

Derandomization of Probabilistic Auxiliary Pushdown Automata Classes*

H. Venkateswaran
venkat@cc.gatech.edu
School of Computer Science
Georgia Institute of Technology, Atlanta, Georgia 30332-0765

March 19, 2009

1 Introduction

In this paper, we extend Nisan's breakthrough derandomization result that $BP_{HL} \subseteq SC^2$ [11] to bounded error probabilistic complexity classes based on auxiliary pushdown automata. (Here, BP_{HL} is the class of languages accepted by logarithmic space, polynomial time two-sided bounded-error probabilistic Turing machines and SC^2 is the class of languages accepted by logarithmic squared space, polynomial time deterministic Turing machines.) In particular, we show that any logarithmic space, polynomial time two-sided bounded-error probabilistic auxiliary pushdown automaton (the corresponding complexity class is denoted by $BP_H LOGCFL$) can be simulated by an SC^2 machine. This derandomization result improves a classical result by Cook [7] that $LOGDCFL \subseteq SC^2$ since $LOGDCFL$ is contained in $BP_H LOGCFL$. We also present a simple circuit-based proof that $BP_H LOGCFL$ is in NC^2 .

An auxiliary pushdown automaton (AuxPDA) is a multi-tape Turing machine with a two-way read-only input tape, a pushdown tape, and one or more work tapes. See [6] for a formal definition. Space on an AuxPDA is the space used on the work tapes without counting the space on the pushdown tape. It is well known [18, 19, 14] that nondeterministic AuxPDAs (NAuxPDA) that are simultaneously $O(\log n)$ space bounded and polynomial time bounded accept exactly the class of languages log-space reducible to context-free languages and such deterministic AuxPDAs (DAuxPDA) accept exactly the class of languages log-space reducible to deterministic context-free languages. Just as NAuxPDAs are a natural generalization of nondeterministic space machines, the probabilistic classes considered in this paper are a natural generalization of the much studied logarithmic space bounded probabilistic classes. Macarie and Ogi-hara [8] considered unbounded error classes based on, among other pushdown models, auxiliary pushdown automata. Our focus in this paper is on bounded error classes based on auxiliary pushdown automata.

For our proof, we generalize many of the key features of the proofs in Nisan's work on derandomization of space bounded computations [10, 11]. The generalizations of the pseudorandom generator (section 5.2) and the hash mixing lemma (section 3.4) may themselves be of independent interest. Also of independent interest are the different types of matrix products introduced in this paper (section 5.1.2) to capture the computations of probabilistic AuxPDAs analogous to matrix powering that captures the computations of probabilistic space bounded machines.

Nisan's derandomization result implicitly uses the fact that a directed path in a directed graph can be uniquely partitioned into two equal length sub-paths to recursively verify the computations of a probabilistic log space machine. In the case of NAuxPDAs Vinay [20] and Neidermeier and Rossmannith [9], in proving a parsimonious simulation of space and time-bounded NAuxPDA classes by circuits, have shown the existence

*This is a revised and corrected version of a preliminary version of a paper that appeared in the Proceedings of the twenty-first IEEE Computational Complexity Conference, Prague, 2006.

of unique balanced partitions of certain kinds of computation paths (called realizable computations, in particular accepting computations) of surface configurations of NAuxPDAs. In this paper, we use the partition proposed by Neidermeier and Rossmanith [9]. Such a balanced partition of a path gives rise to three sub-paths. This necessitates the generalization of the pseudorandom generator used by Nisan [10, 11]. In the case of probabilistic space-bounded machines considered by Nisan [11], the partition of a computation path into two sub-paths makes it possible to express the computations of such a machine in terms of the product of two appropriate matrices which is used in the correctness proof of the derandomization result. But, in the case of NAuxPDAs the sub-paths of a computation path are of two different types, standard paths and paths with gaps, which gives rise to different types of matrix products as defined in section 5.1.2. Our derandomization result also critically uses the classical result by Cook [7] showing that LOGDCFL is in SC^2 .

The power of Nisan's derandomization result was demonstrated by Sivakumar [17] who showed that derandomizations of a number of well known randomized algorithms directly follow from Nisan's result. We believe that our generalization will make it possible to obtain derandomizations of a more general class of randomized algorithms. Building on Nisan's derandomization result [11], Saks and Zhou [16] show that $BP_{HL} \subseteq DSPACE(\log^{3/2}n)$. One of the questions raised by our work is whether such a result can be shown for $BP_H LOGCFL$. (Note that the class $BP_H LOGCFL$ can be shown to be in $DSPACE(\log^2 n)$ as in the case of BP_{HL} .) We identify a number of such questions in section 6.1.

2 The Model

2.1 Auxiliary pushdown automata

We will assume that an AuxPDA accepts with its stack empty and halts on all computations. For a NAuxPDA we will further assume that: (a) every move has exactly two immediate successors; (b) there is a unique accepting configuration (assuming an empty pushdown, an empty work-tape, a fixed position for all tape heads, and a unique final accepting state); (c) every computation path in the NAuxPDA is a simple path (achieved by including a time component with each configuration); (d) the NAuxPDA pushes or pops a symbol in each move; and (e) all computation paths have the same length.

Realizable Pairs of Surface Configurations: Define a *surface configuration* of an AuxPDA on an input w to be: the state, contents and head positions of the work tape, the head position of the input tape, the topmost symbol of the stack, stack height, and a time component. The time component in the initial surface configuration is 0 and the time component in the final surface configuration is $t(n)$, the running time of the machine. If the machine moves from a surface configuration A whose time component is t_A to a surface configuration B , then the time component t_B included in B is $t_A + 1$. Note that for a space $S(n)$ bounded AuxPDA, its surface configurations take only $O(S(n))$ space. In the rest of the paper, we will refer to surface configurations as configurations.

For an input w , a pair of configurations (A, B) is *realizable* if the AuxPDA can move from A to B ending with its stack at the same height as in A , and without popping its stack below its level in B at any point in this computation. An AuxPDA M accepts an input w iff there is a realizable pair (P_0, Q_a) , where P_0 is the initial configuration and Q_a is the unique accepting configuration.

2.2 Probabilistic AuxPDA

A NAuxPDA M is said to be a probabilistic AUXPDA (PAuxPDA) if its each move is considered as a random move with each possible next move assumed to have equal probability. The probability of a computation path is the product of the probabilities associated with the random moves along the path. Given an input w , the probability of acceptance by M , denoted by $p_M(w)$, is the sum over all accepting computation paths P of the the probability associated with P .

We concern ourselves with PAuxPDAs that are simultaneously $O(\log n)$ space bounded and polynomial time bounded. Using the convention in [15], we will say that these are machines that halt absolutely. (Our results can be generalized in a natural way to PAuxPDAs that are simultaneously $O(S(n)) \geq \log n$ space bounded and $2^{O(S(n))}$ time bounded, where $S(n)$ is space-constructible.)

2.2.1 Verifier characterization

Analogous to the case of space-bounded probabilistic machines that halt absolutely, PAuxPDAs that halt absolutely can be characterized using a DAuxPDA verifier that has read-once access to a polynomial number of random bits.

Let M be a $p(n)$ time bounded PAuxPDA, where $p(n)$ is a polynomial. Let $t = p(n)$.

The verifier is a logarithmic space polynomial time DAuxPDA V_M that uses an auxiliary read-only one-way tape containing a string of random bits r .

Any configuration C of M is of the form $\langle C_V, i \rangle$ where C_V is a configuration of V_M and i the head position on the auxiliary tape containing the random bits. The initial configuration of M on an input w is $\langle P_0, 1 \rangle$, where P_0 is the initial configuration of V_M on input w and the unique accepting configuration of M is $\langle Q_a, t \rangle$, where Q_a with an empty stack is the unique accepting configuration of V_M .

On input w , V_M simulates M as follows:

Step 0: Write t on a work tape, CLOCK.

Step 1: Push a bottom marker Z onto the stack. Simulate M on x starting from the initial configuration P_0 using the bits of r as the guide. For every step of M decrement CLOCK. If the configuration reached is Q_a and the stack has only bottom marker Z then accept. If the CLOCK becomes 0 then reject if either Q_a is not reached or the stack top is not Z . If M pushes a symbol, V_M also pushes the same symbol. If M pops a symbol, V_M pops as well; if the symbol popped is Z then M rejects.

In this paper, we will assume such a verifier formulation of probabilistic AuxPDA.

2.3 Randomized complexity classes based on PauxPDA

We define bounded-error complexity classes based on the probabilistic AuxPDA model using the notation proposed in Saks [15] as follows:

The class $BP_HLOGCFL$ is the class of all languages L such that there exists a probabilistic AuxPDA M with space bound $O(\log n)$ and time bound polynomial such that if $w \in L$, $Pr[M \text{ accepts } w] \geq \frac{2}{3}$ and if $w \notin L$, $Pr[M \text{ accepts } w] \leq \frac{1}{3}$. The class $R_HLOGCFL$ is the one-sided version of the class $BP_HLOGCFL$. (The error probabilities can be made small by a standard technique of performing independent trials and taking the majority outcome.)

The inclusions below follow by definition:

Proposition 1

$$R_HL \subseteq R_HLOGCFL \subseteq BP_HLOGCFL \\ BP_HL \subseteq BP_HLOGCFL$$

3 Preliminaries

In this section we will present: (a) two lemmas from [9] that gives a balanced partition of a realizable computation of an NAuxPDA (b) circuit characterizations from [9] based on these lemmas; (c) verification of the realizability of a pair of configurations (without and with gap) using these circuits and a bit string; and (d) a generalization of a property (referred to as hash mixing lemma) of universal family of hash functions proved and used by Nisan in [10, 11].

3.1 Unique balanced partition of a computation path

Let M be an NAuxPDA and w be an input to M .

Definition 2 [9] *Let P be a computation path from a configuration A to a configuration B such that the stack height at B is the same as at A , and M does not pop its stack below its level at A along P . Then, we say that the pair (A, B) is realizable along P .*

Definition 3 [9] *Let A, B, C, D be four configurations such that: (a) P_1 is a computation path from A to C in which the stack height does not go below that at A ; (b) P_2 is a computation path from D to B in which the stack height does not go below that at B ; (c) the stack height at A is the same as that at B and the stack height at C is the same as that at D . Let P be the path obtained by joining P_2 to P_1 . Then, the pair (A, B) with gap (C, D) , referred to as a pair-with-gap and denoted as $(A, (C, D), B)$, is said to be realizable along P .*

Lemma 4 below gives a balanced partition of a computation path into three sub-paths one of which is a path with a gap.

Lemma 4 [9] *Let P be a computation path from configuration A to configuration B along which the pair (A, B) is realizable. Then, there exist unique configurations C, D, E, F, G along P such that: (a) C is a push configuration that pushes a symbol a onto the stack and goes to configuration E , (b) F is the first pop configuration after E such that F pops the same symbol a and goes to G , (c) D is either B or a pop configuration, (d) $|(E, F)|, |(G, D)| \leq |(A, B)|/2 < |(C, D)|$, (e) $|(C, D)| = |(E, F)| + |(G, D)| + 2$, and (f) (E, F) is realizable along the sub-path of P from E to F , (G, D) is realizable along the sub-path of P from G to D , and the pair-with-gap $(A, (C, D), B)$ is realizable along the path obtained by joining the sub-path of P from A to C with the sub-path of P from D to B .*

Lemma 5 below gives a balanced partition of a path with gap into three sub-paths two of which are paths with gaps.

Lemma 5 [9] *Let $(A, (C, D), B)$ be a pair-with-gap that is realizable along a computation path P with $|(C, D)| > \frac{|(A, B)|}{2}$. Then, there exist unique configurations C', D', E', F', G' such that:*

(a) C' is a push configuration that pushes a symbol a onto the stack and goes to E' , (b) F' is a pop configuration that pops the same symbol a and goes to G' , (c) the stack heights of C', G' , and D' are the same, (d) $|(C', D')| = |(E', F')| + |(G', D')| + 2$, (e) $|(C', D')| - |(C, D)| > (|(A, B)| - |(C, D)|)/2$, (f) either $(|(C', G')| > |(A, B)|/2 \text{ and } |(G', D')| < |(A, B)|/2)$ or $(|(C', G')| < |(A, B)|/2 \text{ and } |(G', D')| > |(A, B)|/2)$, and (g) $(A, (C', D'), B)$ is realizable along the path obtained by joining the sub-path of P from A to C' with the sub-path of P from D' to B .

Furthermore, one of the following two (mutually exclusive) cases is true:

If $(|(C', G')| > |(A, B)|/2 \text{ and } |(G', D')| < |(A, B)|/2)$ the gap (C, D) occurs along the segment from E' to F' . In this case, $|(E', F')| - |(C, D)| \leq (|(A, B)| - |(C, D)|)/2$. Moreover, the pair-with-gap $(E', (C, D), F')$ is realizable along the path obtained by joining the sub-path of P from E' to C with the sub-path of P from D to F' and the pair (G', D') is realizable along the sub-path of P from G' to D' .

If $(|(C', G')| < |(A, B)|/2 \text{ and } |(G', D')| > |(A, B)|/2)$ the gap (C, D) occurs along the segment from G' to D' . In this case, $|(G', D')| - |(C, D)| \leq (|(A, B)| - |(C, D)|)/2$. Moreover, the pair (E', F') is realizable along the sub-path of P from E' to F' and the pair-with-gap $(G', (C, D), D')$ is realizable along the path obtained by joining the sub-path of P from G' to C with the sub-path of P from D to D' .

3.2 Circuits using the partition lemmas

Given a pair of configurations (A, B) of a BP_HL machine and a bit string r , Nisan's result uses r to check if a path from A to B exists. This verification implicitly uses the fact that the path can be divided into two equal length parts and that the partition is easy to compute (the partition occurs at the middle point). In the case of AuxPDAs, although a realizable path can be partitioned into three balanced sub-paths, these sub-paths may not be of equal length. More importantly, it is not clear how to obtain such partitions of a realizable path within the resource constraints. So, we make use of the parsimonious circuits constructed in [9] to check the realizability of a pair of configurations. These circuits are described below.

Let H_q be a Boolean circuit with q input variables.

Definition: A *parse-tree* T of the circuit H_q is defined inductively as follows: T includes the output gate; T includes both the immediate predecessors of any AND gate included in T ; and T includes exactly one immediate predecessor of any OR gate included in T .

Given a truth assignment to the q input variables of H_q , an *accepting parse-tree* is a parse-tree in which all the included circuit-input nodes have value one assigned to them.

Given M and w , for each pair of configurations (A, B) , a log-space uniform Boolean circuit $G_{(A,B)}$ and an input truth assignment is constructed in [9] such that the number of accepting parse-trees in $G_{(A,B)}$ is equal to the number of realizable paths from A to B . An accepting parse-tree in $G_{(A,B)}$ verifies that the pair (A, B) is realizable. A similar construction is given in [9] in the case of pairs-with-gaps as well.

Given M and w , we use the constructions in [9] to obtain circuits for the verifier V_M on input w as described below. Let t be the running time of M on input w . Let n be the number of configurations of M on w . Assume that $t = n$. Let $K = \log_2 t$.

3.2.1 Circuits for pairs without gaps

A recursive construction of the circuit $G_{(A,B)}$ for verifying that a pair of configurations (A, B) , with the stack height of A the same as that of B , is described below. We assume that $|(A, B)| > m$. Let 2^q be the smallest power of 2 such that $m2^q \geq |(A, B)|$.

The OR gates in the circuit are labeled as $\langle A, B, d, \text{OR} \rangle$, where A and B are configurations, and d is the depth of the gate. The AND gates in the circuit are labeled as $\langle C, E, F, G, D, d, \text{AND} \rangle$, where C, E, F, G and D are configurations, and d is the depth of the gate.

The output of the circuit is labeled $\langle A, B, q, \text{OR} \rangle$. An OR gate labeled as $\langle A, B, d, \text{OR} \rangle$ has as inputs AND gates labeled as $\langle C, E, F, G, D, d, \text{AND} \rangle$, where C, E, F, G , and D are configurations of V_M . The fan-in of an OR gate is t^5 .

Consider an AND gate $\langle C, E, F, G, D, d, \text{AND} \rangle$. If the configurations C, E, F, G, D do not satisfy the following conditions of the partition lemma then the AND gate has a single input with value 0:

- $|(E, F)| + |(G, D)| = |(C, D)| - 2$,
- $|(E, F)|, |(G, D)| \leq \frac{|(A, B)|}{2} < |(C, D)|$,
- C is a push configuration,
- F is a pop configuration, and
- D is either a pop configuration or B .

If the configurations C, E, F, G, D satisfy the above stated conditions of the partition lemma then the AND gate has the following four inputs:

- an OR gate labeled $\langle A, (C, D), B, d - 1, \text{OR} \rangle$, which is the output of the circuit $G_{(A, (C, D), B)}$,
- a non-constant circuit-input labeled $\langle C, E, F, G, \text{input} \rangle$,

- an OR gate labeled $\langle E, F, d - 1, \text{OR} \rangle$, which is the output of the circuit $G_{(E,F)}$ and
- an OR gate labeled $\langle G, D, d - 1, \text{OR} \rangle$ which is the output of the circuit $G_{(G,D)}$.

3.2.2 Circuits for pairs with gaps

A recursive construction of the circuit $G_{(A,(C,D),B)}$ for verifying that a pair-with-gap $(A, (C, D), B)$ is realizable is described below. We assume that the stack height of A is the same as that of B and the stack height of C is the same as that of D , and that $|(A, (C, D), B)| > m$. Let 2^q be the smallest power of 2 such that $m2^q \geq |(A, B)| - |(C, D)|$.

The output of the circuit is labeled $\langle A, B, C, D, q, \text{OR} \rangle$. The OR gates in the circuit are labeled as $\langle A, B, C, D, d, \text{OR} \rangle$, where A, B, C , and D are configurations, and d is the depth of the gate. The AND gates in the circuit are labeled as $\langle C', E', F', G', D', d, \text{AND} \rangle$, where C', E', F', G' and D' are configurations, and d is the depth of the gate.

An OR gate labeled as $\langle A, B, C, D, d, \text{OR} \rangle$ has as inputs AND gates labeled as $\langle C', E', F', G', D', d, \text{AND} \rangle$, where C', E', F', G' , and D' are configurations of V_M . The fan-in of an OR gate is t^5 .

Consider an AND gate $\langle C', E', F', G', D', d, \text{AND} \rangle$. If the configurations (C', E', F', G', D') do not satisfy the following conditions of the partition lemma then the AND gate has a single input with value 0.

- $|(C', D')| = |(E', F')| + |(G', D')| + 2$,
- $|(C', D')| - |(C, D)| > (|(A, B)| - |(C, D)|)/2$,
- C' is a push configuration,
- F' is a pop configuration, and
- D' is either a pop configuration or B .

If the configurations (C', E', F', G', D') satisfy the above conditions of the partition lemma then the inputs to this AND gate are determined as follows:

- If $(|(C', G')| > |(A, B)|/2$ and $|(G', D')| < |(A, B)|/2$) then the inputs are:
 - an OR gate labeled $\langle A, (C', D'), B, d - 1, \text{OR} \rangle$ which is the output of the circuit $G_{(A,(C',D'),B)}$,
 - a non-constant circuit-input labeled $\langle C', E', F', G', \text{input} \rangle$,
 - an OR gate labeled $\langle E', (C, D), F', d - 1, \text{OR} \rangle$ which is the output of the circuit $G_{(E',(C,D),F)}$, and
 - an OR gate labeled $\langle G', D', d - 1, \text{OR} \rangle$ which is the output of the circuit $G_{(G',D')}$.
- If $(|(C', G')| < |(A, B)|/2$ and $|(G', D')| > |(A, B)|/2$) then the inputs are:
 - an OR gate labeled $\langle A, (C', D'), B, d - 1, \text{OR} \rangle$ which is the output of the circuit $G_{(A,(C',D'),B)}$,
 - a non-constant circuit-input labeled $\langle C', E', F', G', \text{input} \rangle$,
 - an OR gate labeled $\langle E', F', d - 1, \text{OR} \rangle$ which is the output of the circuit $G_{(E',F')}$, and
 - an OR gate labeled $\langle G', (C, D), D', d - 1, \text{OR} \rangle$ which is the output of the circuit $G_{(G',(C,D),D')}$.

3.3 Verification of realizability using circuits

The realizability of a pair of configurations (A, B) of M is verified by a DAuxPDA V_C using a pseudo-random bit string r of appropriate length as described below.

If $|(A, B)| \leq m$, $|r| = m$, the verifier V_C checks the realizability of the pair (A, B) directly using the bit string r .

Suppose $|(A, B)| > m$. Let $G_{(A,B)}$ be the circuit corresponding to the pair (A, B) of configurations of V . Let 2^q be the smallest power of 2 such that $m2^q \geq |(A, B)|$. Let r be a bit string of length $\geq \frac{(4^{s+1} - 1)m}{3}$ for $q \leq s \leq K$. Note that $r = x \circ r_0 \circ r_1 \circ r_2 \circ r_3$, where $|x| = m$ and $|r_i| = \frac{(4^s - 1)m}{3}$ for $i = 1, 2, 3$.

The output gate of $G_{(A,B)}$ is an OR gate labeled as $\langle A, B, q, \text{OR} \rangle$ whose inputs are AND gates labeled as $\langle C, E, F, G, D, d, \text{AND} \rangle$. The verifier V_C evaluates the circuit $G_{(A,B)}$ using the recursive procedure described below. Given an OR gate $\langle A, B, d, \text{OR} \rangle$, this procedure accepts if and only if the pair (A, B) is realizable.

Step 1: Use the first $5K$ bits in the prefix x of r to choose an input AND gate $\langle C, E, F, G, D \rangle$ of the gate $\langle A, B, d, \text{OR} \rangle$. If the chosen AND gate has one input and it has value 0 then REJECT.

Step 2: Push the configurations C, E, F, G and D on to the stack and recursively verify that the pair-with-gap $(A, (C, D), B)$ is realizable using r_0 . If not successful then REJECT.

Step 3: Pop the next five configurations, say C, E, F, G and D , and verify that (a) M moves from C to E pushing a symbol a using the first bit of r_1 and (b) M moves from F to G popping the same symbol a using the second bit of r_1 . If not successful then REJECT.

Step 4: Push the configurations G, D on to the stack. Use r_2 to recursively verify that the pair of configurations (E, F) is realizable. If not successful then REJECT.

Step 5: Pop the next two configurations G, D , and recursively verify that (G, D) is a realizable pair using r_3 . If not successful then REJECT.

Step 6: ACCEPT.

The DAuxPDA V_C uses logarithmic space and polynomial time.

The verification of the realizability of a pair-with-gap $(A, (C, D), B)$, where A, B, C and D are configurations of M on w , is done in a similar way by a DAuxPDA verifier V_C using circuits and a pseudo-random string r of appropriate length.

If $|(A, (C, D), B)| \leq m$, $|r| = m$, the verifier V_C checks the realizability of the pair-with-gap $(A, (C, D), B)$ directly using r .

Suppose $|(A, (C, D), B)| > m$. Let $G_{(A,(C,D),B)}$ be the circuit corresponding to the 4-tuple $(A, (C, D), B)$. Let 2^q be the smallest power of 2 such that $m2^q \geq |(A, B)| - |(C, D)|$. The DAuxPDA V_C uses this circuit and a bit string r of length $\geq \frac{(4^{s+1} - 1)m}{3}$ for $q \leq s \leq K$ to verify that the pair-with-gap $(A, (C, D), B)$ is realizable. Note that $r = x \circ r_0 \circ r_1 \circ r_2 \circ r_3$, where $|x| = m$ and $|r_i| = \frac{(4^s - 1)m}{3}$ for $i = 1, 2, 3$. As in the case of pairs without gaps, the DAuxPDA uses logarithmic space and polynomial time which by the result of Cook [7] can be simulated by an SC^2 machine.

The output gate of $G_{(A,(C,D),B)}$ is an OR gate labeled as $\langle A, B, C, D, q, \text{OR} \rangle$ whose inputs are AND gates labeled as $\langle C', E', F', G', D', d, \text{AND} \rangle$. Given an OR gate $\langle A, B, C, D, d, \text{OR} \rangle$, this procedure accepts if and only if the pair-with-gap $(A, (C, D), B)$ is realizable.

Step 1: Use the first $5K$ bits in the prefix x of r to choose an input AND gate $\langle C', E', F', G', D' \rangle$ of the gate $\langle A, B, C, D, d, \text{OR} \rangle$. If the chosen AND gate has one input and it has value 0 then REJECT.

Step 2: Push the configurations C, D and then the configurations C', E', F', G', D' and D' on to the stack and recursively verify that the pair-with-gap $(A, (C', D'), B)$ is realizable using r_0 . If not successful then REJECT.

Step 3: Pop the next five configurations, say C', E', F', G', D' and verify that (a) M moves from C' to E' pushing a symbol a using the first bit of r_1 and (b) M moves from F' to G' popping the same symbol a using the second bit of r_1 . If not successful then REJECT.

Step 4: If $(|(E', F')| < \frac{|(A, B)|}{2} \text{ and } |(G', D')| > \frac{|(A, B)|}{2})$ then go to Step 5' (check for gap in the second segment).

Step 5: Pop the next two configurations, say C, D and push the two configurations G', D' . Use r_2 to recursively verify that the pair-with-gap $(E', (C, D), F')$ is realizable. If not successful then REJECT.

Step 6: Pop the next two configurations G', D' and recursively verify that (G', D') is a realizable pair using r_3 . If not successful then REJECT.

Step 7: ACCEPT.

Step 5': Push the two configurations G', D' . Use r_2 to recursively verify that (E', F') is a realizable pair. If not successful then REJECT.

Step 6': Pop the next four configurations, say G', D', C, D and recursively verify that the pair with gap $(G', (C, D), D')$ is realizable using r_3 . If not successful then REJECT.

Step 7': ACCEPT.

The DAuxPDA V_c uses logarithmic space and polynomial time.

3.4 A generalized hash mixing lemma

Let \mathcal{H} be a set of functions $h : \{0, 1\}^n \rightarrow \{0, 1\}^m$.

Definition 6 ([5]) \mathcal{H} is called a universal family of hash functions if for any $x_1 \neq x_2 \in \{0, 1\}^n$ and $y_1, y_2 \in \{0, 1\}^m$ we have that

$$Pr_{h \in \mathcal{H}}[h(x_1) = y_1 \text{ and } h(x_2) = y_2] = 2^{-2m}.$$

Lemma 7 below (referred to as the hash mixing lemma) was proved by Nisan in [10] and used in [10, 11].

Lemma 7 [10, 11] Let \mathcal{H} be a family of universal hash functions $h : \{0, 1\}^n \rightarrow \{0, 1\}^m$. Let $A, B \subset \{0, 1\}^m$. Let $\theta > 0$ be given. Then,

$$Pr_{h \in \mathcal{H}}[|Pr_{x \in \{0, 1\}^m}[x \in A, h(x) \in B] - \rho(A)\rho(B)| \geq \theta] \leq \frac{1}{2^m \theta^2}.$$

We need the following extension of the hash-mixing lemma:

Lemma 8 Let \mathcal{H} be a family of universal hash functions $h : \{0, 1\}^n \rightarrow \{0, 1\}^m$. Let $k \geq 0$. For $0 \leq i \leq k$, let $A_i \subset \{0, 1\}^m$. Let $\theta > 0$ be given. Then,

$$Pr_{h_1, \dots, h_k \in \mathcal{H}}[|Pr_{x \in \{0, 1\}^m}[x \in A_0, h_1(x) \in A_1, \dots, h_k(x) \in A_k] - \rho(A_0)\rho(A_1)\dots\rho(A_k)| \geq \theta] \leq \frac{1}{2^m \theta^2}.$$

Proof: The proof of Lemma 8 is obtained by a straightforward adaptation of the proof of Lemma 7. For $x \in \{0, 1\}^m$ and $h_1, \dots, h_k \in \mathcal{H}$, define a random variable Y_{x, h_1, \dots, h_k} as follows:

$$Y_{x, h_1, \dots, h_k} = 1 \text{ if } x \in A_0, h_i(x) \in A_i, (\text{for } 0 \leq i \leq k) \text{ and } 0 \text{ otherwise.}$$

Also, let $I_A(x) = 1$ be the indicator function (that is, $I_A(x) = 1$ if $x \in A$ and 0 otherwise). Then,

$$E_{h_1, \dots, h_k}[Y_{x, h_1, \dots, h_k}] = I_{A_0}(x) \rho(A_1) \dots \rho(A_k),$$

and, for $x_1, x_2 \in \{0, 1\}^m$ with $x_1 \neq x_2$,

$$E_{h_1, \dots, h_k}[Y_{x_1, h_1, \dots, h_k} Y_{x_2, h_1, \dots, h_k}] = I_{A_0}(x) \rho^2(A_1) \dots \rho^2(A_k).$$

Define a random variable Z_{h_1, \dots, h_k} as follows:

$$Z_{h_1, \dots, h_k} = \sum_{x \in \{0, 1\}^m} Y_{x, h_1, \dots, h_k}.$$

We are interested in computing

$$\text{Var}_{h_1, \dots, h_k}[Z_{h_1, \dots, h_k}] = E_{h_1, \dots, h_k}[Z_{h_1, \dots, h_k}^2] - E_{h_1, \dots, h_k}^2[Z_{h_1, \dots, h_k}],$$

where $\text{Var}_{h_1, \dots, h_k}[Z_{h_1, \dots, h_k}]$ is the variance of Z_{h_1, \dots, h_k} .

Now,

$$E[Z_{h_1, \dots, h_k}] = |A_0| \rho(A_1) \dots \rho(A_k),$$

and

$$E[Z_{h_1, \dots, h_k}^2] = |A_0| \rho(A_1) \dots \rho(A_k) + |A_0| (|A_0| - 1) \rho^2(A_1) \dots \rho^2(A_k).$$

Therefore,

$$\text{Var}_{h_1, \dots, h_k}[Z_{h_1, \dots, h_k}] = |A_0| \rho(A_1) \dots \rho(A_k) (1 - \rho(A_1) \dots \rho(A_k)).$$

By Chebyshev's inequality,

$$\Pr_{h_1, \dots, h_k}[|Z_{h_1, \dots, h_k} - |A_0| \rho(A_1) \dots \rho(A_k)| \geq 2^m \theta] \leq \frac{\text{Var}_{h_1, \dots, h_k}[Z_{h_1, \dots, h_k}]}{2^{2m} \theta^2}.$$

Since,

$$\frac{\text{Var}_{h_1, \dots, h_k}[Z_{h_1, \dots, h_k}]}{2^{2m} \theta^2} = \frac{\rho(A_0) \dots \rho(A_k) (1 - \rho(A_1) \dots \rho(A_k))}{2^m \theta^2} \leq \frac{1}{2^m \theta^2},$$

we have,

$$\Pr_{h_1, \dots, h_k}[|Z_{h_1, \dots, h_k} - |A_0| \rho(A_1) \dots \rho(A_k)| \geq 2^m \theta] \leq \frac{1}{2^m \theta^2}.$$

That is,

$$\Pr_{h_1, \dots, h_k}\left[\frac{|Z_{h_1, \dots, h_k} - |A_0| \rho(A_1) \dots \rho(A_k)|}{2^m} \geq \theta\right] \leq \frac{1}{2^m \theta^2}.$$

The lemma follows by noting that, by definition of Z_{h_1, \dots, h_k} and Y_{x, h_1, \dots, h_k} ,

$$\frac{|Z_{h_1, \dots, h_k} - |A_0| \rho(A_1) \dots \rho(A_k)|}{2^m} = |\Pr_{x \in \{0, 1\}^m}[x \in A_0, h_1(x) \in A_1, \dots, h_k(x) \in A_k] - \rho(A_0) \dots \rho(A_k)|.$$

4 The Main Theorem

Theorem 9 $BP_H \text{LOGCFL} \subseteq SC^2$.

5 Proof of the main theorem

The structure of our proof of theorem 9 is similar to Nisan's proof that $BP_{HL} \subseteq SC^2$ [11].

5.1 Norms of probability matrices and their properties

5.1.1 Norms of probability matrices

The partition of a path into sub-paths of two different types (standard paths and paths with gaps) gives rise to two types of matrices: *standard* matrices and *gap* matrices. The entry $\mathcal{AS}[A, B]$ of a standard matrix, indexed by a pair of configurations with same stack height, is the probability that the pair is realizable using a random string (either pure or pseudorandom string). The entry $\mathcal{AG}[A, (C, D), B]$ of a gap matrix, indexed by a quadruple of configurations A, B, C, D , where A, B have the same stack height and C, D have the same stack height, is the probability that the pair-with-gap $(A, (C, D), B)$ is realizable using a random string (either pure or pseudorandom string).

Definition 10 *The norm of a standard matrix \mathcal{AS} is defined as follows:*

$$\|\mathcal{AS}\| = \max_A \sum_B \mathcal{AS}[A, B].$$

Now, we define the norm of a gap matrix. Let \mathcal{APGU} denote a probability matrix whose entry $\mathcal{APGU}[A, B]$, where A, B is a pair of configurations with the stack height of A at most that of B , is the probability that configuration B is reachable from configuration A with the stack height not going below that at A . Let \mathcal{APGL} denote a probability matrix whose entry $\mathcal{APGL}[A, B]$, where A, B is a pair of configurations with the stack height of B at most that of A , is the probability that configuration B is reachable from configuration A with the stack height not going below that at B .

We will refer to the \mathcal{APGU} and \mathcal{APGL} matrices (whose entries are probabilities for reachability) as *reachable-standard* matrices to distinguish them from standard matrices (whose entries are probabilities for realizability).

The norms of the reachable-standard matrices \mathcal{APGL} and \mathcal{APGU} are as follows:

$$\|\mathcal{APGL}\| = \max_A \sum_B \mathcal{APGL}[A, B].$$

$$\|\mathcal{APGU}\| = \max_A \sum_B \mathcal{APGU}[A, B].$$

The entries of the gap matrix \mathcal{AG} , indexed by four configurations A, B, C, D such that the stack height of A and B are the same and the stack height of C and D are the same, can be expressed in terms of the the entries of the two reachable-standard matrices \mathcal{APGL} and \mathcal{APGU} as follows:

$$\mathcal{AG}[A, (C, D), B] = \mathcal{APGU}[A, C] \mathcal{APGL}[D, B].$$

Hence, the gap matrix \mathcal{AG} can be viewed as the tensor product of the two reachable-standard matrices:

$$\mathcal{AG} = \mathcal{APGU} \otimes \mathcal{APGL}.$$

Now, for probability matrices \mathcal{A} , and \mathcal{B} , $\|\mathcal{A} \otimes \mathcal{B}\| \leq \|\mathcal{A}\| \|\mathcal{B}\|$. Therefore, we have the following definition for the norm of the gap matrix \mathcal{AG} :

Definition 11 *The norm of a gap matrix \mathcal{AG} is defined as follows:*

$$\|\mathcal{AG}\| = \max_A \sum_C \max_D \sum_B \mathcal{AG}[A, (C, D), B].$$

5.1.2 Products of standard and gap matrices and their norms

The correctness argument for the algorithms that verify the realizability of a pair of configurations involves (see Lemma 25) five different types of products using standard-matrices and gap-matrices. In this section, we define these product types and show that, for each of these products, the norm of the product of two matrices is at most the product of the norms of the matrices.

(The proof of Lemma 25 also uses the facts that the norm of the sum of two gap matrices (and the sum of two standard matrices) is at most the sum of the norms of the matrices.)

Product of a standard matrix with a standard matrix - extension type: Let \mathcal{AS} and \mathcal{BS} be standard matrices. The standard product \otimes_1 defined below (representing the standard path obtained by extending a standard path by a standard path) results in a standard matrix \mathcal{CS} with entries:

$$(\mathcal{AS} \otimes_1 \mathcal{BS})[A, B] = \sum_G \mathcal{AS}[A, G].\mathcal{BS}[G, B].$$

Since $(\mathcal{AS} \otimes_1 \mathcal{BS})[A, B]$ is a standard-matrix, its norm is:

$$\begin{aligned} \|\mathcal{AS} \otimes_1 \mathcal{BS}\| &= \max_A \sum_B \sum_G \mathcal{AS}[A, G].\mathcal{BS}[G, B] \\ &\leq (\max_A \sum_G \mathcal{AS}[A, G])(\max_G \sum_B \mathcal{BS}[G, B]) \\ &= \|\mathcal{AS}\| \|\mathcal{BS}\|. \end{aligned}$$

Product of a gap matrix with a standard matrix - gap-filling type: Let \mathcal{AG} be a gap matrix and \mathcal{BS} be a standard matrix. The product \otimes_2 defined below (representing the standard path obtained by filling the gap in a gap-path by a standard path) results in a standard matrix \mathcal{CS} with entries:

$$(\mathcal{AG} \otimes_2 \mathcal{BS})[A, B] = \sum_{C,D} \mathcal{AG}[A, (C, D), B].\mathcal{BS}[C, D]$$

Since $(\mathcal{AG} \otimes_2 \mathcal{BS})[A, B]$ is a standard matrix, its norm is:

$$\begin{aligned} \|\mathcal{AG} \otimes_2 \mathcal{BS}\| &= \max_A \sum_B (\mathcal{AG} \otimes_2 \mathcal{BS})[A, B] \\ &= \max_A \sum_B \sum_{C,D} \mathcal{AG}[A, (C, D), B].\mathcal{BS}[C, D] \\ &\leq (\max_A \sum_C \max_D \sum_B \mathcal{AG}[A, (C, D), B])(\max_C \sum_D \mathcal{BS}[C, D]) \\ &= \|\mathcal{AG}\| \|\mathcal{BS}\|. \end{aligned}$$

Product of a gap matrix with a standard matrix - extension type: Let \mathcal{AG} be a gap matrix and \mathcal{BS} be a standard matrix. The product \otimes_3 defined below (representing the gap-path obtained by extending a gap-path by a standard path) results in a gap matrix \mathcal{CG} with entries:

$$(\mathcal{AG} \otimes_3 \mathcal{BS})[A, (C, D), B] = \sum_G \mathcal{AG}[A, (C, D), G].\mathcal{BS}[G, B]$$

Since $(\mathcal{AG} \otimes_3 \mathcal{BS})[A, (C, D), B]$ is a gap-matrix, its norm is:

$$\begin{aligned}
\|\mathcal{AG} \otimes_3 \mathcal{BS}\| &= \max_A \sum_C \max_D \sum_B \sum_G \mathcal{AG}[A, (C, D), G] \cdot \mathcal{BS}[G, B] \\
&\leq (\max_A \sum_C \max_D \sum_G \mathcal{AG}[A, (C, D), G]) (\max_G \sum_B \mathcal{BS}[G, B]) \\
&= \|\mathcal{AG}\| \|\mathcal{BS}\|.
\end{aligned}$$

Product of a standard matrix with a gap matrix - extension type: Let \mathcal{AS} be a standard matrix and \mathcal{BG} be a gap matrix. The product \otimes_4 defined below (representing the gap-path obtained by extending a standard path by a gap path) results in a gap matrix \mathcal{CG} with entries:

$$(\mathcal{AS} \otimes_4 \mathcal{BG})[A, (C, D), B] = \sum_G \mathcal{AS}[A, G] \cdot \mathcal{BG}[G, (C, D), B]$$

Since $(\mathcal{AS} \otimes_4 \mathcal{BG})[A, (C, D), B]$ is a gap-matrix, its norm is:

$$\begin{aligned}
\|\mathcal{AS} \otimes_4 \mathcal{BG}\| &= \max_A \sum_C \max_D \sum_B \sum_G \mathcal{AS}[A, G] \cdot \mathcal{BG}[G, (C, D), B] \\
&\leq (\max_A \sum_G \mathcal{AS}[A, G]) (\max_G \sum_C \max_D \sum_B \mathcal{BS}[G, (C, D), B]) \\
&= \|\mathcal{AS}\| \|\mathcal{BG}\|.
\end{aligned}$$

Product of a gap matrix with a gap matrix - gap-filling type: Let \mathcal{AG} and \mathcal{BG} be gap matrices. The product \otimes_5 defined below (representing the gap-path obtained by filling the gap in a gap-path by a gap-path) results in a gap matrix \mathcal{CG} with entries:

$$(\mathcal{AG} \otimes_5 \mathcal{BG})[A, (C, D), B] = \sum_{C', D'} \mathcal{AG}[A, (C', D'), B] \cdot \mathcal{BG}[C', (C, D), D']$$

Since $(\mathcal{AG} \otimes_5 \mathcal{BG})[A, (C, D), B]$ is a gap-matrix, its norm is:

$$\begin{aligned}
\|\mathcal{AG} \otimes_5 \mathcal{BG}\| &= (\max_A \sum_C \max_D \sum_B (\mathcal{AG} \otimes_5 \mathcal{BG})[A, (C, D), B]) \\
&= \max_A \sum_C \max_D \sum_B \sum_{C', D'} \mathcal{AG}[A, (C', D'), B] \cdot \mathcal{BG}[C', (C, D), D'] \\
&\leq (\max_A \sum_{C'} \max_{D'} \sum_B \mathcal{AG}[A, (C', D'), B]) \\
&\quad (\max_{C'} \sum_C \max_D \sum_{D'} \mathcal{BG}[C', (C, D), D']) \\
&= \|\mathcal{AG}\| \|\mathcal{BG}\|.
\end{aligned}$$

5.2 The Pseudorandom Generator

We begin by extending the generator defined by Nisan [10, 11]. This extension is to take care of the division of the verification of realizability into three parts (rather than two parts as in Nisan [11]).

Notation: For each $i = 1, 2, 3$, let $h_{1..k}^i$ denote k functions $\langle h_1^i, \dots, h_k^i \rangle \in \mathcal{H}^k$.

Definition 12 Let m be an integer. Let \mathcal{H} be a universal family of hash functions from $\{0,1\}^m$ to $\{0,1\}^m$. For $k \geq 0$, define the generator $\mathcal{G}_k : \{0,1\}^m \times \mathcal{H}^{3k} \rightarrow \{0,1\}^{\frac{(4^k + 1 - 1)m}{3}}$. recursively as follows:

$$\begin{aligned} \mathcal{G}_0(x) &= x \\ \mathcal{G}_k(x : h_{1..k}^1; h_{1..k}^2; h_{1..k}^3) &= x \circ \mathcal{G}_{k-1}(x : h_{1..k-1}^1; h_{1..k-1}^2; h_{1..k-1}^3) \\ &\quad \circ \mathcal{G}_{k-1}(h_k^1(x) : h_{1..k-1}^1; h_{1..k-1}^2; h_{1..k-1}^3) \\ &\quad \circ \mathcal{G}_{k-1}(h_k^2(x) : h_{1..k-1}^1; h_{1..k-1}^2; h_{1..k-1}^3) \\ &\quad \circ \mathcal{G}_{k-1}(h_k^3(x) : h_{1..k-1}^1; h_{1..k-1}^2; h_{1..k-1}^3). \end{aligned}$$

5.3 Notations

5.3.1 Definitions for the case of pseudorandom strings

Definition 13 For $k \geq 0$, let $PS_k \langle h_{1..k}^1; h_{1..k}^2; h_{1..k}^3 \rangle [A, B]$ be the probability that $\mathcal{G}_k(x : h_{1..k}^1; h_{1..k}^2; h_{1..k}^3)$, for x chosen uniformly at random from $\{0,1\}^m$, verifies that a pair of configurations (A, B) is realizable.

Definition 14 For $k \geq 0$, let $PG_k \langle h_{1..k}^1; h_{1..k}^2; h_{1..k}^3 \rangle [A, (C, D), B]$ be the probability that $\mathcal{G}_k(x : h_{1..k}^1; h_{1..k}^2; h_{1..k}^3)$, for x chosen uniformly at random from $\{0,1\}^m$, verifies that a pair-with-gap $(A, (C, D), B)$ is realizable.

Definition 15 For every quadruple of configurations (C, E, F, G) , where C is a push configuration and F is a pop configuration, for $k \geq 0$, let $PSG_k \langle h_{1..k}^1; h_{1..k}^2; h_{1..k}^3 \rangle [C, (E, F), G]$ be the probability that $\mathcal{G}_k(x : h_{1..k}^1; h_{1..k}^2; h_{1..k}^3)$ for x chosen uniformly at random from $\{0,1\}^m$, verifies that:

- C pushes a symbol a and goes to E , and
- F pops the same symbol a and goes to G

Note that the verification used in Definition 15 is a special case of the verification that a pair-with-gap is realizable.

(On the right-hand side of the following definitions, we omit the use of the hash functions to reduce clutter. The hash functions used for the probability matrices on both sides of the definition are the same.)

Definition 16 For every pair of configurations (A, B) and for $h_{1..k-1}^1; h_{1..k-1}^2; h_{1..k-1}^3 \in \mathcal{H}^{3k}$ let

$$\begin{aligned} PS_{4k-1} \langle h_{1..k-1}^1; h_{1..k-1}^2; h_{1..k-1}^3 \rangle [A, B] &= \\ \sum_{C, E, F, G, D} PG_{k-1}[A, (C, D), B] PSG_{k-1}[C, (E, F), G] PS_{k-1}[E, F] PS_{k-1}[G, D]. \end{aligned}$$

Definition 17 For every quadruple of configurations (A, B, C, D) and for $h_{1..k-1}^1; h_{1..k-1}^2; h_{1..k-1}^3 \in \mathcal{H}^{3k}$ let

$$\begin{aligned} PG_{4k-1} \langle h_{1..k-1}^1; h_{1..k-1}^2; h_{1..k-1}^3 \rangle [A, (C, D), B] &= \\ \sum_{C', E', F', G', D'} PG_{k-1}[A, (C', D'), B] PSG_{k-1}[C', (E', F'), G'] PG_{k-1}[E', (C, D), F'] PS_k[G', D'] \\ + \sum_{C', E', F', G', D'} PG_{k-1}[A, (C', D'), B] PSG_{k-1}[C', (E', F'), G'] PS_{k-1}[C', G'] PG_k[G', (C, D), D']. \end{aligned}$$

NOTE: Using the notations for matrix products introduced in section 5.1.2, we can write $PS_{4k-1}[A, B]$ and $PG_{4k-1}[A, B]$ as follows (we have omitted the use of the hash functions in the notations for these probability matrices):

$$PS_{4k-1}[A, B] = (PG_{k-1} \otimes_2 ((PSG_{k-1} \otimes_2 PS_{k-1}) \otimes_1 PS_{k-1}))[A, B].$$

$$\begin{aligned} PG_{4k-1}[A, (C, D), B] &= (PG_{k-1} \otimes_5 ((PSG_{k-1} \otimes_5 PG_{k-1}) \otimes_3 PS_{k-1}) \\ &\quad + PG_{k-1} \otimes_5 ((PSG_{k-1} \otimes_2 PS_{k-1}) \otimes_4 PG_{k-1}))[A, (C, D), B]. \end{aligned}$$

5.3.2 Definitions for the case of pure random strings

Definition 18 Let $RS[A, B]$ be the probability that x chosen at random from $\{0, 1\}^{|(A, B)|}$ verifies that the pair of configurations (A, B) is realizable.

Definition 19 Let $RG[A, (C, D), B]$ be the probability that x chosen at random from $\{0, 1\}^{|(A, B)| - |(C, D)|}$ verifies that the pair-with-gap $(A, (C, D), B)$ is realizable.

Definition 20 For every quadruple of configurations (C, E, F, G) , where C is a push configuration and F is a pop configuration, let $RSG[C, (E, F), G]$ be the probability that x chosen at random from $\{0, 1\}^2$ verifies that:

- C pushes a symbol a and goes to E , and
- F pops the same symbol a and goes to G

NOTE: Using the notations for matrix products introduced in section 5.1.2, we can write $RS[A, B]$ as follows:

$$RS[A, B] = (RG \otimes_2 ((RSG \otimes_2 RS) \otimes_1 RS))[A, B].$$

NOTE: For every quadruple of configurations (C, E, F, G) , where C is a push configuration and F is a pop configuration, for all $k \geq 0$, and all $h_{1..k}^1, h_{1..k}^2, h_{1..k}^3 \in \mathcal{H}^{3k}$,

$$PSG_k \langle h_{1..k}^1; h_{1..k}^2; h_{1..k}^3 \rangle [C, (E, F), G] = RSG[C, (E, F), G].$$

5.4 A generalization of theorem 9

The main theorem (Theorem 9) follows as an easy consequence of the more general derandomization result below.

Theorem 21 Let M be a $BP_H \text{LOGCFL}$ machine and w be an input to M . Let $\epsilon > 0$ be a rational number. For all pairs of configurations (A, B) , there is an SC^2 machine that computes hash functions $h_{1..K}^1, h_{1..K}^2, h_{1..K}^3 \in \mathcal{H}^{3K}$ and $PS_K \langle h_{1..K}^1; h_{1..K}^2; h_{1..K}^3 \rangle [A, B]$ such that:

$$|PS_K \langle h_{1..K}^1; h_{1..K}^2; h_{1..K}^3 \rangle [A, B] - RS_K[A, B]| \leq \epsilon.$$

5.5 δ -goodness and the existence of δ -good hash functions

Definition 22 Let $k \geq 1$. A triple $\langle h_k^1, h_k^2, h_k^3 \rangle \in \mathcal{H}^3$ is δ -good for $h_{1..k-1}^1, h_{1..k-1}^2, h_{1..k-1}^3$ if it satisfies the following two conditions:

1. $\|PS_k \langle h_{1..k}^1; h_{1..k}^2; h_{1..k}^3 \rangle - PS_{4k-1} \langle h_{1..k-1}^1; h_{1..k-1}^2; h_{1..k-1}^3 \rangle\| \leq \delta$, and
2. $\|PG_k \langle h_{1..k}^1; h_{1..k}^2; h_{1..k}^3 \rangle - PG_{4k-1} \langle h_{1..k-1}^1; h_{1..k-1}^2; h_{1..k-1}^3 \rangle\| \leq \delta$.

The claim below asserts the existence of good hash functions:

Claim 23 Let n be the number of configurations of M on input w . Let $\delta > 0$ be a rational number. Let $m = 5 + 23 \log_2 n - 2 \log_2 \delta$. For any sequence of functions $h_{1..k-1}^1, h_{1..k-1}^2, h_{1..k-1}^3$ in \mathcal{H} , there exists a triple $\langle h_k^1, h_k^2, h_k^3 \rangle \in \mathcal{H}^3$ such that:

1. For every pair of configurations A, B ,

$$|PS_k \langle h_{1..k}^1; h_{1..k}^2; h_{1..k}^3 \rangle [A, B] - PS_{4k-1} \langle h_{1..k-1}^1; h_{1..k-1}^2; h_{1..k-1}^3 \rangle [A, B]| \leq \frac{\delta}{n^2}.$$

2. For every quadruple of configurations A, B, C, D ,

$$|PG_k \langle h_{1..k}^1; h_{1..k}^2; h_{1..k}^3 \rangle [A, (C, D), B] - PG_{4k-1} \langle h_{1..k-1}^1; h_{1..k-1}^2; h_{1..k-1}^3 \rangle [A, (C, D), B]| \leq \frac{\delta}{n^2}.$$

Proof of Claim: Notations

For a set $A \subseteq \{0, 1\}^m$, let $\rho(A) = \frac{|A|}{2^m}$. Define the following sets:

$AS_{(A,B)}$ is the set of all $x \in \{0, 1\}^m$ such that $\mathcal{G}_{k-1}(x : h_{1..k-1}^1; h_{1..k-1}^2; h_{1..k-1}^3)$ verifies that the pair (A, B) is realizable.

$ASG_{(C,(E,F),G)}$, where C is a push configuration and F is a pop configuration, is the set of all $x \in \{0, 1\}^m$ such that $\mathcal{G}_{k-1}(x : h_{1..k-1}^1; h_{1..k-1}^2; h_{1..k-1}^3)$ verifies that C pushes a symbol a and goes to E and F pops the same symbol a and goes to G . (Note that the definition of ASG is a special case of the definition of AG below.)

$AG_{(A,(C,D),B)}$ is the set of all $x \in \{0, 1\}^m$ such that $\mathcal{G}_{k-1}(x : h_{1..k-1}^1; h_{1..k-1}^2; h_{1..k-1}^3)$ verifies that the pair-with-gap $(A, (C, D), B)$ is realizable.

Using these sets it follows that:

$$PS_k \langle h_{1..k-1}^1, h^1; h_{1..k-1}^2, h^2; h_{1..k-1}^3, h^3 \rangle [A, B] = \sum_{C,E,F,G,D} Pr_x[x \in AG_{(A,(C,D),B)}, h^1(x) \in ASG_{(C,(E,F),G)}, h^2(x) \in AS_{(E,F)}, h^3(x) \in AS_{(G,D)}].$$

$$PG_k \langle h_{1..k-1}^1, h^1; h_{1..k-1}^2, h^2; h_{1..k-1}^3, h^3 \rangle [A, (C, D), B] = \sum_{C',E',F',G'} (Pr_x[x \in AG_{(A,(C',D'),B)}, h^1(x) \in ASG_{(C',(E',F'),G')}, h^2(x) \in AG_{(E',(C,D),F')}, h^3(x) \in AS_{(G',D')}] + Pr_x[x \in AG_{(A,(C',D'),B)}, h^1(x) \in ASG_{(C',(E',F'),G')}, h^2(x) \in AS_{(E',F')}, h^3(x) \in AG_{(G',(C,D),D')}]$$

$$PS_{4k-1} \langle h_{1..k-1}^1; h_{1..k-1}^2; h_{1..k-1}^3 \rangle [A, B] = \sum_{C,E,F,G,D} Pr_x[x \in AG_{(A,(C,D),B)}] Pr_x[x \in ASG_{(C,(E,F),G)}] Pr_x[x \in AS_{(E,F)}] Pr_x[x \in AS_{(G,D)}] = \sum_{C,E,F,G,D} \rho(AG_{(A,(C,D),B)}) \rho(ASG_{(C,(E,F),G)}) \rho(AS_{(E,F)}) \rho(AS_{(G,D)}).$$

$$PG_{4k-1} \langle h_{1..k-1}^1; h_{1..k-1}^2; h_{1..k-1}^3 \rangle [A, (C, D), B] = \sum_{C',E',G',D'} (Pr_x[x \in AG_{(A,(C',D'),B)}] Pr_x[x \in ASG_{(C',(E',F'),G')}] Pr_x[x \in AG_{(E',(C,D),F')}] Pr_x[x \in AS_{(G',D')}] + Pr_x[x \in AG_{(A,(C',D'),B)}] Pr_x[x \in ASG_{(C',(E',F'),G')}] Pr_x[x \in AS_{(E',F')}] Pr_x[x \in AG_{(G',(C,D),D')}] = \sum_{C',E',F',D'} (\rho(AG_{(A,(C',D'),B)}) \rho(ASG_{(C',(E',F'),G')}) \rho(ASG_{(E',(C,D),F')}) \rho(AS_{(G',D')}) + \rho(AG_{(A,(C',D'),B)}) \rho(ASG_{(C',(E',F'),G')}) \rho(AS_{(E',F')}) \rho(AG_{(G',(C,D),D')})$$

Proof of Claim - Inequality 1:

Given a pair of configurations A, B , fix the configurations C, E, F, G, D . By Lemma 8, if a triple $\langle h^1, h^2, h^3 \rangle$ is chosen uniformly at random from \mathcal{H}^3 then with probability at least $(1 - \frac{1}{8n^9})$ the following holds:

$$|Pr_x[x \in AG_{(A,(C,D),B)}, h^1(x) \in ASG_{(C,(E,F),G)}, h^2(x) \in AS_{(E,F)}, h^3(x) \in AS_{(G,D)}] - \rho(AG_{(A,(C,D),B)}) \rho(ASG_{(C,(E,F),G)}) \rho(AS_{(E,F)}) \rho(AS_{(G,D)})| \leq \frac{\delta}{2n^7}$$

(NOTE: In Lemma 8, put $\theta = \frac{\delta}{2n^7}$. Then, $\frac{1}{2^m \theta^2} = \frac{1}{8n^9}$ since $m = 5 + 23 \log_2 n - 2 \log_2 \delta$.)

By an averaging argument, for at least $\frac{7}{8}$ of the triples $\langle h^1, h^2, h^3 \rangle$, the above inequality holds for all configurations A, B, C, E, F, G, D . In the left-hand-side of the inequality, the sum of the first term over all

configurations C, E, F, G, D gives $\text{PS}_k \langle h_{1..k-1}^1, h^1; h_{1..k-1}^2, h^2; h_{1..k-1}^3, h^3 \rangle [A, B]$, and the sum of the second term over all configurations C, E, F, G, D gives $\text{PS}_{4k-1} \langle h_{1..k-1}^1; h_{1..k-1}^2; h_{1..k-1}^3 \rangle [A, B]$. Therefore, for at least $\frac{7}{8}$ of the triples $\langle h^1, h^2, h^3 \rangle$, the first inequality of the Claim holds for all pairs (A, B) :

$$|\text{PS}_k \langle h_{1..k-1}^1, h^1; h_{1..k-1}^2, h^2; h_{1..k-1}^3, h^3 \rangle [A, B] - \text{PS}_{4k-1} \langle h_{1..k-1}^1; h_{1..k-1}^2; h_{1..k-1}^3 \rangle [A, B]| \leq n^5 \left(\frac{\delta}{2n^7} \right) \leq \frac{\delta}{n^2}.$$

Call a triple $\langle h^1, h^2, h^3 \rangle$ *PS-good* if the first inequality of the Claim works for all pairs of configurations (A, B) .

Proof of Claim - Inequality 2 - the two cases

(Case 1) Given configurations A, B, C, D , fix configurations C', E', F', G', D' . By Lemma 8, if a triple $\langle h^1, h^2, h^3 \rangle$ is chosen uniformly at random from \mathcal{H}^3 then with probability at least $(1 - \frac{1}{8n^9})$ the following holds:

$$|\Pr_x [x \in \text{AG}_{(A, (C', D'), B)}, h^1(x) \in \text{ASG}_{(C', (E', F'), G')}, h^2(x) \in \text{AG}_{(E', (C, D), F')}, h^3(x) \in \text{AS}_{(G', D')}] - \rho(\text{AG}_{(A, (C', D'), B)}) \rho(\text{ASG}_{(C', (E', F'), G')}) \rho(\text{AG}_{(E', (C, D), F')}) \rho(\text{AS}_{(G', D')})| \leq \frac{\delta}{2n^7}.$$

By an averaging argument, for at least $\frac{7}{8}$ of the triples $\langle h^1, h^2, h^3 \rangle$, the above inequality holds for all configurations $A, B, C, D, C', E', F', G', D'$. Call a triple $\langle h^1, h^2, h^3 \rangle$ *good* for Case 1 if the inequality holds for all configurations $A, B, C, D, C', E', F', G', D'$.

(Case 2) Similar to Case 1, given configurations A, B, C, D , fix configurations C', E', F', G', D' . By Lemma 8, if a triple $\langle h^1, h^2, h^3 \rangle$ is chosen uniformly at random from \mathcal{H}^3 then with probability at least $(1 - \frac{1}{8n^9})$ the following holds:

$$|\Pr_x [x \in \text{AG}_{(A, (C', D'), B)}, h^1(x) \in \text{ASG}_{(C', (E', F'), G')}, h^2(x) \in \text{AS}_{(E', F')}, h^3(x) \in \text{AG}_{(G', (C, D), D')}] - \rho(\text{AG}_{(A, (C', D'), B)}) \rho(\text{ASG}_{(C', (E', F'), G')}) \rho(\text{AS}_{(E', F')}) \rho(\text{AG}_{(G', (C, D), D')})| \leq \frac{\delta}{2n^7}.$$

By an averaging argument, for at least $\frac{7}{8}$ of the triples $\langle h^1, h^2, h^3 \rangle$, the above inequality holds for all configurations $A, B, C, D, C', E', F', G', D'$. Call a triple $\langle h^1, h^2, h^3 \rangle$ *good* for Case 2 if the inequality holds for all configurations $A, B, C, D, C', E', F', G', D'$.

Proof of Claim - Inequality 2 - Combining the two Cases

By an averaging argument, at least $\frac{3}{4}$ of the triples $\langle h^1, h^2, h^3 \rangle$ are *good* for both cases. Consider a triple $\langle h^1, h^2, h^3 \rangle$ that is *good* for both cases. Fix configurations $A, B, C, D, C', E', F', G', D'$. Then, by adding the left-hand sides of the two inequalities corresponding to Cases 1 and 2, it follows that

$$\begin{aligned} & |\Pr_x [x \in \text{AG}_{(A, (C', D'), B)}, h^1(x) \in \text{ASG}_{(C', (E', F'), G')}, h^2(x) \in \text{AG}_{(E', (C', D'), F')}, h^3(x) \in \text{AS}_{(G', D')}] \\ & + \Pr_x [x \in \text{AG}_{(A, (C', D'), B)}, h^1(x) \in \text{ASG}_{(C', (E', F'), G')}, h^2(x) \in \text{AS}_{(E', F')}, h^3(x) \in \text{AG}_{(G', (C, D), D')}] \\ & \quad - \rho(\text{AG}_{(A, (C', D'), B)}) \rho(\text{ASG}_{(C', (E', F'), G')}) \rho(\text{AG}_{(E', (C, D), F')}) \rho(\text{AS}_{(G', D')}) \\ & \quad + \rho(\text{AG}_{(A, (C', D'), B)}) \rho(\text{ASG}_{(C', (E', F'), G')}) \rho(\text{AS}_{(E', F')}) \rho(\text{AG}_{(G', (C, D), D')})| \\ & \leq \frac{\delta}{n^7}. \end{aligned}$$

In the left-hand-side of the inequality, the sum of the first term over all configurations C', E', F', G', D' gives $\text{PG}_k \langle h_{1..k-1}^1, h^1; h_{1..k-1}^2, h^2; h_{1..k-1}^3, h^3 \rangle [A, (C, D), B]$, and the sum of the second term over all configurations C', E', F', G', D' gives $\text{PG}_{4k-1} \langle h_{1..k-1}^1; h_{1..k-1}^2; h_{1..k-1}^3 \rangle [A, (C, D), B]$. Therefore, for at least $\frac{3}{4}$

of the triples $\langle h^1, h^2, h^3 \rangle$, the second inequality of the claim holds for all $(A, (C, D), B)$:

$$|\text{PG}_k \langle h_{1..k-1}^1, h^1; h_{1..k-1}^2, h^2; h_{1..k-1}^3, h^3 \rangle[A, (C, D), B] - \text{PS}_{4k-1} \langle h_{1..k-1}^1; h_{1..k-1}^2; h_{1..k-1}^3 \rangle[A, (C, D), B]| \leq \frac{\delta}{n^2}.$$

Call a triple $\langle h^1, h^2, h^3 \rangle$ *PG-good* if the second inequality of the Claim works for all quadruple of configurations A, C, D, B .

Proof of Claim - Combining the two inequalities

Since at least $\frac{7}{8}$ of the triples $\langle h^1, h^2, h^3 \rangle$ are *PS-good* and at least $\frac{3}{4}$ of the triples $\langle h^1, h^2, h^3 \rangle$ are *PG-good*, it follows by an averaging argument that at least $\frac{1}{2}$ of the triples are both *PS-good* and *PG-good*.

5.6 The Algorithm

The algorithm described below computes the hash functions and the probability of realizability of pairs of configurations using these hash functions.

We present the algorithm following the structure in Nisan [11]. Let t , a polynomial in $|w|$, be the running time of M on input w . Let n , a polynomial in $|w|$ be the number of configurations of M on input w . We will assume that $t = n$. Let ϵ be a rational number. Let M' be the machine that simulates M on x by executing the main program below.

Note that in procedures $\text{PS}_k\text{COMPUTE}$, $\text{PS}_{4k}\text{M1COMPUTE}$, $\text{PG}_k\text{COMPUTE}$, and $\text{PG}_{4k}\text{M1COMPUTE}$, the verification of realizability is directly done using the machine M in Step 0, and the verification of realizability in the other steps are done using circuits as described in section 3.3.

5.6.1 The main program

Step 1: Define the following constants:

$$K = \log_2 t; \delta = \frac{\epsilon}{8^K}; m = 5 + 23\log_2 n - 2\log_2 \delta;$$

Step 2: For $k = 1 \dots K$ do

Use Procedure FIND to get a triple $\langle h_k^1, h_k^2, h_k^3 \rangle \in \mathcal{H}^3$ that is δ -good for $h_{1..k-1}^1, h_{1..k-1}^2, h_{1..k-1}^3$.

Step 3: For all configurations A, B , with the stack height of A the same as that of B do:

$$\text{PS}_k \langle h_{1..k}^1; h_{1..k}^2; h_{1..k}^3 \rangle[A, B] = \text{PS}_k\text{COMPUTE}(A, B, K) \text{ using } h_{1..k}^1, h_{1..k}^2, h_{1..k}^3.$$

5.6.2 Procedure FIND

INPUT: $k \geq 1$; hash Functions $h_{1..k-1}^1, h_{1..k-1}^2, h_{1..k-1}^3$; rational number $\delta > 0$.

OUTPUT: A triple $\langle h_k^1, h_k^2, h_k^3 \rangle \in \mathcal{H}^3$ that is δ -good for the input hash functions.

Step 1: For all $h_k^1, h_k^2, h_k^3 \in \mathcal{H}$ do

Step 1.1: For all pairs of configurations A, B with the stack height of A the same as that of B :

Compute $p_1 = \text{PS}_k\text{COMPUTE}(A, B, k)$ using $(h_{1..k-1}^1, h_k^1; h_{1..k-1}^2, h_k^2; h_{1..k-1}^3, h_k^3)$.

Compute $p_2 = \text{PS}_{4k}\text{M1COMPUTE}(A, B, k)$ using $(h_{1..k-1}^1, h_k^1; h_{1..k-1}^2, h_k^2; h_{1..k-1}^3, h_k^3)$.

If $|p_1 - p_2| > \frac{\delta}{n^2}$ then exit loop and go to Step 1.3.

Step 1.2: For all quadruples of configurations A, B, C, D with the stack height of A the same as that of B and the stack height of C the same as that of D :

Compute $p_1 = \text{PGkCOMPUTE}(A, B, C, D, k)$ using $(h_{1..k-1}^1, h_k^1; h_{1..k-1}^2, h_k^2; h_{1..k-1}^3, h_k^3)$.
 Compute $p_2 = \text{PG4kM1COMPUTE}(A, B, C, D, k)$ using $(h_{1..k-1}^1, h_k^1; h_{1..k-1}^2, h_k^2; h_{1..k-1}^3, h_k^3)$.
 If $|p_1 - p_2| > \frac{\delta}{n^2}$ then exit loop and go to Step 1.3.

Step 1.3: Next value of h_k^1, h_k^2, h_k^3 .

5.6.3 Procedure PSkCOMPUTE

INPUT: Configurations A, B ; $k \geq 1$; Hash functions $h_{1..k}^1; h_{1..k}^2; h_{1..k}^3$.

OUTPUT: Probability p .

Step 0: ($|A, B| \leq m$):

$count = 0$.

For all $x \in \{0, 1\}^m$ do

Use the first $|A, B|$ bits of x to verify that the pair (A, B) is realizable. If successful
 $count = count + 1$.

(This verification can be done by an SC^2 machine.)

$pvalue = \frac{count}{2^m}$.

Step 1: ($|A, B| > m$): $pvalue = 0$.

$count = 0$.

For all $x \in \{0, 1\}^m$ do

Use $G_k(x)$ to verify that (A, B) is realizable. If successful $count = count + 1$.

(This verification is described in section 3.3. The bits of $G_k(x)$ are not stored but whenever the verification DAuxPDA needs to access a random bit, it generates the bit as part of a block of m bits using the input hash functions. One application of each hash function takes $O(m^2)$ time and $O(m)$ space. So, computing the next m bits takes $O(km^2)$ time and $O(m)$ space. Thus, the verification DAuxPDA uses logarithmic space and polynomial time. By the result of Cook [7], this DAuxPDA can be simulated by an SC^2 machine.)

$pvalue = \frac{count}{2^m}$.

5.6.4 Procedure PGkCOMPUTE

INPUT: Configurations A, B, C, D ; $k \geq 1$; Hash functions $h_{1..k}^1; h_{1..k}^2; h_{1..k}^3$.

OUTPUT: Probability p .

Part 0:

Step 0: ($|A, B| - |C, D| \leq m$):

$count = 0$.

For all $x \in \{0, 1\}^m$ do

Use the first $|A, B| - |C, D|$ bits of x to verify that $(A, (C, D), B)$ is realizable. If successful
 $count = count + 1$.

$pvalue = \frac{count}{2^m}$.

Step 1: ($|A, B| - |C, D| > m$):

$count = 0$.

For all $x \in \{0, 1\}^m$ do

Use $G_k(x)$ to verify that $(A, (C, D), B)$ is realizable. If successful $count = count + 1$.

$pvalue = \frac{count}{2^m}$.

5.6.5 Procedure PS4kM1COMPUTE

INPUT: Configurations A, B ; $k \geq 1$; Hash functions $h_{1..k}^1; h_{1..k}^2; h_{1..k}^3$.

OUTPUT: Probability p .

Step 0: ($|A, B| \leq m$): The verification is done as in Step 0 of procedure PSkCOMPUTE.

Step 1: ($|A, B| > m$): $pvalue = 0$.

Step 2: Take the next configurations C, E, F, G, D such that they satisfy the following conditions of the partition lemma (Lemma 4):

- $|E, F| + |G, D| = |C, D| - 2$,
- $|E, F|, |G, D| \leq \frac{|A, B|}{2} < |C, D|$,
- C is a push configuration,
- F is a pop configuration, and
- D is either a pop configuration or B .

If there are no more valid values for C, E, F, G, D then RETURN($pvalue$).

Step 3:

$count = 0$.

For all $x \in \{0, 1\}^m$ do

Use the string $G_{k-1}(x)$ to verify that (A, B) is realizable with gap (C, D) . If successful then $count = count + 1$.

$p1value = \frac{count}{2^m}$.

$count = 0$.

For all $x \in \{0, 1\}^m$ do

Verify that (a) M moves from C to E pushing a symbol a using the first bit of $G_{k-1}(x)$ and (b) M moves from F to G popping the same symbol a using the second bit of $G_{k-1}(x)$. If successful then $count = count + 1$.

$p2value = \frac{count}{2^m}$.

$count = 0$.

For all $x \in \{0, 1\}^m$ do

Use $G_{k-1}(x)$ to verify that (E, F) is realizable. If successful then $count = count + 1$.

$p3value = \frac{count}{2^m}$.

$count = 0.$

For all $x \in \{0, 1\}^m$ do

Use $G_{k-1}(x)$ to verify that (G, D) is realizable. If successful then $count = count + 1.$

$p4value = \frac{count}{2^m}.$

Step 4: $pvalue = pvalue + p1value * p2value * p3value * p4value.$

Step 5: Repeat from Step 2.

5.6.6 Procedure PG4kM1COMPUTE

INPUT: Configurations A, B, C, D ; $k \geq 1$; Hash functions $h_{1..k}^1$; $h_{1..k}^2$; $h_{1..k}^3$.

OUTPUT: Probability p .

Step 0: ($|(A, (C, D), B)| \leq m$;) The verification is done as in Step 0 of procedure PGkCOMPUTE.

Step 1: ($|(A, (C, D), B)| > m$;) $pvalue = 0.$

Step 2: Take the next configurations C', E', F', G', D' such that they satisfy the following conditions of the partition lemma (Lemma 4:)

- $|(C', D')| = |(E', F')| + |(G', D')| + 2,$
- $|(C', D')| - |(C, D)| > (|(A, B)| - |(C, D)|)/2,$
- C' is a push configuration,
- F' is a pop configuration, and
- D' is either a pop configuration or B .

If there are no more valid values for C', E', F', G', D' then RETURN($pvalue$).

Step 3:

$count = 0.$

For all $x \in \{0, 1\}^m$ do

Use the string $G_{k-1}(x)$ to verify that (A, B) is realizable with gap (C', D') . If successful then $count = count + 1.$

$p1value = \frac{count}{2^m}.$

$count = 0.$

For all $x \in \{0, 1\}^m$ do

Verify that (a) M moves from C' to E' pushing a symbol a using the first bit of $G_{k-1}(x)$ and (b) M moves from F' to G' popping the same symbol a using the second bit of $G_{k-1}(x)$. If successful then $count = count + 1.$

$p2value = \frac{count}{2^m}.$

Step 4: If ($|(E', F')| < \frac{|(A, B)|}{2}$ and $|(G', D')| > \frac{|(A, B)|}{2}$) is not satisfied then do Step 5 and go to Step 6 else do Step 5' and go to Step 6.

Step 5:

$count = 0.$

For all $x \in \{0, 1\}^m$ do

Use $G_{k-1}(x)$ to verify that $(E', (C, D), F')$ is realizable. If successful then $count = count + 1.$

$p3value = \frac{count}{2^m}.$

$count = 0.$

For all $x \in \{0, 1\}^m$ do

Use $G_{k-1}(x)$ to verify that (G', D') is realizable. If successful then $count = count + 1.$

$p4value = \frac{count}{2^m}.$

Step 5':

$count = 0.$

For all $x \in \{0, 1\}^m$ do

Use $G_{k-1}(x)$ to verify that (E', F') is realizable. If successful then $count = count + 1.$

$p3value = \frac{count}{2^m}.$

$count = 0.$

For all $x \in \{0, 1\}^m$ do

Use $G_{k-1}(x)$ to verify that $(G', (C, D), D')$ is realizable. If successful then $count = count + 1.$

next $x.$

$p4value = \frac{count}{2^m}.$

Step 6: $pvalue = pvalue + p1value * p2value * p3value * p4value.$

Step 7: Repeat from Step 2.

5.7 Correctness of FIND

Lemma 24 *Procedure FIND always returns a triple $\langle h_k^1, h_k^2, h_k^3 \rangle \in \mathcal{H}^3$ that is δ -good for $h_{1..k-1}^1, h_{1..k-1}^2, h_{1..k-1}^3$.*

Proof: By Claim 23, for each pair of configurations $(A, B),$

$$|\text{PS}_k \langle h_{1..k}^1; h_{1..k}^2; h_{1..k}^3 \rangle [A, B] - \text{PS}_{4k-1} \langle h_{1..k-1}^1; h_{1..k-1}^2; h_{1..k-1}^3 \rangle [A, B]| \leq \frac{\delta}{n^2}.$$

Therefore, we have

$$\|\text{PS}_k \langle h_{1..k}^1; h_{1..k}^2; h_{1..k}^3 \rangle - \text{PS}_{4k-1} \langle h_{1..k-1}^1; h_{1..k-1}^2; h_{1..k-1}^3 \rangle\| \leq \frac{\delta}{n} \leq \delta.$$

Again, by Claim 23, for each quadruple of configurations $A, B, C, D,$

$$|\text{PG}_k \langle h_{1..k}^1; h_{1..k}^2; h_{1..k}^3 \rangle [A, (C, D), B] - \text{PG}_{4k-1} \langle h_{1..k-1}^1; h_{1..k-1}^2; h_{1..k-1}^3 \rangle [A, (C, D), B]| \leq \frac{\delta}{n^2}.$$

Therefore,

$$\|\text{PG}_k \langle h_{1..k}^1; h_{1..k}^2; h_{1..k}^3 \rangle - \text{PG}_{4k-1} \langle h_{1..k-1}^1; h_{1..k-1}^2; h_{1..k-1}^3 \rangle\| \leq \delta.$$

5.8 Goodness of Approximation at Each Stage

Lemma 25 *If, for every $1 \leq r \leq k$, there is a triple $\langle h_k^1, h_k^2, h_k^3 \rangle \in \mathcal{H}$ that is δ -good for $h_{1..k-1}^1, h_{1..k-1}^2, h_{1..k-1}^3$ then*

$$1. \|PS_k \langle h_{1..k}^1; h_{1..k}^2; h_{1..k}^3 \rangle - RS\| \leq \frac{(8^k - 1)\delta}{7}.$$

$$2. \|PG_k \langle h_{1..k}^1; h_{1..k}^2; h_{1..k}^3 \rangle - RG\| \leq \frac{(8^k - 1)\delta}{7}.$$

(We omit the use of the hash functions in the notations for probability matrices in the proof of this lemma.)

Proof of the first part: The proof is by induction on k . The claim is true when $k = 0$.

Now, $\|PS_k - RS\| \leq \|PS_k - PS_{4k-1}\| + \|PS_{4k-1} - RS\|$.

Applying Lemma 26 with $p = k - 1$, $q = 4$, and $f_p = \frac{(8^{p-1} - 1)\delta}{7}$, we get,

$$\begin{aligned} \|PS_{4k-1} - RS\| &\leq 4 \frac{(8^{k-1} - 1)\delta}{7} \\ &\leq 8 \frac{(8^{k-1} - 1)\delta}{7}. \end{aligned}$$

The first part of Lemma 25 now follows since the first summand is bounded above by δ .

Lemma 26 *Consider the norm:*

$$\|M_p^1 \times_1 M_p^2 \times_2 \cdots M_p^q - R_p^1 \times_1 R_p^2 \times_2 \cdots R_p^q\|$$

where:

- either $(M_p^j = PS_p$ and $R_p^j = RS)$ or $(M_p^j = PG_p$ and $R_p^j = RG)$ or $(M_p^j = PSG_p$ and $R_p^j = RSG)$ for all $1 \leq j \leq q$,
- $\|M_p^j\| \leq 1$ and $\|R_p^j\| \leq 1$ for all $1 \leq j \leq q$,
- $\times_1, \times_2, \dots \in \{\otimes_1, \otimes_2, \otimes_3, \otimes_4, \otimes_5\}$,
- expressions are suitably parenthesized based on the operations, and
- for all $1 \leq j \leq q$, let $\|M_p^j - R_p^j\| \leq f_p$ for some function f_p of p .

Then,

$$\|M_p^1 \times_1 M_p^2 \times_2 \cdots M_p^q - R_p^1 \times_1 R_p^2 \times_2 \cdots R_p^q\| \leq q f_p.$$

Proof: By induction on q and using the fact that $\|PSG_p - RSG\| = 0$ for all p . The left hand side is

$$\begin{aligned} &\leq \|M_p^1 \times_1 M_p^2 \times_2 \cdots M_p^q - R_p^1 \times_1 M_p^2 \times_2 \cdots M_p^q\| + \|R_p^1 \times_1 M_p^2 \times_2 \cdots M_p^q - R_p^1 \times_1 R_p^2 \times_2 \cdots R_p^q\| \\ &\leq \|M_p^1 - R_p^1\| \|M_p^2 \times_2 \cdots M_p^q\| + \|R_p^1\| \|M_p^2 \times_2 \cdots M_p^q - R_p^2 \times_2 \cdots R_p^q\| \\ &\leq f_p + (q-1) f_p = q f_p. \end{aligned}$$

Proof of the second part: The proof is by induction on k . The claim is true when $k = 0$.

Now, $\|\text{PG}_k - \text{RG}\| \leq \|\text{PG}_k - \text{PG}_{4k-1}\| + \|\text{PG}_{4k-1} - \text{RG}\|$.
Applying Lemma 27 with $p = k - 1$, $q = 4$, and $f_p = \frac{(8^p - 1)\delta}{7}$, we get:

$$\|\text{PG}_{4k-1} - \text{RG}\| \leq 8 \delta \frac{(8^{k-1} - 1)\delta}{7}.$$

The second part of Lemma 25 now follows since the first summand is bounded above by δ .

Lemma 27 *Consider the norm*

$$\begin{aligned} & \| (M_p^{11} \times_1 M_p^{12} \times_2 \cdots M_p^{1q} - R^{11} \times_1 R^{12} \times_2 \cdots R^{1q}) \\ & \quad + (M_p^{21} \times_1 M_p^{22} \times_2 \cdots M_p^{2q} - R^{21} \times_1 R^{22} \times_2 \cdots R^{2q}) \| \end{aligned}$$

where:

- either $(M_p^{ij} = \text{PS}_p$ and $(R^{ij} = \text{RS})$ or $(M_p^{ij} = \text{PG}_p$ and $R^{ij} = \text{RG})$ or $(M_p^{ij} = \text{PSG}_p$ and $R^{ij} = \text{RSG})$ for all $1 \leq i \leq 2$ and $1 \leq j \leq q$,
- $\|M_p^{ij}\| \leq 1$ and $\|R^{ij}\| \leq 1$ for all $1 \leq i \leq 2$ and $1 \leq j \leq q$,
- $\times_1, \times_2, \dots \in \{\otimes_1, \otimes_2, \otimes_3, \otimes_4, \otimes_5\}$,
- expressions are suitably parenthesized based on the operations, and
- for all $1 \leq i \leq 2$ and $1 \leq j \leq q$, let $\|M_p^{ij} - R^{ij}\| \leq f_p$ for some function f_p of p .

Then,

$$\begin{aligned} & \| (M_p^{11} \times_1 M_p^{12} \times_2 \cdots M_p^{1q} - R^{11} \times_1 R^{12} \times_2 \cdots R^{1q}) \\ & \quad + (M_p^{21} \times_1 M_p^{22} \times_2 \cdots M_p^{2q} - R^{21} \times_1 R^{22} \times_2 \cdots R^{2q}) \| \leq 2q f_p. \end{aligned}$$

Proof:

$$\begin{aligned} & \| (M_p^{11} \times_1 M_p^{12} \times_2 \cdots M_p^{1q} - R^{11} \times_1 R^{12} \times_2 \cdots R^{1q}) \\ & \quad + (M_p^{21} \times_1 M_p^{22} \times_2 \cdots M_p^{2q} - R^{21} \times_1 R^{22} \times_2 \cdots R^{2q}) \| \\ & \leq \| (M_p^{11} \times_1 M_p^{12} \times_2 \cdots M_p^{1q} - R^{11} \times_1 R^{12} \times_2 \cdots R^{1q}) \| \\ & \quad + \| (M_p^{21} \times_1 M_p^{22} \times_2 \cdots M_p^{2q} - R^{21} \times_1 R^{22} \times_2 \cdots R^{2q}) \| \end{aligned}$$

The conclusion follows since, by Lemma 26, each of the two summands above are at most $q f_p$.

5.9 Approximation Theorem

Theorem 28 *Let t , a polynomial in $|w|$, be the running time of M on input w . Let n , a polynomial in $|w|$ be the number of surface configurations of M on input w . We will assume that $t = n$. Let ϵ be a rational number. Define: $K = \log_2 t$; $\delta = \frac{\epsilon}{8^K}$; $m = 5 + 23 \log_2 n - 2 \log_2 \delta$; $t' = \frac{(4^{K+1} - 1)m}{3}$. Then, there is a standard matrix $\mathcal{A}_{t'}$ such that $\|\mathcal{A}_{t'} - \text{RS}\| \leq \epsilon$.*

Proof It follows from Lemma 24 that, for all $1 \leq k \leq K$, there is a triple $\langle h_k^1, h_k^2, h_k^3 \rangle \in \mathcal{H}$ that is δ -good for $h_{1..k-1}^1, h_{1..k-1}^2, h_{1..k-1}^3$. Let $\mathcal{A}_{t'}[A, B] = \text{PS}_K \langle h_{1..K}^1, h_{1..K}^2, h_{1..K}^3 \rangle [A, B]$ for all pairs of surface configurations A, B . Then, by Lemma 25, we conclude that $\|\mathcal{A}_{t'} - \text{RS}\| \leq \frac{(8^K - 1)\delta}{7} \leq \epsilon$.

5.10 Proof of theorem 21

Let M be a $BP_H\text{LOGCFL}$ machine and w be an input to it. Let A and B be two configurations of M on input w . The probability that (A, B) is a realizable pair is $\text{RS}[A, B]$. From the Approximation Theorem (theorem 28), there exists $h_{1..K}^1, h_{1..K}^2, h_{1..K}^3 \in \mathcal{H}^{3K}$ such that $\text{PS}_K \langle h_{1..K}^1; h_{1..K}^2; h_{1..K}^3 \rangle[A, B]$ approximates this probability within ϵ . The machine M' of section 5.6 computes $h_{1..K}^1, h_{1..K}^2, h_{1..K}^3 \in \mathcal{H}^{3K}$ and $\text{PS}_K \langle h_{1..K}^1; h_{1..K}^2; h_{1..K}^3 \rangle[A, B]$ and, as shown below, uses $O(\log^2 n)$ space and polynomial time simultaneously.

Complexity Analysis: Two key properties of the hash functions used in the analysis are that they: (a) are computable in logarithmic space, and (b) can be represented using logarithmic space. The simulating machine M' uses $O(\log^2 n)$ space to store the hash functions. Further, apart from the space used by the verifiers in the procedures PSkCOMPUTE , PGkCOMPUTE , PS4kM1COMPUTE , and PG4kM1COMPUTE , the machine M' uses $O(\log n)$ space. The verifiers in these procedures are SC^2 machines that are used a polynomial number of times independently. Note that, as in Nisan [11], the pseudorandom strings used by the verifiers in these procedures are not stored but generated one bit at a time. Hence, the total space used by M' is $O(\log^2 n)$. Finally, it can be checked that the machine M' takes polynomial time.

5.11 Proof of Theorem 9

The proof of the main theorem is an easy consequence of theorem 21 as shown below:

Let P_0 be the initial configuration and Q_a be the unique accepting configuration of a $BP_H\text{LOGCFL}$ machine M on input w . The probability that M accepts w is $\text{RS}[P_0, Q_a]$. From theorem 21, we see that an SC^2 machine can compute $h_{1..k}^1, h_{1..k}^2, h_{1..k}^3 \in \mathcal{H}^{3k}$ and $\text{PS}_K \langle h_{1..k}^1; h_{1..k}^2; h_{1..k}^3 \rangle[P_0, Q_a]$ which approximates $\text{RS}[P_0, Q_a]$ within ϵ . Using this approximation, an SC^2 machine can decide if M accepted or rejected w with probability at least $2/3$.

6 Placing $BP_H\text{LOGCFL}$ in the NC hierarchy

As the case for BP_HL , the class $BP_H\text{LOGCFL}$ can be shown to be included in NC^2 .

Theorem 29 $BP_H\text{LOGCFL} \subseteq NC^2$

Proof Let $P_H\text{LOGCFL}$ denote the class of languages accepted by unbounded error such machines PAuxPDA with logarithmic space-bound and polynomial time-bound. These are machines that accept strings whose acceptance probability is greater than $\frac{1}{2}$. Macarie and Ogihara [8] studied unbounded error classes using different variants of the PAuxPDA model and, among other results, showed that $P_H\text{LOGCFL} = \text{GAPLOGCFL}$. Here GAPLOGCFL is the class of functions that are expressible as the difference of two $\sharp\text{LOGCFL}$ functions. Since $\text{GAPLOGCFL} \subseteq TC^1 \subseteq NC^2$ [20, 1], and since $BP_H\text{LOGCFL} \subseteq P_H\text{LOGCFL}$, it follows that $BP_H\text{LOGCFL} \subseteq \text{GAPLOGCFL} \subseteq TC^1 \subseteq NC^2$.

In this section, we present a circuit-based proof of the result that $P_H\text{LOGCFL} \subseteq \text{GAPLOGCFL}$.

Let M be a $P_H\text{LOGCFL}$ machine and let t denote its time bound. Let w be an input to M , $p(M, w)$ be the probability of acceptance of M by w , and $NAP(M, w)$ be the number of accepting paths of M on w . Then,

$$p(M, w) = \frac{NAP(M, w)}{2^t}.$$

From M and w , a uniform family of circuits, say $\{G_t\}$ (here t is the time bound on M), and an input x can be constructed such that the number of accepting parse-trees, $NAT(G_t, x)$ of G_t is the same as the number of accepting paths of M on w [1, 9, 20].

Therefore, $p(M, w) = \frac{NAT(G_t, x)}{2^t}$.

Let A_t be the arithmetic version of G_t obtained by replacing the Boolean operations OR and AND by the arithmetic operations PLUS and MULT respectively. Then, it is well-known that the output of A_t on x is the number of accepting parse-trees of G_t on x . That is, A_t computes a function in $\#LOGCFL$. Let B_t be a semi-unbounded fan-in arithmetic circuit that computes 2^{t-1} . Clearly, the function computed by B_t is in $\#LOGCFL$. The conclusion follows since testing if $p(M, w) > \frac{1}{2}$ is equivalent to testing if the GAPLOGCFL function $NAT(G_t, x) - 2^{t-1}$ is greater than 0.

6.1 Open questions

The new classes in this paper can be characterized in terms of log space probabilistic Turing machines with polynomial size proofs. Our results show that the derandomization result of Nisan [11] and the result showing that BP_{HL} is in NC^2 [3, 2] apply, more generally, to machines with polynomial proof size.

In a similar vein, it would be interesting to study generalizations such as the ones below of other known results about probabilistic log space classes: (a) the generalization of the result in [16] that $BP_{HL} \subseteq DSPACE(\log^{3/2} n)$ to $BP_H LOGCFL$; (b) the analog of the recent time-space trade-off result in [4] (c) the power of read-once access to random bits versus multiple access (as in [12]); and (d) the relationship between $R_H LOGCFL$ and $LOGDCFL$ (analogous to the relationship between R_{HL} and $DSPACE(\log n)$). It would also be interesting to study the relationships among probabilistic log space classes and the ones considered in the paper.

From an algorithms perspective, it would be interesting to identify: (a) natural problems that are in $BP_H LOGCFL$ that can be derandomized using this approach (analogous to Sivakumar's list [17] for BP_{HL}); (b) promise problems that are complete for these classes (as in Reingold et. al [13]).

Another direction is to identify and study natural mathematical structures that capture the computations in the considered classes analogous to finite Markov chains that capture computations in probabilistic log space classes.

Acknowledgments: I thank Nisheeth Vishnoi for simplifying my original formulation of the Pseudorandom Generator, and working with me on the generalization (Lemma 8) of the hash-mixing lemma (Lemma 7). I also thank him for the many discussions that we had on Nisan's derandomization result [11].

References

- [1] E. Allender, J. Jiao, M. Mahajan, and V. Vinay. Non-commutative arithmetic circuits: Depth reduction and size lower bounds. *Theoretical Computer Science* 209:47-86, 1998.
- [2] E. Allender and M. Ogihara. Relationships among PL , $\#L$, and the Determinant. *RAIRO - Theoretical Information and Applications* 30:1-21, 1996.
- [3] A. Borodin, S.A. Cook, and N. Pippenger. Parallel computation for well-endowed rings and space-bounded probabilistic machines. *Information and Control* 48:113-136, 1983.
- [4] J. Cai, V.T. Chakravarthy, and D. van Melkebeek. Time-Space tradeoff in derandomizing probabilistic logspace. In *Symposium on Theoretical Aspects of Computer Science*, pages 511-583, 2004.
- [5] L. Carter and M. Wegman. Universal hash functions. *Journal of Computer and System Sciences*, 18(2):143-154, 1979.
- [6] S.A. Cook. Characterizations of pushdown machines in terms of time-bounded Computers. *Journal of the Association for Computing Machinery* 18:4-18, 1971.

- [7] S.A. Cook. Deterministic CFL's are accepted simultaneously in polynomial time and log squared space. In *11th ACM Symposium on Theory of Computing*, pages 338-345., 1979.
- [8] I. Macarie and M. Ogihara. Properties of probabilistic pushdown automata. *Theoretical Computer Science*, 207(1):117-130, 1998.
- [9] R. Niedermeier and P. Rossmanith. Unambiguous auxiliary pushdown automata and semi-unbounded fan-in circuits. *Information and Computation*, 118(2):227-245, 1995.
- [10] N. Nisan. Pseudorandom generators for space-bounded computation. *Combinatorica*, 12(4):449-461, 1992.
- [11] N. Nisan. $RL \subseteq SC$. In *24th ACM Symposium on Theory of Computing*, pages 619-623, 1992.
- [12] N. Nisan. On read-once vs. multiple access to randomness in logspace. *Theoretical Computer Science* 107:135-144, 1993.
- [13] O. Reingold, L. Trevisan, and S. Vadhan. Pseudorandom walks in biregular graphs and the **RL** versus **L** problem. *ECCC technical report*, Report 22, 2005.
- [14] W. L. Ruzzo. Tree-size bounded alternation, *Journal of Computer and System Sciences* 20:218-235, 1980.
- [15] M. Saks. Randomization and derandomization in space-bounded computation. In *11th Annual Conference on Computational Complexity*, pages 1-22, 1996.
- [16] M. Saks and S. Zhou. $RSPACE(s) \subseteq DSPACE(s^{3/2})$. *Journal of Computer and System Sciences*, 58:376-403, 1999.
- [17] D. Sivakumar. Algorithmic derandomization via complexity theory. In *34th ACM Symposium on Theory of Computing*, 2002.
- [18] I. H. Sudborough. Time and tape bounded auxiliary pushdown automata. In *Mathematical Foundations of Computer Science*, Lecture Notes in Computer Science, No. 53, Springer Verlag, pages 493-503, 1977.
- [19] I. H. Sudborough. On the tape complexity of deterministic context-free languages. *Journal of the Association for Computing Machinery* 25(3):405-414, 1978.
- [20] V. Vinay. Counting auxiliary pushdown automata and semi-unbounded arithmetic circuits. In *6th Annual Conference on Structure in Complexity Theory*, pages 1-22, 1996, pages 270-284.