Expander Construction in VNC¹

Sam Buss* Valentine Kabanets† Antonina Kolokolova‡ Michal Koucký§

December 30, 2016

Abstract

We give a combinatorial analysis (using edge expansion) of a variant of the iterative expander construction due to Reingold, Vadhan, and Wigderson [RVW02], and show that this analysis can be formalized in the bounded-arithmetic system VNC¹ (corresponding to the “NC¹ reasoning”). As a corollary, we prove the assumption made by Jeřábek [Jeř11b] that a construction of certain bipartite expander graphs can be formalized in VNC¹. This in turn implies that every proof in Gentzen’s sequent calculus LK of a monotone sequent can be simulated in the monotone version of LK (MLK) with only polynomial blowup in proof size, strengthening the quasipolynomial simulation result of Atserias, Galesi, and Pudlák [AGP02].

Keywords: expander graphs, bounded arithmetic, VNC¹, sequent calculus LK, monotone LK (MLK)

---

*Department of Mathematics, University of California San Diego, La Jolla, CA, USA, sbuss@ucsd.edu. Supported in part by NSF grant CCF-1213151. Part of the work on VNC¹ was done while Buss was visiting the Chebychev Laboratory, St.Petersburg State University in Spring 2016, supported in part by Skolkovo Institute of Science and Technology.

†School of Computing Science, Simon Fraser University, Burnaby, BC, Canada, kabanets@cs.sfu.ca. Supported in part by an NSERC Discovery grant. Part of the work was done while visiting UCSD.

‡Department of Computer Science, Memorial University of Newfoundland, St. John’s, NL, Canada, kol@cs.mun.ca. Supported in part by an NSERC Discovery grant. Part of the work was done while visiting UCSD.

§Computer Science Institute, Charles University, Prague, Czech Republic, koucky@iuuk.mff.cuni.cz. The research leading to these results has received funding from the European Research Council under the European Unions Seventh Framework Programme (FP/2007-2013)/ERC Grant Agreement n. 616787.
# Contents

1 **Introduction**  
1.1 Our results  
1.2 Relation to previous work  
1.2.1 Expander constructions  
1.2.2 Bounded arithmetic  

2 **Preliminaries**  
2.1 Notation  
2.2 Expanders  
2.3 Bounded arithmetic theory $\text{VNC}^1$  
2.4 LK and MLK proof systems  

3 **Constructing edge expanders**  
3.1 Graph operations  
3.2 Effect of graph operations on edge expansion  
3.3 Construction  
3.4 Explicitness of the construction  

4 **Effect of graph operations on edge expansion: Proofs**  
4.1 Graph powering  
4.1.1 Edge expansion implies fast mixing  
4.1.2 Mixing implies edge expansion  
4.2 Tensor product  
4.3 Replacement product  

5 **Constructing bipartite vertex expanders**  

6 **Formalizing the construction in bounded arithmetic**  
6.1 Defining $\text{NC}^1$ functions within $\text{VNC}^1$  
6.2 A modified tree recursion  
6.3 A conservation result  
6.4 Expressing expander graph properties in $\text{VNC}^1$  
6.5 Formalizing Cauchy-Schwarz  
6.6 Formalizing edge expansion properties in $\text{VNC}^1$  

7 **Application to monotone sequent calculus**  

8 **Conclusions and open problems**
1 Introduction

Expander graphs have become one of the most useful combinatorial objects in theoretical computer science, with many beautiful applications in computer science and mathematics [HLW06], and responsible for several breakthroughs in computational complexity [Rei08, Din07]. These graphs have seemingly contradictory properties: sparseness and high connectivity. The high connectivity can be measured in a number of different, but essentially equivalent ways: vertex expansion (every small subset of vertices “expands”, i.e., has a larger neighborhood), edge expansion (every small subset of vertices has many edges leaving the set), or fast mixing time (a random walk on a regular expander graph quickly converges to the uniform distribution on vertices).

The existence of expander graphs of constant degree can be argued nonconstructively using a simple probabilistic argument: for any constant $d \geq 3$, a random $d$-regular graph is almost surely an expander [Pin73]. Constructing such graphs efficiently deterministically is much more difficult. The first explicit constructions were given by Margulis [Mar73] and Gabber and Galil [GG81]. Lubotzky, Phillips, and Sarnak [LPS88] gave a construction of expanders with particularly interesting properties, called Ramanujan graphs. All of these constructions are algebraic in nature: a graph is defined using a certain algebraic object (e.g., a group). Moreover, the analysis of correctness of the constructions is also algebraic. It relies on the algebraic notion of high connectivity called the eigenvalue gap and defined as follows. Consider the adjacency matrix of a given undirected $d$-regular graph, compute its eigenvalues, and order them according to the absolute value. It can be easily checked that $d$ is the largest value. The difference between $d$ and (the absolute value of) the second largest eigenvalue is the eigenvalue gap. The bigger this eigenvalue gap, the more connected the graph is. From this point of view, a $d$-regular expander is a graph with the eigenvalue gap at least $\Omega(d)$, i.e., the second largest eigenvalue should be at most some constant fraction of the degree.

A simpler, fully combinatorial construction of constant-degree expanders was given by Reingold, Vadhan, and Wigderson [RVW02]. They start with constant-size expander graphs (which can found by brute-force search), and iteratively apply certain graph operations that increase the size of the graph while preserving its expansion property. This way, one can quickly construct an expander graph of any given size. While the construction of [RVW02] is combinatorial, its analysis is still algebraic and is based on estimating the eigenvalue gap. Alon, Schwartz, and Shapira [ASS08] gave a different construction of expanders, which combines algebraically constructed expanders of Alon and Roichman [AR94] with only two applications of a certain graph operation (replacement product), to obtain a constant-degree expander of arbitrary size. They also gave a fully combinatorial analysis of the replacement product operation they used in the second stage of the construction. Their full analysis, however, is still algebraic, as it relies on the algebraic construction and the eigenvalue gap analysis of [AR94]. In this respect, the situation in [ASS08] is similar to that in [RVW02] where the analysis of a related graph operation (zig-zag product) can be done in terms of min-entropy, while the analysis of the complete construction is still based on eigenvalues.

The focus of our paper is to give a construction of expanders with a simple analysis, where simplicity is measured in terms of the power of a system of bounded arithmetic needed to formalize the analysis. Informally, systems of bounded arithmetic are obtained by restricting the power of the standard first-order theory of Peano arithmetic. It is possible to devise systems of bounded arithmetic that correspond to systems of reasoning using only concepts from a given complexity class, e.g., $P$ or $NC^1$. A natural question is: what is the weakest complexity class so that the existence of expander graphs can be proved using only the concepts of that complexity class?

The known expander constructions mentioned above can be formalized within a system of polytime reasoning, intuitively because eigenvalues and matrix determinants are known to be computable in polytime.
Our main result is a construction of expanders that can be formalized within a system of $\text{NC}^1$ reasoning, $\text{VNC}^1$ (see below for a formal definition). As $\text{NC}^1$ algorithms are not known to compute the eigenvalues or determinant of a given matrix, any such formalization of an expander construction in $\text{VNC}^1$ must necessarily avoid the use of eigenvalues, and hence be “combinatorial” in that sense.

As expanders are used in a number of complexity-theoretic results, formalizing the expander construction within a weak system of bounded arithmetic is an important step in formalizing these complexity-theoretic results within the bounded-arithmetic framework, which in turn may have other implications. For example, in proof complexity, we can use our expander construction to argue that any Gentzen’s sequent calculus $\text{LK}$ proof (of a monotone sequent) can be simulated by a monotone $\text{LK}$ ($\text{MLK}$) proof, with only polynomial blowup in proof size, improving upon the quasipolynomial simulation shown by Atserias, Galesi and Pudlák [AGP02], and answering a question of Pudlák and Buss [PB94]. This simulation result follows by the work of Jeřábek [Jeř11b] who proved the result under the assumption that a certain expander graph family can be proved to exist within a system of $\text{NC}^1$ reasoning. Our paper proves a strengthening of the assumption needed by Jeřábek.

1.1 Our results

Our main contribution is the analysis of one of the iterative expander constructions from [RVW02], which we show to be formalizable in the bounded-arithmetic system $\text{VNC}^1$ (of $\text{NC}^1$ reasoning). As in [RVW02], the expander construction is fully explicit in the sense that there is a deterministic polynomial-time algorithm that, given a vertex name $v$ in binary and a number $i$, outputs the value of the rotation map $\text{Rot}(v, i) = (w, j)$, where $w$ is the name of the $i$th neighbor of $v$ in the graph, and $j$ is the number such that $v$ is the $j$th neighbor of $w$. Moreover, we show that there is an alternating linear-time algorithm that accepts exactly the triples of the form $\langle v, i, \text{Rot}(v, i) \rangle$; this kind of explicitness is what we will use to argue that the expander construction is formalizable in $\text{VNC}^1$.

**Theorem 1.1** (Main result: Informal version). The existence of an expander graph family can be proved using $\text{NC}^1$ reasoning only (within the system $\text{VNC}^1$).

As our main application, building on Jeřábek [Jeř11b] and Atserias, Galesi and Pudlák [AGP02], we show that every proof in Gentzen’s sequent calculus $\text{LK}$ of a monotone sequent can be simulated by a monotone $\text{LK}$ ($\text{MLK}$) proof (a sequent calculus proof in which all formulas are positive) with only polynomial blowup in size. This answers a question of Pudlák and Buss [PB94]. Previously, [AGP02] showed such simulation with quasipolynomial blowup in proof size.

**Theorem 1.2** (Main application). $\text{MLK}$ polynomially simulates $\text{LK}$ on monotone sequents.

It is easy to show that the intuitionistic propositional sequent calculus $\text{LJ}$ polynomially simulates $\text{MLK}$ (see Pudlák [Pud99] and Bilková [Bíl01]); thus we get as an immediate corollary that propositional $\text{LJ}$ polynomially simulates $\text{LK}$ on monotone sequents, re-proving the result of Jeřábek [Jer09b, Theorem 3.9]. Many of the principles that have been considered in propositional proof complexity are expressed as monotone sequents, notably the pigeonhole principle and the clique-coloring tautologies. As these principles have polynomial size $\text{LK}$ proofs [Bus87], Theorem 1.2 implies that they also have polynomial size proofs in $\text{MLK}$ as well as in propositional $\text{LJ}$. The prior best known results for the pigeonhole principle were the quasipolynomial size $\text{MLK}$ proofs of Atserias, Galesi and Gavaldà [AGG01].

It remains an open problem whether tree-like $\text{MLK}$ can polynomially simulate $\text{MLK}$, equivalently whether tree-like $\text{MLK}$ can polynomially simulate $\text{LK}$ on monotone sequents. Note that [AGP02] give a quasipolynomial simulation.
Intuitively, to simulate an LK proof within MLK, one needs to construct (and prove correctness of) a monotone formula for the majority function. Such monotone formulas can be built using the classical AKS sorting networks [AKS83]. Jeřábek [Jeř11b] shows that the analysis of AKS sorting networks can be formalized within a certain system of NC\(^1\) reasoning (slightly more powerful than VNC\(^1\)), under the assumption that the existence of expander graphs, with certain parameters, is also formalizable within the same system. Our Theorem 1.1 proves the assumption needed by Jeřábek (actually a slightly stronger version, as our proof of the existence of expanders is in the weaker system VNC\(^1\)), and so Theorem 1.2 immediately follows.

1.2 Relation to previous work

1.2.1 Expander constructions

The expander graph construction that we analyze is a variant of the iterative construction of expanders given in [RVW02]. The idea is to start with a constant-size expander graph (found, say, by exhaustive search), and iteratively increase the size of the graph while keeping its expansion larger than some universal constant. The notion of expansion used by [RVW02] is in terms of the eigenvalue gap. To analyze the expansion of the final graph, Reingold, Vadhan, and Wigderson [RVW02] bound the effect of the graph operations they used (graph powering, graph tensoring, and zig-zag product) on the second largest eigenvalue of the adjacency matrix of the resulting graph. The analysis of graph powering (where an edge of the \(k\)th power of a graph \(G\) is a walk of length \(k\) in \(G\)) and graph tensoring (where an edge of the tensor product of \(G\) and \(H\) consists of a pair of edges, one from \(G\) and one from \(H\)) is immediate from the basic linear algebra. The analysis of the zig-zag product (a way to compose a graph \(G\) with a graph \(H\) so that the new graph has the degree of \(H\)) is technically the most difficult part of the algebraic analysis of the expander construction in [RVW02].

In [ASS08], a graph replacement product (closely related to the zig-zag product) is analyzed in terms of edge expansion, avoiding any mention of the eigenvalue gap. Since replacement product can be used instead of zig-zag product in an iterative expander construction along the lines of [RVW02], this gives a combinatorial analysis of the part of the expander construction. In order to make the entire analysis combinatorial, it suffices to analyze graph powering and graph tensoring also in terms of edge expansion. This is exactly what we do in the present paper.

Our combinatorial analysis of graph tensoring, though subtle, is not very difficult. For the analysis to go through, it turns out necessary to work with graphs that have sufficiently many self-loops around every vertex (at least half the degree). On the other hand, graph powering is much more difficult to analyze combinatorially. Fortunately, here we were able to use the result of Mihail [Mih89] who gave a combinatorial analysis of the mixing time of random walks on expanders in terms of edge expansion. (Interestingly, for her proof, she also had to work with graphs that have many self-loops around every vertex.) Finally, using Mihail’s bounds, we are able to conclude the analysis of graph powering in terms of edge expansion, borrowing some ideas from [AC88].

1.2.2 Bounded arithmetic

There is a long history of formalizing complexity results in bounded arithmetic; indeed, this was one of the main motivations for the definitions of bounded arithmetic. First, bounded arithmetic theories can capture a range of complexity classes, from uniform \(AC^0\) and uniform \(NC^1\), to polynomial time, polynomial space and exponential time (see [Bus86, CN10]). Second, via the Paris-Wilkie or Cook translations, proofs in bounded arithmetic can be viewed as uniform families of propositional proofs. For this reason, a proof in
bounded arithmetic can sometimes yield new propositional proofs.

There has been considerable progress in formalizing advanced results from computational complexity in weak theories of bounded arithmetic; these include approximate counting, randomized computations, and Arthur-Merlin games [Jeř07, Jeř09a], Toda’s theorem [BKZ15], and the PCP theorem [Pic15]. The present paper continues this tradition by formalizing the construction of expander graphs in the weak fragment VNC$^1$ which corresponds to NC$^1$ computation.

There are a number of prior works which use bounded arithmetic to obtain upper bounds in proof complexity. A big advantage of using bounded arithmetic is that the proofs can be considerably simplified. A classic example is the work by Paris and Wilkie [PW81] who showed that the proofs of the weak pigeonhole principle in I$\Delta_0$ constructed by [PWW88] yield constant-depth, polynomial-size Frege proofs of the propositional translations of the weak pigeonhole principle (via the “Paris-Wilkie translation”). Lower-depth, quasipolynomial-size Frege proofs were later given by [MPW02] via a proof of the weak pigeonhole principle in a different fragment of bounded arithmetic. Similarly, [Pud92] gave proofs of Ramsey’s theorem in $S_2$, and these translate into quasipolynomial-size, constant-depth Frege proofs. Recently, [BKZ15] used formalization of Toda’s theorem in bounded arithmetic with modular counting quantifiers to show that constant-depth AC$^0(p)$-proofs, for $p$ a prime, can be translated into depth-three propositional proofs, with formulas being Boolean combinations of mod $p$ gates of small conjunctions. Another classic example is Cook’s theorem that extended Frege proofs have polynomial size proofs of their partial consistency statements, which was established via provability in PV [Coo75].

The present paper establishes a new result of this type via a Cook-style translation: together with earlier work of Jeřábek [Jeř11a], our formalization of expander graphs in VNC$^1$ implies that the monotone propositional proof system MLK polynomially simulates the proof system LK. We will use the system VNC$^1$ defined by Cook and Morioka [CM05]. We conservatively extend VNC$^1$ to facilitate reasoning about the compositions of NC$^1$ functions, which allows us to simplify the formalization of our recursive expander construction.

**Remainder of the paper** Section 2 contains basic definitions. Our expander construction is defined in Section 3. The analysis of the relevant graph operations in terms of edge expansion is given in Section 4. In Section 5, we present a construction of bipartite expanders needed by Jeřábek [Jeř11b]. In Section 6, we show that the existence of our expander graphs is provable in VNC$^1$, thereby proving a formal version of Theorem 1.1. We derive Theorem 1.2 in Section 7. Section 8 contains concluding remarks.

## 2 Preliminaries

### 2.1 Notation

We consider undirected graphs, possibly with parallel edges and self-loops. For an undirected graph $G = (V, E)$ on $n$ nodes, we usually associate the vertex set $V$ with the set $[n] = \{1, 2, \ldots, n\}$, and denote an edge $i \sim j$ between nodes $i$ and $j$ as $\{i, j\} \in E$. In this notation, we also allow self-loops $\{i, i\} \in E$.

The summation $\sum_{\{i, j\} \in E}$ means the sum over all edges in $E$, including parallel edges and self-loops, where each edge $e \in E$ is considered exactly once. With some fixed ordering on the nodes of $G$, we can sum over all its edges between distinct vertices (i.e., excluding self-loops), using the summation $\sum_{\{i < j\} \in E}$; that views each edge in $E$ between nodes $i$ and $j$ as going from the smaller to the large vertex, allowing for parallel edges (i.e., counting each edge from $i$ to $j$ with multiplicity).
The adjacencies of a $d$-regular graph $G$ are given via its rotation map $Rot_G$ so that, for vertex $v$ of $G$ and an index $i \in [d]$, we have $Rot_G(v, i) = (w, j)$ if $w$ is the $i$th neighbor of $v$, and $v$ is the $j$th neighbor of $w$; so, in particular, the rotation map induces some fixed numbering of neighbors of a given vertex.

For an $n$-vertex graph $G$, its adjacency matrix is an $n \times n$ matrix $A'$ whose $(i, j)$th entry contains the number of edges between vertices $i$ and $j$ in $G$. For $d$-regular graphs $G$, it will be more convenient for us to consider the normalized adjacency matrix defined as $\frac{1}{d} \cdot A'$. Note that the normalized adjacency matrix $A$ of $G$ is the probability transition matrix for a random walk on $G$. That is, if $\pi$ is a probability distribution on vertices of $G$, then $A \pi$ is the probability distribution induced by one step of a random walk on $G$ starting from a vertex distributed according to $\pi$. It is also easy to see that $A^k$ is the normalized adjacency matrix of the graph $G^k$.

For a vector $v = (v_1, \ldots, v_n)$, its squared $\ell_2$-norm is $\|v\|^2 = \sum_{i=1}^{n} v_i^2$. For two vectors $v = (v_1, \ldots, v_n)$ and $w = (w_1, \ldots, w_n)$, their inner product is $\langle v, w \rangle = \sum_{i=1}^{n} v_i w_i$. The Cauchy-Schwarz inequality asserts that $\langle v, w \rangle^2 \leq \|v\|^2 \cdot \|w\|^2$.

We think of vectors as column vectors, and so we use $Av$ to denote the multiplication of a matrix $A$ by a column vector $v$. However, we will often abuse the notation and also write $vA$ instead of the more proper notation $v^T A$ (where $v^T$ denotes the transpose of $v$). It will be clear from the context whether a vector is a column or row vector.

### 2.2 Expanders

For a graph $G = (V, E)$ and a subset $U \subseteq V$ of vertices, we denote by $\overline{U}$ the set $V \setminus U$, and by $E(U, \overline{U})$ the set of edges between $U$ and $\overline{U}$. The edge expansion of a $d$-regular graph $G = (V, E)$ on $n$ vertices is defined as

$$\min_{\emptyset \neq U \subseteq V, |U| \leq n/2} \frac{|E(U, \overline{U})|}{d \cdot |U|} = \min_{\emptyset \neq U \subseteq V} \frac{|E(U, \overline{U})|}{d \cdot \min\{|U|, |\overline{U}|\}}.$$  \hspace{1cm} (1)

For a graph $G = (V, E)$ and a subset $U \subseteq V$ of vertices, we denote by $\Gamma_G(U)$ the set of all neighbors of $U$ in $G$, i.e.,

$$\Gamma_G(U) = \{v \in V \mid \exists u \in U, \{u, v\} \in E\}.$$  We drop the subscript $G$ if the graph $G$ is understood from the context. We denote by $\Gamma^+(U)$ the set $\Gamma(U) \setminus U$ of new neighbors of $U$. The vertex expansion of a graph $G = (V, E)$ on $n$ vertices is defined as

$$\min_{\emptyset \neq U \subseteq V, |U| \leq n/2} \frac{|\Gamma^+(U)|}{|U|}.$$  

### 2.3 Bounded arithmetic theory $\text{VNC}^1$

A number of bounded arithmetic theories have been proposed for uniform $\text{NC}^1$: these include the theory $\text{A}^\log$ of Clote and Takeuti [CT92], the theory AID of Arai [Ara00], the theory $\text{VNC}^1$ of Cook and Morioka [CM05], and a reformulated version of $\text{VNC}^1$ by Cook and Nguyen [CN10]. Jeřábek [Jeř11a] describes a theory $\text{VNC}^1_r$ for $\text{NC}^1$ under a relaxed notion of uniformity for logarithmic depth circuits.

Cook and Morioka [CM05] define $\text{VNC}^1$ using tree recursion ($\text{TreeRec}$). Cook and Nguyen [CN10] give an equivalent definition of $\text{VNC}^1$ using the Boolean formula value problem. It is easier to formalize the expander graph construction with tree recursion, so we work with the version of $\text{VNC}^1$ as defined by Cook and Morioka [CM05].

The bounded arithmetic theory $\text{VNC}^1$ is an extension of the theory $\text{V}^0$ of bounded arithmetic; $\text{V}^0$ corresponds in power to $\text{AC}^0$. $\text{V}^0$ is a second-order (two-sorted) system of arithmetic, with two sorts of numbers
(first-order objects) and strings (second order objects). Strings are viewed as members of \( \{0, 1\}^* \). The notation \( X(i) \), where \( X \) is a string and \( i \geq 0 \) is a natural number, means the Boolean value of the \( i \)th entry in string \( X \). Sometimes \( i \in X \) is written instead of \( X(i) \). The constants 0 and 1 are number terms, and addition and multiplication are number functions. Another term of type number is string length \( |X| \), defined to be the value of the largest element in \( X \) when viewed as a set plus 1. Addition and multiplication are defined for numbers only, and equality is defined both for numbers and strings. The axioms of \( \mathcal{V}^0 \) consist of a finite set of “\( \text{BASIC} \)” open axioms defining simple properties of the constants, relation symbols and function symbols, plus \( \Sigma^B_0 \)-Comprehension axioms

\[
\Sigma^B_0\text{-COMP:}\quad \exists X \leq y \forall z < y (X(z) \leftrightarrow \varphi(z))
\]

for any formula \( \varphi \) in \( \Delta^B_0 \) not containing \( X \) as a free variable, but possibly containing free variables other than \( z \). A \( \Delta^B_0 \)-formula is one in which all quantifiers are bounded and which contains no second-order quantifiers. We write \((\exists X \leq y)\psi\) for \( \exists X((|X| \leq y) \land \psi) \).

Let \( \varphi(i, \bar{x}, \bar{X})[p, q] \) and \( \psi(i, \bar{x}, \bar{X}) \) be \( \Sigma^B_0 \)-formulas. The notation “\([p, q]\)” indicates that \( p \) and \( q \) are propositional variables that may occur as atomic subformulas in \( \varphi \). The \( \Sigma^B_0\text{-TreeRec} \) property \([CM05]\) is defined by the formula \( B^{\varphi, \psi}(a, \bar{x}, \bar{X}, Z) \):

\[
(\forall i < a)((Z(a + i) \leftrightarrow \psi(i)) \land (Z(i) \leftrightarrow \varphi(i, \bar{x}, \bar{X})[Z(2i + 1), Z(2i + 2)]))
\]

For \( i < a \), this states that \( Z(i) \) is a Boolean function of the two values \( Z(2i + 1) \) and \( Z(2i + 2) \). Thus \( Z(i) \) is computed by a circuit which is formed as a binary tree with gate types specified by \( \varphi \) and input values specified by \( \psi \). We can always assume w.l.o.g. that \( a = 2^{\lceil a \rceil} - 1 \); we call this the “depth condition” and it means the binary tree is exactly balanced and of depth \( |a| \). This tree is of course a fanin two Boolean circuit. The type of the \( i \)-th gate is determined by \( \varphi(i, \bar{x}, \bar{X}) \) and thus is a \( \Sigma^B_0 \)-property of \( i \) and the inputs \( \bar{x} \) and \( \bar{X} \).

The term \( \mathcal{VNC}^1 \) is defined as \( \mathcal{V}^0 \) plus the \( \Sigma^B_0\text{-TreeRec} \) axioms \((\exists Z \leq 2a)B^{\varphi, \psi}(a, \bar{x}, \bar{X}, Z)\) for all \( \varphi \) and \( \psi \) in \( \Sigma^B_0 \). The language of \( \mathcal{VNC}^1 \) can be extended by adding a new relation symbol \( R^{\varphi, \psi}(i, a, \bar{x}, \bar{X}) \) for every formula \( B^{\varphi, \psi} \). The defining axioms for \( R^{\varphi, \psi} \) are

\[
B^{\varphi, \psi}(a, \bar{x}, \bar{X}, R^{\varphi, \psi}) \quad \text{ and } \quad i \geq 2a \rightarrow \neg R^{\varphi, \psi}(i, a, \bar{x}, \bar{X}).
\]

Note that the defining axioms uniquely specify all values of \( R^{\varphi, \psi} \), provably in \( \mathcal{VNC}^1 \). Adding the predicate symbols \( R^{\varphi, \psi} \) and their defining axioms to \( \mathcal{VNC}^1 \) yields the theory \( \mathcal{VNC}^1(\mathcal{L}\text{TreeRec}) \). As an extension by definitions, this theory is conservative over \( \mathcal{VNC}^1 \).

A key property of \( \mathcal{VNC}^1 \) is that it can \( \Sigma^B_1 \)-define precisely the (uniform) \( \mathcal{NC}^1 \) functions; this is discussed in Section 6.1.

### 2.4 LK and MLK proof systems

The system MLK of monotone reasoning in \([AGP02]\) is a variant of Gentzen’s sequent calculus LK in which all formulas are positive. An LK proof is a list of sequents of the form \( \varphi_1, \ldots, \varphi_n \rightarrow \psi_1, \ldots, \psi_m \), interpreted as \( \bigwedge_{i=1}^n \varphi_i \rightarrow \bigvee_{j=1}^m \psi_j \). The axioms are \( \varphi \rightarrow \varphi \), \( \Gamma \rightarrow 1 \), and \( 0 \rightarrow \Gamma \), for an arbitrary list of formulas \( \Gamma \). Let \( \varphi, \psi \) denote formulas and \( \Gamma, \Delta \) lists of formulas. The main derivation rules of LK are as follows.

- **Left derivation rules:**
  \[
  \begin{array}{ll}
  \varphi, \psi, \Gamma \rightarrow \Delta & \varphi, \Gamma \rightarrow \Delta \\
  (\varphi \land \psi), \Gamma \rightarrow \Delta & (\varphi \lor \psi), \Gamma, \Gamma' \rightarrow \Delta, \Delta' \\
  \end{array}
  \]

  \[
  \begin{array}{ll}
  \varphi, \Gamma \rightarrow \Delta & \varphi, \Delta \rightarrow \Delta \\
  \psi, \Gamma' \rightarrow \Delta' & \neg \psi, \Gamma \rightarrow \Delta
  \end{array}
  \]

\(^1\text{Cook and Morika } [CM05] \text{ call this theory } \mathcal{VNC}^1(\mathcal{L}\text{TreeRec}).\)
• Right derivation rules: 
  \[ \Gamma \to \Delta, \varphi, \psi \quad \Gamma \to \Delta, \varphi \quad \Gamma' \to \Delta', \psi \quad \varphi, \Gamma \to \Delta \]
  \[ \frac{\Gamma \to \Delta, (\varphi \lor \psi)}{\Gamma \to \Delta, \varphi \lor \psi} \quad \frac{\Gamma, \Gamma' \to \Delta, \Delta', (\varphi \land \psi)}{\Gamma, \Gamma' \to \Delta, \Delta', \varphi \land \psi} \quad \frac{\varphi, \Gamma \to \Delta}{\Gamma \to \Delta, \neg \varphi} \]

• Cut rule: 
  \[ \frac{\Gamma \to \Delta, \varphi \quad \varphi, \Gamma' \to \Delta'}{\Gamma, \Gamma' \to \Delta, \Delta'} \]

Additionally, LK includes structural rules on both sides of a sequent such as weakening, contraction of duplicate formulas, and changing order of formulas on the same side. LK is equivalent in power to Frege systems, and tree-like LK is equivalent to LK, thus VNC\(^1\) proofs translate into polynomial-size LK proofs [Ara00, CM05, CN10].

In Monotone LK (MLK), all formulas in the proof are over the \(\land, \lor\) basis with no \(\neg\).

3 Constructing edge expanders

Here we define an iterative construction of a constant-degree edge expander family, and argue its edge expansion properties using simple combinatorial tools. The simplicity of the analysis will allow us (in Section 6) to formalize it within the system VNC\(^1\). The construction is a variant of the iterative construction given by Reingold, Vadhan, and Wigderson [RVW02], using the graph operations described next.

3.1 Graph operations

We define the graph operations that we will use to construct expanders.

• [Powering] For a graph \(G = (V, E)\) and an integer \(k \geq 1\), the \(k\)th power \(G^k\) is the graph on vertices \(V\) where for each walk of length \(k\) from a vertex \(u\) to a vertex \(v\) in \(G\) there is an edge \(u \sim v\) in \(G^k\).

  If \(\text{Rot}_G\) is the rotation map of \(G\), then the rotation map of \(G^k\) is

  \[ \text{Rot}_{G^k}(v, (i_1, \ldots, i_k)) = (w, (j_k, \ldots, j_1)), \]

  where \(w\) is the vertex reached from \(v\) in \(G\) by edges \(i_1, \ldots, i_k\) using the rotation map \(\text{Rot}_G\), and \((j_k, \ldots, j_1)\) describes the same sequence of edges in reverse order from \(w\)’s point of view. For instance, \(\text{Rot}_{G^2}(v, i_1) = (v', j_1)\) for some \(v' \in V\), then \(\text{Rot}_{G^2}(v', i_2) = (v'', j_2)\) for some \(v''\), etc.

• [Tensor product] For graphs \(G_1 = (V_1, E_1)\) and \(G_2 = (V_2, E_2)\), their tensor product \(G_1 \otimes G_2\) is the graph on vertices \(V_1 \times V_2\), where for every pair of edges \(u \sim u'\) in \(G_1\) and \(v \sim v'\) in \(G_2\) there is an edge \((u, v) \sim (u', v')\) in \(G_1 \otimes G_2\).

  If \(\text{Rot}_{G_1}\) and \(\text{Rot}_{G_2}\) are the rotation maps of \(G_1\) and \(G_2\), respectively, then the rotation map of \(G_1 \otimes G_2\) is

  \[ \text{Rot}_{G_1 \otimes G_2}((v, w), (i, j)) = ((v', w'), (i', j')), \]

  where \(\text{Rot}_{G_1}(v, i) = (v', i')\) and \(\text{Rot}_{G_2}(w, j) = (w', j')\).

• [Replacement product] For a \(D\)-regular graph \(G = (V, E)\) on \(n\) vertices and a \(d\)-regular graph \(H = (V', E')\) on \(D\) vertices, the replacement product \(G \circ H\) is a \(2d\)-regular graph on \(nD\) vertices \(\{(v, i) \mid v \in V, 1 \leq i \leq D\}\). The graph \(G \circ H\) has the edges \(\{(v, i) \sim (v, j) \mid v \in V, i \sim j \in E'\}\) as well as, for every edge \(v \sim w\) in \(G\) such that \(w\) is the \(i\)th neighbor of \(v\), and \(v\) is the \(j\)th neighbor of \(w\) (i.e., \(\text{Rot}_G(v, i) = (w, j))\), \(G \circ H\) has \(d\) parallel edges between \((v, i)\) and \((w, j)\).
If \( \text{Rot}_G \) and \( \text{Rot}_H \) are the rotation maps of \( G \) and \( H \), respectively, then the rotation map of the \( G \circ H \) is
\[
\text{Rot}_{G \circ H}((v, i), j) = \begin{cases} 
((v, i'), j') \text{ for } \text{Rot}_H(i, j) = (i', j) & \text{if } j \leq d \\
((w, i'), j) \text{ for } \text{Rot}_G(v, i) = (w, i') & \text{if } j > d.
\end{cases}
\]

- **[Adding self-loops]** For a \( d \)-regular graph \( G = (V, E) \), the graph \( \overline{G} \) is the \( 2d \)-regular graph obtained from \( G \) by adding \( d \) parallel self-loops around every vertex of \( G \); note that we count every self-loop around vertex \( v \) as one edge \( v \sim v \).

If \( \text{Rot}_G \) is the rotation map of \( G \), then the rotation map of \( \overline{G} \) is
\[
\text{Rot}_{\overline{G}}(v, i) = \begin{cases} 
\text{Rot}_G(v, i) & \text{if } i \leq d \\
(v, i) & \text{if } i > d.
\end{cases}
\]

### 3.2 Effect of graph operations on edge expansion

For the operation of adding self-loops, the following lemma is obvious.

**Lemma 3.1** (Self-loops). *If \( G \) is a \( d \)-regular graph with edge expansion \( \epsilon \), then the graph \( \overline{G} \) is \( 2d \)-regular with edge expansion \( \epsilon/2 \).*

For the remaining graph operations, we will give in Sections 4.1, 4.2, and 4.3, respectively, the combinatorial proofs (formalizable in VNC\(^1\)) of the following three lemmas.

**Lemma 3.2** (Powering). *Let \( G \) be a \( d \)-regular graph with edge expansion \( \epsilon \). For every integer \( k \geq 1 \), the powered graph \( (\overline{G})^k \) has edge expansion at least
\[
\frac{1}{2} \left( 1 - \left(1 - \frac{\epsilon^2}{4}\right)^{k/2} \right).
\]

**Lemma 3.3** (Tensoring). *Let \( G = (V_G, E_G) \) be a \( d_G \)-regular graph with \( d_G/2 \) self-loops at every vertex and \( H = (V_H, E_H) \) be a \( d_H \)-regular graph with \( d_H/2 \) self-loops at every vertex. If \( G \) has edge expansion \( \epsilon_G \) and \( H \) has edge expansion \( \epsilon_H \), then the tensor product graph \( G \otimes H \) has edge expansion at least \( \min\{\epsilon_G, \epsilon_H\}/50 \).*

**Lemma 3.4** (Replacement [ASS08]). *Let \( G = (V_G, E_G) \) be a \( D \)-regular graph on \( n \) vertices, and let \( H = (V_H, E_H) \) be a \( d \)-regular graph on \( D \) vertices. If \( G \) has edge expansion \( \epsilon_G \) and \( H \) has edge expansion \( \epsilon_H \), then \( G \circ H \) has edge expansion at least \( \epsilon_G^2 \epsilon_H / 48 \).*

### 3.3 Construction

With the analysis of graph operations in hand, we can now present our iterative construction of edge expanders that will be shown formalizable in VNC\(^1\). Let \( G_0 \) be a \((2d)\)-regular graph of constant size, where \( d \) is some constant. Let \( \epsilon_0 \) be the edge expansion of \( G_0 \) such that \( \epsilon_0 \geq 1/1296 \). Such a graph \( G_0 \) exists (by a counting argument) and can be found in constant time, using exhaustive search. Given \( G_0 \), we will define a bigger graph \( G_1 \) that is also \((2d)\)-regular and has edge expansion at least \( 1/1296 \). In general, given a \((2d)\)-regular graph \( G_i \) with edge expansion at least \( 1/1296 \), we define \( G_{i+1} \) as follows:
\[
G_{i+1} = ((\overline{((\overline{G_i}) \otimes (\overline{G_i})))})^c) \circ H,
\]

(2)
where \( c \) is some constant to be specified later, and \( H \) is a \( d \)-regular expander graph on \((2(4d)^2)^c\) vertices, with edge expansion at least 1/3. Again, such a graph can be found using exhaustive search.

**Theorem 3.5.** There is a constant \( c \) such that the graph \( G_{i+1} = (V_{i+1}, E_{i+1}) \) defined from \( G_i = (V_i, E_i) \) as above has the following parameters:

- \( |V_{i+1}| = |V_i|^2 \cdot D \), where \( D = (2(4d)^2)^c \),
- the degree of \( G_{i+1} \) is \( 2d \),
- the edge expansion of \( G_{i+1} \) is at least \( 1/1296 \).

**Proof.** The bounds on the size and the degree of \( G_{i+1} \) follow easily from the definitions of the graph operations used to define \( G_{i+1} \) from \( G_i \). Let \( \epsilon \geq 1/1296 \) be the edge expansion of \( G_i \). First, by Lemma 3.1, the edge expansion of \( \bigcirc G_i \) is at least \( \epsilon/2 \). By Lemma 3.3, the edge expansion of \( G'' = (\bigcirc G_i) \otimes (\bigcirc G_i) \) is at least \( \epsilon' = \epsilon/100 \). By Lemma 3.2, the \( k \)th power of the graph \( \bigcirc G'' \) has edge expansion at least

\[
\frac{1}{2} \left( 1 - \left( 1 - \frac{c^2}{40000} \right)^{k/2} \right).
\]

Choose \( k \) to be a sufficiently large constant \( c \) so that the edge expansion of the \( k \)th power of our graph, as given by the formula above, is at least 1/3. Finally, by Lemma 3.4, the edge expansion of the graph \( G_{i+1} \) is at least \((1/3)^3/48 = 1/1296\). This completes the proof.

A simple calculation shows that \( |V_i| \) is equal to \((D \cdot |V_0|)^2 \cdot D \), so the size of the \( G_i \) grows rapidly with \( i \). This rapid growth in size helps make it possible to give the explicit constructions of the expanders \( G_i \) of Section 3.4 and to formalize the construction in VNC\(^1\). However, it does make it more difficult to construct an expander of a particular size \( M \). Therefore, we give also a modified construction of expanders that allows explicit constructions of edge expanders \( \tilde{G}_i = (\tilde{V}_i, \tilde{E}_i) \) with \( |	ilde{V}_i| = 2^i \), and more generally of edge expanders on exactly \( M \) vertices for arbitrary \( M \).

Let \( c \) be a constant. Choose the constant \( d \) to be a sufficiently large power of two, \( d = 2^\ell \), so that there is a \( d \)-regular graph \( H \) on \((2(4d)^2)^c \) vertices with edge expansion at least 1/3 and so that for all \( i \leq \ell \), there are \( 2d \)-regular graphs \( \tilde{G}_i \) on \( 2^i \) vertices with edge expansion at least 1/1296. These graphs \( H \) and \( \tilde{G}_i \) can be found by exhaustive search. We construct expander graphs \( \tilde{G}_i \) with edge expansion \( \geq 1/1296 \). For \( i > 2\ell + 7 \), let \( i' = [(i-2\ell-5)/2] \) and \( i'' = [(i-2\ell-5)/2] \), so \( i = i' + i'' + 2\ell + 5 \). Define

\[
\tilde{G}_i = (((\bigcirc((\bigcirc \tilde{G}_{i'} \otimes (\bigcirc \tilde{G}_{i''}))))^c) \circ H.
\]  

**Theorem 3.6.** There is a constant \( c \) such that the graph \( \tilde{G}_i = (\tilde{V}_i, \tilde{E}_i) \) defined as above has the following parameters:

- \( |\tilde{V}_i| = 2^i \),
- the degree of \( \tilde{G}_i \) is \( 2d \),
- the edge expansion of \( \tilde{G}_i \) is at least \( 1/1296 \).

**Proof.** The proof is by induction on \( i \). Assume the induction hypothesis holds for \( \tilde{G}_{i'} \) and \( \tilde{G}_{i''} \). The graph operations used to define \( \tilde{G}_i \) imply that

\[
|\tilde{V}_i| = (2(4d)^2)^c \cdot |V_{i'}| \cdot |V_{i''}| = 2^{5d^2c} \cdot |V_{i'}| \cdot |V_{i''}| = 2^{5d^2c} \cdot 2^i \cdot 2^i = 2^i.
\]
They also imply that $\tilde{G}_i$ is $2d$-regular. The induction hypotheses and Lemma 3.1 imply that $\bigcirc \tilde{G}_{i'}$ and $\bigcirc \tilde{G}_{i'}$ each have edge expansion at least $\epsilon/2$ for $\epsilon = 1/1296$. Now, the same calculation as in the proof of Theorem 3.5 establishes that $\tilde{G}_i$ has edge expansion at least $1/1296$.

Now that we have constructed edge expanders of sizes $2^i$, it is easy to obtain an edge expander $\tilde{G}$ of an arbitrary size $M$. For this, choose $i$ so that $2^{i-1} < M \leq 2^i$. Partition the vertices of $\tilde{G}_i$ into $M$ disjoint subsets each of size 1 or 2. Define the graph $\tilde{G}$ by collapsing each of these subsets of vertices of $\tilde{G}_i$ into a single vertex of $\tilde{G}$, and inheriting the edges from the all of the nodes in the subset.

Clearly, $\tilde{G}$ has exactly $M$ vertices. Since $\tilde{G}_i$ has degree $2d$, the degree of $\tilde{G}$ is at most $4d$; more precisely, each vertex has degree $2d$ or $4d$. By adding extra self-loops, we get a new graph that is $4d$-regular. Let us call this new $4d$-regular graph by the same name $\tilde{G}$. The next claim shows that $\tilde{G}$ has expansion at least $\epsilon/2$ where $\epsilon = 1/1296$.

**Proposition 3.7.** For every set $S$ of nodes in $\tilde{G}$, the number of edges in $\tilde{G}$ between $S$ and $\overline{S}$ is

$$|E(S, \overline{S})| \geq \epsilon \cdot (2d) \cdot \min\{|S|, |\overline{S}|\}.$$  

**Proof.** Let $S'$ be the set of vertices in $\tilde{G}_i$ corresponding to the set $S$ in $\tilde{G}$. The number of edges leaving $S$ in $\tilde{G}$ is exactly the number of edges leaving $S'$ in $\tilde{G}_i$. As $\tilde{G}_i$ has edge expansion at least $\epsilon$, and

$$\min\{|S'|, |\overline{S}'|\} \geq \min\{|S|, |\overline{S}|\},$$

the claim follows.

### 3.4 Explicitness of the construction

The next theorem justifies the claim that our iterative construction of $G_k$ is fully explicit, by describing algorithms for the rotation maps of $G_k$. The input to the algorithm is a pair $(v, i)$ specifying a vertex and an edge in $G_k$; the output is the value of the rotation map. The value $k$ is allowed to vary, and can readily be computed from $v$.

**Theorem 3.8.** Fix constants $c$ and $d$ as above. There is a deterministic polynomial-time algorithm that, given the name of a vertex $v$ (in binary) of $G_k$ and an index $i \in [2d]$, outputs the value $\text{Rot}_{G_k}(v, i)$. Moreover, there is an alternating linear time algorithm which accepts the graph of $G_k$; namely, it accepts exactly the triples of the form $(v, i, \text{Rot}_{G_k}(v, i))$.

It may be unexpected that we discuss alternating linear time, but the point is that this is what we need for the formalization of our arguments in the bounded arithmetic theory VNC$^1$ in Section 6. For that, the important thing is the computational complexity of $\text{Rot}_{G_k}$ as a function of the size $|V_k|$ of the graph, whereas Theorem 3.8 expresses runtimes in terms of the size of the name of the vertex. But, the alternating linear time algorithm of Theorem 3.8 will be viewed as an alternating logarithmic time algorithm for purposes of formalization in VNC$^1$. (In the same setting, the polynomial time algorithm would be a polylogarithmic time algorithm, and it is open whether such algorithms can in general be formalized in VNC$^1$.)

**Proof of Theorem 3.8.** The construction of the polynomial algorithm is very simple. We start with a fixed rotation map $\text{Rot}_{G_0}$ for $G_0 = (V_0, E_0)$ on $n_0 = |V_0|$ vertices, and a fixed $d$-regular graph $H$ with rotation map $\text{Rot}_H$. Computing $\text{Rot}_{G_k}$ requires either an evaluation of $\text{Rot}_{G_0}$ or $\text{Rot}_H$ or at most $2c$ many recursive calls to $\text{Rot}_{G_{k-1}}$. If $v$ is an $\ell$-bit description of a vertex in $G_k$, we have that $G_k$ has approximately $2^\ell$ vertices, and that $k \leq \log \ell$. It follows that the recursive calls are nested to depth $\leq \log \ell$, so the overall runtime of the algorithm is polynomial.
To describe the alternating algorithm, we need to describe the representation of vertices of $G_k$ more carefully. For the purpose of giving a recursive algorithm, it is slightly easier to give an algorithm for the rotation map $\text{Rot}_{G_k'}$ of the graph $G'_k = \bigcirc G_k$. This suffices to prove the theorem, since $\text{Rot}_{G_k}$ and $\text{Rot}_{G_k'}$ have the same vertices and $\text{Rot}_{G_k}$ is just a restriction of $\text{Rot}_{G_k'}$.

We have $G'_k$ equal to the $(4d)$-regular graph $\bigcirc((\bigcirc(G'_{k-1} \otimes G'_{k-1}))^c \circ H)$. A vertex $v$ of $H$ is named by a binary string of length $\ell_v = \lceil \log |H| \rceil$. A vertex $v$ of $G'_k$ will be named by a string of fixed length $\ell_k$. For $k = 0$, a vertex is named using $\ell_0 = \lceil \log |V_0| \rceil$ bits. For $k \geq 1$, a vertex $v$ of $G'_k$ is named by a concatenation $v_1 v_2 i$ of (names of) vertices $v_1$ and $v_2$ of $G'_{k-1}$ and a vertex $i$ of $H$. Thus, $\ell_k = 2\ell_{k-1} + \ell_h$, so $\ell_k = 2^k \ell_0 + (2^k-1)\ell_h$.

The alternating linear time algorithm for the rotation map $\text{Rot}_{G_k'}$ takes as input two names $v_1 v_2 i$ and $v'_1 v'_2 i'$ and values $j, j' < d$, and should accept if and only if

$$\text{Rot}_{G_k'}(v_1 v_2 i, j) = (v'_1 v'_2 i', j').$$

The algorithm works as follows.

First, if either $j \geq d$ or $j' \geq d$, then it accepts iff $v'_1 v'_2 i' = v_1 v_2 i$ and $j = j'$ (since $j \geq d$ indicates a self-loop of $G'_k$). Otherwise, by the definition of replacement product, the vertices $i$ and $i'$ in $H$ indicate edges to traverse in $((\bigcirc(G'_{k-1} \otimes G'_{k-1}))^c \circ H)$. We have that $i < (2(4d)^2)^c$ encodes a sequence of values $\langle i_1, \ldots, i_c \rangle$, with each $i_s < 2(4d)^2$ indicating an edge of $\bigcirc(G'_{k-1} \otimes G'_{k-1})$. Without loss of generality, these edges are encoded with $i_s = \alpha_s(4d)^2 + \beta_s(4d) + \gamma_s$ with $\alpha_s, \beta_s, \gamma_s < 4d$: when $\alpha_s = 1$, the edge is a self-loop, and otherwise $\alpha_s = 0$ and $\beta_s$ and $\gamma_s$ encode edges in the first and second components $G'_{k-1}$ of the tensor product. Likewise, $i'$ encodes a sequence $\langle i'_1, \ldots, i'_c \rangle$, with each $i'_s = \alpha'_s(4d)^2 + \beta'_s(4d) + \gamma'_s$.

The algorithm nondeterministically guesses the values $i_1, \ldots, i_c$, the values $\alpha_k, \beta_k, \gamma_k$, the values $\alpha'_1, \ldots, \alpha'_c$, and finally values $w_0^1 = v_1, w_1^1 = v_2, \ldots, w_{c-1}^1 = v'_1$ and $w_0^2 = v_2, w_1^2 = v'_2, \ldots, w_{c-1}^2 = v'_1$, where each $w_i^1, w_i^2$ is a $\ell_{k-1}$ bit string, intended to name a vertex of $G'_k$. The intent is that the pairs $w_1^i, w_2^i$ are the members of the tensor product $\bigcirc(G_{k-1}' \otimes G_{k-1}')$ which are obtained by traversing the $c$ edges encoded by $i$, and in reverse order by $i'$. The algorithm then branches universally to check one of the following conditions:

a. The guessed sequences $i_1, \ldots, i_c$ and $i'_1, \ldots, i'_c$ are the values actually encoded by $i$ and $i'$.

b. Universally choose a value $s$ and verify (using constantly many alternations) that $i_s = \alpha_s(4d)^2 + \beta_s(4d) + \gamma_s$ and that $i'_s = \alpha'_s(4d)^2 + \beta'_s(4d) + \gamma'_s$.

c. Universally choose a single value $s \in \{1, 2, \ldots, c\}$. If $\alpha_s = 1$, then verify that $\alpha'_s = 1$ and $\beta'_s = \beta_s$ and $\gamma'_s = \gamma_s$ and that $w_1^s = w_{s+1}^1$ and $w_2^s = w_{s+1}^2$. (This corresponds to traversing a self-loop in the tensor product.) Otherwise, universally choose to do one of:

i. Verify that $\text{Rot}_{G_{k-1}'}(w_s, \beta_s) = (w_{s+1}, \beta'_s)$,

ii. Verify that $\text{Rot}_{G_{k-1}'}(w'_s, \gamma'_s) = (w'_{s+1}, \gamma'_s)$

The algorithm accepts iff the check a.-c. accepts.

To analyze the runtime of the algorithm, note that it runs for linear time making constantly many alternations, and then makes one recursive call to $G'_{k-1}$. The recursive call uses a parameter less than half the length of the first input. Thus, the overall runtime is linear. Moreover, since $k = O(\log n)$, the algorithm makes only $O(\log n)$ many alternations.

\[\text{If } i \text{ and } i' \text{ encode the sequences by using a base } (2(4d)^2)^c \text{ representation, this is verifiable in linear time with a constant number of alternations [Lip78]. With this encoding, the correctness is } \Delta_0 \text{-definable as is needed later for formalization in VNC}^1.\]
Remark 3.9. Theorem 3.8 also holds for the graphs $\tilde{G}_k$. The algorithms are almost identical to those given above for $\tilde{G}_k$, and we leave the details to the reader. The principal difference is that the alternating linear time algorithm for recognizing the graph of $\tilde{G}_k$ may make a (single) recursive call to either $\tilde{G}_{k'}$ or $\tilde{G}_{k''}$, where $k' = \lceil (k - 2c\ell - 5)/2 \rceil$ and $k'' = \lceil (k - 2c\ell - 5)/2 \rceil$.

The recursive algorithm is actually a little simpler than it appears at first glance. The graph $\tilde{G}_k$ is defined recursively in terms of the two graphs $\tilde{G}_{k'}$ and $\tilde{G}_{k''}$, with $k''$ equal to either $k'$ or $k' + 1$. In the next round of the iteration, both of $\tilde{G}_{k'}$ and $\tilde{G}_{k''}$ can be defined recursively from $\tilde{G}_{j'}$ and $\tilde{G}_{j''}$ where $j' = \lceil (k' - 2c\ell - 5)/2 \rceil$ and $j'' = \lceil (k'' - 2c\ell - 5)/2 \rceil$, i.e., $j''$ is equal to either $j'$ or $j' + 1$. More generally, iterating the levels of recursion only requires computing pairs of graphs $\tilde{G}_{j'}$ and $\tilde{G}_{j''}$ with $j''$ equal to either $j'$ or $j' + 1$. Although it is not strictly needed, this property of the recursive definition $\tilde{G}_k$ will be used to simplify the proof of Theorem 6.9 that $\text{VNC}^1$ can prove the existence of expander graphs of arbitrary size.

4 Effect of graph operations on edge expansion: Proofs

In this section, we analyze how graph operations used in the recursive construction of Theorem 3.5 affect edge expansion. We analyze powering (in Section 4.1), tensor product (in Section 4.2), and replacement product (in Section 4.3).

4.1 Graph powering

The main result of this subsection is the following.

Lemma 4.1. Let $G$ be a $d$-regular graph with edge expansion $\epsilon$. For every integer $k \geq 1$, the powered graph $(\bigodot G)^k$ has edge expansion at least

$$\frac{1}{2} \cdot \left( 1 - \left( 1 - \frac{\epsilon^2}{4} \right)^{k/2} \right).$$

Our analysis will be done in two stages. First, we use the result of Mihail [Mih89] showing that a random $k$-step walk on an edge expander $\bigodot G$ quickly converges to the uniform distribution over the vertices of $\bigodot G$. Then we show that such convergence to the uniform distribution implies good edge expansion of $(\bigodot G)^k$, using some ideas from [AC88].

4.1.1 Edge expansion implies fast mixing

To see why edge expansion is related to the mixing time, consider the following experiment. Let $G = (V, E)$ be a graph on $n$ vertices, and let $U \subset V$ be a subset of at most $n/2$ vertices. Pick a vertex $u$ uniformly at random from $U$ and then pick an edge $(u, v)$ incident to $u$ uniformly at random. Clearly, the probability that $v$ is outside of $U$ is exactly the edge expansion of $U$ in $G$. Similarly, if we pick a random vertex $u$ from $U$ and then instead of performing a single step of a random walk we perform $k$ steps, then the probability that we end up at a vertex outside of $U$ is exactly the edge expansion of $U$ in $G^k$. It is well known that the probability distribution induced by taking a random $k$-step walk on a $d$-regular graph $G$ tends to the uniform distribution on the vertices of $G$; the mixing time bounds the distance from the uniform distribution in terms of the number of steps $k$. Assuming $k$ is large enough, we get that the edge expansion of $G^k$ is close to $|\bar{U}|/|V|$, which is at least $1/2$ since $|\bar{U}| \leq |V|/2$. The closeness of the edge expansion of $G^k$ to $1/2$ is bounded by a function of the mixing time of $G$. 

12
Mihail [Mih89] gave a combinatorial proof of the following result showing the exponentially fast convergence of a random walk on a regular graph to the uniform distribution.

**Lemma 4.2 ([Mih89]).** Let $G$ be a $d$-regular graph with edge expansion $\epsilon$. Let $A$ be the normalized adjacency matrix of $G' = \bigcirc G$. Let $\pi$ be any initial distribution on vertices of $G'$, and let $u$ be the uniform distribution on vertices of $G'$. Then

$$\|A^k\pi - u\|^2 \leq (1 - (\epsilon^2/4))^k \cdot \|\pi - u\|^2.$$  

**Proof of Lemma 4.2.** Let $e = \pi - u$ be the discrepancy vector between the current probability distribution $\pi$ and the uniform distribution $u$. After one step on $G$, the discrepancy vector becomes $e' = A\pi - u = A\pi - Au = Ae$. The proof proceeds in two stages:

1. Show that $\|e\|^2 - \|e'\|^2 \geq (2d)^{-1} \cdot \sum_{\{i,j\} \in E} (e_i - e_j)^2$.

2. Use the edge expansion of $G$ to show that there are many edges $\{i, j\}$ in $G$ where $e_i$ and $e_j$ are significantly different, implying that $\|e'\|^2 \ll \|e\|^2$.

The analysis of a $k$-step walk follows by induction on $k$.

For stage 1, we prove the following.

**Claim 4.3.** Let $G = (V, E)$ be a $d$-regular graph on $V = \{1, \ldots, n\}$. Let $A$ be the normalized adjacency matrix for the graph $G' = \bigcirc G$. For any $f \in \mathbb{R}^n$, we have

$$\|f\|^2 - \|Af\|^2 \geq \frac{1}{2d} \cdot \sum_{\{i,j\} \in E} (f_i - f_j)^2.$$  

**Proof of Claim 4.3.** For each $i \in V$, we have

$$(Af)_i = \frac{f_i}{2} + \sum_{j : \{i,j\} \in E} \frac{f_j}{2d} = \sum_{j : \{i,j\} \in E} \frac{f_i + f_j}{2d}.$$ 

Using this, we get

$$\|Af\|^2 = \sum_{i \in V} ((Af)_i)^2 = \sum_{i \in V} \left( \frac{1}{d} \cdot \sum_{j : \{i,j\} \in E} \frac{f_i + f_j}{2} \right)^2$$

$$\leq \sum_{i \in V} \frac{1}{d} \cdot \sum_{j : \{i,j\} \in E} \left( \frac{f_i + f_j}{2} \right)^2 \quad [\text{Cauchy-Schwarz applied } |V| \text{ times}]$$

$$= \frac{1}{4d} \cdot \left( \sum_{i \in V} \sum_{j : \{i,j\} \in E} (f_i + f_j)^2 \right)$$

$$= \frac{1}{4d} \cdot \left( 2 \cdot \sum_{\{i,j\} \in E} (f_i + f_j)^2 + \sum_{\{i,i\} \in E} (f_i + f_i)^2 \right)$$

$$= \frac{1}{d} \cdot \left( \frac{1}{2} \cdot \sum_{\{i,j\} \in E} (f_i + f_j)^2 + \sum_{\{i,i\} \in E} f_i^2 \right).$$  

13
Since
\[ \|f\|^2 = \sum_{i \in V} f_i^2 \]
\[ = \frac{1}{d} \cdot \left( \sum_{\{i < j\} \in E} (f_i^2 + f_j^2) + \sum_{\{i, i\} \in E} f_i^2 \right), \]
we get
\[ \|f\|^2 - \|Af\|^2 \geq \frac{1}{d} \cdot \sum_{\{i < j\} \in E} \left( f_i^2 + f_j^2 - \frac{(f_i + f_j)^2}{2} \right) \]
\[ = \frac{1}{2d} \cdot \sum_{\{i < j\} \in E} (f_i - f_j)^2, \]
as required.

Next, for stage 2, we prove the following two claims.

**Claim 4.4.** Let \( G = (V, E) \) be a \( d \)-regular graph on \( n \) vertices with edge expansion \( \epsilon \). For any \( h : V \to \mathbb{R} \), let the vertices in \( V = \{1, \ldots, n\} \) be ordered so that \( h(1) \geq h(2) \geq \cdots \geq h(n) \). Then
\[ \sum_{\{i, j\} \in E} |h(i) - h(j)| \geq (ed) \cdot \sum_{i \in V} |h(i) - h(\lfloor n/2 \rfloor)|. \]

**Proof of Claim 4.4.** For simplicity, assume \( n \) is even. Define \( h_0(i) = h(i) - h(n/2) \). We have
\[ \sum_{\{i, j\} \in E} |h(i) - h(j)| = \sum_{\{i, j\} \in E} |h_0(i) - h_0(j)| \quad \text{[definition of } h_0 \text{]} \]
\[ = \sum_{\{i < j\} \in E} (h_0(i) - h_0(j)) \quad \text{[ordering of } h_0(i) \text{'s]} \]
\[ = \sum_{\{i < j\} \in E} \sum_{k=i}^{j-1} (h_0(k) - h_0(k+1)) \quad \text{[telescoping sum]} \]
\[ = \sum_{k=1}^{n-1} \sum_{\{i \leq k < j\} \in E} (h_0(k) - h_0(k+1)) \quad \text{[change order of summation]} \]
\[ = \sum_{k=1}^{n-1} (h_0(k) - h_0(k+1)) \cdot \sum_{\{i \leq k < j\} \in E} 1 \quad \text{[re-arranging]} \]
\[ = \sum_{k=1}^{n-1} (h_0(k) - h_0(k+1)) \cdot |E([k], [k])|. \quad \text{[definition of } E([k], [k]) \text{]} \]
We split the last summation over $k$ into two sums, over $1 \leq k \leq (n/2) - 1$ and over $n/2 \leq k \leq n - 1$, and use the edge expansion of $G$ to lower-bound $|E([k], [K])|$ by $\epsilon dk$ and $\epsilon d(n - k)$, respectively. We get

$$
\sum_{(i,j) \in E} |h(i) - h(j)| \geq \sum_{k=1}^{n/2-1} (h_0(k) - h_0(k + 1)) \cdot (\epsilon dk) + \sum_{k=n/2}^{n-1} (h_0(k) - h_0(k + 1)) \cdot (\epsilon d(n - k))
$$

$$
= (\epsilon d) \cdot \left( \sum_{k=1}^{n/2-1} k \cdot (h_0(k) - h_0(k + 1)) + \sum_{k=n/2}^{n-1} (n - k) \cdot (h_0(k) - h_0(k + 1)) \right)
$$

$$
= (\epsilon d) \cdot \left( \sum_{k=1}^{n/2-1} h_0(k) + \sum_{k=n/2}^{n-1} (-h_0(k)) \right) \quad \text{[since } h_0(n/2) = 0]\n$$

$$
= (\epsilon d) \cdot \left( \sum_{k=1}^{n/2} |h_0(k)| + \sum_{k=n/2+1}^{n-1} |h_0(k)| \right),
$$

where the last equality is due to the fact that $h_0(k) \geq 0$ for $k \leq n/2$, and $h_0(k) \leq 0$ for $k > n/2$. \hfill \Box

As a corollary, we get the following.

**Claim 4.5.** Let $G = (V, E)$ be a $d$-regular graph on $n$ vertices with edge expansion $\epsilon$. For any $g : V \to \mathbb{R}$, let the vertices in $V = \{1, \ldots, n\}$ be ordered so that $g(1) \geq g(2) \geq \cdots \geq g(n)$. If $g([n/2]) = 0$, then

$$
\sum_{i,j \in E} (g(i) - g(j))^2 \geq \frac{\epsilon^2 d}{2} \cdot \sum_{i \in V} g(i)^2.
$$

**Proof of Claim 4.5.** We define new functions $g^+, g^- : V \to \mathbb{R}$ by $g^+(v) = \max\{g(v), 0\}$ and $g^-(v) = \min\{g(v), 0\}$, for all $v \in V$. Note that $g^+(i) = g(i)$ for $i \leq [n/2]$ and is 0 elsewhere, and $g^-(i) = g(i)$ for $i > [n/2]$ and is 0 elsewhere. It follows that

$$
\sum_{i \in V} g^+(i)^2 + \sum_{i \in V} g^-(i)^2 = \sum_{i \in V} g(i)^2. \quad (5)
$$

Clearly, we have

$$
\sum_{i,j \in E} (g(i) - g(j))^2 \geq \sum_{i,j \in E} (g^+(i) - g^+(j))^2 + \sum_{i,j \in E} (g^-(i) - g^-(j))^2. \quad (6)
$$

We will show that the statement of the claim holds for both $g^+$ and $g^-$, which, by (5) and (6), will imply the claim also for the case of $g$. We consider the case of $g^+$ first; the case of $g^-$ is similar.

Observe that, for every $f : V \to \mathbb{R}$, we have that

$$
\sum_{i \in V} f(i)^2 = \frac{1}{d} \cdot \left( \sum_{\{i<j\} \in E} (f(i)^2 + f(j)^2) + \sum_{\{i,i\} \in E} f(i)^2 \right)
$$

$$
\geq \sum_{\{i<j\} \in E} \frac{f(i)^2 + f(j)^2}{d}
$$

$$
\geq \frac{1}{2d} \cdot \sum_{\{i<j\} \in E} (f(i) + f(j))^2,
$$

15
where the last inequality is by the simple fact that \(a^2 + b^2 \geq (a + b)^2/2\), for all \(a, b \in \mathbb{R}\). Applying this to \(g^+\), we get

\[
\sum_{\{i, j\} \in E} (g^+(i) - g^+(j))^2 \\
\geq \left( \sum_{\{i < j\} \in E} (g^+(i) - g^+(j))^2 \right) \cdot \left( \sum_{\{i < j\} \in E} (g^+(i) + g^+(j))^2 \right) \cdot \left( (2d) \cdot \sum_{i \in V} g^+(i)^2 \right)^{-1}
\]

\[
\geq \left( \sum_{\{i < j\} \in E} (g^+(i) - g^+(j))^2 \right) \cdot \left( (2d) \cdot \sum_{i \in V} g^+(i)^2 \right)^{-1},
\]

where the last inequality is by Cauchy-Schwarz. Since \(g^+(i)^2 \geq g^+(j)^2\) for \(i < j\), we can replace \((g^+(i)^2 - g^+(j)^2)\) inside the first factor above with \(|g^+(i)^2 - g^+(j)^2|\), and then apply Claim 4.4 with \(h(i) = g^+(i)^2\). We get that

\[
\sum_{\{i, j\} \in E} (g^+(i) - g^+(j))^2 \geq \left( \sum_{\{i < j\} \in E} |g^+(i)^2 - g^+(j)^2| \right) \cdot \left( (2d) \cdot \sum_{i \in V} g^+(i)^2 \right)^{-1}
\]

\[
\geq (\epsilon d)^2 \cdot \left( \sum_{i \in V} g^+(i)^2 \right)^{-1} \\cdot \left( (2d) \cdot \sum_{i \in V} g^+(i)^2 \right)^{-1}
\]

\[
\geq \frac{\epsilon^2 d}{2} \cdot \sum_{i \in V} g^+(i)^2,
\]

which means that the required claim holds for \(g^+\).

Finally, for the case of \(g^-\), we use the same reasoning as above applied to \(\sum_{\{i < j\} \in E} (g^-(j) - g^-(i))^2\), and using the fact that \(g^-(j)^2 \geq g^-(i)^2\) for \(i < j\). Reversing the order of vertices of \(V\) (so that node \(n\) becomes the first node, node \(n-1\) the second node, and so on), and applying Claim 4.4 with \(h(i) = g^-(i)^2\) (for the new vertex order), we get the claim for \(g^-\) as well.

We are now ready to finish the proof of the lemma. Let \(e = \pi - u\). Since \(Au = u\), we get that \(A\pi - u = A(\pi - u) = Ae\). We have by Claim 4.3 that

\[
\|e\|^2 - \|Ae\|^2 \geq \frac{1}{2d} \cdot \sum_{\{i, j\} \in E} (e_i - e_j)^2.
\]  

(7)

Order the nodes in \(V\) so that \(e_1 \geq e_2 \geq \cdots \geq e_n\). For each \(i \in V\), define \(g(i) = e_i - e_m\), where \(m = \lfloor n/2 \rfloor\). By Claim 4.5 applied to \(G\) with the function \(g\), we get

\[
\sum_{\{i, j\} \in E} (g(i) - g(j))^2 \geq \frac{\epsilon^2 d}{2} \cdot \sum_{i \in V} g(i)^2.
\]  

(8)

Since \((e_i - e_j)^2 = (g(i) - g(j))^2\), we conclude by Eqs. (7) and (8) that

\[
\|e\|^2 - \|Ae\|^2 \geq \frac{1}{2d} \cdot \frac{\epsilon^2 d}{2} \cdot \sum_{i \in V} g(i)^2 = \frac{\epsilon^2}{4} \cdot \sum_{i \in V} g(i)^2.
\]  

(9)
Finally, we have
\[ \sum_{i \in V} g(i)^2 = \sum_{i \in V} (e_i - e_m)^2 = \left( \sum_{i=1}^n e_i^2 \right) + n \cdot e_m^2 - (2e_m) \cdot \sum_{i=1}^n e_i \geq \|e\|^2, \]
where the last inequality is because
\[ \sum_{i \in V} e_i = \sum_{i \in V} \pi_i - \sum_{i \in V} u_i = 0 \text{ and } n e_m^2 \geq 0. \]
By Eq. 9, this implies that
\[ \|Ae\|^2 \leq (1 - e^2/4) \cdot \|e\|^2. \]
Applying Eq. (10) inductively, we get that
\[ \|A^k e\|^2 \leq (1 - e^2/4)^k \cdot \|e\|^2, \]
as required.

4.1.2 Mixing implies edge expansion

Let \( G' = \bigcirc G \), and let \( G'' = (G')^k \). Next we relate the edge expansion of \( G'' \) to the mixing time of a \( k \)-step random walk on \( G' \). Let \( u \) denote the uniform distribution on the vertices of \( G'' \). For a subset \( U \) of vertices of \( G'' \), we denote by \( u_U \) the probability distribution that is uniform over \( U \), i.e., every vertex in \( U \) gets weight \( 1/n \), and every vertex outside of \( U \) gets weight 0. We denote by \( \chi_U \) the characteristic vector of the set \( U \) (whose \( i \)-th entry is 1 if \( i \in U \), and is 0 otherwise).

**Lemma 4.6.** Suppose \( G'' = (V, E) \) is a regular graph on \( n \) vertices, with normalized adjacency matrix \( A \) such that for some \( \delta > 0 \) the following holds: for every subset \( U \subset V \) of size at most \( |V|/2 \),
\[ \|Au_U - u\|^2 \leq \delta \cdot \|u_U - u\|^2. \]
Then \( G'' \) has edge expansion at least \( (1 - \sqrt{\delta})/2 \).

**Proof.** The edge expansion in \( G'' \) of a given subset \( U \subset V \) of size at most \( n/2 \) is exactly
\[ \Pr_{w \in U, (w, w') \in E} [w' \in \overline{U}] = \langle \chi_{\overline{U}}, Au_U \rangle. \]
Using the decomposition \( u_U = u + (u_U - u) \), and observing that \( Au = u \) for a regular graph, we get
\[ \langle \chi_{\overline{U}}, Au_U \rangle = \langle \chi_{\overline{U}}, Au \rangle + \langle \chi_{\overline{U}}, A(u_U - u) \rangle = \|U\|/n + \langle \chi_{\overline{U}}, A(u_U - u) \rangle. \]
Next we upper-bound \( |\langle \chi_{\overline{U}}, A(u_U - u) \rangle| \), using the Cauchy-Schwarz inequality:

**Claim 4.7.** We have
\[ |\langle \chi_{\overline{U}}, A(u_U - u) \rangle| \leq \frac{\sqrt{\delta} \cdot |U|}{n}. \]

**Proof of Claim 4.7.** To simplify the calculations, we use the following simple fact. For any probability distributions \( \pi_1 \) and \( \pi_2 \) over \( [n] \), any vector \( w \in \mathbb{R}^n \), and any \( C \in \mathbb{R}, \langle w, \pi_1 - \pi_2 \rangle = \langle w - Cu, \pi_1 - \pi_2 \rangle \).
(For the proof, observe that \( C \cdot \langle u, \pi_1 - \pi_2 \rangle = 0 \), since the vectors \( \pi_1 \) and \( \pi_2 \) add up to 1.)
By this fact and Cauchy-Schwarz, we get for any \( C \in \mathbb{R} \) that
\[
|\langle \chi_U, A(u_U - u) \rangle| = |\langle \chi_U - Cu, A(u_U - u) \rangle| \\
\leq \|\chi_U - Cu\| \cdot \|A(u_U - u)\|.
\]
Setting \( C = |U| \), we compute the square of the first factor as follows:
\[
\|\chi_U - Cu\|^2 = \frac{|U| \cdot (n - C)^2}{n^2} + \frac{|U| \cdot C^2}{n^2} = \frac{|U| \cdot |U|}{n}.
\]
By the assumption of the lemma, we upper-bound the second factor as follows:
\[
\|A(u_U - u)\| \leq \sqrt{\delta} \cdot \|u_U - u\| = \sqrt{\delta} \cdot \sqrt{\frac{|U|}{n \cdot |U|}}.
\]
Overall, we obtain that
\[
|\langle \chi_U, A(u_U - u) \rangle| \leq \sqrt{\frac{|U| \cdot |U| \cdot \delta \cdot |U|}{n \cdot |U|}} = \frac{\sqrt{\delta} \cdot |U|}{n},
\]
which proves Claim 4.7.

It follows from Claim 4.7 that \( \langle \chi_U, Au_U \rangle \geq (|U|/n) \cdot (1 - \sqrt{\delta}) \geq \frac{1}{2} \cdot (1 - \sqrt{\delta}) \), since \( |U| \geq n/2 \).

We now give the proof of Lemma 4.1.

**Proof of Lemma 4.1.** By Lemma 4.2, we get for the normalized adjacency matrix \( A \) of the graph \( \bigcirc G \) and for every subset \( U \subset V \) that
\[
\|A^k u_U - u\|^2 \leq (1 - (\epsilon^2/4))^k \cdot \|u_U - u\|^2.
\]
Applying Lemma 4.6 concludes the proof.

**Constructivity.** The proof of Lemma 4.1 provides an efficient (uniform \( \text{NC}^1 \)) algorithm for the contrapositive: Given a non-expanding set \( U \) in the graph \( \bigcirc G^k \), for some constant \( k > 0 \), the algorithm outputs a set \( U' \) that is non-expanding in the graph \( G \). First, for \( \pi = u_U \) (the uniform distribution over the set \( U \)), the lack of edge expansion for \( U \) in \( \bigcirc G^k \) implies by Lemma 4.6 that
\[
\|A^k u_U - u\|^2 > \delta^k \cdot \|\pi - u\|^2,
\]
for \( \delta = 1 - (\epsilon^2/4) \), where \( A \) is the normalized adjacency matrix of the graph \( \bigcirc G \). There must exist an \( i \), \( 1 \leq i \leq k \), such that
\[
\|A^i u_U - u\|^2 > \delta \cdot \|A^{i-1} u_U - u\|^2.
\]
Let \( \pi' = A^{i-1} \pi \). Order the nodes in \( V \) so that \( \pi'(1) \geq \pi'(2) \geq \cdots \geq \pi'(n) \). For some \( 1 \leq \ell \leq n \), we get that a set \([\ell]\), or its complement, is less than \( \epsilon \) edge-expanding in \( G \).

Finding a good \( 1 \leq i < k \) and the distribution \( \pi' \) involves powering the matrix \( A \) up to the constant power \( k \); this is computable in \( \text{NC}^1 \). Sorting the nodes according to \( \pi' \) can also be done in \( \text{NC}^1 \). Finally, there are at most \( n - 1 \) candidate sets \([1],[2],\ldots,[n-1]\) (according to the ordering given by \( \pi' \)) to test for poor edge expansion in \( G \); these tests can be done in parallel, using an \( \text{NC}^1 \) circuit for each test.
Remark 4.8. The bound on the edge expansion of $G^k$ in terms of the edge expansion of $G$ that we get in Lemma 4.1 is almost the same as the one would get using the following well-known connection between edge expansion and the eigenvalue gap. For a regular graph $G$ with a normalized adjacency matrix $A$ and the second largest eigenvalue $\lambda_2$ of $A$, the edge expansion $\epsilon$ of $G$ satisfies the Cheeger inequalities [Alo86, AM85, Dod84]:

$$(1 - \lambda_2)/2 \leq \epsilon \leq \sqrt{2(1 - \lambda_2)}.$$  

Using the left inequality, we can lower-bound the edge expansion of $G^k$ by $(1 - \lambda_2^k)/2$. Using the right inequality, we get $\lambda_2 \leq (1 - \epsilon^2/2)$. So, the edge expansion of $G^k$ is at least $(1/2) \cdot (1 - (1 - \epsilon^2/2)^k)$.

4.2 Tensor product

Here we will show that the tensor product of two regular graphs $G = (V_G, E_G)$ and $H = (V_H, E_H)$ with edge expansions $\epsilon_G$ and $\epsilon_H$, respectively, results in a graph with the edge expansion $\Omega(\min\{\epsilon_G, \epsilon_H\})$. More precisely, we prove the following result (not trying to optimize the constants).

Lemma 4.9. Let $G = (V_G, E_G)$ be a $d_G$-regular graph with $d_G/2$ self-loops at every vertex and $H = (V_H, E_H)$ be a $d_H$-regular graph with $d_H/2$ self-loops at every vertex. If $G$ has edge expansion $\epsilon_G$ and $H$ has edge expansion $\epsilon_H$, then the tensor product graph $G \otimes H$ has edge expansion at least $\min\{\epsilon_G, \epsilon_H\}/50$.

Before proving the lemma, we give some intuition. Suppose $G$ is a $d_G$-regular graph on $n_G$ vertices, and $H$ is $d_H$-regular graph on $n_H$ vertices. As a “warm-up”, consider the special case of a subset of vertices $S$ of the tensor product $G \otimes H$ such that $S = A \times B$. Moreover, assume that $|B| < n_H/2$. Then at least $\epsilon_H d_H |B|$ edges are leaving the set $B$ in graph $H$. If $G$ has edge expansion $\epsilon_G$, then the number of edges leaving $A \times B$ in $G \otimes H$, yielding a total of at least $\epsilon_H d_H |B| d_G |A|$ edges leaving $A \times B$. After normalization (division by $d_G d_H |A||B|$), this yields edge expansion $\epsilon_H$ from the set $S$. In the case, $B$ is larger than $n_H/2$, but $A$ is smaller than $n_G/2$, we can use the edge expansion of $A$, to obtain the edge expansion at least $\epsilon_G$ from $S$.

For general sets $S$ of vertices in $G \otimes H$, we consider the characteristic matrix of $S$, which is an $n_G \times n_H$ 0-1 matrix with $(i, j)$th entry being 1 iff $(i, j) \in S$. We then argue that it is possible to remove some rows or some columns of this matrix so that the resulting matrix has a constant fraction of 1’s in the original matrix (i.e., we removed only a constant fraction of vertices from $S$), and either every row or every column has at most some constant fraction of 1’s.

Suppose we have the former case (the other case is treated similarly). That is, we removed some rows of the characteristic matrix of $S$ to obtain a new subset $S'$ that has the form $\{a_1\} \times B_1 \cup \cdots \cup \{a_k\} \times B_k$, where $a_i \in V_G$ and $B_i \subset V_H$, and moreover, each $|B_i|$ is at most some constant fraction of $n_H$. For each $B_i$, we can use edge expansion of $H$ to argue that $\epsilon_H$ fraction of edges from $B_i$ are leaving $B_i$. Ideally, we would like then to argue that each such edge, when paired up with any edge from vertex $a_i$, will leave $S'$. This may not be true, however, as such an edge may go to some vertex in $\{a_j\} \times B_j$. To circumvent this problem, we use the assumption that both of our graphs $G$ and $H$ have many self-loops around every vertex (say, half of the degree). In that case, it is easy to argue that each edge leaving $B_i$ in $H$, when paired up with any self-loop around $a_i$, yields an edge of $G \otimes H$ that leaves $S$. Since the number of self-loops around $a_i$ is at least half the degree of $G$, this yields edge expansion at least $\epsilon_H/2$ from each set $\{a_i\} \times B_i$. Since $S'$ is the union of the pairwise disjoint such sets, we get the edge expansion at least $\epsilon_H/2$ from $S'$. Finally, since $S'$ contains a constant fraction of vertices from $S$, we conclude that the edge expansion from $S$ is at least $\Omega(\epsilon_H)$.

We now give the formal proof. We start with a simple averaging result that will allow us to argue the existence of a subset $S'$ of $S$ with required properties.
Claim 4.10. Let $A$ be an $n \times m$ 0-1 matrix with $\delta$ fraction of ones, for some $0 \leq \delta \leq 1/2$. Then either there is a set of rows containing a total of at least $(1/5)\delta nm$ ones so that each row contains less than $(5/6)m$ ones, or there is a set of columns containing at least $(1/5)\delta nm$ ones so that each column contains less than $(5/6)n$ ones.

Proof. Let $a$ be the fraction of rows containing at least $5/6$ fraction of ones each, and let $b$ be the fraction of columns containing at least $5/6$ fraction of ones each. For $\alpha = (4/5)\delta$, suppose that both $a \geq \alpha$ and $b \geq \alpha$. Consider an arbitrary subset $I$ of $[\alpha n] = \alpha n + \gamma_1$ such rows, and an arbitrary subset $J$ of $[\alpha m] = \alpha m + \gamma_2$ such columns. Define the sets

$$R = \{(i,j) \mid i \in I, 1 \leq j \leq m, A_{i,j} = 1\},$$

$$C = \{(i,j) \mid 1 \leq i \leq n, j \in J, A_{i,j} = 1\}.$$

Note that $|R|$ and $|C|$ are $\geq (5/6)\alpha nm$. It is also clear that $|R \cup C| \leq \delta nm$. On the other hand, by the Inclusion-Exclusion principle, we have

$$|R \cup C| = |R| + |C| - |R \cap C|$$

$$\geq (5/6)(\alpha n + \gamma_1)m + (5/6)n(\alpha m + \gamma_2) - (\alpha n + \gamma_1)(\alpha m + \gamma_2)$$

$$= (5/3)\alpha nm - \alpha^2 nm + (5/6 - \alpha)(\gamma_2 n + \gamma_1 m) - \gamma_1 \gamma_2$$

$$\geq (5/3)\alpha nm - \alpha^2 nm,$$

where the last inequality follows from $\alpha = (4/5)\delta \leq 2/5$ as $\delta \leq 1/2$ and from $n, m \geq 2$. We may assume w.l.o.g. that $n, m \geq 2$, since the lemma is easy to prove directly if either $n$ or $m$ equals 1. We continue the inequalities above to get

$$|R \cup C| \geq \left(\frac{4}{3} - \frac{16}{25}\delta\right) \cdot \delta nm \geq \left(\frac{4}{3} - \frac{8}{25}\right) \cdot \delta nm = \frac{76}{75} \cdot \delta nm > \delta nm,$$

a contradiction.

Hence either $a$ or $b$ must be small. Assume $a < (4/5)\delta$. Then the rows with at least $5/6$ fraction of ones contain less than $(4/5)\delta nm$ ones from $A$. This leaves more than $(1/5)\delta nm$ ones among the rows containing less than $5/6$ fraction of ones each. The case of $b < (4/5)\delta$ is analogous.

Proof of Lemma 4.9. Let $S$ be any subset of vertices of $G \otimes H$ of density $\delta \leq 1/2$. Let $A$ be the $0$-$1$ $n_G \times n_H$ characteristic matrix of $S$, where $A_{i,j} = 1$ iff $(i,j) \in S$. Apply Claim 4.10 to $A$ to obtain a submatrix $A'$ of $A$ containing at least $(1/5)|S|$ ones in which either each row or each column contains at most $5/6$ fraction of ones. Suppose that $A'$ is obtained by removing some rows of $A$. Then the matrix $A'$ corresponds to a subset $S'$ of $S$, with $|S'| \geq (1/5)|S|$, such that $S'$ is the disjoint union $S_1 \cup \cdots \cup S_k$, where each $S_i = \{a_i\} \times B_i$ for $a_i \in V_G$ and $B_i \subset V_H$, with $|B_i| < (5/6)n_H$.

Observe that every edge leaving $B_i$ in $H$, when paired with any self-loop of vertex $a_i$ of $G$, yields an edge of $G \otimes H$ leaving $S$. By expansion of $H$, we have

$$|E(B_i, B_i)| \geq \epsilon_H \cdot d_H \cdot \min\{|B_i|, |B_i|\}.$$ 

As $|B_i| + |B_i| = n_H$ and $|B_i| < (5/6)n_H$, we get $|B_i| > (1/5)|B_i|$. Thus, for each subset $S_i$ of $S'$, we have

$$|E(S_i, S_i)| \geq \frac{d_G}{2} \cdot \frac{\epsilon_H \cdot d_H \cdot |S_i|}{5}.$$
Summing over all subsets $S_i$ of $S'$, we get
\[
|E(S', \overline{S})| \geq d_G \cdot d_H \cdot \epsilon_H \cdot |S'| / 10 \\
\geq d_G \cdot d_H \cdot \epsilon_H \cdot |S| / 50,
\]
implying the edge expansion for $S$ at least $\epsilon_H / 50$.

The case of $A'$ obtained by removing some columns of the matrix $A$ is analogous, yielding the edge expansion at least $\epsilon_G / 50$. Thus, the edge expansion in $G \otimes H$ is at least $\min\{\epsilon_G, \epsilon_H\} / 50$.

**Constructivity.** The proof of Lemma 4.9 yields an efficient (uniform NC$^1$) algorithm that, given a set $S$ that is non-expanding in a graph $G \otimes H$, finds a non-expanding set either in $G$ or in $H$. First, the algorithm finds either a subset of vertices in $G$, or a subset of vertices in $H$ to remove so as to get a subset $S' \subseteq S$ as in the proof of Lemma 4.9. In the first case, one of the subsets $S_i = \{a_i\} \times B_i$ of $S'$, for $a_i \in V_G$ and $B_i \subseteq V_H$, must be such that either $B_i$ or its complement is non-expanding in $H$. We can check which by trying at most $|V_G|$ such $B_i$’s. In the second case, an analogous algorithm finds a non-expanding set in $G$.

**Remark 4.11.** The conclusion of Lemma 4.9 is not true if there are not enough self-loops. For example, consider a bipartite edge expander $G = (L \cup R, E)$ with both sides of the same size. Then the tensor product $G \otimes G$ does not expand at all if one considers the set $(L \times L) \cup (R \times R)$ of half the vertices of $G \otimes G$. Of course, $G \otimes G$ is not connected, but by adding a single self-loop to any vertex of $G$, we obtain $G \otimes G$ which also does not expand almost at all even though it is connected.

### 4.3 Replacement product

The replacement product was combinatorially analyzed in [ASS08].

**Lemma 4.12 ([ASS08]).** Let $G = (V_G, E_G)$ be a $D$-regular graph on $n$ vertices, and let $H = (V_H, E_H)$ be a $d$-regular graph on $D$ vertices. If $G$ has edge expansion $\epsilon_G$ and $H$ has edge expansion $\epsilon_H$, then $G \circ H$ has edge expansion at least $\epsilon_G^2 \epsilon_H / 48$.

The proof idea is to partition a given subset $S$ of vertices of $G \circ H$ into $n$ clusters $(\{a_1\} \times B_1) \cup \cdots \cup (\{a_n\} \times B_n)$, where each $a_i \in V_G$ and $B_i \subseteq V_H$. View the clusters where $|B_i|$ is at most some fraction of $|V_H|$ as light, and the remaining clusters as heavy. For every light cluster, one can use the expansion of $H$ to lower-bound the expansion of $B_i$ (within the copy of $H$ associated with vertex $a_i$ of $G$). If there are many vertices in light clusters, we get a good lower bound on the edge expansion of $S$. Otherwise, there are many vertices in heavy clusters. Using the expansion properties of $G$, one can argue in this case that there will be many edges between the set of vertices in heavy clusters and the vertices outside $S$.

We now give the proof of Lemma 4.12, along the lines of [ASS08].

**Proof of Lemma 4.12.** Let $\{1, \ldots, n\}$ be the vertices of $G$, and let $V$ be the $nD$ vertices of $G \circ H$, where $G \circ H$ has degree $2d$. Let $S$ be a subset of $V$ of size at most $|V|/2$. We view the vertices of $G \circ H$ as partitioned into clusters $C_1, \ldots, C_n$, where each $C_i = \{i\} \times V_H$. We partition $S$ into subsets $S \cap C_i$, for $1 \leq i \leq n$. Each such subset is called light, if its size is less than $(1 - \epsilon_G / 4)D$, or heavy otherwise. Denote the (non-empty) light subsets by $L_1, \ldots, L_m$, and the heavy subsets by $H_1, \ldots, H_{m'}$, where $m + m' \leq n$.

**Claim 4.13.** For a light set $L$, we have $|E(L, V \setminus S)| \geq \epsilon_H \cdot \epsilon_G \cdot d \cdot |L| / 4$. 

21
Proof of Claim 4.13. Let $L'$ be the complement of $L$ within the cluster containing $L$. Note that $L' \subseteq V \setminus S$. By the expansion of $H$, we have

$$|E(L, L')| \geq \epsilon_H \cdot d \cdot \min\{|L|, |L'|\}.$$ 

Since $L$ is light, we have $|L'| \geq \epsilon_G \cdot D/4 \geq \epsilon_G \cdot |L|/4$, and the claim follows since $\epsilon_G \leq 1$. \hfill \square

CASE 1: If $|\bigcup L_i| \geq \epsilon_G \cdot |S|/6$, then, by Claim 4.13, we get that

$$|E\left(\bigcup_{i} L_i, V \setminus S\right)| \geq \frac{\epsilon_H \cdot (\epsilon_G)^2 \cdot d \cdot |S|}{24}.$$ 

CASE 2: Otherwise, $|\bigcup L_i| < \epsilon_G \cdot |S|/6$ implies that $|\bigcup H_i| > (1 - \epsilon_G/6) \cdot |S| \geq (5/6) \cdot |S|$. Since each $|H_i| \geq (1 - \epsilon_G/4)D \geq (3/4)D$ and $|\bigcup H_i| \leq |S| \leq Dn/2$, we get that $m' \leq (2/3)n$. As $G$ expands, there are at least $\epsilon_G \cdot D \cdot \min\{m', n/3\}$ edges in $G$ leaving the $m'$ many vertices in $G$ associated with the heavy sets. By the definition of $G \circ H$, all but at most $m'D\epsilon_G/4$ of these edges contribute $d$ parallel edges leaving $\bigcup H_i$ in $G \circ H$ (as each heavy set misses at most $D\epsilon_G/4$ vertices of its cluster). Thus, we get that

$$|E\left(\bigcup H_i, V \setminus \bigcup H_i\right)| \geq \epsilon_G \cdot D \cdot d \cdot (\min\{m', n/3\} - m'/4)$$

$$\geq \frac{\epsilon_G \cdot D \cdot d \cdot m'}{4}$$

$$\geq \frac{5}{24} \cdot \epsilon_G \cdot d \cdot |S|,$$

where the last inequality used the fact that $Dm' \geq |\bigcup H_i| \geq (5/6) \cdot |S|$. Suppose at least $4/5$ of these edges go to $\bigcup L_i$. As each vertex in a particular cluster of $G \circ H$ has $d$ neighbors from the outside clusters, this would mean that $|\bigcup L_i| \geq \epsilon_G \cdot |S|/6$, contradicting the assumption of Case 2. Therefore, at least $1/5$ of these edges miss $\bigcup L_i$ (and hence also $S$), which means that

$$|E\left(\bigcup H_i, V \setminus S\right)| \geq \frac{\epsilon_G \cdot d \cdot |S|}{24}.$$ 

In both cases, we get that the edge expansion of $S$ in the graph $G \circ H$ is

$$\frac{|E(S, V \setminus S)|}{(2d) \cdot |S|} \geq \frac{\epsilon_H(\epsilon_G)^2}{48},$$

as required. \hfill \square

Constructivity. The proof of the contra-positive of Lemma 4.12 is constructive: there is an efficient (uniform NC$^1$) algorithm that, given a non-expanding set $S$ in the graph $G \circ H$, will find a non-expanding set either in $G$ or in $H$. If the assumption of Case 1 holds, then one of the light sets $L_i$, $1 \leq i \leq n$, must be non-expanding in $H$; we can decide which, by testing the edge expansion of each $L_i$. Otherwise, by Case 2, we conclude that the set of vertices in $G$ that correspond to the heavy sets $H_i$ in $G \circ H$ must be non-expanding.
5 Constructing bipartite vertex expanders

Jeřábek [Jeř11b] needs the existence of certain bipartite vertex expanders to formalize the AKS sorting networks in VNC. We define these graphs next. Recall that, for a set $S$ of nodes in a graph $G$, $\Gamma(S)$ denotes the set of all neighbors of vertices in $S$.

Given constants $\alpha \in (0, 1)$ and $A > 1$, a bipartite $(\alpha, A)$ vertex expander is a bipartite graph $G = (L \cup R, E)$, where $|L| = |R| = m$, such that

1. the degree of $G$ is at most $A$, and
2. for all $\ell \leq m$, every set $S \subseteq [m]$ of vertices in either partition with $|S| \geq \alpha \ell$ has $|\Gamma(S)| \geq (1 - \alpha)\ell$.

That is, for every set of vertices of size at least $\alpha \ell$ in one partition, there are at least $(1 - \alpha)\ell$ neighbors in the other partition.

The assumption required by [Jeř11b] is:

For $\alpha = 1/600$, there exist a constant $A$ and a parameter-free NC$^1$ function $G(m)$ such that VNC$^1$ proves "\forall m \in \mathbb{N}, G(m)$ is an $(\alpha, A)$ bipartite vertex expander on $m + m$ vertices".

We will argue that such bipartite vertex expanders can be efficiently obtained from our edge expanders defined above.

**Theorem 5.1.** For any constant $0 < \alpha < 1$, there exist a constant $A = 1$ and an efficient (uniform NC$^1$) algorithm that, for every $m \in \mathbb{N}$, computes the rotation map of an $(\alpha, A)$ bipartite vertex expander on $m + m$ vertices.

**Proof.** We use the edge expander $\tilde{G}$ constructed in Proposition 3.7 with $M = m$, based on the construction of Theorem 3.6. As observed in Remark 3.9, the rotation map of $\tilde{G}$ is in uniform NC$^1$. The graph $\tilde{G} = (\tilde{V}, \tilde{E})$ has $|\tilde{V}| = m$, degree $4d$, and expansion at least $\epsilon/2$, where $\epsilon = 1/1296$. Starting with $\tilde{G}$, we will

1. Convert the edge expander $\tilde{G}$ into a vertex expander, and
2. Turn the latter vertex expander into a bipartite $(\alpha, A)$ vertex expander on $m + m$ vertices.

**1. GETTING A VERTEX EXPANDER FROM AN EDGE EXPANDER:** Let $G = (V, E)$ be the graph $\tilde{G}$ on $m$ nodes constructed above, but with a self-loop added to every node. So $G$ has the constant degree $4d + 1$.

By Proposition 3.7, we have for every set $S \subseteq V$ of size $|S| \leq m/2$ that at least $\epsilon(2d)|S|$ edges are leaving $S$ in $\tilde{G}$. As the degree of $\tilde{G}$ is $4d$, we conclude that the neighborhood $\Gamma(S)$ of $S$ in $G$ contains at least

$$\epsilon \cdot (2d) \cdot |S|/(4d) = \epsilon' \cdot |S|$$

distinct nodes from $\overline{S}$, where $\epsilon' = \epsilon/2$. As $G$ has self-loops around every node, we get

$$|\Gamma(S)| \geq (1 + \epsilon') \cdot |S|, \quad (11)$$

for every subset $S$ of $G$ with $|S| \leq m/2$.

Consider the power graph $G^i$, for any $i \geq 1$. Applying Eq. (11) inductively, we get for every subset $S$ of $G^i$ with $|S| \leq m/2$, and for every $i \geq 1$ that

$$|\Gamma_{G^i}(S)| \geq \min\{m/2, (1 + \epsilon')^i \cdot |S|\}. \quad (12)$$
Now let $S$ be a subset of $V$ of size $|S| \geq m/2$. By Proposition 3.7, we have $|\Gamma^+(S)| \geq \epsilon' \cdot |S|$, where $\Gamma^+(S) = \Gamma(S) \cap \overline{S}$ is the set of new neighbors of $S$. It follows that

$$|\Gamma(S)| \leq (1 - \epsilon') \cdot |S|. \tag{13}$$

Applying Eq. (13) inductively, we get for every $i \geq 1$, and for every subset $S$ of $V$ of size $|S| \geq m/2$ that

$$|\Gamma_i(S)| \leq (1 - \epsilon')^i \cdot |S|. \tag{14}$$

Claim 5.2. There exists a constant $t' = t'(\alpha, \epsilon')$ such that, for every $\ell \leq m$ and every set $S$ of $G^\ell$ with $|S| \geq \alpha \ell$, we have $|\Gamma_i(S)| \geq (1 - \alpha) \ell$.

Proof of Claim 5.2. Consider two cases: $\ell \leq m/2$, and $\ell > m/2$. If $\ell \leq m/2$, then by Eq. (12) we get for $t_1 = \lceil \log_{1+\epsilon'}(1/\alpha) \rceil$ that

$$|\Gamma_i(S)| \geq \min\{m/2, (1 + \epsilon')^{t_1} \cdot \alpha \ell\} \geq \min\{m/2, \ell\} = \ell.$$

If $\ell > m/2$, then $|\overline{S}| \leq m - \alpha \ell < (1 - (\alpha/2)) \cdot m < m$. For $t_2 = \lceil (\log 1/\alpha)/(\log 1/(1 - \epsilon')) \rceil$, we get

$$|\Gamma_i(S)| \leq (1 - \epsilon')^{t_2} \cdot m \leq \alpha \cdot m,$$

and hence, $|\Gamma_i(S)| \geq (1 - \alpha)m \geq (1 - \alpha)\ell$. Taking $t' = \max\{t_1, t_2\}$ concludes the proof. \hfill \Box

2. Getsirig a bipartite vertex expander: Let $G'$ be the vertex expander defined above. Observe that it has $m$ nodes, and has the constant degree $A = (4d + 1)/\epsilon'$. We turn this graph into a bipartite graph by taking two copies of the vertices of $G'$, denoted by $L$ and $R$, connecting nodes $i \in L$ and $j \in R$ by an edge iff $\{i, j\}$ is an edge of $G'$. Claim 5.2 implies that the resulting graph is an $(\alpha, A)$ vertex expander.

Finally, the explicitness of this construction of $(\alpha, A)$ vertex expanders can be argued similarly to the case of the edge expanders of Theorem 3.8: we trace the construction of $G'$ to get an efficient (uniform NC$^1$) algorithm for computing the rotation map of the corresponding bipartite $(\alpha, A)$ expander on $m + m$ vertices. \hfill \Box

Constructivity. Given a non-expanding set $S$ for the bipartite graph constructed in Theorem 5.1, we can efficiently (in uniform NC$^1$) reconstruct a non-expanding set for the graph $	ilde{G}$ of Proposition 3.7 upon which this bipartite graph was based. Indeed, if $S$ is non-expanding in the graph $G'$ from the second stage of the proof of Theorem 5.1, then one of the sets $S_i = \Gamma_{G^i}(S)$, for $0 \leq i \leq t'$, is non-expanding in $\tilde{G}$ (where $S_0 = S$). As $t'$ is constant, we can determine such a set $S' = S_i$ in uniform NC$^1$, given the adjacency matrix of $\tilde{G}$. This $S'$ is also a non-expanding subset for $\tilde{G}$. Then, $S'$ corresponds to a non-expanding subset $\overline{S}$ of the graph $\tilde{G}$ constructed by Theorem 3.6.

6 Formalizing the construction in bounded arithmetic

This section discusses the formalization of the expander graph construction in the theory VNC$^1$ of bounded arithmetic. A high-level description of how we formalize the expander graph construction in VNC$^1$ is as follows:
1. The first step is to establish (in Section 6.4) that VNC$^1$ can define the operations of graph powering, replacement product, and tensoring. From this it follows that VNC$^1$ can carry out the definition of $G_{t+1}$ from $G_t$, for the graphs $G_t$ defined in Section 3. Similarly, VNC$^1$ can carry out the construction of $G_i$ from $G_{i'}$ and $G_{i''}$ as in (3).

2. For the second step, we wish to use induction on $t$ to prove the existence of the graph $G_t$ for suitable $t$. However, since VNC$^1$ does not support induction on $\Sigma^B_1$-formulas, we cannot use the usual induction axioms for VNC$^1$. Instead, we exploit the fact that the graph $G_{i+1}$ has size quadratic in the size of $G_i$, namely $|G_{i+1}| = \Theta(|G_i|^2)$. This large growth rate allows us to use $\Sigma^B_1$-induction to prove the existence of $G_t$ for arbitrary (first-order) integers $t$. For this, Theorems 6.3 and 6.4 of Section 6.3 prove that the needed induction principle is provable in VNC$^1$. The intuition is that the computational content of the induction axioms corresponds to composing logarithmic depth circuits, and that since the $G_i$'s are growing quadratically, arbitrary composition of logarithmic depth circuits for the $G_i$'s yields a circuit which is still of only logarithmic depth. The same $\Sigma^B_1$-induction will also be used to prove the existence of the graphs $G_i$, exploiting the fact that the size of $G_i$ is quadratic in the sizes of $G_{i'}$ and $G_{i''}$.

3. Theorems 6.3 and 6.4 give the needed induction principle for handling compositions of circuits, but more work is needed for VNC$^1$ to formalize the iterated composition of circuits. What we mean by “iterated composition” of circuits is that there are multiple circuits (about $|\langle t \rangle|$ many circuits) which are arranged with the outputs of one circuit feeding into the inputs of the next circuit. To formalize this circuit composition in VNC$^1$, we need to modify Cook and Morioka’s definition [CM05] of $\text{TreeRec}$ tree recursion in VNC$^1$. The problem with the $\text{TreeRec}$ form of tree recursion is that the second order inputs to a circuit defined by tree recursion are not used at the input gates of the circuit, but rather are used throughout the circuit, indeed potentially at every gate in the circuit. To fix this, Section 6.2 introduces a modified version of tree recursion, called $\text{TreeRec}'$, which allows the use of second order inputs $X_0(i)$ only as input values. This allows composition of circuits using the inputs $X_0$ for the iteratively computed values. The $\text{TreeRec}'$ tree recursion and the new induction principle of Section 6.3 then suffice to define $G_t$ by using recursively the definition of $G_{i+1}$ from $G_i$.

4. The fourth step is to prove the expansion properties of $G_{i+1}$ follow from those of $G_i$. Or, more precisely, proving that if $G_{i+1}$ does not have the desired edge expansion then $G_i$ also does not. This is done by Lemmas 6.5-6.7 which show how to formalize in VNC$^1$, the arguments of Section 4 about graph powering, replacement product, and tensor product. The arguments in Section 4 are constructive, and as we argue below, they can be adapted to VNC$^1$ with relatively minor modifications.

5. The fifth step is to use induction on $t$ to prove the expansion properties for $G_t$. This is done in Theorem 6.8; its proof again utilizes the induction principle introduced in Section 6.3. This shows that VNC$^1$ can prove the existence of expander graphs.

6. The sixth, and final step, is to note that the proof of Theorem 5.1 can be carried out in VNC$^1$, so VNC$^1$ proves the existence of bipartite vertex expanders.

This proof is given below. We start by proving some useful properties of VNC$^1$ in Sections 6.1–6.3. We show in Section 6.4 that VNC$^1$ can express relevant graph properties. Section 6.5 argues that the Cauchy-Schwarz inequality can be proved within VNC$^1$. Section 6.6 shows that the edge expansion properties of our graph operations can be proved within VNC$^1$.  

25
6.1 Defining $\text{NC}^1$ functions within $\text{VNC}^1$

Cook and Morioka [CM05, Lemma 13] show that $\text{VNC}^1(\text{TreeRec})$ can prove the $\Sigma^B_0(\text{TreeRec})$-COMP axioms. They then define the $\text{FNC}^1$ functions $F$ by using $\Sigma^B_0$-formulas $\varphi(i, \vec{x}, \vec{X})$ and terms $t(\vec{x}, \vec{X})$ and defining the string $F(\vec{x}, \vec{X})$ by

$$F(\vec{x}, \vec{X})(j) \leftrightarrow j < t(\vec{x}, \vec{X}) \land \varphi(j, \vec{x}, \vec{X}).$$

(15)

They also show that the $\Sigma^B_1$-definable functions of $\text{VNC}^1$ are precisely the $\text{FNC}^1$ functions [CM05, Theorem 17]. Recall that a $\Sigma^B_1$-definition is given by $\text{VNC}^1$ proof of $\exists Y \varphi(\vec{x}, \vec{X}, Y)$ where $\varphi \in \Sigma^B_1$; this serves as a definition of the string function $\vec{x}, \vec{X} \rightarrow Y$.

The definition of $\text{FNC}^1$ functions using (15) is equivalent to the usual definition of the $\text{FNC}^1$ functions as the functions whose bit graphs are computable in $U_{E^*}$-uniform $\text{NC}^1$, or equivalently are computable in $\text{ALogTime}$. Those functions are computed by a family $\{C_n\}_n$ of fanin $\leq 2$ Boolean circuits, taking inputs of length $n$ and having depth $O(\log n)$. The $U_{E^*}$-uniformity condition was defined by Ruzzo [Ruz81] and means that the circuits $C_n$ are described by two functions $g(i, n)$ and $p(i, w, n)$ which are computable in the linear time hierarchy (equivalently, they have $\Sigma^B_0$ graphs). The first function $g(i, n)$ returns the type of gate $i$ in $C_n$. The second function $p(i, w, n)$ takes as input also a $w \in \{0, 1\}^*$: the bits of $w$ describe a path in the circuit starting at gate $i$ and following successively the first or second input to gates according to the bits of $w$. The value of $p(i, w, n)$ is the index of the gate reached by following this path specified by $w$ starting from gate $i$ in $C_n$. The functions $g$ and $p$ are in the linear time hierarchy; however, since they have inputs of length $O(\log n)$, they run in time $O(\log n)$ using a constant number of alternations. For more details, see [Ruz81].

We will need to carefully analyze the effect of composing $\text{FNC}^1$ functions; for this reason it is important that the existence of $U_{E^*}$-uniform $\text{NC}^1$ circuits for $\text{FNC}^1$ functions can be proved by the theory $\text{VNC}^1$. This follows from Theorem 6.2 below.

6.2 A modified tree recursion

$\text{TreeRec}$ acts like a fanin two, Boolean circuit where the internal gate types are given by $\varphi = \varphi(i, \vec{x}, \vec{X})$. A disadvantage of this definition of $\text{TreeRec}$ is that the side parameters $\vec{X}$ can be used unrestrictedly by the $\Sigma^B_0$-formulas $\varphi$ and $\psi$. The formula $\varphi(i, \vec{x}, \vec{X})$ defines the type of gate number $i$ when the circuit has $\vec{x}, \vec{X}$ as inputs. Likewise, $\psi(i, \vec{x}, \vec{X})$ defines the True/False value of the $i$-th input. This differs from the usual conventions of having a circuit have fixed gate types, and having the inputs affect only the values of input gates. It also makes it difficult to define the notion of composing circuits, with the outputs of one family of circuits serving as the inputs to another circuit.

We define a new formulation of tree recursion called $\text{TreeRec'}$ to address this problem. In a $\text{TreeRec'}$ definition, one of the second order inputs, $X_0$, will serve as an “ordinary” input to the circuit, with the values $X_0(j)$ specifying the True/False values on inputs to the circuit. The other second order inputs, $\vec{X}'$, can be used to define gate types similarly as is done by $\text{TreeRec}$. This allows recursive computations on the value $X_0$ to be formalized with composition of circuits.

We assume $X_0$ is one of the side string parameters $\vec{X}$, so $\vec{X}$ is $X_0, \vec{X}'$. We modify the definition of $\text{TreeRec}$ so that the values $X_0(i)$ are used only as inputs to the $\text{TreeRec}$ circuit, and are not used to determine the gate types; in particular, $X_0$ is not used by $\varphi$. The basic construction for the definition of $\text{TreeRec'}$ is

---

3This definition of $\text{FNC}^1$ is same as what Cook and Morioka [CM05] call “$\text{VNC}^1(\text{FNC}^1)$”. We use just “$\text{FNC}^1$ to keep the notation less cumbersome.
that a single gate in a TreeRec circuit, of gate type $\phi[-,-]$: \[
\phi[-,-] \quad \begin{array}{c}
p \\swarrow \quad \searrow \quad q \\
\end{array}
\] (16)
is replaced by a small tree of binary gates

\[
\begin{array}{c}
\vee \\
p \quad p \\
\quad \vee \\
\quad p \quad p \\
\quad \wedge \\
\quad p \quad q \quad \phi_{TT} \\
\quad \vee \\
\quad p \quad p \quad p \\
\quad \vee \\
\quad p \quad q \quad \phi_{TF} \\
\quad \wedge \\
\quad p \quad q \quad \phi_{FT} \\
\quad \vee \\
\quad p \quad q \quad \phi_{FF} \\
\end{array}
\] (17)

Here the binary gate $r = s$ is $\neg r \land s$; and the values $\phi_{pq}$ are the truth values of $\phi(i, a, \vec{x}, \vec{X})[p, q]$. By inspection, the circuit is depth four and fanin two: the top $\lor$ gate branches on the value of $p$; the next two $\lor$ gates branch on $q$. The last two levels select the correct value of $\phi_{pq}$, for $p = T,F$ and $q = T,F$ based on the values of $p$ and $q$. In other words, the circuit (17) implements a “lookup table”, using the values $p$ and $q$ to select the appropriate value $\phi_{pq}$. Assuming that the four values of $\phi_{pq}$ are correctly computed, the effect of replacing the binary gates (16) with the circuits (17) gives a circuit of depth $4|a|$ computing the same result as the original TreeRec circuit of depth $|a|$.

We wish to replace the four leaf nodes of (17) labelled $\phi_{pq}$ with Boolean circuits which have as inputs only the values $X_0(i)$. Since $\phi$ is $\Sigma^B_0$-formula, such circuits can easily be described by a polynomial time function of $i, \vec{x}, \vec{X}'$. These circuits are formed by applying the Paris-Wilkie transformation to $\phi$, namely by replacing bounded quantifiers in $\phi$ with conjunctions and disjunctions, and hardcoding the values of $\vec{x}$ and $\vec{X}'$ (but not $X_0$) as constants. The result is that each leaf $\phi_{pq}$ of the circuit (17) can be replaced by a fanin two circuit which (a) has as inputs only $X_0(j)$’s and constants, (b) is size $\leq q(|a|, |\vec{x}|)$ and depth $\leq q(|a|, |\vec{x}|)$ for some polynomial $q$, and (c) there is a $\Sigma^B_0$-definable number function $f(p, q, i, a, \vec{x}, \vec{X})$ of VNC$^1$ which outputs a succinct description of the circuit. VNC$^1$ is able to straightforwardly define $f$ and prove all these properties.

With this construction in hand, we define a modified version of tree recursion:

**Definition 6.1.** Let $\phi(i, a, \vec{x}, y_0, \vec{X}')[p, q]$ be a $\Sigma^B_0$-formula and $k(i, a, \vec{x}, y_0, \vec{y}')$ a $\Sigma^B_0$-definable number function. The $\Sigma^B_0$-TreeRec$'$ property for $\phi$ and $k$ is given by $B^{\phi,k}(a, \vec{x}, X_0, \vec{X}', Z)$: \[
(\forall i < a)[(Z(i) \iff \phi(i, a, \vec{x}, |X_0|, \vec{X}')[Z(2i+1), Z(2i+2)]) \land (Z(a + i) \iff X_0(k(i, a, \vec{x}, |X_0|, |\vec{X}'|))]].
\]
The defining axioms for the predicate symbols $R^{\phi,k}(i, a, \vec{x}, X_0, \vec{X}')$ are the formulas $B^{\phi,k}(a, b, \vec{x}, X_0, \vec{X}', R^{\phi,k})$ and $i \geq 2a \rightarrow \neg R^{\phi,k}(i, a, \vec{x}, X_0, \vec{X}')$. (18)

Note that the gate type depends only on $|X_0|$, not on the values of $X(\cdot)$. VNC$^1$ proves that (18) uniquely specifies all values of $R^{\phi,k}$. Furthermore, it is not hard to see that VNC$^1$ proves the existence of string objects satisfying the conditions of (18). Thus, we may conservatively extend VNC$^1(TreeRec)$ by adding all these predicate symbols along with their defining axioms. The resulting theory is called VNC$^1(TreeRec, TreeRec')$.
The main advantage of TreeRec’ definitions is that they can explicitly give \( U_{E^*} \)-uniform log-depth circuits. For this, we assume that \( X_0 \) is the only second-order input (so \( \vec{X}' \) is missing). We also assume that \( a = s(\vec{x}, |X_0|) \) for some \( V^0 \)-term \( s \). The type of gate \( i \) is determined by \( i, |X_0| \) and \( \vec{x} \) and can be computed in time polynomially bounded by \( \vec{x} \) and \( |X_0| \). It is usually convenient to assume in addition that each \( x_i < |X_0|^{O(1)} \), so that we can think of \( |X_0| \) as the size of the input (up to a polynomial); in fact, often \( \vec{x} \) is missing, so the only input is \( |X_0| \). The other condition needed for \( U_{E^*} \)-uniformity is that there must be a linear time hierarchy algorithm (i.e., a \( \Sigma^B_0 \) formula) determining the extended connection language for the connectivity of gates in the circuit. Since the circuit is formed as a binary tree, with a natural numbering system for gates, the extended connection language of the circuit is trivial. Specifically, suppose \( w \in \{0, 1\} \) is a string of bits and \( i \) is a gate. Interpret bits “0” and “1” as selecting the first or second input to a gate, and let \( w \) specify a path starting at gate \( i \), and traversing inputs according to the bits of \( w \). The gate at the end of this path is gate \( i' \) where \( i' \) has binary representation obtained by concatenating the binary representation of \( i \) and the string \( w \). The type of gate \( i \) can be defined with a \( \Sigma^B_0 \)-formula using the \( \Sigma^B_0 \)-formula \( \varphi \) and the \( \Sigma^B_0 \)-defined function \( k \). Thus, with the assumptions stated above, a TreeRec’ definition defines a \( U_{E^*} \)-uniform circuit.

The next theorem states that every \( \Sigma^B_0 \)(TreeRec)-property has log-depth, fanin two, Boolean circuits in the form used by TreeRec’.

**Theorem 6.2.** Let \( \chi(\vec{x}, X_0, \vec{X}') \) be a \( \Sigma^B_0 \)(TreeRec)-formula. Then there are a \( \Sigma^B_0 \)-formula \( \varphi(i, a, \vec{x}, y_0, \vec{X}') \), a \( \Sigma^B_0 \)-defined function \( k(i, a, \vec{x}, y_0, |\vec{y}'|) \), and a \( V^0 \)-term \( s(\vec{x}, |X_0|, |\vec{X}'|) \) so that \( \mathsf{VNC}^1(\text{TreeRec}, \text{TreeRec}') \) proves

\[
\chi(\vec{x}, X_0, \vec{X}') \leftrightarrow R^{|\varphi|, k}(0, s(\vec{x}, |X_0|, |\vec{X}'|), \vec{x}, X_0, \vec{X}').
\]

\( \Sigma^B_0 \)(TreeRec)-properties may involve composing multiple TreeRec predicates with built-in function symbols, then combining them with Boolean operations and first-order quantifiers. Theorem 6.2 states that any such property \( \chi \) may be expressed as a TreeRec’: the advantage is that this gives an explicit \( \mathsf{NC}^1 \) representation of \( \chi \); namely in terms of logarithmic depth Boolean circuits. “Logarithmic” means as a function of the values \( \vec{x} \) and of the sizes \( |X_0|, |\vec{X}'| \) of the second order inputs \( X_0, X' \).

**Proof of Theorem 6.2.** (Sketch.) The proof is by induction on the complexity of the formula \( \chi \). The induction steps handle Boolean connectives and first-order quantifiers. The cases of Boolean connectives are easily handled; e.g., a conjunction (AND) is handled by joining two TreeRec’ computations with a new \( \wedge \)-gate (after padding as needed to make the two TreeRec’ have the same depth). First-order quantifiers \( \forall y \) and \( \exists y \) are handled by combining multiple TreeRec’ computations with multiple \( \wedge \)-gates or \( \vee \)-gates (respectively). These constructions are fairly straightforward, and we leave the details to the reader.

The base case for the proof by induction handles atomic predicates of the forms \( t_1 = t_2 \) and \( t_1 \leq t_2 \) and \( X_i(t_1) \) where \( t_1 = t_1(\vec{x}, X_0, \vec{X}') \) and \( t_2 = t_2(\vec{x}, X_0, \vec{X}') \) are VNC(TreeRec, TreeRec’) terms. It is easy to give (constant depth) TreeRec’ circuits checking equality (=) and inequality (\( \leq \)), and to output a value of \( X_i \), once TreeRec’ definitions are given for the bitgraphs of \( t_1 \) and \( t_2 \). However, we need to prove that the bitgraph of a VNC(TreeRec, TreeRec’) term \( t(\vec{x}, X_0, \vec{X}') \) can be expressed with a TreeRec’ definition. This fact is proved by induction on the complexity of terms. The base cases for this induction concern the function symbols of \( V^0 \) and the TreeRec predicates. The symbols of \( V^0 \) have very simple logarithmic depth circuits of course. And we already discussed about how to represent TreeRec predicates as TreeRec’ predicates.

The induction step handling composition of function symbols is harder. For simplicity, consider the situation of composing two string functions \( F(\vec{x}, X_0, \vec{X}') \) and \( F'(\vec{x}, \vec{X}') \) to define

\[
G(\vec{x}, \vec{X}') = F(\vec{x}, F'(\vec{x}, \vec{X}'), \vec{X}').
\]
This case is simplified mainly because $G$ has no dependence on any designated $X_0$. When proving this case, since $G$ has no dependence on $X_0$, we end up with essentially a $\text{TreeRec}'$ expression for the bitgraph of $G$, not a $\text{TreeRec}'$ expression. But then, we can choose another member of $\bar{X}'$ to serve as the new $X_0$, and convert this $\text{TreeRec}$ definition to $\text{TreeRec}'$ form by the construction above.

Suppose that $\varphi, k, s$ and $\varphi', k', s'$ respectively give $\text{TreeRec}'$ definitions of the bitgraphs of $F$ and $F'$. Note that $k'$ is not needed, since $F'$ does not depend on $X_0$. We wish to give a $\text{TreeRec}'$ definition for the bitgraph of $G$ by using the $\text{TreeRec}'$ definition of $F$ with its leaves (inputs) replaced with $\text{TreeRec}'$ definitions of $F'$. For this, we can explicitly define $\varphi_G$ and $s_G$ for a $\text{TreeRec}'$ definition of $G$ in terms of $\varphi, k, s$ and $\varphi', k', s'$. (The function $k_G$ is not needed.) What it means for $\varphi, k, s$ to give a $\text{TreeRec}'$ definition of the bitgraph of $F$ is that whenever $B^{\varphi,k}(s(j, \bar{x}, |X_0|, |\bar{X}'|), j, \bar{x}, X_0, \bar{X}', Z)$ holds, we have $F'(\bar{x}, X_0, \bar{X}')(j) \leftrightarrow Z(0)$. And, $\varphi', k', s'$ similarly define the bitgraph of $F'$. Since $F$ and $F'$ have polynomial growth rate, we may assume w.l.o.g. that $s$ and $s'$ do not depend on the input $j$. Also w.l.o.g., $s(\bar{x}, |X_0|, |\bar{X}'|) = 2|s(\bar{x}, |X_0|, |\bar{X}'|)| - 1$ always and similarly for $s'(\bar{x}, |\bar{X}'|)$, and they are monotonically non-decreasing. Define $\text{lvl}(i) = [i + 1] - 1$ so that $\text{lvl}(i)$ is the level of the node $Z(i)$ in the in the tree used in subtree recursion. It is simple to see that if the tree is moved so that it is rooted at $\bar{a}$ instead of $0$, then the node computing the value of $Z(i)$ is moved to position $m = \ell 2^{\text{lvl}(i)} + i$. Conversely, $\ell = \lfloor m/2^{\text{lvl}(m) - \text{lvl}(\bar{a})} \rfloor$, namely the higher order $\text{lvl}(\ell)$ bits of $m$. And $i = m - \ell 2^{\text{lvl}(m) - \text{lvl}(\bar{a})}$, i.e., the remaining low order bits of $m$. To improve readability for these expressions, define $\ell(m, s) = \lfloor m/2^{\text{lvl}(m) - \text{lvl}(s)} \rfloor$ and $\ell(m, s) = m - \ell 2^{\text{lvl}(m) - \text{lvl}(s)}$.

Let $q(\bar{x}, y_0, \bar{y})$ be a polynomial such that $|F'(\bar{x}, |X_0|, |\bar{X}'|)| \leq q(\bar{x}, |X_0|, |\bar{X}'|)$. We set

$$s_G(j, \bar{x}, |X_0|, |\bar{X}'|) = 2|s(\bar{x}, |X_0|, |\bar{X}'|)| + |s'(\bar{x}, q(\bar{x}, |X_0|, |\bar{X}'|), |\bar{X}'|)| - 1$$

This makes $s_G(j, \bar{x}, |X_0|, |\bar{X}'|)$ an upper bound on the depth of a Boolean circuit computing the $j$-th bit of $G(\bar{x}, \bar{X}')$. Note that $s_G(j, \bar{x}, |X_0|, |\bar{X}'|)$ satisfies the depth condition. The first level (at the root) of the circuit is a depth $|s(\bar{x}, |X_0|, |\bar{X}'|)|$ tree recursion ($\text{TreeRec}'$) computation, the nodes at depths $|s(\bar{x}, |X_0|, |\bar{X}'|)|$ are the roots of trees computing a bit of the value of $F'(\bar{x}, \bar{X}')$. The functions $\ell(m, s)$ and $\ell(m, s)$ with $s = s(\bar{x}, |X_0|, |\bar{X}'|)$ help us index into the latter subtrees. Define

$$\varphi_G(i, j, \bar{x}, \bar{X}') \leftrightarrow$$

$$[\text{lvl}(i) < |s(\bar{x}, |X_0|, |\bar{X}'|)| \wedge \varphi(i, s(\bar{x}, |X_0|, |\bar{X}'|), \bar{x}, \bar{X}')]$$

$$\vee [\text{lvl}(i) = |s(\bar{x}, |X_0|, |\bar{X}'|)| \wedge \varphi'(i, j, s(\bar{x}, |X_0|, |\bar{X}'|), k(\ell(i, s(\bar{x}, |X_0|, |\bar{X}'|), \bar{x}, X_0), |\bar{X}'|), \bar{x}, X_0, |\bar{X}'|)]$$

It is not hard to verify this gives a $\text{TreeRec}$ definition for the function $G$. In the last part, the point is that the subtree rooted at position $\ell(i, s(\bar{x}, |X_0|, |\bar{X}'|))$ is computing the value of bit

$$k(\ell(i, s(\bar{x}, |X_0|, |\bar{X}'|), \bar{x}, X_0, |\bar{X}'|))$$

of $F'(\bar{x}, \bar{X}')$. The value $\ell(i, s(\bar{x}, |X_0|, |\bar{X}'|))$ gives the relative position of $i$ within that subtree.

### 6.3 A conservation result

We now prove the closure of $\text{VNC}^1$ under a rule of inference based on a “telescoping” iteration. This turns out to be exactly what is needed for the formalization of the expander graph construction inside $\text{VNC}^1$. We write $\sqrt{a}$ for the greatest integer at most $\sqrt{a}$.

**Theorem 6.3.** Suppose $\chi(X)$ is a $\Sigma_0^B$-formula containing only $X$ free. and let $\psi(a)$ be $(\exists X \leq a) \chi(X)$. Also suppose $\text{VNC}^1$ proves

$$\forall a(\psi(a) \rightarrow \psi(\sqrt{a})).$$

Then $\text{VNC}^1$ proves $\psi(a) \rightarrow \psi(1)$, and thus also proves $\chi(Y) \rightarrow (\exists X \leq 1) \chi(X)$. (19)
Proof. By the Witnessing Lemma for VNC^1, since VNC^1 proves (19), there is an FNC^1 function \( F \) such that

\[
\chi(Y) \land Y \leq a \rightarrow \chi(F(a, Y)) \land |F(a, Y)| \leq \sqrt{a}
\]

is also provable. Furthermore, the bit graph of the function \( F \) is definable with a TreeRec' definition, with \( Y \) playing the role of \( X_0 \). Since we are only interested in \( F(a, Y) \) when \( |Y| \leq a \), we can assume that the height of the TreeRec' tree recursion is \( O(|a|) \).

For a VNC^1 proof of \( \psi(a) \rightarrow \psi(1) \), we iterate the function \( F \). For notational simplicity, henceforth assume \( |Y| = a \). We must construct a uniform description (a TreeRec' description) of a circuit computing the bits of the iterated \( F \); the construction is to be carried out in VNC^1 as a function of \( a = |Y| \) only. Set \( a_0 = a \), and \( a_{m+1} = \sqrt[a]{a_m} \). Define \( Y_0 = Y \), and set \( Y_{m+1} = F(Y_m) \). We can form a circuit \( C \) using bits of \( Y \) as inputs, and computing all the bits of all the \( Y_i \)'s, namely by composing the circuits for the iterated applications of \( F \). We have \( |a_{m+1}| \leq \frac{1}{2}|a_m| \), so \( |a_m| \leq |a|/2^m \), therefore the depth of \( C \) is \( \sum_m O(|a_m|) = \sum_m O(|a|/2^m) = O(|a|) \). That is, \( C \) is an NC^1 circuit. We shall argue that \( C \) can be defined inside VNC^1. This will imply that \( C \) is VNC^1-provably in \( U_{E^*} \)-uniform NC^1, and allow us to prove Theorem 6.3.

By padding circuit depths, the tree recursion circuit for the bit graph of \( F(Y) \) w.l.o.g. has depth \( |s(a)| \) exactly equal to \( c \cdot |a| \) for \( c \) a fixed constant. More generally, we may assume that the circuit computing a bit of \( Y_{m+1} \) from the bits of \( Y_m \) has depth exactly \( c \cdot |a_m| \). Let \( M \) be the first value such that \( a_M = 1 \). As functions of \( a \), the values \( a_1, a_2, \ldots, a_M \) are NC^1 computable, and are \( \Sigma^B_0 \)-definable in VNC^1, namely by the algorithm which non-deterministically guesses the entire sequence of values, and then in parallel verifies that each \( a_{m+1} \leq a_m \). For \( m_1 < m_2 \), the partial sums \( s_{m_1, m_2} = c \cdot (|a_{m_1}| + |a_{m_1+1}| + \cdots + |a_{m_2}|) \) are also NC^1-computable and \( \Sigma^B_0 \)-definable in VNC^1, using vector addition. In addition, \( s_{m_1, m_2} \leq \sigma_{1, M} = O(|a|) \). Thus we can define \( s_{m_1, m_2} = 2^{s_{m_1, m_2}} - 1 \), and \( s_{m_1, m_2} \) equals the combined depth of the tree recursion circuits computing the bit graph of \( Y_{m_2} \) from the bitgraph of \( Y_{m_1} \).

The TreeRec' definition of the \( d \)-th bit of \( F(Y) \) is given by a \( \Sigma^B_0 \)-predicate \( \varphi(i, d, s(a), a) \) and \( \Sigma^B_0 \)-defined number function \( k(i, d, a) \). From these, letting \( j \) range over \( 1, \ldots, M \), a TreeRec'-definition of the \( d \)-th bit of \( Y_j \) is given by a \( \Sigma^B_0 \)-defined predicate \( \varphi^*(i, j, d, a) \) and a polynomial time function \( k^*(i, d, a) \) which are computed as follows. (The argument \( s_G(a) \) is omitted from \( \varphi^* \) as it is not needed.)

The TreeRec'-definition for the \( d \)-th bit of \( Y_j \) describes a circuit \( C \) composed of layers; each layer computes bits of a \( Y_m \) from bits of \( Y_{m-1} \). The levels separating the layers are specified by values \( \lambda_m = c(|a_{j+1}| + \cdots + |a_m|) \); so that \( \lambda_m \) is the depth at which bits of \( Y_m \) are computed. The input \( i \) to \( \varphi^* \) is the index of a gate in \( C \). For \( \lambda_m \leq \ell_m(i) \), let \( \ell_m(i) \) gives the parent of \( i \) at level \( \lambda_m \). Fixing \( m_0 \) to be the greatest value such that \( \lambda_m \leq \ell_m(i) \), let \( \ell = \ell(i, \ell_m) \); this means that \( i \) is a gate in the level computing a bit of \( Y_{m_0} \), and \( \ell \) is the relative position of gate \( i \) within the subcircuit computing a bit of \( Y_{m_0} \). In fact, the gate \( i \) is inside (nested) subcircuits computing bits of each \( Y_m \) for \( j \leq m \leq m_0 \). Let \( d_m = d_m(i, a) \) denote the bit of \( Y_m \) which is being computed. Of course, \( d_j = d \). For \( m > j \), we have \( d_m = k(i, d_{m-1}, a_m) \). The \( d_m \) values are computable with a FNC^1 function which first existentially guesses the the values \( d_j, \ldots, d_m \), and universally checks that each \( d_m = k(\ell, d_{m-1}, a_m) \) holds. Finally, the definition of the gate \( i \) is given by the predicate \( \phi(\ell, d_m, s(a_{m_1}, a_{m_0+1})) \). Putting all this together gives \( \varphi^*(i, j, d, a) \) as an NC^1 computable function. Set \( k(i, d, a) = d_m(i + s_G(a)) \) where \( s_G(a) = \sigma_{1, M} \); this gives the bit of \( Y \) which is the input to the circuit at this leaf gate.

The predicate \( \varphi^* \) gives a TreeRec' definition of the \( d \)-th bit of \( Y_j \). We conclude (provably in the theory VNC^1(\text{TreeRec}, \text{TreeRec}') that, for any fixed \( a, Y \), there is a string object \( Y^*(j, d) \) encoding all the bits of all the \( Y_j \)'s; that is \( Y^*(j, d) \leftrightarrow Y_j(d) \). The above arguments defining \( \varphi^* \) and proving the existence of \( Y^* \)
can all be formalized in VNC¹ (TreeRec, TreeRec'). Therefore, that theory proves

\[ \varphi(Y_j) \land |Y_j| \leq a_j \rightarrow \varphi(Y_{j+1}) \land |Y_{j+1}| \leq a_{j+1}. \]

From this, \( \varphi(Y) \rightarrow \varphi(Y_M) \land |Y_M| < 1 \) follows by \( \Sigma^B_0 \)-IND. The conclusion of Theorem 6.3 follows.

Theorem 6.3 used a descending induction; a similar theorem holds also for ascending induction:

**Theorem 6.4.** Suppose \( \varphi(X) \) is a \( \Sigma^B_0 \)-formula containing only \( X \) free. Also suppose VNC\(^1\) proves

\[ \varphi(Y) \rightarrow (\exists X)(|X| \geq |Y|^2 \land \varphi(Y)). \]

Then VNC\(^1\) proves \((\exists Y)\varphi(Y) \rightarrow (\forall x)(\exists X)(|X| > x \land \varphi(X)).\]

The proof of Theorem 6.4 is almost identical to the proof of Theorem 6.3. The most important difference is that now the sequence \( a_0, a_1, \ldots, a_M \) is increasing instead of decreasing, and \( a_i \) is a lower bound on \(|Y_i|\) instead of an upper bound. The proof also assumes w.l.o.g. that \(|Y_m| = O(1)\) throughout the construction in order to control the growth rates.

### 6.4 Expressing expander graph properties in VNC¹

We now discuss how VNC\(^1\) can express properties about graphs, adjacency matrices, expansion properties, and graph constructions such as powering, tensor product and replacement product. A graph \( G \) on \( n \) vertices will be encoded in VNC\(^1\) as a string object (a second order object). Here \( n \) is a number (a first-order object), and the intent is to represent \( G \) in terms of its adjacency matrix. The \((i, j)\)-th entry of the adjacency matrix is the number of edges between vertices \( i \) and \( j \). It is represented by a three-place second order predicate \( A(i, j, k) \) where \( A(i, j, k) \) is true when there are exactly \( k \) edges between \( i \) and \( j \). (Strictly speaking, we should write \( A((i, j, k)) \), but we suppress this notation.) Each \( i, j, k \) is a number (a first order object); it will be important that we always have \( k < p(n) \) for some fixed polynomial \( p \), since then \( k \) is \( \Sigma^B_0 \)-definable from \( A, i, j \), and we can write \( k = A(i, j) \) for the value of \( k \).

Row vectors and column vectors (containing numbers) are likewise representable by strings, with \( A(i, k) \) meaning that the \( i \)-th entry of the vector is equal to \( k \).

With these conventions it is easy for VNC\(^1\) to \( \Sigma^B_0 \) or \( \Sigma^B_1 \)-define many properties of the graph \( G \) encoded as above. We illustrate this with several examples.

- For \( u < n \), the set of edges containing vertex \( u \) can be defined as the set

\[ E(\{u\}) = \{(u, v, k) : (\exists k' \leq p(n))(k < k' \land A(u, v, k')). \]

Note this allows for multiedges. The degree of \( v \) is \( |E(\{v\})| \) and can be \( \Sigma^B_1 \)-defined with the Numones function. \( G \) has degree \( d \) if each \( u \in [n] \) has degree \( d \). There will always be a polynomial upper bound \( p(n) \) on the degree.

- For \( U \subset [n] \), the set \( E(U, U) \) is defined similarly as

\[ E(U, U) = \{(i, j, k) : i \in U \land j \notin U \land (\exists k' \leq p(n))(k < k' \land A(i, j, k')). \]

31
• Rational numbers \( p/q \) are represented by pairs of integers \((p, q)\) (not necessarily in reduced form). The usual ordering \( p/q < p'/q' \) is of course definable by \( pq' < p'q \), where \( q, q' > 0 \). Pairs of rational numbers may be added or multiplied or divided as usual.

The proof of the Cauchy-Schwarz theorem, and more generally the proofs of expansion properties in Section 4, argue about sums of vectors of rational numbers. VNC\(^1\) can define summations of vectors of integers [CN10], but it is not clear whether it can define summations of vectors of arbitrary rational numbers. This will be handled in our VNC\(^1\) proofs by clearing the denominators so that we can argue about summations of integers instead of about summations of rational numbers. In our applications, the least common multiple of the denominators will be easily computed, making it easy to clear the denominators.

• The edge expansion of a degree \( d \) graph \( G \) can thus be defined by as in equation (1) with \( V = [n] \). This, however, is not a \( \Sigma^B_1 \)-definition, since it requires minimizing over all subsets \( U \subset [n] \). Instead we can define the property “\( G \) has edge expansion \( > p/q \)” as

\[
(\forall U < n) \left( 0 < |U| \leq \frac{n}{2} \rightarrow |E(U, \overline{U})| > \frac{p}{q} \right).
\]

This is a \( \Pi^B_1 \)-condition. Recall that “\( \forall U < n \)” is quantifying over all subsets of \([n]\).

• A rotation map is encoded by a second order object \( Rot(u, i, v, j) \) with the meaning that the \( i \)-th edge of \( u \) is the same as the \( j \)-th edge of \( v \). We can relate the rotation map \( Rot \) and the adjacency matrix \( A \) by letting the \( i \)-th edge from \( u \) to \( v \) be the edge \( \langle u, v, k \rangle \) such that

\[
|\{\langle u, i', v, j \rangle : Rot(u, i', v, j) \land i' < i\}| = k
\]

Furthermore, the adjacency matrix \( A \) is \( \Sigma^B_1 \)-definable in terms of \( Rot \), since \( A(u, v) = k \) holds exactly when there are exactly \( k \) values \( \langle i, j \rangle \) such that \( Rot(u, i, v, j) \). Since \( v, j \) are uniquely determined by \( u, i \), we also use the notation \( Rot(u, i) = \langle v, j \rangle \).

It is also possible to \( \Sigma^B_1 \)-define a canonical rotation map as a function of the adjacency matrix.

Graph operations are also readily defined by VNC\(^1\):

• To add self-loops to convert a \( d \)-regular \( G \) to a \( 2d \)-regular \( G' \), define the adjacency matrix \( A'(u, v, k) \) as

\[
( u \neq v \land A(u, v, k)) \lor (u = v \land (\exists k' \leq d)(A(u, v, k') \land (k = k' + d)))).
\]

• (Graph Powering.) Let \( k > 1 \) be fixed. VNC\(^1\) can \( \Sigma^B_1 \)-define the graph power \( G^k \) from \( G \) as follows. We write \( \langle i_1, \ldots, i_k \rangle \) for an efficient sequence coding so that each \( \langle i_1, \ldots, i_k \rangle \) is represented by an integer \( < d^k \). Then \( Rot(u, \langle i_1, \ldots, i_k \rangle) = (v, \langle j_1, \ldots, j_k \rangle) \) holds iff

\[
(\exists(u_0, \ldots, u_k))(u_0 = u \land u_k = v \land \bigwedge_{s=1}^{k}(Rot(u_{s-1}, i_s) = (u_s, j_{k-s+1}))).
\]

Since \( k \) is fixed and each \( u_i < n \), the quantifier is a bounded number quantifier.

• Similar arguments give \( \Sigma^B_1 \)-definitions of Tensor Product and Replacement Product. The constructions are straightforward and we leave the details to the reader.
These constructions, along with Theorem 6.4, allow VNC\(^1\) to prove the existence of the graphs \(G_i\) as defined by (2). Fix constants \(d\) and \(c\), and fix a \((2d)\)-regular \(G_0\) with edge expansion \(c_0\). Also, fix a rotation map \(R_0 = Rot_{G_0}\) for \(G_0\). Given \(G_i\) and \(Rot_i\), for \(i \geq 0\), VNC\(^1\) can prove the existence of \(G_{i+1}\) satisfying (2) along with the existence of \(Rot_{i+1}\). Furthermore, by Theorem 6.4, VNC\(^1\) can prove the existence of a second-order object encoding a sequence of graphs and rotation maps

\[
(G_0, Rot_0), (G_1, Rot_1), (G_2, Rot_2), \ldots, (G_{|a|}, Rot_{|a|}),
\]

so each \(G_{i+1}\) and associated rotation map \(Rot_{i+1}\) is obtained from \(G_i\) and \(Rot_i\) by Equation (2). Letting the constant \(D = 2(4d)^2c\) as before, each \(G_i\) has \((|V_0| \cdot 4D)^2 / D\) many vertices, provably in VNC\(^1\). (See Theorem 3.5.) The size of \(G_{i+1}\) is greater than the square of the size of \(G_i\); indeed, \(|V_{i+1}| = D \cdot |V_i|^2\).

Therefore, Theorem 6.4 applies, to show that VNC\(^1\) can \(\Sigma_1^B\)-define the sequence (20) as function of \(a\), and hence can \(\Sigma_1^B\)-define \(G|a|\) and \(Rot|a|\) as functions of \(a\).

Similar, only slightly more complicated, arguments allow VNC\(^1\) to prove the existence of the graphs \(\tilde{G}_i\) as defined by (3). Now \(i\) can be an arbitrary first-order (integer) value \(i = a\), not just a length \(|a|\). Fix appropriate constants \(d = 2^\ell\) and \(c\), and for \(i \leq 2c\ell+8\), fix graphs \(\tilde{G}_i\) with edge expansion \(\geq 1/1296\) and their rotation maps \(Rot_i\). Using induction on \(\Sigma_1^B\)-formulas, VNC\(^1\) proves the existence of a sequence of values \(k_0, \ldots, k_s\) such that \(k_0 = a\) and each \(k_{i+1} = \lceil (k_i - 2c\ell - 5)/2 \rceil\), and such that \(s\) is the first value where \(k_s < 2c\ell+7\). Given both \(\tilde{G}_{k_{i+1}}\) and \(\tilde{G}_{k_{i+1}+1}\) and their rotation maps \(Rot_{k_{i+1}}\) and \(Rot_{k_{i+1}+1}\), and using the definition (3), VNC\(^1\) can prove the existence of both \(\tilde{G}_{k_i}\) and \(\tilde{G}_{k_i+1}\) and their rotation maps. Furthermore, the sizes of \(\tilde{G}_{k_i}\) and \(\tilde{G}_{k_i+1}\) are both greater than the square of the size of \(\tilde{G}_{k_i+1}\). Therefore, by Theorem 6.4 again, VNC\(^1\) can prove the existence of a second-order object encoding a sequence of pairs of graphs and rotation maps:

\[
(\tilde{G}_{k_i}, Rot_{k_i}, \tilde{G}_{k_i+1}, Rot_{k_i+1}), (\tilde{G}_{k_{i-1}}, Rot_{k_{i-1}}, \tilde{G}_{k_{i-1}+1}, Rot_{k_{i-1}+1}), \ldots, (\tilde{G}_{k_0}, Rot_{k_0}, \tilde{G}_{k_0+1}, Rot_{k_0+1}),
\]

(21) with successive pairs of expander graphs obtained via (3). Since \(k_0 = a\), this shows that VNC\(^1\) can \(\Sigma_1^B\)-define \(\tilde{G}_a\) and \(Rot_a\) as functions of \(a\).

It is immediate from the definition of \(G_i\), using induction on \(i\), that VNC\(^1\) proves that each \(G_i\) has degree \(2d\) (for the appropriate value of \(d\)). Likewise VNC\(^1\) proves that each \(\tilde{G}_i\) has degree \(2d\). It is more difficult to prove that VNC\(^1\) proves \(G_i\) and \(\tilde{G}_i\) have the edge expansion properties of Theorems 3.5 and 3.6. This is discussed in the next sections.

### 6.5 Formalizing Cauchy-Schwarz

We now discuss how to formalize the proof of the Cauchy-Schwarz lemma in VNC\(^1\). This proof depends on summations of vectors encoded by second-order order objects; in addition, care must be taken to handle summations of rational numbers. Claims 4.3-4.5 supporting the proof of Lemma 4.2 use similar manipulations of summations, and those arguments can be slightly more formalized in VNC\(^1\).

Let \(f, g \in \mathbb{R}^n\) be vectors. The Cauchy-Schwarz identity states that \(\langle f, g \rangle^2 \leq \langle f, f \rangle \cdot \langle g, g \rangle\), where \(\langle \cdot, \cdot \rangle\) denotes inner product. This can be proved as follows. Define \(f_\parallel = \langle f, g \rangle \cdot g / \langle g, g \rangle\), namely the component of \(f\) parallel to \(g\). Define \(f_\perp = f - f_\parallel\). Then,

\[
\langle f, f \rangle = \langle f_\parallel + f_\perp, f_\parallel + f_\perp \rangle
\]

\[
= \langle f_\perp, f_\perp \rangle + 2 \cdot \langle f_\perp, f_\parallel \rangle + \langle f_\parallel, f_\parallel \rangle
\]

\[
= \langle f_\perp, f_\perp \rangle + 2 \cdot 0 + \langle f, g \rangle^2 / \langle g, g \rangle.
\]
The Cauchy-Schwarz inequality follows immediately, since $\langle f_\bot, f_\bot \rangle \geq 0$.

To formalize Cauchy-Schwarz, $\text{VNC}^1$ uses second-order objects $F$ and $G$ encoding integer vectors $f$ and $g$. As discussed above, $F(i, k)$ indicates that $f_i = k$, and similarly for $G(i, k)$. The permitted values of $i$ and $k$ are bounded by integers, e.g., $i < n$ and $k < m$. Given $F$ and $G$ encoding arbitrary (integer) vectors $f$ and $g$, and an integer $c \geq 0$, $\text{VNC}^1$ can prove the existence of vectors encoding $f + g$ and $cf$ using $\Sigma^B_0$-comprehension. Vector summation is well-known to be $\Sigma^S_1$-definable in $\text{VNC}^1$ [CN10] based on the formalization of carry-save-addition ([Bus87]); in fact, since we are only summing vectors of integers, the $\text{Numones}$ function suffices to define vector summation. This allows $\text{VNC}^1$ to define summations such as $\sum_i f_i$, and $\langle f, g \rangle = \sum_i f_i g_i$. In addition, identities such as $\sum_i f_i + \sum_i g_i = \sum_i (f_i + g_i)$ and $\sum_i cf_i = c \sum_i f_i$ and $\langle f, g + h \rangle = \langle f, g \rangle + \langle f, h \rangle$ are $\text{VNC}^1$-provable.

This vector summation is almost enough to permit $\text{VNC}^1$ to formalize the above proof of Cauchy-Schwarz; however, the proof above has a division by $\langle g, g \rangle$, and uses summations of rationals, not of integers. As was discussed earlier, it is not clear whether $\text{VNC}^1$ can be extended to form summations of arbitrary vectors of rational numbers — at least, when the denominators might be relatively prime — because of the difficulty of computing a least common multiple of the denominators. But, for Cauchy-Schwarz for integer vectors $f$ and $g$, $\text{VNC}^1$ can just multiply through by $\langle g, g \rangle$ and use integer vector summation. That is, set $f_2 = \langle f, g \rangle g$, and $f_1 = \langle g, g \rangle f_\bot = \langle g, g \rangle f - f_2$. Then, following the reasoning above, $\text{VNC}^1$ proves

$$\langle g, g \rangle^2 \langle f, f \rangle^2 = \langle f_1, f_1 \rangle + \langle f, g \rangle^2$$

From this, $\langle f_1, f_1 \rangle \geq 0$ and taking square roots, $\text{VNC}^1$ proves the Cauchy-Schwarz inequality.

### 6.6 Formalizing edge expansion properties in $\text{VNC}^1$

We first discuss how Lemma 4.2 can be stated by $\text{VNC}^1$, and how its proof can be formalized in $\text{VNC}^1$. The statement of Lemma 4.2 uses two distributions: an arbitrary distribution $\pi$ and the uniform distribution $u$. The uniform distribution $u$ is just the vector with constant entries $1/n$: these of course all share $n$ as their common denominator. However, even to state Lemma 4.2 in $\text{VNC}^1$, we need to make the additional assumption that the entries in $\pi$ are rational numbers that share a common denominator. (Otherwise it is unclear whether the vector summation implicit in the statement of Lemma 4.2 is definable in $\text{VNC}^1$.) We henceforth make this assumption about $\pi$. Similarly, the Claims 4.3, 4.4 and 4.5 use vectors $f$, $h$ and $g$; we also must assume that these vectors contain either integers, or rationals with a common denominator.

With these assumptions the proofs given earlier formalize directly in $\text{VNC}^1$; however, there are a few new ingredients in the proof that deserve mention. First, in equation (4), there is a common denominator of $4d^2$, which $\text{VNC}^1$ handles by multiplying the equality by $4d^2$. The Cauchy-Schwarz inequality is being applied $|V|$ times in parallel, to vectors of length $d$; as discussed above this is formalizable in $\text{VNC}^1$. Second, in the first large displayed equation for the proof of Claim 4.4, a double summation is reordered: this also is readily formalizable in $\text{VNC}^1$, and $\text{VNC}^1$ can prove summations are preserved under arbitrary reorderings. The next large displayed equation for the proof of the same claim uses $\epsilon$, and hence is working with rational numbers instead of integers. As usual, $\text{VNC}^1$ handles this by clearing the denominators. (This happens whenever $\epsilon$ appears in an equation.) Third, the proof of Claim 4.5 uses a factor $(2d \cdot \sum_i g^+(i)^2)^{-1}$ in many of its equations. $\text{VNC}^1$ handles this by multiplying the equations by $(2d \cdot \sum_i g^+(i)^2)^{-1}$. Fourth, in deriving equation (8), it is necessary to reorder the vertices in $V$ so that $e_1 \geq e_2 \geq \cdots \geq e_n$ so as to apply Claims 4.4 and 4.5. Since $\text{VNC}^1$ can sort vectors of integers (sorting can be defined in terms of the $\text{Numones}$ function), this can also be formalized by $\text{VNC}^1$. Fifth, the proof Lemma 4.2 began by defining $e = \pi - u$. By the assumption that $\pi$ contained rationals with a common denominator, the same holds for $e$. Hence, the invocations of Claims 4.3, 4.4 and 4.5 can all use vectors of rational numbers with common denominators.
To formalize Lemma 4.6, we assume for simplicity that $\sqrt{\delta}$ is a rational. The proof starts with an arbitrary, but fixed subset $U \subset V$. The uniform distribution $u$, and the uniform distribution $u_U$ are both vectors of rationals with a common denominator. The proofs of Lemma 4.6 and Claim 4.7 now formalize straightforwardly in $\text{VNC}^1$.

This lets $\text{VNC}^1$ prove Lemma 4.1 using Lemmas 4.2 and 4.6, where we assume w.l.o.g. that $k$ is even, so that the value for $\sqrt{\delta}$ is a rational. The argument about “Constructivity” at the end of Section 4.1 is directly formalizable in $\text{VNC}^1$. For $U \subset V$, we write $\text{edge-exp}_G(U)$ to denote the edge expansion ratio

$$\text{edge-exp}_G(U) = \frac{|E(U, U)|}{d \cdot \min\{|U|, |U^c|\}}.$$

The use of the notation $\text{edge-exp}_G(U)$ implicitly assumes that $U$ is nonempty and is not equal to $V$. Lemma 4.1 as formalized in $\text{VNC}^1$ becomes:

**Lemma 6.5.** Let $k$ be even. $\text{VNC}^1$ proves the following: Suppose $G^k$ is the graph power of $G$ as defined in Section 6.4, and $V$ is the common vertex set of $G$ and $G^k$. Then

$$(\exists U)[U \subset V \land \text{edge-exp}_{G^k}(U) < \left|\frac{1}{2} - \frac{k^2}{4}\right|] \rightarrow (\exists U)[U \subset V \land \text{edge-exp}_G(U) < \epsilon].$$

The proofs of Claim 4.10 and Lemma 4.9 as given in Section 4.2 formalize directly in $\text{VNC}^1$, at least assuming that the edge expansions $\epsilon_G$ and $\epsilon_H$ are rational numbers:

**Lemma 6.6.** $\text{VNC}^1$ proves the following: Let $G = (V_G, E_G)$ be a $d_G$-regular graph with $d_G/2$ self-loops at every vertex and $H = (V_H, E_H)$ be a $d_H$-regular graph with $d_H/2$ self-loops at every vertex. Let $\epsilon = \min\{\epsilon_G, \epsilon_H\}$. Then,

$$(\exists U)[U \subset (V_G \otimes V_H) \land \text{edge-exp}_{G \otimes H}(U) < \epsilon/50] \rightarrow (\exists U)[U \subset V_G \land \text{edge-exp}_G(U) < \epsilon_G] \lor (\exists U)[U \subset V_H \land \text{edge-exp}_H(U) < \epsilon_H].$$

(The lemma could be simplified somewhat by taking $\epsilon = \epsilon_G = \epsilon_H$.)

Similarly, the proofs of Lemma 4.12 and Claim 4.13 given in Section 4.3 are formalized directly in $\text{VNC}^1$, assuming $\epsilon_G$ and $\epsilon_H$ are rational numbers:

**Lemma 6.7.** $\text{VNC}^1$ proves the following: Let $G = (V_G, E_G)$ be a $D$-regular graph on $n$ vertices, and let $H = (V_H, E_H)$ be a $d$-regular graph on $D$ vertices. Let $\epsilon = \epsilon_G^2\epsilon_H/48$, and let $V_{G \circ H}$ denote the vertices of $G \circ H$. Then,

$$(\exists U)[U \subset V_{G \circ H} \land \text{edge-exp}_{G \circ H}(U) < \epsilon] \rightarrow (\exists U)[U \subset V_G \land \text{edge-exp}_G(U) < \epsilon_G] \lor (\exists U)[U \subset V_H \land \text{edge-exp}_H(U) < \epsilon_H].$$

Finally, the arguments in Section 3.3 also formalize in $\text{VNC}^1$ to combine Lemmas 6.5-6.7 to prove the existence of expander graphs. For this, we need to formulate the arguments so as to apply Theorem 6.3. We first how to prove the existence of the edge expanders $G_i$ in $\text{VNC}^1$. To talk about the edge expansion of $G_i$, we encode a subset $U$ of $V_i$ using a string $Y$ of length exactly $|V_i| + 1 = (|V_0| \cdot 4D)^2/D + 1$, by letting $Y = U \cup \{|V_i|\}$. It follows from the discussion at the end of Section 6.4 that $\text{VNC}^1$ can $\Sigma^B_1$-define $G_i$ as a function of $|V_i|$, hence as a function of $Y$. 

35
Let \( A(Y) \) express the conditions that (a) \(|Y| = |V_i| + 1 \) for some \( i \), and (b) \( Y \) encodes a subset \( U \) of \( V_i \) such that \( \text{edge-exp}_{G_i}(U) < 1/1296 \). The (contrapositive of the) argument in Section 3.3, formalized in \( \text{VNC}^1 \), shows that the following is \( \text{VNC}^1 \) provable:

\[
(\exists Y \leq a) A(Y) \rightarrow (\exists Y \leq \sqrt{a}) A(Y).
\]  

(22)

For \( i = 0 \), this uses the fact that \( G_0 \) has edge expansion \( \geq 1/1296 \), and since \( G_0 \) is a constant graph, this can be checked by a enumerating all of the finitely many subsets.

Applying Theorem 6.3 to (22) gives that \( \text{VNC}^1 \) proves

\[
(\exists Y \leq a) A(Y) \rightarrow (\exists Y \leq 1) A(Y).
\]

There are only four possible \( Y \)'s with \(|Y| \leq 1 \). The righthand side, \( (\exists Y \leq 1) A(Y) \), is a false \( \Sigma_0^B \)-formula asserting a finite property. Hence, \( \text{VNC}^1 \) can trivially disprove \( (\exists Y \leq 1) A(Y) \) by direct evaluation. Therefore, \( \text{VNC}^1 \) proves \( \neg (\exists Y) A(Y) \), i.e., can prove that any \( V_i \) must be an expander. This completes the proof of the following.

**Theorem 6.8.** There is a constant \( d \) so that \( \text{VNC}^1 \) proves the existence of arbitrarily large, degree \( 2d \) graphs with edge expansion \( \geq 1/1296 \). Namely, \( \text{VNC}^1 \) proves

\[
(\forall a)(\exists V, E)[|V| \geq a \wedge (V, E) \text{ is a degree } 2d \text{ graph} \\
\wedge (\forall U)(U \subseteq V \rightarrow \text{edge-exp}_{(V, E)}(U) \geq 1/1296)].
\]

In fact, there is a \( \Sigma^B_1 \)-definable function \( G \) of \( \text{VNC}^1 \) so that that \( \text{VNC}^1 \) proves

\[
(\forall a)[G(a) \text{ is a degree } 2d \text{ graph } G(a) = (V, E) \text{ with } |V| \geq a \\
\wedge (\forall U)(U \subseteq V \rightarrow \text{edge-exp}_{(V, E)}(U) \geq 1/1296)].
\]

\( \text{VNC}^1 \) can also prove the existence of edge expander graphs of arbitrary size.

**Theorem 6.9.** There is a constant \( d = 2^e \) and a \( \Sigma^B_1 \)-definable function \( G \) of \( \text{VNC}^1 \) so that \( \text{VNC}^1 \) proves

\[
(\forall a)[G(a) \text{ is a } 4d \text{-regular graph } G(a) = (V(a), E(a)) \text{ with } |V| = a \\
\wedge (\forall U)(U \subseteq V \rightarrow \text{edge-exp}_{(V, E)}(U) \geq 1/(2 \cdot 1296)].
\]

**Proof.** Pick appropriate constant values for \( d \) and \( c \). \( \text{VNC}^1 \) starts by proving the existence of \( \tilde{G}_i \) for the least \( i \) such that \( 2^i \geq a \). As already argued, \( \text{VNC}^1 \) can prove the existence of the sequence \( k_0, \ldots, k_s \) with \( k_0 = i \), and each \( k_{i+1} = [(k_i - 2c\ell - 5)] \) and \( s \) the first value with \( k_s < 2c\ell + 7 \). In addition, by Section 6.4, \( \text{VNC}^1 \) can prove the existence of second-order objects encoding edge expanders \( \tilde{G}_j = (\tilde{V}_j, \tilde{E}_j) \) for every value \( j = k_i \) or \( j = k_i + 1 \) with \( i \leq s \). Recall that \(|\tilde{V}_j| = 2^j\). Let \( A(Y) \) express the condition that for some \( i \leq s \), either (a) \(|Y| = 2^{k_i} + 1 \) and \( Y \) encodes a subset \( U \) of \( \tilde{V}_{k_i} \) such that \( \text{edge-exp}_{\tilde{G}_{k_i}}(U) < 1/1296 \), or (b) \(|Y| = 2^{k_i+1} + 1 \) and \( Y \) encodes a subset \( U \) of \( \tilde{V}_{k_i+1} \) such that \( \text{edge-exp}_{\tilde{G}_{k_i+1}}(U) < 1/1296 \). The (contrapositive) of the argument in Section 3.3, now shows that

\[
(\exists Y \leq a) A(Y) \rightarrow (\exists Y \leq \sqrt{a}) A(Y).
\]

is \( \text{VNC}^1 \)-provable. Applying Theorem 6.3 gives that \( \text{VNC}^1 \) proves

\[
(\exists Y \leq a) A(Y) \rightarrow (\exists Y \leq 1) A(Y).
\]
Therefore, VNC$^1$ proves $\neg(\exists Y \leq a) A(Y)$, i.e., it proves the edge expansion properties for arbitrary $Y$, and hence the edge expansion properties of $\bar{G}_i$.

We have $a \leq 2^t < 2a$. Now, VNC$^1$ can readily formalize the proof of Proposition 3.7, constructing a graph on exactly $a$ vertices, of degree $4d$, with edge expansion at least $1/(2 \cdot 1296)$.

Finally, VNC$^1$ can also formalize the argument given in Section 5 to construct bipartite vertex expanders. The only new proof ingredient is the use of logarithms to define $t_1$ and $t_2$ in the proof of Claim 5.2. VNC$^1$ can define rational approximations to logarithms; here we need only integers $t$ define rational approximations to logarithms; here we need only integers $t$.

The pigeonhole principle was shown to have polynomial-size monotone sequent calculus proofs by Ježábek [Jež11a].

7 Application to monotone sequent calculus

In [PB94], Pudlák and Buss introduced a proof system for reasoning with monotone formulas, motivated by strong lower bounds results for monotone circuits, and posed the question whether similar difference in complexity holds in the propositional proof system setting. More specifically, they formulated monotone sequent calculus and asked whether any non-monotone proof of a monotone sequent can be replaced by a monotone proof at most polynomially larger. In [Pud99], Pudlák further investigated this question, focusing in particular on the pigeonhole principle. There, he discussed the need to formalize properties of monotone counting formulas such as AKS sorting networks of [AKS83], and asked whether there are small proofs of basic properties of counting formulas.

The pigeonhole principle was shown to have polynomial-size monotone sequent calculus proofs by Atserias, Galesi and Gavaldà in [AGG01]; this paper was the first to use the name MLK for this system. The same paper also gave quasipolynomial-size proofs of basic counting principles. Building upon the latter result, Atserias, Galesi and Pudlák [AGP02] show that, in contrast to monotone circuit classes, monotone proof systems are nearly as powerful as non-monotone ones: polynomial-size non-monotone proofs can be simulated by monotone ones of quasipolynomial size. The quasipolynomial blowup is introduced in the [AGG01] proofs of certain properties of threshold formulas.

To prove that every LK proof can be converted into an MLK proof of quasipolynomial size, [AGP02] use monotone threshold formulas to eliminate negated variables. A threshold formula $TH^n_k(x_1, \ldots, x_n)$ asserts that at least $k$ variables $x_i$ are 1. The standard inductive definition builds $TH^n_k$ as a disjunction of $TH_{i/2}^{n/2}(x_1, \ldots, x_{n/2}) \land TH_{j/2}^{n/2}(x_{n/2+1}, \ldots, x_n)$ for all pairs $i, j \leq n/2$ such that $i + j \geq k$. This definition yields quasipolynomial size formulas $TH^n_k$, and thus gives only quasipolynomial size LK proofs of properties of $TH^n_k$. If LK is polynomially bounded, then so is MLK (as in this case properties of threshold functions would have polynomial-size LK proofs). More generally, they use the following lemma based on results from [AGG01]:

As VNC$^1$ is a subtheory of VNC$^4$, Theorem 6.10 is stronger than the assumption needed by Ježábek [Jež11a].
Lemma 7.1 ([AGP02, Lemma 6]). Let $TH^p_k$ be a polynomial-size monotone threshold formula. Then MLK polynomially simulates LK on monotone sequents, provided that there are polynomial-size LK proofs of the following sequents:

1. $TH^p_k(x_1, \ldots, x_n) \to 0$ and $1 \to TH^p_0(x_1, \ldots, x_n)$ for every $n$ and $k > n$.

2. $TH^p_k(x_1, \ldots, x_i, 0, \ldots, x_n) \to TH^p_{k+1}(x_1, \ldots, x_i, 1, \ldots, x_n)$ for all $n, k, i$ with $0 \leq k, i \leq n$.

Such polynomial-size monotone threshold formulas can be built using the classic construction of monotone log-depth sorting networks by Ajtai, Komlós and Szemerédi [AKS83], known as AKS sorting networks. A sorting network can be thought of as a circuit with $n$ outputs gates, which contain the values of the input gates in sorted order. That is, the $k^{th}$ output of a sorting network is 0 iff there are at least $k$ 0s among inputs to the network. The construction of AKS sorting networks is fairly involved; see [Pat90, Sei09] for expositions. At the end of the paper, Atserias et al. note that replacing their threshold formulas with monotone NC$^1$ sorting networks of Ajtai, Komlós and Szemerédi would remove the blowup and allow for polynomial-size simulation, provided the relevant properties can be proven with NC$^1$ reasoning (not necessarily monotone).

Jefábek [Jeff1b] has shown just that, under the assumption that bipartite expanders graphs with appropriate parameters can be constructed, and their properties proven in NC$^1$ reasoning. More precisely, Jefábek [Jeff1b] has shown that AKS sorting networks (Paterson’s [Pat90] variant) are indeed formalizable in a theory VNC$^1$ of NC$^1$ reasoning, under the assumption of the existence of a family of bipartite expanders provable in VNC$^1$ (with parameters as in Claim 5.2). The theory VNC$^1$ is somewhat stronger than VNC$^1$ that we use, in that it can evaluate and reason about less uniform families of log-depth circuits; however, proofs in VNC$^1$ still translate into polynomial-size LK proofs [Jeff1a]. Thus, Jefábek obtains the following result:

Theorem 7.2 ([Jeff1b, Theorem 5.5]). Suppose that there exists a constant $D$ and a parameter-free NC$^1$ function $G(m)$ such that VNC$^1$ proves that for all numbers $m$, $G(m)$ is a $\langle 1/600, D \rangle$ bipartite $m+m$ expander. Then MLK polynomially simulates LK on monotone sequents.

The construction in Theorem 5.1 gives expanders with the appropriate parameters, and Theorem 6.10 shows that it can be done in VNC$^1$ (and thus VNC$^1$). As this proves the assumption of Theorem 7.2, we immediately get the following corollary.

Theorem 7.3 (Main application). MLK polynomially simulates LK on monotone sequents.

8 Conclusions and open problems

From the point of view of bounded reverse mathematics, the area that tries to pinpoint the minimal reasoning power needed to prove mathematical theorems, it is very interesting to understand what is the complexity of reasoning required to prove properties of expander graphs, and thus what is the the complexity of reasoning in expander-based proofs such as the known proofs of SL = L [Rei08, RV05]. This paper makes a step in this direction by showing that an expander construction can be formalized within the system VNC$^1$.

A number of open questions remain. Can we formalize expanders in a weaker theory than VNC$^1$, e.g., the system of TC$^0$ reasoning? Can Reingold’s result that undirected graph connectivity is in deterministic logspace [Rei08] be formalized in the system of logspace reasoning? The analysis of graph powering given
Finally, as was already asked by Jeřábek, can the AKS construction of expanders be modified to yield $U_{E'}$-uniform sorting networks?

Acknowledgements  We want to thank Denis Thérien and Pascal Tesson for inviting V.K., A.K., and M.K. to the 2007 McGill Complexity Workshop in Barbados, where this paper was initiated. V.K. and A.K. also wish to thank Josh Buresh-Oppenheim, Shlomo Hoory, and Rahul Santhanam for our many discussions on expander graphs. V.K. and A.K. are particularly thankful to Russell Impagliazzo for inviting them to spend a semester at UCSD in the spring of 2016, where this work was finally completed. S.B. thanks Amir Akbar Tabatabai and Raheleh Jalali for useful discussions on VNC, and Rosalie Iemhoff and Anupam Das for discussions on intuitionistic logic. We also thank Anupam Das for his comments on our paper, and Albert Atserias for clarifying to us the history of the MLK proof system. We are especially grateful to Emil Jeřábek for carefully reading our manuscript and pointing out some errors in the early versions.

References


\[4\] Indeed, for a graph $G$ with edge expansion $\epsilon$, our Lemma 4.1 says that the edge expansion of the graph $G^k$ is approximately $\epsilon^2k/16$. When $\epsilon$ is sub-constant (as may be the case when we start with an arbitrary, not necessarily expander, graph $G$) but $k$ is constant (which is necessary for Reingold’s algorithm), this does not show any improvement in the edge expansion of $G^k$. Similarly, Lemma 4.12 only says that the edge expansion of $G \circ H$ is at least $\Omega(\epsilon^2)$ (assuming $H$ is a good edge expander). Since the point of applying the replacement product to $G$ and $H$ in (a variant of) Reingold’s algorithm would be to create a small-degree graph while not losing more than a constant fraction of the edge expansion of $G$, Lemma 4.12 does not help in the case of graphs $G$ with sub-constant edge expansion.


Irit Dinur. The PCP theorem by gap amplification. *J. ACM*, 54(3), June 2007. 1


