A machine model for data stream computation

Sumit Ganguly

Indian Institute of Technology, Kanpur

Abstract. We consider online and space-bounded computations in the data stream processing model where a stream is a sequence of records of the form $(i, 1)$ signifying insertion of item i, or, $(i, -1)$, signifying deletion of item i, where, $i \in \{1, 2, \ldots, n\}$. This model finds applications in network monitoring, approximate query answering in databases, computational geometry, where, items signify points in d-dimensional space, graph streams, where, items signify edges of a streaming graph and compressed sensing. We abstract computations over data streams using the *stream automata* model of computation. Our main result is that a certain natural generalization of the transition function of any stream automaton is essentially a linear mapping. This is used to derive space lower bounds for deterministic data stream computations.

1 Introduction

Recently, there has been substantial interest in space-bounded online computations, called data stream processing, from networking and database researchers, algorithm designers and computational complexity theorists. In this paper, we consider data stream processing, where, the input stream of records can correspond to both insertion and deletion of basic entities. A data stream [2] is modelled as a sequence of records (i, δ), where, $i \in [n] = \{1, 2, \ldots, n\}$ and $\delta \in \{+1, -1\}$, where, $\delta = 1$ signifies an insertion and $\delta = -1$ signifies deletion. The frequency f_i of an item i is defined as $f_i = \sum_{(i,\delta) \text{ appears in stream}} \delta$. Among the well-studied problems in this model include, finding approximate frequent items [29,10,28,6,8], approximate ϕ-quantiles [20,19,8,21], approximately optimal histograms using b-buckets and approximately optimal b-coefficient wavelet representation [18,25], and many others [1,27]. Other examples include, the streaming graph model [12,13,11], where, the stream is a sequence of records of the form (e, δ), where, e is an edge and $\delta \in \{1, -1\}$ and, computational geometry applications [14,26], where, the stream is a sequence of records of the form (P, δ), where, $P \in [n]^d$ is a d-dimensional point and $\delta \in \{+1, -1\}$.

1.1 Summary

In this work, we present a generic model of data stream computations, called *stream automata*. A deterministic stream automaton that is space-bounded to use s bits is a Turing machine with a finite state control that uses two tapes, namely, an s-bit work-tape, and a one-way input tape that is potentially unbounded. The first two records on the input tape signify m and n, where, $[n] = \{1, 2, \ldots, n\}$ is the domain of the inputs and m represents an upper bound on the frequencies, that is, $|f_i| \le m$, for $1 \le i \le m$. The remainder of the input tape consists of the stream records of the form (i, δ). A randomized stream automaton uses an additional s-bit tape consisting of the random coin tosses that is initialized at the beginning of the computation and is read (but not written) any number of times during the computation. The configuration of a stream automaton is identified with the triple (q, h, w), where, q is the current state of the finite control, h is the current position on the work-tape and w is the current contents of the work-tape. The configuration *transition function* is the mapping that takes one configuration of the stream automaton to the next configuration, after processing a stream record of the form (i, δ). We show that for every deterministic stream automaton $\mathcal{A}$ using s bit bounded work-tape, there exists another s-bit bounded deterministic stream automaton $\mathcal{B}$ whose generalized transition function is isomorphic to a linear mapping. More precisely, the configuration of $\mathcal{B}$ is isomorphic to Af, where, A is a $w \times n$ matrix, f is the n-dimensional frequency vector and $w \le s$. Further, Af can be expressed using $s + \log s + O(1)$ bits.

Deterministic computations over update streams. We now discuss the implications of this result for update data streams. Firstly, it effectively shows that computations over update streams are essentially *path-independent*, that is, the configuration of the stream automaton is dependent only on the value of the current frequency vector f and not on the precise sequence of updates. This is true since, effectively, any stream automaton can be equivalently simulated by another automaton whose configuration transition function is a linear map of the form Af. Thus, if $f = g$, then, $Af = Ag$. This conclusion is to be contrasted with computations for insert-only (and sliding window) models of streams that are not path-independent, and hence, are not expressible as linear mappings. Examples include the Misra-Gries algorithm for finding frequent items [29] and the Greenwald-Khanna algorithm for finding approximate quantiles [21].

A second consequence of linear mapping is that all deterministic computation over data streams is naturally composable. Composability refers to the property that the summary structure for the union of two (or more) independent streams can be easily composed from the summary structures of the independent streams themselves. This property is highly desirable in distributed systems. Clearly, any linear mapping is composable, since, $A(f + g) = Af + Ag$, where, Af and Ag represent the summary structures of the computations over independent streams whose frequency vectors are f and g respectively. This simple consequence of linear mapping is once again in contrast with computations for insert-only streams, where, computations are not easily composable (e.g., the Misra-Gries algorithm, Greenwald-Khanna algorithm, etc.).

The linear mapping property is used to derive space lower bounds for approximate and deterministic computation of functions over data streams.

Randomized computations over update streams. Randomized computations over update streams are modelled using stream automata that uses three tapes, namely, a one-way input tape containing the data stream, a work tape limited to s-bits and an s-bit tape containing the random bits that are to be used during the computation. The random bits are written after the values of the input parameters, m and n are read from the input tape; thereafter, the random tape is used as a read-only tape. The main result for deterministic computation extends to this scenario, namely, the generalized transition function–which is now a function of the random bit string ρ, is essentially a linear mapping.

Organization The remainder of the paper is organized as follows. In Section 2, we present the stream automaton computational model and compare it with related work. Section 3 presents the basic properties of computations using stream automata. Finally, we conclude in Section 5.

2 Stream Automaton

In this section, we present a machine model for space bounded, online computations involving involving both insertion and deletion operations in the input sequence.

A stream automaton A is modelled as a Turing machine (TM) with two tapes, namely, a one-way read-only input tape and a work tape. The input tape is divided into two parts as follows. The first part contains the input size parameters m and n. The second part of the input tape contains the sequence of updates of the form $(i, 1)$ or $(i, -1)$, where, $i \in [n]$ and such that $|f_i| \leq m$ for all items i at each stage of processing. [1] The language accepted by A is defined as the set of input sequences (streams) for which A terminates in an accepting state (we assume that A always terminates on all inputs). The frequency vector of the stream defines the current input, after processing the insertions and deletions. We therefore place an important restriction regarding the language accepted by a stream automaton, namely, that it is based on the frequency vector of the input stream only. This property is a common characteristic of all known summary structures for update streams [1,6,8,9,15,16]. It is also true in many online computational applications that handle both insertions and deletions, including, online computational geometry applications, graph streaming applications, linear algebra-based applications, etc.. An implication of this property is that if σ and τ are input sequences with the same frequency vector f, then, either both σ and τ are accepted by A, or, both are rejected by A.

Related Work. The main difference between our model and the existing models is in the notion of the input size. In general, the input size is modelled as a function of the size of the input string. In our case, the input size parameters are given to be m and n, but otherwise, the size of the update sequence (i.e., the

[1] This can be extended to randomized algorithms, where, an additional read-many, write-once tape is used to store the random bits used by the machine.

stream) can be unbounded. A well-studied class is $1L$, or the class of languages that can be accepted by a TM that reads its input tape unidirectionally and uses $O(\log I)$ space, where, I is the length of the input. Recently [22,23,24] have proposed a model for proving lower bounds for external memory and streaming computations. Their model allows TMs that have two kinds of tapes, namely, length unbounded tapes that are bounded in total number of head reversals $r(I)$, and, bounded length tapes that use a total of $s(I)$ cells, where, I is the input size. They define the complexity class $ST(r(I), s(I))$ as the set of languages that can be accepted by a Turing machine that is subject to the above constraints. Streaming computations are intended to be modelled by letting $r(I) = 0$ (i.e., no reversals). The problem with the classes such as $1L$ or $ST(\cdot, \cdot)$ is that the workspace is determined by the size of the input, whereas, for update streams, it is possible for very long update sequences to have $I = 0$ (or, very small values of I), making this notion unsuitable.

3 Computing using stream automata

In this section, we prove properties about computations using stream automata. We assume that the alphabet of the work tape of a stream automaton A is binary and is bounded to use at most s bits. Fix a value of $m, n \geq 2$, and consider the execution of A on the class of inputs with these input values for m and n.

Transition function. We consider the configuration of the TM A just before it processes a new input record, and after it has read the input parameters m and n. The configuration can be modelled as the triple (q, x, w), where, $q \in Q$ is the current state of the finite control, x is the index of the current cell of the work tape, and w is the current contents of the work-tape. The processing of each record can be viewed as as a function $\oplus(a, v)$, where, a is the current configuration of A, and v is the current record (of the form $(i, \pm 1)$) on the input tape. The transition function $\oplus = \oplus_{m,n}$ is in general a function of m and n. Let $\Delta_n = \{e_1, e_2, \ldots, e_n, -e_1, -e_2, \ldots, -e_n\}$ denote a set of vectors, called transition vectors. A stream update of the form $(i, 1)$ is equivalently viewed as e_i and $(i, -1)$ as $-e_i$, where, e_i is the i^{th} standard basis vector of $\mathbb{R}^n$. The transition function is written in infix form as $a \oplus x$ with the assumption that the $\oplus$ operator is left-associative. If transition vectors $u_1, u_2, \ldots, u_r$ belonging to Δ appear consecutively on the input tape, the notation $u_1 \circ u_2 \circ \ldots \circ u_r$ is used to denote the sequence. Similarly, the concatenation of two sequences of vectors σ and τ is denoted by $\sigma \circ \tau$. The *trace* of a vector sequence $\sigma = u_1 \circ u_2 \circ \ldots \circ u_r$, where, $u_1, u_2, \ldots, u_r \in \Delta$, is defined as $\text{tr}\, \sigma = u_1 + u_2 + \ldots + u_r$. Let O denote the initial configuration of A. During the processing of update sequence, the stream automata can be modelled as a deterministic finite automaton $(\mathcal{C}, \oplus, O, \mathcal{C}_F)$, where, $\mathcal{C} = \mathcal{C}_{m,n}$ denotes the set of all configurations reachable from O on input streams with parameters m and n and $\oplus$ is the transition function. $\mathcal{C}_F$ denotes the subset of configurations from $\mathcal{C}$ that are accepted by A.[2]

We classify stream automata into one of three types, depending on the nature of the transition functions, namely, path independent, path reversible and non-reversible. An automaton A is said to be *path independent*, provided, for any input stream σ, the configuration of A, after starting from O and after processing σ, is dependent only on tr σ. The automata is said to be *path reversible* provided, for any configuration a and feasible input sequence σ, $a \oplus \sigma \circ \sigma^r = a$, where, σ^r is the reverse of σ, that is, if $\sigma = u_1 \circ u_2 \circ \ldots \circ u_r$, then, $\sigma^r = -u_r \circ -u_{r-1} \circ \ldots \circ -u_1$. The intermediate vectors, namely, $u_1, u_1 + u_2, \ldots, u_1 + \ldots + u_n$ must be all feasible vectors. Finally, *non-reversible* automata are those that are not path reversible.

All known summary structures for update streams (i.e., practical stream automata) have path independent transition functions, including, SKETCHES [1], COUNT-MIN sketch [8], random subset sums [19], COUNTSKETCH [6], Coppersmith and Kumar's sketches [7], Indyk and Woodruff's structure for estimating frequency moments [27], the HSS structure [3,4], the k-set structure [17] and CR-precis [16] etc.. We also note that there exist well-known and efficient algorithms for insert-only streams whose transition functions are *not* path reversible, for example, the Frequent algorithm of [29,10,28,5], finding approximate ϕ-quantiles [21], etc..

3.1 Path independent automata

Consider the execution of a path independent automaton A for a fixed value of m and n. The space of feasible vectors is denoted by $F = F_{m,n} = \{f \in \mathbb{Z}^n \mid |f_i| \leq m\}$. By path independence, the following function Ψ that maps feasible frequency vectors to configurations, is well-defined.

$$\Psi(f) = O \oplus \sigma, \text{ for any } \sigma \text{ s.t. tr } \sigma = f$$

Further, it follows that, if $\Psi(f) = \Psi(g)$, then, for all update vectors $e \in \Delta$, $\Psi(f + e) = \Psi(g + e)$. It follows by induction that for all vectors h such that $f + h$ and $g + h$ are feasible, $\Psi(f + h) = \Psi(g + h)$. Let $K = \{h \mid \Psi(h) = \Psi(0)\}$. If $h \in K$ and $2h$ is a feasible vector, then, $\Psi(h + h) = \Psi(0 + h) = \Psi(0)$, or that, $2h \in K$. Further, if $h \in K$, then, $-h$ is feasible, and therefore, $\Psi(0) = \Psi(h - h) = \Psi(0 - h) = \Psi(-h)$, or, that $-h \in K$. By induction, we obtain that for any integer c such that ch is feasible and $h \in K$, $ch \in K$. Suppose that $g, h \in K$. Then, $-g \in K$ and therefore, $\Psi(0) = \Psi(-g) = \Psi(h)$. It follows that $\Psi(-g+g) = \Psi(0) = \Psi(g+h)$ or that $g+h \in K$. We therefore conclude that K *is a linear subspace of a vector space* V *of dimension* n *over a field* $\mathbb{F} = \mathbb{F}_{m,n}$ *that is, either, (a)* $\mathbb{Q}$ *or an extension of* $\mathbb{Q}$*, or, (b) a finite field of characteristic at*

[2] Acceptance in a TM is based on whether the current state is an accepting state. Our generalization is motivated by typical data stream processing, where, after A processes the stream, the analysis is done by another TM that takes the current configuration as input and either accepts or rejects it. This is discussed in the generalization later.

least $2m$, in order to distinguish between the elements $-m, \ldots, m$. Equivalently, K can be expressed as follows.

$$K = \{h \in F_{m,n} \mid Lh = 0\}, \text{ where, } L : \mathbb{F}^n \rightarrow \mathbb{F}^w \text{ is a linear mapping.}$$

Given a feasible vector g, define the coset $K + g = \{h \mid h \text{ is feasible and } g - h \in K\}$. Consider the quotient space F/K defined as the set of cosets $\{K + g \mid g \text{ is feasible}\}$ and define the addition of cosets as follows: $(K + g) + (K + h) = K + (g + h)$. The coset-addition is well-defined, since, K is a linear space. The set of reachable configurations is isomorphic to the *quotient* space, which is isomorphic to the range of L. Further, if $b = a \oplus e$, and $\Psi(f) = a$, then, a and b can be identified with $K + f$ and with $K + f + e$, respectively. Therefore, $(K + f + e) - (K + f) = K + e$ and can be represented as Le. Suppose that the rank of L is r. Let $L_{i_1}, \ldots, L_{i_r}$ denote the linearly independent columns of L. Thus, $\sum_{j=1}^{r} c_j L_{i_j} = \sum_{j=1}^{r} d_j L_{i_j}$ iff $c_j = d_j$, for each $j = 1, 2, \ldots, s$. There are a possible of $(2m + 1)$ feasible values of each c_j, and therefore, at least $(2m + 1)^r$ possible values of $\sum_{j=1}^{r} c_j L_{i_j}$. Each such value is isomorphic to a distinct configuration of A. Therefore, the number of configurations $|Q| \cdot s \cdot 2^s$ satisfies

$$|Q| \cdot s \cdot 2^s \geq (2m + 1)^r, \text{ or } r \leq \frac{1}{\log(2m + 1)} \left(s + \log s + \log|Q|\right) \quad . \qquad (1)$$

The space of reachable configurations of A is isomorphic to the set F/K. Further, $a \oplus e = a' + Le$, where, a' is the element of F/K that is isomorphic to a. This implies that $O \oplus \sigma = O' + L(\text{tr } \sigma)$; in other words, the $\oplus$ function is an affine function. It is a linear function provided $O' = 0$, which is what all existing summary structures assume [6,8,9,15,16]. We summarize the above discussion in the following lemma.

Lemma 1. *Let A be a path independent automaton that uses s bits on the work tape and has starting configuration O. For every $m, n \geq 2$, there exists a linear function $L_{m,n} : \mathbb{F}_{m,n}^n \rightarrow \mathbb{F}_{m,n}^{r(m,n)}$ of rank $r(m,n)$ with kernel $K_{m,n}$ satisfying the following properties.*

1. *$\mathbb{F}_{m,n}$ is either (a) a finite field of characteristic at least $2m$ and of size at least n, or, (b) is a field of characteristic 0.*
2. *$\mathcal{C}_{m,n}$ is isomorphic to $F_{m,n}/K_{m,n}$; let $\lambda_{m,n}$ denote the isomorphism.*
3. *$O \oplus \sigma = \lambda_{m,n}(O) + L_{m,n}(tr\ \sigma)$, for any update stream σ corresponding to input parameters m and n. In general, $C \oplus \sigma = \lambda_{m,n}(C) + L_{m,n}(tr\ \sigma)$.*
4. *$r(m,n) \leq r \frac{1}{\log(2m+1)} \left(s + \log s + \log|Q_A|\right)$.* □

3.2 Path Reversible Computations

In this section, we study stream automata whose transition function is path reversible, that is, for any feasible stream $\sigma = u_1 \circ u_2 \circ \ldots \circ u_r$ and configuration a, $a \oplus \sigma \circ \sigma^r = a$, where, $\sigma^r = -u_r \circ -u_{r-1} \circ \ldots - u_1$.

We say that a configuration a can be attained by a frequency vector f if there exists a stream of update vectors σ such that $O \oplus \sigma = a$ and tr $\sigma = f$. Let $\Phi(f)$ denote the set of attainable configurations by f, that is, $\Phi(f) = \{O \oplus \sigma \mid \text{tr } \sigma = f\}$. Let $K = \{f \mid O \in \Phi(f)\}$. If $f \in K$, then, there exists a sequence σ such that $O = O \oplus \sigma$ and tr $\sigma = f$. Therefore, $O \oplus \sigma \circ \sigma = O \oplus \sigma = O$, and tr $(\sigma \circ \sigma) = \text{tr } \sigma + \text{tr } \sigma = 2f$. Therefore, $O \in \Phi(f+f)$, or that $2f \in K$, assuming $2f$ is a feasible frequency vector. More importantly, by path reversibility, $O = O \oplus \sigma^r$ and tr $\sigma^r = -f$. Therefore, $O \in \Phi(-f)$. Generalizing, for $f \in K$ and for all $c \in \mathbb{Z}$ such that cf is feasible, $cf \in K$. Now suppose that $f, g \in K$ and $f+g$ is feasible. Therefore $O \in \Phi(g) \cap \Phi(f)$ and therefore, there exist update vector sequences σ and τ such that tr $\sigma = g$, tr $\sigma = f$ and $O = O \oplus \sigma = O \oplus \tau$. Thus, $O \oplus (\tau \circ \sigma) = (O \oplus \tau) \oplus \sigma) = O \oplus \sigma = O$, and, tr $(\tau \circ \sigma) = \text{tr } (\tau) + \text{tr } (\sigma) = f + g$. Therefore, $f + g \in K$. It follows that K is a linear subspace and can be written as the kernel of a linear transformation $L : \mathbb{F}^n \rightarrow \mathbb{F}^w$, that is, $K = \{f \mid Lf = 0\}$. As before, we are led to the quotient space $(F/K, +)$, where, $F/K = \{K + g \mid g \text{ feasible}\}$ and the sum of two cosets $K + f$ and $K + g$ is defined as $K + (f + g)$. Suppose that $g \in K + f$, or that, $O \in \Phi(g - f)$. If $a \in \Phi(f)$, that is, there exists σ such that $O \oplus \sigma = a$ and tr $\sigma = f$, then, it follows that $a \in \Phi(f + g - f) = \Phi(g)$, since, $O \in \Phi(g - f)$. It follows that $\Phi(f) \subset \Phi(g)$ and vice-versa, that is, $\Phi(f) = \Phi(g)$.

Recall that we have made a basic assumption that queries are based on the frequency vector only. For deterministic computations, this means that if any configuration in $\Phi(f)$ is accepted by A, then, all configurations in $\Phi(f)$ must be accepted by A. Therefore, if $\Phi(f) = \Phi(g)$, it means that the automaton A cannot distinguish between f and g, that is, A either rejects all input streams σ with tr $\sigma = f$ or tr $\sigma = g$, or, accepts all σ with tr $\sigma = f$ or tr $\sigma = g$. Thus, the notion that the coset $K+g$ is either accepted or rejected is well-defined, that is, A either accepts every stream σ such that tr $\sigma \in K+g$, or rejects every such stream. This analysis shows that the set of reachable configurations of the work tape of A is isomorphic to F/K. Essentially, this shows that path reversible automata are path independent. We summarize the above discussion in the following lemma.

Lemma 2. *For each value of m, n, if the transition function of A is path reversible, then, it is path independent.* □

3.3 Non-reversible automata

In this section, we study non-reversible automata. Keep the parameters m and n fixed. An *edge-reversible* automaton is an automaton whose transition function satisfies the property that $a \oplus v_j \circ -v_j = a$, for $a \in \mathcal{C}$ and $v_j \in \Delta$. We first note that path non-reversible automata are also edge non-reversible (as expected).

Fact 3 *An automaton is path reversible iff it is edge-reversible.*

Proof. Clearly, path reversibility implies edge-reversibility. To show the converse, suppose that the given automaton is not path reversible, that is, there exists a configuration $a \in \mathcal{C}$ and a feasible stream σ such that $a \oplus \sigma \circ \sigma^r = b$ and $a \neq b$. Let $\sigma = u_1 \circ u_2 \circ \ldots \circ u_r$, $a_0 = a$ and $a_j = a \oplus u_1 \circ u_2 \circ \cdots \circ u_j$, for

$j = 1, 2, \dots, r$. Let $b = a_r = b_r$ and let $b_{r-j} = b \oplus -u_r \circ -u_{r-1} \circ \cdots \circ -u_j$, for $j = 1, 2, \dots, r$. Let k be the largest value of j such that $a_k \neq b_k$. Then, $b_{k+1} = a_{k+1} \circ u_{k+2} \circ \cdots \circ u_r \circ -u_r \circ -u_{r-1} \circ \cdots - u_{k+1}$ and $a_{k+1} = b_{k+1}$. Therefore, $b_k = b_{k+1} \circ -u_{r-k} = a_{k+1} \circ -u_{k+1} = a_k \circ u_{k+1} \circ -u_{k+1}$. □

For any sequence σ, let σ^j denote $\sigma \circ \sigma \circ \dots \circ \sigma$ repeated j times. For every configuration a and vector $v \in \Delta$ such that $a \oplus (v \circ -v) \neq a$, define the sequence of configurations

$$\mu_j(a, v) = a \oplus (v \circ -v)^j, \quad j \geq 0 \ .$$

Since the configuration space $\mathcal{C}$ is finite, the configurations in the sequence $\mu_j(a, v)$ repeat. Consider the smallest index $\kappa = \kappa(a, v)$ such that $\mu_\kappa(a, v) = \mu_{k'}(a, v)$, for some $k' > \kappa$ and let $\theta = \theta(a, v)$ be the smallest value l such that $\mu_\kappa(a, v) = \mu_{\kappa+\theta}(a, v)$. Define

$$\lambda(a, v) = \min_{\text{lexicographic}} \{\mu_j(a, v) \mid \kappa \leq j \leq \kappa + \theta - 1\}$$

Here the operator min returns the minimum in terms of some fixed lexicographic ordering of the configurations. We impose an arbitrary but fixed ordering on the transition vectors in $\Delta = v_1, v_2, \dots, v_{|\Delta|}$. For a given configuration a, define the following sequence of configurations.

$$a^{(0)} = a \text{ and } a^{(j)} = \lambda(a^{(j-1)}, v_{j \bmod |\Delta|}), \quad j = 1, 2, \dots \ .$$

In the sequence of configurations, $\{a^{(j)}\}_{j \geq 0}$, let $t = t(a)$ and $o = o(a)$ be the smallest indices such that $a^{(t)} = a^{(t+o)}$ and $o \equiv 0 \mod |\Delta|$. By finiteness of the configuration space $\mathcal{C}$, integers $t \geq 0$ and $o \geq 1$ exist. Let $[a]$ denote the following set of configurations.

$$[a] = \{a^{(t(a)+j)} \mid 0 \leq j \leq o(a) - 1\} \tag{2}$$

The set $[a]$ behaves as an equivalence class (and is dependent on the choice of the ordering of the vectors in Δ). Let $\mathcal{C}_F$ denote the set of configurations of A corresponding to acceptance. Note that, once again, we use a more general notion of accepting configurations, to model the fact that after the input stream is processed, a query is either accepted or rejected by processing the configuration of A immediately after the stream has been processed. With this view, we can model the processing of A on stream records as a DFA $(\mathcal{C}, \oplus, \Delta, \mathcal{C}_F)$.

We construct a deterministic finite automaton (DFA) $B = (\mathcal{C}', \odot, \Delta, F')$ with set of states $\mathcal{C}'$, input alphabet Δ, set of final states F' and transition function $\odot$. Let $\mathcal{C}' = \{[a] \mid a \in \mathcal{C}\}$, the set of final states as $F' = \{[a] \mid a \in \mathcal{C}_F\}$ and transition function $\odot$ defined as follows.

$$[a] \odot v_j = [\lambda(a^{(t(a)+j-1)} \oplus v_j)], \quad j = 1, 2, \dots, |\Delta| \ .$$

Lemma 4. *(1) B is well-defined, and (2) $L(B) = L(A)$.*

Proof. To prove (1), we have to show that the set of accepting configurations F' is well-defined. Suppose that $[a] = [b]$ and a is accepting. Further, there exists streams σ, τ such that $a \oplus \sigma = b \oplus \tau = a^{(t(a))}$, and tr $\sigma =$ tr $\tau = 0$. If a is accepting, then, so is $a^{(t(a))}$. Since, b is in $\mathcal{C}$ and therefore, reachable, there exists $f \in \Phi^{-1}(b)$. Hence, $f \in \Phi^{-1}(a^{(t(a))})$ and so, all streams with trace f are accepted by A. Thus, b is an accepting configuration. Thus, the set F' of accepting configurations well-defined.

We now prove statement (2) of the lemma. Suppose $a \oplus v_j = a'$, for $a, b \in \mathcal{C}$ and $v_j \in \Delta$. Then, $[a] \odot v_j = [b]$, where, $b = \lambda(a^{(t(a)+j-1)} \oplus v_j)$. Therefore, $\Phi^{-1}(a) + v_j \subset \Phi^{-1}(a') \cap \Phi^{-1}(b)$. Continuing, by induction, it follows that for any stream σ, if $a \oplus \sigma = a'$ and $[a] \odot \sigma = [b]$, then, $\Phi^{-1}(a) + \text{tr } \sigma \subset \Phi^{-1}(a') \cap \Phi^{-1}(b)$. Thus, $a' \approx b$, that is, a' is an accepting configuration iff b is an accepting configuration. Therefore, letting a to be the initial configuration O, we have $L(A) = L(B)$. □

Denote the operation of constructing a new DFA B from a given automaton A viewed as a DFA as $B = T(A)$. We can repeat the above process to obtain a series of automata

$$A, T(A), T(T(A)), \ldots \quad . \tag{3}$$

Let $A^s = (\mathcal{C}^s, \oplus^s, \Delta, F^s)$ denote the automaton $\underbrace{T(T(\cdots T(A)))}_{s \text{ times}}$, $s \geq 0$. Denote $\mathcal{C}^s = \{[a]_s \mid a \in \mathcal{C}^{s-1}\}$, where, the notation $[a]_s$ distinguishes the configuration aggregation at level $s - 1$, given by (2). Inductively, it follows that each configuration of $\mathcal{C}^s$ can be associated with a set of configurations of $\mathcal{C}$, and each configuration of $\mathcal{C}$ is associated with a unique configuration of $\mathcal{C}^s$. Define the relation $a \sim_s b$, if, a and b are both mapped to the same configuration in $\mathcal{C}^s$ is an equivalence relation, with equivalence classes $(a)_s$. The mapping $[a]_s \to (a)_s$ is a set isomorphism. Further, by construction, if $a \sim_s b$ then $a \sim_{s+1} b$.

Lemma 5. *In the sequence of automata given by* (3)*, there exists an index j such that $A^s = A^{s+1}$, A^s is edge-reversible, $|\mathcal{C}^s| \leq |\mathcal{C}|$ and $L(A^s) = L(A)$.*

Proof. Since, $a \sim_s b$ implies that $a \sim_{s+1} b$, it follows that $\sim_s$ is a refinement of the relation $\sim_{s+1}$. If $\sim_s$ is not the same as $\sim_{s+1}$, $|\mathcal{C}^{s+1}| < |\mathcal{C}^s|$. Thus, the sequence $|\mathcal{C}^j|_{j \geq 0}$ cannot be infinite. Let s be the first index such that $|\mathcal{C}^s| = |\mathcal{C}^{s+1}|$. This implies that $\mathcal{C}^s = \mathcal{C}^{s+1}$. Suppose that there exists $q \in \mathcal{C}^s$ and $v_j \in \Delta$ such that $q \oplus^s v_j \neq q$. In the fixed ordering that we have assumed of the members of Δ, let k be the smallest index such that $q \oplus^s v_k = q' \neq q$. Then, by construction, it follows that (q) contains q and q', and therefore, $|\mathcal{C}^{s+1}| < |\mathcal{C}^s|$. Since, $\mathcal{C}^s = \mathcal{C}^{s+1}$, therefore, $(q)_{s+1} = (q)_s$, and therefore, for every $q \in \mathcal{C}^s$ and every $v_j \in \Delta$, $q \oplus^s v_j = q$. Thus, A^s is edge-reversible. We have already shown that the language accepted by A is the same as that of A^1, which is the same as the language accepted by A^2, etc.. □

Relation to DFA state minimization. We note the following relation with the classical state minimization procedure (the Myhill-Nerode theorem) for deterministic finite automata (DFA). In the procedure we have followed, we treat the configuration graph as a deterministic finite automaton, whose states are the configurations and there is an edge from C_1 to C_2 labelled with v_j provided, after processing v_j in configuration C_1, the automaton A is in configuration C_2. It follows that there exists an equivalent automaton B with the minimum number of states, where, each state of B is a set of configurations of A. If the DFA B is not reversible, then, the procedure described earlier can be used to obtain a DFA with fewer states that accepts the same language as B. However, B has the fewest number of states. We conclude that the DFA state minimization procedure applied to the space of configurations of A is the desired path-reversible automaton. Combining the discussion in this section with the earlier discussion, we obtain the following theorem.

Theorem 1. *Let A be a stream automaton that uses s bits on the work-tape. For each value of $m \geq 1$ and $n \geq 2$, let $\mathcal{C} = \mathcal{C}_{m,n}$ denote the set of reachable configurations of A and $O = O_{m,n}$ denote the initial configuration just prior to processing the stream. Let $F = \{f \in \mathbb{Z}^n \mid |f_i| \leq m\}$. Then, there exists a field $\mathbb{F} = \mathbb{F}_{m,n}$ and a linear function $L = L_{m,n} : \mathbb{F}^n \to \mathbb{F}^{r(m,n)}$ with kernel $K = K_{m,n}$ and a many-to-one mapping $\lambda = \lambda_{m,n} : \mathcal{C}_{m,n} \to F/K$ with the following properties.*

1. *$\mathbb{F}$ is either (a) a finite field of characteristic at least $2m$ and of size at least n, or, (b) is a field of characteristic 0 that is either $\mathbb{Q}$ or extends it.*
2. *$\mathcal{C}$ is isomorphic to F/K via the function λ (say).*
3. *$\lambda(O \oplus \sigma) = \lambda(O) + L_n(tr\ \sigma)$ for any legal update stream σ.*
4. *λ preserves acceptance, that is, for configurations C_1 and C_2 such that $\lambda(C_1) = \lambda(C_2)$, C_1 is an accepting configuration if and only if C_2 is too.*
5. *$r(m,n) \leq \frac{1}{\log(2m+1)}(s + \log s + \log|Q_A|)$.* □

3.4 Generalization

In practical scenarios, data stream processing is done in two functional parts, namely, the processing of input stream records, done by A and the query processing, done by a TM B. These two parts may interleave over time. The processing of stream records is done using a limited work-space of s bits. The query processing sub-routine B takes a query string as an input, that is appended to the input tape, and uses the current contents of the work-space to answer the query. We say that the stream automaton accepts its input stream and query string, provided, after the complete round of processing, B terminates in an accepting state. Although this model appears to be more general than that discussed earlier, the salient properties remain the same. For each query string *qstr*, Theorem 1 holds subject to the assumption that if configurations a and b are reachable from O using input streams σ and τ such that tr σ = tr τ, then, a is an accepting configuration if and only if b is too. Theorem 1 shows that the transition function is essentially a linear function L. Hence, for a fixed query string, Theorem 1 continues to hold.

3.5 Randomized Stream Automata

A randomized stream automaton is a stream automaton that uses an additional tape for the random bits that is space-bounded to use at most s bits. The contents of the random bits tape are written at the beginning of the computation, immediately after the input parameters m and n are read. Thereafter, the random bits tape functions as a read-only tape. Suppose that ρ denotes the contents of the s-bit random tape written just prior to the processing of the input stream records. The subsequent computation can be viewed as a deterministic stream automaton that is a function of ρ. Theorem 1 continues to hold for each choice of ρ, that is, the linear mapping $L = L_{m,n,\rho}$ and the analogous parameters are now additionally a function of the random bit string ρ. We restate the statement for randomized automata.

Corollary 1. *Let A be a randomized stream automaton that uses s bits on the work-tape and s bits on the random tape. For each value of $m \geq 1$ and $n \geq 2$, let $\mathcal{C} = \mathcal{C}_{m,n}$ denote the set of reachable configurations of A and $O = O_{m,n}$ denote the initial configuration just prior to processing the stream. Let $F = \{f \in \mathbb{Z}^n \mid |f_i| \leq m\}$ and ρ be the s-bit string that occupies the random tape. Then, there exists a field $\mathbb{F} = \mathbb{F}_{m,n}$ and for every s-bit choice of the random string ρ, there exists a linear function $L = L_{m,n,\rho} : \mathbb{F}^n \to \mathbb{F}^{r(m,n)}$ with kernel $K = K_{m,n,\rho}$ and a many-to-one mapping $\lambda = \lambda_{m,n,\rho} : \mathcal{C}_{m,n} \to F/K$ with the following properties.*

1. *$\mathbb{F}$ is either (a) a finite field of characteristic at least $2m$ and of size at least n, or, (b) is a field of characteristic 0 that is either $\mathbb{Q}$ or extends it.*
2. *$\mathcal{C}$ is isomorphic to F/K via the function λ .*
3. *$\lambda(O \oplus \sigma) = \lambda(O) + L(tr\ \sigma)$ for any legal update stream σ.*
4. *λ preserves acceptance, that is, for configurations C_1 and C_2 such that $\lambda(C_1) = \lambda(C_2)$, C_1 is an accepting configuration if and only if C_2 is too.*
5. *$r(m,n) \leq \frac{1}{\log(2m+1)}(s + \log s + \log|Q_A|)$.* □

4 Lower bounds

We present lower bounds that follow from Theorem 1. Let $\mu = \mu_{m,n}$ be any function over a data stream with input parameters m and n, for example, the L_p norm, ϕ-quantile, entropy and higher order entropy, etc.. The kernel of μ is defined as the set of frequency vectors $Z_{m,n}(\mu) = \{f \mid \mu(f) = 0\}$.

Lemma 6. *Suppose a function $\mu_{m,n}$ over a data stream having a non-empty kernel is approximated to within some multiplicative factor by a deterministic stream automaton that uses $s = s(m,n)$ bits. Then,*

$$s2^s = \Omega\left(\frac{m^n}{|Z_{m,n}(\mu)|}\right) \quad .$$

Proof. Suppose that there exists a stream automaton $\mathcal{A}$ that returns an α-approximation to $\mu(f)$. By Theorem 1, we can assume that the transition function of $\mathcal{A}$ is a linear mapping. The number of frequency vectors with L_∞ norm at most $\frac{m}{2}$ is $(m+1)^n$ (assume m is even to keep notation simple) and the number of configurations is $N = s \cdot 2^s \cdot |Q|$. Therefore, there exists a configuration C such that at least $\frac{(m+1)^n}{N}$ frequency vectors map to C. Let S denote the set of frequency vectors that map to C. Let Z' denote $Z_{\frac{m}{2},n}(\mu)$ and Z denote $Z_{m,n}(\mu)$.

From S, we construct a set T as follows. If $S \cap Z' \neq \phi$, then $T = S$. Otherwise, let g be any vector in $Z' - S$ and let T be defined as $T = S + g = \{f + g \mid f \in S\}$. Let $h = h' + g \in T$, where, $h' \in S$. Then, $\|h\|_\infty \leq \|h'\|_\infty + \|g\|_\infty \leq \frac{m}{2} + \frac{m}{2} = m$, that is, h is a feasible frequency vector for input parameters m and n. By linearity (or path independence), all vectors in T map to the same configuration, and hence, must return the same estimate $\hat{\mu}(g)$, for every $g \in T$. By construction, $T \cap Z \supset T \cap Z' \neq \phi$. Further, if $|Z| < \frac{(m+1)^n}{s2^s|Q_A|}$ then $T - Z \neq \phi$. The automaton returns the same estimate for vectors in $T - Z$ as it does for vectors in $T \cap Z$, implying that the estimate returned has arbitrary relative error. Hence, if the estimate returned has finite relative error, then,

$$|Z| \geq \frac{(m+1)^n}{s2^s|Q_A|}, \text{ or, } s2^s = \Omega\left(\frac{m^n}{|Z|}\right)$$

since, $|Q_A|$ is a constant. □

Lemma 6 can be used to easily infer space lower bounds for deterministic approximation over update streams. For example, consider the L_p norm, for any $p \geq 0$. The kernel of $Z_{m,n}(L_p)$ is the zero vector. By Lemma 6, the space s satisfies $s2^s = \Omega(m^n)$, or, $s = \Omega(n \log m)$. As another example, consider the entropy function, defined as $H(f) = \sum_{i:f_i \neq 0} \frac{|f_i|}{L_1} \log \frac{L_1}{|f_i|}$. By convention, $H(0) = 0$. The kernel $Z_{m,n}(H)$ consists of the zero vector and all vectors that have exactly one non-zero coordinate. Therefore, $|Z_{m,n}(H)| = 2mn + 1$. By Lemma 6, the space required is s bits, where, $s2^s = \Omega\left(\frac{m^n}{mn}\right)$, or, that $s = \Omega(n \log m)$.

5 Conclusions

We present stream automata, that is a computational model for online, space-bounded computation over update data streams. We show that in this model, a natural generalization of the transition function of the automaton is essentially a linear mapping. This property is used to derive space lower bounds for deterministic stream computations.

References

1. N. Alon, Y. Matias, and M. Szegedy. "The space complexity of approximating frequency moments". *J. Comp. Sys. and Sc.*, 58(1):137–147, 1998.

2. B. Babcock, S. Babu, M. Datar, R. Motwani, and J. Widom. "Models and Issues in Data Stream Systems". In *Proc. ACM PODS*, 2002.
3. L. Bhuvanagiri, S. Ganguly, D. Kesh, and C. Saha. "Simpler algorithm for estimating frequency moments of data streams". In *Proc. ACM SODA*, 2006.
4. Lakshminath Bhuvanagiri and S. Ganguly. "Estimating Entropy over Data Streams". In *Proc. ESA*, 2006.
5. P. Bose, E. Kranakis, P. Morin, and Y. Tang. "Bounds for Frequency Estimation of Packet Streams". In *SIROCCO*, pages 33–42, 2003.
6. M. Charikar, K. Chen, and M. Farach-Colton. "Finding frequent items in data streams". In *Proc. ICALP, 2002*, pages 693–703.
7. D. Coppersmith and R. Kumar. "An improved data stream algorithm for estimating frequency moments". In *Proc. ACM SODA*, 2004.
8. G. Cormode and S. Muthukrishnan. "An Improved Data Stream Summary: The Count-Min Sketch and its Applications". *J. Algorithms*, 55(1):58–75, April 2005.
9. G. Cormode and S. Muthukrishnan. "What's hot and what's not: tracking most frequent items dynamically". *ACM Trans. Database Syst.*, 30(1):249–278, 2005.
10. E. D. Demaine, A. López-Ortiz, and J. I Munro. "Frequency estimation of internet packet streams with limited space". In *Proc. ESA*, 2002.
11. C. Demetrescu, I. Finocchi, and A. Ribichini. "Trading off space for passes in graph streaming problems". In *Proc. ACM SODA*, 2006.
12. J. Feigenbaum, S. Kannan, A. McGregor, S. Suri, and J. Zhang. "On Graph Problems in a Semi-streaming Model". In *Proc. ICALP*, pages 531–543, 2004.
13. J. Feigenbaum, S. Kannan, A. McGregor, S. Suri, and J. Zhang. Graph distances in the streaming model: the value of space. In *Proc. ACM SODA*, 2005.
14. Gereon Frahling, Piotr Indyk, and Christian Sohler. Sampling in dynamic data streams and applications. In *SCG '05: Proceedings of the twenty-first annual symposium on Computational geometry*, pages 142–149, 2005.
15. S. Ganguly. "Counting distinct items over update streams". In *ISAAC*, 2005.
16. S. Ganguly and Majumder A. "CR-precis: A Deterministic Summary Structure for Update Streams". Manuscript, July, 2006.
17. S. Ganguly and A. Majumder. "Deterministic K-set Structure". In *Proc. ACM PODS*, 2006.
18. A. Gilbert, S. Guha, P. Indyk, Y. Kotidis, S. Muthukrishnan, and M. Strauss. "Fast Small-space Algorithms for Approximate Histogram Maintenance". In *Proc. ACM STOC*, 2002.
19. A. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. Strauss. "How to Summarize the Universe: Dynamic Maintenance of Quantiles". In *Proc. VLDB*, 2002.
20. A. C. Gilbert, Y. Kotidis, S. Muthukrishnan, and M. J. Strauss. "Surfing Wavelets on Streams: One-pass Summaries for Approximate Aggregate Queries". In *Proc. VLDB*, September 2001.
21. M. Greenwald and S. Khanna. "Space-efficient online computation of quantile summaries". In *SIGMOD*, 2001.
22. M. Grohe, A. Hernich, and N. Schweikardt. "Randomized Computations on Large Data Sets: Tight Lower Bounds". In *Proc. ACM PODS*, pages 243–252, 2006.
23. M. Grohe, C. Koch, and N. Schweikardt. "The Complexity of Querying External Memory and Streaming Data". In *Symp. on Fundamentals of Computation Theory, 2005. Vol. 3623 LNCS Springer-Verlag*, pages 1–16.
24. M. Grohe, C. Koch, and N. Schweikardt. "Tight Lower Bounds for Query Processing on Streaming and External Memory Data". In *Proc. ICALP*, pages 1076–1088, 2005.

25. S. Guha, P. Indyk, S. Muthukrishnan, and M. Strauss. "Histogramming Data Streams with Fast Per-Item Processing". In *Proc. ICALP*, pages 681–692, 2002.
26. J. Hershberger and S. Suri. "Adaptive sampling for geometric problems over data streams". In *Proc. ACM PODS*, pages 252–262, 2004.
27. P. Indyk and D. Woodruff. "Optimal Approximations of the Frequency Moments". In *Proc. ACM STOC*, 2005.
28. R.M. Karp, S. Shenker, and C.H. Papadimitriou. "A Simple Algorithm for Finding Frequent Elements in Streams and Bags". *ACM TODS*, 28(1):51–55, 2003.
29. J. Misra and Gries. D. "Finding repeated elements". *Sci. Comput. Programm.*, 2:143–152, 1982.