A sorting network in bounded arithmetic

Emil Jeřábek*

Institute of Mathematics of the Academy of Sciences
Žitná 25, 115 67 Praha 1, Czech Republic, email: jerabek@math.cas.cz

December 9, 2008

Abstract

We formalize the construction of Paterson's variant of the Ajtai–Komlós–Szemerédi sorting network of logarithmic depth in the bounded arithmetical theory VNC^1_* (an extension of VNC^1), under the assumption of existence of suitable expander graphs. We derive a conditional p-simulation of the propositional sequent calculus in the monotone sequent calculus MLK.

1 Introduction

Sorting is one of the most fundamental algorithmic operations, thus it is not surprising that much effort in theoretical computer science was invested in investigation of its computational complexity in various contexts. In particular, its exact parallel complexity was open for a long time. It has been known since the 1960s that it is fairly easy to construct parallel sorting algorithms using $O(\log^2 n)$ steps (Batcher [5]), but it proved quite difficult to further improve on this upper bound. It was only in 1983 when Ajtai, Komlós, and Szemerédi [1, 2] devised an ingenious algorithm achieving $O(\log n)$ parallel operations. The algorithm and its analysis were subsequently simplified by Paterson [11]. An important feature of the AKS algorithm is that the pattern of comparisons and swaps is fixed in advance independent of the data, hence the construction in fact gives a *sorting network* of depth $O(\log n)$. (This result is asymptotically optimal, as there is an obvious $\Omega(\log n)$ depth lower bound.) A sorting network is a structure consisting of comparators connected by wires, where a comparator is a device which takes two inputs and outputs them in sorted order.

In the present paper we are going to formalize the core of the AKS sorting network (or rather its version by Paterson) in the theory VNC^1_* of bounded arithmetic. More precisely, the basic building blocks of the AKS network, the so-called ε-halvers, are constructed using a certain kind of expander graphs. Construction of the expanders is a separate issue rather tangential to analysis of the main part of the network, we thus leave it out completely: we simply assume that VNC^1_* proves the existence of appropriate expanders, and all our results

*Supported by grant IAA1019401 of GA AV ČR, and grant 1M0545 of MŠMT ČR.

are conditional on this assumption. We note that some research towards formalization of expanders in bounded arithmetic is in progress [9].

There are several reasons why such a formalization is desirable. It is a basic problem in the development of bounded arithmetic to find what results in mathematics or computer science are provable in a given theory. In the other direction, the program of reverse mathematics seeks to find the minimal theory capable of proving a given statement. In particular, it is a natural foundational problem whether various properties of a given complexity class are provable using only concepts from the same class. Since the AKS network is a kind of a circuit of logarithmic depth, the natural class it fits into is (nonuniform) NC^1; it is thus reassuring to have a proof of its correctness in an NC^1-theory such as VNC^1_*.

The formalization has applications in propositional proof complexity. The monotone sequent calculus MLK is the fragment of the usual propositional sequent calculus LK using only sequents consisting of monotone formulas. Atserias et al. [4] have shown that MLK quasipolynomially simulates LK (with respect to monotone sequents), but it is an open problem whether one can give a polynomial simulation. It was also shown in [4] that it is sufficient for an affirmative answer to construct monotone formulas for threshold functions such that their basic properties have polynomial-size proofs in LK. Such monotone formulas can be obtained by evaluation of the AKS network on 0-1 inputs. Since VNC^1_* proves soundness of the network, and translates into polynomial LK-proofs, the properties of these formulas required by [4] indeed have polynomial LK-proofs. We thus obtain a p-simulation of LK by MLK under our basic assumption on formalizability of expanders in VNC^1_*.

There are other potential applications of the AKS network in bounded arithmetic. As shown in [6], the closure of the class *NL* under complement is provable in the bounded arithmetic for *NL*. However, it is not known whether we can formalize the closure of the related class *SL* under complement in an *SL*-theory. Formalization of the AKS network is the first step, as the network is involved in the proof of $SL = coSL$ from [10].

Our formalization is carried out in a not quite standard theory VNC^1_* introduced for this very purpose in [8]. This theory was chosen to satisfy two conflicting goals. On the one hand, the application to monotone sequent calculus described above requires that propositional translations of Π^B_1-formulas provable in the theory have polynomial-size proofs in LK, or equivalently, in Frege systems, hence we need some kind of an NC^1-theory. On the other hand, successful formalization of the AKS network requires at the very least that the theory proves that the network can be evaluated. We thus need the ability to evaluate (sufficiently uniformly described) circuits of logarithmic depth. The standard NC^1-theory VNC^1 is too weak for this purpose, as evaluation of log-depth circuits is not known to be possible in uniform NC^1 (i.e., $ALOGTIME$). VNC^1 can only evaluate log-depth circuits described by their extended connection language (see [13]), which is however not available for the AKS network.

The paper is organized as follows. Section 2 gives definitions of VNC^1_* and basic notions like comparator networks, as well as their elementary properties. In Section 3 we formally describe the AKS network. In Section 4 we carry out the analysis of the network in VNC^1_*, and in Section 5 we give a p-simulation of LK by MLK as an application.

2 Preliminaries

We will work with second-order (i.e., two-sorted) arithmetical theories as in [14, 7], but for convenience we include the function $|x| = \lceil \log_2(x+1) \rceil$ among the basic symbols. Our theories thus have two sorts of variables: numbers, denoted by lowercase letters, and finite sets or strings, denoted by uppercase letters. The basic language is $L_0 = \langle 0, \mathrm{s}, +, \cdot, |x|, \leq, \in, |X| \rangle$. The theory *BASIC* consists of the axioms

$$
\begin{array}{ll}
x + 0 = x & x + \mathrm{s}\, y = \mathrm{s}(x+y) \\
x \cdot 0 = 0 & x \cdot \mathrm{s}\, y = x \cdot y + x \\
\mathrm{s}\, y \leq x \to y < x & x \neq 0 \to \exists y\, x = \mathrm{s}\, y \\
x \in X \to x < |X| & \mathrm{s}\, x = |X| \to x \in X \\
|0| = 0 & x \neq 0 \to |x + x| = \mathrm{s}|x| \\
\forall x\, (x \in X \leftrightarrow x \in Y) \to X = Y & |\mathrm{s}(x+x)| = \mathrm{s}|x|
\end{array}
$$

where $x < y$ is an abbreviation for $x \leq y \wedge x \neq y$. We also write $X(x)$ for $x \in X$. We define the constants $1 = \mathrm{s}\,0$, $2 = \mathrm{s}\,\mathrm{s}\,0$, $3 = \mathrm{s}\,\mathrm{s}\,\mathrm{s}\,0$, $\ldots$, and we will often write $x+1$ for $\mathrm{s}\,x$ (the two expressions being equal by the *BASIC* axioms). We introduce the bounded quantifiers

$$
\begin{aligned}
\exists x \leq t\, \varphi &\Leftrightarrow \exists x\, (x \leq t \wedge \varphi), \\
\forall x \leq t\, \varphi &\Leftrightarrow \forall x\, (x \leq t \to \varphi), \\
\exists X \leq t\, \varphi &\Leftrightarrow \exists X\, (|X| \leq t \wedge \varphi), \\
\forall X \leq t\, \varphi &\Leftrightarrow \forall X\, (|X| \leq t \to \varphi),
\end{aligned}
$$

where t is a term not involving x or X (respectively), and similarly for strict inequalities. A formula is bounded if it uses only bounded quantifiers. A bounded L_0-formula without set quantifiers is called Σ^B_0 or Π^B_0. Inductively, Σ^B_{i+1} consists of formulas of the form

$$\exists X_1 \leq t_1 \ldots \exists X_n \leq t_n\, \varphi$$

for $\varphi \in \Pi^B_i$, and Π^B_{i+1} consists of formulas of the form

$$\forall X_1 \leq t_1 \ldots \forall X_n \leq t_n\, \varphi$$

for $\varphi \in \Sigma^B_i$. Note that we use Σ^B_i and Π^B_i to denote formulas of the basic language L_0 only. If we expand the definition to allow atomic formulas in a richer language L, we will call the corresponding classes $\Sigma^B_i(L)$ and $\Pi^B_i(L)$, respectively.

If Γ is a set of formulas, the Γ-comprehension axiom is the schema

$$(\Gamma\text{-}COMP) \qquad \exists X \leq x\, \forall u < x\, (u \in X \leftrightarrow \varphi),$$

where $\varphi \in \Gamma$ has no free occurrence of X. We define the theory V^0 as $BASIC + \Sigma^B_0\text{-}COMP$.

As our sorting network is a kind of a circuit of logarithmic depth, we will use theories VNC^1_* and $\overline{VNC}^1_*$ corresponding to "slightly nonuniform" NC^1 which allow evaluation of uniform families of log-depth circuits, introduced in [8]. Let $\varphi(d,x,y)$ be a formula, possibly with other free variables. We put

$$\begin{aligned}\varphi^*(d,x,y) &\Leftrightarrow \varphi(d,x,y) \land (\forall z < y\, \neg\varphi(d,x,z) \lor \forall z > y\, \neg\varphi(d,x,z)),\\ \mathrm{eval}(n,m,\varphi,I,Y) &\Leftrightarrow \forall x < n\, [(Y(0,x) \leftrightarrow I(x))\\ &\qquad \land \forall d < m\, (Y(d+1,x) \leftrightarrow ((2 \mid d \land \exists y < n\, (\varphi^*(d,x,y) \land Y(d,y)))\\ &\qquad\qquad \lor (2 \nmid d \land \forall y < n\, (\varphi^*(d,x,y) \to Y(d,y)))))],\end{aligned}$$

where $Y(d,x)$ stands for $dn+x \in Y$. (By abuse of notation, we include φ among the arguments of eval to indicate the dependence of eval on φ, even though φ is a formula, not a variable. Note that free variables of eval include parameters of φ, i.e., its free variables other than d,x,y.) The meaning of eval is that Y is the evaluation of a bounded fan-in monotone circuit described by φ on input I. The circuit consists of $m+1$ layers, each with n nodes. Nodes on layer 0 are truth constants given by I. Layers $d>0$ consist of alternating disjunction (odd d) and conjunction (even d) gates. Gates on level d can only use nodes on level $d-1$ as inputs. The formula $\varphi(d,x,y)$ means that node x on level $d+1$ uses node y on level d as input. The formula φ^* is actually employed instead of φ to force each gate to have at most two inputs.

We define VNC^1_* to be the closure of V^0 under the derivation rule

$$(\Delta^B_1\text{-}SCV) \qquad \frac{\varphi \leftrightarrow \neg\varphi'}{\exists Y \le (|m|+1)n\ \mathrm{eval}(n,|m|,\varphi,I,Y)},$$

where φ and φ' are Σ^B_1-formulas with no free set variables. (A Σ^B_1-formula provably equivalent to a Π^B_1-formula in a theory T is called a $\Delta^B_1(T)$-formula.)

The language $L_{\overline{VNC}^1_*}$ contains L_0, and a function symbol $C_\varphi(n,\vec{x},\vec{X})$ for each Σ^B_0-formula $\varphi(u,\vec{x},\vec{X})$ (with all free variables indicated). Moreover, it is closed under the following rule: for each open $L_{\overline{VNC}^1_*}$-formula $\varphi(\vec{p},d,x,y)$ without free set variables, we include a function symbol $Y_\varphi(\vec{p},n,m,I)$. We will usually denote $C_\varphi(n,\vec{x},\vec{X})$ by $\{u<n \mid \varphi(u,\vec{x},\vec{X})\}$.

$\overline{VNC}^1_*$ is a theory in $L_{\overline{VNC}^1_*}$ consisting of the axioms of $BASIC$, the axiom

$$(\Sigma^B_0\text{-}\overline{COMP}) \qquad u \in C_\varphi(n,\vec{x},\vec{X}) \leftrightarrow u < n \land \varphi(u,\vec{x},\vec{X})$$

for each Σ^B_0-formula $\varphi(u,\vec{x},\vec{X})$, and the axiom

$$(Open\text{-}\overline{SCV}) \qquad |Y_\varphi(\vec{p},n,m,I)| \le (|m|+1)n \land \mathrm{eval}(n,|m|,\varphi,I,Y_\varphi(\vec{p},n,m,I))$$

for each open $L_{\overline{VNC}^1_*}$-formula $\varphi(\vec{p},d,x,y)$.

As shown in [8], $\overline{VNC}^1_*$ is an open conservative extension of VNC^1_*, and $L_{\overline{VNC}^1_*}$-functions have Σ^B_1-definitions in VNC^1_*. For this reason, we will not distinguish the two theories, and we will work freely with $L_{\overline{VNC}^1_*}$-functions in VNC^1_*. Every $\Sigma^B_0(L_{\overline{VNC}^1_*})$-formula (or indeed, $\Delta^B_1(VNC^1_*)$-formula) is equivalent to an open $L_{\overline{VNC}^1_*}$-formula in VNC^1_*. We will denote these formulas simply as NC^1_*-formulas, and likewise, we will call functions given by $L_{\overline{VNC}^1_*}$-terms as NC^1_*-functions. VNC^1_* proves NC^1_*-$COMP$ and NC^1_*-IND, where Γ-IND is the usual induction schema

$$(\Gamma\text{-}IND) \qquad \varphi(0) \land \forall x\, (\varphi(x) \to \varphi(x+1)) \to \forall x\, \varphi(x)$$

for $\varphi \in \Gamma$. VNC^1_* contains VNC^1, and is contained in VL. The provably total computable functions of VNC^1_* are those definable by NC^1_*-functions in the standard model of arithmetic; this class fits between uniform NC^1 and L-uniform NC^1.

As $VNC^1_* \supseteq VNC^1 \supseteq VTC^0$, there is a well-behaved NC^1_*-function computing cardinality of sets. We will denote it $\operatorname{card} X$ in order to distinguish it from the basic symbol $|X|$ of L_0.

The Δ^B_1-SCV and $Open$-$\overline{SCV}$ axioms provide evaluation of a certain type of circuits, but they were designed to be formally simple rather than feature-rich. We may introduce a more elaborate setting for convenient evaluation of log-depth circuits. We can describe circuits using the following data:

- Numbers k, m, and s, where k is the number of inputs, m is the number of layers, and s is the size of each layer.
- A function $T\colon m \times s \to \{\ulcorner\vee\urcorner, \ulcorner\wedge\urcorner, \ulcorner\neg\urcorner\} \cup \{\ulcorner x_i \urcorner \mid i < k\}$ indicating the type of each node, where we put e.g. $\ulcorner\vee\urcorner = 0$, $\ulcorner\wedge\urcorner = 1$, $\ulcorner\neg\urcorner = 2$, and $\ulcorner x_i \urcorner = i + 3$, and we represent T by its graph, i.e., as a set $T \leq ms(k+3)$.
- A formula $\varphi(d, x, d', x')$ (possibly with other parameters) which states that node x' on layer d' is an input of gate x on layer d.

In order for a circuit to be well-formed, we demand that any gate uses only nodes on lower layers as inputs (but not necessarily from the adjacent layer), and all nodes have the correct number of inputs: 1 for negation nodes, 0 for input nodes, and at most 2 for conjunction and disjunction gates.

Lemma 2.1 ([8]) *(in VNC^1_*) A circuit described as above with φ an NC^1_*-formula without set parameters, and m bounded by some $|a|$, can be evaluated on any input string. Moreover, the evaluation is computable by an NC^1_*-function.* □

Definition 2.2 A *comparator network* on n inputs is a directed acyclic graph without duplicate edges with three types of vertices: *input nodes* with fan-in 0 and fan-out 1, *comparators* with fan-in 2 and fan-out 2, and *output nodes* with fan-in 1 and fan-out 0. Input and output nodes are labelled by numbers $k < n$, and there exists exactly one input node and one output node labelled k for every k. For each comparator, one of its outgoing edges is labelled h (higher) and the other one is labelled l (lower). The *size* of a network is the number of its comparators. We represent a comparator network N by a sequence $N = \{w_i \mid i < s\}$, where w_i describes the ith node of N: its type, adjacent nodes, and labels. We require the sequence to start with the input nodes and end with the output nodes, both ordered according to their labels. If there is an edge going from node i to node j, we further require $i < j$. The network has *depth* at most d, if we can partition the comparators of N into at most d blocks (called *layers*), such that each layer is contiguous in the sequence ordering, and there are no edges going between two nodes of the same layer.

Let $\vec{X} = \{X_k \mid k < n\}$ be a sequence of sets, and $\leq$ a total ordering whose domain includes every X_k. An *evaluation* of a network N with respect to $\leq$ on input $\vec{X}$ is a sequence of sets E_e indexed by edges e of N such that $E_e = X_k$ if e is the outgoing edge of an input node with label k, and if l and h are the lower and higher outgoing edges of a comparator with incoming edges e, f, then $E_l = \min_{\leq}\{E_e, E_f\}$, and $E_h = \max_{\leq}\{E_e, E_f\}$. The *result* of an evaluation E is the sequence of sets $\vec{Y} = \{Y_k \mid k < n\}$ such that $Y_k = E_e$, where e is the incoming edge of the output node with label k. We write $\vec{Y} = \mathrm{eval}(N, \leq, \vec{X})$ (the context should suffice to disambiguate between this notation and the eval-formula from the definition of VNC^1_*).

Since comparators have the same number of incoming and outgoing edges, there are exactly n edges at any section of a network with n inputs. That is, if $N = \{w_i \mid i < s\}$ is a network with n inputs, and $i < s$ is a comparator node, then we can show by straightforward induction on i that there are n edges going from nodes $j \leq i$ to nodes $j > i$. Consequently, each layer has size at most $n/2$, and a network of depth d has size at most $nd/2$.

A comparator network of logarithmic depth resembles an NC^1-circuit. Indeed, if we want to evaluate a uniformly described network on a 0-1 input, we can replace each comparator by a pair of $\wedge$ and $\vee$ gates (i.e., min and max in the Boolean domain), turning it into a logarithmic depth bounded fan-in circuit, which can be evaluated in VNC^1_*. This argument does not work for nonconstant domains, as we then cannot compute the required comparisons by bounded depth bounded fan-in circuits. Nevertheless, we will show that we can evaluate a log-depth network on arbitrary inputs in VNC^1_* using a simple trick based on a variant of the 0-1 principle.

Lemma 2.3 (*in* VNC^1_*) *Let N be a comparator network on n inputs of depth $d \leq \log m$ for some m defined by an NC^1_*-formula without set parameters, $\leq$ a total ordering defined by an NC^1_*-formula, and $\{X_k \mid k < n\}$ a sequence of sets in the domain of $\leq$. Then there exists a unique evaluation of N on input $\vec{X}$ with respect to $\leq$.*

Proof: Uniqueness: if E and E' are two evaluations of $N = \{w_i \mid i < s\}$, we prove by straightforward induction on $i < s$ that $E_e = E'_e$ for all edges e incident with a node $j \leq i$.

Existence: the basic idea is to represent the input value X_i by the set $B_i \subseteq n$ such that $j \in B_i$ iff $X_i \geq X_j$. Then $X_i \leq X_j$ iff $B_i \subseteq B_j$, hence $\max_{\leq}\{X_i, X_j\}$, $\min_{\leq}\{X_i, X_j\}$ are represented by $B_i \cup B_j$, $B_i \cap B_j$, respectively. In other words, we can compute min or max by n parallel binary conjunctions or disjunctions in this representation, as in the 0-1 case. In more detail, we construct a circuit C as follows. For each edge e in N, we put in C nodes e_i for all $i < n$. If w is a comparator in N with incoming edges e, f and outgoing edges l, h, we include in C the gates $l_i = e_i \wedge f_i$, $h_i = e_i \vee f_i$. If e^k is the outgoing edge of the kth input node, we make e^k_i an input node of the circuit and initialize it to 1 iff $X_i \leq X_k$ using NC^1_*-comprehension. Since C is a circuit of logarithmic depth defined by an NC^1_*-formula without set parameters, we can evaluate it by Lemma 2.1. Let V be its valuation, let $\varphi(e)$ denote the NC^1_*-formula $\exists k < n\, \forall i < n\, V(e_i) = V(e^k_i)$, and for each edge e, define the set

$$E_e = \{j \mid \exists k < n\, (\forall i < n\, V(e_i) = V(e^k_i) \wedge j \in X_k)\}$$

by NC^1_*-comprehension. If w is a comparator with incoming edges e, f and outgoing edges l, h, and if we assume $\varphi(e) \wedge \varphi(f)$, then it is easy to see that $\varphi(l) \wedge \varphi(h)$, and $E_l = \min_{\leq}\{E_e, E_f\}$, $E_h = \max_{\leq}\{E_e, E_f\}$. We can thus prove by induction $\varphi(e)$ for all e, which implies that E is a correct evaluation of N. $\square$

Lemma 2.4 *(in VNC^1_*) Let N, $\vec{X}$, and $\leq$ be as in Lemma 2.3. Let $\preceq$ be an NC^1_*-defined total order, and F an NC^1_*-function such that $X \leq X'$ implies $F(X) \preceq F(X')$. Then* $\mathrm{eval}(N, \preceq, F(\vec{X})) = F(\mathrm{eval}(N, \leq, \vec{X}))$.

Proof: Let E be the evaluation of N on $\vec{X}$ wrt $\leq$, and put $E'_e = F(E_e)$. Then E' is an evaluation of N on $F(\vec{X})$ wrt $\preceq$. □

Lemma 2.5 *(in VNC^1_*) Let N, $\vec{X}$, and $\leq$ be as in Lemma 2.3, and $\vec{Y} = \mathrm{eval}(N, \leq, \vec{X})$. Then there exists a permutation π of n such that $Y_i = X_{\pi(i)}$ for all $i < n$.*

Proof: The proof of Lemma 2.3 shows that

$$(*) \qquad \forall i < n\, \exists j < n\, Y_i = X_j.$$

On the other hand, if $j < n$, $N = \langle w_k \mid k < s\rangle$, and $\vec{E}$ is the evaluation of N on $\vec{X}$ wrt $\leq$, we can show by induction on k the following property: if w_k is a comparator node, there exists an edge e going from a node $\leq k$ to a node $> k$ such that $E_e = X_j$. Therefore,

$$(**) \qquad \forall j < n\, \exists i < n\, X_j = Y_i.$$

Assume first that the X_i's are pairwise distinct. Then for each i there is a *unique* j such that $Y_i = X_j$ by $(*)$. We put $\pi(i) = j$. Then $(**)$ implies that π is surjective, hence it is a bijection by *PHP* (provable in $VTC^0 \subseteq VNC^1_*$), and $Y_i = X_{\pi(i)}$ by the definition.

In the general case, we define

$$i \preceq j \quad \text{iff} \quad X_i \leq X_j \wedge (X_j \leq X_i \to i \leq j).$$

It is easy to see that $\preceq$ is a total order on n, hence by the previous part of the proof, there exists a permutation π such that $\mathrm{eval}(N, \preceq, \langle 0, \dots, n-1\rangle)_i = \pi(i)$. Using Lemma 2.4 for $F(i) = X_i$, we obtain

$$Y_i = \mathrm{eval}(N, \leq, \langle F(0), \dots, F(n-1)\rangle)_i = F(\mathrm{eval}(N, \preceq, \langle 0, \dots, n-1\rangle)_i) = X_{\pi(i)}. \qquad \square$$

Lemma 2.6 *(in VNC^1_*) If $\leq$ is a total ordering defined by an NC^1_*-formula, and $\langle X_i \mid i < n\rangle$ a sequence of sets in the domain of $\leq$, then there exists a permutation π of n such that $X_{\pi(i)} \leq X_{\pi(j)}$ for each $i \leq j < n$.*

Proof: Define

$$i \preceq j \quad \text{iff} \quad X_i < X_j \vee (X_i = X_j \wedge i \leq j).$$

It is easy to see that $\preceq$ is a total order on n. Put

$$\sigma(i) := \mathrm{card}\{k < n \mid k \prec i\}.$$

Clearly $i \prec j$ implies $\sigma(i) < \sigma(j)$. In particular, σ is injective, hence it is a permutation by *PHP*. We can thus define $\pi = \sigma^{-1}$, and then $i \leq j$ implies $\pi(i) \preceq \pi(j)$, which gives $X_{\pi(i)} \leq X_{\pi(j)}$. □

3 Ajtai–Komlós–Szemerédi–Paterson network

In this section we will define in detail Paterson's variant of the Ajtai–Komlós–Szemerédi network. Before describing the sorting network proper, let us start with a few auxiliary structures.

Definition 3.1 Let D and $0 < \varepsilon < 1$ be constants. An *$\langle \varepsilon, D\rangle$-expander on $m + m$ vertices* is a bipartite graph $G \subseteq m \times m$ such that every vertex (in either partition) has degree at most D, and for every $k \leq m$, every subset of one partition with more than εk vertices has at least $(1 - \varepsilon)k$ neighbours in the other partition.

From now on, we fix the first parameter ε_0 of our sorting network, say $\varepsilon_0 = 1/600$.

Assumption 3.2 *There exists a constant D and a (parameter-free) NC^1_*-function $G(m)$ such that VNC^1_* proves: for all numbers m, $G(m)$ is an $\langle \varepsilon_0, D\rangle$-expander on $m + m$ vertices.*

We fix D from Assumption 3.2 as our next parameter.

Definition 3.3 An *ε_0-halver on m elements,* where m is even, is a comparator network with m inputs whose output is partitioned into two blocks (left and right) of size $m/2$ with the following property: for each $k \leq m/2$, if a 0-1 input contains k zeros, then at most $\varepsilon_0 k$ zeros get in the right output block, and if the input contains k ones, then at most $\varepsilon_0 k$ ones get in the left output block.

Lemma 3.4 *There is an NC^1_*-function which, provably in VNC^1_*, computes for any even m an ε_0-halver on m inputs of depth D^2.*

Proof: Let G be an $\langle \varepsilon_0, D\rangle$-expander on $m/2 + m/2$ vertices given by Assumption 3.2. For each partition and each its vertex, we enumerate its outgoing edges by numbers $i < D$. In this way, every edge is labelled by a pair of numbers $\langle i, j\rangle \in D \times D$. As different edges with the same label are disjoint, the labelling defines a partition of the edges of G into D^2 partial matchings, which we denote by $\{G_k \mid k < D^2\}$. We construct a comparator network on m inputs as follows. We split the wires into a left and right block as in Definition 3.3, and we identify each block with the vertices of one partition of G. For each $k < D^2$, we include a layer of comparators corresponding to the edges in G_k (with the higher output of the comparator landing in the right block).

Consider an evaluation of the network on a 0-1 input, and a wire a from the left block. In each layer of the network, the value of a is either unchanged, or it is replaced with the minimum of the value of a and of a value of some wire in the right block, hence the value of a never increases during the computation. Symmetrically, the value of a wire in the right block never decreases. Let $\langle a, b\rangle \in G$. We have $\langle a, b\rangle \in G_k$ for some k. After the kth layer, the value of wire a is less that or equal to the value of wire b, and then the former can only decrease, and the latter only increase, hence the relation is preserved. It follows that the output of the network is compatible with G in the following sense: the output value of a wire

a in the left block is less that or equal to the value of a wire b in the right block whenever they are joined by an edge in G.

Let there be $k \leq m/2$ zeros and $m-k$ ones in the input (or in the output, for that matter), and assume for contradiction that the right output block contains strictly more than $\varepsilon_0 k$ zeros. As G is an expander, the positions of these zeros are connected by an edge to at least $(1-\varepsilon_0)k$ positions in the left block. By the compatibility property, the value of each of them is also zero, hence the total number of zeros in the output is more than $\varepsilon_0 k + (1-\varepsilon_0)k = k$, a contradiction. The case of k ones in the input is symmetric. □

Definition 3.5 Let N be a comparator network with $m+k$ inputs. A network N' on m inputs is constructed from N by *chopping from left* as follows. We pick k input wires (say, the wires with the smallest index), and mark them for deletion. If both inputs of a comparator are marked, we mark both outputs as well. If only one input of a comparator is marked, we mark its lower output. When we finish the marking, we delete all marked wires and comparators with both inputs marked, and we replace each comparator with one marked input with a wire connecting its unmarked input and output. (This is equivalent to the following operation: we expand the m inputs with k virtual elements, we apply N while considering the virtual elements to order below the real elements, and then we delete the virtual elements from output.) *Chopping from right* is defined symmetrically.

Definition 3.6 Let $\varepsilon_0 \leq \varepsilon < 1$ be a rational constant, and $m \geq l > 0$ be even integers. An *$\langle l, \varepsilon, \varepsilon_0\rangle$-separator on m elements* is a comparator network N whose m outputs are partitioned into four blocks, FL, CL, CR, FR, of sizes $\operatorname{card} FL = \operatorname{card} FR = l/2$, $\operatorname{card} CL = \operatorname{card} CR = (m-l)/2$, such that N is an ε_0-halver with respect to the blocks $L = FL \cup CL$ and $R = FR \cup CR$, and satisfies the following additional property: for any $k \leq l/2$, if a 0-1 input contains k zeros, then at most εk zeros are output outside FL, and if the input contains k ones, then at most εk ones are output outside FR.

Lemma 3.7 *Let $p \geq 0$ be an integer constant. There exists an NC^1_*-function which, provably in VNC^1_*, computes an $\langle l, (p+1)\varepsilon_0, \varepsilon_0\rangle$-separator on m elements of depth $(p+1)D^2$ for any given even m and even $l \leq m$ such that $l \geq m2^{-p}$.*

Proof: We proceed by induction on p. (Notice that the induction is external, as p is standard.) If $p = 0$, it suffices to take an ε_0-halver on m elements from Lemma 3.4. Let $p > 0$, and assume that the statement is true for $p-1$. We are given even m, l such that $2^{-p}m \leq l \leq m$. If $2^{1-p}m \leq l$, we may simply use the induction hypothesis, hence we assume $l \leq 2^{1-p}m$. We distinguish two cases.

First, assume that $p > 1$, so that $2l \leq m$. By the induction hypothesis, we obtain a $\langle 2l, p\varepsilon_0, \varepsilon_0\rangle$-separator on m inputs. We denote its output blocks by FL', CL', CR', FR'. We take an ε_0-halver H on l elements. We apply H to FL', denoting its output blocks as FL (the left one) and CL'' (the right one), and symmetrically we apply a copy of H in parallel to FR' obtaining CR'' and FR. We put $CL = CL' \cup CL''$ and $CR = CR' \cup CR''$. Clearly the resulting network is an ε_0-halver. Consider an input with k zeros, where $k \leq l/2$. Then $k \leq 2l/2$, hence at most $p\varepsilon_0 k$ zeros land outside FL' by the induction hypothesis. There remain at most $k \leq l/2$ zeros in FL', and H is a halver, thus at most $\varepsilon_0 k$ zeros end up in CL''. In total, at most $(p+1)\varepsilon_0 k$ zeros end up outside FL. The case of an input with k ones is symmetric.

Finally, let $p = 1$, thus $m/2 \leq l \leq m$. We construct our network as follows. First, we apply an ε_0-halver on m elements, obtaining the blocks L and R. We fix an ε_0-halver H on l elements. We chop H from right to $m/2$ inputs, and apply it to L, denoting its left output block with $l/2$ elements as FL, and its chopped right block as CL. Symmetrically, we chop a copy of H from left to $m/2$ inputs, and apply it in parallel to R, obtaining FR and CR. Again, it is clear that the network is an ε_0-halver. Consider an input with k zeros, where $k \leq l/2$. At most $k\varepsilon_0$ zeros end up in R. We can simulate the effect of chopped H on L as follows: we extend the partial result in L with ones to l elements, apply H, and discard the excessive ones from the right block CL. The number of zeros in the extended input is thus still at most $k \leq l/2$, hence at most $\varepsilon_0 k$ zeros land in CL. In total, at most $2\varepsilon_0 k$ zeros end up outside FL, as required. The case of an input with k ones is symmetric. $\square$

We now proceed to the description of the sorting network. Let n be the number of inputs. We fix the parameters $p = 4$, $\lambda_0 = 2^{-p} = 1/16$, $\varepsilon = (p+1)\varepsilon_0 = 1/120$, $\lambda = 1/8$, $A = 3$, $C = 150$, $\nu = 2\lambda A + (1-\lambda)/2A = 43/48$, $c_m = \lceil -\log 2A/\log \nu \rceil = 17$. Without loss of generality, we assume

$$n \geq C/\nu.$$

The sorting network consists of $O(\log n)$ *stages*, where the transition from one stage to the next one is computed by a constant depth comparator network. In each stage, the n elements are divided into a number of *bags*. Each bag is capable of accommodating a certain number of elements, called its *capacity*, but some of the bags may actually hold less elements than its capacity. The bags are organized in a subset of an ambient binary tree. All bags on the same level of the tree have the same capacity. In stage t, bags with nonempty capacity only appear at levels d such that $d \equiv t \pmod 2$ and $d_0(t) \leq d \leq d_1(t)$. (Note that we number stages and levels of the tree starting from 0.) We will write just d_0, d_1 if t is understood from the context. We label bags on level d by numbers $i < 2^d$ in the natural order from left to right (i.e., the children of the ith bag on level d are the $2i$th and $(2i+1)$th bags on level $d+1$).

The level d_1 is called the *bottom level*, and it is the only one which may contain bags not filled up to their full capacity. The level d_0 is the *root level*. The condition $d \geq d_0$ effectively means that the structure consists of 2^{d_0} disjoint trees with roots at the root level. Each of these 2^{d_0} trees also has a *cold storage* attached to it, which is a special bag sitting outside the ambient tree structure. Note that the roots of the trees may be empty, if $d_0(t) \not\equiv t \pmod 2$. We label the trees by numbers $i < 2^{d_0}$ in the left-to-right order, the same as their roots.

The parameters and sizes of various parts of the structure are as follows. For any $t \leq c_m \lceil \log n \rceil$ and $d \leq 2\lceil \log n \rceil$, put

$$s'(t,d) = \frac{n}{2^d}\left(1 - (2A)^{d-2}\nu^t\right).$$

(Notice that here and below, the exponentiation has a fixed base, and the exponent is bounded by $O(\log n)$, hence the expression is definable by a well-behaved bounded formula in $I\Delta_0 \subseteq$

V^0.) We define

$$
\begin{aligned}
d_1'(t) &= \max\{d \mid (2A)^{d-2} < \nu^{-t}\} = \max\{d \mid s'(t,d) > 0\},\\
d_1(t) &= d_1'(t) - ((d_1'(t) - t) \bmod 2),\\
d_0'(t) &= \min\{d \mid nA^d\nu^t \geq C\},\\
d_0(t) &= \begin{cases} d_0'(t) - 1 & \text{if } t > 0, d_0'(t) > d_0'(t-1), d_0'(t-1) \equiv t \pmod 2,\\ d_0'(t) & \text{otherwise.}\end{cases}
\end{aligned}
$$

As $A > 1 > \nu$, $d_\alpha(t)$ are well-defined by a bounded formula, and $d_\alpha(t) = O(t)$.

There are $n \bmod 2^{d_0}$ trees of size $\lceil n2^{-d_0}\rceil$, and $2^{d_0} - (n \bmod 2^{d_0})$ trees of size $\lfloor n2^{-d_0}\rfloor$. These sizes are distributed so that the leftmost i trees have total size $\lfloor in2^{-d_0}\rfloor$, thus the tree with label i has size

$$
T(t,i) = \lfloor (i+1)n2^{-d_0}\rfloor - \lfloor in2^{-d_0}\rfloor = \begin{cases} \lceil n2^{-d_0}\rceil & \text{if } (in \bmod 2^{d_0}) > ((i+1)n \bmod 2^{d_0}),\\ \lfloor n2^{-d_0}\rfloor & \text{otherwise.}\end{cases}
$$

If $d \geq d_0$ and $d \equiv t \pmod 2$, each subtree rooted at level d has nominal capacity

$$
s(t,d) = 2\lceil s'(t,d)/2\rceil,
$$

and actually holds $\max\{0, s(t,d)\}$ elements. This means that the capacity of any bag at level d is

$$
b(t,d) = \begin{cases} s(t,d) - 4s(t,d+2) & \text{if } d_0 \leq d \leq d_1, d \equiv t \pmod 2,\\ 0 & \text{otherwise,}\end{cases}
$$

and the number of elements it holds is

$$
h(t,d) = \begin{cases} s(t,d) & \text{if } d = d_1,\\ b(t,d) & \text{otherwise.}\end{cases}
$$

Note that the capacity and actual content of each bag is even. The capacity (and content) of cold storage is accordingly

$$
c(t,i) = \begin{cases} T(t,i) - s(t,d_0) & \text{if } d_0 \equiv t \pmod 2,\\ T(t,i) - 2s(t,d_0+1) & \text{otherwise,}\end{cases}
$$

where $i < 2^{d_0}$ is the label of the tree.

We also define "ideal sizes" of the various parameters, which are rational numbers approximated by the real sizes. The ideal size of each tree is $T'(t) = n2^{-d_0}$. We already know the ideal subtree capacity $s'(t,d)$. The ideal bag capacity is defined by

$$
b'(t,d) = \begin{cases} s'(t,d) - 4s'(t,d+2) = \left(1 - \dfrac{1}{4A^2}\right) nA^d\nu^t & \text{if } d_0 \leq d \leq d_1, d \equiv t \pmod 2,\\ 0 & \text{otherwise,}\end{cases}
$$

and the ideal cold storage capacity is

$$c'(t) = \begin{cases} T'(t) - s'(t, d_0) = \dfrac{1}{4A^2} n A^{d_0} \nu^t & \text{if } d_0 \equiv t \pmod 2, \\[2ex] T'(t) - 2s'(t, d_0 + 1) = \dfrac{1}{2A} n A^{d_0} \nu^t & \text{otherwise.} \end{cases}$$

Notice that $d_0(0) = d_1(0) = 0$, thus the structure at stage 0 consists of a single root bag and the associated cold storage. We initialize the network by putting arbitrary $s(0, 0)$ elements to the root bag, and the rest to the cold storage.

Let t_m be the least $t > 0$ such that $d_0(t) = d_1(t)$. We will see below (Lemma 4.4) that t_m exists, $t_m \le c_m \lceil \log n \rceil$, and $T(t_m, i)$ is bounded by a constant. The stage t_m will be the last regular stage of our network. After this stage, we sort each of the 2^{d_0} constant-size trees using a suitable constant-size sorting network, and stop.

We have to define the constant-depth network which makes the transition from stage $t < t_m$ to stage $t + 1$. A general overview is that we will apply a suitable constant-depth subnetwork to each nonempty bag to split its content into a few parts, which we send to its parent and children bags. Root bags will exchange elements with their cold storage instead of a parent. Notice that when a bag is nonempty at stage t, then its children and parent are empty (except for the cold storage), whereas the opposite holds at stage $t + 1$. Now we describe the actual network fragments. We have to distinguish several cases.

Case 1: we consider a nonempty bag B on level d such that $d_0 < d \le d_1$. If $d = d_1(t) > d_1(t + 1)$, we send all of B to its parent. Otherwise, we use an $\langle l, \varepsilon, \varepsilon_0 \rangle$-separator of depth $(p + 1)D^2$ from Lemma 3.7 to split B into FL, CL, CR, and FR, where $l = s(t, d) - 2s(t + 1, d + 1)$. We send CL to the left child, CR to the right child, and $FL \cup FR$ to the parent.

Case 2: a nonempty root bag B, assuming $d_0(t) = d_0(t + 1)$. We apply a separator just like in Case 1, except that we send $FL \cup FR$ to the cold storage instead of B's parent.

Case 3: a root bag B of the ith tree, assuming $d_0(t) \ne d_0(t+1)$. We will see in Lemma 4.2 that $d_0(t + 1) = d_0(t) + 1$, and $b(t, d_0) + c(t, i)$ is bounded by a constant. Note that $d_1(t) \ge d_0(t) + 2$. We merge the bag with its cold storage, and apply a constant-size sorting network to split it to two pieces, L of size $T(t + 1, 2i) - 2s(t, d_0(t) + 2)$, and R of size $T(t + 1, 2i + 1) - 2s(t, d_0(t) + 2)$, so that each element of L is less than or equal to each element of R. We put arbitrary $c(t + 1, 2i)$ elements from L to the newly created cold storage of the left child of B, and send the rest of L to the left child itself. We do the same with R and the right child.

Case 4: a cold storage. If $d_0 \equiv t \pmod 2$, we expand the storage with some elements sent from its root bag, as described in Case 2, or merge it with the root bag and split it to children, as described in Case 3. If $d_0 \not\equiv t \pmod 2$ (which implies $d_0(t) = d_0(t + 1)$, as we will see), we send arbitrary $s(t + 1, d_0) - 2s(t, d_0 + 1)$ elements to the root bag.

We observe that the network is defined by an NC^1_*-function $F(n)$.

4 Analysis of the network

We first check that our definition of the various parameters of the network are sensible, and that all sizes work out correctly when shuffling elements around.

We have already seen why d_0 and d_1 are well-defined.

Lemma 4.1 *(in VNC^1_*)*

(i) $d'_1(t) \le d'_1(t+1) \le d'_1(t)+1$.

(ii) $d_1(t+1) - d_1(t) = \pm 1$.

(iii) If $t > 0$, then $d_1(t) > 0$.

Proof: (i) $d'_1(t) \le d'_1(t+1)$ is clear as $\nu < 1$. We have $(2A)^{d_1(t)-1} \ge \nu^{-t}$, hence $(2A)^{d_1(t)} \ge \nu^{-(t+1)}$ as $A \ge \nu^{-1}$, which implies $d_1(t+1) < d_1(t)+2$.

(ii) Since $d'_1(t) - 1 \le d_1(t) \le d'_1(t)$, we obtain $|d_1(t+1) - d_1(t)| \le 2$ from (i). However, $d_1(t) \equiv t \not\equiv t+1 \equiv d_1(t+1) \pmod 2$, hence $|d_1(t+1) - d_1(t)| = 1$.

(iii) Since $(2A)^0 = 1 < \nu^{-t}$, we have $d'_1(t) \ge 2$ and $d_1(t) \ge 1$. $\square$

Lemma 4.2 *(in VNC^1_*)*

(i) $d'_0(t) \le d'_0(t+1) \le d'_0(t)+1$.

(ii) $d'_0(t-1) \le d_0(t) \le d'_0(t)$.

(iii) $d_0(t) \le d_0(t+1) \le d_0(t)+1$.

(iv) If $d_0(t) < d_0(t+1)$, then $d_0(t) \equiv t \pmod 2$, and $b(t, d_0(t)) + c(t,i) \le \lceil C/\nu \rceil$.

Proof: (i) follows from $\nu < 1 < A\nu$ as in the proof of Lemma 4.1.

(ii) If $d_0(t) \neq d'_0(t)$, then $d'_0(t) > d'_0(t-1)$ and $d_0(t) = d'_0(t) - 1$ by the definition.

(iii) We have $d_0(t) \le d'_0(t) \le d_0(t+1)$ from (ii). $d_0(t+1) \ge d_0(t)$ could only happen if $d_0(t) < d'_0(t) < d'_0(t+1) = d_0(t+1)$. The former inequality implies $d_0(t) = d'_0(t-1) \equiv t \pmod 2$, hence $d'_0(t) = d_0(t)+1 \equiv t+1 \pmod 2$, thus by the definition $d_0(t+1) = d'_0(t+1)-1$, a contradiction.

(iv) If $d_0(t) < d'_0(t)$, then $d_0(t) = d'(t-1) \equiv t \pmod 2$. If $d_0(t) = d'_0(t)$, we must have $d_0(t+1) = d'_0(t+1) > d'_0(t)$, hence $d_0(t) = d'_0(t) \equiv t \pmod 2$.

Since $d_0(t) < d'_0(t+1)$, we have $nA^{d_0(t)}\nu^{t+1} < C$, hence

$$\begin{aligned} b(t, d_0(t)) + c(t,i) = T(t,i) - 4s(t, d_0(t)+2) &< T'(t) - 4s'(t, d_0(t)+2) + 1 \\ &= b'(t, d_0(t)) + c'(t) + 1 = nA^{d_0(t)}\nu^t + 1 < 1 + C\nu^{-1}. \end{aligned}$$

$\square$

Lemma 4.3 *(in VNC^1_*) If $d_0 \le d \le d_1$, $d \equiv t \pmod 2$, and $i < 2^{d_0}$, then $b(t,d) > 0$ and $c(t,i) > 0$.*

Proof: Since $d_0(t) \ge d'_0(t-1)$, we have $nA^{d_0}\nu^{t-1} \ge C$. As $s(t,d) < s'(t,d) + 2$, we obtain

$$\begin{aligned} b(t,d) &= s(t,d) - 4s(t,d+2) > s'(t,d) - 4s'(t,d+2) - 8 = b'(t,d) - 8 \\ &= \left(1 - \frac{1}{4A^2}\right) nA^d\nu^t - 8 \ge \left(1 - \frac{1}{4A^2}\right) nA^{d_0}\nu^t - 8 \ge \left(1 - \frac{1}{4A^2}\right) C\nu - 8 \ge 0. \end{aligned}$$

If $d_0 \equiv t \pmod 2$, we have

$$c(t,i) = T(t,i) - s(t,d_0) > c'(t) - 3 = \frac{n}{4A^2} A^{d_0} \nu^t - 3 \geq \frac{C\nu}{4A^2} - 3 \geq 0.$$

Similarly, if $d_0 \not\equiv t \pmod 2$, then

$$c(t,i) > \frac{C\nu}{2A} - 5 \geq 0. \qquad \square$$

Lemma 4.4 *(in VNC^1_*) There exists*

$$t_m = \min\{t > 0 \mid d_0(t) = d_1(t)\},$$

which satisfies $t_m \leq c_m \lceil \log n \rceil$. We have $d_0(t) < d_1(t)$ for all $0 < t < t_m$. Moreover, $d'_1(t) \leq \lfloor \log n \rfloor$ for all $t \leq t_m$, and $T(t_m, i) \leq \lceil C/\nu \rceil$.

Proof: Put $t = c_m \lceil \log n \rceil$ and $d = d'_0(t)$. As $\nu^{-c_m} \geq 2A$, we have

$$C \leq nA^d \nu^t \leq nA^d (2A)^{-\lceil \log n \rceil} \leq A^{d - \lceil \log n \rceil},$$

thus $d \geq \lceil \log n \rceil$ and $2^d \geq n$. This implies

$$(2A)^{d-2} \nu^t \geq \frac{1}{4A^2} nA^d \nu^t \geq \frac{C}{4A^2} \geq 1,$$

hence $d'_1(t) < d'_0(t)$, and $d_1(t) \leq d_0(t)$.

On the other hand, $d_1(0) = 1 > d_0(0) = 0$ from Lemma 4.1 and $n\nu \geq C$, hence there exists

$$t_m := \min\{t > 0 \mid d_1(t) \leq d_0(t)\} \leq c_m \lceil \log n \rceil.$$

We have $d_0(t_m - 1) < d_1(t_m - 1)$. By Lemmas 4.1 and 4.2 we obtain $d_0(t_m) = d_1(t_m)$ unless $d_0(t_m - 1) = d_1(t_m)$, $d_1(t_m - 1) = d_0(t_m) = d_1(t_m) + 1$. But then $t_m - 1 \equiv d_0(t_m - 1) = d_1(t_m) \equiv t_m \pmod 2$, a contradiction.

As $d_0(t_m) = d_1(t_m)$, we have

$$T(t_m, i) = c(t_m, i) + s(t_m, d_0) \leq c(t_m, i) + b(t_m, d_0) \leq \lceil C/\nu \rceil$$

by Lemma 4.2. Finally, $\lfloor n 2^{-d_0(t_m)} \rfloor = T(t_m, 0) = s(t_m, d_0(t_m)) + c(t_m, 0) \geq 2$ by Lemma 4.3, hence $d'_1(t) \leq d_1(t_m) + 1 = d_0(t_m) + 1 \leq \lfloor \log n \rfloor$ for any $t \leq t_m$. $\square$

The Lemma below implies, among others, that Case 4 makes sense.

Lemma 4.5 *(in VNC^1_*) If $d_0(t+1) \leq d < d_1(t)$ and $d \not\equiv t \pmod 2$, then $s(t+1, d) > 2s(t, d+1)$.*

Proof: We have

$$\begin{aligned} s(t+1,d) - 2s(t,d+1) &\geq s'(t+1,d) - 2s'(t,d+1) - 4 \\ &= \frac{n}{2^d}\left(1 - (2A)^{d-2}\nu^{t+1} - 1 + (2A)^{d-1}\nu^t\right) - 4 \\ &= \frac{n}{4A^2} A^d \nu^t (2A - \nu) - 4 \geq \frac{C}{4A^2}(2A - \nu) - 4 > 0. \end{aligned}$$

$\square$

The following Lemma ensures that the $\langle l,\varepsilon,\varepsilon_0\rangle$-separator in Cases 1 and 2 is used correctly.

Lemma 4.6 *(in VNC^1_*) Let $d_0(t) \le d < d_1(t+1)$, $d \equiv t \pmod 2$, $m = h(t,d)$, and $l = s(t,d) - 2s(t+1,d+1)$. Then $m \ge l \ge m\lambda_0$.*

Proof: Recall that $b'(t,d) = \big(1-(4A^2)^{-1}\big)nA^d\nu^t$. Put $l' = s'(t,s) - 2s'(t+1,d+1)$. We have

$$l' = \frac{n}{2^d}\left(1-(2A)^{d-2}\nu^t\right) - \frac{n}{2^d}\left(1-(2A)^{d-1}\nu^{t+1}\right) = \frac{n}{2^d}(2A)^{d-2}\nu^t(2A\nu-1)$$
$$= nA^d\nu^t\frac{1}{4A^2}(4A^2\lambda+1-\lambda-1) = \left(1-\frac{1}{4A^2}\right)nA^d\nu^t\lambda = \lambda b'(t,d).$$

If $d = d_1(t)$, then $l \le s(t,d) = m$. Otherwise

$$m - l = 2s(t+1,d+1) - 4s(t,d+2) > 0$$

by Lemma 4.5.

For the other inequality, we have

$$\frac{l}{m} \ge \frac{l}{b(t,d)} = 1 - \frac{2s(t+1,d+1)-4s(t,d+2)}{s(t,d)-4s(t,d+2)} \ge 1 - \frac{2s(t+1,d+1)-4s(t,d+2)}{s'(t,d)-4s(t,d+2)}$$
$$= \frac{s'(t,d)-2s(t+1,d+1)}{s'(t,d)-4s(t,d+2)} \ge \frac{s'(t,d)-2s'(t+1,d+1)-4}{s'(t,d)-4s'(t,d+2)}$$
$$= \frac{l'-4}{b'(t,d)} \ge \lambda - \frac{4}{\big(1-(4A^2)^{-1}\big)C\nu} \ge \lambda_0.$$

□

The next Lemma shows that the splitting in Case 3 make sense.

Lemma 4.7 *(in VNC^1_*) Let $t < t_m$ be such that $d_0(t) < d_0(t+1)$, and $i < 2^{d_0(t)}$. Put $x_\alpha = T(t+1,2i+\alpha) - 2s(t,d_0(t)+2)$ for $\alpha = 0,1$. Then $x_0 + x_1 = b(t,d_0(t)) + c(t,i)$ and $x_\alpha \ge c(t+1,2i+\alpha)$.*

Proof: As $d_0(t+1) = d_0(t)+1$, we have

$$T(t+1,2i) + T(t+1,2i+1) = \lfloor (2i+2)n2^{-d_0(t+1)}\rfloor - \lfloor 2in2^{-d_0(t+1)}\rfloor$$
$$= \lfloor (i+1)n2^{-d_0(t)}\rfloor - \lfloor in2^{-d_0(t)}\rfloor = T(t,i).$$

Then clearly

$$b(t,d_0(t)) + c(t,i) = T(t,i) - 4s(t,d_0(t)+2) = x_0 + x_1.$$

Since $d_0(t+1) \equiv t+1 \pmod 2$, we have

$$x_\alpha - c(t+1,2i+\alpha) = s(t+1,d_0(t)+1) - 2s(t,d_0(t)+2) > 0$$

by Lemma 4.5. □

Lemma 4.8 *(in VNC^1_*) Let $t < t_m$, $d_0(t+1) \le d \le d_1(t+1)$, $d \equiv t+1 \pmod 2$. Then the total number of elements sent from stage t to any bag of level d is $h(t+1,d)$. If $i < 2^{d_0(t+1)}$, the number of elements sent to the ith cold storage is $c(t+1,i)$.*

Proof: We start with the cold storage. If $d_0(t) < d_0(t+1)$, the cold storage gets $c(t+1,i)$ elements by Case 3. Let thus $d_0(t) = d_0(t+1)$. If $d_0 \not\equiv t \pmod 2$, there remain

$$c(t,i) - (s(t+1,d_0) - 2s(t,d_0+1)) = T(t,i) - s(t+1,d_0) = c(t+1,i)$$

elements in the cold storage by Case 4. Otherwise, we get

$$c(t,i) + (s(t,d_0) - 2s(t+1,d_0+1)) = T(t,i) - 2s(t+1,d_0+1) = c(t+1,i)$$

elements by Case 2.

Now we turn to regular bags. First assume $d > d_1(t)$, hence $d = d_1(t+1)$. We get

$$\tfrac{1}{2}\big(s(t,d-1) - (s(t,d-1) - 2s(t+1,d))\big) = s(t+1,d) = h(t+1,d)$$

elements from the parent by Case 1 or 2. Let thus assume $d < d_1(t)$.

If $d > d_0(t+1)$, we obtain

$$\tfrac{1}{2}\big(b(t,d-1) - (s(t,d-1) - 2s(t+1,d))\big) = s(t+1,d) - 2s(t,d+1)$$

elements from the parent by Case 1 or 2. If $d = d_0(t+1) = d_0(t)$, we get $s(t+1,d) - 2s(t,d+1)$ elements from cold storage by Case 4. If $d = d_0(t+1) > d_0(t)$, we get

$$T(t+1,i) - 2s(t,d+1) - c(t+1,i) = s(t+1,d) - 2s(t,d+1)$$

elements from splitting of the parent by Case 3. Thus, in all cases, the bag obtains

$$s(t+1,d) - 2s(t,d+1)$$

elements "from above".

If $d = d_1(t+1)$, then we obtain $s(t,d+1)$ elements from each child by Case 1, hence we get $s(t+1,d) = h(t+1,d)$ elements in total.

If $d < d_1(t+1)$, then we cannot have $d+1 = d_1(t) > d_1(t+1)$. We thus obtain $s(t,d+1) - 2s(t+1,d+2)$ elements from each child by Case 1. We have

$$\begin{aligned} s(t+1,d) - 2s(t,d+1) + 2(s(t,d+1) - 2s(t+1,d+2)) \\ = s(t+1,d) - 4s(t+1,d+2) = b(t+1,d) = h(t+1,d) \end{aligned}$$

elements in total. $\square$

Having checked that the network is coherently defined, we turn our attention to its behaviour when evaluated. In order to simplify the analysis, we first consider the special case when the input is a permutation of the sequence $0,\dots,n-1$ (the most important point being that the inputs are pairwise distinct), and $\le$ is the usual ordering. We fix an evaluation of the network on such input. (Strictly speaking, we only defined evaluation of a network on set inputs, not number inputs. We can encode numbers $k < n$ by sets in a straightforward way, e.g., by $\{k\}$.)

We associate with each bag B its *natural interval* in $[0, n)$: the ith bag on level d in the left-to-right order corresponds to the interval

$$I(d, i) = \bigl[\lfloor in2^{-d} \rfloor, \lfloor (i+1)n2^{-d} \rfloor\bigr).$$

An element x of the bag B whose value is outside $I(d, i)$ is called a *stranger*, and its *strangeness* is defined as the smallest number j such that x belongs to the natural interval of B's ancestor on level $d - j$, i.e., $I(d - j, \lfloor i2^{-j} \rfloor)$. We let $S_j(t, d, i)$ denote the number of elements of B at stage t of strangeness at least j. Let $\xi(t)$ denote the NC^1_*-formula which is the conjunction of the following conditions:

(i) For every $i < 2^{d_0}$, the values of all elements of the ith tree (including its cold storage) at stage t belong to $I(d_0, i)$.

(ii) For every d, i, j such that $d_0 \le d \le d_1$, $i < 2^d$, and $0 < j \le d - d_0$, we have

$$S_j(t, d, i) \le \mu \delta^j b'(t, d),$$

where we put $\mu = 10$, $\delta = 1/270$.

Lemma 4.9 *(in VNC^1_*) If $\xi(t)$ holds, then condition (i) of $\xi(t+1)$ also holds.*

Proof: If $d_0(t) = d_0(t+1)$, the conclusion is trivial, as element movements respect tree boundaries. Let us thus assume $d_0(t+1) = d_0(t) + 1$, and denote $d_0 = d_0(t)$ for short. Fix $i < 2^{d_0}$. We know by $\xi(t)$ that all elements of the $2i$th and the $(2i+1)$th tree at stage $t+1$, which come from the ith tree at stage t, belong to $I(d_0, i) = I(d_0 + 1, 2i) \,\dot\cup\, I(d_0 + 1, 2i + 1)$.

Consider $d > d_0$ such that $d \equiv t \pmod 2$, and $i' < 2^d$ such that $\lfloor i' 2^{d_0 - d} \rfloor = i$. Note that $d \ge d_0 + 2$. Since $A\delta \le 1$, we have

$$\begin{aligned} S_{d-d_0}(t, d, i') &\le \mu \delta^{d-d_0} b'(t, d) = \mu \delta^{d-d_0} A^{d-d_0} b'(t, d_0) \le \mu (A\delta)^2 b'(t, d_0) \\ &= \mu (A\delta)^2 \left(1 - \frac{1}{4A^2}\right) n A^{d_0} \nu^t < \mu (A\delta)^2 \left(1 - \frac{1}{4A^2}\right) C \nu^{-1} \le 1, \end{aligned}$$

using $\xi(t)$, and $nA^{d_0}\nu^{t+1} < C$, which follows from $d_0 < d_0'(t+1)$. This means that every element of the i'th bag on level d at stage t has strangeness less than $d - d_0$, i.e., it belongs to the interval $I(d_0 + 1, \lfloor i' 2^{d_0 + 1 - d} \rfloor)$. On the other hand, these elements end up in the $\lfloor i' 2^{d_0 + 1 - d} \rfloor$th tree at stage $t + 1$, as required.

The remaining elements of the $2i$th and $(2i+1)$th tree at stage $t+1$ come from the root and cold storage of the ith tree at stage t. We know from above that there are exactly $2s(t, d_0 + 2)$ elements of $I(d_0 + 1, 2i)$ and $2s(t, d_0 + 2)$ elements of $I(d_0 + 1, 2i + 1)$ in the rest of the ith tree at stage t. Since $I(d_0 + 1, 2i)$ and $I(d_0 + 1, 2i + 1)$ have $T(t + 1, 2i)$ and $T(t+1, 2i+1)$ elements in total, respectively, the root and cold storage of the ith tree at stage t contain $T(t + 1, 2i) - 2s(t, d_0 + 2)$ elements of $I(t + 1, 2i)$, and $T(t + 1, 2i + 1) - 2s(t, d_0 + 2)$

elements of $I(t+1, 2i+1)$. By Case 3 of the definition of the network, we send the smallest $T(t+1, 2i) - 2s(t, d_0+2)$ of these elements to the $2i$th tree at stage $t+1$, and the largest $T(t+1, 2i+1) - 2s(t, d_0+2)$ elements to the $(2i+1)$th tree. As elements of $I(t+1, 2i)$ are smaller than elements of $I(t+1, 2i+1)$, all these elements end up in the correct tree. $\square$

Lemma 4.10 *(in VNC^1_*) If $\xi(t)$ holds, $d_0(t+1) \le d \le d_1(t+1)$, $d \equiv t+1 \pmod 2$, $i < 2^d$, and $2 \le j \le d - d_0(t+1)$, then $S_j(t+1, d, i) \le \mu\delta^j b'(t+1, d)$.*

Proof: Note that $d_0(t+1) < d-1$. Denote by B the ith bag on level d. Elements of B of strangeness j or more at stage $t+1$ come from two sources: elements of B's children at stage t of strangeness at least $j+1$, and elements of B's parent at stage t of strangeness at least $j-1$, both using Case 1 of the definition of the network.

Using $\xi(t)$, the number of elements of B's children with strangeness $j+1$ or more is at most

$$(*) \qquad 2\mu\delta^{j+1} b'(t, d+1).$$

Let P be B's parent. The number of elements of P of strangeness $j-1$ or more at stage t is

$$k := S_{j-1}(t, d-1, \lfloor i/2 \rfloor) \le \mu\delta^{j-1} b'(t, d-1).$$

Let a be the number of elements of P whose value is smaller than x, and b the number of elements of P whose value is at least y, where $I(d-j, \lfloor i2^{-j} \rfloor) = [x, y)$, so that $a+b=k$. Put

$$l = s(t, d-1) - 2s(t+1, d).$$

By Case 1, we apply to P's content an $\langle l, \varepsilon, \varepsilon_0 \rangle$-separator S, and send the part $CL \cup CR$ of its output to B.

Notice that $b'(t, d-1) \ge \bigl(1 - (4A^2)^{-1}\bigr)C\nu$ by the proof of Lemma 4.3. Using the proof of Lemma 4.6, we obtain

$$\begin{aligned} l \ge s'(t, d-1) - 2s'(t+1, d) - 4 &= \lambda b'(t, d-1) - 4 \\ &\ge \left(\lambda - \frac{4}{\bigl(1-(4A^2)^{-1}\bigr)C\nu}\right) b'(t, d-1) \ge 2\mu\delta\, b'(t, d-1) \ge 2k, \end{aligned}$$

hence $a, b \le l/2$. Let $F(u) \in \{0, 1\}$ be defined by $F(u) = 1$ iff $u \ge x$. The application of F to the elements of P gives a 0-1 sequence with a zeros. If we evaluate S on this input, at most εa zeros end up outside FL by Definition 3.6. Using Lemma 2.4, the application of S to P sends at most εa elements smaller than x to $CL \cup CR \cup FR$. By a similar argument, at most εb elements greater than or equal to y end up in $CL \cup CR \cup FL$. In total, the number of elements outside $I(d-j, \lfloor i2^{-j} \rfloor)$ sent from P to B is at most

$$(**) \qquad \varepsilon a + \varepsilon b = \varepsilon k \le \varepsilon\mu\delta^{j-1} b'(t, d-1).$$

Putting $(*)$ and $(**)$ together, we see that at stage $t+1$, B contains at most

$$2\mu\delta^{j+1} b'(t, d+1) + \varepsilon\mu\delta^{j-1} b'(t, d-1) = \left(\frac{2A\delta}{\nu} + \frac{\varepsilon}{A\delta\nu}\right)\mu\delta^j b'(t+1, d) \le \mu\delta^j b'(t+1, d)$$

elements of strangeness j or more. $\square$

Lemma 4.11 *(in VNC^1_*) If $\xi(t)$ holds, $d_0(t+1) < d \le d_1(t+1)$, $d \equiv t+1 \pmod 2$, and $i < 2^d$, then $S_1(t+1,d,i) \le \mu\delta\, b'(t+1,d)$.*

Proof: Let B be the ith bag on level d, P its parent, and B' its sibling. Let $i' = i + (-1)^i$ be the label of B', and put $I = I(d,i)$, $I' = I(d,i')$.

Strangers in B at stage $t+1$ come from two sources: elements of strangeness at least 2 in B's children at stage t, of whom there are at most

$$(*) \qquad 2\mu\delta^2 b'(t,d+1) = 2A\mu\delta^2 b'(t,d),$$

and elements of P at stage t sent downwards to B which are either strangers in P, or belong to I'.

The number of elements of the subtree below B' at stage t which do not belong to I' is

$$\sum_{\substack{j=1\\ j \text{ odd}}}^{d_1-d} \sum_{\hat\imath=2^j i'}^{2^j(i'+1)-1} S_{j+1}(t,d+j,\hat\imath) \le \sum_{\substack{j=1\\ j \text{ odd}}}^{d_1-d} 2^j \mu\delta^{j+1} A^j b'(t,d)$$
$$= 2A\mu\delta^2 b'(t,d) \sum_{k=0}^{(d_1-d-1)/2} (2A\delta)^k \le \frac{2A\mu\delta^2}{1-4A^2\delta^2} b'(t,d) =: \alpha b'(t,d).$$

This subtree thus contains at least $2s(t,d+1) - \alpha b'(t,d)$ elements of I', hence P contains

$$x \le \operatorname{card} I' - 2s(t,d+1) + \alpha b'(t,d) \le 1 + \alpha b'(t,d) + \frac{n}{2^d} - 2s'(t,d+1)$$
$$= 1 + \alpha b'(t,d) + \frac{n}{2^d}(2A)^{d-1}\nu^t = 1 + \left(\alpha + \frac{2A}{4A^2-1}\right) b'(t,d)$$

elements of I'. P also contains

$$y + z \le \mu\delta\, b'(t,d-1) = \frac{\mu\delta}{A} b'(t,d)$$

strangers, where y is the number of elements below $\min(I \cup I')$, and z the number of elements above $\max(I \cup I')$.

Assume that i is even (i.e., $I < I'$); the other case is symmetric. Remember that we apply a $\langle l, \varepsilon, \varepsilon_0\rangle$-separator (hence an ε_0-halver) to P, and send the content of CL to B. Let c be the element of P which splits it in half, i.e., there are $\frac{1}{2}b(t,d-1)$ elements of P greater than c. Define $F(u) \in \{0,1\}$ by $F(u) = 1$ iff $u > c$. By Definition 3.3 and Lemma 2.4, there are at most $(\varepsilon_0/2)b(t,d-1)$ elements of P greater than c which end up in $FL \cup CL$. Furthermore, there are at most $\max\bigl\{0, x + z - \frac{1}{2}b(t,d-1)\bigr\}$ elements greater than $\max I$ below c, and y elements smaller than $\min I$. The total number of elements outside I in CL is thus bounded by

$$
\begin{aligned}
y + \max\bigl\{0, x + z - \tfrac{1}{2}b(t,d-1)\bigr\} + \frac{\varepsilon_0}{2}b(t,d-1)& \\
&\le 1 + \left(\alpha + \frac{2A}{4A^2-1} + \frac{\mu\delta}{A}\right) b'(t,d) - \frac{1-\varepsilon_0}{2} b(t,d-1) \\
&\le 5 + \left(\alpha + \frac{2A}{4A^2-1} + \frac{\mu\delta}{A}\right) b'(t,d) - \frac{1-\varepsilon_0}{2} b'(t,d-1) \\
&= 5 + \left(\alpha + \frac{2A}{4A^2-1} + \frac{\mu\delta}{A} - \frac{1-\varepsilon_0}{2A}\right) b'(t,d) \\
&\le \left(\alpha + \frac{1}{2A(4A^2-1)} + \frac{\mu\delta}{A} + \frac{\varepsilon_0}{2A} + \frac{5}{\bigl(1-(4A^2)^{-1}\bigr)AC\nu}\right) b'(t,d),
\end{aligned}
$$

as $b'(t,d-1) \ge \bigl(1-(4A^2)^{-1}\bigr)C\nu$ by the proof of Lemma 4.3. Combining this with $(*)$, we see that the number of strangers in B at stage $t+1$ is at most

$$
\begin{aligned}
&\left(\frac{2A\mu\delta^2}{1-4A^2\delta^2} + \frac{1}{2A(4A^2-1)} + \frac{\mu\delta}{A} + \frac{\varepsilon_0}{2A} + \frac{5}{\bigl(1-(4A^2)^{-1}\bigr)AC\nu} + 2A\mu\delta^2\right) b'(t,d) \\
&\qquad\qquad\qquad\qquad \le \mu\delta\nu\, b'(t,d) = \mu\delta\, b'(t+1,d).
\end{aligned}
$$

□

Theorem 4.12 *Under Assumption 3.2, there exists an NC^1_*-function $N(n)$, and a constant c such that VNC^1_* proves the following:*

For every $n > 0$, $N(n)$ is a comparator network on n inputs of depth at most $c\log n$. If $\le$ is a total ordering defined by an NC^1_*-formula, and $\langle X_i \mid i < n\rangle$ a sequence of sets in the domain of $\le$, then there exists a permutation π of n such that $\mathrm{eval}(N(n), \le, \vec{X}) = \langle X_{\pi(i)} \mid i < n\rangle$, *and $X_{\pi(i)} \le X_{\pi(j)}$ for every $i \le j < n$.*

Proof: If $n \le C/\nu$, we let $N(n)$ be any sorting network on n inputs, otherwise we define $N(n)$ as the network described in Section 3. Clearly, $N(n)$ is a comparator network on n inputs of depth at most $c_m(p+1)D^2\lceil\log n\rceil + O(1)$.

First, let $\vec{X}$ be a permutation of $\langle 0,\dots,n-1\rangle$, and $\le$ the usual ordering. For every $t < t_m$, $\xi(t)$ implies $\xi(t+1)$ by Lemmas 4.9, 4.10, and 4.11, and $\xi(0)$ holds trivially. Using induction, we obtain $\xi(t_m)$. By condition (i), each of the 2^{d_0} constant-size trees at stage t_m contains elements of its corresponding subinterval of $[0,n)$, hence after the final application of sorting subnetworks on the trees, the result is fully sorted.

In the general case, we pick a permutation π on n such that $X_{\pi(i)} \le X_{\pi(j)}$ for each $i \le j$ by Lemma 2.6. Put $x_i = \pi^{-1}(i)$, and $F(i) = X_{\pi(i)}$. Clearly $F(\vec{x}) = \vec{X}$, and $\mathrm{eval}(N(n), \le, \vec{x}) = \langle 0,\dots,n-1\rangle$ by the first part of the proof, hence $\mathrm{eval}(N(n), \le, \vec{X}) = \langle F(0),\dots,F(n-1)\rangle = \langle X_{\pi(i)} \mid i < n\rangle$ by Lemma 2.4. □

5 Monotone sequent calculus

The monotone sequent calculus *MLK* is the fragment of the usual Gentzen propositional sequent calculus *LK* where we allow only sequents consisting of monotone formulas, i.e., propositional formulas built using the connectives $\{\wedge, \vee, \bot, \top\}$. The calculus thus uses structural rules, the initial rule (axiom), the cut rule, and left and right introduction rules for $\wedge$, $\vee$, $\bot$, and $\top$. Its introduction was originally motivated by results in circuit complexity [12, 3] showing exponential lower bounds on the size of monotone circuits; the hope was that these can be transformed to an exponential separation between *MLK* and *LK*. Atserias et al. [4] proved that this is not the case, as *MLK* quasipolynomially simulates *LK*:

Theorem 5.1 ([4]) *A monotone sequent in n variables which has an LK-proof of size s has also an MLK-proof of size $s^{O(1)}n^{O(\log n)}$ with $s^{O(1)}$ lines.* □

It remains an open problem whether we can improve this quasipolynomial simulation to a p-simulation, i.e., whether there exists a polynomial-time algorithm transforming an *LK*-proof of a monotone sequent to an *MLK*-proof of the same sequent. Atserias et al. [4] suggested the following approach to attack the problem, relying on a construction of suitable monotone formulas for the threshold functions

$$\theta^n_m(x_0, \dots, x_{n-1}) = 1 \Leftrightarrow \mathrm{card}\{i \mid x_i = 1\} \geq m.$$

Theorem 5.2 ([4]) *Assume that there are monotone formulas $T^n_m(p_0, \dots, p_{n-1})$ for $m \leq n+1$ such that the formulas*

$$T^n_0(p_0, \dots, p_{n-1}) \tag{1}$$

$$\neg T^n_{n+1}(p_0, \dots, p_{n-1}) \tag{2}$$

$$T^n_m(p_0, \dots, p_{k-1}, \bot, p_{k+1}, \dots, p_{n-1}) \to T^n_{m+1}(p_0, \dots, p_{k-1}, \top, p_{k+1}, \dots, p_{n-1}) \tag{3}$$

for $m \leq n$, $k < n$ have LK-proofs constructible in time $n^{O(1)}$. Then MLK p-simulates LK-proofs of monotone sequents. □

A remarkable feature of Theorem 5.2 is that in the conclusion we construct *MLK*-proofs from *LK*-proofs, nevertheless in the assumption we only require the existence of *LK*-proofs. This significantly broadens the range of methods admissible for proving (1)–(3), and in particular, we can use propositional translations of proofs in bounded arithmetic.

Recall that the sequent calculus *LK* is p-equivalent to Frege systems: these are proof systems given by a sound and implicationally complete finite set of rules of the form $\varphi_1, \dots, \varphi_n/\varphi$, such that a Frege proof of φ is a sequence of formulas ending with φ where each formula is derived from previous formulas by a substitution instance of a basic rule. As shown in [8], NC^1_*-formulas provable in VNC^1_* translate to families of propositional tautologies with polynomial-time Frege proofs. The translation works as follows. For each NC^1_*-formula $\varphi(x_1, \dots, x_r, X_1, \dots, X_s)$ (i.e., $\varphi \in \Sigma^B_0(L_{\overline{VNC^1_*}})$) and natural numbers $n_1, \dots, n_r, m_1, \dots, m_s$, we define a propositional formula

$$[\![\varphi(\vec{x}, \vec{X})]\!]_{\vec{n},\vec{m}}(p_{1,0}, \dots, p_{1,m_1-1}, \dots, p_{s,0}, \dots, p_{s,m_s-1}).$$

Let $X_1, \dots, X_s$ be sets such that $|X_i| \leq m_i$, and let $\tilde{X}_i$ denote the propositional assignment which gives the value 1 to the variable $p_{i,k}$ iff $k \in X_i$. Then the translation satisfies

$$[\![\varphi]\!]_{\vec{n},\vec{m}}(\tilde{X}_1, \dots, \tilde{X}_s) = 1 \Leftrightarrow \mathbb{N} \vDash \varphi(\vec{n}, \vec{X}). \tag{$*$}$$

In particular, if (the universal closure of) φ is valid in $\mathbb{N}$, then $[\![\varphi]\!]_{\vec{n},\vec{m}}$ is a sequence of tautologies. The translation of compound formulas is defined by

$$\begin{aligned}
[\![\varphi \circ \psi]\!]_{\vec{n},\vec{m}} &= [\![\varphi]\!]_{\vec{n},\vec{m}} \circ [\![\psi]\!]_{\vec{n},\vec{m}}, \qquad \circ \in \{\wedge, \vee, \neg\}, \\
[\![\exists x \le t\, \varphi]\!]_{\vec{n},\vec{m}} &= \bigvee_{k \le b_t(\vec{n},\vec{m})} ([\![x \le t]\!]_{k,\vec{n},\vec{m}} \wedge [\![\varphi]\!]_{k,\vec{n},\vec{m}}), \\
[\![\forall x \le t\, \varphi]\!]_{\vec{n},\vec{m}} &= \bigwedge_{k \le b_t(\vec{n},\vec{m})} ([\![x \le t]\!]_{k,\vec{n},\vec{m}} \to [\![\varphi]\!]_{k,\vec{n},\vec{m}}),
\end{aligned}$$

where b_t is a suitable L_0-term such that $t(\vec{n}, \vec{X}) \le b_t(\vec{n}, \vec{m})$ whenever $|X_i| \le m_i$ for each i. The definition of $[\![\varphi]\!]$ for atomic formulas φ is more tedious and involves translation of terms as well as formulas, but it proceeds in a more-or-less expected way, we refer the reader to [8] for details.

Theorem 5.3 ([8]) *If φ is an NC^1_*-formula such that $VNC^1_* \vdash \varphi$, then the tautologies $[\![\varphi]\!]_{\vec{n},\vec{m}}$ have Frege proofs constructible in time poly$(\vec{n}, \vec{m})$.* □

Sorting a 0-1 input amounts to counting the number of ones, hence the AKS network evaluated on a 0-1 input gives monotone circuits for threshold functions of logarithmic depth, which can be unwinded into polynomial-size formulas. Since fundamental properties of the network are provable in VNC^1_*, we can use Theorem 5.3 to construct polynomial-time Frege proofs of (1)–(3). We proceed with the details.

Let $N(n)$ be the NC^1_*-function computing a log-depth sorting network as in Theorem 4.12. Let $\varphi(n, e, f, h, l)$ be an NC^1_*-formula expressing that there exists a comparator in $N(n)$ whose input edges are e, f, and whose higher and lower output edges are h and l, respectively. Using Lemma 2.3, there is an NC^1_*-formula $\psi(n, e, X)$ expressing that edge e in $N(n)$ evaluates to 1 on a 0-1 input X. Finally, let $\chi(n, i, X)$ denote the NC^1_*-formula $i \in \mathrm{eval}(N(n), \le, X)$. We define a monotone propositional formula $A_{n,e}$ for each edge e of $N(n)$ as follows. If e is the outgoing edge of the ith input node, we put

$$A_{n,e} = p_i.$$

If $\varphi(n, e, f, h, l)$, we define

$$\begin{aligned}
A_{n,h} &= A_{n,e} \vee A_{n,f}, \\
A_{n,l} &= A_{n,e} \wedge A_{n,f}.
\end{aligned}$$

Notice that the depth of $A_{n,e}$ is the depth of e in $N(n)$, which is bounded by $O(\log n)$, thus $A_{n,e}$ has polynomial size. If $0 < m \le n$, and e is the incoming edge of the $(n-m)$th output node of $N(n)$, we put

$$T^n_m = A_{n,e}.$$

We also define

$$\begin{aligned}
T^n_{n+1} &= \bot, \\
T^n_0 &= \top.
\end{aligned}$$

Lemma 5.4 *There are polynomial-time Frege proofs of the formulas*

$$T^n_m \leftrightarrow [\![m \le \operatorname{card} X]\!]_{m,n}.$$

Proof: As VNC^1_* proves the formula

$$\alpha = \varphi(n,e,f,h,l) \to [(\psi(n,h,X) \leftrightarrow \psi(n,e,X) \vee \psi(n,f,X)) \\ \wedge (\psi(n,l,X) \leftrightarrow \psi(n,e,X) \wedge \psi(n,f,X))],$$

its translation $[\![\alpha]\!]_{n,e,f,h,l,n}$ has polynomial-time Frege proofs. If e,f,h,l are the respective input and output edges of a comparator in $N(n)$, then $[\![\varphi]\!]_{n,e,f,h,l}$ is a true Boolean sentence, hence it has a polynomial-time Frege proof. We obtain proofs of the formulas

$$[\![\psi]\!]_{n,h,n} \leftrightarrow [\![\psi]\!]_{n,e,n} \vee [\![\psi]\!]_{n,f,n},$$
$$[\![\psi]\!]_{n,l,n} \leftrightarrow [\![\psi]\!]_{n,e,n} \wedge [\![\psi]\!]_{n,f,n}.$$

If e is the outgoing edge of the ith input node in $N(n)$, and f is the incoming edge of the ith output node, we can similarly construct proofs of

$$[\![\psi]\!]_{n,e,n} \leftrightarrow p_i,$$
$$[\![\psi]\!]_{n,f,n} \leftrightarrow [\![\chi(n,i,X)]\!]_{n,i,n}.$$

Then we can construct proofs of

$$A_{n,e} \leftrightarrow [\![\psi]\!]_{n,e,n}$$

by induction on the depth of e, and we derive

$$T^n_m \leftrightarrow [\![\chi]\!]_{n,n-m,n}$$

for $0 < m \le n$. By Theorem 4.12 and the proof of Lemma 2.6, VNC^1_* proves

$$|X| \le n \to (\chi(n,i,X) \leftrightarrow \operatorname{card} X \ge n-i).$$

As there are short proofs of $[\![|X| \le n]\!]_{n,n}$, we obtain short proofs of

$$[\![\chi]\!]_{n,n-m,n} \leftrightarrow [\![\operatorname{card} X \ge m]\!]_{m,n},$$

and we conclude

$$T^n_m \leftrightarrow [\![\operatorname{card} X \ge m]\!]_{m,n}$$

for $0 < m \le n$. The cases $m = 0$ and $m = n+1$ follow from translation of the formulas

$$\operatorname{card} X \ge 0,$$
$$|X| \le n \to \neg(\operatorname{card} X \ge n+1),$$

provable in VNC^1_*. □

Theorem 5.5 *Under Assumption* 3.2*, the monotone sequent calculus MLK p-simulates LK-proofs of monotone sequents.*

Proof: In view of Theorem 5.2, it suffices to construct polynomial-time Frege proofs of (1)–(3) for the formulas T^n_m defined above. (1) and (2) are trivial. VNC^1_* proves

$$\forall u < n\,(u \neq k \wedge X(u) \to Y(u)) \wedge \neg X(k) \wedge Y(k) \wedge |Y| \leq n \\ \to (m \leq \operatorname{card} X \to m + 1 \leq \operatorname{card} Y),$$

thus its (slightly simplified) propositional translation

$$\bigwedge_{\substack{u<n \\ u\neq k}} (p_u \to q_u) \wedge \neg p_k \wedge q_k \to (\llbracket u \leq \operatorname{card} X\rrbracket_{m,n}(\vec{p}) \to \llbracket u \leq \operatorname{card} X\rrbracket_{m+1,n}(\vec{q}))$$

for $n \in \omega$, $m \leq n$, and $k < n$ has poly-time constructible Frege proofs. We substitute $\bot$ for p_k, $\top$ for q_k, and p_u for q_u, $u \neq k$, in the proof. We obtain

$$\llbracket u \leq \operatorname{card} X\rrbracket_{m,n}(p_0, \dots, p_{k-1}, \bot, p_{k+1}, \dots, p_{n-1}) \\ \to \llbracket u \leq \operatorname{card} X\rrbracket_{m+1,n}(p_0, \dots, p_{k-1}, \top, p_{k+1}, \dots, p_{n-1}),$$

from which we derive

$$T^n_m(p_0, \dots, p_{k-1}, \bot, p_{k+1}, \dots, p_{n-1}) \to T^n_{m+1}(p_0, \dots, p_{k-1}, \top, p_{k+1}, \dots, p_{n-1})$$

using Lemma 5.4. □

References

[1] Miklós Ajtai, János Komlós, and Endre Szemerédi, *Sorting in $c \log n$ parallel steps*, Combinatorica 3 (1983), no. 1, pp. 1–19.

[2] ———, *An $O(n \log n)$ sorting network*, in: Proceedings of the 15th Annual ACM Symposium on Theory of Computing, 1983, pp. 1–9.

[3] Noga Alon and Ravi B. Boppana, *The monotone circuit complexity of Boolean functions*, Combinatorica 7 (1987), no. 1, pp. 1–22.

[4] Albert Atserias, Nicola Galesi, and Pavel Pudlák, *Monotone simulations of non-monotone proofs*, Journal of Computer and System Sciences 65 (2002), no. 4, pp. 626–638.

[5] Kenneth E. Batcher, *Sorting networks and their applications*, in: Proceedings of the AFIPS Spring Joint Computer Conference, vol. 32, 1968, pp. 307–314.

[6] Stephen A. Cook and Antonina Kolokolova, *A second-order theory for NL*, in: Proceedings of the 19th Annual IEEE Symposium on Logic in Computer Science, 2004, pp. 398–407.

[7] Stephen A. Cook and Phuong Nguyen, *Logical foundations of proof complexity*, book in preparation, http://www.cs.toronto.edu/~sacook/homepage/book/.

[8] Emil Jeřábek, *On theories of bounded arithmetic for NC^1*, preprint, 2008.

[9] Michal Koucký, Valentine Kabanets, and Antonina Kolokolova, *Expanders made easy: The combinatorial analysis of an expander construction*, unpublished manuscript, 2007.

[10] Noam Nisan and Amnon Ta-Shma, *Symmetric Logspace is closed under complement*, in: Proceedings of the 27th Annual ACM Symposium on Theory of Computing, 1995, pp. 140–146.

[11] Michael S. Paterson, *Improved sorting networks with $O(\log N)$ depth*, Algorithmica 5 (1990), no. 1, pp. 75–92.

[12] Alexander A. Razborov, *Lower bounds on the monotone complexity of some Boolean functions*, Mathematics of the USSR, Doklady 31 (1985), pp. 354–357.

[13] Walter L. Ruzzo, *On uniform circuit complexity*, Journal of Computer and System Sciences 22 (1981), no. 3, pp. 365–383.

[14] Domenico Zambella, *Notes on polynomially bounded arithmetic*, Journal of Symbolic Logic 61 (1996), no. 3, pp. 942–966.