{\mathsf{H}}_{35}(\mathtt{k}_2)$ contain a total of $3 + 3 = 6$ queries. This example illustrates the quadratic blowup for Antimirov derivatives, which is avoided using hierarchical derivatives.

6 Conclusion

This paper introduces syntactic derivatives for the Quantitative Regular Expressions (QREs) of [5]. The most natural generalization of the classical Brzozowski derivative to QREs is appropriate for the so-called multiset semantics of QREs, which records the possibility of several (finitely many) outputs. Since QREs are meant to describe well-defined functions on streams, we consider a projection of the multiset semantics into the so-called *unambiguous semantics*. Using a representation of derivatives that is inspired from Antimirov's variant of classical derivatives, we obtain an evaluation algorithm for QREs with streaming space and time complexity that is constant in the stream and exponential in the query. We then restrict attention to the strongly-typed QREs, also considered in [5], which admit more efficient streaming evaluation. We devise a novel representation of derivatives on QREs, which we call *hierarchical derivatives*, and we obtain an evaluation algorithm that streaming space and time complexity that is polynomial in the query. This matches the bounds of [5] and [21], but the algorithm presented here is much easier to describe, prove correct, and analyze.

The treatment of QRE evaluation using derivatives is a significant step towards developing automata models for QREs that play the same role as NFAs do for plain regular exressions. The definition of the space-efficient hierarchical derivatives of Sect. 5 suggests that the *parallel evaluation* of $\mathtt{f}$, $\mathtt{g}$ in subqueries of the form $op(\mathtt{f}, \mathtt{g})$ and some form of *hierarchical nesting* are essential features of a model that can support efficient evaluation of QREs. A hierarchical automaton model for the streaming computation of quantitative queries is described in [6], but its precise relationship to the QREs of [5] remains to be clarified. Finding the appropriate model of automata for QREs is an important direction for future work, since it would also open the door for query optimization by applying equivalence preserving transformations on the automata.

References

1. Abadi, D.J., Carney, D., Cetintemel, U., Cherniack, M., Convey, C., Lee, S., Stonebraker, M., Tatbul, N., Zdonik, S.: Aurora: A new model and architecture for data stream management. The VLDB Journal 12(2), 120–139 (2003)
2. Alon, N., Matias, Y., Szegedy, M.: The space complexity of approximating the frequency moments. Journal of Computer and System Sciences 58(1), 137–147 (1999)
3. Alur, R., D'Antoni, L.: Streaming tree transducers. In: Proceedings of the 39th International Colloquium on Automata, Languages, and Programming (ICALP '12). pp. 42–53 (2012)
4. Alur, R., D'Antoni, L., Deshmukh, J., Raghothaman, M., Yuan, Y.: Regular functions and cost register automata. In: Proceedings of the 28th Annual ACM/IEEE Symposium on Logic in Computer Science (LICS '13). pp. 13–22 (2013)
5. Alur, R., Fisman, D., Raghothaman, M.: Regular programming for quantitative properties of data streams. In: Proceedings of the 25th European Symposium on Programming (ESOP '16). pp. 15–40 (2016)

6. Alur, R., Mamouras, K., Stanford, C.: Automata-based stream processing (2017), to appear in ICALP'17
7. Antimirov, V.: Partial derivatives of regular expressions and finite automaton constructions. Theoretical Computer Science 155(2), 291–319 (1996)
8. Arasu, A., Babu, S., Widom, J.: The CQL continuous query language: Semantic foundations and query execution. The VLDB Journal 15(2), 121–142 (2006)
9. Babu, S., Widom, J.: Continuous queries over data streams. ACM Sigmod Record 30(3), 109–120 (2001)
10. Bonchi, F., Bonsangue, M.M., Hansen, H.H., Panangaden, P., Rutten, J.J.M.M., Silva, A.: Algebra-coalgebra duality in brzozowski's minimization algorithm. ACM Transactions on Computational Logic (TOCL) 15(1), 3:1–3:29 (2014)
11. Brzozowski, J.A.: Derivatives of regular expressions. Journal of the ACM 11(4), 481–494 (1964)
12. Colcombet, T.: Forms of determinism for automata (invited talk). In: Proceedings of the 29th International Symposium on Theoretical Aspects of Computer Science (STACS '12). Leibniz International Proceedings in Informatics (LIPIcs), vol. 14, pp. 1–23 (2012)
13. Colcombet, T.: Unambiguity in automata theory. In: Proceedings of the 17th International Workshop on Descriptional Complexity of Formal Systems (DCFS '15). pp. 3–18 (2015)
14. Courcelle, B.: Monadic second-order definable graph transductions: A survey. Theoretical Computer Science 126(1), 53–75 (1994)
15. Datar, M., Gionis, A., Indyk, P., Motwani, R.: Maintaining stream statistics over sliding windows. SIAM Journal on Computing 31(6), 1794–1813 (2002)
16. Droste, M., Kuich, W., Vogler, H. (eds.): Handbook of Weighted Automata. Springer (2009)
17. Flajolet, P., Martin, G.N.: Probabilistic counting algorithms for data base applications. Journal of Computer and System Sciences 31(2), 182–209 (1985)
18. Greenwald, M., Khanna, S.: Quantiles and equidepth histograms over streams. In: Garofalakis, M., Gehrke, J., Rastogi, R. (eds.) Data Stream Management: Processing High-Speed Data Streams. Springer (2016)
19. Hopcroft, J.E., Motwani, R., Ullman, J.D.: Introduction to Automata Theory, Languages, and Computation. Pearson Education, 3rd edn. (2006)
20. Lombardy, S., Sakarovitch, J.: Derivatives of rational expressions with multiplicity. Theoretical Computer Science 332(1), 141–177 (2005)
21. Mamouras, K., Raghothaman, M., Alur, R., Ives, Z.G., Khanna, S.: StreamQRE: Modular specification and efficient evaluation of quantitative queries over streaming data (2017), to appear in PLDI'17
22. Schützenberger, M.P.: On the definition of a family of automata. Information and control 4(2), 245–270 (1961)