Journal of Universal Computer Science, vol. 21, no. 12 (2015), 1563-1576
submitted: 22/5/14, accepted: 27/1/15, appeared: 1/12/15 © J.UCS

Deterministic Frequency Pushdown Automata

Cristian S. Calude
(Department of Computer Science, The University of Auckland
Auckland, New Zealand
cristian@cs.auckland.ac.nz)

Rūsiņš Freivalds
(Institute of Mathematics and Computer Science
University of Latvia, Riga, Latvia
rusins.freivalds@mii.lu.lv)

Sanjay Jain
(School of Computing, National University of Singapore
Singapore
sanjay@comp.nus.edu.sg)

Frank Stephan
(Department of Mathematics and School of Computing
National University of Singapore, Singapore
fstephan@comp.nus.edu.sg)

Abstract: A set L is (m, n)-computable iff there is a mechanism which on input of n different words produces n conjectures whether these words are in L, respectively, such that at least m of these conjectures are right. Prior studies dealt with (m, n)-computable sets in the contexts of recursion theory, complexity theory and the theory of finite automata. The present work aims to do this with respect to computations by deterministic pushdown automata (using one common stack while processing all input words in parallel).

We prove the existence of a deterministic context-free language L which is recognised by an $(1, 1)$-DPDA but fails to be recognised by any (m, n)-DPDA, where $n \geq 2$ and $m \geq n/2+1$. This answers a question posed by Eli Shamir at LATA 2013. Furthermore, it is shown that there is a language L such that, for all m, n with $m \leq n/2$, L can be recognised by an (m, n)-DPDA but, for all m, n with $1 \leq m \leq n$, L cannot be recognised by (m, n)-DFA.

Key Words: frequency computation, deterministic pushdown automata, context-free sets, regular sets.
Category: F.1.1, F.1.2.

1 Introduction

During a discussion of the paper [Freivalds et al. 2013] at the conference *LATA* 2013 in Bilbao, Spain, Eli Shamir asked whether the results on frequency Turing machines and frequency finite automata hold for pushdown automata as well.

The difficulty of the question is in the fact that an (n, n)-Turing machine or an (n, n)-finite automaton can be presented as a Cartesian product of n separate Turing machines or finite automata but this construction does not seem to increase the power of the machine. However, an arbitrary Turing machine can be simulated by an automaton with 3 pushdown tapes (and allowing some rearrangement, even with 2 pushdown tapes [Bārzdiņš 1962]). Hence the possible definition of a frequency pushdown automaton should avoid the use of several pushdown stacks in a single automaton.

2 Frequency computation

The notion of frequency computation was introduced by Rose [Rose 1960] as an attempt to have a deterministic notion of computation with properties similar to probabilistic algorithms. Let $\mathbb{N} = \{0, 1, 2, \ldots\}$ denote the set of all natural numbers, $\mathbb{N}_+ = \mathbb{N} \setminus \{0\}$. Fix $m, n \in \mathbb{N}, 1 \leq m \leq n$. The ith component of the m-tuple $(x_1, \ldots, x_m)$ is denoted by $(x_1, \ldots, x_m)_i$.

A function $f : \mathbb{N} \to \mathbb{N}$ is (m, n)-*computable* if there exists a computable function $R : \mathbb{N}^n \to \mathbb{N}^n$ such that for all n-tuples $(x_1, \ldots, x_n) \in \mathbb{N}^n$ of mutually distinct natural numbers we have the following property:

$$\text{card}\{i : 1 \leq i \leq n \text{ and } (R(x_1, \ldots, x_n))_i = f(x_i)\} \geq m.$$

Answering a problem by Myhill, see McNaughton [McNaughton 1961], Trakhtenbrot [Trakhtenbrot 1964] proved the following: (1) if $2m > n$ then every (m, n)-computable function is computable, (2) if $2m = n$, then f can be not computable. Kinber [Kinber 1972, Kinber 1976] extended these results by considering frequency enumeration of sets and proved that the class of (m, n)-computable sets equals the class of computable sets if and only if $2m > n$.

The notion of frequency computation has been extended to other models of computation. Frequency computation in polynomial time was discussed in detail by Hinrichs and Wechsung [Hinrichs and Wechsung 1997]. For resource bounded computations, the behaviour of frequency computability is completely different. For example, under any reasonable resource bound, whenever $n' - m' > n - m$, there exist sets which are (m', n')-computable, but not (m, n)-computable. However, scaling down to finite automata, the analogue of Trakhtenbrot's result holds again: the class of languages (m, n)-recognisable by deterministic frequency automata equals the class of regular languages if and only if $2m > n$; for $2m \leq n$, the class of languages (m, n)-recognisable by deterministic frequency automata is uncountable for a two-letter alphabet, see [Austinat et al. 2005, Kinber 1976]. When restricted to a one-letter alphabet, only regular languages can be (m, n)-recognised by any finite automaton [Austinat et al. 2005, Kinber 1976].

Frequency computations became increasingly popular after the discovery of the various links between frequency computation and computation with a constant number of queries and were studied in various research articles like [Ablaev and Freivalds 1986, Austinat et al. 2005, Balodis et al. 2012] as well as [Beigel et al. 1996, Case et al. 1997, Dëgtev 1981, Freivalds 1991] and furthermore [Harizanov et al. 1992, Kinber et al. 1995, Kummer 1992].

3 Frequency pushdown automata

Let Σ be any finite alphabet, and let Σ^* be the free monoid generated by Σ. The binary alphabet $\{0,1\}$ is denoted by $\mathbb{B}$; $\mathbb{B}^\infty$ is the set of binary ω-words, i.e. infinite sequences of bits. Every subset $L \subseteq \Sigma^*$ is said to be a *language*. The elements of Σ^* are called *strings*; $|x|$ denotes the *length* of a string $x \in \Sigma^*$. By $\chi_L \colon \Sigma^* \to \{0,1\}$ we denote the *characteristic function* of L.

A *deterministic pushdown automaton* (DPDA) is a pushdown automaton which always has at most one choice about how to procede; formally the DPDA is a 7-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, Z, F)$, where Q is a finite set of states, Σ is a finite set called the input alphabet, Γ is a finite set called the stack alphabet, $q_0 \in Q$ is the start state, $Z \in \Gamma$ is the initial stack symbol and $F \subseteq Q$ is the set of accepting states. Furthermore, $\delta \subseteq Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma \times Q \times \Gamma^*$, where (for determinism) it is required that for all $q \in Q, a \in \Sigma \cup \{\epsilon\}$ and $A \in \Gamma$, there is at most one element in δ of the form $(q, a, A, \cdot, \cdot)$. Furthermore, if $(q, \epsilon, A, \cdot, \cdot) \in \delta$, then for all $a \in \Sigma$, $(q, a, A, \cdot, \cdot) \notin \delta$. An element $(p, a, A, q, \alpha) \in \delta$ is a transition of M. Its meaning is that M, in state $p \in Q$, consuming $a \in \Sigma \cup \{\varepsilon\}$ from the input and with $A \in \Gamma$ as the topmost stack symbol, M changes the state to q, pops A from the stack and pushes α onto the stack (by convention, the last symbol of α is pushed first onto the stack); here $a = \epsilon$ means that no input symbol is consumed. Note that we can also consider δ as a function from $Q \times (\Sigma \cup \{\epsilon\}) \times \Gamma^*$ to $Q \times \Gamma^*$, where $(q, a, A, p, \beta) \in \delta$ means $\delta(q, a, A\alpha) = (p, \beta\alpha)$, for all $\alpha \in \Gamma^*$. Here, $A\alpha$ represents the content of the stack (topmost symbol first), before the transition and $\beta\alpha$ represents the content of the stack after the transition. Then, $\delta^*(q, w, \alpha) = (p, \beta)$, where one repeatedly applies δ, on initial symbol (or ϵ) of remaining part of w until the string w is consumed and no further moves are possible. If w is never consumed by the DPDA, or it keeps on making ϵ moves after consuming w, then $\delta^*(q, w, \alpha)$ is undefined. More formally, one can define δ^* as follows.

Base Case: Suppose $A \in \Gamma$, $\alpha \in \Gamma^*$, $w \in \Sigma^*$. If $\delta(q, \epsilon, A)$ is not defined, then $\delta^*(q, \epsilon, A\alpha) = (q, A\alpha)$. For $a \in \Sigma$, if $\delta(q, \epsilon, A)$ and $\delta(q, a, A)$ are not defined, then $\delta^*(q, aw, A)$ is not defined. Furthermore, $\delta^*(q, w, \epsilon)$ is not defined for any non-empty string w.

Inductive step: Suppose $A \in \Gamma$, $\alpha \in \Gamma^*$, $w \in \Sigma^*$, and $a \in \Sigma \cup \{\epsilon\}$. If $\delta(q, a, A) = (p, \beta)$, then $\delta^*(q, aw, A\alpha) = \delta^*(p, w, \beta\alpha)$.

Note that it is possible that inductive step never ends for some strings (due to repeated application of ϵ moves which never empties the stack). In this case also we say that $\delta^*(q, w, \alpha)$ is undefined. The DPDA accepts a string w if $\delta^*(q_0, w, Z_0) = (q_f, \alpha)$, for some $q_f \in F$.

For an (m, n) frequency pushdown automaton, we modify the above definition allowing n input strings. However, we need to be aware that for the general case input strings can be of distinct lengths. Our definition closely models the definition of frequency computation by finite automata [Austinat et al. 2005, Freivalds et al. 2013, Kinber 1976].

A *deterministic* (m, n)*-frequency automaton* ((m, n)-DFA) is a 7-tuple $M = (Q, \Sigma, \#, \delta, q_0, \tau, n)$, where $1 \leq m \leq n$, Q is a finite set of states, q_0 is the initial state, Σ is a finite alphabet and $\#$ is a symbol not in Σ. The mapping $\delta\colon Q \times (\Sigma \cup \{\#\})^n \to Q$ is the transition function; the function $\tau\colon Q \to \mathbb{B}^n$ is the type of state used for outputs. The type is interpreted as an n-tuple of answers α_i: its i-th component records whether the i-th input string read from the i-th input up to the current position belongs to the language. For simplicity, we use the notation $M(x_1, \ldots, x_n)$ to denote the type of the state after reading as input the strings $(x_1\#^{\ell_1}, \ldots, x_n\#^{\ell_n})$.

A language $L \subseteq \Sigma^*$ is said to be (m, n)*-recognised* by an (m, n)-DFA M if for each n-tuple $(x_1, \ldots, x_n) \in (\Sigma^*)^n$ of pairwise distinct strings the tuples $M(x_1, \ldots, x_n)$ and $(\chi_L(x_1), \ldots, \chi_L(x_n))$ coincide on at least m components.

To define deterministic (m, n)-frequency pushdown automata (with only one pushdown stack) the transition function δ can be extended to n-tuples. A *deterministic* (m, n)*-frequency pushdown automaton* ((m, n)-DPDA) is a 9-tuple $M = (Q, \Sigma, \#, \Gamma, \delta, q_0, \tau, Z, F)$, where $\# \notin \Sigma$ and $(Q, (\Sigma \cup \{\#\})^n, \Gamma, \delta, q_0, \tau, Z, F)$ is a DPDA.

For $n \geq 1$, let $x = (x_1, \ldots, x_n) \in (\Sigma^*)^n$ be an n-tuple of strings. We define $|x| = \max\{|x_i| : 1 \leq i \leq n\}$ and let, more formally, M work on the padded input tuple $(x_1\#^{\ell_1}, \ldots, x_n\#^{\ell_n})$, where $\ell_i = |x| - |x_i|$ for all $i = 1, \ldots, n$. Then the output of M is defined to be the type $\tau(q)$ if $\delta^*(q_0, (x_1\#^{\ell_1}, \ldots, x_n\#^{\ell_n}), Z) = (q, \beta)$ for some β; $M(x_1, \ldots, x_n)$ is undefined if $\delta^*(q_0, (x_1\#^{\ell_1}, \ldots, x_n\#^{\ell_n}), Z)$ is undefined.

We point out that the (m, n)-DPDA contains only one pushdown stack which is used to process all n inputs in parallel.

A language $L \subseteq \Sigma^*$ is said to *recognised by an* (m, n)*-DPDA* M if for each n-tuple $(x_1, \ldots, x_n) \in (\Sigma^*)^n$ of pairwise distinct strings, $M(x_1, \ldots, x_n)$ is defined and coincides with $(\chi_L(x_1), \ldots, \chi_L(x_n))$ on at least m components.

4 Basic Facts about the Inclusion Structure

We start with the following obvious facts. Kinber [Kinber 1976] noted the basic properties for frequency computation with finite automata.

Proposition 1. *If L is recognised by an (m,n)-DPDA then L is also recognised by an $(m,n+1)$-DPDA and, in the case that $m,n>1$, also by an $(m-1,n-1)$-DPDA.*

Austinat, Diekert, Hertrampf and Petersen [Austinat et al. 2005] as well as Kinber [Kinber 1976] showed that there is a continuum of sets which is recognisable by a $(1,2)$-DFA. Such a $(1,2)$-DFA is of course also a $(1,2)$-DPDA. Thus one gets the following proposition.

Proposition 2. *There exists a continuum of languages that are recognisable by an $(1,2)$-DPDA.*

Kinber [Kinber 1975, Kinber 1976] and in particular Dëgtev [Dëgtev 1981] gave criteria for proving non-inclusions and one important notion is that of an (m,n)-admissible set [Dëgtev 1981].

Definition 3. A set $V \subseteq \{0,1\}^k$ of vectors is called (m,n)-admissible iff $k \geq n$ and for every projection of V onto n coordinates there is a vector $(b_1,\ldots,b_n)$ which coincides with every member of the projection in at least m coordinates.

An example is the set $\{000,111,100,010,001\}$ which is $(1,3)$-admissible and not $(1,2)$-admissible; furthermore, the set

$$\{00000,11111,00001,00010,00100,01000,10000\}$$

is $(2,5)$-admissible and $(1,3)$-admissible but not $(2,4)$-admissible and not $(1,2)$-admissible. Note that one can always assume, without loss of generality, that one vector in V consists only of zeroes; the reason is that a set $V \subseteq \{0,1\}^k$ is (m,n)-admissible iff $W = \{(b_1 \oplus c_1,\ldots,b_k \oplus c_k) : (b_1,\ldots,b_k) \in V\}$ is (m,n)-admissible, where $(c_1,\ldots,c_k)$ is a fixed vector in $\{0,1\}^k$ and $\oplus$ is the exclusive or; if $(c_1,\ldots,c_k) \in V$ then $(0,\ldots,0) \in W$.

A language L is called (m,n)-verbose if a machine (say DFA or DPDA or TM) reads n inputs $(w_1,\ldots,w_n)$ in parallel and produces m vectors $(b_1,\ldots,b_n)$ such that one of them satisfies $\chi_L(w_1)=b_1,\ldots,\chi_L(w_n)=b_n$; Tantau [Tantau 2002] showed that for each $(m,n),(h,k)$ the following statements are equivalent:

- There is a language L for which some DFA witnesses that L is (m,n)-verbose but no DFA witnesses that L is (h,k)-verbose.
- There is a language L for which some TM witnesses that L is (m,n)-verbose but no TM witnesses that L is (h,k)-verbose.
- There is a language L for which some DFA witnesses that L is (m,n)-verbose but no TM witnesses that L is (h,k)-verbose.

Tantau's techniques to show the non-inclusions can also be adjusted to the world of DPDA and the following result is parallel to the construction of a set which is (m, n)-verbose as witnessed by a DFA but not (h, k)-verbose as witnessed by a Turing machine [Tantau 2002, Theorems 10-12].

Proposition 4. *For all m, n, h, k with $1 \leq m \leq n/2$ and $1 \leq h \leq k$, if there is a set $V \subseteq \{0, 1\}^k$ which is $(m - t, n - 2t)$-admissible for all t with $t < m$ and $n - 2t \leq k$, but not (h, k)-admissible, then there is a language L which is recognised by an (m, n)-DFA and not recognised by any (h, k)-DPDA.*

Proof. Assume that m, n, h, k, V are given as stated in the proposition. Without loss of generality the vector 0^k is in V.

One defines $\Sigma = \{1, 2, \ldots, |V| + k\}$ and let $v_1, \ldots, v_{|V|}$ be the vectors in V. Furthermore, one defines inductively an ω-word $a_0 a_1 \ldots \in \{1, \ldots, |V|\}^\omega$ as follows: For each ℓ, one considers the ℓ's (h, k)-DPDA and its output $(b_1, \ldots, b_k)$ on the k-tuple $(w_1, \ldots, w_k)$ with $w_j = a_0 a_1 \ldots a_{\ell-1} \cdot (|V| + j)$. By assumption there is one vector $v_a \in V$ such that $(b_1, \ldots, b_k)$ differs from v_a in at least $k - h + 1$ coordinates. Now one chooses $a_\ell = a$ for the least such a and defines that $\chi_L(a_0 a_1 \ldots a_{\ell-1} \cdot (|V| + j)) = v_a(j)$ for $j = 1, \ldots, k$. In summary,

$$L = \{a_0 a_1 \ldots a_{\ell-1} \cdot (|V| + j) : v_{a_\ell}(j) = 1\}.$$

So L is based on the inductively defined ω-word $a_0 a_1 \ldots \in \{1, \ldots, |V|\}^\omega$ and L is defined in such a way that no (h, k)-DPDA recognises L.

Next we construct an (m, n)-DFA which recognises L. The (m, n)-DFA reads in parallel words $w_1, \ldots, w_n$. Whenever the (m, n)-DFA detects that there is a pair (i, j) of coordinates such that it has not yet assigned answers to w_i, w_j and either $w_i \notin \{1, \ldots, |V|\}^* \cdot \{|V| + 1, \ldots, |V| + k\}$ or the first digit where w_i, w_j differ is from $\{1, \ldots, |V|\}$ for both inputs then the (m, n)-DFA assigns the value 0 to both coordinates and at least one is right, as it cannot be that both vectors are of the form $a_0 a_1 \ldots a_{\ell-1} \cdot \{|V| + 1, \ldots, |V| + k\}$ for some $\ell \in \omega$.

Whenever the (m, n)-DFA detects that there is a pair (i, j) such that the (m, n)-DFA has not yet assigned answers for w_i, w_j and the first digit where w_i, w_j differ is for w_i a value $|V| + b$ and for w_j a value $a \in \{1, \ldots, |V|\}$ and b is also the last digit of w_i then the (m, n)-DFA assigns to w_i the value $v_a(b)$ and to w_j the value 0. In the case that the 0 for w_j is incorrect, w_j is a member of L and w_i is of the form $a_0 \ldots a_{\ell-1} \cdot (|V| + b)$ for some ℓ and $a_\ell = a$ and thus $\chi_L(w_i) = v_{a_\ell}(b)$. Again at least one of the two guesses is correct.

Let t be the number of pairs which will be processed and for whose members will be assigned answers as above until all inputs are read completely. The remaining $n - 2t$ inputs are taken from the set

$$\tilde{a}_0 \tilde{a}_1 \ldots \tilde{a}_{\ell-1} \cdot \{|V| + 1, \ldots, |V| + k\},$$

for some $\ell \in \omega$ and $\tilde{a}_0\tilde{a}_1 \ldots \tilde{a}_{\ell-1} \in \{1, \ldots, |V|\}^*$. Note that by this construction there are at most k of these inputs. If $t \geq m$ then the (m, n)-DFA assigns just 0 for these remaining inputs, as there are already t correct answers. If $t < m$ then the projection of V onto the corresponding coordinates is $(m - t, n - 2t)$-admissible and one can find values for the remaining coordinates which coincide with the projections on $m - t$ coordinates; furthermore, $m - t$ of the coordinates must be 0, as 0^m is among the projected vectors. In the case that there is an $\ell' < \ell$ such that $\tilde{a}_{\ell'} \neq a_{\ell'}$ then the $m - t$ zeroes are correct and so the (m, n)-DFA provides in total at least m correct answers. In the case that there is no such ℓ' then the projections of v_{a_ℓ} onto the corresponding coordinates coincide with χ_L on these coordinates and thus, by the $(m - t, n - 2t)$-admissibility of V, out of the chosen answers, at least $m - t$ are correct so that the overall correct answers are at least m again. Thus the (m, n)-DFA described above recognises L. $\Box$

Note that the above diagonalisation can also be carried out for diagonalising against (h, k)-TM in place of (h, k)-DPDA. Thus the separation obtained is quite general. The following example provides some separations based on this method.

Example 5. *The set*

$$\{0000, 1111, 0001, 0010, 0100, 1000\}$$

is $(1, 3)$-admissible but not $(2, 4)$-admissible; thus this set witnesses that there is a language L which is recognised by an $(1, 3)$-DFA and a $(2, 5)$-DFA but L is not recognised by a $(2, 4)$-DPDA. The set

$$\{0^{2n}, 1^{2n}, 0^i 1 0^{2n-i-1} : i \in \{1, \ldots, 2n\}\}$$

is $(m-t, 2m+1-2t)$-admissible for all m, t with $2m+1-2t < 2n$ but not $(n, 2n)$-admissible. Thus this set witnesses that there is a language L which is recognised by a $(m, 2m+1)$-DFA *for all m but L is not recognised by a $(n, 2n)$*-DPDA. *The set*

$$\{1^{2k+h}\} \cup \{v \in \{0, 1\}^{2k+h} : v \text{ has at most } h + 1 \text{ times a } 1\}$$

is $(m-t, 2m+h+1-2t)$-admissible for all m, t with $2m+h+1-2t \leq 2k+h$ via the vector $0^{m+h+1-t}1^{m-t}$ but not $(k, 2k + h)$-admissible. Thus this set witnesses that there is a language L which is recognised by a $(m, 2m + 1 + h)$-DFA *for all m but L is not recognised by a $(k, 2k + h)$*-DPDA*.*

The next result does not directly follow from Proposition 4; instead, it needs some adjustment of the proof method in order to obtain the desired separation.

Proposition 6. *For every m, n, k with $1 \leq k \leq n - m$ and $2m \leq n$ there is a language L recognisable by a (m, n)*-DFA *but not by an $(1, k)$*-DPDA*.*

Proof. Assume that $n \geq m+k$ and $m \leq n/2$ and $V = \{0,1\}^k$. Then one can modify the (m,n)-DFA from Proposition 4 such that on n distinct inputs, it computes as follows. The DFA internally brings the strings into length-lexicographic order; so let $w_1, \ldots, w_n$ denote the inputs ordered in this way. Note that $\max\{m+1, i+k\} \leq j \leq n$ by the assumptions. Next for each i such that $1 \leq i \leq m$ and $j = i+n-m$, the DFA decides as follows the outputs for w_i and w_j.

- If w_i, w_j first differ on a symbol which is in $\{1, \ldots, |V|\}$ for both w_i and w_j or if one of w_i, w_j is not from the set $\{1, \ldots, |V|\}^* \cdot \{|V|+1, \ldots, |V|+k\}$ then output 0 for both w_i and w_j.
- Otherwise w_i is of the form $\tilde{a}_0\tilde{a}_1 \ldots \tilde{a}_{\ell-1} \cdot (|V|+b)$ and $w_j \in \tilde{a}_0\tilde{a}_1 \ldots \tilde{a}_{\ell-1}\tilde{a}_\ell \cdot \{1, \ldots, |V|+k\}^+$ for some ℓ and b and $\tilde{a}_0, \ldots, \tilde{a}_\ell \in \{1, \ldots, |V|\}$. In this case output $w_{a_\ell}(b)$ for w_i and 0 for w_j.

In each of the above cases at least one of the answers is correct. The remaining outputs for $w_{m+1}, \ldots, w_{n-m}$ can be set to 0. By the above case analysis, at least m of the output bits are correct. □

The following corollary is a special case of this proposition; it is also directly implied from Tantau's results [Tantau 2002].

Corollary 7. *For every k there is a language recognised by a $(1,k+1)$-DFA but not by an $(1,k)$-DPDA.*

5 Shamir's Question

Shamir asked at LATA 2013 whether there is a deterministic context-free non-regular language L such that, for all $m > 1$, L is not recognised by a (m,m)-DPDA. The next result shows that this is the case. Indeed, it shows that there is a deterministic context-free language L which is thus recognised by a $(1,1)$-DPDA but fails to do so for many (m,n)-DPDA where $n \geq 2$ and m is "near to" n.

Note that the following implication is known for frequency computation [Dëgtev 1981] and also true for DFAs as stated now: If $n \leq k$ and every (m,n)-admissible subset $V \subseteq \{0,1\}^k$ is (h,k)-admissible then every set recognised by an (m,n)-DFA is also recognised by a (h,k)-DFA. The corresponding statement is disproven for DPDAs by the next result, as every $(1,1)$-admissible set $V \subseteq \{0,1\}^2$ consists only of one vector and is therefore $(2,2)$-admissible. Recall that a set is recognised by a $(1,1)$-DPDA iff it is deterministic context-free.

Theorem 8. *The deterministic context-free language $L = \{0^i1^j2^k : i+k = j\}$ is not recognisable by any (m,n)-DPDA for any m,n with $n/2+1 \leq m \leq n$.*

Proof. It is easy to verify that L can be recognised by a DPDA. Intuitively, the DPDA can first push the 0's. When 1's are read, it can pull out the corresponding number of 0's. Then the remaining 1's can again by pushed onto the stack, and on reading 2's they can be pulled out. Formally, the following DPDA recognises L.

$$\begin{aligned} Q &= \{q_0, q_1, q_2, q_3, q_4, q_5, q_6\}, \\ \Sigma &= \{0, 1, 2\}, \\ \Gamma &= \{0, 1, Z_0\}, \\ F &= \{q_0, q_3, q_6\}. \end{aligned}$$

The transition function δ is defined as follows:

$$\begin{aligned} \delta(q_0, 0, Z_0) &= (q_1, 0Z_0), \\ \delta(q_0, 1, Z_0) &= (q_4, 1Z_0), \\ \delta(q_1, 0, 0) &= (q_1, 00), \\ \delta(q_1, 1, 0) &= (q_2, \epsilon), \\ \delta(q_2, 1, 0) &= (q_2, \epsilon), \\ \delta(q_2, \epsilon, Z_0) &= (q_3, Z_0), \\ \delta(q_3, 1, Z_0) &= (q_4, 1Z_0), \\ \delta(q_4, 1, 1) &= (q_4, 11), \\ \delta(q_4, 2, 1) &= (q_5, \epsilon), \\ \delta(q_5, 2, 1) &= (q_5, \epsilon), \\ \delta(q_5, \epsilon, Z_0) &= (q_6, \epsilon). \end{aligned}$$

Assume by way of contradiction that there are m, n and a (m, n)-DPDA M recognising L with $n/2 + 1 \leq m \leq n$. The languages

$$L_{b_1,\ldots,b_n} = \{(u_1, \ldots, u_n) : M(u_1, \ldots, u_n) = (b_1, \ldots, b_n)\}$$

are all context-free languages of convoluted tuples. Furthermore, there is a constant c which is a common pumping constant for all these languages.

Next consider the working of M on inputs $(u_1, \ldots, u_n)$ with

$$u_h = 0^{2ch} 1^{4cn} 2^{4cn-2ch};$$

all u_h are in L and have the same length $8cn$. Due to the pumping lemma, each u_h can be split into words v_h, w_h, x_h, y_h, z_h such that all v_h have the same length, all w_h have the same length, all x_h have the same length, all y_h have the same length, all z_h have the same length,

$$|w_h x_h y_h| \leq c,\ |w_h y_h| \geq 1$$

and the tuple $(\tilde{u}_1, \ldots, \tilde{u}_n)$ of all $\tilde{u}_h = v_h w_h w_h x_h y_h y_h z_h$ is in $L_{b_1,\ldots,b_n}$ for the output $(b_1, \ldots, b_n)$ of M on $(u_1, \ldots, u_n)$, that is,

$$M(u_1, \ldots, u_n) = M(\tilde{u}_1, \ldots, \tilde{u}_n).$$

Note that the border from the 0 part to the 1 part as well as the border from the 1 part to the 2 part in u_h and u_{h+1} differ by $2c$, which is more than the pumping constant c. So, if the length of the v_h is below $2cn$, then, for all h except at most one, either $w_h x_h y_h \in 0^+$ or $w_h x_h y_h \in 1^+$. Similarly, if the length of v_h is at least $2cn$ then for all h except at most one of them, either $w_h x_h y_h \in 1^+$ or $w_h x_h y_h \in 2^+$.

It follows that for all h except at most one, the number of digits of one type in $\tilde{u}_h$ differs from that in u_h while the number of digits of the other two types are the same, thus the constraint that there are as many 1 as 0 and 2 combined gets destroyed. So $u_h \notin L$ for all but at most one h.

By the assumption that M is a (m, n)-DPDA recognising L, at least $n/2+1$ of the bits $b_1, \ldots, b_n$ are 1 due to $M(u_1, \ldots, u_n) = (b_1, \ldots, b_n)$ and at least $n/2$ of those bits are 0 due to $M(\tilde{u}_1, \ldots, \tilde{u}_n) = (b_1, \ldots, b_n)$. These requirements contradict each other, thus there cannot be an (m, n)-DPDA recognising L when $m \geq n/2 + 1$. □

6 Relating DPDAs and DFAs

One could ask whether, in general, there is a closer relation between regularity and recognisability by an (m, n)-DPDA. If $2m \leq n$ then there are uncountably many sets which are recognisable by a (m, n)-DFA [Austinat et al. 2005, Kinber 1976]. So the question would be more precisely phrased as follows: does there exist a pair (m, n) with $1 < m \leq n$ such that there exist sets which are recognisable by an (m, n)-DPDA but not recognisable by a (m, n)-DFA? The answer is affirmative, as the next theorem shows.

Theorem 9. *There exists a language L recognisable by an $(m, 2m)$-DPDA for each $m > 0$, but not recognisable by any $(1, m)$-DFA for every $m > 0$.*

Proof. Let $\Sigma = \{0, 1, 2\}$. Let M_i^j denote the i-th DFA using $j+1$ inputs. For any n, let $s_n = \sum_{\langle i', j' \rangle < n} (j' + 1)$.

A sequence of words $w_0, w_1, \ldots$ in $\{0, 1\}^*$ will be defined below. Let $v_k = w_0 2 w_1 2 \ldots w_k 2$. The following properties will be satisfied:

- $L \subseteq \{v_k : k \in \mathbb{N}\}$;
- For any $n = \langle i, j \rangle$, M_i^j fails to $(1, j+1)$-recognise the set L on input $(v_{s_n}, v_{s_n+1}, \ldots, v_{s_n+j})$;
- For $k \in \mathbb{N}$, $v_k \in L$ iff w_k has equal number of 0's and 1's.

For $n = \langle i, j \rangle$, we suppose that we have defined $w_0, w_1, \ldots, w_{s_n-1}$. Then, we define $w_{s_n}, w_{s_n+1}, \ldots, w_{s_n+j}$ as follows. Suppose the pumping constant for M_i^j

is $h > 2$. Then, initially select w_r, $r \in \{s_n, s_{n+1}, \ldots, s_n + j\}$, as $w_r = 0^{h!+h}1^h$. Suppose that

$$M_i^j(v_{s_n}, v_{s_n+1}, \ldots, v_{s_n+j}) = (b_{s_n}, \ldots, b_{s_n+j}).$$

If b_r is 1, then we leave w_r unchanged. Otherwise, we change w_r to $w_r 1^{h!}$ by pumping in the last part 1^h of w_r; this pumping does not change the behaviour of M_i^j on the input words. Thus all answers of M_i^j are made false.

Next we consider recognition of L by an $(m, 2m)$-DPDA. Suppose the $2m$ input strings are $(x_1, x_2, \ldots, x_{2m})$. The algorithm is as follows: The DPDA distinguishes inputs for which it has settled to an answer and those which are not yet settled. All the not yet settled ones will agree on the input read so far and can therefore be treated by using the same stack. The stack is used to count whether the most recent run of $\{0,1\}^*$ (after the last 2 or the start of the input) has the same number of 0 and 1. Furthermore, whenever the DPDA settles a pair of inputs by assigning answers, one of them has to be correct. The settling (after reading each new input bit of each word) is done as follows:

- If there are x_i, x_j which are not yet settled and are discovered to differ on a bit, then both will get the answer 0 assigned.
  At least one of the answers is correct as at least one of x_i, x_j differs from all members of L (which are prefixes of each other);
- If there are x_i, x_j which are not yet settled such that x_i turns out to be a proper prefix of x_j, then the DPDA checks using its memory/stack whether $x_i \in \{0,1,2\}^* \cdot \{2w2\}$ for a word $w \in \{0,1\}^*$ having as many 0 as 1; if so it settles with 1 for x_i and 0 for x_j else it settles with 0 for both x_i and x_j.
  If $x_j \in L$, then x_i is a prefix of x_j and $x_i \in L$ iff $x_i \in \{0,1,2\}^* \cdot \{2w2\}$ for a word $w \in \{0,1\}^*$ having as many 0 as 1. Hence, if the answer for x_j is incorrect, then the answer for x_i is correct. It follows that at least one of the answers is correct.

Note that settling as above is done for as many pairs as possible after reading a new input symbol for each word.

As all inputs are distinct, for each input x_i there is an x_j such that the pair x_i, x_j gets eventually settled. Furthermore, at any time, all the not yet settled inputs have not shown any disagreement and therefore the DPDA can use its stack in order to check whether the current run of 0 and 1 has both digits in the same quantity; thus whenever an input string ends which has not been settled previously, then the information about whether it ends with $2w2$ such that w has as many 0 as 1 is available. □

Corollary 10. *For all m, n such that $1 \leq m \leq n/2$, there exists a language L recognisable by an (m, n)-DPDA but not by an (m, n)-DFA.*

7 Conclusion

Frequency computation for Turing machines or finite automata permits trivial inclusions in the sense that when L is recognised by an (m, n)-DFA and a (h, k)-DFA then it is also recognised by a $(m + h, n + k)$-DFA. The corresponding statement for DPDA does not hold; the main reason is that the stack management for the combined DPDA does not allow to simulate both stacks of the two original pushdown automata. This fact permitted to show that there is a language L which is recognised by a $(1, 1)$-DPDA but not by a $(2, 2)$-DPDA. This shows that also the inclusion problem in general – for which (m, n), (h, k) is every language recognised by a (m, n)-DPDA also recognised by a (h, k)-DPDA – is different from that for DFA and that in the original frequency computation; it is also different from the inclusion problem in complexity theory.

An interesting question is whether this difference is limited to comparing $(1, n)$ with (h, k) for $h \geq 2$ or whether it is a more fundamental difference. So one might ask whether there is, for example, a language which can be recognised by a $(2, n)$-DPDA for all $n \geq 2$ or at least for almost all n but not by a $(3, m)$-DPDA for any m. Thus the language would be solved at two instances using one stack, but not at three. So far, the construction of such a language is open. Furthermore, in frequency computation by finite automata, one can observe that languages recognised by $(1, n)$-DFAs with $n \geq 2$ have the following property: the ratio h/k of the maximal h for which they are recognised by an (h, k)-DFA goes to $1/2$ for $k \to \infty$. The reason is that for $k \to \infty$, the maximal h where all $(1, n)$-admissible sets $V \subseteq \{0, 1\}^k$ are (h, k)-admissible satisfies this property [Kummer and Stephan 1995a, Kummer and Stephan 1995b]. However, for the case of DPDAs, the situation might be different, as the common stack might have to make one choice which falls into one of the constantly many cases and would then allow only for a frequency which is approximately the share $(k/c, k)$ for all k. Studies in this direction would also certainly be fruitful.

It is easy to show that if L is recognised by an (m, n)-DPDA and H is regular, then also $L \cap H$ and $L \cup H$ are recognised by (m, n)-DPDAs. This holds as one can run the DFA for H in parallel with a (m, n)-DPDA and correspondingly update the outputs. It would be interesting to consider which other closure properties are satisfies by languages recognised by (m, n)-DPDA.

Acknowledgments

The research of the second author was supported by Grant No. 271/2012 from the Latvian Council of Science and in part by Latvian State Research Programme NexIT project No 1. The research of the third author was supported by NUS grants R146-000-181-112 and C252-000-087-001; the research of the fourth author was also supported by NUS grant R146-000-181-112.

References

[Ablaev and Freivalds 1986] Farid M. Ablaev and Rūsiņš Freivalds. Why sometimes probabilistic algorithms can be more effective. *Lecture Notes in Computer Science,* vol. 233, pp. 1–14, 1986.

[Austinat et al. 2005] Holger Austinat, Volker Diekert, Ulrich Hertrampf and Holger Petersen. Regular frequency computations. *Theoretical Computer Science,* vol. 330 no. 1, pp. 15–20, 2005.

[Bārzdiņš 1962] Jānis Bārzdiņš (J.M. Barzdin). On a class of Turing machines (Minsky machines). *Algebra i Logika,* vol. 1, no. 6, pp. 42–51, 1962.

[Balodis et al. 2012] Kaspars Balodis, Ilja Kucevalovs and Rūsiņš Freivalds. Frequency Prediction of Functions. *Lecture Notes in Computer Science,* vol. 7119, pp. 76–83, 2012.

[Beigel et al. 1996] Richard Beigel, William I. Gasarch and Efim B. Kinber. Frequency computation and bounded queries. *Theoretical Computer Science,* vol. 163 no. 1/2, pp. 177–192, 1996.

[Case et al. 1997] John Case, Susanne Kaufmann, Efim B. Kinber and Martin Kummer. Learning recursive functions from approximations. *Journal of Computer and System Sciences,* vol. 55, no. 1, pp. 183–196, 1997.

[Dëgtev 1981] Alexander N. Dëgtev. On (m, n)-computable sets. *Algebraic Systems*, Edited by D.I. Moldavanskij, Ivanovii Gosudarstvennii Universitet, pp. 88–99, 1981.

[Freivalds 1991] Rūsiņš Freivalds. Complexity of probabilistic versus deterministic automata. *Lecture Notes in Computer Science,* vol. 502, pp. 565–613, 1991.

[Freivalds et al. 2013] Rūsiņš Freivalds, Thomas Zeugmann and Grant R. Pogosyan. On the size complexity of deterministic frequency automata. *Lecture Notes in Computer Science,* vol. 7810, pp.287–298, 2013.

[Harizanov et al. 1992] Valentina Harizanov, Martin Kummer and Jim Owings. Frequency computations and the cardinality theorem. *The Journal of Symbolic Logic,* vol. 57, no. 2, pp. 682–687, 1992.

[Hinrichs and Wechsung 1997] Maren Hinrichs and Gerd Wechsung. Time bounded frequency computations. *Information and Computation,* vol. 139, pp. 234-257, 1997.

[Kinber 1972] Efim B. Kinber. Frequency calculations of general recursive predicates and frequency enumeration of sets. *Soviet Mathematics Doklady,* vol. 13, pp. 873–876, 1972.

[Kinber 1975] Efim B. Kinber. *Frequency-computable functions and frequency-enumerable sets.* Candidate Dissertation, Riga, 1975 (Russian).

[Kinber 1976] Efim B. Kinber. Frequency computations in finite automata, *Kibernetika,* no. 2, pp. 7–15, 1976 (Russian; English translation in Cybernetics 12 (1976) 179-187).

[Kinber et al. 1995] Efim B. Kinber, Carl H. Smith, Mahendran Velauthapillai and Rolf Wiehagen. On learning multiple concepts in parallel. *Journal of Computer and System Sciences,* vol. 50, no. 1, pp. 41–52, 1995.

[Kummer 1992] Martin Kummer. A proof of Beigel's cardinality conjecture. *The Journal of Symbolic Logic,* vol. 57, no. 2, pp. 677–681, 1992.

[Kummer and Stephan 1995a] Martin Kummer and Frank Stephan. Recursion theoretic properties of frequency computation and bounded queries. *Information and Computation*, vol. 120, no. 1, pp. 59–77, 1995.

[Kummer and Stephan 1995b] Martin Kummer and Frank Stephan. The power of frequency computation. *Proceedings of the Tenth International Conference on Fundamentals of Computation Theory* – FCT 1995, Springer LNCS 969, pp. 323–332, 1995.

[McNaughton 1961] Robert McNaughton. The theory of automata, a survey. *Advances in Computers,* vol. 2, pp. 379–421, 1961.

[Rose 1960] Gene F. Rose. An extended notion of computability. *Abstracts of International Congress for Logic, Methodology and Philosophy of Science,* p.14, 1960.

[Tantau 2002] Till Tantau. Comparing verboseness for finite automata and Turing machines. *Proceedings of the Nineteenth International Symposium on Theoretical Aspects of Computer Science*, STACS 2002, Proceedings. *Springer LNCS* 2285:465–476, 2002.

[Trakhtenbrot 1964] Boris A. Trakhtenbrot. On the frequency computation of functions. *Algebra i Logika,* vol. 2, pp.25–32, 1964 (Russian).