Branching Programs for Tree Evaluation

Mark Braverman[1], Stephen Cook[2], Pierre McKenzie[3], Rahul Santhanam[4], and Dustin Wehr[2]

[1] Microsoft Research
[2] University of Toronto
[3] Université de Montréal
[4] University of Edinburgh

Abstract. The problem $FT_d^h(k)$ consists in computing the value in $[k] = \{1, \ldots, k\}$ taken by the root of a balanced d-ary tree of height h whose internal nodes are labelled with d-ary functions on $[k]$ and whose leaves are labelled with elements of $[k]$. We propose $FT_d^h(k)$ as a good candidate for witnessing $\mathbf{L} \subsetneq \mathbf{LogDCFL}$. We observe that the latter would follow from a proof that k-way branching programs solving $FT_d^h(k)$ require $\Omega(k^{\text{unbounded function}(h)})$ size. We introduce a "state sequence" method that can match the size lower bounds on $FT_d^h(k)$ obtained by the Nečiporuk method and can yield slightly better (yet still subquadratic) bounds for some nonboolean functions. Both methods yield the tight bounds $\Theta(k^3)$ and $\Theta(k^{5/2})$ for deterministic and nondeterministic branching programs solving $FT_2^3(k)$ respectively. We propose as a challenge to break the quadratic barrier inherent in the Nečiporuk method by adapting the state sequence method to handle $FT_d^4(k)$.

1 Introduction

Let T_d^h be the balanced d-ary ordered tree T_d^h of height h, where we take *height* to mean the number of levels in the tree and we number the nodes as suggested by the heap data structure. Thus the root is node 1, and in general the children of node i are (when $d = 2$) nodes $2i, 2i + 1$ (see Figure 1). For every $d, h, k \geq 2$ we define the *Tree Evaluation* problem and its associated decision problem:

Definition 1.1 **($FT_d^h(k)$ and $BT_d^h(k)$)**
Given: T_d^h with each non-leaf node i independently labelled with a function $f_i : [k]^d \to [k]$ and each leaf node independently labelled with an element from $[k]$.
Function evaluation problem $FT_d^h(k)$*: Compute the value $v_1 \in [k]$ of the root 1 of T_d^h, where in general $v_i = a$ if i is a leaf labelled a, and $v_i = f_i(v_{j_1}, \ldots, v_{j_d})$ if $j_1, \ldots, j_d$ are the children of i.*
Boolean evaluation problem $BT_d^h(k)$*: Decide whether $v_1 = 1$.*

In the context of uniform complexity measures such as Turing machine space we rewrite $FT_d^h(k)$ and $BT_d^h(k)$ as $FT_d(h, k)$ and $BT_d(h, k)$ to indicate that d is fixed but h, k are input parameters. It is not hard to show that for each $d \geq 2$ a deterministic logspace-bounded poly-time auxiliary pushdown automaton solves

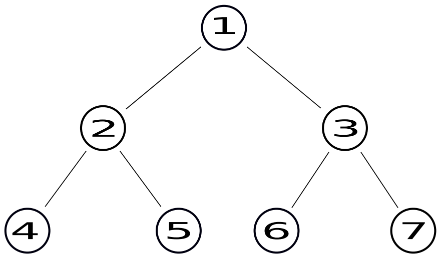

Fig. 1. A height 3 binary tree T_2^3 with nodes numbered heap style.

$BT_d(h, k)$, implying by [Sud78] that $BT_d(h, k)$ belongs to the class **LogDCFL** of languages logspace reducible to a deterministic context-free language. We know $\mathbf{L} \subseteq \mathbf{LogDCFL} \subseteq \mathbf{P}$ (see [Mah07] for up to date information on **LogDCFL**). The special case $BT_d(h, 2)$ was investigated under a different name in [KRW95] as part of an attempt to separate $\mathbf{NC}^1$ from $\mathbf{NC}^2$. In this paper, we suggest investigating the space complexity of $BT_d(h, k)$ and $FT_d(h, k)$.

We choose to study the Tree Evaluation problem as a particularly interesting candidate for non-membership in **L** or **NL** (deterministic or nondeterministic log space) because pebble games on trees provide natural space bounded algorithms for solving it: Black pebbling provides deterministic algorithms and, though we do not consider these in this paper, black-white pebbling provides nondeterministic algorithms. We choose k-way branching programs (BPs) as our model of Turing machine because the inputs to our problems are tuples of numbers in $[k]$.

For fixed d, h we are interested in how the size (number of states) of BPs solving $FT_d^h(k)$ and $BT_d^h(k)$ grows with k. One of our contributions is an alternative approach to Nečiporuk's lower bound method [Neč66] for this size. Applied to the problem $BT_d^h(k)$, our "state sequence" approach does as well as (but, so far, no better than) Nečiporuk's method. On the other hand, our approach does not suffer in principle from the quadratic limitation inherent in Nečiporuk's method. Hence there is hope that the approach can be extended. The current bottleneck stands at height 4. Proving our conjectured lower bound of $\Omega(k^7/\lg k)$ (writing lg for $\log_2$) for the size of deterministic BPs solving $BT_3^4(k)$ would constitute a breakthrough and would overcome the n^2 Nečiporuk limitation. However we do not yet know how to do this.

The more specific contributions of this paper are the following:

- we observe that for any $d \geq 2$ and unbounded $r(h)$, a lower bound of the form $\Omega(k^{r(h)})$ on the size of BPs solving $FT_d^h(k)$ would prove $BT_d(h, k) \notin \mathbf{L}$;
- we prove tight black pebbling bounds for T_d^h and transfer the upper bounds to size upper bounds of the form $k^{O(h)}$ for deterministic k-way BPs for $FT_d^h(k)$ and $BT_d^h(k)$;
- we prove tight size bounds of $\Theta(k^{2d-1})$ and $\Theta(k^{2d-1}/\lg k)$ for deterministic k-way BPs solving $FT_d^3(k)$ and $BT_d^3(k)$ respectively;
- we prove tight size bounds of $\Theta(k^{3d/2-1/2})$ for nondeterministic k-way BPs solving $BT_d^3(k)$; the argument yields an $\Omega(n^{3/2}/(\lg n)^{3/2})$ bound for the

number of *states* in nondeterministic binary BPs of arbitrary outdegree when n is the input length; a comparable bound of $\Omega(n^{3/2})$ was known but the latter only applied to the number of *edges* [Pud87,Raz91] in such BPs;
- we give examples of functions, such as the restriction $SumMod_2^3(k)$ of $FT_2^3(k)$ in which the root function is fixed to the sum modulo k, and the function $Children_2^4(k)$ which is required to simultaneously compute the root values of two instances of $FT_2^3(k)$, for which the state sequence method yields a better k-way BP size lower bound than a direct application of Nečiporuk's method ($\Omega(k^3)$ versus $\Omega(k^2)$ for $SumMod_2^3(k)$, and $\Omega(k^4)$ versus $\Omega(k^3)$ for $Children_2^4(k)$).

Section 2 defines branching programs and pebbling. Section 3 relates pebbling and branching programs to Turing machine space, and proves the pebbling bounds exploited in Section 4 to prove BP size upper bounds. BP lower bounds obtained using the Nečiporuk method are stated in Subsection 4.1. Our state sequence method is introduced in Subsection 4.2. The proofs left out of this abstract will appear in the full version of the paper.

2 Preliminaries

We assume some familiarity with complexity theory, such as can be found in [Gol08]. We write $[k]$ for $\{1, 2, \ldots, k\}$ and let $k \geq 2$.
Warning: Recall that the *height* of a tree is the number of levels in the tree, as opposed to the distance from root to leaf. Thus T_2^2 has just 3 nodes.

2.1 Branching programs

Many variants of the branching program model have been studied [Raz91,Weg00]. Our definition below is inspired by Wegener [Weg00, p. 239], by the k-way branching program of Borodin and Cook [BC82] and by its nondeterministic variant [BRS93,GKM08]. We depart from the latter however in two ways: nondeterministic branching program labels are attached to states rather than edges (because we think of branching program states as Turing machine configurations) and cycles in branching programs are allowed (because our lower bounds apply to this more powerful model).

Definition 2.1 (Branching programs) *A* nondeterministic k-way branching program B *computing a total function* $g : [k]^m \to R$*, where* R *is a finite set, is a directed rooted multi-graph whose nodes are called* states*. Every edge has a label from* $[k]$*. Every state has a label from* $[m]$*, except* $|R|$ final *sink states consecutively labelled with the elements from* R*. An input* $(x_1, \ldots, x_m) \in [k]^m$ *activates, for each* $1 \leq j \leq m$*, every edge labelled* x_j *out of every state labelled* j*. A* computation *on input* $\vec{x} = (x_1, \ldots, x_m) \in [k]^m$ *is a directed path consisting of edges activated by* $\vec{x}$ *which begins with the unique start state (the root), and either it is infinite, or it ends in the final state labelled* $g(x_1, \ldots, x_m)$*, or it ends* *in a nonfinal state labelled j with no outedge labelled x_j (in which case we say the computation* aborts*). At least one such computation must end in a final state. The* size *of B is its number of states. B is* deterministic k-way *if every non-final state has precisely k outedges labelled $1,\ldots,k$. B is* binary *if $k=2$.*

We say that B solves a decision problem (relation) if it computes the characteristic function of the relation.

A k-way branching program computing the function $FT_d^h(k)$ requires k^d k-ary arguments for each internal node i of T_d^h in order to specify the function f_i, together with one k-ary argument for each leaf. Thus in the notation of Definition 1.1 $FT_d^h(k)$: $[k]^m \to R$ where $R=[k]$ and $m=\frac{d^{h-1}-1}{d-1}\cdot k^d + d^{h-1}$. Also $BT_d^h(k)$: $[k]^m \to \{0,1\}$.

We define $\#\mathsf{detFstates}_d^h(k)$ (resp. $\#\mathsf{ndetFstates}_d^h(k)$) to be the minimum number of states required for a deterministic (resp. nondeterministic) k-way branching program to solve $FT_d^h(k)$. Similarly, $\#\mathsf{detBstates}_d^h(k)$ and $\#\mathsf{ndetBstates}_d^h(k)$ denote the number of states for solving $BT_d^h(k)$.

The next lemma is easy to prove and shows that the function problem is not much harder to solve than the Boolean problem.

Lemma 2.2 $\#\mathsf{detBstates}_d^h(k) \le \#\mathsf{detFstates}_d^h(k) \le k\cdot\#\mathsf{detBstates}_d^h(k)$ *and* $\#\mathsf{ndetBstates}_d^h(k) \le \#\mathsf{ndetFstates}_d^h(k) \le k\cdot\#\mathsf{ndetBstates}_d^h(k)$.

2.2 Pebbling

The pebbling game for dags was defined by Paterson and Hewitt [PH70] and was used as an abstraction for deterministic Turing machine space in [Coo74]. Black-white pebbling was introduced in [CS76] as an abstraction of nondeterministic Turing machine space (see [Nor09] for a recent survey).

We will only make use of a simple 'black pebbling' game in this paper. Here a pebble can be placed on any leaf node, and in general if all children of a node i have pebbles, then one of the pebbles on the children can be moved to i (this is a "sliding" move). The goal is to pebble the root. A *pebbling* of a tree T using p pebbles is any sequence of pebbling moves on nodes of T which starts and ends with no pebbles, and at some point the root is pebbled, and no configuration has more than p pebbles.

We allow "sliding moves" as above (as opposed to placing a new pebble on node i) because we want pebbling algorithms for trees to closely correspond to k-way branching program algorithms for the tree evaluation problem.

We use $\#\mathsf{pebbles}(T)$ to denote the minimum number of pebbles required to pebble T. The following result is proved easily using standard techniques.

Theorem 2.3. *For every $d,h\ge 2$, $\#\mathsf{pebbles}(T_d^h) = (d-1)h-d+2$.*

3 Connecting TMs, BPs, and Pebbling

Let $FT_d(h,k)$ be the same as $FT_d^h(k)$ except now the inputs vary with both h and k, and we assume the input to $FT_d(h,k)$ is a binary string X which codes

h and k and codes each node function f_i for the tree T_d^h by a sequence of k^d binary numbers and each leaf value by a binary number in $[k]$, so X has length

$$|X| = \Theta(d^h k^d \lg k) \tag{1}$$

The output is a binary number in $[k]$ giving the value of the root. The problem $BT_d(h,k)$ is the Boolean version of $FT_d(h,k)$: The input is the same, and the instance is true iff the value of the root is 1.

Obviously $BT_d(h,k)$ and $FT_d(h,k)$ can be solved in polynomial time, but we can prove a stronger result.

Theorem 3.1. *For each $d \geq 2$ the problem $BT_d(h,k)$ is in* **LogDCFL**.

The best known upper bounds on the number of states required by a BP to solve $FT_d^h(k)$ grow as $k^{\Omega(h)}$. The next result shows (Corollary 3.3) that any provable nontrivial dependency on h, for the power of k expressing the minimum number of such states, would separate **L**, and perhaps **NL** (deterministic and nondeterministic log space), from **LogDCFL**.

Theorem 3.2. *For each $d \geq 2$, if $BT_d(h,k)$ is in* **L** *(resp.* **NL***) then there is a constant ω_d and a function $c_d(h)$ such that $\#\mathsf{detFstates}_d^h(k) \leq c_d(h)k^{\omega_d}$ (resp. $\#\mathsf{ndetFstates}_d^h(k) \leq c_d(h)k^{\omega_d}$) for all $h, k \geq 2$.*

Proof. By Lemma 2.2, arguing for $\#\mathsf{detBstates}_d^h(k)$ and $\#\mathsf{ndetBstates}_d^h(k)$ instead of $\#\mathsf{detFstates}_d^h(k)$ and $\#\mathsf{ndetFstates}_d^h(k)$ suffices. In general a Turing machine which can enter at most C different configurations on all inputs of a given length n can be simulated (for inputs of length n) by a binary (and hence k-ary) branching program with C states. Each Turing machine using space $O(\lg n)$ has at most n^c possible configurations on any input of length $n \geq 2$, for some constant c. By (1) the input for $BT_d(h,k)$ has length $n = \Theta(d^h k^d \lg k)$, so there are at most $(d^h k^d \lg k)^{c'}$ possible configurations for a log space Turing machine solving $BT_d(h,k)$, for some constant c'. So we can take $c_d(h) = d^{c'h}$ and $\omega_d = c'(d+1)$. $\square$

Corollary 3.3 *Fix $d \geq 2$ and any unbounded function $r(h)$. If $\#\mathsf{detFstates}_d^h(k)$ (resp. $\#\mathsf{ndetFstates}_d^h(k)$) $\in \Omega(k^{r(h)})$ then $BT_d(h,k) \notin$* **L** *(resp. $\notin$* **NL***).*

The next result connects pebbling upper bounds with BP upper bounds.

Theorem 3.4. *If T_d^h can be pebbled with p pebbles, then deterministic branching programs with $O(k^p)$ states can solve $FT_d^h(k)$ and $BT_d^h(k)$.*

Corollary 3.5 $\#\mathsf{detFstates}_d^h(k) = O(k^{\#\mathsf{pebbles}(T_d^h)})$.

4 Branching Program Bounds

In this section we prove upper bounds for the number of states required for deterministic k-way branching programs to solve both the function problems $FT_d^h(k)$ and the Boolean problems $BT_d^h(k)$. The upper bounds for the function problems come from pebbling via Theorem 3.4. We prove that these bounds are tight for all trees of heights 2 and 3, and it seems plausible that they might be tight for all trees T_d^h: a proof would separate **L** from **LogDCFL** (see Section 3).

For the Boolean problems the deterministic upper bounds can be improved by a factor of $\lg k$ over the function problems for height $h \geq 3$ (the improvement is tight for height 3).

We also prove tight bounds for nondeterministic BPs solving the Boolean problems for heights 2 and 3.

Theorem 4.1 (BP Upper Bounds).

$$\#\mathsf{detFstates}_d^h(k) = O(k^{(d-1)h-d+2}) \tag{2}$$

$$\#\mathsf{detBstates}_d^h(k) = O(k^{(d-1)h-d+2}/\lg k), \text{ \textit{for} } h \geq 3 \tag{3}$$

$$\#\mathsf{ndetBstates}_2^3(k) = O(k^{5/2}) \tag{4}$$

We can combine the above upper bounds with the Nečiporuk lower bounds in Subsection 4.1, Figure 2, to obtain the tight bounds in the next theorem for height 3 trees. (The optimal bounds for all cases for height 2 trees are given by the size of the input: $\Theta(k^d)$.)

Corollary 4.2 (Height 3 trees)

$$\#\mathsf{detFstates}_d^3(k) = \Theta(k^{2d-1})$$
$$\#\mathsf{detBstates}_d^3(k) = \Theta(k^{2d-1}/\lg k)$$
$$\#\mathsf{ndetBstates}_2^3(k) = \Theta(k^{5/2})$$

4.1 The Nečiporuk method

The Nečiporuk method still yields the strongest explicit binary branching program size lower bounds known today, namely $\Omega(\frac{n^2}{(\lg n)^2})$ for deterministic [Neč66] and $\Omega(\frac{n^{3/2}}{\lg n})$ for nondeterministic (albeit for a weaker nondeterministic model in which states have bounded outdegree [Pud87], see [Raz91]).

By *applying the Nečiporuk method* to a k-way branching program B computing a function $f : [k]^m \to R$, we mean the following well known steps [Neč66]:

1. Upper bound the number $N(s, v)$ of (syntactically) distinct branching programs of type B having s non-final states, each labelled by one of v variables.
2. Pick a partition $\{V_1, \ldots, V_p\}$ of $[m]$.
3. For $1 \leq i \leq p$, lower bound the number $r_{V_i}(f)$ of restrictions $f_{V_i} : [k]^{|V_i|} \to R$ of f obtainable by fixing values of the variables in $[m] \setminus V_i$.
4. Then $\text{size}(B) \geq |R| + \sum_{1 \leq i \leq p} s_i$, where $s_i = \min\{\ s : N(s, |V_i|) \geq r_{V_i}(f)\ \}$.

Model	Lower bound for $FT_d^h(k)$	Lower bound for $BT_d^h(k)$
Deterministic k-way branching program	$\boxed{\frac{d^{h-2}-1}{4(d-1)^2} \cdot k^{2d-1}}$	$\boxed{\frac{d^{h-2}-1}{3(d-1)^2} \cdot \frac{k^{2d-1}}{\lg k}}$
Deterministic binary branching program	$\frac{d^{h-2}-1}{5(d-1)^2} \cdot k^{2d} = \Omega(n^2/(\lg n)^2)$	$\frac{d^{h-2}-1}{4d(d-1)} \cdot \frac{k^{2d}}{\lg k} = \Omega(n^2/(\lg n)^3)$
Nondeterministic k-way BP	$\frac{d^{h-2}-1}{2d-2} \cdot k^{\frac{3d}{2}-\frac{1}{2}}\sqrt{\lg k}$	$\boxed{\frac{d^{h-2}-1}{2d-2} \cdot k^{\frac{3d}{2}-\frac{1}{2}}}$
Nondeterministic binary BP	$\frac{d^{h-2}-1}{2d-2} \cdot k^{\frac{3d}{2}}\sqrt{\lg k} = \Omega(n^{3/2}/\lg n)$	$\frac{d^{h-2}-1}{2d-2} \cdot k^{\frac{3d}{2}} = \Omega(n^{3/2}/(\lg n)^{3/2})$

Fig. 2. Size bounds, expressed in terms of $n = \Theta(k^d \lg k)$ in the binary cases, obtained by applying the Nečiporuk method. Rectangles indicate optimality in k when $h = 3$ (Cor. 4.2). Improving any entry to $\Omega(k^{\text{unbounded } f(h)})$ would prove $\mathbf{L} \subsetneq \mathbf{P}$ (Cor. 3.3).

Theorem 4.3. *Applying the Nečiporuk method yields Figure 2.*

Remark 4.4. The $\Omega(n^{3/2}/(\lg n)^{3/2})$ binary nondeterministic BP lower bound for the $BT_d^h(k)$ problem and in particular for $BT_2^3(k)$ applies to the number of *states* when these can have arbitrary outdegree. This seems to improve on the best known former bound of $\Omega(n^{3/2}/\lg n)$, slightly larger but obtained for the weaker model in which states have bounded degree, or equivalently, for the switching and rectifier network model in which size is defined as the number of edges [Pud87,Raz91].

Let $Children_d^h(k)$ have the same input as $FT_d^h(k)$ with the exception that the root function is deleted. The output is the tuple $(v_2, v_3, \ldots, v_{d+1})$ of values for the children of the root.

Theorem 4.5. *For any $d, h \geq 2$, the best k-way deterministic BP size lower bound attainable for $Children_d^h(k)$ by applying the Nečiporuk method is $\Omega(k^{2d-1})$.*

Proof. The function $Children_d^h(k) : [k]^m \to R$ has $m = \Theta(k^d)$. Any partition $\{V_1, \ldots, V_p\}$ of the set of k-ary input variables thus has $p = O(k^d)$. Claim: for each i, the best attainable lower bound on the number of states querying variables from V_i is $O(k^{d-1})$.

Consider such a set V_i, $|V_i| = v \geq 1$. Here $|R| = k^d$, so the number $N_{\text{det}}^{k\text{-way}}(s, v)$ of distinct deterministic BPs having s non-final states querying variables from V_i satisfies

$$N_{\text{det}}^{k\text{-way}}(s, v) \geq 1^s \cdot (s + |R|)^{sk} \geq (1 + k^d)^{sk} \geq k^{dsk}.$$

Hence the estimate used in the Nečiporuk method to upper bound $N_{\text{det}}^{k\text{-way}}(s, v)$ will be at least k^{dsk}. On the other hand, the number of functions $f_{V_i} : [k]^v \to R$ obtained by fixing variables outside of V_i cannot exceed $k^{O(k^d)}$ since the number of variables outside V_i is $\Theta(k^d)$. Hence the best lower bound on the number of states querying variables from V_i obtained by applying the method will be no larger than the smallest s verifying $k^{ck^d} \leq k^{dsk}$ for some c depending on d and k. This proves our claim since then this number is at most $s = O(k^{d-1})$. □

Let $SumMod_d^h(k)$ have the same input as $FT_d^h(k)$ with the exception that the root function is preset to the sum modulo k. In other words the output is $v_2 + v_3 + \cdots + v_{d+1} \mod k$.

Theorem 4.6. *The best k-way deterministic BP size lower bound attainable for $SumMod_2^3(k)$ by applying the Nečiporuk method is $\Omega(k^2)$.*

4.2 The state sequence method

Here we give alternative proofs for some of the lower bounds given in Section 4.1. These proofs are more intricate than the Nečiporuk proofs but they do not suffer a priori from a quadratic limitation. The method also yields stronger lower bounds to $Children_2^4(k)$ and $SumMod_2^3(k)$ than those obtained by applying the Nečiporuk's method as expressed in Subsection 4.1 (see Theorems 4.5 and 4.6).

Theorem 4.7. $\#\mathsf{ndetBstates}_2^3(k) \geq k^{2.5}$ *for sufficiently large k.*

Proof. Consider an input I to $BT_2^3(k)$. We number the nodes in T_2^3 as in Figure 1, and let v_j^I denote the value of node j under input I. We say that a state in a computation on input I *learns* v_j^I if that state queries $f_j^I(v_{2j}^I, v_{2j+1}^I)$ (recall $2j, 2j+1$ are the children of node j).
Definition [Learning Interval] *Let B be a k-way nondeterministic BP that solves $BT_2^3(k)$. Let $\mathcal{C} = \gamma_0, \gamma_1, \cdots, \gamma_T$ be a computation of B on input I. We say that a state γ_i in the computation is* critical *if one or more of the following holds:*

1. *$i = 0$ or $i = T$*
2. *γ_i learns v_2^I and there is an earlier state which learns v_3^I with no intervening state that learns v_2^I.*
3. *γ_i learns v_3^I and no earlier state learns v_3^I unless an intervening state learns v_2^I.*

We say that a subsequence $\gamma_i, \gamma_{i+1}, \cdots \gamma_j$ is a learning interval *if γ_i and γ_j are consecutive critical states. The interval is* type 3 *if γ_i learns v_3^I, and otherwise the interval is* type 2.

Thus type 2 learning intervals begin with γ_0 or a state which learns v_2^I, and never learn v_3^I until the last state, and type 3 learning intervals begin with a state which learns v_3^I and never learn v_2^I until the last state.

Now let B be as above, and for $j \in \{2, 3\}$ let Γ_j be the set of all states of B which query the input function f_j. We will prove the theorem by showing that for large k

$$|\Gamma_2| + |\Gamma_3| > k^2\sqrt{k}. \tag{5}$$

For $r, s \in [k]$ let $F^{r,s}_{yes}$ be the set of inputs I to B whose four leaves are labelled r, s, r, s respectively, whose middle node functions f^I_2 and f^I_3 are identically 0 except $f^I_2(r, s) = v^I_2$ and $f^I_3(r, s) = v^I_3$, and $f^I_1(v^I_2, v^I_3) = 1$ (so $v^I_1 = 1$). Thus each such I is a 'YES input', and should be accepted by B.

Note that each member I of $F^{r,s}_{yes}$ is uniquely specified by a triple

$$(v^I_2, v^I_3, f^I_1) \text{ where } f^I_1(v^I_2, v^I_3) = 1 \tag{6}$$

and hence $F^{r,s}_{yes}$ has exactly $k^2(2^{k^2-1})$ members.

For $j \in \{2, 3\}$ and $r, s \in [k]$ let $\Gamma^{r,s}_j$ be the subset of Γ_j consisting of those states which query $f_j(r, s)$. Then Γ_j is the disjoint union of $\Gamma^{r,s}_j$ over all pairs (r, s) in $[k] \times [k]$. Hence to prove (5) it suffices to show

$$|\Gamma^{r,s}_2| + |\Gamma^{r,s}_3| > \sqrt{k} \tag{7}$$

for large k and all r, s in $[k]$. We will show this by showing

$$(|\Gamma^{r,s}_2| + 1)(|\Gamma^{r,s}_3| + 1) \geq k/2 \tag{8}$$

for all $k \geq 2$. (Note that given the product, the sum is minimized when the summands are equal.)

For each input I in $F^{r,s}_{yes}$ we associate a fixed accepting computation $\mathcal{C}(I)$ of B on input I.

Now fix $r, s \in [k]$. For $a, b \in [k]$ and $f : [k] \times [k] \to \{0, 1\}$ with $f(a, b) = 1$ we use (a, b, f) to denote the input I in $F^{r,s}_{yes}$ it represents as in (6).

To prove (8), the idea is that if it is false, then as I varies through all inputs (a, b, f) in $F^{r,s}_{yes}$ there are too few states learning $v^I_2 = a$ and $v^I_3 = b$ to verify that $f(a, b) = 1$. Specifically, we can find a, b, f, g such that $f(a, b) = 1$ and $g(a, b) = 0$, and by cutting and pasting the accepting computation $\mathcal{C}(a, b, f)$ with accepting computations of the form $\mathcal{C}(a, b', g)$ and $\mathcal{C}(a', b, g)$ we can construct an accepting computation of the 'NO input' (a, b, g).

We may assume that the branching program B has a unique initial state γ_0 and a unique accepting state δ_{ACC}.

For $j \in \{2, 3\}$, $a, b \in [k]$ and $f : [k] \times [k] \to \{0, 1\}$ with $f(a, b) = 1$ define $\varphi_j(a, b, f)$ to be the set of all state pairs (γ, δ) such that there is a type j learning interval in $\mathcal{C}(a, b, f)$ which begins with γ and ends with δ. Note that if $j = 2$ then $\gamma \in (\Gamma^{r,s}_2 \cup \{\gamma_0\})$ and $\delta \in (\Gamma^{r,s}_3 \cup \{\delta_{ACC}\})$, and if $j = 3$ then $\gamma \in \Gamma^{r,s}_3$ and $\delta \in (\Gamma^{r,s}_2 \cup \{\delta_{ACC}\})$

To complete the definition, define $\varphi_j(a, b, f) = \varnothing$ if $f(a, b) = 0$.

For $j \in \{2, 3\}$ and $f : [k] \times [k] \to \{0, 1\}$ we define a function $\varphi_j[f]$ from $[k]$ to sets of state pairs as follows:

$$\varphi_2[f](a) = \bigcup_{b \in [k]} \varphi_2(a, b, f) \ \subseteq S_2$$

$$\varphi_3[f](b) = \bigcup_{a \in [k]} \varphi_3(a, b, f) \ \subseteq S_3$$

where $S_2 = (\varGamma_2^{r,s} \cup \{\gamma_0\}) \times (\varGamma_3^{r,s} \cup \{\delta_{ACC}\})$ and $S_3 = \varGamma_3^{r,s} \times (\varGamma_2^{r,s} \cup \{\delta_{ACC}\})$.

For each f the function $\varphi_j[f]$ can be specified by listing a k-tuple of subsets of S_j, and hence there are at most $2^{k|S_j|}$ distinct such functions as f ranges over the 2^{k^2} Boolean functions on $[k] \times [k]$, and hence there are at most $2^{k(|S_2|+|S_3|)}$ pairs of functions $(\varphi_2[f], \varphi_3[f])$. If we assume that (8) is false, we have $|S_2| + |S_3| < k$. Hence by the pigeonhole principle there must exist distinct Boolean functions f, g such that $\varphi_2[f] = \varphi_2[g]$ and $\varphi_3[f] = \varphi_3[g]$.

Since f and g are distinct we may assume that there exist a, b such that $f(a, b) = 1$ and $g(a, b) = 0$. Since $\varphi_2[f](a) = \varphi_2[g](a)$, if (γ, δ) are the endpoints of a type 2 learning interval in $\mathcal{C}(a, b, f)$ there exists b' such that (γ, δ) are the endpoints of a type 2 learning interval in $\mathcal{C}(a, b', g)$ (and hence $g(a, b') = 1$). Similarly, if (γ, δ) are endpoints of a type 3 learning interval in $\mathcal{C}(a, b, f)$ there exists a' such that (γ, δ) are the endpoints of a type 3 learning interval in $\mathcal{C}(a', b, f)$.

Now we can construct an accepting computation for the 'NO input' (a, b, g) from $\mathcal{C}(a, b, f)$ by replacing each learning interval beginning with some γ and ending with some δ by the corresponding learning interval in $\mathcal{C}(a, b', g)$ or $\mathcal{C}(a', b, g)$. (The new accepting computation has the same sequence of critical states as $\mathcal{C}(a, b, f)$.) This works because a type 2 learning interval never queries v_3 and a type 3 learning interval never queries v_2.

This completes the proof of (8) and the theorem. $\square$

Theorem 4.8. *Every deterministic branching program that solves $BT_2^3(k)$ has at least $k^3/\lg k$ states for sufficiently large k.*

Proof. We modify the proof of Theorem 4.7. Let B be a deterministic BP which solves $BT_2^3(k)$, and for $j \in \{2, 3\}$ let $\varGamma_j$ be the set of states in B which query f_j (as before). It suffices to show that for sufficiently large k

$$|\varGamma_2| + |\varGamma_3| \geq k^3/\lg k. \tag{9}$$

For $r, s \in [k]$ we define the set $F^{r,s}$ to be the same as $F_{yes}^{r,s}$ except that we remove the restriction on f_1^I. Hence there are exactly $k^2 2^{k^2}$ inputs in $F^{r,s}$.

As before, for $j \in \{2, 3\}$, $\varGamma_j$ is the disjoint union of $\varGamma^{r,s}$ for $r, s \in [k]$. Thus to prove (9) it suffices to show that for sufficiently large k and all r, s in $[k]$

$$|\varGamma_2^{r,s}| + |\varGamma_3^{r,s}| \geq k/\lg_2 k. \tag{10}$$

We may assume there are unique start, accepting, and rejecting states γ_0, δ_{ACC}, δ_{REJ}. Fix $r, s \in [k]$.

For each root function $f : [k] \times [k] \to \{0, 1\}$ we define the functions

$$\psi_2[f] : [k] \times (\varGamma_2^{r,s} \cup \{\gamma_0\}) \to (\varGamma_3^{r,s} \cup \{\delta_{ACC}, \delta_{REJ}\})$$
$$\psi_3[f] : [k] \times \varGamma_3^{r,s} \to (\varGamma_2^{r,s} \cup \{\delta_{ACC}, \delta_{REJ}\})$$

by $\psi_2[f](a, \gamma) = \delta$ if δ is the next critical state after γ in a computation with input (a, b, f) (this is independent of b), or $\delta = \delta_{REJ}$ if there is no such critical state.

Similarly $\psi_3[f](b,\delta) = \gamma$ if γ is the next critical state after δ in a computation with input (a,b,f) (this is independent of a), or $\delta = \delta_{REJ}$ if there is no such critical state.

CLAIM: The pair of functions $(\psi_2[f], \psi_3[f])$ is distinct for distinct f.

For suppose otherwise. Then there are f, g such that $\psi_2[f] = \psi_2[g]$ and $\psi_3[f] = \psi_3[g]$ but $f(a,b) \neq g(a,b)$ for some a, b. But then the sequences of critical states in the two computations $C(a,b,f)$ and $C(a,b,g)$ must be the same, and hence the computations either accept both (a,b,f) and (a,b,g) or reject both. So the computations cannot both be correct.

Finally we prove (10) from the CLAIM. Let $s_2 = |\Gamma_2^{r,s}|$ and let $s_3 = |\Gamma_3^{r,s}|$, and let $s = s_2 + s_3$. Then the number of distinct pairs (ψ_2, ψ_3) is at most

$$(s_3+2)^{k(s_2+1)}(s_2+2)^{ks_3} \leq (s+2)^{k(s+1)}$$

and since there are 2^{k^2} functions f we have

$$2^{k^2} \leq (s+2)^{k(s+1)}$$

so taking logs, $k^2 \leq k(s+1)\lg_2(s+2)$ so $k/\lg_2(s+2) \leq s+1$, and (10) follows. $\square$

Recall from Theorem 4.5 that applying the Nečiporuk method to $Children_2^4(k)$ yields an $\Omega(k^3)$ size lower bound and from Theorem 4.6 that applying it to $SumMod_2^3(k)$ yields $\Omega(k^2)$. The state sequence method also proves the next two theorems.

Theorem 4.9. *Any deterministic k-way BP for $Children_2^4(k)$ has at least $k^4/2$ states.*

Theorem 4.10. *Any deterministic k-way BP for $SumMod_2^3(k)$ requires at least k^3 states.*

5 Conclusion

Our main open question is whether we can adapt the state sequence method to break the $\Omega(n^2)$ barrier for the size of deterministic branching programs. In particular, can the method be extended to handle trees of height 4? Specifically, can we prove a lower bound of $\Omega(k^7/\lg k)$ for $BT_3^4(k)$ (see Theorem 4.1)?

Another question arises from the $O(k^{5/2})$ upper bound from Theorem 4.1. Is there a pebbling to justify such a non-integral exponent? As it turns out, the answer is yes. One can introduce fractional black-white pebbling and develop an interesting theory. Our work on that issue will be the subject of another paper.

Acknowledgment James Cook played a helpful role in the early parts of this research. The second author is grateful to Michael Taitslin for suggesting a version of the tree evaluation problem in which the nodes are labelled by fixed quasi groups (see [Tai05]).

References

[BC82] A. Borodin and S. Cook. A time-space tradeoff for sorting on a general sequential model of computation. *SIAM J. Comput.*, 11(2):287–297, 1982.

[BRS93] A. Borodin, A. Razborov, and R. Smolensky. On lower bounds for read-k-times branching programs. *Computational Complexity*, 3:1–18, 1993.

[Coo74] S. Cook. An observation on time-storage trade off. *J. Comput. Syst. Sci.*, 9(3):308–316, 1974.

[CS76] S. Cook and R. Sethi. Storage requirements for deterministic polynomial time recognizable languages. *J. Comput. Syst. Sci.*, 13(1):25–37, 1976.

[GKM08] A. Gál, M. Koucký, and P. McKenzie. Incremental branching programs. *Theory Comput. Syst.*, 43(2):159–184, 2008.

[Gol08] O. Goldreich. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press, 2008.

[KRW95] M. Karchmer, R. Raz, and A. Wigderson. Super-logarithmic depth lower bounds via direct sum in communication complexity. *Computational Complexity*, 5:191–204, 1995. An abstract appeared in the *6th Structure in Complexity Theory Conference (1991)*.

[Mah07] M. Mahajan. Polynomial size log depth circuits: between $\mathbf{NC}^1$and $\mathbf{AC}^1$. *Bulletin of the EATCS*, 91:30–42, February 2007.

[Neč66] È. Nečiporuk. On a boolean function. *Doklady of the Academy of the USSR*, 169(4):765–766, 1966. English translation in *Soviet Mathematics Doklady* 7:4, pp. 999-1000.

[Nor09] J. Nordström. New wine into old wineskins: A survey of some pebbling classics with supplemental results. Available on line at `http://people.csail.mit.edu/jakobn/research/`, 2009.

[PH70] M. Paterson and C. Hewitt. Comparative schematology. In *Record of Project MAC Conference on Concurrent Systems and Parallel Computations*, pages 119–128, 1970. (June 1970) ACM. New Jersey.

[Pud87] P. Pudlák. The hierarchy of boolean circuits. *Computers and artificial intelligence*, 6(5):449–468, 1987.

[Raz91] A. Razborov. Lower bounds for deterministic and nondeterministic branching programs. In *8th Internat. Symp. on Fundamentals of Computation Theory*, pages 47–60, 1991.

[Sud78] H. Sudborough. On the tape complexity of deterministic context-free languages. *J. ACM*, 25(3):405–414, 1978.

[Tai05] M.A. Taitslin. An example of a problem from PTIME and not in NLogSpace. In *Proceedings of Tver State University*, volume 6(12) of *Applied Mathematics, issue 2, Tver State University, Tver*, pages 5–22, 2005.

[Weg00] I. Wegener. *Branching Programs and Binary Decision Diagrams*. SIAM Monographs on Discrete Mathematics and Applications. Soc. for Industrial and Applied Mathematics, Philadelphia, 2000.