Bounded Arithmetic and Formalizing Probabilistic Proofs

by

Dai Tri Man Lê

A thesis submitted in conformity with the requirements
for the degree of Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto

Abstract

Bounded Arithmetic and Formalizing Probabilistic Proofs

Dai Tri Man Lê
Doctor of Philosophy
Graduate Department of Computer Science
University of Toronto
2014

The first theme of this thesis investigates the complexity class CC and its associated bounded-arithmetic theory. Subramanian defined CC as the class of problems log-space reducible to the comparator circuit value problem (Ccv). Using the Cook-Nguyen method we define the two-sorted theory *VCC* whose provably-total functions are exactly the CC functions. To apply this method, we show CC is the same as the class of problems computed by uniform AC^0 circuits with unbounded Ccv oracle gates. We prove that $VNL \subseteq VCC \subseteq VP$, where *VNL* and *VP* are theories for the classes NL and P respectively. We strengthen Subramanian's work by showing that the problems in his paper are indeed complete for CC under many-one AC^0 reductions. We then prove the correctness of these reductions in *VCC*.

The second theme of this thesis is formalizing probabilistic proofs in bounded arithmetic. In a series of papers, Jeřábek argued that the universal polynomial-time theory *VPV* augmented with the surjective weak pigeonhole principle $\mathsf{sWPHP}(\mathcal{L}_{\mathsf{FP}})$ for all *VPV* functions is the 'right' theory for randomized polynomial-time reasoning in bounded arithmetic.

Motivated from the fact that no one had used Jeřábek's framework for feasible reasoning about specific interesting randomized algorithms in classes such as RP and RNC^2, we formalize in *VPV* the correctness of two fundamental RNC^2 algorithms for testing if a bipartite graph has a perfect matching and for finding a bipartite perfect matching.

Using Moser's recent constructive proof technique for the Lovász Local Lemma, we show that $VPV + \mathsf{sWPHP}(\mathcal{L}_{\mathsf{FP}})$ proves the existence of a satisfying assignment for every instance of k-SAT in which every clause shares a variable with up to 2^{k-3} other clauses. This result implies the existence of a randomized polynomial-time algorithm for find satisfying assignments such k-SAT instances.

The remainder of this thesis was motivated by the lack of fundamental probability concepts like random variables, expectation and variance in Jeřábek's work, which means basic yet useful theorems like Markov's inequality, Chebyshev's inequality, linearity of expectation, etc were not available in his work. By choosing suitable definitions of random variables, approximate probability and approximate expectation, we are able prove these theorems and utilize them to prove the Goldreich-Levin theorem within the conservative extension $HARD^A$ of $VPV + \mathsf{sWPHP}(\mathcal{L}_{\mathsf{FP}})$.

Dedication

To my parents and sisters
for their unconditional love, guidance, and support.

Acknowledgements

I want to thank my advisor Steve Cook. It has been a real privilege to have Steve as my supervisor. My profound gratitude to him for his wisdom, constructive criticism, and mentorship. During my transition from the software engineering group to the theory group in 2008, I remember asking Steve about what it takes to work in theory and he said:

> "To do theory one only needs a good taste when choosing problems, and the persistency to follow them through."

A good taste like Steve's is exceptional, but I have learned in a hard way that when things get difficult, persistency is crucially needed.

Many thanks to my coauthors Steve Cook, Yuval Filmus, Yuli Ye and Wesley George for our joint papers that form the bulk of this thesis.

I would also like to thank my other committee members, Charles Rackoff, Alasdair Urquhart and Toni Pitassi, and my external examiner Jan Krajíček. They provided valuable feedback on my work and helped shape this thesis into its current form. Charles Rackoff told me his interesting perspective on the Lovász Local Lemma, which inspired the work in Chapter 5.

I would like to thank Phuong The Nguyen, whose work with Steve laid the foundation for all of my work on bounded arithmetic. He also gave me very useful feedback on my research direction. I am also very fortunate to have him as a good friend.

I am indebted to Jan Krajíček and Emil Jeřábek in Prague. Jan generously invited me to the Fall Schools of Logic and Complexity 2009 and 2010, and the Special Semester in Logic and Complexity 2011 which he organized. Emil's seminal work on approximate counting in bounded arithmetic is the main inspiration for the main results of this thesis. The meetings with Emil during my visit at Prague really helped me gain deeper understanding of his work.

Gregory Moore played an important role in my decision of going to graduate school. His wonderful undergraduate set theory courses at McMaster University inspired me to do research relating to set theory and mathematical logic. He is always willing to give me valuable advice whenever I have academic or personal difficulty. I am extremely grateful for his friendship for the last 10 years.

I am grateful to Yuli Ye for being a wonderful friend during the last five years. I will always miss the walks we had around the university campus regardless of the weather and our memorable conversations about research, philosophy and life. His emotional support and advice during the most difficult time have contributed greatly to the finishing of this thesis.

Many thanks to Yuval Filmus for listening to my half-baked ideas and for his valuable feedback. His construction of universal comparator circuits opens possibilities for making progress in understanding the complexity class CC. While his scientific ability and work ethics are second to none, he is also my great friend.

I would like to thank Sergey Gorbunov for organizing the trips to classical music concerts, and for taking part in our "food critic" adventures. He is also a great friend and a very promising scientist, whom I learnt a lot from.

I thank my other colleagues and friends at the University of Toronto, especially Jaiganesh Balasundaram, Steve Chaplick, Jackie CK Cheung, Lila Fontes, Wesley George, Kaveh Ghasemloo, Daniel Lister, David Liu, Lalla Mouatadid, Arron Norwell, Joel Oren, Periklis Papakonstantinou, Robert Robere, Jocelyn Simmonds, Dhinakaran Vinayagamurthy, Justin Ward and Dustin Wehr, for not only their intellectual insights but also their humour, friendship and moral support.

My work would not have been possible without the excellent service provided by the departmental administrative staff members, especially Lisa DeCaro, Marina Haloulos, Vinita Krishnan and Wah-Ming Wong.

I thank the Department of Computer Science, the Ontario Graduate Scholarships (2008, 2009 and 2010), and the Alfred B. Lehman Graduate Scholarship (2012 and 2013) for providing financial support for my study at the University of Toronto. I also thank the [European Community's] Seventh Framework Programme [FP7/2007-2013] under grant agreement n^{o} 238381 for their generous funding during the Special Semester in Logic and Complexity 2011 in Prague.

Contents

Chapter 1

Introduction

The field of proof complexity studies different complexity measures of proofs. In the context of propositional logic, we would like to understand whether there exists a proof system in which every tautology has a short proof, i.e., the length of the proof is bounded by some polynomial in the length of the formula. With the right definition of proof systems [17], this question is equivalent to asking if NP is closed under complementation. In the context of first-order logic, we study what theorems can or cannot be proven in a given first-order theory when the scheme of induction is restricted to predicates that belong to a particular computational complexity class. First-order theories mentioned in the latter context are called theories of bounded arithmetic [10, 33, 14]. These two approaches of proof complexity are closely related since propositional proof systems can be seen as nonuniform versions of bounded arithmetic theories.

My work has been mainly inspired by a recent research program proposed Cook and Nguyen called *bounded reverse mathematics* [49, 14], which studies questions of the type:

Given a finite combinatorial theorem,
what is the weakest bounded arithmetic theory required to prove it?

Bounded reverse mathematics bears some similarity to the *reverse mathematics* research program proposed by Friedman and Simpson (cf. [57]); however reverse mathematics considers theories that can define all primitive recursive functions, while here we focus on functions contained in the polynomial hierarchy.

1.1 The complexity class CC and its two-sorted theory

Comparator networks were originally introduced as a method of sorting numbers (as in Batcher's even-odd merge sort [7]), but they are still interesting when the numbers are restricted to the Boolean values $\{0, 1\}$ (in fact, a sorting network made from comparators is valid if and only if it works on Boolean inputs). A comparator gate has two inputs p, q and two outputs p', q', where $p' = \min(p, q)$ and $q' = \max(p, q)$. In the Boolean case (which is the one we consider) $p' = p \wedge q$ and $q' = p \vee q$. A comparator circuit (i.e. network) is presented as a set of m horizontal lines in which the m inputs are presented at the left ends of the lines and the m outputs are presented at the right ends of the lines, and in between there is a sequence of comparator gates, each represented as a vertical arrow connecting

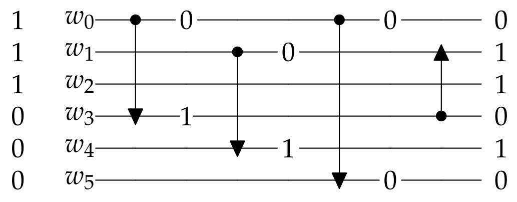


Figure 1.1

some wire w_i with some wire w_j as shown in Figure 1.1. These arrows divide each wire into segments, each of which gets a Boolean value. The values of wires w_i and w_j after the arrow are the comparator outputs of the values of wires w_i and w_j right before the arrow, with the tip of the arrow representing the maximum.

The comparator circuit value problem (Ccv) is: given a comparator circuit with specified Boolean inputs, determine the output value of a designated wire. Subramanian [40, 61] defined CC as the class of problems log-space reducible to the comparator circuit value problem.

The class CC has several disparate complete problems. As shown in [40, 61] both the lexicographical first maximal matching problem (Lfmm) and the stable marriage problem (Sm) are complete for CC under log-space reductions. The Sm problem introduced by Gale and Shapley in 1962 [20] is especially interesting: Given n men and n women, each with a complete ranking according to preference of all n members of the opposite sex, find a complete matching of the men and women such that there are no two people of opposite sex who would both rather have each other than their current partners. Gale and Shapley proved that such a 'stable' matching always exists, although it may not be unique. Subramanian [61] showed that Sm treated as a search problem (i.e. find any stable marriage) is complete for CC under log-space reducibility. Other natural and interesting complete problems include: the stable roommate problem [61], the telephone connection problem [54], the problem of predicting internal diffusion-limited aggregation clusters from theoretical physics [43], and the decision version of the hierarchical clustering problem [25].

Since Ccv is a special case of the monotone circuit value problem, CC is contained in P. A result in [40] (attributed to Feder) also shows that $\mathsf{NL} \subseteq \mathsf{CC}$. It is worth noting that CC is among a very few classes between NL and P that are not known to be contained in NC. It is conjectured that CC is incomparable with NC. For example, the CC-complete problem Lfmm seems inherently sequential, and thus is not believed to be contained in NC. On the other hand, computing integer matrix powering is in the function class of NC, but is not believed to be in the function class of CC due to the fanout restriction of comparator circuits. The reader is referred to the work by Cook, Filmus and Lê [16], which provided evidence for the conjecture by giving oracle settings in which relativized CC and relativized NC are incomparable. We will not discuss these oracle separation results in this thesis.

Let SucCC be the class of problems p-reducible to succinct CC (where a description of an exponential size comparator circuit is given using linear size Boolean circuits). It is easy to show that SucCC lies between PSPACE and EXPTIME, but we we are unable to show that it is equal to either class[1]. We are not aware of other complexity classes which appear to lie properly between PSPACE and EXPTIME.

In Chapter 3, we will study the complexity-theoretic questions related to the complexity CC from the proof-complexity theoretic point of view. We want to define a two-sorted theory that characterizes the

[1]Thanks to Scott Aaronson for pointing this out.

complexity class CC using the method developed by [14, Chapter 9]. In general this method associates a theory *VC* with a suitable complexity class C in such a way that a function is in the function class FC associated with C if and only if it is provably total in *VC*. (A string-valued function is in FC iff it is polynomially bounded and its bit-graph is in C.) However to apply this method, we need to show that CC is the same as the class of problems AC^0 'circuit' reducible to Ccv (i.e. computed by uniform AC^0 circuits with CCV gates). Note that AC^0 'circuit' reducibility is called simply AC^0 reducibility in [14]; in this thesis we will use "AC^0 reducibility" as an abbreviation for AC^0 'circuit' reducibility.

We will start out by defining CC using the even weaker AC^0 *many-one* reducibility. We will show in Section 3.4 that CC is closed under AC^0 reductions, and in Section 3.5 that CC is closed under log-space many-one reductions, which implies that these three types of reducibility define the same complexity class CC. Our key technical tool is an elegant construction of universal comparator circuits due to Yuval Filmus. Using universal comparator circuits, in Section 3.4 we also characterize CC as the class of problems computed by AC^0-uniform polynomial-size families of comparator circuits supplied with copies of the input and its negation, which tells us that CC can be characterized in terms of uniform circuit families, as in the definitions of the complexity classes NC^k and AC^k.

Chapter 3 contributes to the complexity theory of CC by sharpening these early results and simplifying their proofs. For example we prove that the three problems Ccv, Lfmm, and Sm are inter-reducible under AC^0 many-one reductions as opposed to log-space many-one reductions. Also we introduce a three-valued logic version of Ccv to facilitate its reduction to Sm.

Chapter 3 contributes to the proof complexity of CC by introducing a two-sorted formal theory *VCC* which captures the class CC and which can formalize the proofs of the above results. Analogous to the complexity-theoretic relationship $\mathsf{NL} \subseteq \mathsf{CC} \subseteq \mathsf{P}$, we will also show that $VNL \subseteq VCC \subseteq VP$. To show the containment $VNL \subseteq VCC$, we will give a simplified construction of comparator circuits that simulates the depth-first search algorithm on directed acyclic graphs. However it is nontrivial to talk about this depth-first search algorithm directly since it would require *VNL* reasoning. It turns out we can show that $VTC^0 \subseteq VNC^1 \subseteq VCC$ and use the counting ability of VTC^0 to analyze the computation of the comparator circuits in the above construction

1.2 Formalizing probabilistic reasoning

The second theme of my thesis has to do with formalizing probabilistic proofs in bounded arithmetic. One reason of choosing this research direction is due to my fascination of the unusual effectiveness of the probabilistic method in theoretical computer science (cf. [3, 46, 42]). But a more important reason is that techniques for formalizing probabilistic proofs have not been explored or understood as much in bounded arithmetic.

It is worth noting that we cannot hope to find theories that exactly capture probabilistic complexity classes such as ZPP, RP and BPP because these are 'semantic' classes which we suppose are not recursively enumerable (cf. [62]). Nevertheless there has been significant progress toward developing tools in weak theories that might be used to describe some of the algorithms in these classes.

The difficulty of formalizing the probabilistic proofs can be summarized as follows. Since the notion of probability can be essentially reduced to the cardinality ratio of definable sets, it seems natural to require the ability to count definable-sets that are exponentially large. This is problematic since Toda's theorem [63] implies that if exact counting of polynomial-time definable sets is expressible by a bounded formula of bounded arithmetic, then the polynomial hierarchy collapses.

Paris, Wilkie and Woods [52] and later Pudlak [53] observed that in many cases exact counting is not needed to formalize probabilistic arguments; in other words, we can utilize *approximate counting* by applying variants of the weak pigeonhole principle. It seems unlikely that any of these variants can be proven in the theories for polynomial-time reasoning, but they can be proven in Buss's theory S_2 or T_2 for the polynomial hierarchy. The first connection between the weak pigeonhole principle and randomized algorithms was noticed by Wilkie (cf. [33]), who showed that randomized polynomial-time functions witness Σ_1^B-consequences of $V^1 + \mathsf{sWPHP}(\mathcal{L}_{\mathsf{FP}})$, where V^1 is the two-sorted version of Buss's theory S_2^1 for polynomial-time reasoning, and $\mathsf{sWPHP}(\mathcal{L}_{\mathsf{FP}})$ denotes the surjective weak pigeonhole principle for all VPV functions (i.e. polynomial-time functions).

Building on these early results, Jeřábek [28] showed that we can "compare" the sizes of two bounded P/poly definable sets within VPV by constructing a surjective mapping from one set onto another. Using this method, Jeřábek developed tools for describing algorithms in ZPP and RP. He also showed in [28, 29] that the theory $VPV + \mathsf{sWPHP}(\mathcal{L}_{\mathsf{FP}})$ is powerful enough to formalize proofs of very sophisticated derandomization results, e.g. the Nisan-Wigderson theorem [50] and the Impagliazzo-Wigderson theorem [27]. (Note that Jeřábek actually used the single-sorted theory $PV_1 + \mathsf{sWPHP}(PV)$, but these two theories are isomorphic.)

In [30], Jeřábek developed an even more systematic approach by showing that for any bounded P/poly definable set, there exists a suitable pair of surjective "counting functions" in a suitable conservative extension of $VPV + \mathsf{sWPHP}(\mathcal{L}_{\mathsf{FP}})$ which can approximate the cardinality of the set up to a polynomially small error. From this and other results he argued convincingly that $VPV + \mathsf{sWPHP}(\mathcal{L}_{\mathsf{FP}})$ is the "right" theory for reasoning about probabilistic polynomial-time algorithms. However so far no one has used his framework for feasible reasoning about specific interesting randomized algorithms in classes such as RP and RNC^2.

1.2.1 Formalizing randomized matching algorithms

In Chapter 4 we analyze in VPV two randomized algorithms using Jeřábek's framework. The first one is the RNC^2 algorithm for determining whether a bipartite graph has a perfect matching, based on the Schwartz-Zippel Lemma [55, 65] for polynomial identity testing applied to the Edmonds polynomial [18] associated with the graph. The second algorithm, due to Mulmuley, Vazirani and Vazirani [47], is in the function class associated with RNC^2, and uses the Isolating Lemma to find such a perfect matching when it exists. Proving correctness of these algorithms involves proving that the probability of error is bounded above by $1/2$. We formulate this assertion in a way suggested by Jeřábek's framework (see Definition 57). This involves defining polynomial-time functions from $\{0,1\}^n$ onto $\{0,1\} \times \mathcal{B}_n$, where $\mathcal{B}_n$ is the set of "bad" random bit strings of length n which cause an error in the computation. We then show that VPV proves that the function is a surjection.

Our proofs are carried out in the theory VPV for polynomial-time reasoning, without the surjective weak pigeonhole principle $\mathsf{sWPHP}(\mathcal{L}_{\mathsf{FP}})$. Jeřábek used the $\mathsf{sWPHP}(\mathcal{L}_{\mathsf{FP}})$ principle to prove theorems justifying the above definition of error probability, but we do not need it to apply his definition in this chapter.

Many proofs concerning determinants are based on the *Lagrange expansion* (also known as the

Leibniz formula)

$$\mathsf{Det}(A) = \sum_{\sigma \in S_n} \mathsf{sgn}(\sigma) \prod_{i=1}^{n} A(i, \sigma(i))$$

where the sum is over exponentially many terms. Since our proofs in *VPV* can only use polynomial-time concepts, we cannot formalize such proofs, and must provide alternative proofs.

In the same vein, the standard proof of the Schwartz-Zippel Lemma assumes that a multivariate polynomial given by an arithmetic circuit can be expanded to a sum of monomials. But this sum in general has exponentially many terms so again we cannot directly formalize this proof in *VPV*. Thus we have to provide a new proof for a special case Schwartz-Zippel Lemma that is sufficient for testing if a bipartite graph has a perfect matching.

1.2.2 Formalizing Moser's constructive proof of the Lovász Local Lemma

The Lovász Local Lemma is a powerful combinatorial result used in many applications to show that a particular event happens with positive probability. It is well-known that if a large number of independent events each happen with positive probability, then there is positive (possibly exponentially small) probability that they all happen at the same time. The Lovász Local Lemma, first proved by Erdős and Lovász [19], extends this result to the case when the events only have some limited dependencies. We will be particularly interested in the application of this lemma to the k-SAT problem. Then by this lemma, it follows that if every clause of a k-CNF formula F shares a variable with at most $2^k/\mathrm{e}$ other clauses (e is the base of the natural logarithm), then F is satisfiable. However the Lovász Local Lemma is non-constructive and does not provide an efficient algorithm to find a satisfying assignment for such a formula. In the breakthrough work [8], Beck demonstrated that if every clause shares a variable with at most $O(2^{k/48})$ other clauses, then we can find a satisfying assignment in polynomial time. Alon [2] simplified and randomized Beck's algorithm and improved the bound to $O(2^{k/8})$. Srinivasan presented in [59] a variant that achieves a bound of essentially $O(2^{k/4})$. In [44], Moser gave an elegant constructive proof of the Lovász Local Lemma for k-SAT, which not only is simpler than the previous approaches, but also achieves a better bound of $2^k/8$. Moser's proof technique was later generalized by Moser and Tardos to give a constructive proof for a general (not necessarily symmetric) version of the Lovász Local Lemma [45], which they claimed to be sufficient for almost all known applications of the lemma. As a corollary, their constructive proof gives a randomized polynomial-time algorithm for k-SAT, where every clause shares a variable with up to $2^k/\mathrm{e}$ other clauses.

It is worth emphasizing that the proofs given by Alon, Srinivasan, Moser, and Moser-Tardos are constructive in the sense that *randomized* polynomial-time algorithms are provided for finding a satisfying assignment. This notion of "randomized constructivity" has recently been elaborated by Gasarch and Haeupler [21].

In Chapter 5, using Moser's technique in [44], we show that if every clause of a k-CNF formula F shares a variable with at most $2^k/8$ other clauses, then $VPV + \mathsf{sWPHP}(\mathcal{L}_{\mathsf{FP}})$ proves the existence of a satisfying assignment for F. By Wilkie's Witnessing Theorem, this implies that there exists randomized polynomial-time algorithm that outputs a satisfying assignment for such a formula F with probability at least $1/2$. The work of this chapter is my first attempt to characterize the class of Moser's "randomized constructive" proofs, which can be formalized in $VPV + \mathsf{sWPHP}(\mathcal{L}_{\mathsf{FP}})$.

1.2.3 Formalizing the proof of the Goldreich-Levin theorem

In Chapter 6 we will analyze the Goldreich-Levin theorem, a fundamental theorem in cryptography stating that every one-way permutation F can be used to construct another one-way permutation F' that has a hardcore predicate. The Goldreich-Levin theorem is an important building block in modern cryptography. For example, together with Yao's theorem, it is not hard to use a hybrid argument to construct pseudorandom generators from one-way permutations. Recall that Yao's theorem gives an equivalence between the indistinguishability of a pseudorandom generator and the unpredictability of the next bit given the earlier bits of a sequence.

Although the proof of the Goldreich-Levin theorem and many other theorems in cryptography are constructive in nature, we often need some nontrivial probabilistic tools, e.g. Markov's inequality, Chebyshev's inequality, linearity of expectation, or linearity of variance for pairwise-independent random variables. None of these concepts was formalized in Jeřábek's work [28, 29, 30] because he did not define the notion of expectation $\mathbb{E}[\bullet]$ in probability. Thus it was not clear how to formalize the proof of the Goldreich-Levin theorem in Jeřábek's approximate counting framework.

We will show in this chapter how to extend Jeřábek's approximate counting framework in [30] so that we can talk about basic concepts of finite probability theory more easily. The starting point of our work was an observation due to Jeřábek [28]: by working in a suitable conservative extension $HARD^A$ of $VPV + \mathsf{sWPHP}(\mathcal{L}_{\mathsf{FP}})$, we can define an integer valued function $\mathsf{Size}(C, n, \epsilon)$, where $C : \{0,1\}^n \to \{0,1\}$ is a circuit and $\epsilon = 1/\mathsf{poly}(n)$, that gives the approximate size of the set

$$\{X \in \{0,1\}^n \mid C(X) = 1\}$$

within error at most $\epsilon \cdot 2^n$. The basic idea is that the theory $HARD^A$ contains oracle function symbols for appropriate hard-in-average functions to construct the Nisan-Wigderson generators, which can then be used to generate the random samples to approximate the probability

$$\mathbb{P}_{X \in \{0,1\}^n}[C(X) = 1]$$

within any error $\epsilon = 1/\mathsf{poly}(n)$. From there, we can easily define the notion *approximate probability* of the event $C(X) = 1$ as

$$\mathbb{P}^{\epsilon}_{X \in \{0,1\}^n}[C(X)] = \frac{\mathsf{Size}(C, n, \epsilon)}{2^n}.$$

Informally, $\mathbb{P}^{\epsilon}_{X \in \{0,1\}^n}[C(X)]$ approximates the usual probability notion $\mathbb{P}_{X \in \{0,1\}^n}[C(X)]$ within ϵ-error since it follows from the definition of approximate counting that

$$\mathbb{P}_{X \in \{0,1\}^n}[C(X)] - \epsilon \leq \mathbb{P}^{\epsilon}_{X \in \{0,1\}^n}[C(X)] \leq \mathbb{P}_{X \in \{0,1\}^n}[C(X)] + \epsilon.$$

Next we define a suitable notion of approximate expectation using approximate probability. From these definitions we are able to formalize standard properties of approximate probability and approximate expectation including suitable versions of Markov's inequality, Chebyshev's inequality, linearity of expectation, and linearity of variance for pairwise-independent random variables.

Note that when computing the function $\mathsf{Size}(C, n, \epsilon)$ that approximates the size of of the set

$$\{X \in \{0,1\}^n \mid C(X) = 1\},$$

Jeřábek's method chooses a unique Nisan-Wigderson generator based on parameters n, the approximation error $\epsilon = 1/\mathsf{poly}(n)$ and the size of circuit C. We make a simple but useful observation (see Corollary 82 for the detailed statement) that when ϵ is sufficiently small, the choice of a Nisan-Wigderson generator in Jeřábek's method only depends on n and ϵ. Thus if we let $\mathcal{X}_1, \ldots, \mathcal{X}_k$ be sets defined as

$$\mathcal{X}_i = \{X \in \{0,1\}^n \mid C_i(X) = 1\}$$

for some circuits C_i, then from this observation we know that when ϵ is sufficiently small, the same Nisan-Wigderson will be used to approximate the sizes of $\mathcal{X}_1, \ldots, \mathcal{X}_k$ for a given ϵ.

An example application of this observation is the following. Assume that

$$C_k(X) = C_1(X) \vee \ldots \vee C_{k-1}(X).$$

Then by this observation we can show that for sufficiently small $\epsilon = 1/\mathsf{poly}(n)$, we get the following version of "union-bound"

$$\mathbb{P}^{\epsilon}_{X \in \{0,1\}^n}[C_k(X) = 1] \leq \mathbb{P}^{\epsilon}_{X \in \{0,1\}^n}[C_1(X) = 1] + \ldots + \mathbb{P}^{\epsilon}_{X \in \{0,1\}^n}[C_{k-1}(X) = 1].$$

Without this careful choice of ϵ, because of the nature of approximate counting, we can only guarantee that

$$\mathbb{P}^{\epsilon}_{X \in \{0,1\}^n}[C_k(X) = 1] \leq \mathbb{P}^{\epsilon}_{X \in \{0,1\}^n}[C_1(X) = 1] + \ldots + \mathbb{P}^{\epsilon}_{X \in \{0,1\}^n}[C_{k-1}(X) = 1] + \delta,$$

where the error δ depends on ϵ and k. We use Corollary 82 extensively to reduce approximation errors and simplify many proofs when formalizing Markov's inequality, Chebyshev's inequality, linearity of expectation, and linearity of variance for pairwise-independent random variables.

Finally we show that these properties of approximate probability, approximate expectation and approximate variance are sufficient for proving the Goldreich-Levin theorem in the conservative extension $HARD^A$ of $VPV + \mathsf{sWPHP}(\mathcal{L}_{\mathsf{FP}})$.

1.3 Organization of the thesis

In Chapter 2 we cover the necessary background on bounded arithmetic and notation needed to understand this thesis.

Chapter 3 is dedicated to the complexity class CC, its two-sorted theory VCC, and our analysis of several CC-complete problems in the theory VCC. This chapter is based on as the joint work with Cook and Ye [38] and a portion of the joint work with Cook and Filmus [16].

In Chapter 4 we formalize in the theory VPV two RNC^2 randomized algorithms. The first algorithm is for testing if a bipartite graph has a perfect matching, and and the second is for finding a perfect matching in a bipartite graph. The content of this chapter is based on the joint work with Cook [36, 37].

In Chapter 5 we formalize Moser's constructive proof of the Lovász Local Lemma for k-SAT in the theory $VPV + \mathsf{sWPHP}(\mathcal{L}_{\mathsf{FP}})$.

In Chapter 6 we show how to extend Jeřábek's approximate counting framework [30] to formalize the Goldreich-Levin Theorem. This chapter is based on unpublished joint work with Wesley George.

Chapter 7 contains our final remarks and discusses some open problems.

Chapter 2

Preliminaries

2.1 Background on bounded arithmetic

The area of bounded arithmetic was pioneered by Parikh [51], where he defined the theory now known as $I\Delta_0$, whose definable functions by bounded formulas are precisely those in the Linear Time Hierarchy. A milestone in this line of research was Buss's PhD thesis [10], which introduced the hierarchies of bounded theories

$$S^1_2 \subseteq T^1_2 \subseteq S^2_2 \subseteq T^2_2 \subseteq \ldots \subseteq S^i_2 \subseteq T^i_2 \subseteq \ldots$$

These theories, whose definable functions by bounded formulas are those in the polynomial hierarchy, are of central importance in the study of bounded arithmetic. In this thesis we will follow the two-sorted treatment of bounded theories by Cook and Nguyen [14]. One main advantage is that two-sorted theories are more appropriate when working with "small" complexity classes within P.

2.1.1 Two-sorted vocabularies

We use two-sorted vocabularies for our theories as described by Cook and Nguyen [14]. Two-sorted languages have variables $x, y, z, \ldots$ ranging over $\mathbb{N}$ and variables $X, Y, Z, \ldots$ ranging over finite subsets of $\mathbb{N}$, interpreted as bit strings. Two sorted vocabulary $\mathcal{L}^2_A$ includes the usual symbols $0, 1, +, \cdot, =, \leq$ for arithmetic over $\mathbb{N}$, the length function $|X|$ for strings ($|X|$ is zero if X is empty, otherwise $1 + \max(X)$), the set membership relation $\in$, and string equality $=_2$ (subscript 2 is usually omitted). We will use the notation $X(t)$ for $t \in X$, and think of $X(t)$ as the t^{th} bit in the string X.

The number terms in the base language $\mathcal{L}^2_A$ are built from the constants $0, 1$, variables $x, y, z, \ldots$ and length terms $|X|$ using $+$ and $\cdot$. The only string terms are string variables, but when we extend $\mathcal{L}^2_A$ by adding string-valued functions, other string terms will be built as usual. The atomic formulas are $t = u$, $X = Y$, $t \leq u$, $t \in X$ for any number terms t, u and string variables X, Y. Formulas are built from atomic formulas using $\wedge, \vee, \neg$ and $\exists x$, $\exists X$, $\forall x$, $\forall X$. Bounded number quantifiers are defined as usual, and bounded string quantifier $\exists X \leq t, \varphi$ stands for $\exists X(|X| \leq t \wedge \varphi)$ and $\forall X \leq t, \varphi$ stands for $\forall X(|X| \leq t \rightarrow \varphi)$, where X does not appear in term t.

The class Σ^B_0 consists of all $\mathcal{L}^2_A$-formulas with no string quantifiers and only bounded number quantifiers. The class Σ^B_1 consists of formulas of the form $\exists \vec{X} < \vec{t}\, \varphi$, where $\varphi \in \Sigma^B_0$ and the prefix of the bounded quantifiers might be empty. These classes are extended to Σ^B_i (and Π^B_i) for all $i \geq 0$, in the usual way. More generally we write $\Sigma_i^B(\mathcal{L})$ to denote the class of Σ_i^B-formulas which may have function and predicate symbols from $\mathcal{L} \cup \mathcal{L}_A^2$.

2.1.2 Two-sorted complexity classes and reductions

Two-sorted complexity classes contain relations $R(\vec{x}, \vec{X})$, where $\vec{x}$ are number arguments and $\vec{X}$ are string arguments. (In the sequel we refer to a relation $R(\vec{x}, \vec{X})$ as a *decision problem*: given $(\vec{x}, \vec{X})$ determine whether $R(\vec{x}, \vec{X})$ holds.) In defining complexity classes using machines or circuits, the number arguments are represented in unary notation and the string arguments are represented in binary. The string arguments are the main inputs, and the number arguments are auxiliary inputs that can be used to index the bits of strings. Using these input conventions, we define the two-sorted version of AC^0 to be the class of relations $R(\vec{x}, \vec{X})$ such that some alternating Turing machine accepts R in time $O(\log n)$ with a constant number of alternations. Then the descriptive complexity characterization of AC^0 gives rise to the following theorem [14, Chapter IV].

Theorem 1. *A relation $R(\vec{x}, \vec{X})$ is in AC^0 if and only if it is represented by some Σ_0^B-formula $\varphi(\vec{x}, \vec{X})$.*

Given a class of relations C, we associate a class FC of string-valued functions $F(\vec{x}, \vec{X})$ and number functions $f(\vec{x}, \vec{X})$ with C as follows. We require these functions to be p-bounded, i.e., $|F(\vec{x}, \vec{X})|$ and $f(\vec{x}, \vec{X})$ are bounded by a polynomial in $\vec{x}$ and $|\vec{X}|$. Then we define FC to consist of all p-bounded number functions whose graphs are in C and all p-bounded string functions whose bit graphs are in C. (Here the *bit graph* of $F(\vec{x}, \vec{X})$ is the relation $B_F(i, \vec{x}, \vec{X})$ which holds iff the ith bit of $F(\vec{x}, \vec{X})$ is 1.)

Most of the computational problems we consider here can be nicely expressed as decision problems (i.e. relations), but the stable marriage problem SM is an exception, because in general a given instance has more than one solution (i.e. there is more than one stable marriage). Thus SM is properly described as a search problem. Following [14, Section VIII.5] we define a two-sorted *search problem Q_R* to be a multivalued function with graph $R(\vec{x}, \vec{X}, Z)$, so

$$Q_R(\vec{x}, \vec{X}) = \{Z \mid R(\vec{x}, \vec{X}, Z)\} \tag{2.1}$$

The search problem is *total* if the set $Q_R(\vec{x}, \vec{X})$ is non-empty for all $\vec{x}, \vec{X}$. The search problem is a *function problem* if $|Q_R(\vec{x}, \vec{X})| = 1$ for all $\vec{x}, \vec{X}$. A function $F(\vec{x}, \vec{X})$ *solves* Q_R if

$$F(\vec{x}, \vec{X}) \in Q_R(\vec{x}, \vec{X})$$

for all $\vec{x}, \vec{X}$. Here we will be concerned only with total search problems.

Definition 2. Let C be a complexity class. A relation $R_1(\vec{x}, \vec{X})$ is C many-one reducible to a relation $R_2(\vec{y}, \vec{Y})$ (written $R_1 \leq_{\mathsf{m}}^{\mathsf{C}} R_2$) if there are functions $\vec{f}, \vec{F}$ in FC such that

$$R_1(\vec{x}, \vec{X}) \leftrightarrow R_2(\vec{f}(\vec{x}, \vec{X}), \vec{F}(\vec{x}, \vec{X})).$$

A search problem $Q_{R_1}(\vec{x}, \vec{X})$ is C many-one reducible to a search problem $Q_{R_2}(\vec{y}, \vec{Y})$ if there are functions $G, \vec{f}, \vec{F}$ in FC such that

$$G(\vec{x}, \vec{X}, Z) \in Q_{R_1}(\vec{x}, \vec{X}) \text{ for all } Z \in Q_{R_2}(\vec{f}(\vec{x}, \vec{X}), \vec{F}(\vec{x}, \vec{X})).$$

Here we are mainly interested in the cases that C is either AC^0 or L (log space). We also need a generalization of AC^0 many-one reducibility called simply AC^0 reducibility in [14, Definition IX.1.1]. Roughly speaking a function or relation is AC^0-reducible to a collection $\mathcal{L}$ of functions and relations if it can be computed by a uniform polynomial size constant depth family of circuits which have unbounded fan-in gates computing functions and relations from $\mathcal{L}$ (i.e. 'oracle gates'), in addition to Boolean gates. Formally:

Definition 3. A string function F is AC^0-reducible to a collection $\mathcal{L}$ of relations and functions (written $F \leq^{\mathsf{AC}^0} \mathcal{L}$) if there is a some constant d and a sequence of string functions $F_1, \ldots, F_d = F$ such that each F_i is p-bounded and its bit graph is represented by a $\Sigma_0^B(\mathcal{L}, F_1, \ldots, F_{i-1})$-formula.

A number function f is AC^0-reducible to $\mathcal{L}$ if $f = |F|$ for some string function F which is AC^0-reducible to $\mathcal{L}$. A relation R is AC^0-reducible to $\mathcal{L}$ if its characteristic function is AC^0-reducible to $\mathcal{L}$.

We note that standard small complexity classes including AC^0, TC^0, NC^1, NL and P (as well as their corresponding function classes) are closed under AC^0 reductions.

2.1.3 Two-sorted theories

The theory V^0 for AC^0 is the basis for developing theories for small complexity classes within P in [14]. V^0 has the vocabulary $\mathcal{L}_A^2$ and is axiomatized by the set of *2-BASIC* axioms as given in Figure 2.1, which express basic properties of symbols in $\mathcal{L}_A^2$, together with the *comprehension* axiom schema

$$\Sigma_0^B\text{-}COMP: \qquad \exists X \leq y \forall z < y\big(X(z) \leftrightarrow \varphi(z)\big),$$

where $\varphi \in \Sigma_0^B(\mathcal{L}_A^2)$ and X does not occur free in φ. Note that the axioms B1–B12 of *2-BASIC* are based on the *1-BASIC* axioms of theory $I\Delta_0$; the axioms L1 and L2 characterize $|X|$.

Although V^0 has no explicit induction axiom, nevertheless, using Σ_0^B-*COMP* and the fact that $|X|$ produces the maximum element of the finite set X, the following schemes are provable in V^0 for every formula $\varphi \in \Sigma_0^B(\mathcal{L}_A^2)$

$$\begin{aligned}
\Sigma_0^B\text{-}IND: &\qquad \big[\varphi(0) \wedge \forall x\big(\varphi(x) \rightarrow \varphi(x+1)\big)\big] \rightarrow \forall x \varphi(x),\\
\Sigma_0^B\text{-}MIN: &\qquad \varphi(y) \rightarrow \exists x\big(\varphi(x) \wedge \neg\exists z < x\, \varphi(z)\big),\\
\Sigma_0^B\text{-}MAX: &\qquad \varphi(0) \rightarrow \exists x \leq y\big[\varphi(x) \wedge \neg\exists z \leq y\big(z > x \wedge \varphi(z)\big)\big].
\end{aligned}$$

In [14, Chapter V], it was shown that V^0 is finitely axiomatizable and a p-bounded function is in FAC^0 iff it is provably total in V^0. A universally-axiomatized conservative extension $\overline{V^0}$ of V^0 was also obtained by introducing function symbols and their defining axioms for all FAC^0 functions.

In general, we say that a string function $F(\vec{x}, \vec{X})$ is Σ_1^B-definable (or provably total) in a two-sorted theory T if there is a Σ_1^B formula $\varphi(\vec{x}, \vec{X}, Y)$ representing the graph $Y = F(\vec{x}, \vec{X})$ of F such that

$$T \vdash \forall \vec{x} \forall \vec{X} \exists! Y \varphi(\vec{x}, \vec{X}, Y).$$

Similarly for a number function $f(\vec{x}, \vec{X})$.

In [14, Chapter IX], Cook and Nguyen develop a general method for associating a theory VC with certain complexity classes $\mathsf{C} \subseteq \mathsf{P}$, where VC extends V^0 with an additional axiom asserting the existence

B1. $x + 1 \neq 0$	**B8.** $x \leq x + y$
B2. $x + 1 = y + 1 \rightarrow x = y$	**B9.** $0 \leq x$
B3. $x + 0 = x$	**B10.** $x \leq y \vee y \leq x$
B4. $x + (y + 1) = (x + y) + 1$	**B11.** $x \leq y \leftrightarrow x < y + 1$
B5. $x \cdot 0 = 0$	**B12.** $x \neq 0 \rightarrow \exists y \leq x\, (y + 1 = x)$
B6. $x \cdot (y + 1) = (x \cdot y) + x$	**L1.** $X(y) \rightarrow y < \lvert X \rvert$
B7. $(x \leq y \wedge y \leq x) \rightarrow x = y$	**L2.** $y + 1 = \lvert X \rvert \rightarrow X(y)$

$$\textbf{SE.}\ \Big(|X| = |Y| \wedge \forall i < |X| \big(X(i) = Y(i)\big)\Big) \rightarrow X = Y$$

Figure 2.1: The *2-BASIC* axioms

of a solution to a complete problem for C. In order for this method to work, the class C must be closed under AC^0-reducibility (Definition 3). The method shows how to define a universal conservative extension $\overline{VC}$ of VC, where $\overline{VC}$ has string function symbols for precisely the string functions of FC, and terms for precisely the number functions of FC. Further, $\overline{VC}$ proves the $\Sigma_0^B(\mathcal{L})$-*IND* and $\Sigma_0^B(\mathcal{L})$-*MIN* schemes, where $\mathcal{L}$ is the vocabulary of $\overline{VC}$. It follows from the Herbrand Theorem that the Σ_1^B-definable functions of both VC and $\overline{VC}$ are precisely those in FC.

Using this framework Cook and Nguyen define specific theories for several complexity classes and give examples of theorems formalizable in each theory. The theories of interest to us in this thesis are VTC^0, VNC^1, VNL and VP for the complexity classes TC^0, NC^1, NL and P respectively. All of these theories have vocabulary $\mathcal{L}_A^2$. Let $\langle x, y\rangle$ denote the *pairing function*, which is the $\mathcal{L}_A^2$ term $(x + y)(x + y + 1) + 2y$. The theory VTC^0 is axiomatized by the axioms of V^0 and the axiom:

$$NUMONES: \qquad \exists Z \leq 1 + \langle n, n\rangle, \delta_{\mathsf{NUM}}(n, X, Z), \tag{2.2}$$

where the formula $\delta_{\mathsf{NUM}}(n, X, Z)$ asserts that Z is a matrix consisting of n rows such that for every $y \leq n$, the y^{th} row of Z encodes the number of ones in the prefix of length y of the binary string X. Thus, the n^{th} row of Z essentially "counts" the number of ones in X. Because of this counting ability, VTC^0 can prove *the pigeonhole principle* $\mathsf{PHP}(n, F)$ saying that if F maps a set of $n + 1$ elements to a set of n elements, then F is not an injection.

The theory VNC^1 is axiomatized by the axioms of V^0 and the axiom:

$$MFV: \qquad \exists Y \leq 2n + 1, \delta_{\mathsf{MFV}}(n, F, I, Y), \tag{2.3}$$

where F encodes a monotone Boolean formula presented as a binary tree with n distinct variables as leaves, and I encodes the Boolean values of its variables, and the formula $\delta_{\mathsf{MFV}}(n, G, I, Y)$ holds iff Y correctly encodes the values of all nodes in the tree, including the value of the root, which is the value of the formula with input I. Recall that the *monotone Boolean formula value* problem is complete for NC^1 [11, 6].

The theory VP is axiomatized by the axioms of V^0 and the axiom MCV, which is defined very similarly to MFV, but the *monotone circuit value* problem is used instead.

The theory *VNL* is axiomatized by the axioms of V^0 and the axiom:

$$CONN: \qquad \exists U \leq \langle n,n \rangle + 1, \delta_{\mathsf{CONN}}(n,E,U), \tag{2.4}$$

where E encodes the edge relation of a directed graph G with n vertices $v_0, \dots, v_{n-1}$, and the formula $\delta_{\mathsf{CONN}}(n,E,U)$ is intended to mean that U is a matrix of n rows, where each row has a Boolean value for each vertex in G, and $U(d,i)$ holds iff vertex v_i has distance at most d from v_0. More directly, the formula $\delta_{\mathsf{CONN}}(n,E,U)$ asserts

$$\begin{array}{c} U(0,i) \text{ holds iff } i=0, \text{ and } U(d+1,i) \text{ holds iff either } U(d,i) \text{ holds or} \\ \text{there is } j \text{ such that } U(d,j) \text{ holds and there is an edge in } G \text{ from } v_j \text{ to } v_i. \end{array} \tag{2.5}$$

Since we need some basic linear algebra in this thesis, we are interested in the two-sorted theory *V#L* [13]. Recall that #L is usually defined as the class of functions F such that for some nondeterministic logspace Turing machine M, $F(X)$ is the number of accepting computations of M on input X. Since counting the number of accepting paths of nondeterministic logspace is AC^0-equivalent to matrix powering, *V#L* is defined to be the extension of the base theory V^0 with an additional axiom stating the existence of powers A^k for every matrix A over $\mathbb{Z}$. The closure of #L under AC^0 reductions is called DET. It turns out that computing the determinant of integer matrices is complete for DET under AC^0 reductions since Berkowitz's algorithm [9] can be used to reduce the determinant to matrix powering. Moreover, *V#L* proves that the totality of the function Det, which computes the determinant of integer matrices based on Berkowitz's algorithm. It is an open question whether the theory *V#L* also proves the *cofactor expansion formula* and other basic properties of determinants. However from results in [58] it follows that *V#L* proves that the usual properties of determinants follow from the Cayley-Hamilton Theorem (which states that a matrix satisfies its characteristic polynomial). Using the Cook-Nguyen framework we can define the universally-axiomatized conservative extension $\overline{V\#L}$ of the two-sorted theory *V#L* [13]. As a consequence the theory $\overline{V\#L}$ proves that the function Det is in the language of $\overline{V\#L}$.

Similar to what is currently known about complexity classes, it was shown in [14, Chapter IX] that the following inclusions hold:

$$V^0 \subsetneq VTC^0 \subseteq VNC^1 \subseteq VNL \subseteq V\#L \subseteq VP. \tag{2.6}$$

2.1.4 Theories for polynomial-time reasoning

In this thesis we are particularly interested in the universal theory *VPV* for polynomial-time reasoning [14, Chapter VIII.2] since we will use it extensively in this thesis. The universal theory *VPV* is based on Cook's single-sorted theory *PV* [15], which was historically the first theory designed to capture polynomial-time reasoning. A nice property of *PV* (and *VPV*) is that their universal theorems translate into families of propositional tautologies with polynomial size proofs in any extended Frege proof system.

The vocabulary $\mathcal{L}_{\mathsf{FP}}$ of *VPV* extends that of $\overline{V^0}$ with additional symbols introduced based on Cobham's machine independent characterization of FP [12]. Let $Z^{<y}$ denote the first y bits of Z. Formally the vocabulary $\mathcal{L}_{\mathsf{FP}}$ of *VPV* is the smallest set satisfying

1. $\mathcal{L}_{\mathsf{FP}}$ contains the vocabulary of $\overline{V^0}$
2. For any two function $G(\vec{x},\vec{X})$, $H(y,\vec{x},\vec{X},\vec{Z})$ over $\mathcal{L}_{\mathsf{FP}}$ and a $\mathcal{L}^2_A$-term $t = t(y,\vec{x},\vec{X})$, if F is defined by *limited recursion* from G, H and t, i.e.,

$$F(0,\vec{x},\vec{X}) = G(\vec{x},\vec{X}),$$
$$F(y+1,\vec{x},\vec{X}) = H(y,\vec{x},\vec{X},F(y,\vec{x},\vec{X}))^{<t(y,\vec{x},\vec{X})},$$

then $F \in \mathcal{L}_{\mathsf{FP}}$.

We will often abuse the notation by letting $\mathcal{L}_{\mathsf{FP}}$ denote the set of function symbols in $\mathcal{L}_{\mathsf{FP}}$.

The theory *VPV* can then be defined to be the theory over $\mathcal{L}_{\mathsf{FP}}$ whose axioms are those of $\overline{V^0}$ together with defining axioms for every function symbols in $\mathcal{L}_{\mathsf{FP}}$. *VPV* proves the scheme $\Sigma^B_0(\mathcal{L}_{\mathsf{FP}})$-*COMP* and the following schemes

$$\begin{array}{ll} \Sigma^B_0(\mathcal{L}_{\mathsf{FP}})\text{-}\mathit{IND}: & \big(\varphi(0) \wedge \forall x\big(\varphi(x) \to \varphi(x+1)\big)\big) \to \forall x \varphi(x) \\ \Sigma^B_0(\mathcal{L}_{\mathsf{FP}})\text{-}\mathit{MIN}: & \varphi(y) \to \exists x\big(\varphi(x) \wedge \neg\exists z < x \varphi(z)\big) \\ \Sigma^B_0(\mathcal{L}_{\mathsf{FP}})\text{-}\mathit{MAX}: & \varphi(0) \to \exists x \le y\big(\varphi(x) \wedge \neg\exists z \le y\big(z > x \wedge \varphi(z)\big)\big) \end{array}$$

where φ is any $\Sigma^B_0(\mathcal{L}_{\mathsf{FP}})$-formula. It follows from Herbrand's Theorem that the provably-total functions in *VPV* are precisely the functions in $\mathcal{L}_{\mathsf{FP}}$.

Observe that *VPV* extends $\overline{V\#L}$ since matrix powering can easily be carried out in polynomial time, and thus all theorems of $\overline{V\#L}$ from [13, 58] are also theorems of *VPV*. From results in [58] (see page 44 of [29] for a correction) it follows that *VPV* proves the Cayley-Hamilton Theorem, and hence the cofactor expansion formula and other usual properties of determinants of integer matrices.

Another theory for polynomial-time reasoning is V^1 [14]. In general, the theories V^i for $\mathsf{FP}^{\Sigma^{\mathsf{P}}_i}$ reasoning is axiomatized by *2-BASIC* and the *comprehension* axiom schema

$$\Sigma^B_i\text{-}\mathit{COMP}: \qquad \exists X \le y\, \forall z < y\big(X(z) \leftrightarrow \varphi(z)\big),$$

where $\varphi \in \Sigma^B_i(\mathcal{L}^2_A)$ and X does not occur free in φ. Theories V^i are two-sorted versions of Buss's S^i_2 single-sorted theories [10], and V^i and S^i_2 are isomorphic under the notion of RSUV isomorphism [14, Section VIII.8.5].

Thus the theory V^1 is axiomatized by *2-BASIC* and the Σ^B_1-*COMP* schema. The Σ^B_1-definable functions of V^1 are FP functions, and V^1 is Σ^B_1-conservative over *VPV*. However there is evidence showing that V^1 is stronger than *VPV*. For example, the theory V^1 proves the Σ^B_1-*IND*, Σ^B_1-*MIN* and Σ^B_1-*MAX* schemes while *VPV* cannot prove these Σ^B_1 schemes, assuming the polynomial hierarchy does not collapse [34].

2.1.5 The surjective weak pigeonhole principle and Wilkie's Witnessing Theorem

The *surjective weak pigeonhole principle* for a function F, denoted by $\mathsf{sWPHP}(F)$, is the universal closure of the formula

$$(n > 0 \wedge A > 0) \to \exists Y < (n+1)A\, \forall X < nA,\ F(X) \neq Y,$$

stating that F cannot map $[0, nA)$ onto $[0,(n+1)A)$.

Remark 4. We write "$\forall X < nA$" to mean that we want to universally quantify over all binary numbers X less than the binary number $n \cdot A$. The same interpretation applies to the existential quantifier "$\exists Y < (n+1)A$" above. Note that this way of writing "string quantifiers" is incorrect according to the conventions in Cook-Nguyen two-sorted theories, but it can easily be translated to a correct one. We will use quantifiers of binary numbers bounded by other binary numbers freely in this thesis for convenience.

The surjective weak pigeonhole principle for *VPV* functions, denoted by $\mathsf{sWPHP}(\mathcal{L}_{\mathsf{FP}})$, is the schema

$$\{\mathsf{sWPHP}(F) \mid F \in \mathcal{L}_{\mathsf{FP}}\}.$$

Note that this principle is believed to be weaker than the usual surjective "strong" pigeonhole principle stating that we cannot map $[0, A)$ onto $[0, A+1)$. For example, $\mathsf{sWPHP}(\mathcal{L}_{\mathsf{FP}})$ can be proven in the theory V^3 for $\mathsf{FP}^{\Sigma_3^{\mathsf{P}}}$ reasoning (cf. [62]), but it is not known if the usual surjective pigeonhole principle for *VPV* functions can be proven within the theory $\bigcup_{i \geq 1} V^i$ for the polynomial hierarchy.

Next we will recall an important theorem due to Alex Wilkie, which gives the earliest evidence that sWPHP is the reasonable principle for witnessing probabilistic algorithms in bounded arithmetic. Although the theorem was proved by Wilkie, the first published proof can only be found in [33, Theorem 7.3.7]. An alternative and more detailed proof was later given by Thapen in [62, Theorem 3.11]. Wilkie's Witnessing Theorem states that when the sWPHP principle is added to the theory V^1, there exists a probabilistic polynomial-time function witnessing the formula $\forall X \exists Z\, \varphi(X, Z)$ for every Σ_1^B formula $A(X, Z)$.

Theorem 5 (Wilkie's Witnessing Theorem)**.** *Let $\varphi(X, Z)$ be a Σ_1^B-formula and assume that*

$$V^1 + \mathsf{sWPHP}(\mathcal{L}_{\mathsf{FP}}) \vdash \forall X \exists Z\ \varphi(X, Z).$$

Then there is a probabilistic polynomial-time algorithm which, for each input X, outputs with probability at least $1/2$ some Y satisfying the formula $\varphi(X, Y)$.

Jeřábek showed in [30] that the provably total search problems of $VPV + \mathsf{sWPHP}(\mathcal{L}_{\mathsf{FP}})$ are precisely the ones reducible to the following NP search problem: given a pair of circuits $G : \{0,1\}^n \to \{0,1\}^{2n}$ and $H : \{0,1\}^{2n} \to \{0,1\}^n$, find a $Y \in \{0,1\}^{2n}$ such that $G(H(Y)) \neq Y$.

2.2 Notation

We write the notation "$(T \vdash)$" in front of the statement of a theorem to indicate that the statement is formulated and proved within the theory T. We write the notation "(in T)" in front of a definition to indicate that the definition is introduced in the theory T.

We will often work with *bounded definable sets*, which are collections of binary strings of the form

$$\mathcal{X} = \{X \in \{0,1\}^n \mid \varphi(X)\}$$

where φ is a formula (possibly with other free variables). Bounded sets are not genuine objects in our arithmetical theories: $X \in \mathcal{X}$ is an abbreviation for $|X| \leq n \wedge \varphi(X)$. We will use uppercase calligraphic letters $\mathcal{X}, \mathcal{Y}, \mathcal{Z}, \ldots$ to represent these bounded definable sets. The sets we will encounter most often will be *defined by Boolean circuits* in the following sense: a circuit $C : \{0,1\}^n \to \{0,1\}$ defines the set $\{X \in \{0,1\}^n \mid C(X) = 1\}$. Hence we say $\mathcal{X}$ is definable by circuit $C : \{0,1\}^n \to \{0,1\}$ to indicate that $\mathcal{X} = \{X \in \{0,1\}^n \mid C(X) = 1\}$.

If F is a function of n variables, then let $F(X_1, \ldots, X_{n-1}, \bullet)$ denote the function of one variable resulting from F by fixing the first $n-1$ arguments to $X_1, \ldots, X_{n-1}$. We write $F : \mathcal{X} \twoheadrightarrow \mathcal{Y}$ to denote that F is a surjection from $\mathcal{X}$ *onto* $\mathcal{Y}$. We write $F : \mathcal{X} \hookrightarrow \mathcal{Y}$ to denote that F is an injection from $\mathcal{X}$ *into* $\mathcal{Y}$.

We use lowercase Greek letters $\alpha, \beta, \gamma, \ldots$ to denote binary rational numbers. And we write $\gamma = \beta \pm \epsilon$ to denote that $\beta - \epsilon \leq \gamma \leq \beta + \epsilon$.

Let $[\beta, \gamma)$ denote the set $\{Z \in \mathbb{Z} \mid \beta \leq Z < \gamma\}$. Let $[\beta)$ denote the set $\{Z \in \mathbb{Z} \mid 0 \leq Z < \beta\}$. We also use similar notation for unary or binary numbers. We often use the standard notation $[n]$ to denote the set $\{1, \ldots, n\}$.

Given a square matrix M, we write $M[i \mid j]$ to denote the (i,j)-*minor* of M, i.e., the square matrix formed by removing the ith row and jth column from M.

We let $\vec{x}$ denote a number sequence $\langle x_1, \ldots, x_k \rangle$ and let $\vec{X}$ denote a string sequence $\langle X_1, \ldots, X_k \rangle$. We write $\vec{x}_{k \times k}$ and $\vec{X}_{k \times k}$ to denote two-dimensional arrays $\langle x_{i,j} \mid 1 \leq i, j \leq k \rangle$ and $\langle X_{i,j} \mid 1 \leq i, j \leq k \rangle$ respectively, where the elements of these two-dimensional arrays are listed by rows. Note that $\vec{x}_{k \times k}$ and $\vec{X}_{k \times k}$ can be simply encoded as integer matrices, and thus we will use matrix notation freely on them.

We often use two-dimensional matrices to encode binary relations, e.g. the edge relation of a graph, matching, etc. In this thesis, it is more convenient for our purpose to index the entries of matrices starting from 0 instead of 1. In other words, if $X_{n \times n}$ is a two-dimensional matrix, then entries of X consist of all $X(i,j)$ for $0 \leq i, j < n$, and $X(0,0)$ is the top leftmost entry of X.

We will use the following shorthand notation to indicate that x equals the logarithm of some unary number:

$$x \in \mathit{Log} \leftrightarrow \exists m,\ 2^x = m + 1$$

(Note that the *Log* notation in our two-sorted framework corresponds to the *LogLog* notation in the first-order setup in Jeřábek's work [28, 29, 30].)

We also write $\log(x)$ to denote $\lceil \log_2(x) \rceil$, i.e. the ceiling of the base-2 logarithm of x.

We write $x = \mathsf{poly}(n)$ to mean that $x = n^k$ for some $k > 0$. Similarly we write $\epsilon = 1/\mathsf{poly}(n)$ to denote $\epsilon = \frac{1}{n^k}$ for some $k > 0$.

Let C be a circuit, then we let $|C|$ to denote the length of binary string encoding the circuit C.

2.3 Jeřábek's approximate counting framework

We will give a brief and simplified overview of Jeřábek's framework [28, 29, 30] for probabilistic reasoning within $VPV + \mathsf{sWPHP}(\mathcal{L}_{\mathsf{FP}})$. For more complete definitions and results, the reader is referred to Jeřábek's work.

Let $F(R)$ be a VPV boolean function (which may have other arguments). We think of F as defining a predicate on binary numbers R. Let $\mathcal{X}_n = \{R \in \{0,1\}^n \mid F(R) = 0\}$. Observe that bounding the probability $\mathbb{P}_{R\in\{0,1\}^n}\left[F(R)=0\right]$ from above by the ratio s/t is the same as showing that $t \cdot |\mathcal{X}_n| \leq s \cdot 2^n$. More generally, many probability inequalities can be restated as inequalities between cardinalities of sets. This is problematic since even for the case of polynomial-time definable sets, it follows from Toda's theorem [63] that we cannot express their cardinalities directly using bounded formulas (assuming that the polynomial hierarchy does not collapse). Hence we need an alternative method to compare the sizes of definable sets without exact counting.

The method proposed by Jeřábek in [28, 29, 30] is based on the following simple observation: if $\mathcal{X}$ and $\mathcal{Y}$ are definable sets and there is a function F mapping $\mathcal{X}$ onto $\mathcal{Y}$, then the cardinality of $\mathcal{Y}$ is at most the cardinality of $\mathcal{X}$. Thus instead of counting the sets $\mathcal{X}$ and $\mathcal{Y}$ directly, we can compare the sizes of $\mathcal{X}$ and $\mathcal{Y}$ by showing the existence of a surjection F, which in many cases can be easily carried out within weak theories of bounded arithmetic.

The remaining challenge is to formally verify that the definition of cardinality comparison through the use of surjections is a meaningful and well-behaved definition. The basic properties of surjections like "any set can be mapped onto itself" and "surjectivity is preserved through function compositions" roughly correspond to the usual reflexivity and transitivity of cardinality ordering, i.e., $|\mathcal{X}| \leq |\mathcal{Y}|$ and $|\mathcal{Y}| \leq |\mathcal{Z}| \to |\mathcal{X}| \leq |\mathcal{Z}|$ for all bounded definable sets $\mathcal{X}$, $\mathcal{Y}$ and $\mathcal{Z}$. However, more sophisticated properties, e.g., dichotomy $|\mathcal{X}| \leq |\mathcal{Y}| \vee |\mathcal{Y}| \leq |\mathcal{X}|$ or "uniqueness" of cardinality, turn out to be much harder to show.

As a result, Jeřábek proposed in [30] a systematic and sophisticated framework to justify his definition of size comparison. He observed that estimating the size of a set $\mathcal{X} \subseteq \{0,1\}^n$, whose characteristic function is computed by a polysize Boolean circuit, within an error $2^n/\mathsf{poly}(n)$ is the same as estimating $\mathbb{P}_{X\in\{0,1\}^n}[X \in \mathcal{X}]$ within an error $1/\mathsf{poly}(n)$, which can be solved by drawing $\mathsf{poly}(n)$ independent random samples $X \in \{0,1\}^n$ and check if $X \in \mathcal{X}$. This gives us a polynomial-time random sampling algorithm for approximating the size of $\mathcal{X}$. Since a counting argument [28] can be formalized within $VPV + \mathsf{sWPHP}(\mathcal{L}_{\mathsf{FP}})$ to show the existence of suitable average-case hard functions for constructing Nisan-Wigderson generators, this random sampling algorithm can be derandomized to show the existence of an *approximate cardinality* S of $\mathcal{X}$ for any given error $\epsilon = 1/\mathsf{poly}(n)$ in the following sense. The theory $VPV + \mathsf{sWPHP}(\mathcal{L}_{\mathsf{FP}})$ proves the existence of S and suitable surjections witnessing that

$$S - \epsilon \cdot 2^n \leq |\mathcal{X}| \leq S + \epsilon \cdot 2^n.$$

We start by reviewing the formal definitions taken from [28, 30].

Definition 6 ([30])**.** (in *VPV*) Let $f : \{0,1\}^k \to \{0,1\}$ be a truth-table of a Boolean function encoded as a string of length 2^k and $k \in Log$.

- We say that f is (worst-case) ϵ-hard, written as $\mathsf{Hard}_\epsilon(f)$, if there does not exist a circuit C of size at most $2^{\epsilon k}$ which computes f.

- We say that f is average-case ϵ-hard, written as $\mathsf{Hard}^A_\epsilon(f)$, if there does not exist a circuit C of size at most $2^{\epsilon k}$ such that

$$\left|\{u < 2^k \mid C(u) = f(u)\}\right| \geq \left(\frac{1}{2} + 2^{-\epsilon k}\right) 2^k$$

Note that in the above definition we can count the size of the set precisely since the domain of the function f is small. The following key lemma shows that the theory $VPV + \mathsf{sWPHP}(\mathcal{L}_{\mathsf{FP}})$ proves the existence of a family of average-case ϵ-hard functions.

Lemma 7 ([30]). *For every constant $0 < \epsilon < 1/3$, there exists a constant c such that $VPV + \mathsf{sWPHP}(\mathcal{L}_{\mathsf{FP}})$ proves that:*

For every $k \geq c$ and $k \in Log$, there exists average-case ϵ-hard functions $f : \{0,1\}^k \to \{0,1\}$.

We next recall the definitions of the combinatorial designs used in the Nisan-Wigderson generator construction.

Definition 8 ([30]). (in VPV) Let $k, \ell, t \in Log$ satisfy $k \leq \ell \leq t$. A $\langle k, \ell, t, m\rangle$-design is a sequence $\langle S_i\rangle_{i<m}$ of subsets $S_i \subseteq [0,t)$ such that $|S_i| = \ell$ and $|S_i \cap S_j| \leq k$ for all $i < j < m$.

Jeřábek showed that VPV proves that combinatorial designs of appropriate parameters can be produced uniformly by a VPV function.

Lemma 9 ([30]). *Let $0 < \gamma < 1$. There are constants $\delta > 0$, $c > 1$ and a function $F \in \mathcal{L}_{\mathsf{FP}}$ such that*

$$VPV \vdash F(X) \text{ is a } \left\langle \gamma\ell, \ell, c\ell, 2^{\delta\ell} \right\rangle\text{-design, where } \ell = \log(|X|+1)$$

We next state the formal definition of the Nisan-Wigderson generator that "stretches" a seed of t bits to pseudorandom string of m bits.

Definition 10 ([30]). (in VPV) Let $X \in \{0,1\}^t$ and $Y \subseteq [0,t)$, where $Y = \{s_0, \ldots, s_{\ell-1}\}$ and $s_i < s_{i+1}$ for all $i < \ell$. Then we define $X \upharpoonright Y := Z$, where $|Z| \leq \ell$ and $Z(i) = X(s_i)$ for all $i < \ell$.

If $f : \{0,1\}^\ell \to \{0,1\}$ and $S = \langle S_i\rangle_{i<m}$ is a $\langle k, \ell, t, m\rangle$-design, then the Nisan-Wigderson generator is a function $\mathsf{NW}_{f,S} : \{0,1\}^t \to \{0,1\}^m$ defined by

$$\mathsf{NW}_{f,S}(X)(i) = f(X \upharpoonright S_i)$$

We now state formally Jeřábek's theorem from [30], which plays a central role in this thesis. Recall that we say $\mathcal{X}$ is definable by circuit $C : \{0,1\}^n \to \{0,1\}$ to indicate that $\mathcal{X} = \{X \in \{0,1\}^n \mid C(X) = 1\}$.

Theorem 11 (Jeřábek's Theorem [30]). *($VPV + \mathsf{sWPHP}(\mathcal{L}_{\mathsf{FP}}) \vdash$) Let $C : \{0,1\}^n \to \{0,1\}$ be a boolean circuit, and $\epsilon = 1/\mathsf{poly}(n)$. Let the set $\mathcal{X}$ be definable by the circuit C. Then there exist $S \leq 2^n$, $v \leq \mathsf{poly}(n\epsilon^{-1}|C|)$, and circuits G_i, H_i of size $\mathsf{poly}(n\epsilon^{-1}|C|)$ such that*

$$G_0 : \left[v(S+\epsilon 2^n)\right) \twoheadrightarrow [v) \times \mathcal{X} \qquad H_0 : [v) \times \mathcal{X} \hookrightarrow \left[v(S+\epsilon 2^n)\right) \tag{2.7}$$

$$G_1 : [v) \times \left(\mathcal{X} \uplus [0, \epsilon 2^n)\right) \twoheadrightarrow [vS) \qquad H_1 : [vS) \hookrightarrow [v) \times \left(\mathcal{X} \uplus [0, \epsilon 2^n)\right) \tag{2.8}$$

and $G_i \circ H_i = id$ on their respective domains.

In other words, the theorem states that given a set $\mathcal{X}$ defined by a circuit C, the theory $VPV + \mathsf{sWPHP}(\mathcal{L}_{\mathsf{FP}})$ proves the existence of the approximate size S that is ϵ-close to the actual size of the set $\mathcal{X}$. This size S is constructed by choosing an appropriate Nisan-Wigderson generator $\mathsf{NW}_{f,S} : \{0,1\}^t \rightarrow \{0,1\}^n$ with $t = O(\log(n))$ that generates the samples needed to approximate the fraction α of elements in $\{0,1\}^n$ that also belong to $\mathcal{X}$. For this purpose, we can let

$$\alpha = \frac{\left|\{X \in \{0,1\}^t \mid C(\mathsf{NW}_{f,S}(X)) = 1\}\right|}{2^t}$$

Then we define $S := \alpha 2^n$ to be the approximate size of $\mathcal{X}$ in this theorem.

It is worth noting that Lemma 7 is the only place we need $\mathsf{sWPHP}(\mathcal{L}_{\mathsf{FP}})$ to prove the existence of average-case ϵ-hard functions. Once we have these average-case ϵ-hard functions, the rest of the proof of Theorem 11 can be proved in VPV.

Theorem 11 allows Jeřábek to show many properties, expected from cardinality comparison, that are satisfied by his method within the theory $VPV + \mathsf{sWPHP}(\mathcal{L}_{\mathsf{FP}})$ (see Lemmas 13 and 14). It is worth noting that proving the uniqueness of cardinality within some error seems to be the best we can do within bounded arithmetic, where exact counting is not available.

We need the following formal definition of approximate size comparison based on surjections between definable sets.

Definition 12 ([30])**.** (in $VPV + \mathsf{sWPHP}(\mathcal{L}_{\mathsf{FP}})$) Let $\mathcal{X}, \mathcal{Y} \subseteq \{0,1\}^n$ be definable by circuits and $0 \leq \epsilon \leq 1$. We say that the size of $\mathcal{X}$ is *approximately less than* the size of $\mathcal{Y}$ with error ϵ, denoted as

$$\mathcal{X} \precsim_\epsilon \mathcal{Y},$$

if there exists a circuit G, and $V \neq 0$, such that

$$G : [V] \times \left(\mathcal{Y} \uplus [\epsilon 2^n]\right) \twoheadrightarrow [V] \times \mathcal{X}.$$

The sets $\mathcal{X}$ and $\mathcal{Y}$ are *approximately equinumerous* with error ϵ, written

$$\mathcal{X} \approx_\epsilon \mathcal{Y},$$

if $\mathcal{X} \precsim_\epsilon \mathcal{Y}$ and $\mathcal{Y} \precsim_\epsilon \mathcal{X}$.

Note that in Definition 12 we allow V to be binary, and thus this approximate size comparison definition is more general than what we have in Theorem 11.

The following properties follow directly from the definition of $\precsim_\epsilon$ and $\approx_\epsilon$.

Lemma 13 ([30])**.** *($VPV \vdash$) Let $\mathcal{X}, \mathcal{Y}, \mathcal{X}', \mathcal{Y}', \mathcal{Z} \subseteq \{0,1\}^n$. Let $\mathcal{W}, \mathcal{W}' \subseteq \{0,1\}^m$ be sets definable by circuits. Let $0 \leq \epsilon, \delta \leq 1$.*

1. *If $\mathcal{X} \precsim_\epsilon \mathcal{Y}$ and $\epsilon \leq \delta$, then $\mathcal{X} \precsim_\delta \mathcal{Y}$.*

2. *If $\mathcal{X} \subseteq \mathcal{Y}$, then $\mathcal{X} \precsim_0 \mathcal{Y}$.*

3. *If $\mathcal{X} \precsim_\epsilon \mathcal{Y} \precsim_\delta \mathcal{Z}$, then $\mathcal{X} \precsim_{\epsilon+\delta} \mathcal{Z}$.*

4. *If $\mathcal{X} \precsim_\epsilon \mathcal{X}'$ and $\mathcal{Y} \precsim_\delta \mathcal{Y}'$, and $\mathcal{X}'$ and $\mathcal{Y}'$ are separable by a circuit, then $\mathcal{X} \cup \mathcal{Y} \precsim_{\epsilon+\delta} \mathcal{X}' \cup \mathcal{Y}'$.*

5. *If $\mathcal{X} \precsim_\epsilon \mathcal{X}'$ and $\mathcal{W} \precsim_\delta \mathcal{W}'$, then $\mathcal{X} \times \mathcal{W} \precsim_{\epsilon+\delta+\epsilon\delta} \mathcal{X}' \times \mathcal{W}'$.*

The following less elementary properties exploit consequences of approximate counting technique.

Lemma 14 ([30])**.** *($VPV + \mathsf{sWPHP}(\mathcal{L}_{\mathsf{FP}}) \vdash$) Let $\mathcal{X}, \mathcal{Y} \subseteq \{0,1\}^n$ be sets definable by circuits. Let $S, T, U \leq 2^n$, and let $0 \leq \epsilon, \delta, \eta, \gamma \leq 1$ and $\gamma = 1/\mathsf{poly}(n)$.*

1. *There exists $S \leq 2^n$ such that $\mathcal{X} \approx_\gamma [S)$.*

2. *If $[S) \precsim_\epsilon [T)$, then $S \leq T + (\epsilon + \gamma)2^n$. And if $[S) \approx_\epsilon [T)$, then $S = T \pm (\epsilon + \gamma)2^n$.*

3. *$\mathcal{X} \precsim_\gamma \mathcal{Y}$ or $\mathcal{Y} \precsim_\gamma \mathcal{X}$.*

4. *If $[S) \precsim_\epsilon \mathcal{X} \precsim_\delta [T)$, then $S \leq T + (\epsilon + \delta + \gamma)2^n$*

5. *If $\mathcal{X} \precsim_\epsilon \mathcal{Y}$, then $\{0,1\}^n \setminus \mathcal{Y} \precsim_{\epsilon+\gamma} \{0,1\}^n \setminus \mathcal{X}$.*

6. *If $\mathcal{X} \approx_\epsilon [S)$, $\mathcal{Y} \approx_\delta [T)$ and $\mathcal{X} \cap \mathcal{Y} \approx_\eta [U)$, then $\mathcal{X} \cup \mathcal{Y} \approx_{\epsilon+\delta+\eta+\gamma} [S + T - U)$.*

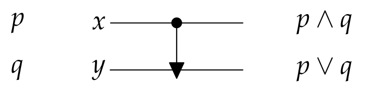
Chapter 3

The complexity class CC and its theory

3.1 Ccv and the complexity class CC

A *comparator gate* is a function $C : \{0,1\}^2 \to \{0,1\}^2$, that takes an input pair (p,q) and outputs a pair $(p \wedge q, p \vee q)$. Intuitively, the first output in the pair is the smaller bit among the two input bits p, q, and the second output is the larger bit.

We will use the graphical notation on the right to denote a comparator gate, where x and y denote the names of the wires, and the direction of the arrow denotes the direction to which we move the larger bit as shown in the picture below.



A *comparator circuit* can be defined as a directed acyclic graph consisting of: *input nodes* with in-degree zero and out-degree one, *output nodes* with in-degree one and out-degree zero, and *internal nodes* with in-degree two and out-degree two. Each internal node represents a comparator gate, with one out-edge labeled AND and the other labeled OR. If there are m input nodes then there must be m output nodes, and the circuit computes a function $f : \{0,1\}^m \to \{0,1\}^m$ in the obvious way.

Under this definition each comparator circuit can be represented by m horizontal wires that carry bit values (see Figure 1.1 on Page 3), where m is the number of input nodes. Each comparator gate is represented by a vertical arrow connecting two of the wires. The arrowhead (representing the OR of the two inputs) can be chosen at will to point up or down, but the decision affects which wire future gates connect. To see this, topologically sort the internal nodes of the graph (representing the comparator gates), and arrange the corresponding arrows in order from left to right. The left endpoints of the wires represent the m input nodes, and the gates can be placed one by one from left to right, each connecting the appropriate wires (determined by looking back to the last output gate touching the wire).

In this paper we present a comparator circuit by specifying its representation as horizontal wires connected by arrows, as explained in the previous paragraph. We use the two-sorted notation given in Section 2.1.3. We encode the circuit by a triple (m, n, X), where m is the number of wires and n is the number of gates, and X encodes a sequence of n pairs $\langle i, j\rangle$ with $i, j < m$, where each pair $\langle i, j\rangle$ encodes a comparator gate that swaps the values of the wires i and j iff the value on wire i is greater than the value of wire j. We also allow "dummy" gates of the form $\langle i, i\rangle$, which do nothing. We write $(X)^i$ to denote the i^{th} comparator gate.

The comparator circuit value problem (Ccv) is the decision problem: Given a comparator circuit, an assignment of bits to the input nodes, and a designated output wire, decide whether the circuit outputs one on that wire. The next definition formalizes this, and defines the corresponding complexity classes.

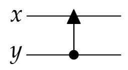
Definition 15. Ccv is the two-sorted relation $\text{Ccv}(m, n, i, X, Y)$ which holds iff $i < m$ and (m, n, X) codes a comparator circuit as above for which the output of wire i is one when the inputs are as specified by Y. We often write $\text{Ccv}(i, C, Y)$ instead of $\text{Ccv}(m, n, i, X, Y)$, where the string $C = \langle m, n, X\rangle$ (for 'circuit') codes the triple (m, n, X).

The complexity class CC is the class of decision problems that are AC^0 many-one reducible to Ccv.

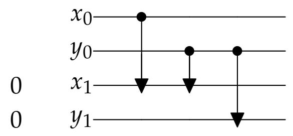
In our definition of comparator circuit each comparator gate can point in either direction, up or down (see Figure 1.1). As mentioned earlier, an equivalent circuit with the same number of wires and gates can always be constructed so that all gates point up, or all gates point down, or in fact each gate can be made to point up or down arbitrarily. Transforming a circuit to one of these forms (with the same number of gates) cannot necessarily be done by an AC^0 function, but it is not hard to show the following.

Proposition 16. *The* Ccv *problem with the restriction that all comparator gates point in the same direction is* CC-*complete.*

Proof. Suppose we have a gate with the arrow pointing upward like following:



We will build a circuit that outputs the same values as those of x and y, but all the gates will now point downward.



It is not hard to see that the wires x_1 and y_1 in this new comparator circuit will output the same values with the wires x and y respectively in the original circuit. For the general case, we can simply make copies of all wires for each layer of the comparator circuit, where each copy of a wire will be used to carry value of a wire at a single layer of the circuit. Then apply the above construction to simulate the effect of each gate. Note that additional comparator gates are also needed to forward the values of the wires from one layer to another, in the same way that we use the gate $\langle y_0, y_1\rangle$ to forward the value carried in wire y_0 to wire y_1 in the above construction. □

3.2 The stable marriage problem

An instance of the stable marriage problem (Sm) is given by a number n (specifying the number of men and the number of women), together with a preference list for each man and each woman specifying a total ordering on all people of the opposite sex. The goal of Sm is to produce a perfect matching between men and women, i.e., a bijection from the set of men to the set of women, such that the following *stability* condition is satisfied: there are no two people of the opposite sex who like each other more than their current partners. Such a stable solution always exists, but it may not be unique. Thus SM is a search problem (2.1), rather than a decision problem.

Gale and Shapley showed in their seminal work [20] that there are always a unique *man-optimal* and a unique *woman-optimal* solution, and presented a polynomial-time algorithm to find these solutions. In the man-optimal solution each man is matched with a woman whom he likes at least as well as any woman that he is matched with in any stable solution. Dually for the woman-optimal solution. Thus we define the *man-optimal stable marriage decision problem* (MOSM) as follows: given an instance of the stable marriage problem together with a designated man-woman pair, determine whether that pair is married in the man-optimal stable marriage. We define the *woman-optimal stable marriage decision problem* (WOSM) analogously.

We show here that the search version and the decision versions are computationally equivalent, and each is complete for CC with respect to the appropriate reducibility in Definition 2. Section 3.7.1 shows how to reduce the lexicographical first maximal matching problem (which is complete for CC) to the SM search problem, and Section 3.7.2 shows how to reduce both the MOSM and WOSM problems to CCV.

3.3 The new theory *VCC*

This section requires some background on two-sorted bounded arithmetic theories from the Cook-Nguyen framework that we reviewed in Section 2.1, and the notation developed in Section 2.2.

We want to define a formula $\delta_{\mathsf{CCV}}(m, n, X, Y, Z)$, where

- (m, n, X) encodes a comparator circuit with m wires and n gates as explained before Definition 15.
- $Y(i)$ encodes the input value for the i^{th} wire as a truth value, and
- Z is an $(n+1) \times m$ array, where $Z(i, j)$ is the value of wire j at layer i, where each layer is simply a sequence of values carried by all the wires right after a comparator gate.

Although X encodes a circuit with only n gates, the matrix Z actually encodes $n+1$ layers $0, 1, \ldots, n$ since we use layer 0 to encode the input values of the comparator circuit and layer n to encode the output values. The formula $\delta_{\mathsf{CCV}}(m, n, X, Y, Z)$ holds iff Z encodes the correct values of the layers computed by the comparator circuit encoded by X with input Y, and thus is defined as the following Σ_0^B-formula:

$$\forall i < m\big(Y(i) \leftrightarrow Z(0,i)\big) \wedge \forall i < n \forall x < m \forall y < m,$$

$$(X)^i = \langle x, y\rangle \rightarrow \left[\begin{array}{ll} & Z(i+1,x) \leftrightarrow \big(Z(i,x) \wedge Z(i,y)\big) \\ \wedge & Z(i+1,y) \leftrightarrow \big(Z(i,x) \vee Z(i,y)\big) \\ \wedge & \forall j < m\big[(j \neq x \wedge j \neq y) \rightarrow \big(Z(i+1,j) \leftrightarrow Z(i,j)\big)\big] \end{array}\right] \tag{3.1}$$

Definition 17. The theory *VCC* has vocabulary $\mathcal{L}_A^2$ and is axiomatized by the axioms of V^0 and the following axiom

$$AX\textsc{ccv}: \qquad \exists Z \leq \langle m, n+1 \rangle + 1,\ \delta_{\mathsf{CCV}}(m,n,X,Y,Z), \tag{3.2}$$

where $\delta_{\mathsf{CCV}}(m,n,X,Y,Z)$ is defined as in (3.1).

Define the string function $F_{\mathsf{CCV}}(m,n,X,Y)$ to be the Z satisfying $\delta_{\mathsf{CCV}}(m,n.X,Y,Z)$ (with each $Z(i)$ set false when it is not determined). By a slight abuse of notation we will write $F_{\mathsf{CCV}}(C,Y)$ instead of $F_{\mathsf{CCV}}(m,n,X,Y)$ when $C = \langle m,n,X \rangle$. Thus

$$C = \langle m,n,X \rangle \wedge Z = F_{\mathsf{CCV}}(C,Y) \rightarrow \delta_{\mathsf{CCV}}(m,n,X,Y,Z) \tag{3.3}$$

It follows from the axiom (3.2) that F_{CCV} is Σ_1^B-definable in VCC.

There is a technical lemma required to show that VCC fits the framework described in [14, Chapter IX]. We need to show that the *aggregate* F^*_{CCV} of F_{CCV} is Σ_1^B-definable in VCC, where (roughly speaking) F^*_{CCV} is the string function that gathers the values of F_{CCV} for a polynomially long sequence of arguments. The nature of CC circuits makes this easy: The sequence of outputs for a sequence of circuits can be obtained from a single circuit which computes them all in parallel: the lines of the composite circuit comprise the union of the lines of each component circuit.

Thus the framework of [14, Chapter IX] does apply to VCC, and in particular the theory $\overline{VCC}$ is a universal conservative extension of VCC whose function symbols are precisely those AC^0-reducible to the function F_{CCV}. However we will postpone to Theorem 26 in Section 3.4 the proof that the Σ_1^B-definable functions of VCC are precisely those in FCC, the class of functions AC^0 many-one reducible to the Ccv problem.

It is hard to work with VCC up to this point since we have not shown whether VCC can prove the definability of basic counting functions (as in VTC^0). However, we have the following theorem.

Theorem 18 ($VNC^1 \subseteq VCC$)**.** *The theory VCC proves the axiom MFV defined in (2.3).*

Proof. Observe that each comparator gate can produce simultaneously an AND gate and an OR gate with the only restriction that each of these gates must have fan-out one. Since all AND and OR gates of a monotone Boolean formula also have fan-out one, and the axiom *MFV* assumes that the variables of the formula are all distinct, it is easy to transform the instance of the monotone Boolean formula value problem specified by the parameters (n,F,I) in the axiom to an instance of Ccv. □

Since $VTC^0 \subseteq VNC^1$ (see (2.6)) the next result is an immediate consequence of this theorem.

Corollary 19 ($VTC^0 \subseteq VCC$)**.** *The theory VCC proves the axiom NUMONES defined in (2.2).*

This corollary is important since it allows us to use the counting ability of VTC^0 freely in VCC proofs. In particular using counting and induction on the layers of a comparator circuit, we can prove in VTC^0 the following fundamental property of comparator circuits.

Corollary 20. *Given a comparator circuit computation, the theory VTC^0 (and hence VCC) proves that all layers of the computation have the same number of ones and zeros.*

Theorem 21 ($VCC \subseteq VP$)**.** *The theory VP proves the axiom AX*Ccv *defined in (3.2).*

Proof. This easily follows since Ccv is a special case of the *monotone circuit value* problem. □

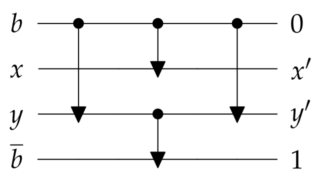


Figure 3.1: Conditional comparator gadget

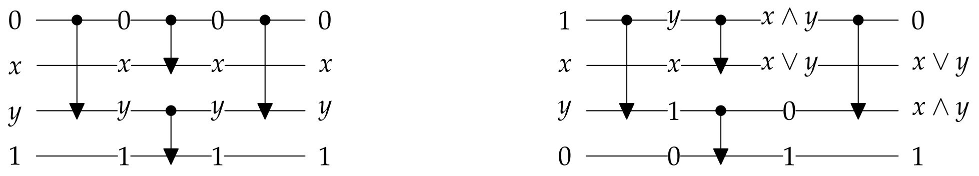


Figure 3.2: Operation of conditional comparator gadget

3.4 Universal comparator circuits

Here we present a construction of universal comparator circuits due to Yuval Filmus. The key idea is a *gadget*: a comparator circuit with four wires and four gates which allows a conditional application of a comparator to two of its inputs x, y, depending on whether a bit b is 0 or 1. The other two inputs are $\overline{b}$ and b (see Figure 3.1). The comparator is applied only when $b = 1$ (see Figure 3.2).

In order to simulate a single arbitrary comparator in a circuit with m wires we put in $m(m-1)$ gadgets in a row, for the $m(m-1)$ possible comparators. Simulating n comparators requires $m(m-1)n$ gadgets.

Thus there is an AC^0 function UNIV such that if m, n are arbitrary parameters, then $U = \mathsf{UNIV}(m,n) = \langle m', n', U' \rangle$ is a universal circuit which simulates all comparator networks with at most m wires and at most n comparators, where the number of wires in U is $m' = 4m(m-1)n + m$ (note that the original m wires are common to all of the gadgets) and the number of gates is $n' = 7m(m-1)n$.

The AC^0 function $\mathsf{INPUT}(C, Y) = Y'$ computes the input bits Y' for the universal circuit $U = \mathsf{UNIV}(m,n)$, where $C = \langle \hat{m}, \hat{n}, X \rangle$ with $\hat{m} \le m$ and $\hat{n} \le n$. Then U with input Y' simulates the circuit C with input Y. We may arrange the universal circuit so that the $\hat{m}$ wires of the original circuit C correspond to wires number $0, 1, \ldots, \hat{m} - 1$ in $\mathsf{UNIV}(m,n)$ (and the remaining input wires specified by Y' code the inputs to the gadgets in U to simulate the comparator gates of C).

Theorem 22. *VCC proves the following formula, stating that the circuit* $\mathsf{UNIV}(m,n)$ *has the intended universal property.*

$$C = \langle \hat{m}, \hat{n}, X \rangle \wedge \hat{m} \le m \wedge \hat{n} \le n \wedge j < \hat{m} \quad \to$$
$$\big[F_{\mathsf{CCV}}(C, Y)(\hat{n}, j) \quad \leftrightarrow \quad F_{\mathsf{CCV}}\big(\mathsf{UNIV}(m,n), \mathsf{INPUT}(C,Y)\big)(n', j)\big]$$

where $n' = 7m(m-1)n$.

Proof. Use induction on i to show that

$$\begin{aligned} C = \langle \hat{m}, \hat{n}, X\rangle \wedge \hat{m} \le m \wedge \hat{n} \le n \quad &\to \\ \forall j < \hat{m}\big[F_{\mathsf{CCV}}(C,Y)(i,j) \quad &\leftrightarrow \quad F_{\mathsf{CCV}}\big(\mathsf{UNIV}(m,n), \mathsf{INPUT}(C,Y)\big)(i',j)\big] \end{aligned}$$

where i' is the layer in $\mathsf{UNIV}(m,n)$ which corresponds to layer i in the circuit C. □

The next result is an important application of universal comparator circuits. It tells us that the class CC can be characterized in terms of uniform circuit families, as in the definitions of the complexity classes NC^k and AC^k.

Theorem 23. *For each relation $R(X)$ in* CC *there is a family $\{C_k^R\}_{k\in\mathbb{N}}$ of comparator circuits described by* AC^0 *functions of k such that wire zero of C_k^R outputs $R(X)$ when supplied with copies of $X(i), \neg X(i)$ for $i < k$ as inputs. More precisely, there are* AC^0 *functions* $\mathsf{CIRCUIT}^R$ *and* IN^R *such that VCC proves*

$$|X| \le k \to \big[R(X) \leftrightarrow F_{\mathsf{CCV}}\big(\mathsf{CIRCUIT}^R(k), \mathsf{IN}^R(k,X)\big)(n,0)\big]$$

where n is the number of comparator gates in $\mathsf{CIRCUIT}^R(k)$ and $Y = \mathsf{IN}^R(k,X)$ consists of (for some polynomial $p = p(k)$) p copies of $X(i)$ and p copies of $\neg X(i)$, for $i = 0, 1, \ldots, k-1$. A similar result holds more generally for relations $R(\vec{x}, \vec{X})$ in CC.

Proof. First observe that the theorem holds in case the relation $R(X)$ is in AC^0. This is because an AC^0 circuit is easily converted to a tree whose internal nodes are AND or OR gates with fan-out one (which can be converted to comparator gates) and whose leaves have inputs of the form $\mathsf{IN}^R(k,X)$ as described in the theorem. (No universal circuit is needed for this.)

If $F(X)$ is an AC^0 function, then its bit graph $B_F(i,X)$ is an AC^0 relation. Hence the above construction can be used to construct an AC^0-uniform comparator circuit family $\mathsf{CIRCUIT}^F(k)$ which on input $\mathsf{IN}^F(k,X)$ of the form described in the theorem, outputs the bits of $F(X)$. This construction will be applied to the function $F(X)$ defined in (3.4) below.

Now suppose $R(X)$ is in CC. Then $R(X)$ is AC^0 many-one reducible to Ccv (see Definition 15). Hence there are AC^0 functions $\mathsf{CIR}^R(X)$ and $\mathsf{INP}^R(X)$ such that

$$R(X) \leftrightarrow \mathrm{Ccv}\left(0, \mathsf{CIR}^R(X), \mathsf{INP}^R(X)\right)$$

where we assume the output wire $i = 0$. (This can be accomplished by renumbering the wires in the comparator circuit.)

Now given k, choose m_k and n_k such that for all X with $|X| \le k$, if $\langle m, n, X'\rangle = \mathsf{CIR}^R(X)$ then $m \le m_k$ and $n \le n_k$. We may choose m_k and n_k to be polynomials in k, because CIR^R is an AC^0 function. Then we define the circuit $\mathsf{CIRCUIT}^R(k)$ in the theorem to consist of the universal circuit $\mathsf{UNIV}(m_k, n_k)$ preceded by a comparator circuit $\mathsf{CIRCUIT}^F(k)$ computing the AC^0 function

$$F(X) = \mathsf{INPUT}\left(\mathsf{CIR}^R(X), \mathsf{INP}^R(X)\right). \tag{3.4}$$

for $|X| \le k$, where INPUT is as in Theorem 22. The outputs of this circuit are connected to the inputs of $\mathsf{UNIV}(m_k, n_k)$. The function $\mathsf{IN}^R(k,X)$ is the same as $\mathsf{IN}^F(k,X)$, which supplies inputs to $\mathsf{CIRCUIT}^F(k)$ (see Fig. 3.3).

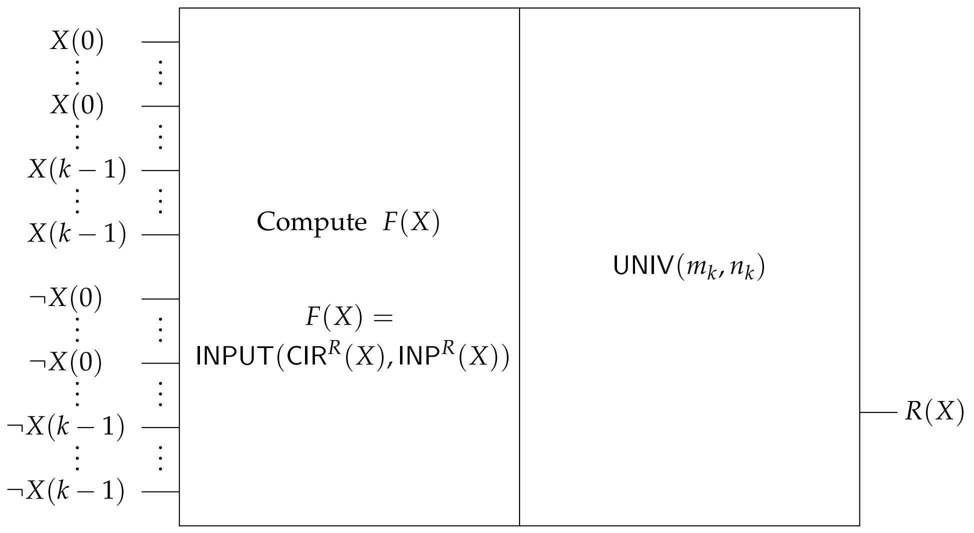


Figure 3.3: The construction of a comparator circuit for any relation $R(X)$ in CC for a fixed k.

Thus by (3.4) and Theorem 22, wire 0 of $\mathsf{UNIV}(m_k, n_k)$ outputs $R(X)$. □

Corollary 24. *The function class* FCC *is closed under composition.*

Proof. For notational simplicity we prove this for unary string functions $G(X)$. We must show that if $G(X)$ and $H(X)$ are in FCC then so is $H \circ G(X)$. Recall that a function $G(X)$ is in FCC if and only if $G(X)$ is polynomially bounded and its bit graph $B_G(i, X)$ is in CC. Thus by Theorem 23 and its proof, a function $G(X)$ is in FCC if and only if there is an AC^0-uniform family $\{C_k^G\}_{k \in \mathbb{N}}$ such that for $|X| \leq k$, when polynomially many copies of $X(i)$ and $\neg X(i), i < k$ are fed as inputs to the circuit C_k^G using the AC^0 function IN^G, then the outputs of the bottom wires of C_k^G are the first $p(k)$ bits of $G(X)$, where $p(k)$ is a polynomial upper bound on $|G(X)|$ for $|X| \leq k$.

The circuit $C_k^{H \circ G}$ can be constructed from polynomially many copies of comparator circuits C_k^G stacked vertically (computing many copies of $G(X)$) serve as inputs to the circuit $C_{p(k)}^H$ computing H. Actually the circuit $C_{p(k)}^H$ also needs negations of the bits of $G(X)$ as inputs. These are easily supplied, using the fact that if C is a comparator circuit and C' is the result of flipping each gate in C (interchanging AND and OR gates), then (by De Morgan's Laws) the output bits of C' with input Y' are the negations of the output bits of C with input Y, where Y' is the string of negations of the inputs Y. □

Theorem 25. CC *is closed under* AC^0 *reductions.*

Proof. By [14, Theorem IX.1.7], it suffices to show that FCC is closed under composition and string comprehension. The latter is explained by the following definition: For a number function $f(x)$ (which may contain other arguments), the *string comprehension of* f is the string function $F(y)$ satisfying

$$F(y)(i) \leftrightarrow \exists x \leq y,\ i = f(x)$$

We know FCC is closed under composition by Corollary 24, so it suffices to show that FCC is closed under string comprehension. We can show this directly from the definition of FCC, without referring to

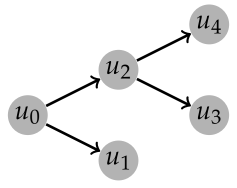


Figure 3.4

any of the previous theorems. We just observe that a collection of $y+1$ instances of Ccv determining whether $i = f(x)$ for $x = 0, 1, \ldots, y$ can be combined by AC^0 functions of y to form an instance of Ccv determining $F(y)(i)$. □

An important consequence of Theorem 25 is the following theorem, which shows that *VCC* the correct theory for CC.

Theorem 26. *The Σ_1^B-definable functions of VCC are precisely those in* FCC.

3.5 The theory *VCC* contains *VNL*

Each instance of the Reachability problem consists of a directed acyclic graph $G = (V, E)$, where $V = \{u_0, \ldots, u_{n-1}\}$, and we want to decide if there is a path from u_0 to u_{n-1}. It is well-known that Reachability is NL-complete. Since a directed graph can be converted into a layered graph with an equivalent reachability problem, it suffices to give a comparator circuit construction that solves instances of Reachability satisfying the following assumption:

$$\text{The graph } G \text{ only has directed edges of the form } (u_i, u_j), \text{ where } i < j. \tag{3.5}$$

We believe that our new construction for showing that $\mathsf{NL} \subseteq \mathsf{CC}$ is more intuitive than the one in [60, 40]. Moreover, we reduce Reachability to Ccv directly without going through some intermediate complete problem, and this was stated as an open problem in [60, Chapter 7.8.1].

We will demonstrate our construction through a simple example, where we have the directed graph in Figure 3.4 satisfying the assumption (3.5). We will build a comparator circuit as in Figure 3.5, where the wires $\nu_0, \ldots, \nu_4$ represent the vertices $u_0, \ldots, u_4$ of the preceding graph and the wires $\iota_0, \ldots, \iota_4$ are used to feed 1-bits into the wire ν_0, and from there to the other wires ν_i reachable from ν_0. We let every wire ι_i take input 1 and every wire ν_i take input 0.

We next show how to construct the gadgets in the boxes. For a graph with n vertices ($n = 5$ in our example), the k^{th} gadget is constructed as follows:

```
1: Introduce a comparator gate from wire $\iota_k$ to wire $\nu_0$
2: for $i = 0, \ldots, n-1$ do
3:     for $j = i+1, \ldots, n-1$ do
4:         Introduce a comparator gate from $\nu_i$ to $\nu_j$ if $(u_i, u_j) \in E$, or a dummy gate on $\nu_i$ otherwise.
5:     end for
6: end for
```

Note that the gadgets are identical except for the first comparator gate.

We only use the loop structure to clarify the order the gates are added. The construction can easily be done in AC^0 since the position of each gate can be calculated exactly, and thus all gates can be added independently from one another. Note that for a graph with n vertices, we have at most n vertices reachable from a single vertex, and thus we need n gadgets described above. In our example, there are at most 5 wires reachable from wire ν_0, and thus we utilize the gadget 5 times.

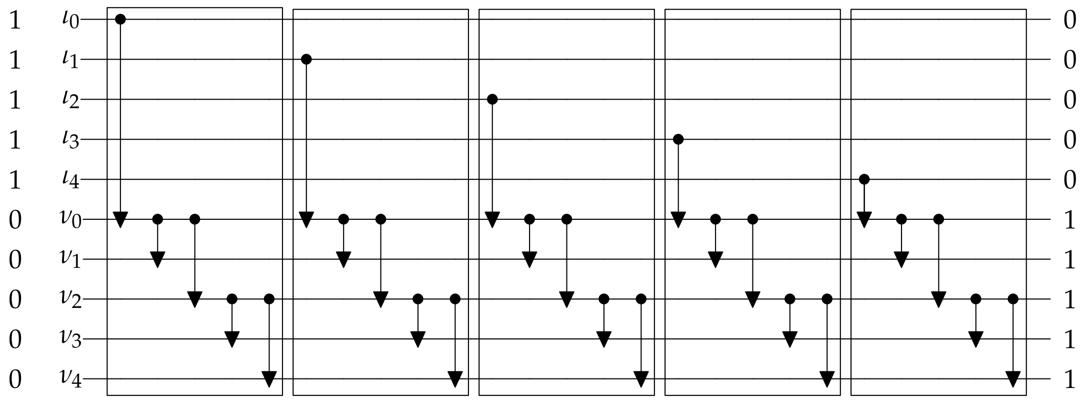


Figure 3.5: A comparator circuit that solves REACHABILITY. (The dummy gates are omitted.)

Intuitively, the construction works since each gadget from a box looks for the *lexicographical first maximal path* starting from v_0 (with respect to the *natural lexicographical ordering* induced by the vertex ordering $v_0, \ldots, v_n$), and then the vertex at the end of the path will be marked (i.e. its wire will now carry 1) and thus excluded from the search of the gadgets that follow. For example, the gadget from the left-most dashed box in Figure 3.5 will move a value 1 from wire ι_0 to wire ν_0 and from wire ν_0 to wire ν_1. This essentially "marks" the wire ν_1 since we cannot move this value 1 away from ν_1, and thus ν_1 can no longer receive any new incoming 1. Hence, the gadget from the second box in Figure 3.5 will repeat the process of finding the lex-first maximal path from v_0 to the remaining (unmarked) vertices. These searches end when all vertices reachable from v_0 are marked. Note that this has the same effect as applying the *depth-first search* algorithm to find all the vertices reachable from v_0.

Theorem 27 (*VNL $\subseteq$ VCC*)**.** *The theory VCC proves the axiom CONN defined in (2.4).*

Proof. Recall that if G is a directed graph on n vertices $u_0, \ldots, u_{n-1}$ with the edge relation E, then the formula $\delta_{\mathsf{CONN}}(n, E, U)$ holds iff U is a matrix of n rows, where row d encodes the set of all vertices in G that are reachable from u_0 using a path of length at most d.

We start by converting G into a layered graph $G' = (V', E')$ which satisfies (3.5), where

$$V' = \{u_i^\ell \mid 0 \le i, \ell < n\}, \tag{3.6}$$
$$E' = \{(u_i^\ell, u_j^{\ell+1}) \mid 0 \le i, j, \ell < n \text{ and } (i = j \text{ or } (u_i, u_j) \in E)\}.$$

Observe that a vertex u_i^ℓ is reachable from u_0^0 in G' iff u_i is reachable from u_0 by a path of length at most ℓ in G. Moreover, if we enumerate the vertices of G' by layers, then G' satisfies the condition (3.5). We now apply the above construction to G' to find all vertices in V' reachable from u_0^0. Then we construct a matrix U witnessing the formula $\delta_{\mathsf{CONN}}(n, E, U)$ by letting the row d encode the set of vertices u_i in V such that u_i^d is reachable from u_0^0 in G'.

We want to show that the comparator circuit constructed for G' using the above method produces the correct set of vertices u_i^d reachable from u_0^0 for every d. Although the correctness of the construction follows from the intuition that the circuit simulates the depth-first search algorithm, we cannot formalize this intuition directly since it would require *VNL* reasoning. Recall that up to this point, we only know that $VTC^0 \subseteq VNC^1 \subseteq VCC$ by Theorem 18. It turns out that using only the counting ability of VTC^0, we can analyze the computation of the circuit in the above construction and argue that since we feed in as many 1-bits as the number of wires representing the vertices of G', these 1-bits will eventually fill all the lower wires that are reachable from ν_0^0.

Now we begin the detailed proof. (To follow the proof it will be helpful to keep an eye on Figure 3.5.) Let C be the comparator circuit constructed from the layered graph G' of G using the above construction, where C consists of the wires $\{\nu_i \mid 0 \leq i, \ell < n^2\}$ representing the vertices of G' and wires $\{\iota_0, \dots, \iota_{n^2}\}$ are used to feed ones to the wire ν_0. It is important to note that we will order the wires from high to low by layers of G' so that the sequence of wires

$$\nu_0, \dots, \nu_{n^2-1}$$

corresponds to the sequence of nodes

$$u_0^0, u_1^0, \dots, u_{n-1}^0, u_0^1, u_1^1, \dots, u_{n-1}^1, \dots, u_0^{n-1}, u_1^{n-1}, \dots, u_{n-1}^{n-1}.$$

Let $\gamma_0, \gamma_1, \dots, \gamma_{n^2-1}$ be the successive gadgets in the circuit C.

Lemma 28. *($VCC \vdash$) For each $k \leq n^2 - 1$, if wire ν_0 has value 1 at the input of gadget γ_k then the value of each wire $\nu_i, 0 \leq i \leq n^2 - 1$, is the same at the output of γ_k as at the input of γ_k. If ν_0 has value 0 at the input of γ_k then the above is true of γ_k with exactly one exception: some wire ν_j is 0 at the input of γ_k and 1 at the output of γ_k.*

Proof. Note that for $k < n^2 - 1$ the output values of γ_k are the same as the input values of γ_{k+1}.

We proceed by induction on k. For $k = 0$ the input values of wires $(\nu_0, \dots, \nu_{n^2-1})$ are 0, but after the first gate (ι_0, ν_0) the value of wire ν_0 becomes 1. Hence by Corollary 20 applied to γ_0 starting after that gate (or by induction on the depth of the gates in γ_0), the output values of wires $(\nu_0, \dots, \nu_{n^2-1})$ contain a single 1, and the rest are 0.

For the induction step suppose $k > 0$. Suppose first that the value of ν_0 at the input of γ_k is 1, so the output value of ν_0 for γ_{k-1} is 1. Then it follows from the induction hypothesis that the tuple of values for wires $(\nu_0, \dots, \nu_{n^2-1})$ is either the same for the input and output of γ_{k-1} or wire ν_0 is the only exception (it must be an exception if its input value is 0 because by assumption its output value is 1). Note that after the first gate (ι_0, ν_0) in γ_{k-1} the value of wire ν_0 is certainly 1 and so at this point in the gadget γ_{k-1} the tuple of values for wires $(\nu_0, \dots, \nu_{n^2-1})$ is the same as at this point in γ_k. Since gadgets γ_{k-1} and γ_k are identical after the first gate, the outputs for wires $(\nu_0, \dots, \nu_{n^2-1})$ are the same for the two gadgets. Thus the lemma follows for this case.

Now suppose that the value of ν_0 at the input of γ_k (and hence the output of γ_{k-1}) is 0. Then by the induction hypothesis the value of ν_0 at the input of γ_{k-1} is also 0 and the values of $(\nu_0, \dots, \nu_{n^2-1})$ at the inputs of γ_{k-1} and γ_k are identical except some wire ν_j $(j \neq 0)$ is 0 for γ_{k-1} and 1 for γ_k. After the first gate in each gadget the value of ν_0 is 1, and since the gadgets are identical except for the first gate, it follows by induction on p that the value of each wire at position p in γ_{k-1} is the same as the value of that wire at position p in γ_k, except for each p one wire is 0 in γ_{k-1} and 1 in γ_k. The lemma follows when p is the output position. □

The next result follows easily from the preceding lemma by induction on k.

Lemma 29. *(VCC $\vdash$) If a wire ν_i has value* 1 *at the output of some gadget γ_k, then it has value* 1 *at the outputs of all succeeding gadgets. If the tuple of output values for $(\nu_0, \ldots, \nu_{n^2-1})$ is the same for two successive gadgets γ_k and γ_{k+1} then $\nu_0 = 1$ and the tuple remains the same for the outputs of all succeeding gadgets.*

The next result is the only place in the proof of Theorem 27 that makes essential use of the ability of *VCC* to count, as in Corollary 20.

Lemma 30. *(VCC $\vdash$) Wire ν_0 has value* 1 *at the output of the circuit C.*

Proof. By Lemma 29, if ν_0 has value 0 at the output of C, then it has value 0 at the output of every gadget γ_k. Since there are n^2 ones at the input to C (one for every wire ι_i), by Corollary 20 there are n^2 ones at the output of C. But if ν_0 remains 0 after every gadget, then by the construction of C we see that the final output of every wire ι_i is 0, and hence outputs of all n^2 wires ν_i must be 1, including ν_0. □

Lemma 31. *(VCC $\vdash$) In the final gadget, no wire ν_i changes its value after any comparator gate, except possibly ν_0 changes from* 0 *to* 1 *after the initial gate feeds a* 1 *from ι_{n^2-1}.*

Proof. We use induction on i. For $i = 0$, by Lemma 30 the output of ν_0 in the final gadget is 1. At the input to this gadget ν_0 can be either 0 or 1, but after the first gate it is certainly 1. The value of ν_0 cannot change from 1 to 0 in the gadget, because after the first gate there is no further gate leading down to ν_0 which could bring down a 1 to change the value of ν_0 from 0 to 1, but we know the final output value is 1.

For $i > 0$ it follows from Lemmas 28 and 30 that the value of wire ν_i is the same at the input and output of the final gadget. We note that all gates leading down to ν_i precede all gates leading away from ν_i. The only way that ν_i can change from 0 to 1 is at a gate bringing down a 1 from above, but that would change the value of the wire above, violating the induction hypothesis. The only way that ν_i can change from 1 to 0 is at a gate leading away from ν_0, but then ν_i cannot change back to 1, so the input and output cannot be the same. □

Lemma 32. *(VCC $\vdash$) If a wire ν_i has value* 1 *at some position p in some gadget γ_k, then the output of ν_i in the final gadget is* 1.

Proof. By Lemma 28 the tuple of input values for wires $\nu_0, \ldots, \nu_{n^2-1}$ is the same for gadgets γ_k and γ_{k+1} except possibly some input changes from 0 to 1. From this and the fact that the gadgets γ_k and γ_{k+1} are the same except for the first gate, it is easy to prove by induction on p that at position in p each wire has the same value in γ_{k+1} as in γ_k, except the value might be 0 in γ_k and 1 in γ_{k+1}.

Therefore by induction on k, if some wire ν_i has value 1 in position p in some gadget then wire ν_i has value 1 in position p in the final gadget. In this case, by Lemma 31 the output of ν_i in the final gadget is 1. □

Lemma 33. *(VCC $\vdash$) Let $j > 0$ and let y be the node in G' corresponding to wire ν_j. Then the output of ν_j in C is* 1 *iff there is $i < j$ such that the output of wire ν_i in C is* 1 *and there is an edge from x to y, where x is the node in G' which corresponds to ν_i.*

Proof. For the direction ($\Leftarrow$), suppose that the output of wire ν_i is 1 and there is an edge from x to y in G'. Then there is a gate from ν_i to ν_j in the final gadget. Then the value of ν_j must be 1 by Lemma 31, since otherwise this gate would change ν_i from 1 to 0. Since the value of ν_j cannot change, its value is 1 at the output.

For the direction ($\Rightarrow$) suppose the value of ν_j is 1 at the output. Since the value of ν_j at the input to C is 0, there is some gadget γ_k such that the value of ν_j changes from 0 to 1. For this to happen there must be some comparator gate in γ_k from some wire ν_i down to ν_j with $i < j$ such that the value of ν_i before the gate is 1, and there is an edge from the node x corresponding to ν_i to node y. By Lemma 32 the value of ν_i at the output of the final gadget is 1. □

To complete the proof of Theorem 27 recall the meaning of the array U given in (2.5), and the relation between the graph $G = (V, E)$ and the graph $G' = (V', E')$ given by (3.6). We define $U(d, i)$ (the truth value of node u_i of G in row d of the array U) to be true iff the the output of wire ν_j in circuit C is 1, where ν_j is the wire corresponding to node u_i^d in G'.

We prove that this definition of U satisfies the formula $\delta_{\mathsf{CONN}}(n, E, U)$ (2.5) (which appears in the axiom (2.4) for *VNL*) by induction on d. The base case is $d = 0$. We have $U(0, 0)$ holds because the output of ν_0 (the wire corresponding to node u_0^0) is 1 by Lemma 30. For $i > 0$, $U(0, i)$ is false because there is no edge in G' leading to node u_i^0, and hence there is no comparator gate in any gadget in C leading down to the wire corresponding to u_i^0, so that wire has output 0.

The induction step follows directly from our definition of U above, together with (2.5) and Lemma 33. □

As a of consequence of Theorem 27, we have the following result.

Theorem 34. CC *is closed under* NL *many-one reductions, and hence is closed under logspace many-one reductions.*

Proof. This follows from the following three facts: The function class FCC is closed under composition, FNL $\subseteq$ FCC, and a decision problem is in CC if and only if its characteristic function is in FCC. □

3.6 Lexicographical first maximal matching problem is CC-complete

Let $G = (V, W, E)$ be a bipartite graph, where $V = \{v_i\}_{i=0}^{m-1}$, $W = \{w_i\}_{i=0}^{n-1}$ and $E \subseteq V \times W$. The *lexicographical first maximal matching* (lfm-matching) is the matching produced by successively matching each vertex $v_0, \ldots, v_{m-1}$ to the least vertex available in W.

Formally, let $E_{m\times n}$ be a matrix encoding the edge relation of a bipartite graph with m bottom nodes and n top nodes, where $E(i, j) = 1$ iff the bottom node v_i is adjacent to the top node w_j. Let L be a matrix of the same size as E with the following intended interpretation: $L(i, j) = 1$ iff the edge (v_i, w_j) is in the lfm-matching. We can define a Σ_0^B-formula $\delta_{\mathsf{LFMM}}(m, n, X, L)$, which asserts that L properly encodes the lfm-matching of the bipartite graph represented by X as follows:

$$\forall i < m \forall j < n,\ L(i,j) \leftrightarrow \left[\begin{array}{lll} E(i,j) & \wedge & \forall k < j \forall \ell < i\big(\neg L(i,k) \wedge \neg L(\ell,j)\big) \\ & \wedge & \forall k < j\big(\neg E(i,k) \vee \exists i' < i\, L(i',k)\big) \end{array} \right]. \tag{3.7}$$

$$\begin{tikzcd}
\text{vL\textsc{fmm}} \arrow[dr, "\S3.6.2"'] & & \text{3vL\textsc{fmm}} \arrow[ll] \\
& \text{C\textsc{cv}} \arrow[ur, "\S3.6.1"'] \arrow[dr, "\S3.6.3"] & \\
\text{C\textsc{cv}}\neg \arrow[ur, "\S3.6.4"'] & & \text{3L\textsc{fmm}} \arrow[dl] \\
& \text{L\textsc{fmm}} \arrow[ul, "\S3.6.5"] &
\end{tikzcd}$$

Figure 3.6: Label of an arrow denotes the section in which the reduction is described. Arrows without labels denote trivial reductions.

In this section we will show that two decision problems concerning the lexicographical matching of a bipartite graph are CC-complete (under AC^0 many-one reductions). The lexicographical first maximal matching problem (Lfmm) is to decide if a designated edge belongs to the lfm-matching of a bipartite graph G. The vertex version of lexicographical first maximal matching problem (vLfmm) is to decide if a designated top node is matched in the lfm-matching of a bipartite graph G. Lfmm is the usual way to define a decision problem for lexicographical first maximal matching as seen in [40, 61]; however, as shown in Sections 3.6.1 and 3.6.2, the vLfmm problem is even more closely related to the Ccv problem.

We will show that the following two more restricted lexicographical matching problems are also CC-complete. We define 3Lfmm to be the restriction of Lfmm to bipartite graphs of degree at most three. We define 3vLfmm to be the restriction of vLfmm to bipartite graphs of degree at most three.

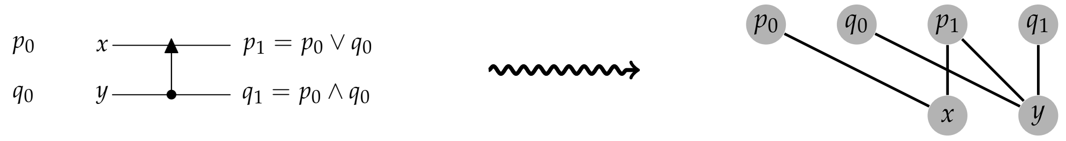
To show that the problems defined above are equivalent under AC^0 many-one reductions, it turns out that we also need the following intermediate problem. A negation gate flips the value on a wire. For comparator circuits with negation gates, we allow negation gates to appear on any wire (see the left diagram of Figure 3.10 below for an example). The comparator circuit value problem with negation gates (Ccv$\neg$) is: given a comparator circuit with negation gates and input assignment, and a designated wire, decide if that wire outputs 1.

All reductions in this section are summarized in Figure 3.6.

3.6.1 Ccv $\leq_{\mathsf{m}}^{\mathsf{AC}^0}$ 3vLfmm

By Proposition 16 it suffices to consider only instances of Ccv in which all comparator gates point upward. We will show that these instances of Ccv are AC^0 many-one reducible to instances of 3vLfmm, which consist of bipartite graphs with *degree at most three*.

The key observation is that a comparator gate on the left below closely relates to an instance of 3vLfmm on the right. We use the top nodes p_0 and q_0 to represent the values p_0 and q_0 carried by the wires x and y respectively before the comparator gate, and the nodes p_1 and q_1 to represent the values of x and y after the comparator gate, where a top node is matched iff its respective value is one.



If nodes p_0 and q_0 are not previously matched, i.e. $p_0 = q_0 = 0$ in the comparator circuit, then edges $\langle x, p_0 \rangle$ and $\langle y, q_0 \rangle$ are added to the lfm-matching. So the nodes p_1 and q_1 are not matched. If p_0 is previously matched, but q_0 is not, then edges $\langle x, p_1 \rangle$ and $\langle y, q_0 \rangle$ are added to the lfm-matching. So the node p_1 will be matched but q_1 will remain unmatched. The other two cases are similar.

Thus, we can reduce a comparator circuit to the bipartite graph of an 3vLfmm instance by converting each comparator gate into a "gadget" described above. We will describe our method through an example, where we are given the comparator circuit in Figure 3.7.

We divide the comparator circuit into vertical layers 0, 1, and 2 as shown in Figure 3.7. Since the circuit has three wires a, b and c, for each layer i, we use six nodes, including three top nodes a_i, b_i and c_i representing the values of the wires a, b and c respectively, and three bottom nodes a'_i, b'_i, c'_i, which are auxiliary nodes used to simulate the effect of the comparator gate at layer i.

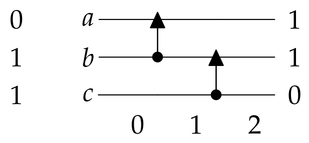
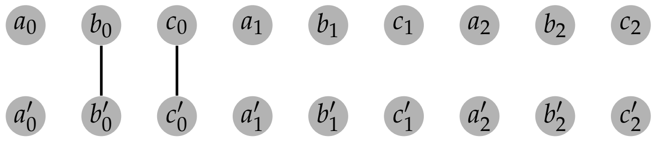
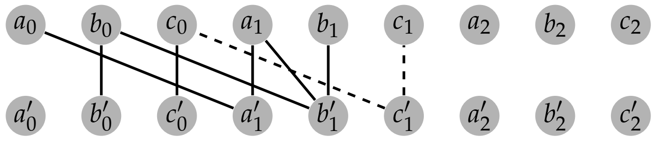


Figure 3.7

Layer 0: This is the input layer, so we add an edge $\{x_i, x'_i\}$ iff the wire x takes input value 1. In this example, since b and c are wires taking input 1, we need to add the edges $\{b_0, b'_0\}$ and $\{c_0, c'_0\}$.

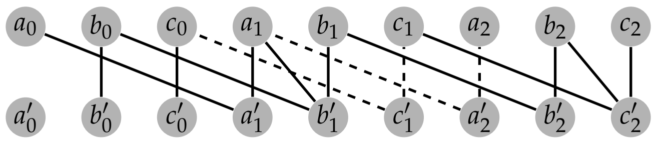


Layer 1: We then add the gadget simulating the comparator gate from wire b to wire a as follows.



Since the value of wire c does not change when going from layer 0 to layer 1, we can simply propagate the value of c_0 to c_1 using the pair of dashed edges in the picture.

Layer 2: We proceed very similarly to layer 1 to get the following bipartite graph.



Finally, we can get the output values of the comparator circuit by looking at the "output" nodes a_2, b_2, c_2 of this bipartite graph. We can easily check that a_2 is the only node that remains unmatched, which corresponds exactly to the only zero produced by wire a of the comparator circuit in Figure 3.7.

Remark 35. The construction above is an AC^0 many-one reduction since each gate in the comparator circuit can be reduced to exactly one gadget in the bipartite graph that simulates the effect of the comparator gate. Note that since it can be tedious and unintuitive to work with AC^0-circuits, it might seem hard to justify that our reduction is an AC^0-function. However, thanks to Theorem 1, we do not have to work with AC^0-circuits directly; instead, it is not hard to construct a Σ_0^B-formula that defines the above reduction.

The correctness of our construction can be proved in *VCC* by using Σ_0^B induction on the layers of the circuits and arguing that the matching information of the nodes in the bipartite graph can be correctly translated to the values carried by the wires at each layer.

3.6.2 vLFMM $\leq_{\mathsf{m}}^{\mathsf{AC}^0}$ CCV

Consider an instance of vLFMM consisting of a bipartite graph in Figure 3.8. Recall that we find the lfm-matching by matching the bottom nodes $x, y, \ldots$ successively to the first available node on the top. Hence we can simulate the matching of the bottom nodes to the top nodes using comparator circuit on the right of Figure 3.8, where we can think of the moving of a one, say from wire x to wire a, as the matching of node x to node a in the original bipartite graph. In this construction, a top node is matched iff its corresponding wire in the circuit outputs 1.

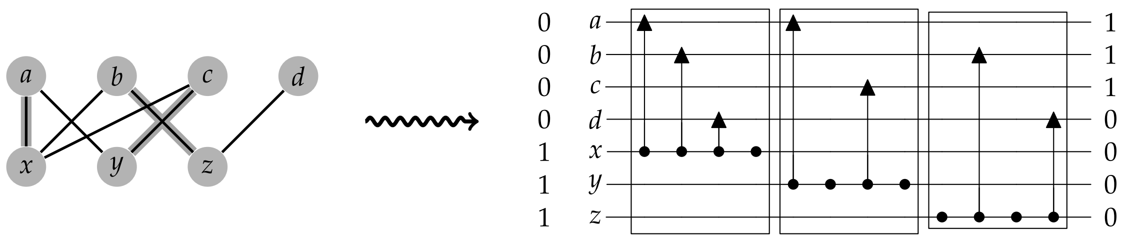


Figure 3.8

Note that we draw bullets without any arrows going out from them in the circuit to denote dummy gates, which do nothing. These dummy gates are introduced for the following technical reason. Since the bottom nodes might not have the same degree, the position of a comparator gate really depends on the number of edges that do not appear in the bipartite graph, which makes it harder to give a direct AC^0 reduction. By using dummy gates, we can treat the graph as if it is a complete bipartite graph, where the missing edges are represented by dummy gates. This can easily be shown to be an AC^0 reduction from vLFMM to CCV, and its correctness can be carried out in *VCC* using Σ_0^B-induction on the layers of the circuit. This together with the reduction from Section 3.6.1 gives us the following theorem.

Theorem 36. *(VCC $\vdash$) The problems* CCV, 3vLFMM *and* vLFMM *are equivalent under* AC^0 *many-one reductions.*

3.6.3 CCV $\leq_{\mathsf{m}}^{\mathsf{AC}^0}$ 3LFMM

We start by applying the reduction CCV $\leq_{\mathsf{m}}^{\mathsf{AC}^0}$ 3vLFMM of Section 3.6.1 to get an instance of 3vLFMM, and notice that the degrees of the top "output" nodes of the resulting bipartite graph, e.g. the nodes a_2, b_2, c_2 in the example of Section 3.6.1, have degree at most two. Now we show how to reduce such instances of 3vLFMM (i.e. those whose designated top vertices have degree at most two) to 3LFMM. Consider the graph G with degree at most three and a designated top vertex b of degree two as shown on the left of Figure 3.9. We construct a bipartite graph G', which contains a copy of the graph G and one additional top node w_t and one additional bottom node w_b, and two edges $\{b, w_b\}$ and $\{w_t, w_b\}$, as shown in Figure 3.9. Observe that the degree of the new graph G' is at most three.

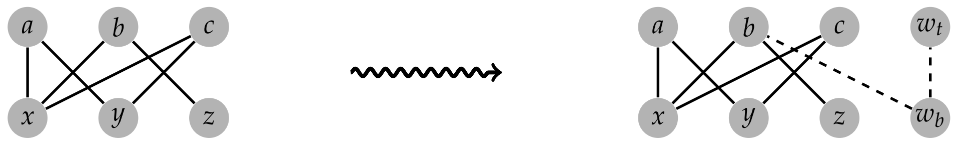


Figure 3.9

We treat the resulting bipartite graph G' and the edge $\{w_t, w_b\}$ as an instance of 3Lfmm. It is not hard to see that the vertex b is matched in the lfm-matching of the original bipartite graph G iff the edge $\{w_t, w_b\}$ is in the lfm-matching of the new bipartite graph G'.

3.6.4 Ccv¬ $\leq_m^{\mathsf{AC}^0}$ Ccv

Recall that a comparator circuit value problem with negation gates (Ccv¬) is the task of deciding, on a comparator circuit with negation gates and an input assignment, if a designated wire outputs one. It should be clear that Ccv is a special case of Ccv¬ and hence AC^0 many-one reducible to Ccv¬. Here, we show the nontrivial direction that Ccv¬ $\leq_m^{\mathsf{AC}^0}$ Ccv.

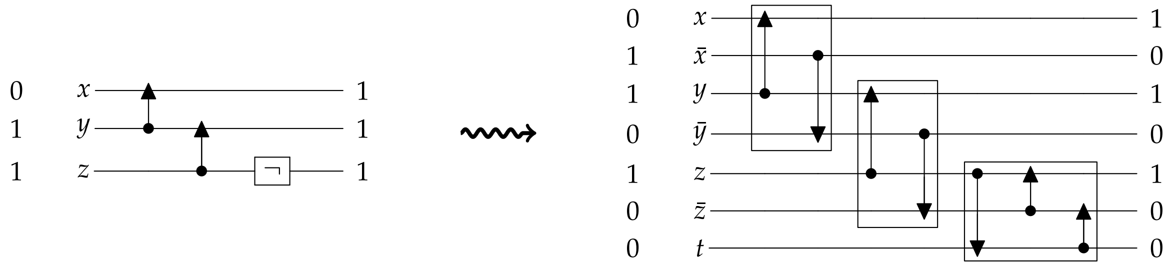


Figure 3.10: Successive gates on the left circuit corresponds to the successive boxes of gates on the right circuit.

This reduction is based on "double-rail" logic. Given an instance of Ccv¬ consisting of a comparator circuit with negation gates C with its input I and a designated wire s, we construct an instance of Ccv consisting of a comparator circuit C' with its input I' and a designated wire s' as follows. For every wire w in I we put in two corresponding wires, w and $\bar{w}$, in C'. We define input I' of C' such that the input value of $\bar{w}$ is the negation of the input value of w. We want to fix things so that the value carried by the wire $\bar{w}$ at each layer is always the negation of the value carried by w. For any comparator gate $\langle y, x\rangle$ in C we put in both the gate $\langle y, x\rangle$ and a second gate $\langle \bar{x}, \bar{y}\rangle$ in C' immediately after $\langle y, x\rangle$. It is easy to check by De Morgan's laws that the wires x and y in C' carry the corresponding values of x and y in C, and the wires $\bar{x}$ and $\bar{y}$ in C' carry the corresponding negations of x and y in C.

The circuit C' has one extra wire t with input value 0 to help in translating negation gates. For each negation gate on a wire, says z in the example from Figure 3.10, we add and three comparator gates $\langle z, t\rangle$, $\langle \bar{z}, z\rangle$, $\langle t, \bar{z}\rangle$ as shown in the right circuit of Figure 3.10. Thus t as a temporary "container" that we use to swap the values of carried by the wires z and $\bar{z}$. Note that the swapping of values of z and $\bar{z}$ in C' simulates the effect of a negation in C. Also note that after the swap takes place the value of t is restored to 0.

Finally note that the output value of the designated wire s in C is 1 iff the output value of the corresponding wire s in C' with input I' is 1. Thus we set the designated wire s' in I' to be s.

3.6.5 LFMM $\leq_{\mathsf{m}}^{\mathsf{AC}^0}$ CCV$\neg$

Consider an instance of LFMM consisting of a bipartite graph on the left of Figure 3.11, and a designated edge $\{y, c\}$. Without loss of generality, we can safely ignore all top vertices occurred after c, all bottom vertices occurred after y and all the edges associated with them, since they are not going to affect the outcome of the instance. Using the construction from Section 3.6.2, we can simulate the matching of the bottom nodes to the top nodes using comparator circuit in the upper box on the right of Figure 3.11.

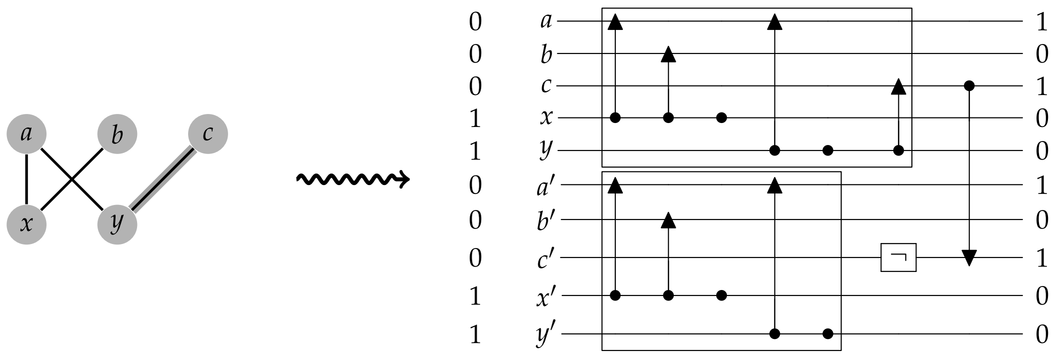


Figure 3.11

We keep another running copy of this simulation on the bottom, (see the wires labelled a', b', c', x', y' in Figure 3.11). The only difference is that the comparator gate $\langle y', c' \rangle$ corresponding to the designated edge $\{y, c\}$ is not added. We finally add a negation gate on c' and a comparator gate $\langle c, c' \rangle$. We let the desired output of the Ccv instance be the output of c, since c outputs 1 iff the edge $\{y, c\}$ is added to the lfm-matching. It is not hard to see that such construction can be generalized, and the output correctly computes if the designated edge is in the lfm-matching.

Theorem 37. *(VCC $\vdash$) The problems* CCV, CCV$\neg$, 3LFMM *and* LFMM *are equivalent under* AC^0 *many-one reductions.*

Combined with the results from Sections 3.6.1 and 3.6.2, we have the following corollary.

Corollary 38. *(VCC $\vdash$) The problems* CCV, 3vLFMM, vLFMM, CCV$\neg$, 3LFMM *and* LFMM *are equivalent under* AC^0 *many-one reductions.*

Since CCV$\neg$ is complete for CC, we can use comparator circuits to decide the complement of the Ccv problem: given a comparator circuit and and input assignment, does a designated wire output 0? Thus, we have the following corollary.

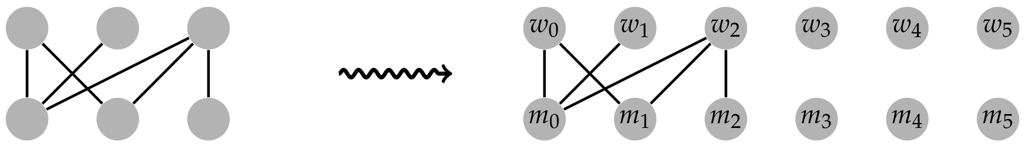
Corollary 39. CC *is closed under complementation.*

3.7 The Sm problem is CC-complete

3.7.1 3Lfmm is AC^0 many-one reducible to Sm, MoSm and WoSm

We start by showing that 3Lfmm is AC^0 many-one reducible to Sm in the second sense of Definition 2; i.e. we regard both 3Lfmm and Sm as search problems. (Of course the lfm-matching is the unique solution to 3Lfmm formulated as a search problem, but it is still a total search problem.)

Let $G = (V, W, E)$ be a bipartite graph from an instance of 3Lfmm, where V is the set of bottom nodes, W is the set of top nodes, and E is the edge relation such that the degree of each node is at most three (see the example in the figure on the left below). Without loss of generality, we can assume that $|V| = |W| = n$. To reduce it to an instance of Sm, we double the number of nodes in each partition, where the new nodes are enumerated after the original nodes and the original nodes are enumerated using the ordering of the original bipartite graph, as shown in the diagram on the right below. We also let the bottom nodes and top nodes represent the men and women respectively.



It remains to define a preference list for each person in this Sm instance. The preference list of each man m_i, who represents a bottom node in the original graph, starts with all the woman w_j (at most three of them) adjacent to m_i in the order that these women are enumerated, followed by all the women $w_n, \ldots, w_{2n-1}$; the list ends with all women w_j not adjacent to m_i also in the order that they are enumerated. For example, the preference list of m_2 in our example is $w_2, w_3, w_4, w_5, w_0, w_1$. The preference list of each newly introduced man m_{n+i} simply consists of $w_0, \ldots, w_{n-1}, w_n, \ldots, w_{2n-1}$, i.e., in the order that the top nodes are listed. Preference lists for the women are defined dually.

Intuitively, the preference lists are constructed so that any stable marriage (not necessarily man-optimal) of the new Sm instance must contain the lfm-matching of G. Furthermore, if a bottom node u from the original graph is not matched to any top node in the lfm-matching of G, then the man m_i representing u will marry some top node w_{n+j}, which is a dummy node that does not correspond to any node of G.

Formally, let $\mathcal{I}$ be an instance of Sm constructed from a bipartite graph $G = (V, W, E)$ using the above construction, where the set of men is $\{m_i\}_{i=0}^{2n-1}$ and the set of women is $\{w_i\}_{i=0}^{2n-1}$, and the preference lists are defined as above. For convenience, assume that the set of bottom nodes and top nodes of G are $V = \{m_i\}_{i=0}^{n-1}$ and $W = \{w_i\}_{i=0}^{n-1}$ respectively; the set of newly added bottom nodes and top nodes are $V' = \{m_i\}_{i=n}^{2n-1}$ and $W' = \{w_i\}_{i=n}^{2n-1}$ respectively. We will encode the edge relation of G by a Boolean matrix $E_{n\times n}$, where $E(i,j) = 1$ iff m_i is adjacent to w_j in G. Similarly, we encode the lfm-matching of G by Boolean matrix $L_{n\times n}$. We encode a stable marriage by a Boolean matrix $M_{2n\times 2n}$, and thus $M(i,j) = 1$ iff m_i marries w_j in M. We first prove the following lemma.

Lemma 40. *(VCC $\vdash$) Given a stable marriage M, if $M(i,j) = 1$ for some $i, j < n$, then $E(i,j) = 1$.*

Proof. We prove by contradiction. Suppose $M(i,j) = 1$ for some $i, j < n$, but $E(i,j) = 0$, then since M is a perfect matching, by the pigeonhole principle $\mathsf{PHP}(n-1, M)$, which is provable in VTC^0 (and hence in VCC), we cannot map the set of n men V' into the set of $n-1$ women W. Thus, there must exist some $p \geq n$ and $q \geq n$ such that $M(p,q) = 1$. Since m_i prefers w_q to w_j and w_j prefers m_q to m_i, M is not stable; hence a contradiction. □

Lemma 41. *(VCC $\vdash$) Let M be a stable marriage of the* SM *instance $\mathcal{I}$ reduced from the graph G, and let L be the lfm-matching of G. Then we have $L = \big(\{0,\ldots,n-1\} \times \{0,\ldots,n-1\}\big) \cap M$, where here L and M are treated as relations.*

Proof. First, we will show that L is contained in $\big(\{0,\ldots,n-1\} \times \{0,\ldots,n-1\}\big) \cap M$. Suppose otherwise, then there must exist a pair (i,j), called "bad pair", $i,j < n$, such that $L(i,j) = 1$ but $M(i,k) = 1$ for some $k \neq j$. Using the Σ_0^B-*MIN* principle, we pick the "bad pair" (i,j) with minimum man index. There are two cases:

1. If $k < j$, then we cannot have $L(h,k) = 1$ for any $h < i$ for otherwise the pair (h,k) is a "bad pair" with a smaller man index than (i,j), which is a contradiction. Therefore, $L(h,k) = 0$ for any $h < i$. By Lemma 40, we have $E(i,k) = 1$. Note that for any $\ell < k$, we have $L(i,\ell) = 0$, and furthermore, by the property of lfm-matching, for any $\ell < k$, either $E(i,\ell) = 0$ or there exists some $i' < i$ such that $L(i',\ell) = 1$. Therefore, $L(i,k) = 1$ by Eq. (3.7), which contradicts the fact that $L(i,j) = 1$.

2. Otherwise $k > j$, then we cannot have $M(h,j) = 1$ for any $h > i$ for otherwise m_i prefers w_j to w_k and w_j prefers m_i to m_h, which implies that M is not stable. Therefore, $M(h,j) = 1$ for some $h < i$. By Lemma 40, we have $E(h,j) = 1$. Note that for any $\ell < j$, $L(h,\ell) = 0$, for otherwise the pair (h,ℓ) is a "bad pair" with a smaller man index than (i,j), which is a contradiction. Furthermore, by the property of lfm-matching, for any $\ell < j$, either $E(h,\ell) = 0$ or there exists some $i' < i$ such that $L(i',\ell) = 1$. Since for any $p < h$, we have $L(p,j) = 0$, by Eq. (3.7), we have $L(h,j) = 1$, which contradicts the fact that $L(i,j) = 1$.

Next, it remains to show that L cannot be strictly contained in $\big(\{0,\ldots,n-1\} \times \{0,\ldots,n-1\}\big) \cap M$. Suppose otherwise, let (i,j), $i,j < n$, be a pair such that $M(i,j) = 1$ but for all $k < n$ and for all $\ell < n$, we have $L(i,k) = 0$ and $L(\ell,j) = 0$. By Lemma 40, we have $E(i,j) = 1$. Furthermore, by the property of lfm-matching, for any $\ell < j$, either $E(i,\ell) = 0$ or there exists some $i' < i$ such that $L(i',\ell) = 1$. By Eq. (3.7), we have $L(i,j) = 1$, which is a contradiction. □

Since Lemma 41 directly implies the correctness of the AC^0 many-one reduction from 3LFMM to SM as search problems, any solution of a stable marriage instance constructed by the above reduction provides us all the information to decide if an edge is in the lfm-matching of the original 3LFMM instance. The key explanation is that every instance of stable marriage produced by the above reduction has a unique solution; thus the man-optimal solution coincides with the woman-optimal solution. Further Lemma 41 also shows that the decision version of 3LFMM is AC^0 many-one reducible to either of the decision problems MOSM and WOSM. Hence we have proven the following theorem.

Theorem 42. *(VCC $\vdash$)* 3LFMM *is* AC^0 *many-one reducible to* SM, MOSM *and* WOSM.

3.7.2 MOSM and WOSM are AC^0 many-one reducible to CCV

In this section, we formalize a reduction from SM to CCV due to Subramanian [60, 61]. Subramanian did not reduce SM to CCV directly, but to the *network stability problem* built from the less standard X gate, which takes two inputs p and q and produces two outputs $p' = p \wedge \neg q$ and $q' = \neg p \wedge q$. It is important to note that the "*network*" notion in Subramanian's work denotes a generalization of circuits by allowing a connection from the output of a gate to the input of any gate including itself, and thus a network in his definition might contain cycles. An X-network is a network consisting only of

X gates under the important restriction that each X gate has fan-out exactly one for each output it computes. The network stability problem for X gate (Xns) is then to decide if an X-network has a stable configuration, i.e., a way to assign Boolean values to the wires of the network so that the values are compatible with all the X gates of the network. Subramanian showed in his dissertation [60] that Sm, Xns and Ccv are all equivalent under log space reductions.

We do not work with Xns in this paper since networks are less intuitive and do not have a nice graphical representation as do comparator circuits. By utilizing Subramanian's idea, we give a direct AC^0 many-one reduction from Sm to Ccv. For this goal, it turns out to be conceptually simpler to go through a new variant of Ccv, where the comparator gates are three-valued instead of Boolean.

3.7.2.1 Three-valued Ccv is CC-complete

We define the Three-valued Ccv problem similarly to Ccv, i.e., we want to decide, on a given input assignment, if a designated wire of a comparator circuit outputs one. The only difference is that each wire can now take either value 0, 1 or $*$, where a wire takes value $*$ when its value is not known to be 0 or 1. The output values of the comparator gate on two input values p and q will be defined as follows.

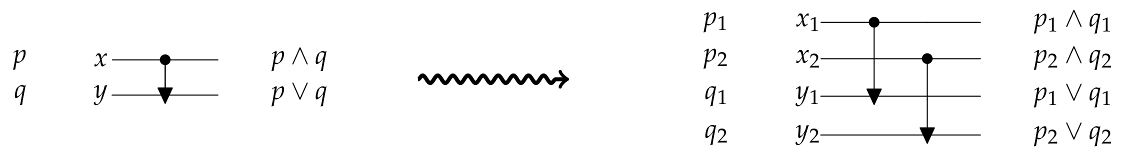
$$p \wedge q = \begin{cases} 0 & \text{if } p = 0 \text{ or } q = 0 \\ 1 & \text{if } p = q = 1 \\ * & \text{otherwise.} \end{cases} \qquad p \vee q = \begin{cases} 0 & \text{if } p = q = 0 \\ 1 & \text{if } p = 1 \text{ or } q = 1 \\ * & \text{otherwise.} \end{cases}$$

Every instance of Ccv is also an instance of Three-valued Ccv. We will show that every instance of Three-valued Ccv is AC^0 many-one reducible to an instance of Ccv by using a pair of Boolean wires to represent each three-valued wire and adding comparator gates appropriately to simulate three-valued comparator gates.

Theorem 43. *(VCC $\vdash$) Three-valued Ccv and Ccv are equivalent under AC^0 many-one reductions.*

Proof. Since each instance of Ccv is a special case of Three-valued Ccv, it only remains to show that every instance of Three-valued Ccv is AC^0 many-one reducible to an instance of Ccv.

First, we will describe a gadget built from standard comparator gates that simulates a three-valued comparator gate as follows. Each wire of an instance of Three-valued Ccv will be represented by a pair of wires in an instance of Ccv. Each three-valued comparator gate on the left below, where $p, q, p \wedge q, p \vee q \in \{0, 1, *\}$, can be simulated by a gadget with two standard comparator gates on the right below.



The wires x and y are represented using the two pairs of wires $\langle x_1, x_2 \rangle$ and $\langle y_1, y_2 \rangle$, and three possible values 0, 1 and $*$ will be encoded by $\langle 0, 0 \rangle$, $\langle 1, 1 \rangle$, and $\langle 0, 1 \rangle$ respectively. The fact that our gadget correctly simulates the three-valued comparator gate is shown in the following table.

p	q	$\langle p_1, p_2\rangle$	$\langle q_1, q_2\rangle$	$p \wedge q$	$p \vee q$	$\langle p_1 \wedge q_1, p_2 \wedge q_2\rangle$	$\langle p_1 \vee q_1, p_2 \vee q_2\rangle$
0	0	$\langle 0,0\rangle$	$\langle 0,0\rangle$	0	0	$\langle 0,0\rangle$	$\langle 0,0\rangle$
0	1	$\langle 0,0\rangle$	$\langle 1,1\rangle$	0	1	$\langle 0,0\rangle$	$\langle 1,1\rangle$
0	$*$	$\langle 0,0\rangle$	$\langle 0,1\rangle$	0	$*$	$\langle 0,0\rangle$	$\langle 0,1\rangle$
1	0	$\langle 1,1\rangle$	$\langle 0,0\rangle$	0	1	$\langle 0,0\rangle$	$\langle 1,1\rangle$
1	1	$\langle 1,1\rangle$	$\langle 1,1\rangle$	1	1	$\langle 1,1\rangle$	$\langle 1,1\rangle$
1	$*$	$\langle 1,1\rangle$	$\langle 0,1\rangle$	$*$	1	$\langle 0,1\rangle$	$\langle 1,1\rangle$
$*$	0	$\langle 0,1\rangle$	$\langle 0,0\rangle$	0	$*$	$\langle 0,0\rangle$	$\langle 0,1\rangle$
$*$	1	$\langle 0,1\rangle$	$\langle 1,1\rangle$	$*$	1	$\langle 0,1\rangle$	$\langle 1,1\rangle$
$*$	$*$	$\langle 0,1\rangle$	$\langle 0,1\rangle$	$*$	$*$	$\langle 0,1\rangle$	$\langle 0,1\rangle$

Using this gadget, we can reduce an instance of THREE-VALUED CCV to an instance of CCV by doubling the number of wires, and for every three-valued comparator gate of the THREE-VALUED CCV instance, we will add a gadget with two standard comparator gates simulating it.

The above construction shows how to reduce the question of whether a designated wire outputs 1 for a given instance of THREE-VALUED CCV to the question of whether a *pair* of wires of an instance of CCV outputs $\langle 1, 1\rangle$. However for an instance of CCV we are only allowed to decide whether a *single* designated wire outputs 1. This technical difficulty can be easily overcome since we can use an $\wedge$-gate (one of the two outputs of a comparator gate) to test whether a pair of wires outputs $\langle 1, 1\rangle$, and outputs the result on a single designated wire. □

3.7.2.2 A fixed-point method for solving stable marriage problems

We formalize a method for solving SM using three-valued comparator circuits based on [60, 61]. Consider an instance $\mathcal{I}$ of SM consisting of n men and n women and preference lists for each man and woman. From this instance we construct a three-valued comparator circuit $C_{\mathcal{I}}$. Figure 3.12 illustrates $C_{\mathcal{I}}$ when $\mathcal{I}$ consists of two men a, b and two women x, y with preference lists given by the matrices.

For each man m and woman w in $\mathcal{I}$ and each pair j, k with $j, k < n$ we say $\mathsf{Pair}(m_j, w_k)$ holds iff w is at the j^{th} position of m's preference list and m is at the k^{th} position of w's preference list. For each such pair there are two consecutive input wires of $C_{\mathcal{I}}$ labelled m^i_j and w^i_k respectively. (Here the superscript i stands for 'input'.) Hence there are n^2 pairs of input wires, making a total of $2n^2$ input wires.

In addition there are $2n^2$ other wires called 'output wires' labelled in the same order as above; two consecutive wires with labels m^o_j and w^o_k for each pair satisfying $\mathsf{Pair}(m_j, w_k)$. These output wires have fixed input values: We let output wire m^o_0 take input one for every man m, and let the rest of output wires have zero inputs.

The circuit $C_{\mathcal{I}}$ has the following comparator gates. For each pair (m^i_j, w^i_k) of consecutive inputs there is a gate from wire m^i_j to w^i_k. After these gates, for every person p, we add a gate from wire p^i_j to p^o_{j+1} for every $j < n - 1$. Note that the order of this last group of wires does not matter. (See Figure 3.12.)

Given the instance $\mathcal{I}$ of SM with n men and n women, define $\mathcal{M} : \{0, 1, *\}^{2n^2} \to \{0, 1, *\}^{2n^2}$ to be the function computed by the preceding circuit construction, where the inputs of $\mathcal{M}$ are those fed into the input wires, and the outputs of $\mathcal{M}$ are those produced by the output wires. Explicitly, if $\mathsf{Pair}(m_j, w_k)$

Men:

a	x	y
b	y	x

Women:

x	a	b
y	a	b

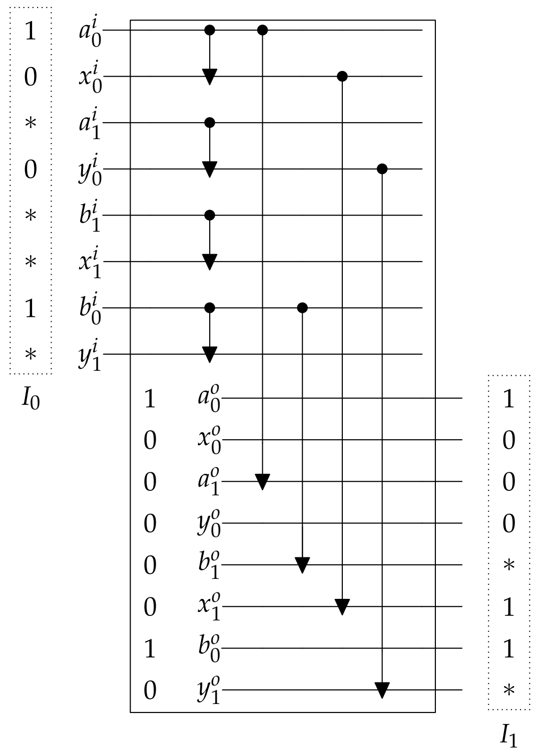

1 0 * 0 * * 1 *
I_0
a^i_0 x^i_0 a^i_1 y^i_0 b^i_1 x^i_1 b^i_0 y^i_1
1 0 0 0 0 0 1 0
a^o_0 x^o_0 a^o_1 y^o_0 b^o_1 x^o_1 b^o_0 y^o_1
1 0 0 0 * 1 1 *
I_1

Figure 3.12

then:

$$m^o_j = \begin{cases} 1 & \text{if } j = 0, \\ m^i_{j-1} \wedge w^i_r & \text{otherwise, where } \mathsf{Pair}(m_{j-1}, w_r). \end{cases}$$

$$w^o_k = \begin{cases} 0 & \text{if } k = 0, \\ w^i_{k-1} \vee m^i_r & \text{otherwise, where } \mathsf{Pair}(m_r, w_{k-1}). \end{cases}$$

We will use the following notation. Any sequence $I \in \{0, 1, *\}^{2n^2}$ can be seen as an input of function $\mathcal{M}$, and thus we write $I(p^i_j)$ to denote the input value of wire p^i_j with respect to I. Similarly, if a sequence $J \in \{0, 1, *\}^{2n^2}$ is an output of $\mathcal{M}$, then we write $J(p^o_j)$ to denote the output value of wire p^o_j.

Let sequence $I_0 \in \{0, 1, *\}^{2n^2}$ be an input of $\mathcal{M}$ defined as follows: $I_0(m^i_0) = 1$ for every man m, and $I_0(w^i_0) = 0$ for every woman w, and $I_0(p^i_j) = *$ for every person p and every j, $1 \leq j < n$. Note that the number of $*$'s in the sequence I_0 is

$$c(n) = 2n^2 - 2n. \tag{3.8}$$

Our version of Subramanian's method [60, 61] consists of computing

$$I_{c(n)} = \mathcal{M}^{c(n)}(I_0),$$

where $\mathcal{M}^d$ simply denotes the d^{th} power of $\mathcal{M}$, i.e. the function we get by composing $\mathcal{M}$ with itself d times. It turns out that $I_{c(n)}$ is a fixed point of $\mathcal{M}$, i.e. $I_{c(n)} = \mathcal{M}(I_{c(n)})$. To show this, we define a sequence I' to be an *extension* of a sequence I if $I(p) = I'(p)$ for every person p such that $I(p) \in \{0,1\}$. In other words, all Boolean values in I are preserved in J, even though some $*$-values in I might be changed to Boolean values in J. We can show that $\mathcal{M}(I)$ is an extension of I for every I which extends I_0, and hence $\mathcal{M}^d(I_0)$ extends I_0 for all d. It follows that $\mathcal{M}^{c(n)}(I_0)$ is a fixed point because there are at most $c(n)$ $*$'s to convert to 0 or 1.

Now we can extract a stable marriage from the fixed point $I_{c(n)}$ by letting B be the sequence obtained by substituting zeros for all remaining $*$-values in $I_{c(n)}$. Then B is also a fixed point of $\mathcal{M}$. A stable marriage can then be extracted from B by announcing the marriage of a man m and a woman w iff $\mathsf{Pair}(m_j, w_k)$ and $B(m^o_j) = 1$ and $B(w^o_k) = 0$. Our goal is to formalize the correctness of this method.

In the example in Figure 3.12, the computation of the fourth power of the function $\mathcal{M}$ can easily be done by using the comparator circuit in Figure 3.13. We show the outputs, produced as a result of the iteration, which comprise the fixed point $I_4 = \mathcal{M}^4(I_0)$ of $\mathcal{M}$. In this case the fixed point consists of Boolean values, where $(I_4(a^o_0), I_4(x^o_0)) = (1,0)$ and $(I_4(b^o_0), I_4(y^o_1)) = (1,0)$. Thus, woman x is married to man a, and woman y is married to man b, which is a stable marriage.

Henceforth, we will let I_ℓ denote the output of $\mathcal{M}^\ell(I_0)$. We want to show that the $c(n)^{\text{th}}$ power of $\mathcal{M} : \{0,1,*\}^{2n^2} \to \{0,1,*\}^{2n^2}$ on input I_0 defined above in fact produces a fixed point of $\mathcal{M}$.

Theorem 44. *(VCC $\vdash$) The three-valued sequence $I_{c(n)} = \mathcal{M}^{c(n)}(I_0)$ is a fixed point of $\mathcal{M}$.*

To prove this theorem, we need a new definition. The proof of Theorem 44 follows from the important observation that every sequence I_q is extended by all sequences I_ℓ, $\ell \geq q$. And thus when going from I_ℓ to $I_{\ell+1}$, the only change that can be made is to change some $*$-values in I_ℓ to Boolean values in $I_{\ell+1}$. But since we started out with at most $c(n)$ $*$-values, we will reach a fixed point in at most $c(n)$ steps. Before proving Theorem 44, we need the following lemma, which says that any given sequence I_q is extended by all sequences I_ℓ, $\ell \geq q$. The lemma can be formulated as the following Σ^B_0 statement.

Lemma 45. *(VCC $\vdash$) For every $q \leq c(n)+1$, for every person p, and for every $k < n$, if we have $I_q(p^i_k) = v \in \{0,1\}$, then $I_\ell(p^i_k) = v$ for all ℓ satisfying $q \leq \ell \leq c(n)+1$.*

Proof of Lemma 45. We prove by Σ^B_0 induction on $q \leq c(n)+1$. The base case ($q = 0$) is easy. With respect to I_0 the only wires having non-$*$ values are p^i_0 for every person p. But the output wires p^o_0 corresponding to these input values are fed these same Boolean values as constant inputs, and these wires are not involved with any comparator gates. Thus, these values will be preserved for every I_ℓ with $q \leq \ell \leq c(n)+1$.

For the induction step, we are given q such that $0 < q \leq c(n)+1$, and assume that $I_q(p^i_k) = v \in \{0,1\}$ for some person p and $k < n$, we want to show that $I_\ell(p^i_k) = v$ for all ℓ satisfying $q \leq \ell \leq c(n)+1$. We will only argue for the case when p is a man m since the case when p is a woman can be argued similarly. We consider two cases. We may have $k = 0$, in which case, as argued in the base case, we have $I_\ell(m^i_0) = 1$ for all ℓ satisfying $q \leq \ell \leq c(n)+1$. Otherwise, we have $k \geq 1$, then since $I_q(m^o_k) = v \in \{0,1\}$, from how $\mathcal{M}$ was constructed, the output wire p^o_k must have got its non-$*$ value v from the wire m^i_{k-1}, which in turn must carry value v before transferring it to wire m^o_k. But then we observe that wire m^i_{k-1} is connected to some wire w^i_r by a comparator gate (i.e. $\mathsf{Pair}(m_{k-1}, w_r)$ holds) before being connected by a comparator gate to m^o_k. Thus, from the definition of three-valued

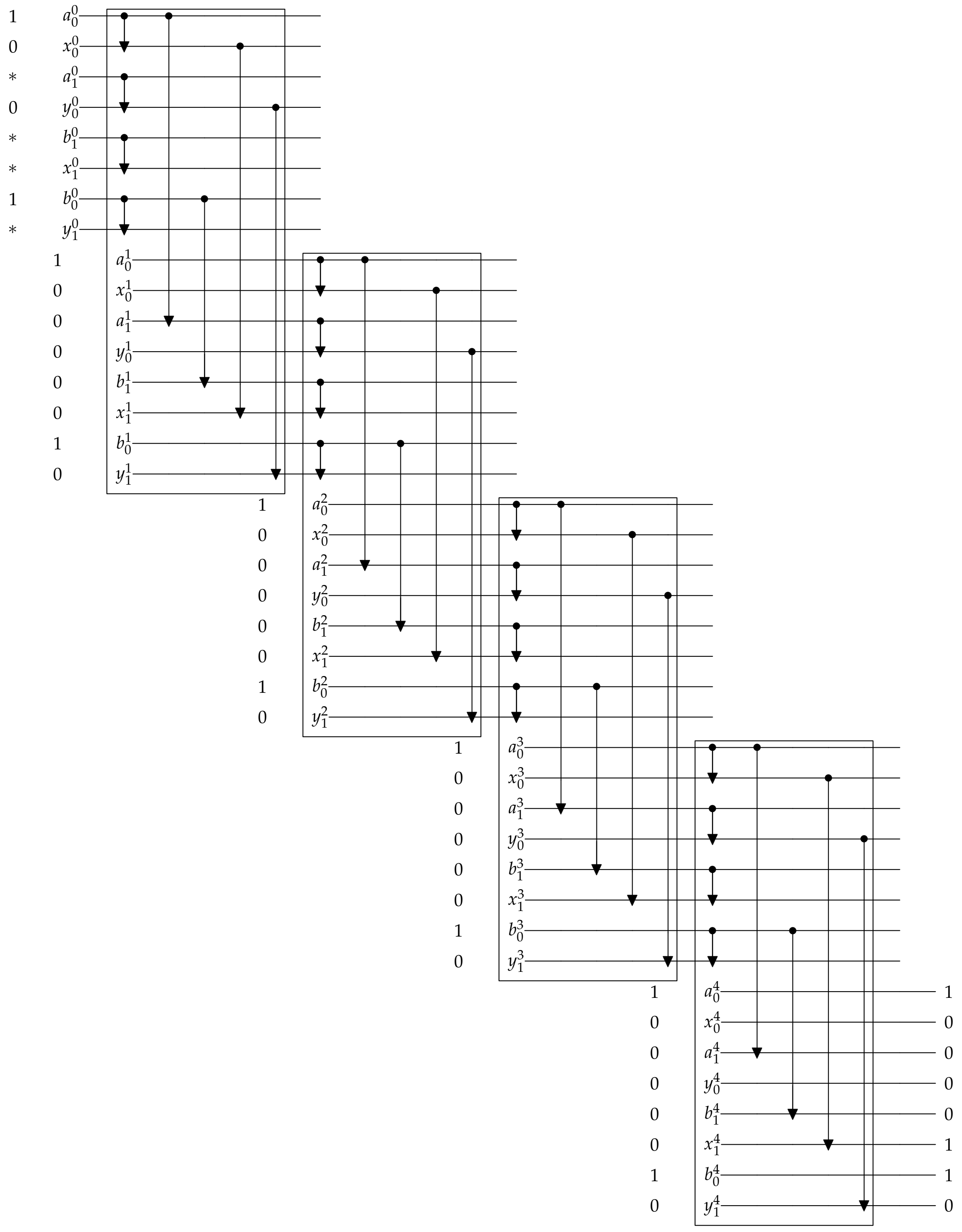


Figure 3.13: The comparator circuit computing the 4th power of $\mathcal{M}$ constructed from the example in Figure 3.12. Since the output wires of the first three blocks serve as input wires to the next block, we use superscripts 1,2,3,4 for these wires on successive blocks, instead of the letter 'o' used in Figure 3.12.

comparator gate, the value v produced on m^i_{k-1} by the gate $\left\langle m^i_{k-1}, w^i_r \right\rangle$ only depends on the non-$*$ value(s) of either $I_{q-1}(m^i_{k-1})$ or $I_{q-1}(w^i_r)$ or both. In any of these cases, by the induction hypothesis, these non-$*$ values of I_{q-1} will be preserved in I_ℓ for all ℓ with $q-1 \le \ell \le c(n)+1$. Hence, we will always get $I_\ell(m^o_k) = v$ for all ℓ satisfying $q \le \ell \le c(n)+1$. □

Proof of Theorem 44. Suppose for a contradiction that for every $\ell \le c(n)$, I_ℓ is not a fixed point. In other words, $I_{\ell+1} = \mathcal{M}(I_\ell) \ne I_\ell$ for all $\ell \le c(n)$. Is is important to note that, by Lemma 45, when going from I_ℓ to $I_{\ell+1}$, we know that $I_{\ell+1}$ extends I_ℓ. Thus, the only change that $\mathcal{M}$ can make at each stage is to switch the $*$-values at some positions in I_ℓ to Boolean values in $I_{\ell+1}$. If $I_{c(n)}$ is not a fixed point, then by utilizing the counting in VTC^0 (Corollary 19), we can show that the number of $*$-values that are switched to Boolean values when going from I_0 to $I_{c(n)+1}$ is at least $c(n)+1$. This is a contradiction, since we started out with only $c(n)$ $*$-values in I_0, and no additional $*$-value was supplied during the iterations of $\mathcal{M}$. The final argument, i.e., the number of $*$-values never increases, can be proved more formally by Σ^B_0-induction on the layers of the comparator circuit computing $\mathcal{M}^{c(n)}(I_0)$. □

Although Theorem 44 gives us a fixed point of $\mathcal{M}$, this fixed point may still be three-valued and thus does not give us all the information needed to extract a stable marriage. However, every three-valued fixed point can easily be extended to a Boolean fixed point as follows. Given a three-valued sequence I, we let $I[* \to v]$ denote the sequence we get by substituting v for all the $*$-values in I.

Proposition 46. *(VCC $\vdash$) If I is a three-valued fixed point of $\mathcal{M}$, then $I[* \to 0]$ and $I[* \to 1]$ are Boolean fixed points of $\mathcal{M}$.*

Proof. Suppose that I is a three-valued fixed point of $\mathcal{M}$. Then when the circuit $C_{\mathcal{I}}$ is presented with input I the output is also I. Without studying the detailed structure of the circuit but just observing that the gates compute monotone functions, by induction on the depth of a gate g in the circuit we can compare the values v and v' of g under the two inputs I and $I[* \to 0]$ as follows: $v' = v$ if $v \in \{0,1\}$ and $v' = 0$ if $v = *$. In particular this is true of the output gates of the circuit. Since I is a fixed point, the output of $C_{\mathcal{I}}$ under input I is I, and so the output under input $I[* \to 0]$ is $I[* \to 0]$. Thus $I[* \to 0]$ is a fixed point of the circuit.

A similar argument works for $I[* \to 1]$ □

To show that the above method for solving SM using three-valued comparator circuits is correct, it remains to justify Subramanian's method for extracting a stable marriage from a Boolean fixed point. Define G to be an AC^0-function, i.e., Σ^B_0-definable, which takes as input a Boolean fixed point B of $\mathcal{M}$, and returns a marriage M such that the pair of man m and woman w is in M iff when j,k are chosen such that $\mathsf{Pair}(m_j, w_k)$ holds, then $B(m^o_j) = 1$ and $B(w^o_k) = 0$. It is worth noting that since $B = \mathcal{M}(B)$, we have $B(p^i_k) = B(p^o_k)$ for every person p and every $k < n$; however, the superscripts o and i are useful for distinguishing between input and output values of the comparator circuit $C_{\mathcal{I}}$ computing $\mathcal{M}$. From the construction of G and the fixed-point property of $\mathcal{M}$, we have the following theorem.

Theorem 47. *(VCC $\vdash$) If B is a Boolean fixed point of $\mathcal{M}$ then $M = G(B)$ is a stable marriage.*

To prove this theorem, we first need to establish the next two lemmas that capture the basic properties of the comparator circuit computing $\mathcal{M}$.

Lemma 48. *(VCC $\vdash$) Let B be any Boolean input to the circuit $C_{\mathcal{I}}$.*

1. *For every man m and every $k < n-1$, if $B(m^i_k) = 1$ then $B(m^o_{k+1}) = 0$ iff $B(w^i_j) = 0$, where w^i_j is the wire that satisfies* $\mathsf{Pair}(m_k, w_j)$.

2. *For every woman w and every $j < n-1$, if $B(w^i_j) = 0$ then $B(w^o_{j+1}) = 1$ iff $B(m^i_k) = 1$, where m^i_k is the wire that satisfies* $\mathsf{Pair}(m_k, w_j)$.

Proof. We will only prove Part 1 since Part 2 can be shown using a dual argument. For the ($\Leftarrow$) direction, we recall that since m^i_k was paired with w^i_j when constructing $C_{\mathcal{I}}$, we have a comparator gate going from m^i_k to w^i_j. Thus, since $B(m^i_k) = 1$ and $B(w^i_j) = 0$, after the comparator gate $\left\langle m^i_k, w^i_j \right\rangle$, the wire m^i_k now carries value zero. But since the output wire $B(m^o_{k+1})$ will carry whatever value forwarded from the wire m^i_k, in this case, we have $B(m^o_{k+1}) = 0$.

For the ($\Rightarrow$) direction, from the construction of $C_{\mathcal{I}}$, the only way that we can change from $B(m^i_k) = 1$ to $B(m^o_{k+1}) = 0$ is by having a comparator gate $\left\langle m^i_k, w^i_j \right\rangle$ connecting m^i_k with some wire w^i_j, and $B(w^i_j) = 0$. □

Lemma 49. *(VCC $\vdash$) Let B be any Boolean fixed point for $\mathcal{M}$.*

1. *For every man m and every $k < n$, if $B(m^i_k) = 1$ and $B(m^i_{k+1}) = 0$, then*
$$B(m^i_0) = \ldots = B(m^i_k) = 1, \qquad\qquad B(m^i_{k+1}) = \ldots = B(m^i_n) = 0.$$

2. *For every woman w and every $j < n$, if $B(w^i_j) = 0$ and $B(m^o_{j+1}) = 1$, then*
$$B(w^i_0) = \ldots = B(w^i_j) = 0, \qquad\qquad B(w^i_{j+1}) = \ldots = B(w^i_n) = 1.$$

Proof. We will only prove Part 1 since Part 2 can be proved by a dual argument. Assume that m is a man and $k < n$ is such that $B(m^i_k) = 1$ and $B(m^i_{k+1}) = 0$. We will use Σ^B_0-*MIN* to choose the least $k_0 \geq 0$ satisfying $B(m^i_{k_0}) = 1$ and $B(m^i_{k_0+1}) = 0$. We can then prove by Σ^B_0 induction on ℓ, $k_0 + 1 \leq \ell < n$, that $B(m^i_\ell) = 0$. The base case when $\ell = k_0 + 1$ trivially holds. For the induction step, by the construction of $C_{\mathcal{I}}$, we observe that when $B(m^i_{\ell-1}) = 0$, then the wire $m^i_{\ell-1}$ will always carry value zero. But since m^o_ℓ will receive whatever value forwarded to it from $m^i_{\ell-1}$, we get $B(m^i_\ell) = B(m^o_\ell) = 0$. Thus, we have just shown that
$$B(m^i_0) = \ldots = B(m^i_{k_0}) = 1, \qquad\qquad B(m^i_{k_0+1}) = \ldots = B(m^i_{n-1}) = 0.$$
But this implies that k_0 is the only subscript at which the elements of the sequence
$$B(m^i_0), B(m^i_2), \ldots, B(m^i_{n-1})$$
change their values from one to zero. Thus, we get $k = k_0$, and we are done. □

From the above two lemmas, we can show that using G we can extract from every Boolean fixed point of $\mathcal{M}$ which extends I_0 a perfect matching.

Lemma 50. *(VCC $\vdash$) If B is a Boolean fixed point of $\mathcal{M}$ then $M = G(B)$ is a perfect matching between the men and women of $\mathcal{I}$.*

Proof. We will only prove that every man is married to a unique woman in M since the claim that every woman is married to a unique man can be shown similarly. Given a man m we want to show that he is married to a unique woman. Since B is a fixed point we know $B(m_0^i) = B(m_0^o) = 1$, and by Lemma 49 the elements of the sequence

$$B(m_0^i), B(m_1^i), \ldots, B(m_{n-1}^i)$$

can only change their values from one to zero at most once. Thus by Lemma 48 and the definition of M, m can marry at most once. It remains for us to show that m indeed gets married. Suppose to the contrary that m remains single in M. Then

$$B(m_0^i) = B(m_1^i) = \ldots = B(m_{n-1}^i) = 1.$$

For every woman w we can choose $k, j < n$ so that $\mathsf{Pair}(m_k, w_j)$ holds, and so by Lemma 48 it follows that $B(w_j^i) = 1$. But $B(w_0^i) = B(w_0^o) = 0$, so the elements of the sequence

$$B(w_0^i), B(w_1^i), \ldots, B(w_{n-1}^i)$$

must change their values from zero to one at least once, and by Lemma 49 they change their values exactly once. Thus by Lemma 48 and the definition of M, every woman is married to exactly one man. Since m was excluded, we have n women paired with at most $n-1$ men, and this contradicts the pigeonhole principle $\mathsf{PHP}(n-1, M)$. □

Proof of Theorem 47. By Lemma 50, we know that M is a perfect matching. Thus it only remains to show that M satisfies the stability condition. Suppose not. Then there exist men a, b and women x, y such that x is married to a and y is married to b, but man a prefers y to x and woman y prefers a to b. Since x is married to a, by how M was constructed and Lemma 49, there are some $k < n$ and $p < n$ such that $\mathsf{Pair}(a_k, x_p)$, and

$$B(a_0^i) = B(a_1^i) = \ldots = B(a_k^i) = 1. \tag{3.9}$$

Similarly, since y is married to b, there are some $\ell < n$ and $q < n$ such that $\mathsf{Pair}(b_\ell, y_q)$, and

$$B(y_0^i) = B(y_1^i) = \ldots = B(y_q^i) = 0. \tag{3.10}$$

Now by the definition of Pair there must be some $s, t < n$ such that $\mathsf{Pair}(a_s, y_t)$ holds. But since man a prefers y to x, and woman y prefers a to b, we have $s < k$ and $t < q$. Thus from (3.9) and (3.10), we get $B(a_s^i) = 1$ and $B(y_t^i) = 0$ respectively. Hence, y is also married to a, and this contradicts Lemma 50. □

Fix a stable marriage instance $\mathcal{I}$ with n men and n women, and let $\mathcal{M}$ be the function computed by the comparator circuit $C_{\mathcal{I}}$. Let Φ_{sm} denote the set of all stable marriages of $\mathcal{I}$, and let Φ_{fxp} denote the set of all Boolean fixed points of $\mathcal{M}$ which extend the input I_0 defined from $\mathcal{I}$. Note that Φ_{sm} and Φ_{fxp} are exponentially large sets, so they are not really objects of our theories. In other words, we write $M \in \Phi_{\mathsf{sm}}$ to denote that M satisfies a formula asserting the stable marriage property, and we write $I \in \Phi_{\mathsf{fxp}}$ to denote that I satisfies a formula asserting that I is a fixed point of $\mathcal{M}$. It was proved in [61] that there is a one-to-one correspondence between Φ_{sm} and Φ_{fxp}, and that the matchings extracted from $I_{c(n)}[* \to 0]$ and $I_{c(n)}[* \to 1]$ are man-optimal and woman-optimal respectively. We now show

how to formalize these results.

We define $F : \Phi_{\mathsf{sm}} \to \Phi_{\mathsf{fxp}}$ to be a function that takes as input a stable marriage M of $\mathcal{I}$, and outputs a sequence $I \in \{0,1\}^{2n^2}$ defined as follows. For every man m and every woman w that are matched in M, if $j, k < n$ are subscripts such that $\mathsf{Pair}(m_j, w_k)$ holds, then we assign

$$I(m_0^i) = \ldots = I(m_j^i) = 1 \text{ and } I(m_{j+1}^i) = \ldots = I(m_{n-1}^i) = 0 \tag{3.11}$$

$$I(w_0^i) = \ldots = I(w_k^i) = 0 \text{ and } I(w_{k+1}^i) = \ldots = I(w_{n-1}^i) = 1. \tag{3.12}$$

From this definition of F, we can prove the following lemma.

Lemma 51. *(VCC $\vdash$) The function $F : \Phi_{\mathsf{sm}} \to \Phi_{\mathsf{fxp}}$ is a bijection.*

We first need to verify that the range of F is indeed contained in Φ_{fxp}.

Lemma 52. *(VCC $\vdash$) If M is a stable marriage, then $I = F(M)$ is a fixed point of $\mathcal{M}$.*

Proof. We start by stating the following:

Claim: For every pair of wires (m_j^i, w_k^i) satisfying $\mathsf{Pair}(m_j, w_k)$, we have $(I(m_j^i), I(w_k^i)) = (1, 0)$ iff man m is matched to woman w in M.

To see that I is a fixed point of $\mathcal{M}$ it suffices to show that equations (3.11) and (3.12) hold with the superscripts i replaced by o. This follows from the Claim and Lemma 48 with $B = I$, together with the observation that for every man m and woman w, the circuit $C_{\mathcal{I}}$ assigns the outputs $m_0^o = 1$ and $w_0^o = 0$.

It remains to prove the Claim. The direction ($\Leftarrow$) follows immediately from the definition of I. To prove the direction ($\Rightarrow$), suppose that for some man m and woman v, $\mathsf{Pair}(m_\ell, v_s)$ holds for some $\ell, s < n$ and $(I(m_\ell^i), I(v_s^i)) = (1, 0)$ but m is not matched to v. Then m is matched to some other woman w, and $\mathsf{Pair}(m_j, w_k)$ holds for some $j, k < n$. Since $I(m_\ell^i) = 1$, it follows from (3.11) that $\ell < j$, so m prefers v to w. Since $I(v_s^i) = 0$, it follows from (3.12) applied to the woman v that v prefers m to the man that she is matched with. Therefore the marriage is not stable. □

Proof of Lemma 51. From Lemma 52, we know that the function F is properly defined. It also follows from how F was defined that two distinct stable marriages will get mapped to distinct fixed points of $\mathcal{M}$, and hence F is injective. It only remains to show that F is surjective. But then it is not hard to check that the function G defined before Theorem 47 is a left inverse of F. □

We next want to show that $G(I_{c(n)}[* \to 0])$ and $G(I_{c(n)}[* \to 1])$ are man-optimal and woman-optimal stable marriages of $\mathcal{I}$ respectively. A technical difficult is that it might be tricky to compare $G(I_{c(n)}[* \to 0])$ and $G(I_{c(n)}[* \to 1])$ with $G(J)$ for some arbitrary Boolean fixed point J of $\mathcal{M}$. However the following lemma shows that every Boolean fixed point of $\mathcal{M}$ is an extension of $I_{c(n)}$, which means that it suffices to work with only Boolean fixed-point extensions of $I_{c(n)}$.

Lemma 53. *(VCC $\vdash$) If J is a Boolean fixed point of $\mathcal{M}$, then J extends I_ℓ for every $\ell \leq c(n)$.*

Proof. We prove by Σ_0^B induction on $\ell \leq c(n)$. Base case ($\ell = 0$): we have $I_0(m_0^o) = 1$ for every man m and $I_0(w_0^o) = 0$ for every woman w. But from how $\mathcal{M}$ was constructed, $\mathcal{M}$ always outputs value one on wire m_0^o for every man m and zero on wire w_0^o for every woman w. Thus since J is a Boolean fixed point of $\mathcal{M}$, we also have $J(m_0^o) = 1$ for every man m and $J(w_0^o) = 1$ for every man w, and hence J extends J_0.

For the induction step, we are given ℓ such that $0 < \ell \leq c(n)$, and assume that J extends I_ℓ. We want to show that for every person p and $k < n$, if $I_\ell(p^i_k) = v \in \{0,1\}$, then $J(p^i_k) = v$. We will only argue for the case when p is a man m since the case when p is a woman can be argued similarly. We consider two cases. We may have $k = 0$, then we can argue as in the base case. Otherwise, we have $k \geq 1$, then since $I_\ell(m^o_k) = v \in \{0,1\}$, from how $\mathcal{M}$ was constructed, the output wire p^o_k must have got its non-$*$ value v from the wire m^i_{k-1}, which in turn must have carried value v before transferring it to wire m^o_k. But then we observe that wire m^i_{k-1} is connected to some wire w^i_r by a comparator gate (i.e. $\mathsf{Pair}(m_{k-1}, w_r)$ holds) before being connected by a comparator gate to m^o_k. Thus from the definition of three-valued comparator gate, the value v produced on m^i_{k-1} by the gate $\left\langle m^i_{k-1}, w^i_r \right\rangle$ only depends on the non-$*$ value(s) of either $I_{m-1}(m^i_{k-1})$ or $I_{q-1}(w^i_r)$ or both. In any of these cases, since J extends $I_{\ell-1}$ (by the induction hypothesis), these non-$*$ values of I_{q-1} will also be contained J. But since $J = \mathcal{M}(J)$, we get $J(m^o_k) = v$. □

From Lemma 51 and Lemma 53, we can prove the following theorem.

Theorem 54. *(VCC $\vdash$) Let M be a stable marriage of the* SM *instance $\mathcal{I}$. Let*

$$M_0 = G(I_{c(n)}[* \to 0]) \qquad\qquad M_1 = G(I_{c(n)}[* \to 1]).$$

Then M_0 and M_1 are stable marriages, and every man gets a partner in M_0 no worse than the one he gets in M, and every woman gets a partner in M_1 no worse than the one she gets in M. In other words, M_0 and M_1 are the man-optimal and woman-optimal solutions respectively.

Proof. We only prove that M_0 is man-optimal since the proof that M_1 is woman-optimal is similar. From Lemma 51 and Lemma 53, if we let $K = F(M)$, then K is a Boolean fixed point of $\mathcal{M}$ extending the three-valued fixed point $I_{c(n)}$ and K uniquely determines M. Suppose for a contradiction that some man m gets a better partner in M than in M_0. Let w and u be the women m marries in M and M_0, and assume that $j, k, \ell, s < n$ are subscripts such that $\mathsf{Pair}(m_j, w_k)$ and $\mathsf{Pair}(m_\ell, u_s)$ hold. For brevity, let $O = I_{c(n)}[* \to 0]$. Then from how M and M_0 are constructed, we have $(K(m^i_j), K(w^i_k)) = (1,0)$ and $(O(m^i_\ell), O(u^i_s)) = (1,0)$. Note that we construct O by substituting zeros for all $*$-values in $I_{c(n)}$, so we must have $I_{c(n)}(m^i_\ell) = 1$ originally. By Lemma 49, we have $O(m^i_0) = \ldots = O(m^i_\ell) = 1$. But since m prefers w to u, we also have $j < \ell$, and hence $O(m^i_j) = 1$. Since we cannot introduce additional ones to $I_{c(n)}$ to get O, we also have $I_{c(n)}(m^i_j) = 1$. We next show the following claim, which will imply a contradiction since m cannot marry both w and u in the stable marriage M_0.

Claim: We must have $O(w^i_k) = 0$.

We cannot have $(I_{c(n)}(m^i_j), I_{c(n)}(w^i_k)) = (1,0)$; otherwise, m has no choice but to marry w in both M and M_0 since both K and O are extensions of $I_{c(n)}$. This forces $I_{c(n)}(w^i_k) = *$. But then since we must substitute zeros for all $*$-values when producing O, we have $O(w^i_k) = 0$. □

Theorem 55. *(VCC $\vdash$)* MOSM *and* WOSM *are* AC^0 *many-one reducible to* CCV¬.

Proof. We will show only the reduction from MOSM to CCV¬ since the reduction from WOSM to CCV¬ works similarly.

Following the above construction, we can write a Σ^B_0-formula defining an AC^0 function that takes as input an instance of MOSM with preference lists for all the men and women, and produces a three-valued comparator circuit that computes the three-valued fixed point $I_{c(n)} = \mathcal{M}^{c(n)}(I_0)$, and then extracts the man-optimal stable marriage from $I_{c(n)}[* \to 0]$. Although the first step of computing $I_{c(n)} = \mathcal{M}^{c(n)}(I_0)$ can easily be done as shown in the example from Figure 3.13, the second step of computing $I_{c(n)}[* \to 0]$ from the output $I_{c(n)}$ and extracting the stable marriage cannot be trivially done using a three-valued comparator circuit. However, we can apply the construction from the proof of Theorem 43 to simulate the three-valued computation of $I_{c(n)} = \mathcal{M}^{c(n)}(I_0)$ using an instance of CCV¬, where we can then utilize the available negation gates, $\wedge$-gates and $\vee$-gates to build the necessary gadget to decide if a designated pair of man and woman are married in the man-optimal stable marriage. The use of negation gates is essential in our construction.

Let $\mathcal{I}$ be an instance of MOSM, where (m, w) is the designated pair of man and woman. Let M denote the man-optimal stable marriage of $\mathcal{I}$. We choose j, k such that $\mathsf{Pair}(m_j, w_k)$ holds. Then we recall that $(m, w) \in M$ iff $I_{c(n)}[* \to 0](m^o_j) = 1$ and $I_{c(n)}[* \to 0](w^o_k) = 0$. Observe that $I_{c(n)}[* \to 0](m^o_j) = 1$ and $I_{c(n)}[* \to 0](w^o_k) = 0$ iff

$$\big(I_{c(n)}(m^o_j), I_{c(n)}(w^o_k)\big) = (1, 0) \;\vee\; \big(I_{c(n)}(m^o_j), I_{c(n)}(w^o_k)\big) = (1, *). \tag{3.13}$$

Let C denote the three-valued circuit computing $I_{c(n)} = \mathcal{M}^{c(n)}(I_0)$. Then (3.13) simply asserts that the wire carrying $I_{c(n)}(m^o_j)$ of C must output 1 and the wire carrying $I_{c(n)}(w^o_k)$ of C must output either 0 or $*$. Let C' be the boolean comparator circuit that simulates the three-valued computation of C using the construction from the proof of Theorem 43, where now we use a pair of wires in C' to simulate each three-valued wire of C. From (3.13) it suffices to check that $I_{c(n)}(m^o_j)$ is coded by $\langle 1, 1\rangle$ and $I_{c(n)}(w^o_k)$ has first component 0 in its code (the two possibilities are $\langle 0, 0\rangle$ and $\langle 0, 1\rangle$). This checking is easily done with comparator gates, together with a negation gate to verify the 0 output. □

Corollary 38 and Theorems 43, 42 and 55 give us the following corollary.

Corollary 56. *(VCC ⊢) The ten problems* MOSM, WOSM, SM, CCV, CCV¬, THREE-VALUED CCV, 3LFMM, LFMM, 3VLFMM *and* VLFMM *are all equivalent under* AC^0 *many-one reductions, where the equivalence of* SM *is with respect to the search problem version of the reduction defined in Definition 2.*

Proof. Corollary 38 and Theorem 43 show that CCV, CCV¬, THREE-VALUED CCV, 3LFMM, LFMM, 3VLFMM and VLFMM are all equivalent under AC^0 many-one reductions.

Theorem 55 shows that MOSM and WOSM are AC^0 many-one reducible to THREE-VALUED CCV. Theorem 42 also shows that 3LFMM is AC^0 many-one reducible to MOSM, WOSM, and SM. Hence, MOSM, WOSM, and SM is equivalent to the above problems under AC^0 many-one reductions. □

Chapter 4

Formalizing randomized matching algorithms

In this chapter, we will analyze in *VPV* two randomized matching algorithms using Jeřábek's framework. The first one is the RNC^2 algorithm for determining whether a bipartite graph has a perfect matching, based on the Schwartz-Zippel Lemma [55, 65] for polynomial identity testing applied to the Edmonds polynomial [18] associated with the graph. The second algorithm, due to Mulmuley, Vazirani and Vazirani [47], is in the function class associated with RNC^2, and uses the Isolating Lemma to find such a perfect matching when it exists.

We will use the method proposed and utilized by Jeřábek in [28, 29] which is based on the following simple observation: if $\mathcal{X}$ and $\mathcal{Y}$ are sets and there is a function F mapping $\mathcal{X}$ onto $\mathcal{Y}$, then the cardinality of $\mathcal{Y}$ is at most the cardinality of $\mathcal{X}$. Thus instead of counting the sets $\mathcal{X}$ and $\mathcal{Y}$ directly, we can compare the sizes of $\mathcal{X}$ and $\mathcal{Y}$ by showing the existence of a surjection F, which in many cases can be easily carried out within weak theories of bounded arithmetic. In this chapter, we will restrict our discussion to the case when the sets are bounded polynomial-time definable sets and the surjections are polytime functions, all of which can be defined within *VPV* or other variants of polytime theories. The following definition is all we need to know about Jeřábek's framework.

Definition 57. Let $\mathcal{X}_n = \{R \in \{0,1\}^n \mid F(R) = 1\}$, where $F(R)$ is a *VPV* function (which may have other arguments) and let s, t be *VPV* terms. Then

$$\mathsf{Pr}_{R<2^n}\left[R \in \mathcal{X}_n\right] \preceq s/t$$

means that either $\mathcal{X}_n$ is empty, or there exists a *VPV* function $G(n, \bullet)$ (which may have other arguments) mapping the set $[s] \times \{0,1\}^n$ onto the set $[t] \times \mathcal{X}_n$.

Since we are not concerned with justifying the above definition, our theorems can be formalized in *VPV* without sWPHP.

The reader is advised to review the notation defined in Section 2.2 since we will use it extensively in this chapter.

4.1 Edmonds' Theorem

Let G be a bipartite graph with two disjoint sets of vertices $U = \{u_1, \ldots, u_n\}$ and $V = \{v_1, \ldots, v_n\}$. We use a pair (i, j) to encode the edge $\{u_i, v_j\}$ of G. Thus the edge relation of the graph G can be encoded by a boolean matrix $E_{n\times n}$, where we define $(i, j) \in E$, i.e. $E(i, j) = 1$, iff $\{u_i, v_j\}$ is an edge of G.

Each perfect matching in G can be encoded by an $n \times n$ permutation matrix M satisfying $M(i, j) \to E(i, j)$ for all $i, j \in [n]$. Recall that a *permutation matrix* is a square boolean matrix that has exactly one entry of value 1 in each row and each column and 0's elsewhere.

Recall that *VPV* extends $\overline{V\#L}$ since matrix powering can easily be carried out in polynomial time, and thus all theorems of $\overline{V\#L}$ from [13, 58] are also theorems of *VPV*. In particular the language of *VPV* contains the function Det, which computes the determinant of integer matrices based on Berkowitz's algorithm. From results in [58] (see page 44 of [29] for a correction) it follows that *VPV* proves the Cayley-Hamilton Theorem, and hence the cofactor expansion formula and other usual properties of determinants of integer matrices.

Let $A_{n\times n}$ be the matrix obtained from G by letting $A_{i,j}$ be an indeterminate $X_{i,j}$ for all $(i, j) \in E$, and let $A_{i,j} = 0$ for all $(i, j) \notin E$. The matrix of indeterminates $A(\vec{X})$ is called the *Edmonds matrix* of G, and $\mathsf{Det}(A(\vec{X}))$ is called the *Edmonds polynomial* of G. In general this polynomial has exponentially many monomials, so for the purpose of proving its properties in *VPV* we consider $\mathsf{Det}(A(\vec{X}))$ to be a function which takes as input an integer matrix $\vec{W}_{n\times n}$ and returns an integer $\mathsf{Det}(A(\vec{W}))$. Thus $\mathsf{Det}(A(\vec{X})) \equiv 0$ means that this function is identically zero.

The following theorem draws an important connection between determinants and matchings. The standard proof uses the *Lagrange expansion* which has exponentially many terms, and hence cannot be formalized in *VPV*. However we will give an alternative proof which can be so formalized.

Theorem 58 (Edmonds' Theorem [18]). *(VPV $\vdash$) Let* $\mathsf{Det}(A(\vec{X}))$ *be the Edmonds polynomial of the bipartite graph G. Then G has a perfect matching if and only if*

$$\mathsf{Det}(A(\vec{X})) \not\equiv 0$$

(i.e. if and only if there exists an integer matrix $\vec{W}$ *such that* $\mathsf{Det}(A(\vec{W})) \neq 0$*).*

Proof. For the direction ($\Rightarrow$) we need the following lemma.

Lemma 59. *(VPV $\vdash$)* $\mathsf{Det}(M) \in \{-1, 1\}$ *for any permutation matrix M.*

Proof of Lemma 59. We will construct a sequence of matrices

$$N_n, N_{n-1}, \ldots, N_1,$$

where $N_n = M$, $N_1 = (1)$, and we construct N_{i-1} from N_i by choosing j_i satisfying $N(i, j_i) = 1$ and letting $N_{i-1} = N_i[i \mid j_i]$.

From the way the matrices N_i are constructed, we can easily show by $\Sigma_0^B(\mathcal{L}_{\mathsf{FP}})$ induction on $\ell = n, \ldots, 1$ that the matrices N_ℓ are permutation matrices. Finally, using the cofactor expansion formula, we prove by $\Sigma_0^B(\mathcal{L}_{\mathsf{FP}})$ induction on $\ell = 1, \ldots, n$ that $\mathsf{Det}(N_\ell) \in \{-1, 1\}$. □

From the lemma we see that if M is the permutation matrix representing a perfect matching of G, then *VPV* proves $\mathsf{Det}(A(M)) = \mathsf{Det}(M) \in \{1, -1\}$, so $\mathsf{Det}(A(\vec{X}))$ is not identically 0.

For the direction ($\Leftarrow$) it suffices to describe a polytime function F that takes as input an integer matrix $B_{n\times n} = A(\vec{W})$, where $A(\vec{X})$ is the Edmonds matrix of a bipartite graph G and $\vec{W}_{n\times n}$ is an integer value assignment, and reason in VPV that if $\mathsf{Det}(B) \neq 0$, then F outputs a perfect matching of G.

Assume $\mathsf{Det}(B) \neq 0$. Note that finding a perfect matching of G is the same as extracting a nonzero diagonal, i.e., a sequence of nonzero entries $B(1,\sigma(1)), B(2,\sigma(2)), \ldots, B(n,\sigma(n))$, where σ is a permutation of the set $[n]$. For this purpose, we construct a sequence of matrices

$$B_n, B_{n-1}, \ldots, B_1,$$

as follows. We let $B_n = B$. For $i = n, \ldots, 2$, we let $B_{i-1} = B_i[i \mid j_i]$ and the index j_i is chosen using the following method. Suppose we already know B_i satisfying $\mathsf{Det}(B_i) \neq 0$. By the cofactor expansion along the last row of B_i,

$$\mathsf{Det}(B_i) = \sum_{j=1}^{i} B_i(i,j)(-1)^{i+j}\mathsf{Det}(B_i[i \mid j]).$$

Thus, since $\mathsf{Det}(B_i) \neq 0$, at least one of the terms in the sum on the right-hand side is nonzero. Thus, we can choose the least index j_i such that $B_i(i,j_i) \cdot \mathsf{Det}(B_i[i \mid j_i]) \neq 0$.

To extract the perfect matching, we let Q be an $n \times n$ matrix, where $Q(i,j) = j$. Then we construct a sequence of matrices

$$Q_n, Q_{n-1}, \ldots, Q_1,$$

where $Q_n = Q$ and $Q_{i-1} = Q_i[i \mid j_i]$, i.e., we delete from Q_i exactly the row and column we deleted from B_i. We define a permutation σ by letting $\sigma(i) = Q_i(i,j_i)$. Then $\sigma(i)$ is the column number in B which corresponds to column j_i in B_i, and the set of edges

$$\big\{(i,\sigma(i)) \mid 1 \leq i \leq n\big\}$$

is our desired perfect matching. □

4.2 The Schwartz-Zippel Lemma

The Schwartz-Zippel Lemma [55, 65] is one of the most fundamental tools in the design of randomized algorithms. The lemma provides us a coRP algorithm for the *polynomial identity testing problem* (PIT): given an arithmetic circuit computing a multivariate polynomial $P(\vec{X})$ over a field $\mathbb{F}$, we want to determine if $P(\vec{X})$ is identically zero. The PIT problem is important since many problems, e.g., primality testing [1], perfect matching [47], and software run-time testing [64], can be reduced to PIT. Moreover, many fundamental results in complexity theory like $\mathsf{IP} = \mathsf{PSPACE}$ [56] and the PCP theorem [4, 5] make heavy use of PIT in their proofs. The Schwartz-Zippel lemma can be stated as follows.

Theorem 60 (Schwartz-Zippel Lemma). *Let $P(X_1, \ldots, X_n)$ be a non-zero polynomial of degree $D \geq 0$ over a field (or integral domain) $\mathbb{F}$. Let S be a finite subset of $\mathbb{F}$ and let $\vec{R}$ denote the sequence $\langle R_1, \ldots, R_n\rangle$. Then*

$$\mathsf{Pr}_{\vec{R}\in S^n}\big[P(\vec{R}) = 0\big] \leq \frac{D}{|S|}.$$

Using this lemma, we have the following coRP algorithm for the PIT problem when $\mathbb{F} = \mathbb{Z}$. Given a polynomial

$$P(X_1, \ldots, X_n)$$

of degree at most D, we choose a sequence $\vec{R} \in [0, 2D)^n$ at random. If P is given implicitly as a circuit, the degree of P might be exponential, and thus the value of $P(\vec{R})$ might require exponentially many bits to encode. In this case we use the method of Ibarra and Moran [26] and let Y be the result of evaluating $P(\vec{R})$ using arithmetic modulo a random integer from the interval $[1, D^k]$ for some fixed k. If $Y = 0$, then we report that $P \equiv 0$. Otherwise, we report that $P \not\equiv 0$. (Note that if P has small degree, then we can evaluate $P(\vec{R})$ directly.)

Unfortunately the Schwartz-Zippel Lemma seems hard to prove in bounded arithmetic. The main challenge is that the degree of P can be exponentially large. Even in the special case when P is given as the symbolic determinant of a matrix of indeterminates and hence the degree of P is small, the polynomial P still has up to $n!$ terms. Thus, we will focus on a much weaker version of the Schwartz-Zippel Lemma that involves only Edmonds' polynomials since this will suffice for us to establish the correctness of a FRNC^2 algorithm for deciding if a bipartite graph has a perfect matching.

4.2.1 Edmonds' polynomials for complete bipartite graphs

In this section we will start with the simpler case when every entry of an Edmonds matrix is a variable, since it clearly demonstrates our techniques. This case corresponds to the Schwartz-Zippel Lemma for Edmonds' polynomials of complete bipartite graphs.

Let A be the full $n \times n$ Edmonds' matrix A, where $A_{i,j} = X_{i,j}$ for all $1 \le i, j \le n$. We consider the case that S is the interval of integers $S = [0, s)$ for $s \in \mathbb{N}$, so $|S| = s$. Then $\mathsf{Det}(A(\vec{X}))$ is a nonzero polynomial of degree exactly n, and we want to show that

$$\mathsf{Pr}_{\vec{r} \in S^{n^2}} \left[\mathsf{Det}(A(\vec{r})) = 0 \right] \preceq \frac{n}{s}.$$

Let

$$\mathcal{Z}(n, s) := \left\{ \vec{r} \in S^{n^2} \mid \mathsf{Det}(A(\vec{r})) = 0 \right\},$$

i.e., the set of zeros of the Edmonds polynomial $\mathsf{Det}(A(\vec{X}))$. Then by Definition 57, it suffices to exhibit a VPV function mapping $[n] \times S^{n^2}$ onto $S \times \mathcal{Z}(n, s)$. For this it suffices to give a VPV function mapping $[n] \times S^{n^2-1}$ onto $\mathcal{Z}(n, s)$. We will define a VPV function

$$F(n, s, \bullet) : [n] \times S^{n^2-1} \twoheadrightarrow \mathcal{Z}(n, s),$$

so $F(n, s, \bullet)$ takes as input a pair $(i, \vec{r})$, where $i \in [n]$ and $\vec{r} \in S^{n^2-1}$ is a sequence of $n^2 - 1$ elements.

Let B be an $n \times n$ matrix with elements from S. For $i \in [n]$ let B_i denote the leading principal submatrix of B that consists of the $i \times i$ upper-left part of B. In other words, $B_n = B$, and $B_{i-1} := B_i[i \mid i]$ for $i = n, \ldots, 2$. The following fact follows easily from the least number principle $\Sigma_0^B(\mathcal{L}_{\mathsf{FP}})$-*MIN*.

Fact 61. *($VPV \vdash$) If* $\mathsf{Det}(B) = 0$*, then there is* $i \in [n]$ *such that* $\mathsf{Det}(B_j) = 0$ *for all* $i \le j \le n$*, and either* $i = 1$ *or* $i > 1$ *and* $\mathsf{Det}(B_{i-1}) \neq 0$*.*

We claim that given $\mathsf{Det}(B) = 0$ and given i as in the fact, the element $B(i, i)$ is uniquely determined by the other elements in B. Thus if $i = 1$ then $B(i, i) = 0$, and if $i > 1$ then by the cofactor expansion of

$\mathsf{Det}(B_i)$ along row i,

$$0 = \mathsf{Det}(B_i) = B_i(i,i) \cdot \mathsf{Det}(B_{i-1}) + \mathsf{Det}(B_i') \tag{4.1}$$

where B_i' is obtained from B_i by setting $B'(i,i) = 0$. This equation uniquely determines $B_i(i,i)$ because $\mathsf{Det}(B_{i-1}) \neq 0$.

The output of $F(n,s,(i,\vec{r}))$ is defined as follows. Let B be the $n \times n$ matrix determined by the $n^2 - 1$ elements in $\vec{r}$ by inserting the symbol $*$ (for unknown) in the position for $B(i,i)$. Try to use the method above to determine the value of $* = B_i(i,i)$, assuming that $*$ is chosen so that $\mathsf{Det}(B) = 0$. This method could fail because $\mathsf{Det}(B_{i-1}) = 0$. In this case, or if the solution to the equation (4.1) gives a value for $B_i(i,i)$ which is not in S, output the default "dummy" zero sequence $\vec{0}_{n\times n}$. Otherwise let C be B with $*$ replaced by the obtained value of $B_i(i,i)$. If $\mathsf{Det}(C) = 0$ then output C, otherwise output the dummy zero sequence.

Theorem 62. *($VPV \vdash$) Let $A(\vec{X})$ be the Edmonds matrix of a complete bipartite graph $K_{n,n}$. Let S denote the set $[0,s)$. Then the function $F(n,s,\bullet)$ defined above is a polytime surjection that maps $[n] \times S^{n^2-1}$ onto $\mathcal{Z}(n,s)$.*

Proof. It is easy to see that $F(n,s,\bullet)$ is polytime (in fact it belongs to the complexity class DET). To see that F is surjective, let C be an arbitrary matrix in $\mathcal{Z}(n,s)$, so $\mathsf{Det}(C) = 0$. Let $i \in [n]$ be determined by Fact 61 when $B = C$. Let $\vec{r}$ be the sequence of $n^2 - 1$ elements consisting of the rows of C with $C(i,i)$ deleted. Then the algorithm for computing $F(n,s,(i,\vec{r}))$ correctly computes the missing element $C(i,i)$ and outputs C. □

4.2.2 Edmonds' polynomials for general bipartite graphs

For general bipartite graphs, an entry of an Edmonds matrix A might be 0, so we cannot simply use leading principal submatrices in our construction of the surjection F. However given a sequence $\vec{W}_{n\times n}$ making $\mathsf{Det}(A(\vec{W})) \neq 0$, it follows from Theorem 58 that we can find a perfect matching M in polytime. Thus, the nonzero diagonal corresponding to the perfect matching M will play the role of the main diagonal in our construction. The rest of the proof will proceed similarly. Thus, we have the following theorem.

Theorem 63. *($VPV \vdash$) There is a VPV function $H(n,s,A,\vec{W},\bullet)$ where $A_{n\times n}$ is the Edmonds matrix for an arbitrary bipartite graph and $\vec{W}$ is a sequence of n^2 (binary) integers, such that if $\mathsf{Det}(A(\vec{W})) \neq 0$ then $H(n,s,A,\vec{W},\bullet)$ maps $[n] \times S^{n^2-1}$ onto $\{\vec{r} \in S^{n^2} \mid \mathsf{Det}(A(\vec{r})) = 0\}$, where $S = [0,s)$.*

In other words, it follows from Definition 57 that the function $H(n,s,A,\vec{W},\bullet)$ in the theorem witnesses that

$$\mathsf{Pr}_{\vec{r}\in S^{n^2}}\left[\mathsf{Det}(A(\vec{r})) = 0\right] \preceq \frac{n}{s}.$$

Proof. Assume $\mathsf{Det}(A(\vec{W})) \neq 0$. Then the polytime function described in the proof of Theorem 58 produces an $n \times n$ permutation matrix M such that for all $i,j \in [n]$, if $M(i,j) = 1$ then the element $A(i,j)$ in the Edmonds matrix A is not zero. We apply the algorithm in the proof of Theorem 62, except that the sequence of principal submatrices of B used in Fact 61 is replaced by the sequence $B_n, B_{n-1}, \ldots, B_1$ determined by M as follows. We let $B_n = B$, and for $i = n, \ldots, 2$ we let $B_{i-1} = B_i[i \mid j_i]$, where the indices j_i are chosen the same way as in the proof of Theorem 58 when constructing the perfect matching M. □

We note that the mapping $H(n, s, A, \bullet)$ in this case may not be in DET since the construction of M depends on the sequential polytime algorithm from Theorem 58 for extracting a perfect matching.

4.2.3 The RNC^2 algorithm for the bipartite perfect matching decision problem

An instance of the *bipartite perfect matching decision* problem is a bipartite graph G encoded by a matrix $E_{n\times n}$, and we are to decide if G has a perfect matching. Here is an RDET algorithm for the problem. The algorithm is essentially due to Lovász [39]. From E, construct the Edmonds matrix $A(\vec{X})$ for G and choose a random sequence $\vec{r}_{n\times n} \in [2n]^{n^2}$. If $\mathsf{Det}(A(\vec{r})) \neq 0$ then we report that G has a perfect matching. Otherwise, we report G does not have a perfect matching.

We claim that VPV proves correctness of this algorithm. The correctness assertion states that if G has a perfect matching then the algorithm reports NO with probability at most $1/2$, and otherwise it certainly reports NO. Theorem 58 shows that VPV proves the latter. Conversely, if G has a perfect matching given by a permutation matrix M then the function $H(n, 2n, A, M, \bullet)$ of Theorem 63 witnesses that the probability of $\mathsf{Det}(A(\vec{r})) = 0$ is at most $1/2$, according to Definition 57, where A is the Edmonds matrix for G. Hence VPV proves the correctness of this case too.

Since $\mathsf{RDET} \subseteq \mathsf{FRNC}^2$, this algorithm (which solves a decision problem) is also an RNC^2 algorithm.

4.3 Formalizing the Hungarian algorithm

The Hungarian algorithm is a combinatorial optimization algorithm which solves the *maximum-weight bipartite matching* problem in polytime and anticipated the later development of the powerful *primal-dual method*. The algorithm was developed by Kuhn [35], who gave the name "Hungarian method" since it was based on the earlier work of two Hungarian mathematicians: D. Kőnig and J. Egerváry. Munkres later reviewed the algorithm and showed that it is indeed polytime [48]. Although the Hungarian algorithm is interesting by itself, we formalize the algorithm since we need it in the VPV proof of the Isolating Lemma for perfect matchings in Section 4.4.1.

The Hungarian algorithm finds a maximum-weight matching for any weighted bipartite graph. The algorithm and its correctness proof are simpler if we make the two following changes. First, since edges with negative weights can never be in a maximum-weight matching, and thus can be safely deleted, we can assume that every edge has *nonnegative* weight. Second, by assigning zero weight to every edge not present, we only need to consider weighted *complete* bipartite graphs.

Let $G = (X \uplus Y, E)$ be a complete bipartite graph, where $X = \{x_i \mid 1 \le i \le n\}$ and $Y = \{y_i \mid 1 \le i \le n\}$, and let $\vec{w}$ be an integer weight assignment to the edges of G, where $w_{i,j} \ge 0$ is the weight of the edge $\{x_i, y_j\} \in E$.

A pair of integer sequences $\vec{u} = \langle u_i \rangle_{i=1}^n$ and $\vec{v} = \langle v_i \rangle_{i=1}^n$ is called a *weight cover* if

$$\forall i, j \in [n],\ w_{i,j} \le u_i + v_j. \tag{4.2}$$

The *cost* of a cover is $\mathsf{cost}(\vec{u}, \vec{v}) := \sum_{i=1}^n (u_i + v_i)$. We also define $w(M) := \sum_{(i,j)\in M} w_{i,j}$. The Hungarian algorithm is based on the following important observation.

Lemma 64. *($VPV \vdash$) For any matching M and weight cover $(\vec{u}, \vec{v})$, we have $w(M) \le \mathsf{cost}(\vec{u}, \vec{v})$.*

Proof. Since the edges in a matching M are disjoint, summing the constraints $w_{i,j} \leq u_i + v_j$ over all edges of M yields $w(M) \leq \sum_{(i,j)\in M}(u_i + v_j)$. Since no edge has negative weight, we have $u_i + v_j \geq 0$ for all $i, j \in [n]$. Thus,

$$w(M) \leq \sum_{(i,j)\in M} (u_i + v_j) \leq \mathsf{cost}(\vec{u}, \vec{v})$$

for every matching M and every weight cover $(\vec{u}, \vec{v})$. □

Given a weight cover $(\vec{u}, \vec{v})$, the *equality subgraph* $H_{\vec{u},\vec{v}}$ is the subgraph of G whose vertices are $X \uplus Y$ and whose edges are precisely those $\{x_i, y_j\} \in E$ satisfying $w_{i,j} = u_i + v_j$.

Theorem 65. *($VPV \vdash$) Let $H = H_{\vec{u},\vec{v}}$ be the equality subgraph, and let M be a maximum cardinality matching of H. Then the following three statements are equivalent*

1. *$w(M) = \mathsf{cost}(\vec{u}, \vec{v})$.*
2. *M is a maximum-weight matching of G and $(\vec{u}, \vec{v})$ is a minimum-weight cover of G.*
3. *M is a perfect matching of the equality subgraph H.*

(The reader is referred to (A.3) for the full proof of this theorem.)

Below we give a simplified version of the Hungarian algorithm which runs in polynomial time when the edge weights are small (i.e. presented in unary notation). The correctness of the algorithm easily follows from Theorem 65.

Algorithm 66 (The Hungarian algorithm)**.** We start with an arbitrary weight cover $(\vec{u}, \vec{v})$ with small weights: e.g. let

$$u_i = \max\{w_{i,j} \mid 1 \leq j \leq n\}$$

and $v_i = 0$ for all $i \in [n]$. If the equality subgraph $H_{\vec{u},\vec{v}}$ has a perfect matching M, we report M as a maximum-weight matching of G. Otherwise, change the weight cover $(\vec{u}, \vec{v})$ as follows. Since the maximum matching M is not a perfect matching of H, the Hall's condition fails for H. Thus it is not hard (cf. Corollary 121 from (A.2)) to construct in polytime a subset $S \subseteq X$ satisfying $|N(S)| < |S|$, where $N(S)$ denotes the neighborhood of S. Hence we can calculate the quantity

$$\delta = \min\{u_i + v_j - w_{i,j} \mid x_i \in S \ \wedge\ y_j \notin N(S)\},$$

and decrease u_i by δ for all $x_i \in S$ and increase v_j by δ for all $y_j \in N(S)$ without violating the weight cover property (4.2). This strictly decreases the sum $\sum_{i=1}^{n}(u_i + v_i)$. Thus this process can only repeat at most as many time as the initial cost of the cover $(\vec{u}, \vec{v})$. Assuming that all edge weights are small (i.e. presented in unary), the algorithm terminates in polynomial time. Finally we get an equality subgraph $H_{\vec{u},\vec{v}}$ containing a perfect matching M, which by Theorem 65 is also a maximum-weight matching of G.

When formalizing the Isolating Lemma for bipartite matchings, we need a VPV function Mwpm that takes as inputs an edge relation $E_{n\times n}$ of a bipartite graph G and a nonnegative weight assignment $\vec{w}$ to the edges in E, and outputs a minimum-weight perfect matching if such a matching exists, or outputs $\varnothing$ to indicate that no perfect matching exists. Recall that the Hungarian algorithm returns a maximum-weight matching, and not a *minimum*-weight *perfect* matching. However we can use the Hungarian algorithm to compute $\mathsf{Mwpm}(n, E, \vec{w})$ as follows.

Algorithm 67 (Finding a minimum-weight perfect matching)**.**

```
1: Let $c = n \cdot \max\{w_{i,j} \mid (i,j) \in E\} + 1$.
2: Construct the sequence $\vec{w}'$ as follows
   $$w'_{i,j} = \begin{cases} c - w_{i,j} & \text{if } (i,j) \in E \\ 0 & \text{otherwise.} \end{cases}$$
3: Run the Hungarian algorithm on the complete bipartite graph $K_{n,n}$ with weight assignment $\vec{w}'$ to
   get a maximum-weight matching $M$.
4: if $M$ contains an edge that is not in $E$ then
5:     return the empty matching $\varnothing$
6: else
7:     return $M$
8: end if
```

Note that since we assign zero weights to the edges not present and very large weights to other edges, the Hungarian algorithm will always prefer the edges that are present in the original bipartite graph. More formally for any perfect matching M and non-perfect matching N we have

$$\begin{aligned} w'(M) &\geq nc - n \cdot \max\{w_{i,j} \mid (i,j) \in E\} \\ &= (n-1)c + \big(c - n \cdot \max\{w_{i,j} \mid (i,j) \in E\}\big) \\ &= (n-1)c + 1 \\ &> (n-1)c \\ &\geq w'(N) \end{aligned}$$

The last inequality follows from the fact that $w'_{i,j} \leq c$ for all $(i,j) \in N$. Thus, if the Hungarian algorithm returns a matching M with at least one edge not in E, then the original graph cannot have a perfect matching. Also from the way the weight assignment $\vec{w}'$ was defined, every maximum-weight perfect matching of $K_{n,n}$ with weight assignment $\vec{w}'$ is a minimum-weight matching of the original bipartite graph.

It is straightforward to check that the above argument can be formalized in *VPV*, so *VPV* proves the correctness of Algorithm 67 for computing the function Mwpm.

4.4 FRNC^2 algorithm for finding a bipartite perfect matching

Below we recall the elegant FRNC^2 (or more precisely RDET) algorithm due to Mulmuley, Vazirani and Vazirani [47] for finding a bipartite perfect matching. Although the original algorithm works for general undirected graphs, we will only focus on bipartite graphs in this thesis.

Let G be a bipartite graph with two disjoint sets of vertices $U = \{u_1, \dots, u_n\}$ and $V = \{v_1, \dots, v_n\}$. We first consider the *minimum-weight bipartite perfect matching problem*, where each edge $(i,j) \in E$ is assigned an integer weight $w_{i,j} \geq 0$, and we want to a find a minimum-weight perfect matching of G. It turns out there is a DET algorithm for this problem under two assumptions: the weights must be polynomial in n, and the minimum-weight perfect matching must be *unique*. We let $A(\vec{X})$ be an

Edmonds matrix of the bipartite graph. Replace $X_{i,j}$ with $W_{i,j} = 2^{w_{i,j}}$ (this is where we need the weights to be small). We then compute $\mathsf{Det}(A(\vec{W}))$ using Berkowitz's FNC^2 algorithm. Assume that there exists exactly one (unknown) minimum-weight perfect matching M. We will show in Theorem 72 that $w(M)$ is exactly the position of the least significant 1-bit, i.e., the number of trailing zeros, in the binary expansion of $\mathsf{Det}(A(\vec{W}))$. Once having $w(M)$, we can test if an edge $(i,j) \in E$ belongs to the unique minimum-weight perfect matching M as follows. Let w' be the position of the least significant 1-bit of $\mathsf{Det}(A[i \mid j](\vec{W}))$. We will show in Theorem 73 that the edge (i,j) is in the perfect matching if and only if w' is precisely $w(M) - w_{i,j}$. Thus, we can test all edges in parallel. Note that up to this point, everything can be done in $\mathsf{DET} \subseteq \mathsf{FNC}^2$ since the most expensive operation is the Det function, which is complete for DET.

What we have so far is that, assuming that the minimum-weight perfect matching exists and is unique, there is a DET algorithm for finding this minimum-weight perfect matching. But how do we guarantee that if a minimum-weight perfect matching exists, then it is unique? It turns out that we can assign every edge $(u_i, v_j) \in E$ a random weight $w_{i,j} \in [2m]$, where $m = |E|$, and use the Isolating Lemma [47] to ensure that the graph has a *unique* minimum-weight perfect matching with probability at least $1/2$.

The $\mathsf{RDET} \subseteq \mathsf{FRNC}^2$ algorithm for finding a perfect matching is now complete: assign random weights to the edges, and run the DET algorithm for the unique minimum-weight perfect matching problem. If a perfect matching exists, with probability at least $1/2$, this algorithm returns a perfect matching.

4.4.1 Isolating a perfect matching

We will recall the Isolating Lemma [47], the key ingredient of Mulmuley-Vazirani-Vazirani FRNC^2 algorithm for finding a perfect matching. Let X be a set with m elements $\{a_1, \ldots, a_m\}$ and let $\mathcal{F}$ be a family of subsets of X. We assign a weight $w_i \geq 0$ to each element $a_i \in X$, and define the weight of a set $Y \in \mathcal{F}$ to be $w(Y) := \sum_{a_i \in Y} w_i$. Let *minimum-weight* be the minimum of the weights of all the sets in $\mathcal{F}$. Note that several sets of $\mathcal{F}$ might achieve minimum-weight. However, if minimum-weight is achieved by a unique $Y \in \mathcal{F}$, then we say that the weight assignment $\vec{w} = \langle w_i \rangle_{i=1}^m$ is *isolating* for $\mathcal{F}$. (Every weight assignment is isolating if $|\mathcal{F}| \leq 1$.)

Theorem 68 (Isolating Lemma [47])**.** *Let $\mathcal{F}$ be a family of subsets of an m-element set $X = \{a_1, \ldots, a_m\}$. Let $\vec{w} = \langle w_i \rangle_{i=1}^m$ be a random weight assignment to the elements in X. Then for all $k \geq 1$*

$$\mathsf{Pr}_{\vec{w} \in [k]^m}[\vec{w} \textit{ is not isolating for } \mathcal{F}] \leq \frac{m}{k}.$$

To formalize the Isolating Lemma in *VPV* it seems natural to present the family $\mathcal{F}$ by a polytime algorithm. This is difficult to do in general (see Remark 70 below), so we will formalize a special case which suffices to formalize the FRNC^2 algorithm for finding a bipartite perfect matching. Thus we are given a bipartite graph G, and the family $\mathcal{F}$ is the set of perfect matchings of G. We want to show that if we assign random weights to the edges, then the probability that this weight assignment does not isolate a perfect matching is small. Note that although the family $\mathcal{F}$ here might be exponentially large, $\mathcal{F}$ is polytime definable, since recognizing a perfect matching is easy.

Theorem 69 (Isolating a Perfect Matching). *($VPV \vdash$) Let $\mathcal{F}$ be the family of perfect matchings of a bipartite graph G with edges $E = \{e_1, \dots, e_m\}$. Let $\vec{w}$ be a random weight assignment to the edges in E. Then*

$$\mathsf{Pr}_{\vec{w} \in [k]^m} \left[\vec{w} \textit{ is not isolating for } \mathcal{F} \right] \preceq m/k.$$

For brevity, we will call a weight assignment $\vec{w}$ "bad" if $\vec{w}$ is not isolating for $\mathcal{F}$. Let

$$\mathcal{B} := \left\{ \vec{w} \in [k]^m \mid \vec{w} \text{ is bad for } \mathcal{F} \right\}.$$

Then to prove Theorem 69, it suffices to construct a *VPV* function mapping $[m] \times [k]^{m-1}$ onto $\mathcal{B}$. Note that the upper bound m/k is independent of the size n of the two vertex sets. The set $\mathcal{B}$ is polytime definable since $\vec{w} \in \mathcal{B}$ iff

$$\exists i, j \in [n] \left(\begin{array}{c} E(i,j) \text{ and } M(i,j) \text{ and } \neg M'(i,j) \text{ and } M, M' \\ \text{encode two perfect matchings with the same weight} \end{array} \right),$$

where M denotes the output produced by applying the Mwpm function (Algorithm 67) on G, and M' denotes the output produced by applying Mwpm on the graph obtained from G by deleting the edge (i,j).

Proof of Theorem 69. By Definition 57 we may assume that $\mathcal{B}$ is nonempty, so there is an element $\delta \in \mathcal{B}$. (We will use δ as a "dummy" element.) It suffices for us to construct explicitly a *VPV* function F mapping $[m] \times [k]^{m-1}$ onto $\mathcal{B}$. For each $i \in [k]$ we interpret the set $\{i\} \times [k]^{m-1}$ as the set of all possible weight assignments to the $m-1$ edges $E \setminus \{e_i\}$. Our function F will map each set $\{i\} \times [k]^{m-1}$ onto the set of those bad weight assignments $\vec{w}$ such that the graph G contains two *distinct* minimum-weight perfect matchings M and M' with $e_i \in M \setminus M'$.

The function F takes as input a sequence

$$\langle i, w_1, \dots, w_{i-1}, w_{i+1}, \dots, w_m \rangle$$

from $[m] \times [k]^{m-1}$ and does the following. Use the function Mwpm (defined by Algorithm 67) to find a minimum-weight perfect matching M' of G with the edge e_i deleted. Use Mwpm to find a minimum-weight perfect matching M_1 of the subgraph $G \setminus \{u_j, v_\ell\}$, where u_j and v_ℓ are the two endpoints of e_i. If both perfect matchings M' and M_1 exist and satisfy $w(M') - w(M_1) \in [k]$, then F outputs the sequence

$$\left\langle w_1, \dots, w_{i-1}, w(M') - w(M_1), w_{i+1}, \dots, w_m \right\rangle. \tag{4.3}$$

Otherwise F outputs the dummy element δ of $\mathcal{B}$.

Note that if both M' and M_1 exist, then (4.3) is a bad weight assignment, since M' and $M = M_1 \cup \{e_i\}$ are distinct minimum-weight perfect matchings of G under this assignment.

To show that F is surjective, consider an arbitrary bad weight assignment $\vec{w} = \langle w_i \rangle_{i=1}^m \in \mathcal{B}$. Since $\vec{w}$ is bad, there are two distinct minimum-weight perfect matchings M and M' and some edge $e_i \in M \setminus M'$. Thus from how F was defined,

$$\langle i, w_1, \dots, w_{i-1}, w_{i+1}, \dots, w_m \rangle \in [m] \times [k]^{m-1}$$

is an element that gets mapped to the bad weight assignment $\vec{w}$. $\square$

Remark 70. The above proof uses the fact that there is a polytime algorithm for finding a minimum-weight perfect matching (when one exists) in an edge-weighted bipartite graph. This suggests limitations on formalizing a more general version of Theorem 69 in *VPV*. For example, if $\mathcal{F}$ is the set of Hamiltonian cycles in a complete graph, then finding a minimum weight member of $\mathcal{F}$ is NP hard.

4.4.2 Extracting the unique minimum-weight perfect matching

Let G be a bipartite graph and assume that G has a perfect matching. Then in Section 4.4.1 we formalized a version of the Isolating Lemma, which with high probability gives us a weight assignment $\vec{w}$ for which G has a unique minimum-weight perfect matching. This is the first step of the Mulmuley-Vazirani-Vazirani algorithm. Now we proceed with the second step, where we need to output this minimum-weight perfect matching using a DET function.

Let B be the matrix we get by substituting $W_{i,j} = 2^{w_{i,j}}$ for each nonzero entry (i, j) of the Edmonds matrix A of G. We want to show that if M is the unique minimum weight perfect matching of G with respect to $\vec{w}$, then the weight $w(M)$ is exactly the position of the least significant 1-bit in the binary expansion of $\mathsf{Det}(B)$. The usual proof of this fact is not hard, but it uses properties of the Lagrange expansion for the determinant, which has exponentially many terms and hence cannot be formalized in *VPV*. Our proof avoids using the Lagrange expansion, and utilizes properties of the cofactor expansion instead.

Lemma 71. *($VPV \vdash$) There is a VPV function that takes as inputs an $n \times n$ Edmonds' matrix A and a weight sequence*

$$\vec{W} = \left\langle W_{i,j} = 2^{w_{i,j}} \mid 1 \leq i, j \leq n \right\rangle .$$

And if $B = A(\vec{W})$ satisfies $\mathsf{Det}(B) \neq 0$ and p is the position of the least significant 1-bit of $\mathsf{Det}(B)$, then the VPV function outputs a perfect matching M of weight at most p.

It is worth noting that the lemma holds regardless of whether or not the bipartite graph corresponding to A and weight assignment $\vec{W}$ has a unique minimum-weight perfect matching.

The proof of Lemma 71 is very similar to that of Theorem 58. Recall that in Theorem 58, given a matrix B satisfying $\mathsf{Det}(B) \neq 0$, we want to extract a nonzero diagonal of B. In this lemma, we are given the position p of the least significant 1-bit of $\mathsf{Det}(B)$, and we want to get a nonzero diagonal of B whose product has the least significant 1-bit at position at most p. For this, we can use the same method of extracting the nonzero diagonal from Theorem 58 with the following modification. When choosing a term of the Lagrange expansion on the recursive step, we will also need to make sure the chosen term produces a nonzero sub-diagonal of B that will not contribute too much weight to the diagonal we are trying to extract. This ensures that the least significant 1-bit of the weight of the chosen diagonal is at most p.

For the rest of this section, we define $\mathsf{numz}(Y)$ to be the position of the least significant 1-bit of the binary string Y. Thus if $\mathsf{numz}(Y) = q$ then $Y = \pm 2^q Z$ for some positive odd integer Z.

Proof of Lemma 71. We construct a sequence of matrices

$$B_n, B_{n-1}, \ldots, B_1$$

where $B_n = B$ and $B_{i-1} = B_i[i \mid j_i]$ for $i = n \ldots, 2$ where the index j_i is chosen as follows. Define

$$T_i := \left(\prod_{\ell=i+1}^{n} B_\ell(\ell, j_\ell) \right) \mathsf{Det}(B_i).$$

Assume we are given $j_n, \ldots, j_{i+1}$ such that $\mathsf{numz}(T_i) \leq p$. We want to choose j_i such that $\mathsf{numz}(T_{i-1}) \leq p$, where by definition

$$T_{i-1} = \left(\prod_{\ell=i}^{n} B_\ell(\ell, j_\ell) \right) \mathsf{Det}(B_{i-1}) = \left(\prod_{\ell=i}^{n} B_\ell(\ell, j_\ell) \right) \mathsf{Det}(B_i[i \mid j_i]).$$

This can be done as follows. From the cofactor expansion of $\mathsf{Det}(B_i)$, we have

$$T_i = \sum_{j=1}^{i} (-1)^{i+j} \left(\prod_{\ell=i+1}^{n} B_\ell(\ell, j_\ell) \right) B_i(i,j) \mathsf{Det}(B_i[i \mid j]).$$

Since $\mathsf{numz}(T_i) \leq p$, at least one of the terms in the sum must have its least significant 1-bit at position at most p. Thus, we can choose j_i such that

$$\mathsf{numz} \left(\left(\prod_{\ell=i+1}^{n} B_\ell(\ell, j_\ell) \right) B_i(i, j_i) \mathsf{Det}(B_i[i \mid j_i]) \right) = \mathsf{numz}(T_{i-1})$$

is minimized, which guarantees that $\mathsf{numz}(T_{i-1}) \leq p$.

Since by assumption $\mathsf{numz}(T_n) = \mathsf{numz}(\mathsf{Det}(B_n)) = p$, *VPV* proves by $\Sigma_0^B(\mathcal{L}_{\mathsf{FP}})$ induction on $i = n, \ldots, 1$ that

$$\mathsf{numz}(T_i) \leq p.$$

If we define $j_1 = 1$, then when $i = 1$ we have

$$T_1 = \left(\prod_{\ell=2}^{n} B_\ell(\ell, j_\ell) \right) \mathsf{Det}(B_1) = \prod_{\ell=1}^{n} B_\ell(\ell, j_\ell).$$

Thus it follows that $\mathsf{numz}(T_1) = \mathsf{numz}\left(\prod_{\ell=1}^{n} B_\ell(\ell, j_\ell)\right) \leq p$.

Similarly to the proof of Theorem 58, we can extract a perfect matching with weight at most p by letting Q be a matrix, where $Q(i,j) = j$ for all $i, j \in [n]$. Then we compute another sequence of matrices

$$Q_n, Q_{n-1}, \ldots, Q_1,$$

where $Q_n = Q$ and $Q_{i-1} = Q_i[i \mid j_i]$, i.e., we delete from Q_i exactly the row and column we deleted from B_i.

To prove that $M = \{(\ell, Q_\ell(\ell, j_\ell)) \mid 1 \leq \ell \leq n\}$ is a perfect matching, we note that whenever a pair (i,k) is added to the matching M, we delete the row i and column j_i, where j_i is the index satisfying $Q_i(i, j_i) = k$. So we can never match any other vertex to k again.

It remains to show that $w(M) \leq p$. Since

$$\prod_{\ell=1}^{n} B_\ell(\ell, j_\ell) = 2^{w(M)},$$

the binary expansion of $\prod_{\ell=1}^{n} B_\ell(\ell, j_\ell)$ has a unique one at position $w(M)$ and zeros elsewhere. Thus it follows from how the matching M was constructed that $w(M) \leq p$. □

The next two theorems complete our description and justification of our RDET algorithm for finding a perfect matching. For these theorems we are given a bipartite graph $G = (U \uplus V, E)$, where we have $U = \{u_1, \ldots, u_n\}$ and $V = \{v_1, \ldots, v_n\}$, and each edge $(i, j) \in E$ is assigned a weight $w_{i,j}$ such that G has a *unique* minimum-weight perfect matching (see Theorem 69). Let $\vec{W}_{n\times n}$ be a sequence satisfying $W_{i,j} = 2^{w_{i,j}}$ for all $(i, j) \in E$. Let A be the Edmonds matrix of G, and let $B = A(\vec{W})$. Let M denote the unique minimum weight perfect matching of G.

Theorem 72. *($VPV \vdash$) The weight $p = w(M)$ is exactly* $\mathsf{numz}(\mathsf{Det}(B))$.

If in Lemma 71 we tried to extract an appropriate nonzero diagonal of B using the determinant and minors of B as our guide, then in the proof of this theorem we do the reverse. From a minimum-weight perfect matching M of G, we want to rebuild in polynomially many steps suitable minors of B until we fully recover the determinant of B. We can then prove by $\Sigma_0^B(\mathcal{L}_{\mathsf{FP}})$ induction that in every step of this process, each "partial determinant" of B has the least significant 1-bit at position p. Note that the technique we used to prove this theorem does have some similarity to that of Lemma 59, even though the proof of this theorem is more complicated.

Proof. Let Q be a matrix, where $Q(i, j) = j$ for all $i, j \in [n]$. For $1 \leq i \leq n$ let B_i be the result of deleting rows $i + 1, \ldots, n$ and columns $M(i + 1), \ldots, M(n)$ from B and let Q_i be Q with the same rows and columns deleted. We can construct these matrices inductively in the form of two matrix sequences

$$B_n, B_{n-1}, \ldots, B_1 \qquad\qquad Q_n, Q_{n-1}, \ldots, Q_1$$

where

- we let $B_n = B$ and $Q_n = Q$, and
- for $i = n, n - 1, \ldots, 2$, define j_i to be the unique index satisfying

$$M(i) = Q_i(i, j_i),$$

  and then let $B_{i-1} = B_i[i \mid j_i]$ and $Q_{i-1} = Q_i[i \mid j_i]$.

Then (setting $j_1 = 1$)

$$B_i(i, j_i) = B(i, M(i)) = 2^{w_{i,M(i)}},\ 1 \leq i \leq n \tag{4.4}$$

Claim: $\mathsf{numz}(\mathsf{Det}(B_i)) = \sum_{\ell=1}^{i} w_{\ell,M(\ell)}$ for all $i \in [n]$.

The theorem follows from this by setting $i = n$. We will prove the claim by induction on i. The base case $i = 1$ follows from (4.4).

For the induction step, it suffices to show

$$\mathsf{numz}(\mathsf{Det}(B_{i+1})) = \mathsf{numz}(\mathsf{Det}(B_i)) + w_{i+1,M(i+1)}$$

From the cofactor expansion formula we have

$$\mathsf{Det}(B_{i+1}) = \sum_{j=1}^{i+1}(-1)^{(i+1)+j}B_{i+1}(i+1,j)\mathsf{Det}(B_{i+1}[i+1 \mid j])$$

Since $B_{i+1}(i+1, j_{i+1}) = 2^{w_{i+1,M(i+1)}}$ by (4.4), and $\mathsf{Det}(B_{i+1}[i+1 \mid j_{i+1}]) = \mathsf{Det}(B_i)$, it suffices to show that if

$$R := B_{i+1}(i+1, j_{i+1})\mathsf{Det}(B_{i+1}[i+1 \mid j_{i+1}])$$

then

$$\mathsf{numz}(R) < \mathsf{numz}\big(B_{i+1}(i+1,j)\mathsf{Det}(B_{i+1}[i+1 \mid j])\big)$$

for all $j \neq j_{i+1}$.

Suppose for a contradiction that there is some $j' \neq j_\ell$ such that

$$\mathsf{numz}\big(B_{i+1}(i+1,j')\mathsf{Det}(B_{i+1}[i+1 \mid j'])\big) \leq \mathsf{numz}(R).$$

Then, we can extend the set of edges

$$\big\{(n, M(n)), \ldots, (i+2, M(i+2)), (i+1, j')\big\}$$

with i edges extracted from $B_{i+1}[i+1 \mid j']$ (using the method from Lemma 71) to get a perfect matching of G with weight at most p, which contradicts that M is the unique minimum-weight perfect matching of G. □

To extract the edges of M in DET, we need to decide if an edge (i, j) belongs to the unique minimum-weight perfect matching M without knowledge of other edges in M. The next theorem, whose proof follows directly from Lemma 71 and Theorem 72, gives us that method.

Theorem 73. *($VPV \vdash$) For every edge $(i,j) \in E$, we have $(i,j) \in M$ if and only if*

$$w(M) - w_{i,j} = \mathsf{numz}\big(\mathsf{Det}(B[i \mid j])\big).$$

Proof. ($\Rightarrow$): Assume $(i,j) \in M$. Then the bipartite graph $G' = G \setminus \{u_i, v_j\}$ must have a unique minimum-weight perfect matching of weight $w(M) - w_{i,j}$. Thus from Theorem 72,

$$\mathsf{numz}\big(\mathsf{Det}(B[i \mid j])\big) = w(M) - w_{i,j}.$$

($\Leftarrow$): We prove the contrapositive. Assume $(i,j) \notin M$. Suppose for a contradiction that

$$w(M) - w_{i,j} = \mathsf{numz}\big(\mathsf{Det}(B[i \mid j])\big).$$

Then by Lemma 71 we can extract from the submatrix $B[i \mid j]$ a perfect matching Q of the bipartite graph $G' = G \setminus \{u_i, v_j\}$ with weight at most $w(M) - w_{i,j}$. But then $M' = Q \cup \{(i,j)\}$ is another perfect matching of G with $w(M') \leq w(M)$, a contradiction. □

Theorems 69, 72, and 73 complete the description and justification of our RDET algorithm for finding a perfect matching in a bipartite graph. Since these are theorems of VPV, it follows that VPV

proves the correctness of the algorithm.

4.4.3 Related bipartite matching problems

The correctness of the Mulmuley-Vazirani-Vazirani algorithm can easily be used to establish the correctness of RDET algorithms for related matching problems, for example, the maximum (cardinality) bipartite matching problem and the minimum-weight bipartite perfect matching problem, where the weights assigned to the edges are small. We refer to [47] for more details on these reductions.

Chapter 5

Moser's constructive proof of the Lovász Local Lemma

We will recall the statement of the Lovász Local Lemma (LLL). Consider the situation, where we have some collection of "bad" events $A_1, \ldots, A_m$ satisyfing

$$\mathbb{P}(A_i) \leq p_i < 1,$$

and we want to find a lower bound for $\mathbb{P}\left(\bigcap_{i=1}^m \overline{A}_i\right)$, the probability that none of A_i's happens. If all A_i's are independent, then

$$\mathbb{P}\left(\bigcap_{i=1}^m \overline{A}_i\right) \geq \prod_{i=1}^m (1 - p_i).$$

The LLL gives us a way to deal with situations when the events only have some limited dependencies. For simplicity, we will consider only the symmetric version of the LLL in this paper.

Theorem 74 (symmetric)**.** *Consider a set of events $A_1, \ldots, A_m$, where $\mathbb{P}(A_i) \leq p < 1$ and each A_i depends on at most d other A_j's. If $\mathrm{e} \cdot p \cdot (d+1) \leq 1$ (e is the base of the natural logarithm), then $\mathbb{P}\left(\bigcap_{i=1}^m \overline{A}_i\right) > 0$.*

We will be particularly interested in the application of this lemma to the k-SAT problem. Let F be a k-CNF formula with m clauses, and let A_i be the event that the ith clause is not satisfied. On a random assignment, we have $\mathbb{P}(A_i) = 2^{-k}$. For each clause $C \in F$, we let $\mathsf{VBL}(C)$ denote the set of variables occurring in C, and let

$$\Gamma(C) = \left\{\text{clause } D \text{ in } F \mid \mathsf{VBL}(C) \cap \mathsf{VBL}(D) \neq \varnothing\right\}.$$

To apply Theorem 74 to such a formula F, we set $p = 2^{-k}$ and $(d+1) = \Gamma(C)$, so if $|\Gamma(C)| \leq 2^k/\mathrm{e}$ then $\mathrm{e} \cdot p \cdot (d+1) \leq 1$, and hence the formula F is satisfiable.

Note that the LLL does not give us any information of how to construct a satisfying assignment for a formula satisfying the LLL assumption. Recall that Beck [8] was the first to give a polytime algorithm for finding satisfying assignments when the neighborhood size is at most $2^{k/48}$. Alon [2] simplified and randomized Beck's algorithm, and improved the neighborhood size to $2^{k/4}$. Srinivasan presented in [59] a variant that achieves a bound of essentially $O(2^{k/4})$. Moser [44] improved the bound to $2^k/8$.

Most recently, Moser and Tardos gave a constructive proof for a general (not necessarily symmetric) version of the LLL [45]. As a corollary, their constructive proof gives a randomized polytime algorithm for k-SAT with the neighborhood size up to $2^k/\mathrm{e}$.

It is worth emphasizing that the proofs given by Alon, Srinivasan, Moser, and Moser-Tardos are constructive in the sense that *randomized* polytime algorithms are provided for finding a satisfying assignment; deterministic polytime algorithms are known only for much weaker versions of the LLL. The notion of "randomized constructivity" has recently been elaborated by Gasarch and Haeupler [21], where Moser's technique was applied to yield a randomized algorithm that computes witnesses to a lower-bound of the van der Waerden number $W(k,2)$. Recall $W(k,2)$ is the smallest integer n such that every 2-coloring of the set $[n]$ has a monochromatic arithmetic progression of length k. Thus, it is interesting to analyze Moser's "randomized constructive" technique more formally using Jeřábek's framework.

5.1 Constructive proof for k-SAT with neighborhood size up to 2^{k-3}

For the rest of this section, we will consider a very slick and beautiful constructive proof by Moser for the special case of k-SAT, where the neighborhood size $\Gamma(C)$ is at most 2^{k-3} for every clause C. Let's consider the following randomized algorithm.

```
solve($F$):
1: Pick a random assignment for $F$
2: while ($\exists C \in F$, $C$ is unsatisfied) do
3:     Pick the lexicographically first such $C$
4:     call locally_fix($C$)
5: end while
```

The procedure for locally fixing a clause C (and its unsatisfied neighbors) is defined as follows:

```
locally_fix($C$):
1: Pick a random assignment for variables in VBL($C$)
2: while ($\exists D \in \Gamma(C)$, $D$ is unsatisfied) do
3:     Pick the lexicographically first such $D$
4:     call locally_fix($D$)
5: end while
```

In the analysis of this algorithm, we will treat the random source used by the algorithm as an additional random binary string input. Thus if the algorithm makes at most r calls to locally_fix (including all the recursive calls), then a binary random string input of length $n+rk$ is sufficient since n random bits are needed for the initial truth assignment and at most rk random bits are used for at most r calls to the locally_fix procedure. When the algorithm runs out of random bits, we will assume that the algorithm terminates with not necessarily a satisfying assignment.

Theorem 75. *On input a k-CNF formula F with n variables and m clauses such that $\Gamma(C) \le 2^{k-3}$ for every clause C, the theory $VPV + \mathsf{sWPHP}(\mathsf{FP})$ proves that there exists a random binary string of length $n+(m+1)k$*

on which the procedure solve *outputs a satisfying assignment for F.*

Proof. The proof of this theorem rests on the following important lemma.

Lemma 76. *If the algorithm makes at least $r = m + 1$ calls to* locally_fix *using a random binary string R of length $rk + n$, then we can losslessly encode R using a string of length $rk + n - 1$.*

Suppose the algorithm makes r calls to the locally_fix procedure (including all the recursive calls). Then the length of the random string R we used is $n + rk$, where n random bits are needed for the initial truth assignment and the remaining rk random bits are used for r calls to the locally_fix procedure.

It is important to observe that if we know the final truth assignment (after r calls to locally_fix procedure), and the whole execution history of the algorithm, then we can fully recover the original random string R used by the algorithm. The reason is that whenever a clause is unsatisfied, it reveals k random bits of the bit string R. So if we know the clause the algorithm has just fixed, then we also know the k random bits that have just been used by the algorithm. And thus, if we go backward in time and recover k random bits at each step, we will eventually recover the whole original random string R.

We next analyze the number of bits we need to encode an execution history of the algorithm. We first need to keep track of a list of clauses from the top level procedure solve that trigger the recursive procedure locally_fix. We observe that whenever locally_fix terminates, the number of satisfying clauses strictly increases. Actually we can prove by induction on the depth of all polynomial size recursion trees produced by a call to locally_fix that the following holds.

Lemma 77. *Whenever a call to* locally_fix *from* solve *terminates, there are always strictly less unsatisfied clauses, and moreover no new unsatisfied clause is introduced.*

Naively, we will need in the worst-case $m \log m$ bits to store this list of clauses that trigger the locally_fix procedure from the solve procedure. However, from the point of view of solve, by Lemma 77, no new unsatisfied clause is produced after each call to locally_fix terminates. And since the clauses are considered lexicographically, we can store the set of clauses that trigger locally_fix from solve instead, and it only takes m bits to store the bit-vector encoding the set of at most m clauses.

Now, whenever a clause C triggers the locally_fix procedure, we can store the information of the clauses that are locally fixed efficiently; since $|\Gamma(C)| \leq 2^{k-3}$, by using the lexicographical ordering on the clauses in $\Gamma(C)$, we need only $k - 3$ bits to index all clauses in $\Gamma(C)$. Thus, to store the full history tree of how the clauses in $\Gamma(C)$ are fixed, we can simply store the list of words that encode the nodes of the recursion tree, where each word has length

$$(k-3) + \text{ 2 bits to encode position of a clause in the recursion tree,}$$

which will be shown in the next lemma.

Lemma 78. *Two bits are needed to encode position of a clause in the recursion tree.*

Proof. We can use the first bit to encode whether or not fixing the current clause triggers another recursive call to locally_fix. For this we simply need to encode whether the next node in the list is a child of the current node.

Since we want to recover a recursion tree from a list of nodes, another difficulty is that whenever we return from a leaf of the recursion tree, how do we know the position of next node in the list? One possible way is to use the second bit to indicate the nodes that are expecting a neighbor, so whenever we return from a leaf, we will find the deepest node v on the path from root to the current leaf that is expecting a neighbor, and then insert the new node as the right-most child of the parent of v. □

Thus, in total, we need

$$m + r((k-3)+2) + \underbrace{n}_{\text{final truth assignment}} = m + r(k-1) + n$$

bits to recover the original random string R. When $r = m + 1$, we have

$$\begin{aligned} m + r(k-1) + n &= (r-1) + r(k-1) + n \\ &= rk + n - 1 \\ &< rk + n = \text{ length of } R. \end{aligned}$$

We have just shown Lemma 76.

Let $r = m + 1$. Then, using the above encoding method, we can define a function $G : \{0,1\}^{rk+n-1} \to \{0,1\}^{rk+n}$, where G treats an input $X \in \{0,1\}^{rk+n-1}$ as an encoding of an execution history of the procedure solve and outputs a random string of length $rk + n$. Note that if the input X doesn't encode a valid execution history, and thus cannot be decoded, then G simply returns some default string, say $\vec{0} \in \{0,1\}^{rk+n}$.

Now we observe that the range of G contains all the random strings on which at least $m + 1$ calls to locally_fix are made. Thus, $\mathsf{sWPHP}(G)$ proves that there exists a "good" random string of length $(m+1)k + n$ that lies outside the range of G on which at most m calls to locally_fix are made, and thus the procedure solve terminates with satisfying assignment within $O(m)$ steps. □

We have just shown in Theorem 75 that the theory $VPV + \mathsf{sWPHP}(\mathcal{L}_{\mathsf{FP}})$ proves the existence of a satisfying assignment for each instance of k-SAT with neighborhood size up to 2^{k-3}. The next corollary shows that we can also compute a satisfying assignment efficiently.

Corollary 79. *There is a randomized polynomial-time algorithm that, on input a k-CNF formula F with n variable and m clauses such that $\Gamma(C) \leq 2^{k-3}$ for every clause C, outputs a satisfying assignment for F with probability at least* $1/2$.

Proof. Since $VPV + \mathsf{sWPHP}(\mathcal{L}_{\mathsf{FP}}) \subseteq V^1 + \mathsf{sWPHP}(\mathcal{L}_{\mathsf{FP}})$, and we have just shown in Theorem 75 that $VPV + \mathsf{sWPHP}(\mathcal{L}_{\mathsf{FP}})$ proves the existence of a satisfying assignment for each formula F of k-SAT with neighborhood size up to 2^{k-3}. Thus, it follows from Wilkie's Witnessing Theorem (see Theorem 5) that there is a randomized polynomial-time algorithm that outputs a satisfying assignment for F with probability at least $1/2$. □

5.2 Constructive proofs for more general versions of the LLL

Messner and Thierauf [41] generalized Moser's elegant proof to instances of k-SAT with neighborhood size up to $2^k/e$. I believe that this proof is also formalizable in $VPV + \mathsf{sWPHP}(\mathsf{FP})$ by formalizing a suitable version of Stirling's approximation using methods developed by Jeřábek in [28, 29].

However, it is not clear at all how to formalize the full proof in [45], which uses properties of *infinite branching processes*. Thus we leave open the question of whether the constructive proof of the LLL by Moser-Tardos [45] is formalizable in $VPV + \mathsf{sWPHP}(\mathsf{FP})$.

Chapter 6

Formalizing the Goldreich-Levin Theorem

So far we have seen how to formalize simple theorems where the statements with probability are simple enough to be reformulated as surjective mappings between definable sets. However sophisticated proofs involve more complicated statements about probability or expectation cannot easily be translated to such mappings. As a result, we need to develop a more systematic framework to talk about these basic probabilistic concepts. This chapter requires some background on Jeřábek's approximate counting framework [30], which was summarized in Section 2.3.

6.1 Formalizing basic probability definitions in the conservative extension $HARD^A$ of $VPV + \mathsf{sWPHP}(\mathcal{L}_{\mathsf{FP}})$

The definition of $\mathcal{X} \precsim_\epsilon \mathcal{Y}$ from Section 2.3 is problematic if we want to use it in inductive formulas involving approximate counting or situations where we need to approximate size of sets which themselves are defined using approximate counting. The reason is that stating $\mathcal{X} \precsim_\epsilon \mathcal{Y}$ would require an unbounded $\exists\Pi^B_2$-formula; even when restricting its usage to the case covered in Theorem 11, we cannot do better than Σ^B_2. Jeřábek solved this problem by working in a suitable fully conservative extension $HARD^A$ of $VPV + \mathsf{sWPHP}(\mathcal{L}_{\mathsf{FP}})$ introduced in [28]. Let α be a new oracle function symbol. We define $VPV(\alpha)$ similarly to VPV but we allow the oracle function symbol α to appear in functions constructed by limited recursion on notation. Then $VPV(\alpha)$-functions correspond to polynomial-time algorithms with access to oracle α.

Definition 80 ([28])**.** The theory $HARD^A$ is an extension of $VPV(\alpha) + \mathsf{sWPHP}(\mathcal{L}_{\mathsf{FP}}(\alpha))$ by the axioms

$\alpha(x)$ is a binary string encoding a truth-table of a Boolean function in $\log(x+1)$ variables,

$x \geq c \to \mathsf{Hard}^A_{1/3}(\alpha(x))$,

$\log(x+1) = \log(y+1) \to \alpha(x) = \alpha(y)$

where c is the constant from Lemma 7. (Recall that the notation Hard^A_ϵ was defined in Definition 6.)

The theory $HARD^A$ contains oracle function symbol α that outputs appropriate hard-in-average functions needed to construct the Nisan-Wigderson generators used in Theorem 11. Thus, we have the following theorem.

Theorem 81 (adapted from [30]). *There is an integer-valued $VPV(\alpha)$-function $\mathsf{Size}(C,n,\epsilon)$ such that $HARD^A$ proves that if the set $\mathcal{X} \subseteq \{0,1\}^n$ is definable by some polysize circuit $C : \{0,1\}^n \to \{0,1\}$, then for all $\epsilon = 1/\mathsf{poly}(n)$,*

$$\mathcal{X} \approx_\epsilon [\mathsf{Size}(C,n,\epsilon)) .$$

The circuits in (2.7) and (2.8) are also constructible by $VPV(\alpha)$-functions where $S = \mathsf{Size}(C,n,\epsilon)$. We often abuse the notation and write $\mathsf{Size}(\mathcal{X},\epsilon)$ instead of $\mathsf{Size}(C,n,\epsilon)$.

By analyzing the proof of Jeřábek's Theorem (i.e. Theorem 11), we can make the following interesting observation by analyzing the choice of Nisan-Wigderson generators in the proof.

Corollary 82. *($VPV + \mathsf{sWPHP}(\mathcal{L}_{\mathsf{FP}}) \vdash$) Let $C_0 : \{0,1\}^n \to \{0,1\}$ and $C_1 : \{0,1\}^n \to \{0,1\}$ be Boolean circuits of sizes $\mathsf{poly}(n)$, and $\epsilon = 1/\mathsf{poly}(n)$. Suppose at least one of the following condition holds*

1. *C_0 and C_1 have the same size*
2. *ϵ is sufficiently small, in particular*

$$4\log\left(\frac{n}{\epsilon}+1\right) \geq \max\left\{12\log(n+1), \frac{1}{\delta}\log(n+1), 4\Big(\log\big(\max(|C_0|,|C_1|)+1\big)+1\Big)\right\},$$

where δ is the constant from Lemma 9.

Then the proof of Theorem 11 chooses the same Nisan-Wigderson generator to approximate the sizes of the sets defined by C_0 and C_1.

Proof. This follows from the choice of the parameters in the proof of Theorem 11. The Nisan-Wigderson generator used by Jeřábek's proof to approximate the size of the set defined by each circuit C_i is chosen based on parameters n and ℓ, where

$$\ell = \max\left\{4\log\left(\frac{n}{\epsilon}+1\right), 12\log(n+1), \frac{1}{\delta}\log(n+1), 4\Big(\log\big(\max(|C_0|,|C_1|)+1\big)+1\Big)\right\}$$

Thus when ϵ is sufficiently small such that the condition in (2) is satisfied, we have

$$\ell = 4\log\left(\frac{n}{\epsilon}+1\right)$$

which depends only on ϵ and n. $\square$

By the observation from Corollary 82, we can prove the following lemmas, which are used very often in this chapter, but are not available in Jeřábek's work [30].

Lemma 83. *($HARD^A \vdash$) Let $\mathcal{X}$, $\mathcal{X}^c := \{0,1\}^n \setminus \mathcal{X}$, $\mathcal{Y}$ and $\mathcal{X}_1,\dots,\mathcal{X}_k$ be subsets of $\{0,1\}^n$ definable by circuits. Then for all sufficiently small $\epsilon = 1/\mathsf{poly}(n)$,*

1. *If $\mathcal{X} \subseteq \mathcal{Y}$, then $\mathsf{Size}(\mathcal{X},\epsilon) \leq \mathsf{Size}(\mathcal{Y},\epsilon)$.*
2. *If $\mathcal{X} = \mathcal{Y}$, then $\mathsf{Size}(\mathcal{X},\epsilon) = \mathsf{Size}(\mathcal{Y},\epsilon)$.*

3. *If $\mathcal{X} = \bigcup_{i=1}^{k} \mathcal{X}_i$, then* $\mathsf{Size}(\mathcal{X}, \epsilon) \leq \sum_{i=1}^{k} \mathsf{Size}(\mathcal{X}_i, \epsilon)$.
4. *If $\mathcal{X} = \biguplus_{i=1}^{k} \mathcal{X}_i$, then* $\mathsf{Size}(\mathcal{X}, \epsilon) = \sum_{i=1}^{k} \mathsf{Size}(\mathcal{X}_i, \epsilon)$.
5. *If* $\mathsf{Size}(\mathcal{X}, \epsilon) = S$, *then* $\mathsf{Size}(\mathcal{X}^c, \epsilon) = 2^n - S$.
6. $\mathsf{Size}(\mathcal{X}, \epsilon) \leq S$ *iff* $\mathsf{Size}(\mathcal{X}^c, \epsilon) \geq 2^n - S$.

Proof. When ϵ is sufficiently small as stated in Corollary 82, we actually use the same Nisan-Wigderson generator to compute $\mathsf{Size}(\mathcal{X}, \epsilon)$, $\mathsf{Size}(\mathcal{X}^c, \epsilon)$, $\mathsf{Size}(\mathcal{Y}, \epsilon)$ and $\mathsf{Size}(\mathcal{X}_i, \epsilon)$ for all $i \in [k]$. □

Lemma 84. *($HARD^A \vdash$) Let $\mathcal{X}, \mathcal{Y} \subseteq \{0,1\}^n$, $\mathcal{U} \subseteq \{0,1\}^p$ and $\mathcal{V} \subseteq \{0,1\}^q$, where $p + q = n$ and all of these sets are defined by circuits of $\mathsf{poly}(n)$ size. Let $\gamma < 1$. Then for all $\epsilon, \delta = 1/\mathsf{poly}(n)$,*

1. *If $\mathcal{X} \precsim_\gamma \mathcal{Y}$, then* $\mathsf{Size}(\mathcal{X}, \epsilon) \leq \mathsf{Size}(\mathcal{Y}, \epsilon) + (\gamma + 2\epsilon + \delta)2^n$.
2. *If $\mathcal{X} \approx_\gamma \mathcal{Y}$, then* $\mathsf{Size}(\mathcal{X}, \epsilon) = \mathsf{Size}(\mathcal{Y}, \epsilon) \pm (\gamma + 2\epsilon + \delta)2^n$.
3. *If $\mathcal{X} \precsim_\gamma \mathcal{U} \times \mathcal{V}$, then* $\mathsf{Size}(\mathcal{X}, \epsilon) \leq \mathsf{Size}(\mathcal{U}, \epsilon)\mathsf{Size}(\mathcal{V}, \epsilon) + (\gamma + 3\epsilon + \epsilon^2 + \delta)2^n$.
4. *If $\mathcal{X} \approx_\gamma \mathcal{U} \times \mathcal{V}$, then* $\mathsf{Size}(\mathcal{X}, \epsilon) = \mathsf{Size}(\mathcal{U}, \epsilon)\mathsf{Size}(\mathcal{V}, \epsilon) \pm (\gamma + 3\epsilon + \epsilon^2 + \delta)2^n$.
5. *If $\mathcal{X} \precsim_\gamma \mathcal{U} \times \{0,1\}^q$, then* $\mathsf{Size}(\mathcal{X}, \epsilon) \leq \mathsf{Size}(\mathcal{U}, \epsilon)2^q + (\gamma + 2\epsilon + \delta)2^n$.
6. *If $\mathcal{X} \approx_\gamma \mathcal{U} \times \{0,1\}^q$, then* $\mathsf{Size}(\mathcal{X}, \epsilon) = \mathsf{Size}(\mathcal{U}, \epsilon)2^q \pm (\gamma + 2\epsilon + \delta)2^n$.

Proof. 1. We have

$$[\mathsf{Size}(\mathcal{X}, \epsilon)) \precsim_\epsilon \mathcal{X} \precsim_\gamma \mathcal{Y} \precsim_\epsilon [\mathsf{Size}(\mathcal{Y}, \epsilon)).$$

This by Lemma 13 (3) implies that

$$[\mathsf{Size}(\mathcal{X}, \epsilon)) \precsim_{\gamma+2\epsilon} [\mathsf{Size}(\mathcal{Y}, \epsilon)).$$

Then by Lemma 14 (2) the result follows.

2. Follows from (1).

3. We have

$$\begin{aligned} [\mathsf{Size}(X, \epsilon)) &\precsim_\epsilon \mathcal{X} \\ &\precsim_\gamma \mathcal{U} \times \mathcal{V} \\ &\precsim_{2\epsilon+\epsilon^2} [\mathsf{Size}(\mathcal{U}, \epsilon)\mathsf{Size}(\mathcal{V}, \epsilon)) && \text{(by Lemma 13 (5))} \end{aligned}$$

Thus by Lemma 13 (3) we get

$$[\mathsf{Size}(\mathcal{X}, \epsilon)) \precsim_{\gamma+3\epsilon+\epsilon^2} [\mathsf{Size}(\mathcal{U}, \epsilon)\mathsf{Size}(\mathcal{V}, \epsilon)).$$

By Lemma 14 (2) this implies the result.

4. Follows from (3).

5. We have
$$\big[\mathsf{Size}(\mathcal{X},\epsilon)\big) \precsim_{\epsilon} \mathcal{X} \precsim_{\gamma} \mathcal{U} \times \{0,1\}^q \precsim_{\epsilon} \big[\mathsf{Size}(\mathcal{U},\epsilon)\cdot 2^q\big)$$
Thus by Lemma 13 (3) we get
$$\big[\mathsf{Size}(\mathcal{X},\epsilon)\big) \precsim_{\gamma+2\epsilon} \big[\mathsf{Size}(\mathcal{U},\epsilon)\cdot 2^q\big).$$
Then we can apply Lemma 14 (2) to get the result.

6. Follows from (5).

□

6.1.1 Approximate probability

Using the function Size defined in the previous section, we can easily define the notion of *approximate probability* as follows.

Definition 85 (adapted from [30])**.** (in *HARD*^{A}) Let $C : \{0,1\}^n \to \{0,1\}$ be a circuit. Given an error $\epsilon = 1/\mathsf{poly}(n)$, we define
$$\mathbb{P}^{\epsilon}_{X\in\{0,1\}^n}[C(X)] = \frac{\mathsf{Size}(C,n,\epsilon)}{2^n}.$$

Informally, $\mathbb{P}^{\epsilon}_{X\in\{0,1\}^n}[C(X)]$ approximates the usual probability notion $\mathbb{P}_{X\in\{0,1\}^n}[C(X)]$ within ϵ-error since it follows from the definition of approximate counting that
$$\mathbb{P}_{X\in\{0,1\}^n}[C(X)] - \epsilon \le \mathbb{P}^{\epsilon}_{X\in\{0,1\}^n}[C(X)] \le \mathbb{P}_{X\in\{0,1\}^n}[C(X)] + \epsilon.$$

Note that we often need to talk about binary rational numbers when dealing with probability. However in *VPV* we can easily define binary rational numbers and prove their basic field properties. Using this definition of approximate probability, we can show the following properties.

Corollary 86. *(HARD*^{A} $\vdash$*) Let $\mathcal{X}$, $\mathcal{X}^c := \{0,1\}^n \setminus \mathcal{X}$, $\mathcal{Y}$, and $\mathcal{X}_1,\dots,\mathcal{X}_k$ be subsets of $\{0,1\}^n$ definable by circuits. Then for all sufficiently small $\epsilon = 1/\mathsf{poly}(n)$,*

1. *If $\mathcal{X} \subseteq \mathcal{Y}$, then $\mathbb{P}^{\epsilon}_{X\in\{0,1\}^n}[X\in\mathcal{X}] \le \mathbb{P}^{\epsilon}_{X\in\{0,1\}^n}[X\in\mathcal{Y}]$.*
2. *If $\mathcal{X} = \mathcal{Y}$, then $\mathbb{P}^{\epsilon}_{X\in\{0,1\}^n}[X\in\mathcal{X}] = \mathbb{P}^{\epsilon}_{X\in\{0,1\}^n}[X\in\mathcal{Y}]$.*
3. *If $\mathcal{X} = \bigcup_{i=1}^k \mathcal{X}_i$, then $\mathbb{P}^{\epsilon}_{X\in\{0,1\}^n}[X\in\mathcal{X}] \le \sum_{i=1}^k \mathbb{P}^{\epsilon}_{X\in\{0,1\}^n}[X\in\mathcal{X}_i]$.* *(union bound)*
4. *If $\mathcal{X} = \biguplus_{i=1}^k \mathcal{X}_i$, then $\mathbb{P}^{\epsilon}_{X\in\{0,1\}^n}[X\in\mathcal{X}] = \sum_{i=1}^k \mathbb{P}^{\epsilon}_{X\in\{0,1\}^n}[X\in\mathcal{X}_i]$.*
5. *If $\mathbb{P}^{\epsilon}_{X\in\{0,1\}^n}[X\in\mathcal{X}] = \beta$, then $\mathbb{P}^{\epsilon}_{X\in\{0,1\}^n}[X\in\mathcal{X}^c] = 1-\beta$.*
6. *$\mathbb{P}^{\epsilon}_{X\in\{0,1\}^n}[X\in\mathcal{X}] \le \beta$ iff $\mathbb{P}^{\epsilon}_{X\in\{0,1\}^n}[X\in\mathcal{X}^c] \ge 1-\beta$.*

Proof. Follows from Lemma 83. □

Corollary 87. *(HARD*^{A} $\vdash$*) Let $\mathcal{X},\mathcal{Y} \subseteq \{0,1\}^n$, $\mathcal{U} \subseteq \{0,1\}^p$ and $\mathcal{V} \subseteq \{0,1\}^q$, where $p+q=n$ and all of these sets are definable by circuits. Let $\gamma,\delta \le 1$ and $\delta > 0$. Then for all sufficiently small $\epsilon = 1/\mathsf{poly}(n)$,*

1. *If $\mathcal{X} \precsim_{\gamma} \mathcal{Y}$, then $\mathbb{P}^{\epsilon}_{X\in\{0,1\}^n}[X\in\mathcal{X}] \le \mathbb{P}^{\epsilon}_{X\in\{0,1\}^n}[X\in\mathcal{Y}] + (\gamma+2\epsilon+\delta)$.*

2. *If $\mathcal{X} \approx_\gamma \mathcal{Y}$, then* $\mathbb{P}^\epsilon_{X\in\{0,1\}^n}[X \in \mathcal{X}] = \mathbb{P}^\epsilon_{X\in\{0,1\}^n}[X \in \mathcal{Y}] \pm (\gamma + 2\epsilon + \delta)$.
3. *If $\mathcal{X} \precsim_\gamma \mathcal{U} \times \mathcal{V}$, then* $\mathbb{P}^\epsilon_{X\in\{0,1\}^n}[X \in \mathcal{X}] \leq \mathbb{P}^\epsilon_{X\in\{0,1\}^p}[X \in \mathcal{U}]\mathbb{P}^\epsilon_{X\in\{0,1\}^q}[X \in \mathcal{V}] + (\gamma + 3\epsilon + \epsilon^2 + \delta)$.
4. *If $\mathcal{X} \approx_\gamma \mathcal{U} \times \mathcal{V}$, then* $\mathbb{P}^\epsilon_{X\in\{0,1\}^n}[X \in \mathcal{X}] = \mathbb{P}^\epsilon_{X\in\{0,1\}^p}[X \in \mathcal{U}]\mathbb{P}^\epsilon_{X\in\{0,1\}^q}[X \in \mathcal{V}] \pm (\gamma + 3\epsilon + \epsilon^2 + \delta)$.
5. *If $\mathcal{X} \precsim_\gamma \mathcal{U} \times \{0,1\}^q$, then* $\mathbb{P}^\epsilon_{X\in\{0,1\}^n}[X \in \mathcal{X}] \leq \mathbb{P}^\epsilon_{X\in\{0,1\}^p}[X \in \mathcal{U}] + (\gamma + 2\epsilon + \delta)$.
6. *If $\mathcal{X} \approx_\gamma \mathcal{U} \times \{0,1\}^q$, then* $\mathbb{P}^\epsilon_{X\in\{0,1\}^n}[X \in \mathcal{X}] = \mathbb{P}^\epsilon_{X\in\{0,1\}^p}[X \in \mathcal{U}] \pm (\gamma + 2\epsilon + \delta)$.

Proof. Follows from Lemma 84. □

6.1.2 Averaging principles

In [30], Jeřábek formalized a version of the averaging principle which says that the average size of a family of sets must be smaller than or to equal to the size of at least one set in the family. We will need a slightly more general proposition:

Theorem 88 (Averaging principles). *($HARD^A \vdash$) Let $\mathcal{X} \subseteq \{0,1\}^n \times \{0,1\}^m$ and $\mathcal{Y} \subseteq \{0,1\}^m$ be definable by circuits. Let $\mathcal{X}_Y := \{X \mid \langle X, Y\rangle \in \mathcal{X}\}$ and $\gamma = 1/\mathsf{poly}(n)$.*

1. *If $\mathcal{X}_Y \precsim_\epsilon [S)$ for every $Y \in \mathcal{Y}$ and $\mathcal{Y} \precsim_\delta [T)$, then $\mathcal{X} \cap (\{0,1\}^n \times \mathcal{Y}) \precsim_{\epsilon+\delta+\epsilon\delta+\gamma} [ST)$.*
2. *If $\mathcal{X}_Y \succsim_\epsilon [S)$ for every $Y \in \mathcal{Y}$ and $\mathcal{Y} \succsim_\delta [T)$, then $\mathcal{X} \cap (\{0,1\}^n \times \mathcal{Y}) \succsim_{\epsilon+\delta+\epsilon\delta+\gamma} [ST)$.*

It might be more intuitive to read Part (1) of Theorem 88 contrapositively: if we have a family of at most T sets, such that the size of their union is more than ST, then one of the sets must be of size at least $ST/T = S$. Part (2) is simply the dual of Part (1).

Proof. Since Part (1) was already shown in [30], we only need to prove Part (2). This also helps us review the common techniques used in [30].

We know by Theorem 81 applied to (2.8) there exist functions F and v definable by circuits such that

$$F(Y, \bullet) : [v(Y)) \times \big(\mathcal{X}_Y \uplus [\gamma 2^n)\big) \twoheadrightarrow [v(Y)) \times \big[\mathsf{Size}(\mathcal{X}_Y, \gamma)\big).$$

By letting γ be a sufficiently small γ_0, we can assume that $v(Y)$ does not depend on Y. Also for each $Y \in \mathcal{Y}$, since $[\mathsf{Size}(\mathcal{X}_Y, \gamma_0)) \succsim_{\gamma_0} \mathcal{X}_Y \succsim_\epsilon [S)$, by Lemma 14 (4) we have $\mathsf{Size}(\mathcal{X}_Y, \gamma_0) + (\epsilon + 2\gamma_0)2^n \geq S$. Thus by Lemma 13 (3) we can obtain a function F' such that

$$F'(Y, \bullet) : [v) \times \big(\mathcal{X}_Y \uplus [(\epsilon + 3\gamma_0)2^n)\big) \twoheadrightarrow [v) \times \big[S\big)$$

for every $Y \in \mathcal{Y}$.

Again by Theorem 11 there is a function G definable by circuit and a number w such that

$$G : [w) \times \big(\mathcal{Y} \uplus [\delta 2^m)\big) \twoheadrightarrow [w) \times [T).$$

Thus, suitable composition of F' and G and a slight extension of the domain give us a surjection

$$[vw) \times \Big(\big(\mathcal{X} \cap (\{0,1\}^n \times \mathcal{Y}\big) \uplus \big[(\epsilon + \delta + \delta\epsilon + 3\gamma_0 + 3\delta\gamma_0)2^{m+n}\big)\Big) \twoheadrightarrow [vw) \times \big[ST\big).$$

For sufficiently small γ_0, we have

$$\epsilon + \delta + \delta\epsilon + 3\gamma_0 + 3\delta\gamma_0 \le \epsilon + \delta + \delta\epsilon + \gamma,$$

and thus the result follows. □

From Theorem 88, we can prove another useful version of the averaging principle using the approximate probability language developed above.

Proposition 89. *($HARD^A \vdash$) Let $Q : \{0,1\}^n \times \{0,1\}^m \to \{0,1\}$ be a predicate defined by a polysize circuit. Let $0 \le \gamma \le \beta < 1$. For every sufficiently small $\epsilon = 1/\mathsf{poly}(n)$, if*

$$\mathbb{P}^{\epsilon}_{(X,R)\in\{0,1\}^n\times\{0,1\}^m}\left[Q(X,R)\right] \ge \beta,$$

then we have

$$\mathbb{P}^{\epsilon}_{X\in\{0,1\}^n}\left[\mathbb{P}^{\epsilon}_{R\in\{0,1\}^m}\left[Q(X,R)\right] \ge \gamma\right] \ge \beta - \gamma - 7\epsilon.$$

Proof. Let

$$\mathcal{U} = \left\{(X,R) \in \{0,1\}^n \times \{0,1\}^m \mid Q(X,R)\right\}.$$

Let $P(X)$ be the predicate asserting that

$$\mathbb{P}^{\epsilon}_{R\in\{0,1\}^m}\left[Q(X,R)\right] \ge \gamma,$$

and thus $P(X)$ is definable within $HARD^A$. Let

$$\mathcal{S} = \{X \in \{0,1\}^n \mid P(X)\}.$$

Since

$$\mathcal{U} = \left\{(X,R) \in \{0,1\}^n \times \{0,1\}^m \mid P(X) \wedge Q(X,R)\right\} \cup \left\{(X,R) \in \{0,1\}^n \times \{0,1\}^m \mid \neg P(X) \wedge Q(X,R)\right\},$$

for we can choose sufficiently small ϵ and apply Lemma 83 (3) to get,

$$\begin{aligned}
&\mathsf{Size}(\mathcal{U},\epsilon)\\
&\le \mathsf{Size}\left(\left\{(X,R) \in \{0,1\}^n \times \{0,1\}^m \mid P(X) \wedge Q(X,R)\right\},\epsilon\right)\\
&\qquad\qquad + \mathsf{Size}\left(\left\{(X,R) \in \{0,1\}^n \times \{0,1\}^m \mid \neg P(X) \wedge Q(X,R)\right\},\epsilon\right)\\
&\le \mathsf{Size}\left(\left\{(X,R) \in \{0,1\}^n \times \{0,1\}^m \mid P(X)\right\},\epsilon\right)\\
&\qquad\qquad + \mathsf{Size}\left(\left\{(X,R) \in \{0,1\}^n \times \{0,1\}^m \mid \neg P(X) \wedge Q(X,R)\right\},\epsilon\right)\\
&\le \mathsf{Size}\left(\mathcal{S} \times \{0,1\}^m,\epsilon\right) + \mathsf{Size}\left(\left\{(X,R) \in \{0,1\}^n \times \{0,1\}^m \mid \neg P(X) \wedge Q(X,R)\right\},\epsilon\right).
\end{aligned}$$

Claim 1: $\mathsf{Size}\left(\left\{(X,R) \in \{0,1\}^n \times \{0,1\}^m \mid \neg P(X) \wedge Q(X,R)\right\},\epsilon\right) \le \gamma \cdot 2^{n+m} + 4\epsilon 2^{n+m}$.

Proof of Claim 1. Let $\mathcal{R}_X$ denote the set $\{R \in \{0,1\}^m \mid Q(X,R)\}$. Then we observe that for all X satisfying $\neg P(X)$, i.e., $X \in \mathcal{S}^c = \{0,1\}^n \setminus \mathcal{S}$, it follows from how $P(X)$ is defined that

$$\mathsf{Size}(\mathcal{R}_X,\epsilon) \le \gamma \cdot 2^m,$$

which implies that

$$\mathcal{R}_X \precsim_\epsilon [\gamma 2^m)\,.$$

Since we also have $\mathcal{S}^c \precsim_0 \{0,1\}^n$, by applying Theorem 88 we can show that

$$\begin{aligned}\big\{(X,R) \in \{0,1\}^n \times \{0,1\}^m \mid \neg P(X) \wedge Q(X,R)\big\} &= \big\{(X,R) \in \{0,1\}^n \times \{0,1\}^m \mid X \in \mathcal{S}^c \wedge R \in \mathcal{R}_X\big\} \\ &\precsim_{2\epsilon} [2^n \cdot \gamma \cdot 2^m)\end{aligned}$$

Since

$$\big[\mathsf{Size}\left(\big\{(X,R) \in \{0,1\}^n \times \{0,1\}^m \mid \neg P(X) \wedge Q(X,R)\big\}, \epsilon\right)\big) \precsim_\epsilon \big\{(X,R) \in \{0,1\}^n \times \{0,1\}^m \mid \neg P(X) \wedge Q(X,R)\big\},$$

it follows by Lemma 14 (4) that

$$\mathsf{Size}\left(\big\{(X,R) \in \{0,1\}^n \times \{0,1\}^m \mid \neg P(X) \wedge Q(X,R)\big\}, \epsilon\right) \leq \gamma \cdot 2^{n+m} + 4\epsilon 2^{n+m}.$$

□

We can apply Lemma 84 to show that

$$\mathsf{Size}\left(\mathcal{S} \times \{0,1\}^m, \epsilon\right) \leq \mathsf{Size}\left(\mathcal{S}, \epsilon\right) \cdot 2^m + 3\epsilon 2^{n+m}.$$

Together with Claim 1, we have

$$\begin{aligned}\mathsf{Size}(\mathcal{U},\epsilon) &\leq \mathsf{Size}\left(\mathcal{S} \times \{0,1\}^m, \epsilon\right) + \mathsf{Size}\left(\big\{(X,R) \in \{0,1\}^n \times \{0,1\}^m \mid \neg P(X) \wedge Q(X,R)\big\}, \epsilon\right) \\ &\leq \mathsf{Size}\left(\mathcal{S}, \epsilon\right) \cdot 2^m + \gamma \cdot 2^{n+m} + 7\epsilon 2^{n+m}.\end{aligned}$$

Since we have $\mathsf{Size}(\mathcal{U},\epsilon) \geq \beta 2^{n+m}$ by the assumption of the lemma, it follows that

$$\beta 2^{n+m} \leq \mathsf{Size}\left(\mathcal{S}, \epsilon\right) \cdot 2^m + \gamma \cdot 2^{n+m} + 7\epsilon 2^{n+m}.$$

After simplification, we get

$$\mathsf{Size}\left(\mathcal{S}, \epsilon\right) \geq (\beta - \gamma) 2^n - 7\epsilon 2^n.$$

Hence,

$$\mathbb{P}^\epsilon_{X \in \{0,1\}^n}\left[\mathbb{P}^\epsilon_{R \in \{0,1\}^m}\left[Q(X,R)\right] \geq \gamma\right] = \mathbb{P}^\epsilon_{X \in \{0,1\}^n}\left[X \in \mathcal{S}\right] \geq \beta - \gamma - 7\epsilon.$$

□

6.1.3 Approximate expectation

A *feasible random variable* is a circuit computing a function

$$F : \{0,1\}^n \to \{\alpha_1, \ldots, \alpha_m\},$$

where $\alpha_1, \ldots, \alpha_m$ are arbitrary binary rational numbers, and $m = \mathsf{poly}(n)$. Note that the *sample space* here is the set of binary strings $\{0,1\}^n$ with the uniform distribution

Define the ϵ-approximate expectation of F as

$$\mathbb{E}^{\epsilon}[F] := \sum_{i=1}^{m} \alpha_i \cdot \mathbb{P}^{\epsilon}_{X\in\{0,1\}^n}[F(X)=\alpha_i]. \tag{6.1}$$

Lemma 90 (Markov's inequality). *($HARD^A$ $\vdash$) Let $F : \{0,1\}^n \to \{\alpha_1,\ldots,\alpha_m\}$ be a feasible random variable. Let $\beta > 0$. Then for all sufficiently small $\epsilon = 1/\mathsf{poly}(n)$,*

$$\mathbb{P}^{\epsilon}_{X\in\{0,1\}^n}[F(X)\geq\beta] \leq \frac{\mathbb{E}^{\epsilon}[F]}{\beta}$$

Proof. We have

$$\begin{aligned}
\mathbb{E}^{\epsilon}[F] &= \sum_{i=1}^{m} \alpha_i \cdot \mathbb{P}^{\epsilon}_{X\in\{0,1\}^n}[F(X)=\alpha_i] \\
&= \sum_{i:\alpha_i<\beta} \alpha_i \cdot \mathbb{P}^{\epsilon}_{X\in\{0,1\}^n}[F(X)=\alpha_i] + \sum_{i:\alpha_i\geq\beta} \alpha_i \cdot \mathbb{P}^{\epsilon}_{X\in\{0,1\}^n}[F(X)=\alpha_i] \\
&\geq \sum_{i:\alpha_i\geq\beta} \alpha_i \cdot \mathbb{P}^{\epsilon}_{X\in\{0,1\}^n}[F(X)=\alpha_i] \\
&\geq \beta \cdot \sum_{i:\alpha_i\geq\beta} \mathbb{P}^{\epsilon}_{X\in\{0,1\}^n}[F(X)=\alpha_i]
\end{aligned}$$

Thus, it only remains to relate the sum $\sum_{i:\alpha_i\geq\beta} \mathbb{P}^{\epsilon}_{X\in\{0,1\}^n}[F(X)=\alpha_i]$ with $\mathbb{P}^{\epsilon}_{X\in\{0,1\}^n}[F(X)\geq\beta]$.

Let $\mathcal{Q}_i = \{X \in \{0,1\}^n \mid F(X)=\alpha_i\}$, and let $\mathcal{Q} = \{X \in \{0,1\}^n \mid F(X)\geq\beta\}$. Since $\mathcal{Q}_i$ and $\mathcal{Q}_j$ are disjoint for every $i \neq j$, we have

$$\mathcal{Q} = \biguplus_{i:S_i\geq\beta} \mathcal{Q}_i.$$

Thus for sufficiently small $\epsilon = 1/\mathsf{poly}(n)$ we can apply Corollary 86(4) to get

$$\sum_{i:\alpha_i\geq\beta} \mathbb{P}^{\epsilon}_{X\in\{0,1\}^n}[F(X)=\alpha_i] = \sum_{i:\alpha_i\geq\beta} \mathbb{P}^{\epsilon}_{X\in\{0,1\}^n}[X\in\mathcal{Q}_i] = \mathbb{P}^{\epsilon}_{X\in\{0,1\}^n}[X\in\mathcal{Q}] = \mathbb{P}^{\epsilon}_{X\in\{0,1\}^n}[F(X)\geq\beta].$$

Putting everything together, we have

$$\begin{aligned}
\mathbb{E}^{\epsilon}[F] &\geq \beta \cdot \sum_{i:S_i\geq\beta} \mathbb{P}^{\epsilon}_{X\in\{0,1\}^n}[F(X)=\alpha_i] \\
&\geq \beta \cdot \mathbb{P}^{\epsilon}_{X\in\{0,1\}^n}[F(X)\geq\beta].
\end{aligned}$$

□

Lemma 91. *($HARD^A$ $\vdash$) Let $F : \{0,1\}^n \to \{\alpha_1,\ldots,\alpha_m\}$ be a feasible random variable. Let*

$$\mathcal{Q}_i = \{X \in \{0,1\}^n \mid F(X)=\alpha_i\},$$

be defined by circuits of $\mathsf{poly}(n)$ size. Let

$$\mathsf{NW}^{n,\epsilon} : \{0,1\}^t \to \{0,1\}^n$$

be the Nisan-Wigderson constructed as in Jeřábek's proof of [30, Theorem 2.7], where $t = O(\log n)$. Then for all sufficiently small $\epsilon = 1/\mathsf{poly}(n)$,

$$\mathbb{E}^{\epsilon}[F] = \frac{1}{2^t} \sum_{S \in \{0,1\}^t} F(\mathsf{NW}^{n,\epsilon}(S)).$$

Proof. Note that since $2^t = \mathsf{poly}(n)$, we can compute the cardinalities of subsets of $\{0,1\}^t$ directly.

$$\begin{aligned}\mathbb{E}^{\epsilon}[F] &= \sum_{i=1}^{m} \alpha_i \cdot \mathbb{P}^{\epsilon}_{X \in \{0,1\}^n}[F(X) = \alpha_i] \\ &= \sum_{i=1}^{m} \alpha_i \cdot \frac{|\{X \in \{0,1\}^t \mid F(\mathsf{NW}^{n,\epsilon}(X)) = \alpha_i\}|}{2^t} \\ &= \frac{1}{2^t} \sum_{S \in \{0,1\}^t} F(\mathsf{NW}^{n,\epsilon}(S)).\end{aligned}$$

□

Lemma 92 (linearity of expectation). *($HARD^A \vdash$) Let $F_1, \ldots, F_m$ be feasible random variables, where*

$$F_i : \{0,1\}^n \to \{\alpha_1, \ldots, \alpha_k\}.$$

Let $\beta_1, \ldots, \beta_m$ be binary rational numbers and $m = \mathsf{poly}(n)$ such that the size of the range of the function $\beta_1 \cdot F_1 + \ldots + \beta_m \cdot F_m$ is small, i.e.

$$(\beta_1 \cdot F_1 + \ldots + \beta_m \cdot F_m) : \{0,1\}^n \to \{\gamma_1, \ldots, \gamma_\ell\}$$

for binary rational numbers $\gamma_1, \ldots, \gamma_\ell$ and $\ell = \mathsf{poly}(n)$. Then for all sufficiently small $\epsilon = 1/\mathsf{poly}(n)$,

$$\mathbb{E}^{\epsilon}\left[\sum_{i=1}^{m} \beta_i \cdot F_i\right] = \sum_{i=1}^{m} \beta_i \cdot \mathbb{E}^{\epsilon}[F_i].$$

Proof. Then by Lemma 91 for sufficiently small $\epsilon > 0$, there exists a Nisan-Wigderson generator $\mathsf{NW}^{n,\epsilon} : \{0,1\}^t \to \{0,1\}^n$ such that

$$\begin{aligned}\mathbb{E}^{\epsilon}\left[\sum_{i=1}^{m} \beta_i \cdot F_i\right] &= \frac{1}{2^t} \sum_{S \in \{0,1\}^t} \left(\sum_{i=1}^{m} \beta_i \cdot F_i\right)(\mathsf{NW}^{n,\epsilon}(S)) \\ &= \sum_{i=1}^{m} \left(\frac{1}{2^t} \sum_{S \in \{0,1\}^t} \beta_i \cdot F_i\,(\mathsf{NW}^{n,\epsilon}(S))\right) \\ &= \sum_{i=1}^{m} \beta_i \cdot \mathbb{E}^{\epsilon}[F_i]\end{aligned}$$

□

6.1.4 Approximate variance of a feasible random variable

Let $F : \{0,1\}^n \to \{\alpha_1, \ldots, \alpha_m\}$ be a feasible random variable. Let $\mu = \mathbb{E}^\epsilon[F]$. Let $\epsilon = 1/\mathsf{poly}(n)$. Define the ϵ-approximate variance of F as

$$\mathbb{V}\mathsf{ar}^\epsilon[F] := \mathbb{E}^\epsilon[(F-\mu)^2]. \tag{6.2}$$

For all sufficiently small ϵ, it follows from linearity of expectation that

$$\mathbb{V}\mathsf{ar}^\epsilon[F] = \mathbb{E}^\epsilon[F^2] - \mu^2. \tag{6.3}$$

Lemma 93 (Chebyshev's inequality). *($HARD^A \vdash$) Let $F : \{0,1\}^n \to \{\alpha_1, \ldots, \alpha_m\}$ be a feasible random variable. Then for all sufficiently small $\epsilon = 1/\mathsf{poly}(n)$,*

$$\mathbb{P}^\epsilon_{X \in \{0,1\}^n}\left[|F(X) - \mathbb{E}^\epsilon[F]| \geq \beta\right] \leq \frac{\mathbb{V}\mathsf{ar}^\epsilon[F]}{\beta^2}$$

Proof. First we observe that

$$\{X \in \{0,1\}^n \mid |F(X) - \mathbb{E}^\epsilon[F]| \geq \beta\} = \{X \in \{0,1\}^n \mid (F(X) - \mathbb{E}^\epsilon[F])^2 \geq \beta^2\}.$$

Thus for sufficiently small $\epsilon = 1/\mathsf{poly}(n)$ we have

$$\begin{aligned}\mathbb{P}^\epsilon_{X \in \{0,1\}^n}\left[|F(X) - \mathbb{E}^\epsilon[F]| \geq \beta\right] &= \mathbb{P}^\epsilon_{X \in \{0,1\}^n}\left[(F(X) - \mathbb{E}^\epsilon[F])^2 \geq \beta^2\right] \\ &\leq \frac{\mathbb{E}^\epsilon\left[(F(X) - \mathbb{E}^\epsilon[F])^2\right]}{\beta^2} \qquad \text{(by Markov's inequality)} \\ &= \frac{\mathbb{V}\mathsf{ar}^\epsilon[F]}{\beta^2}\end{aligned}$$

□

Lemma 94 (linearity of variance). *($HARD^A \vdash$) Let $F_1, \ldots, F_m$ be feasible random variables, where $F_i : \{0,1\}^n \to \{\alpha_1, \ldots, \alpha_k\}$. Assume the size of the range of the function $F_1 + \ldots + F_m$ is small, i.e.*

$$(F_1 + \ldots + F_m) : \{0,1\}^n \to \{\gamma_1, \ldots, \gamma_\ell\}$$

for binary rational numbers $\gamma_1, \ldots, \gamma_\ell$ and $\ell = \mathsf{poly}(n)$. Let $\delta = 1/\mathsf{poly}(n)$ such that for all distict $i, j \in [m]$, $F_1, \ldots, F_m$ are δ-close to being pairwise independent, i.e.,

$$\mathbb{E}^\epsilon[F_i \cdot F_j] = \mathbb{E}^\epsilon[F_i]\mathbb{E}^\epsilon[F_i] \pm \delta$$

Then for all sufficiently small $\epsilon = 1/\mathsf{poly}(n)$,

$$\mathbb{V}\mathsf{ar}^\epsilon\left[\sum_{i=1}^m F_i\right] = \sum_{i=1}^m \mathbb{V}\mathsf{ar}^\epsilon[F_i] \pm m^2\delta.$$

Proof. For sufficiently small $\epsilon = 1/\mathsf{poly}(n)$, we can apply linearity of expectation to get

$$\mathbb{V}\mathsf{ar}^{\epsilon}\left[\sum_{i=1}^{m} F_i\right] = \sum_{i=1}^{m} \mathbb{V}\mathsf{ar}^{\epsilon}[F_i] + 2\sum_{i<j} \mathbb{C}\mathsf{ov}^{\epsilon}(F_i, F_j),$$

where

$$\mathbb{C}\mathsf{ov}^{\epsilon}(F_i, F_j) = \mathbb{E}^{\epsilon}[F_i \cdot F_j] - \mathbb{E}^{\epsilon}[F_i]\mathbb{E}^{\epsilon}[F_j] = \pm\delta.$$

From this we get

$$\mathbb{V}\mathsf{ar}^{\epsilon}\left[\sum_{i=1}^{m} F_i\right] = \sum_{i=1}^{m} \mathbb{V}\mathsf{ar}^{\epsilon}[F_i] \pm m^2\delta.$$

□

6.2 A collection of pairwise independent random variables

Our sample space is the set of all $m \times n$ 0-1 valued matrices equipped with the uniform measure. For each non-empty $S \subseteq [m]$, $F_S(M)$ is the string obtained by xor'ing together the rows of M whose index appears in S. In this section we will show that this collection of random variables is pairwise independent (which will contribute in an important way in the proof of the Goldreich-Levin Theorem).

Note that the columns of each of the F_S are independent of each other, so we start by analyzing one column at a time. In other words, we first consider the case when $n = 1$.

We will use the following notation for the rest of this section. For a nonempty $S \subseteq [m]$, and a single bit $b \in \{0,1\}$, let

$$\mathcal{X}_{S,b} = F_S^{-1}(b) = \Big\{X \in \{0,1\}^m \mid \bigoplus_{i \in S} X_i = b\Big\}.$$

We also use the notation $\mathcal{A} \approx \mathcal{B}$ to denote that there is a bijection between $\mathcal{A}$ and $\mathcal{B}$. It important to note that $\mathcal{A} \approx \mathcal{B}$ implies $\mathcal{A} \approx_0 \mathcal{B}$, but it is not known if we can prove within $HARD^A$ that $\mathcal{A} \approx_0 \mathcal{B}$ implies $\mathcal{A} \approx \mathcal{B}$.

Lemma 95. *(VPV $\vdash$) For every nonempty subset S of $[m]$ and for every $b \in \{0,1\}$,*

$$\mathcal{X}_{S,b} \approx \{0,1\}^{m-1}.$$

Informally, this proposition say that $\mathbb{P}_{X \in \{0,1\}^m}[F_S(X) = b] = |\mathcal{X}_{S,b}|/2^m = 1/2$, which is a way to show that the random bit $F_S(X) = \bigoplus_{i \in S} X_i$ is uniform.

Proof. We need to construct a bijection G_1 from $\mathcal{X}_{S,b}$ onto $\{0,1\}^{m-1}$.

Let $i \in [m]$ be the smallest index in S (i.e. $i = \min S$). Let $Y = G_1(X) \in \{0,1\}^{m-1}$ be the sequence produced from X by discarding the ith bit.

G_1 is easily seen to be injective, surjective and polytime computable. (In fact, G_1 is simply a projection function and can be computed by an NC^0 function.)

To invert G_1, we need to show how to recover the discarded bit X_i. But this is easy: since we have all but one of the bits of the sequence $\langle X_j \mid j \in S \rangle$, and their xor (i.e. b), X_i is uniquely determined as

$$X_i = \left(\bigoplus_{j \in S \setminus \{i\}} Y_j \right) \oplus b.$$

□

Lemma 96. *(VPV ⊢) For all nonempty subsets $S \neq T$ of $[m]$ and boolean values a and b,*

$$\mathcal{X}_{S,a} \cap \mathcal{X}_{T,b} \approx \{0,1\}^{m-2}.$$

Informally, this proposition say that $\mathbb{P}_{X \in \{0,1\}^m} [F_S(X) = a \wedge F_T(X) = b] = |\mathcal{X}_{S,a} \cap \mathcal{X}_{T,b}|/2^m = 1/4$, which shows that the two random bits $F_X(X)$ and $F_T(X)$ are independent.

Proof. Here we need to construct a bijection G_2 from $\mathcal{X}_{S,a} \cap \mathcal{X}_{T,b}$ onto $\{0,1\}^{m-2}$.

Since $S \neq T$, we can suppose, without loss of generality, that $T \setminus S \neq \varnothing$. Let $i = \min S$, $j = \min T \setminus S$. Let $Y = G_2(X) \in \{0,1\}^{m-1}$ be the sequence produced from X by discarding the ith and the jth bits.

G_2 is easily seen to be injective, surjective and efficiently computable. It remains to be shown that G_2 is efficiently invertible.

If we can show how to recover X_i, X_j, we are done. So we must chose X_i, X_j such that

$$\bigoplus_{k \in S} X_k = a \tag{6.4}$$

$$\bigoplus_{k \in T} X_k = b \tag{6.5}$$

The point is that since j was chosen to be an index in T but not S, there is only one unknown in (6.4), namely X_i. Once we have recovered X_i, we can solve for X_j in (6.5). □

Corollary 97. *(VPV ⊢) For all nonempty subsets $S \neq T$ of $[m]$ and boolean values a and b,*

$$(\mathcal{X}_{S,a} \cap \mathcal{X}_{T,b}) \times \{0,1\}^m \approx \mathcal{X}_{S,a} \times \mathcal{X}_{T,b}.$$

Proof. From Lemmas 95 and 96,

$$(\mathcal{X}_{S,a} \cap \mathcal{X}_{T,b}) \times \{0,1\}^m \approx \{0,1\}^{m-2} \times \{0,1\}^m \approx \{0,1\}^{m-1} \times \{0,1\}^{m-1} \approx \mathcal{X}_{S,a} \times \mathcal{X}_{T,b}.$$

□

We can extend Lemma 95 and Corollary 97 to the more general case where we generate pairwise independent binary sequences of length $n \geq 1$. For $S \subseteq [m]$, $B \in \{0,1\}^n$, let

$$\mathcal{Y}_{S,B} = F_S^{-1}(B) = \left\{ M \in \{0,1\}^{mn} \mid (\textstyle\bigoplus_{i \in S} \mathsf{Row}(i, M)) = B \right\}.$$

Lemma 98. *(VPV ⊢) For every nonempty subset S of $[m]$ and for every $B \in \{0,1\}^n$,*

$$\mathcal{Y}_{S,B} \approx \{0,1\}^{(m-1)n}.$$

Proof. We need to construct a bijection H_1 from $\mathcal{Y}_{S,B}$ onto $\{0,1\}^{(m-1)n}$. The function H_1 take as input an $m \times n$ Boolean matrix M, and produces an $(m-1) \times n$ matrix M' as follows. Each column $\mathsf{Col}(i, M')$ is the the Boolean sequence produced by the bijection from Lemma 95 on input sequence $\mathsf{Col}(i, M)$. It is not hard to check that H_1 is a bijection. □

Lemma 99. *(VPV ⊢) For all nonempty subsets $S \neq T$ of $[m]$ and boolean sequences $A, B \in \{0,1\}^n$,*

$$(\mathcal{Y}_{S,A} \cap \mathcal{Y}_{T,B}) \times \{0,1\}^{mn} \approx \mathcal{Y}_{S,A} \times \mathcal{Y}_{T,B}.$$

Proof. We need to construct a bijection H_2 from $(\mathcal{Y}_{S,A} \cap \mathcal{Y}_{T,B}) \times \{0,1\}^{mn}$ onto $\mathcal{Y}_{S,A} \times \mathcal{Y}_{T,B}$. The function H_2 takes as input a pair $(M, N) \in (\mathcal{Y}_{S,A} \cap \mathcal{Y}_{T,B}) \times \{0,1\}^{mn}$, where M and N are interpreted as $m \times n$ Boolean matrices, and produces a pair of $m \times n$ matrices M' and N' as follows. For every $i \in [n]$, the columns $\mathsf{Col}(i, M')$ and $\mathsf{Col}(i, N')$ are the two Boolean sequences produced by the bijection from Corollary 97 on input sequences $\mathsf{Col}(i, M)$ and $\mathsf{Col}(i, N)$. It is not hard to check that H_2 is a bijection. □

Now we can show that this pairwise independence guarantees approximately zero covariance.

Proposition 100. *($HARD^A$ ⊢) Let $P : \{0,1\}^n \to \{0,1\}$ and $Q : \{0,1\}^n \to \{0,1\}$ be predicates definable by circuits. Let $\epsilon = 1/\mathsf{poly}(n)$. Then for all nonempty subsets $S \neq T$ of $[m]$,*

1. $\mathbb{P}^{\epsilon}_{M\in\{0,1\}^{mn}}[P \circ F_S(M) \wedge Q \circ F_T(M)] = \mathbb{P}^{\epsilon}_{M\in\{0,1\}^{mn}}[P \circ F_S(M)]\mathbb{P}^{\epsilon}_{M\in\{0,1\}^{mn}}[Q \circ F_T(M)] \pm 4\epsilon$, *and*
2. $\mathbb{E}^{\epsilon}[(P \circ F_S) \cdot (Q \circ F_T)] = \mathbb{E}^{\epsilon}[P \circ F_S]\mathbb{E}^{\epsilon}[Q \circ F_T] \pm 4\epsilon$.

Proof. Let

$$\begin{aligned}
\mathcal{S} &= \left\{M^{m\times n} \in \{0,1\}^{mn} \mid P \circ F_S(M) \wedge Q \circ F_T(M)\right\},\\
\mathcal{U} &= \left\{M^{m\times n} \in \{0,1\}^{mn} \mid P \circ F_S(M)\right\},\\
\mathcal{V} &= \left\{M^{m\times n} \in \{0,1\}^{mn} \mid Q \circ F_T(M)\right\}.
\end{aligned}$$

We want to show the following claim.

Claim 1: $\mathcal{S} \times \{0,1\}^{mn} \approx \mathcal{U} \times \mathcal{V}$.

Proof of Claim 1. We will construct a bijection G from $\mathcal{S} \times \{0,1\}^{mn}$ onto $\mathcal{U} \times \mathcal{V}$ as follows. The input of G is a pair $(M, N) \in \mathcal{S} \times \{0,1\}^{mn}$, where M and N are interpreted as $m \times n$ Boolean matrices. Let $A = F_S(M)$ and $B = F_T(M)$. Then $M \in \mathcal{Y}_{S,A} \cap \mathcal{Y}_{T,B}$, and thus we can let G map (M, N) to a pair $(M', N') \in \mathcal{Y}_{S,A} \times \mathcal{Y}_{T,B}$ using the bijection from Lemma 99. It is not hard to check that G is also a bijection. □

We also have $\mathcal{S} \approx_{\epsilon} \left[\mathsf{Size}(\mathcal{S}, \epsilon)\right)$, $\mathcal{U} \approx_{\epsilon} \left[\mathsf{Size}(\mathcal{U}, \epsilon)\right)$, and $\mathcal{V} \approx_{\epsilon} \left[\mathsf{Size}(\mathcal{V}, \epsilon)\right)$. Thus,

$$\begin{aligned}
\left[\mathsf{Size}(\mathcal{S}, \epsilon) \cdot 2^{mn}\right) &\approx_{\epsilon} \mathcal{S} \times \{0,1\}^{mn} \\
&\approx_{0} \mathcal{U} \times \mathcal{V} && \text{(by Claim 1)}\\
&\approx_{2\epsilon+\epsilon^2} \left[\mathsf{Size}(\mathcal{U}, \epsilon)\mathsf{Size}(\mathcal{V}, \epsilon)\right) && \text{(by Lemma 13 (5))}
\end{aligned}$$

By Lemma 13 (3), it follows that

$$\left[\mathsf{Size}(\mathcal{S}, \epsilon) \cdot 2^{mn}\right) \approx_{3\epsilon+\epsilon^2} \left[\mathsf{Size}(\mathcal{U}, \epsilon)\mathsf{Size}(\mathcal{V}, \epsilon)\right).$$

Then we can apply Lemma 14 (2) to get

$$\mathsf{Size}(\mathcal{S},\epsilon)\cdot 2^{mn} = \mathsf{Size}(\mathcal{U},\epsilon)\mathsf{Size}(\mathcal{V},\epsilon) \pm (3\epsilon+\epsilon^2+\gamma)2^{2mn}.$$

By choosing $\gamma \le \epsilon - \epsilon^2$, we have

$$\mathsf{Size}(\mathcal{S},\epsilon)\cdot 2^{mn} = \mathsf{Size}(\mathcal{U},\epsilon)\mathsf{Size}(\mathcal{V},\epsilon) \pm 4\epsilon 2^{2mn}.$$

This implies part (1) of this proposition. Since $P \circ F_S$ and $Q \circ F_T$ are 0-1 valued random variables, part (2) also follows. □

The following proposition is also useful when dealing with pairwise independent sequences since we want to show that running a randomized algorithm on a pairwise independent distribution constructed above will produce "identical" results to running on a uniform distribution.

Proposition 101. *($HARD^A$ ⊢) Let $P : \{0,1\}^n \to \{0,1\}$ be a predicate definable by a circuit. Let $\epsilon = 1/\mathsf{poly}(n)$. Then for every nonempty $S \subseteq [m]$,*

$$\mathbb{P}^{\epsilon}_{M\in\{0,1\}^{mn}}[P\circ F_S(M)] = \mathbb{P}^{\epsilon}_{X\in\{0,1\}^n}[P(X)] \pm 3\epsilon.$$

Proof. Let $\mathcal{U} = \{M^{m\times n} \in \{0,1\}^{mn} \mid P\circ F_S(M)\}$ and $\mathcal{V} = \{X \in \{0,1\}^n \mid P(X)\}$. By Corollary 87 (6), it suffices to show that $\mathcal{U} \approx \mathcal{V} \times \{0,1\}^{m(n-1)}$.

We want to construct a bijection G from $\mathcal{U}$ onto $\mathcal{V}\times\{0,1\}^{m(n-1)}$. On input M of G, we let $B = F_S(M)$. Since $M \in \mathcal{Y}_{S,B}$, we can let $M' = H_1(M)$, where H_1 is the bijection in the proof of Lemma 98. Then we define $G(M) = (B, M')$. We can easily check that G is a bijection. □

6.3 Formalizing the Goldreich-Levin Theorem

The Goldreich-Levin Theorem is the result that every one-way function can be modified to have a hardcore predicate. The informal statement without specifying the approximation parameters can be stated as follows.

Theorem 102 (Goldreich-Levin [24]). *Let $F : \{0,1\}^n \to \{0,1\}^n$ be a one-way function and let $F' : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}^n \times \{0,1\}^n$ be the following function $F'(X,R) = (F(X),R)$. Define*

$$\mathsf{gl}(X,R) = \bigoplus_{i=1}^{n} X_i \cdot R_i.$$

Then F' is also a one-way function and gl is a hardcore predicate for F'.

In other words, the theorem says that if F is a one-way function, and we pick a random sequence $X \in \{0,1\}^n$ and a random subset $S \subseteq [n]$, then given the value of $F(X)$ and the set S it is hard for the adversary to guess the value $\bigoplus_{i\in S} X(i)$ with probability significantly better than $1/2$.

The proof we present here is due to Charles Rackoff, and is a simplification of the original proof in [24]. Our formal proof of this theorem will be based on the excellent exposition in [31].

6.3.1 Formalizing basic cryptography definitions

Definition 103 (One-way function – finite version)**.** (in $HARD^A$) A function $F : \{0,1\}^n \to \{0,1\}^n$ is (s,ϵ)-one-way if it is computable by a circuit of size $\mathsf{poly}(n)$, but for every (randomized) circuit $C : \{0,1\}^n \times \{0,1\}^r \to \{0,1\}$ of size at most s, there exits $\gamma > 0$ such that for all $0 < \delta < \gamma$ with $\delta = 1/\mathsf{poly}(n,r)$,

$$\mathbb{P}^{\delta}_{(X,R)\in\{0,1\}^n\times\{0,1\}^r}\Big[F\big(C(F(X),R)\big) = F(X)\Big] < \epsilon.$$

A function $F : \{0,1\}^n \to \{0,1\}^n$ which is both (s,ϵ)-one-way and bijective is called an (s,ϵ)*-one-way permutation*.

Using the above definition, we can show the following property of one-way functions.

Proposition 104. *($HARD^A$ $\vdash$) If $F : \{0,1\}^n \to \{0,1\}^n$ is (s,ϵ)-one-way with $s = \mathsf{poly}(n)$ and $\epsilon = 1/\mathsf{poly}(n)$, then the function $F' : \{0,1\}^n \times \{0,1\}^q \to \{0,1\}^n \times \{0,1\}^q$ defined as $F'(X,R) = (F(X),R)$ is also (s,ϵ)-one-way.*

Proof. We prove the contrapositive. Suppose F' is not (s,ϵ)-one-way. Then by Definition 103 there exists a circuit C of size at most s such that for all $\gamma > 0$, there exists $0 < \delta < \gamma$ with $\delta = 1/\mathsf{poly}(n,q,r)$,

$$\mathbb{P}^{\delta}_{(X,R,R')\in\{0,1\}^n\times\{0,1\}^q\times\{0,1\}^r}\Big[F'\big(C(F'(X,R),R')\big) = (F(X),R)\Big] \geq \epsilon.$$

For sufficiently small $\delta = 1/\mathsf{poly}(n,q,r)$, this together with Corollary 86 (1) implies that

$$\mathbb{P}^{\delta}_{(X,R)\in\{0,1\}^n\times\{0,1\}^{q+r}}\Big[F\big(C(F(X),R)\big) = F(X)\Big] \geq \epsilon,$$

where F simply acts on the first n output bits of C. Thus, $F : \{0,1\}^n \to \{0,1\}^n$ is also not (s,ϵ)-one-way. □

Definition 105 (Hardcore predicate – finite version)**.** (in $HARD^A$) A predicate $B : \{0,1\}^n \to \{0,1\}$ is an (s,ϵ)-hardcore predicate for a function $F : \{0,1\}^n \to \{0,1\}^n$ if it is computable by a circuit of size $\mathsf{poly}(n)$, but for every (randomized) circuit $C : \{0,1\}^n \times \{0,1\}^r \to \{0,1\}$ of size at most s, there exits $\gamma > 0$ such that for all $0 < \delta < \gamma$ with $\delta = 1/\mathsf{poly}(n,r)$,

$$\mathbb{P}^{\delta}_{(X,R)\in\{0,1\}^n\times\{0,1\}^r}\big[C(F(X),R) = B(X)\big] < 1/2 + \epsilon.$$

6.3.2 A special case of the Goldreich-Levin Theorem

In this section, we will formalize the following weaker version of the Goldreich-Levin Theorem, which we state in the contrapositive.

Theorem 106 (Goldreich-Levin – weak version)**.** *Let $F : \{0,1\}^n \to \{0,1\}^n$ be a function computed by a circuit of size t, and suppose that there exists a circuit C of size s such that*

$$\mathbb{P}^{\epsilon}_{(X,R)\in\{0,1\}^{2n}}\big[C(F(X),R) = \mathsf{gl}(X,R)\big] \geq \frac{3}{4} + \frac{1}{p(n)}. \tag{6.6}$$

If $\epsilon = 1/\mathsf{poly}(n)$ is sufficiently small, then there is a circuit C' of size at most $s \cdot \mathsf{poly}(n, 1/\epsilon)$ and $q = \mathsf{poly}(n)$ such that

$$\mathbb{P}^{\epsilon}_{(X,R')\in\{0,1\}^n\times\{0,1\}^q}\big[C'(F(X),R') = X\big] \geq \frac{1}{4p(n)} - \frac{15\epsilon}{2} \tag{6.7}$$

Note that (6.7) automatically implies by Corollary 86 (1) that

$$\mathbb{P}^{\epsilon}_{(X,R')\in\{0,1\}^n\times\{0,1\}^q}\left[F(C'(F(X),R'))=F(X)\right]\geq\frac{1}{4p(n)}-\frac{15\epsilon}{2}$$

This version of the theorem is weak in the sense that the right-hand side of (6.6) is $3/4+\frac{1}{p(n)}$ instead of $1/2+\frac{1}{p(n)}$. The main observation is that for every $R\in\{0,1\}^n$, the values $\mathsf{gl}(X,R)$ and $\mathsf{gl}(X,R\oplus E^i)$ together determine $X(i)$ (here E^i denotes the n-bit binary string with zeros in all but the i^{th} position). This follows from the property of the $\oplus$ operator, which implies the following equality:

$$\mathsf{gl}(X,R)\oplus\mathsf{gl}(X,R\oplus E^i)=X(i). \tag{6.8}$$

Thus if C answers correctly on both $(F(X),R)$ and $(F(X),R\oplus E^i)$, then we can correctly compute $X(i)$ using the identity (6.8). However we are only guaranteed that C computes gl correctly with high probability, so we are left with a noisy guess for $X(i)$. To overcome this, we take a majority of polynomially many noisy guesses, which will result in a correct guess of $X(i)$ with sufficiently high probability.

Note that the probability that C computes gl correctly is over both X and R. As we are building an inverter for F, we will be asked to invert F for a particular X. So C will only be useful if it answers $\mathsf{gl}(X,R)$ correctly for the same X and many R. Thus if we let $Q(X,R)$ be the predicate:

$$Q(X,R):=(C(F(X),R)=\mathsf{gl}(X,R))$$

then we will be interested in X's such that $\mathbb{P}^{\epsilon}_{R\in\{0,1\}^n}\left[Q(X,R)\right]$ is large. Thanks to Proposition 89, we have the following lemma:

Lemma 107. *($HARD^A\vdash$) For sufficiently small $\epsilon=1/\mathsf{poly}(n)$, if*

$$\mathbb{P}^{\epsilon}_{(X,R)\in\{0,1\}^{2n}}\left[Q(X,R)\right]\geq\frac{3}{4}+\frac{1}{p(n)}, \tag{6.9}$$

then we have

$$\mathbb{P}^{\epsilon}_{X\in\{0,1\}^n}\left[\mathbb{P}^{\epsilon}_{R\in\{0,1\}^n}\left[Q(X,R)\right]\geq\frac{3}{4}+\frac{1}{2p(n)}\right]\geq\frac{1}{2p(n)}-7\epsilon.$$

□

For the task of inverting F, it will suffice to invert F on X's where

$$\mathbb{P}^{\epsilon}_{R\in\{0,1\}^n}\left[Q(X,R)\right]\geq\frac{3}{4}+\frac{1}{2p(n)}$$

(as, when X is chosen uniformly at random, this occurs with significant probability). Let $P(X)$ be the predicate that is true when the above inequality holds, i.e.

$$P(X):=\left(\mathbb{P}^{\epsilon}_{R\in\{0,1\}^n}\left[Q(X,R)\right]\geq\frac{3}{4}+\frac{1}{2p(n)}\right)$$

For the purposes of exposition and as suggested in the sketch of the proof of 106, C' is constructed primarily from C'_i sub-circuits, each of which guesses $X(i)$ with considerable accuracy; in turn, C'_i is constructed by taking the majority vote from m copies of circuit C_i^{noisy}, each of which takes $F(X)$ and a random string $R \in \{0,1\}^n$ as inputs and uses the circuit C and the identity (6.8) to produce a noisy guess for $X(i)$. In other words, we define

$$C'_i(F(X), R_1, \ldots, R_m) = \mathsf{Maj}\left(C_i^{noisy}(F(X), R_1), \ldots, C_i^{noisy}(F(X), R_m)\right)$$
$$C'_i(F(X), R_1, \ldots, R_m) := \mathsf{Maj}\left(C_i^{noisy}(F(X), R_1), \ldots, C_i^{noisy}(F(X), R_m)\right)$$
$$C_i^{noisy}(F(X), R_j) := C(F(X), R_j) \oplus C(F(X), R \oplus E^i)$$

We begin by showing how well C_i^{noisy} guesses $X(i)$. This is the probability that C answers correctly for both $(F(X), R)$ and $(F(X), R \oplus E^i)$.

Lemma 108. *($HARD^A \vdash$) For sufficiently small $\epsilon = 1/\mathsf{poly}(n)$, and for every X satisfying $P(X)$ we have*

$$\mathbb{P}^{\epsilon}_{R\in\{0,1\}^n}\left[Q(X,R) \wedge Q(X, R \oplus E^i)\right] \geq \left(\frac{1}{2} + \frac{1}{p(n)}\right) - 3\epsilon.$$

Proof. Let $\mathcal{B}$ denote the set $\{R \in \{0,1\}^n \mid \neg Q(X,R) \vee \neg Q(X, R \oplus E^i)\}$, which contains the "bad" sequences R's. We want to upper-bound the probability that R is a bad sequence.

Since

$$\mathcal{B} = \left\{R \in \{0,1\}^n \mid \neg Q(X,R)\right\} \cup \left\{R \in \{0,1\}^n \mid \neg Q(X, R \oplus E^i)\right\},$$

it follows by Corollary 86 (3) (union bound) that

$$\mathbb{P}^{\epsilon}_{R\in\{0,1\}^n}[X \in \mathcal{B}] \leq \mathbb{P}^{\epsilon}_{R\in\{0,1\}^n}[\neg Q(X,R)] + \mathbb{P}^{\epsilon}_{R\in\{0,1\}^n}[\neg Q(X, R \oplus E^i)].$$

We next upper-bound the two probabilities in this sum.

It follows from the assumption of this lemma and Corollary 86 (5), we have

$$\mathbb{P}^{\epsilon}_{R\in\{0,1\}^n}[\neg Q(X,R)] \leq \left(\frac{1}{4} - \frac{1}{2p(n)}\right). \tag{6.10}$$

To bound $\mathbb{P}^{\epsilon}_{R\in\{0,1\}^n}[\neg Q(X, R \oplus E^i)]$, we observe that

$$\{R \in \{0,1\}^n \mid \neg Q(X, R \oplus E^i)\} \approx \{R \in \{0,1\}^n \mid \neg Q(X,R)\},$$

where we recall that $\approx$ denotes the existence of a bijection between these two sets. Thus we can apply Corollary 87 (2) (with $\gamma = \epsilon$) to get

$$\mathbb{P}^{\epsilon}_{R\in\{0,1\}^n}[\neg Q(X, R \oplus E^i)] \leq \mathbb{P}^{\epsilon}_{R\in\{0,1\}^n}[\neg Q(X,R)] + 3\epsilon = \left(\frac{1}{4} - \frac{1}{2p(n)}\right) + 3\epsilon. \tag{6.11}$$

From (6.10) and (6.11), we have

$$\begin{aligned}\mathbb{P}^{\epsilon}_{R\in\{0,1\}^n}[R\in\mathcal{B}] &\le \mathbb{P}^{\epsilon}_{R\in\{0,1\}^n}[\neg Q(X,R)] + \mathbb{P}^{\epsilon}_{R\in\{0,1\}^n}[\neg Q(X,R\oplus E^i)]\\ &\le \left(\frac{1}{4}-\frac{1}{2p(n)}\right)+\left(\frac{1}{4}-\frac{1}{2p(n)}\right)+3\epsilon\\ &= \left(\frac{1}{2}-\frac{1}{p(n)}\right)+3\epsilon.\end{aligned}$$

Hence, by Corollary 86 (5), we can lower-bound the probablity that R is a good sequence as follows

$$\mathbb{P}^{\epsilon}_{R\in\{0,1\}^n}[X\notin\mathcal{B}] \ge 1-\left(\frac{1}{2}-\frac{1}{p(n)}\right)-3\epsilon = \left(\frac{1}{2}+\frac{1}{p(n)}\right)-3\epsilon.$$

□

Now we will consider how well C'_i recovers $X(i)$. Recall that

$$C'_i(F(X),\vec{R}) := \mathsf{Maj}\left(C^{noisy}_i(F(X),R_1),\ldots,C^{noisy}_i(F(X),R_m)\right),$$

where $\vec{R}=\langle R_1,\ldots,R_m\rangle$.

Lemma 109. *Given X satisfying $P(X)$. For sufficiently small $\epsilon = 1/\mathsf{poly}(n)$ and sufficiently large $m = \mathsf{poly}(n)$,*

$$\mathbb{P}^{\epsilon}_{\vec{R}\in\{0,1\}^{nm}}\left[C'_i(F(X),\vec{R})\neq X(i)\right]\le\frac{2}{n^2}$$

Proof. Let $G(R):\{0,1\}^n\to\{0,1\}$ be the indicator random variable for the event that $C^{noisy}_i(F(X),R)$ guesses the value of $X(i)$ correctly (i.e. that $Q(X,R)\wedge Q(X,R\oplus E^i)$ holds). Define the random variable

$$Z(R_1,\ldots,R_m)=\sum_{j=1}^{m}G(R_j).$$

Note that C'_i errs if and only if $Z<m/2$. As we will use Chebyshev's inequality to bound this probability, we will need information about the moments of Z. We need to show the following two facts:

$$\mathbb{E}^{\epsilon}[Z]\ge m/2+m/p(n)-3m\epsilon \tag{6.12}$$

$$\mathbb{V}\mathsf{ar}^{\epsilon}[Z]\le m/4+m^2 4\epsilon \tag{6.13}$$

Inequality (6.12) follows directly from Lemma 92 and Lemma 108.

Intuitively, since Z is a sum of independent random variables, (6.13) will follow from Lemma 94. We proceed with the detailed proof. It suffices to show that the approximate covariance of $\{G(R_j)\}_{j=1}^m$ is within 4ϵ. We note that

$$\begin{aligned}&\{(R_j,R_k)\in\{0,1\}^n\times\{0,1\}^n \mid G(R_j)=1\wedge G(R_k)=1\}\\ \approx_0&\{R_j\in\{0,1\}^n\mid G(R_j)=1\}\times\{R_k\in\{0,1\}^n\mid G(R_k)=1\}\end{aligned}$$

Following Corollary 87 (4), we have that

$$\mathbb{E}^{\epsilon}[G(R_j)G(R_k)] = \mathbb{E}^{\epsilon}[G(R_k)]\mathbb{E}^{\epsilon}[G(R_k)] \pm 4\epsilon$$

Applying Lemma 94 we have that

$$\mathbb{V}\mathsf{ar}^{\epsilon}[Z] = m\mathbb{V}\mathsf{ar}^{\epsilon}[G] + 4m^2\epsilon$$

To upperbound $\mathbb{V}\mathsf{ar}^{\epsilon}[G]$, we start with (6.3) which simplifies to

$$\mathbb{V}\mathsf{ar}^{\epsilon}[G] = \mathbb{E}^{\epsilon}[G^2] - (\mathbb{E}^{\epsilon}[G])^2 = \mathbb{E}^{\epsilon}[G] - (\mathbb{E}^{\epsilon}[G])^2$$

since $G(R)$ is an indicator random variable (i.e. 0-1 valued). Now consider the function $f(\gamma) = \gamma - \gamma^2$, where $\gamma = \mathbb{E}^{\epsilon}[G]$. By rewriting

$$f(\gamma) = -(\gamma - 1/2)^2 + 1/4,$$

it is clear that f takes its maximum at $1/4$. So we get $\mathbb{V}\mathsf{ar}^{\epsilon}[G] \leq 1/4$. Thus, (6.13) follows.

We can bound the event that $Z(\vec{R}) < m/2$ as follows. We rewrite

$$\mathbb{P}^{\epsilon}_{\vec{R}\in\{0,1\}^{nm}}\left[Z(\vec{R}) < m/2\right] = \mathbb{P}^{\epsilon}\left[Z(\vec{R}) - (m/2 + m/p(n) - 3m\epsilon) < -(m/p(n) - 3m\epsilon)\right]$$

Then for sufficiently small ϵ, we have that $-(m/p(n) - 3m\epsilon) < 0$, and thus

$$\begin{aligned}
\mathbb{P}^{\epsilon}_{\vec{R}\in\{0,1\}^{nm}}\left[Z(\vec{R}) < m/2\right] &\leq \mathbb{P}^{\epsilon}\left[Z(\vec{R}) - \mathbb{E}^{\epsilon}[Z] < -(m/p(n) - 3m\epsilon)\right] && \text{(by (6.12))}\\
&\leq \mathbb{P}^{\epsilon}\left[\left|Z(\vec{R}) - \mathbb{E}^{\epsilon}[Z]\right| > (m/p(n) - 3m\epsilon)\right]\\
&\leq \frac{\mathbb{V}\mathsf{ar}^{\epsilon}[Z]}{(m/p(n) - 3m\epsilon)^2}\\
&\leq \frac{m/4 + 4m^2\epsilon}{m^2(1/p(n) - 3\epsilon)^2}\\
&= \frac{1}{4m(1/p(n) - 3\epsilon)^2} + \frac{4\epsilon}{(1/p(n) - 3\epsilon)^2}\\
&\leq \frac{1}{4m\left(\frac{1}{2p(n)}\right)^2} + \frac{4\epsilon}{\left(\frac{1}{2p(n)}\right)^2} && \text{(by choosing } \epsilon \leq \tfrac{1}{6p(n)}\text{)}\\
&= \frac{p^2(n)}{m} + 16\epsilon p^2(n)
\end{aligned}$$

Note that when $m \geq n^2p^2(n)$ and $\epsilon \leq \dfrac{1}{(4np(n))^2}$, we get

$$\mathbb{P}^{\epsilon}_{\vec{R}\in\{0,1\}^{nm}}[C'_i(F(X), \vec{R}) \neq X(i)] \leq \frac{1}{n^2} + \frac{1}{n^2} = \frac{2}{n^2}$$

□

From here, we can now show the correctness of C' inverting F on 'good' X's.

Lemma 110. *For sufficiently small $\epsilon = 1/\mathsf{poly}(n)$ and sufficiently large n and $m = \mathsf{poly}(n)$, if X satisfies*

$P(X)$, then

$$\mathbb{P}^{\epsilon}_{\vec{R}'\in\{0,1\}^{n^2m}}[C'(F(X),\vec{R}') \neq X] < 1/2$$

where $\vec{R}' = \left\langle \vec{R}_1, ..., \vec{R}_n \right\rangle$ are the random strings used by C', and each $\vec{R}_i$ consists of nm bits used by circuit C'_i.

Proof. Since we can recover each bit with high probability, it only remains to show that we recover all bits with high probability, which comes to us through the union bound (and our judicious choice of m and ϵ).

$$\begin{aligned}\mathbb{P}^{\epsilon}_{\vec{R}\in\{0,1\}^{n^2m}}\left[C'(F(X),\vec{R}') \neq X\right] &= \mathbb{P}^{\epsilon}_{\vec{R}\in\{0,1\}^{n^2m}}\left[\bigvee_{i=1}^{n} C'_i(F(X),\vec{R}_i) \neq X(i)\right] \\ &\leq \sum_{i=1}^{n}\mathbb{P}^{\epsilon}_{\vec{R}\in\{0,1\}^{n^2m}}[C'_i(F(X),\vec{R}_i) \neq X(i)] \qquad \text{(by Proposition 86 (3))}\end{aligned}$$

At this point, we would like to apply Lemma 109 to upperbound each $\mathbb{P}^{\epsilon}_{\vec{R}\in\{0,1\}^{n^2m}}[C'_i(F(X),\vec{R}_i) \neq X(i)]$. However, before doing so, we need to apply Corollary 87 (6) to get

$$\mathbb{P}^{\epsilon}_{\vec{R}\in\{0,1\}^{n^2m}}[C'_i(F(X),\vec{R}_i) \neq X(i)] = \mathbb{P}^{\epsilon}_{\vec{R}_j\in\{0,1\}^{nm}}[C'_i(F(X),\vec{R}_j) \neq X(i)] \pm 3\epsilon$$

Putting things together, we have

$$\begin{aligned}\mathbb{P}^{\epsilon}_{\vec{R}\in\{0,1\}^{n^2m}}\left[C'(F(X),\vec{R}') \neq X\right] &\leq \left(\sum_{i=1}^{n}\mathbb{P}^{\epsilon}_{\vec{R}\in\{0,1\}^{n^2m}}\left[C'_i(F(X),\vec{R}_i) \neq X(i)\right]\right) + 3n\epsilon \\ &\leq n\left(\frac{2}{n^2}\right) + 3n\epsilon \qquad \text{(by Lemma 109)} \\ &= \frac{2}{n} + 3n\epsilon \\ &< 1/2\end{aligned}$$

for sufficiently small $\epsilon = 1/\mathsf{poly}(n)$ and large enough m and n. □

Now the proof of Theorem 106 follows easily from this development and the earlier Averaging Principle, Theorem 88 (2).

Proof of Theorem 106. Let

$$\begin{aligned}\mathcal{I} &= \left\{(X,R) \in \{0,1\}^n \times \{0,1\}^{n^2m} \mid C'(F(X),R) = X \wedge P(X)\right\} \\ \mathcal{I}_X &= \left\{R \in \{0,1\}^{n^2m} \mid C'(F(X),R) = X\right\} \\ \mathcal{P} &= \{X \in \{0,1\}^n \mid P(X)\}\end{aligned}$$

Lemma 107 gives us that

$$\mathbb{P}^{\epsilon}_{X\in\{0,1\}^n}[X \in \mathcal{P}] \geq \frac{1}{2p(n)} - 7\epsilon.$$

Hence, $\left[2^n \cdot (\frac{1}{2p(n)} - 7\epsilon)\right) \precsim_{\epsilon} \mathcal{P}$.

Meanwhile, for $X \in \mathcal{P}$, Lemma 110 gives us that

$$\mathbb{P}^{\epsilon}_{R \in \{0,1\}^{n^2 m}}[C'(F(X), R) = X] \geq 1/2.$$

It follows that $[2^{n^2 m - 1}) \precsim_{\epsilon} \mathcal{I}_X$.

Applying Theorem 88 (2) (with $\mathcal{I}$ playing the role of $\mathcal{X}$ and $\mathcal{P}$ playing the role of $\mathcal{Y}$, we have that

$$\left[\left(\frac{1}{4p(n)} - \frac{7\epsilon}{2}\right) \times 2^{n^2 m}\right) \precsim_{2\epsilon + \epsilon^2 + \gamma} \mathcal{I}$$

Thus we have the upperbound

$$\mathbb{P}^{\epsilon}_{(X,R) \in \{0,1\}^n \times \{0,1\}^{n^2 m}}[(X,R) \in \mathcal{I}] \geq \left(\frac{1}{4p(n)} - \frac{7\epsilon}{2}\right) - 4\epsilon = \left(\frac{1}{4p(n)} - \frac{15\epsilon}{2}\right)$$

by choosing $\gamma = \epsilon - \epsilon^2$.

Given that $\mathcal{I} \subseteq \left\{(X,R) \in \{0,1\}^n \times \{0,1\}^{n^2 m} \mid C'(F(X),R) = X\right\}$, the set whose probability we are computing, the result follows by Corollary 86 (1). □

6.3.3 The Goldreich-Levin Theorem

We are now ready to proceed with the full version of the Goldreich-Levin Theorem.

Theorem 111 (Goldreich-Levin)**.** *Let $F : \{0,1\}^n \to \{0,1\}^n$ be a function computed by a circuit of size t, and suppose that there exists a circuit C of size s such that*

$$\mathbb{P}^{\epsilon}_{(X,R) \in \{0,1\}^{2n}}\left[C(F(X),R) = \mathsf{gl}(X,R)\right] \geq \frac{1}{2} + \frac{1}{p(n)} \tag{6.14}$$

If $\epsilon = 1/\mathsf{poly}(n)$ is sufficiently small, then there is a circuit C' of size at most $(s+t) \cdot \mathsf{poly}(n, 1/\epsilon)$ and $q = \mathsf{poly}(n)$ such that

$$\mathbb{P}^{\epsilon}_{(X,R') \in \{0,1\}^n \times \{0,1\}^q}\left[F(C'(F(X),R')) = F(X)\right] \geq \frac{1}{4p(n)} - \frac{15\epsilon}{2}$$

For the general case of the Goldreich-Levin Theorem we have a weaker predictor of the gl predicate. If we proceed similarly to the proof of Lemma 108, we can only get

$$\mathbb{P}^{\epsilon}_{R \in \{0,1\}^n}\left[Q(X,R) \wedge Q(X, R \oplus E^i)\right] \geq \frac{1}{p(n)} - O(\epsilon).$$

This means that if we construct a similar circuit that guesses $X(i)$ then we can only guarantee that the probability that the circuit succeeds is at least $\frac{1}{p(n)}$, which is no better than randomly guessing the value of X_i.

To overcome this, we will modify C_i^{noisy} to query only $C(X, R_j \oplus E^i)$, and xor this with a guess of $\mathsf{gl}(X, R_j)$. The naive way to make these guesses would be choosing the $R_1, \ldots, R_m$ independently as before and making an independent guess of $\mathsf{gl}(X, R_j)$ for each R_j. Unfotunately the probabilty that all guesses are correct would be negligible because m is polynomial in n.

They key idea is that we can generate the R_j's in a pairwise-independent manner as follows. In order to generate $R_1, \dots, R_m$, we choose $\ell = \log(m+1)$ independent and uniformly-distributed strings $Y_1, \dots, Y_\ell \in \{0,1\}^n$. Let $S_1, \dots, S_m$ be an enumeration of all nonempty subsets of $[\ell]$. Then we define $R_j = \bigoplus_{i \in S_j} Y_i$. It follows from the results we formalized in Section 6.2 that the collection of strings $R_1, \dots, R_m$ is pairwise-independent and each string R_j when considered in isolation is uniformly distributed.

It remains to guess correctly the values of $\mathsf{gl}(X, R_1). \dots, \mathsf{gl}(X, R_m)$. For this purpose we exploit the linearity of gl:

> if we have the correct values of $\mathsf{gl}(X, Y_1), \dots, \mathsf{gl}(X, Y_\ell)$ and let $b_j = \bigoplus_{i \in S_j} \mathsf{gl}(X, Y_j)$, then b_j is the correct value for $\mathsf{gl}(X, R_j)$.

Thus it suffices to guess correctly the values of $\mathsf{gl}(X, Y_1), \dots, \mathsf{gl}(X, Y_\ell)$. But since ℓ is $O(\log n)$, we can eventually get the correct values of $\mathsf{gl}(X, Y_1), \dots, \mathsf{gl}(X, Y_\ell)$ by simply going through all possible binary strings B of length ℓ, where $B(k)$ is a guess for the value of $\mathsf{gl}(X, Y_k)$, and letting a circuit C'_B carry out the computation using B as the values of $\mathsf{gl}(X, Y_1), \dots, \mathsf{gl}(X, Y_\ell)$. Furthermore we can verify if C'_B succeeds by checking if $\hat{X} = C'_B\big(F(X), R_1, R_2, \dots, R_m\big)$ satisfies

$$F(\hat{X}) = F(X).$$

Hence we can go through all possible strings B of length ℓ and output the first $\hat{X}$ satisfying $F(\hat{X}) = F(X)$. If we fail to find any such $\hat{X}$, we will output some arbitrary string, say $\vec{0}$.

Thus for the rest of the proof, we can assume for simplicity that the circuit C' is provided with the correct values of

$$\mathsf{gl}(X, R_1), \dots, \mathsf{gl}(X, R_m).$$

Our exposition follows that of Theorem 106 closely. The parameter m will be fixed later. So we have in mind that:

$$\begin{aligned} C'(F(X), R_1, \dots, R_m) &:= \big(C'_1(F(X), R_1, \dots, R_m), \dots, C'_n(F(X), R_1, \dots, R_m)\big) \\ C'_i(F(X), R_1, \dots, R_m) &:= \mathsf{Maj}\left(C_i^{noisy}(F(X), R_1), \dots, C_i^{noisy}(F(X), R_m)\right) \\ C_i^{noisy}(F(X), R_j) &:= \mathsf{gl}(X, R_j) \oplus C(F(X), R_j \oplus E_i) \end{aligned}$$

where $R_1, \dots, R_m$ are generated in a pairwise-independent manner as discussed above.

Similar to as before, we define the predicate $P : \{0,1\}^n \to \{0,1\}$ as

$$P(X) := \left(\mathbb{P}^{\epsilon}_{R \in \{0,1\}^n}[Q(X,R)] \geq \frac{1}{2} + \frac{1}{2p(n)}\right)$$

Similarly to Lemma 107 we have

$$\mathbb{P}^{\epsilon}_{X \in \{0,1\}^n}[P(X)] \geq \frac{1}{2p(n)} - 7\epsilon \tag{6.15}$$

Lemma 112. *Let X be fixed such that $P(X)$. For sufficiently small $\epsilon = 1/\mathsf{poly}(n)$ and sufficiently large m,*

$$\mathbb{P}^{\epsilon}_{\vec{Y}\in\{0,1\}^{n\ell}}\left[C'_i(F(X),R_1,\ldots,R_m) \neq X(i)\right] \leq \frac{2}{n^2}$$

Proof. Let $G(R)$ be the indicator random variable for the event that $C_i^{noisy}(F(X),R)$ guesses $X(i)$ correctly (i.e. that $Q(X,R)$ holds). Let Z be the random variable over $\vec{Y} = \langle Y_1,\ldots,Y_\ell\rangle \in \{0,1\}^{n\ell}$ where

$$Z(\vec{Y}) = \sum_{j=1}^{m} G(R_j)$$

and recall that $R_1,\ldots,R_m$ are the pairwise independent strings generated from the uniformly-distributed strings $Y_1,\ldots,Y_\ell \in \{0,1\}^n$.

Note that C'_i errs iff $Z(\vec{R}) < M/2$. As we will use Chebyshev's inequality to bound this probability, we will need information about the moments of Z. The following facts will suffice:

$$\mathbb{E}^{\epsilon}[Z] \geq \frac{m}{2} + \frac{m}{2p(n)} - 6m\epsilon \tag{6.16}$$

$$\mathbb{V}\mathsf{ar}^{\epsilon}[Z] \leq \frac{m}{4} + 4\epsilon m^2 \tag{6.17}$$

As Z is a sum of random variables, inequality (6.16) follows directly from Lemma 92, the fact that X satisfies $P(X)$, and an application of Proposition 101 since we are working with pairwise independent strings R_j.

Inequality (6.17) follows from Lemma 94, where the bound on covariances comes from Proposition 100, and we also get $\mathbb{V}\mathsf{ar}^{\epsilon}[G] \leq 1/4$ as in the proof of Lemma 109.

Now we proceed similarly as in the proof of Lemma 109 as follows

$$\begin{aligned}
\mathbb{P}^{\epsilon}[Z(\vec{R}) < m/2] &\leq \mathbb{P}^{\epsilon}\left[Z(\vec{R}) - \left(\frac{m}{2} + \frac{m}{2p(n)} - 6m\epsilon\right) < -\left(\frac{m}{2p(n)} - 6m\epsilon\right)\right] \\
&\leq \mathbb{P}^{\epsilon}\left[|Z(\vec{R}) - \mathbb{E}^{\epsilon}[Z]| > \frac{m}{2p(n)} - 6m\epsilon\right] \\
&\leq \frac{\mathbb{V}\mathsf{ar}^{\epsilon}[Z]}{\left(\frac{m}{2p(n)} - 6m\epsilon\right)^2} \\
&\leq \frac{m/4 + 4\epsilon m^2}{\left(\frac{m}{2p(n)} - 6m\epsilon\right)^2} \\
&= \frac{1}{4m\left(\frac{1}{2p(n)} - 6\epsilon\right)^2} + \frac{4\epsilon}{\left(\frac{1}{2p(n)} - 6\epsilon\right)^2} \\
&\leq \frac{1}{4m\left(\frac{1}{4p(n)}\right)^2} + \frac{4\epsilon}{\left(\frac{1}{4p(n)}\right)^2} && \text{(by choosing } \epsilon \leq \tfrac{1}{24p(n)}\text{)} \\
&= \frac{4p^2(n)}{m} + 64\epsilon p^2(n)
\end{aligned}$$

For $m \geq 4n^2p^2(n)$ and $\epsilon \leq \frac{1}{64n^2p^2(n)}$ this probability of error reduces to $2/n^2$. □

Lemma 113. *For X satisfying $P(X)$ and sufficiently small $\epsilon = 1/\mathsf{poly}(n)$ and sufficiently large n and $m = \mathsf{poly}(n)$, we have*

$$\mathbb{P}^{\epsilon}_{\vec{Y}\in\{0,1\}^{n\ell}}\left[C'(F(X),R_1,\dots,R_m) \neq X\right] < 1/2$$

Proof. This lemma follows from Lemma 112 exactly as Lemma 110 follows from Lemma 109. □

From here the rest of the proof of the Goldreich-Levin Theorem is similar to that of the special case.

6.4 Construction of pseudorandom generators from the Goldreich-Levin Theorem

One nice implication of the existence of hardcore predicates as shown in the Goldreich-Levin Theorem is the construction of pseudorandom generators from one-way permutations. In our theory, we can formalize the definition of a pseudorandom generator as follows.

Definition 114. A function $G : \{0,1\}^m \to \{0,1\}^n$ is a (s,ϵ)-pseudorandom generator if all circuits D of size at most s, there exits γ such that for all $0 < \delta < \gamma$ and $\delta = 1/\mathsf{poly}(n)$,

$$\left|\mathbb{P}^{\delta}_{X\in\{0,1\}^n}\left[C(X)\right] - \mathbb{P}^{\delta}_{R\in\{0,1\}^m}\left[C(G(R))\right]\right| < \epsilon.$$

Then we would like to formalize the following simple textbook theorem that gives us a pseudorandom generator that maps n random bits to $n+1$ pseudorandom bits from a one-way permutation and its hardcore predicate.

Theorem 115. *Let $F : \{0,1\}^n \to \{0,1\}^n$ be a one-way permutation and let $P(X)$ be an (s,ϵ)-hardcore predicate of F, then*

$$G(X) = F(X), P(X)$$

is a $(s-d, \epsilon+\gamma)$-pseudorandom generator for some constant d and any $\gamma = 1/\mathsf{poly}(n)$.

To prove this theorem, we need to show that the outputs of the one-way permutation F are random and uniformly distributed. Due to errors from approximate counting, we would need to show that

$$\mathbb{P}^{\delta}_{X\in\{0,1\}^n, r\in\{0,1\}}\left[D(X,r)\right] - \mathbb{P}^{\delta}_{X\in\{0,1\}^n, r\in\{0,1\}}\left[D(F(X),r)\right] = \pm c\delta,$$

for some constant $c > 0$. For this, we need to show that

$$\{(X,r) \in \{0,1\}^n \times \{0,1\} \mid D(X,r)\} \approx_{a\delta} \{(X,r) \in \{0,1\}^n \times \{0,1\} \mid D(F(X),r)\},$$

for some constant $a > 0$.

Due to the fact that F is a permutation, we can show that

$$F : \{(X,r) \in \{0,1\}^n \times \{0,1\} \mid D(F(X),r)\} \twoheadrightarrow \{(X,r) \in \{0,1\}^n \times \{0,1\} \mid D(X,r)\}.$$

However since F is one-way, it is not clear how to construct a surjection to show that

$$\{(X,r) \in \{0,1\}^n \times \{0,1\} \mid D(X,r)\} \precsim_{a\delta} \{(X,r) \in \{0,1\}^n \times \{0,1\} \mid D(F(X),r)\}.$$

Thus we leave open the question of whether $HARD^A$ proves Theorem 115.

Chapter 7

Conclusion and future work

Our correctness proof of the reduction from Sᴍ to Cᴄᴠ in Section 3.7.2 is a nice example showing the utility of three-valued logic for reasoning about uncertainty. Since an instance of Sᴍ might not have a unique solution, the fact that the fixed point $I_{c(n)} = \mathcal{M}^{c(n)}(I_0)$ is three-valued indicates that the construction cannot fully determine how all the men and women can be matched. Thus, different Boolean fixed-point extensions of $I_{c(n)}$ give us different stable marriages.

It is worth noting that Subramanian's method is not the "textbook" method for solving Sᴍ. The most well-known is the Gale-Shapley algorithm [20]. In fact, our original motivation was to formalize the correctness of the Gale-Shapley algorithm, but we do not know how to talk about the computation of the Gale-Shapley algorithm in *VCC* due to the fan-out restriction in comparator circuits. Thus, we leave open the question whether *VCC* proves the correctness of the Gale-Shapley algorithm.

Despite our recent effort, the following main complexity-theoretic question remains unanswered about CC. Is Cᴄᴠ complete for P? And if not, is CC comparable to NC? We conjecture that CC is incomparable with the parallel class NC, and therefore CC is not P-complete. In the joint work with Cook and Filmus [16], we provide evidence for our conjecture by giving oracle settings in which relativized CC and relativized NC are incomparable, which implies that relativized CC is strictly contained in relativized P.

The results in Chapter 4 only consider randomized matching algorithms for *bipartite* graphs. For general undirected graphs, we need Tutte's matrix (cf. [47]), a generalization of Edmonds' matrix. Since every Tutte matrix is a skew symmetric matrix where each variable appears exactly twice, we cannot directly apply our technique for Edmonds' matrices, where each variable appears at most once. However, by using the recursive definition of the pfaffian instead of the cofactor expansion, we believe that it is also possible to generalize our results to general undirected graphs. We also note that the Hungarian algorithm only works for weighted bipartite graphs. To find a maximum-weight matching of a weighted undirected graph, we need to formalize Edmonds' *blossom algorithm* (cf. [32]). Once we have the correctness of the blossom algorithm, the proof of the Isolating Lemma for undirected graph perfect matchings will be the same as that of Theorem 69. We leave the detailed proofs for the general undirected graph case for future work.

It is worth noticing that symbolic determinants of Edmonds' matrices result in very special polynomials, whose structures can be used to define the *VPV* surjections witnessing the probability bound in the Schwartz-Zippel Lemma as demonstrated in this paper. It remains an open problem whether we can prove the full version of the Schwartz-Zippel Lemma using Jeřábek's method within the theory VPV or $VPV + \mathsf{sWPHP}(\mathsf{FP})$.

It was suggested to us by Russell Impagliazzo, Valentine Kabanets and Antonina Kolokolova (personal communication) that there might be some hope of proving the Schwartz-Zippel Lemma for multilinear arithmetic circuits and skew arithmetic circuits in VPV. Recall that skew circuits are defined by restricting every multiplication gate to have at least one of its inputs equal to a variable or a field constant. Since this class of arithmetic circuits has special structures, we hope to manipulate these circuits more easily. Unfortunately the problem remains elusive even for this class of circuits.

We have shown that the correctness proofs for several randomized algorithms can be formalized in the theory VPV for polynomial time reasoning. But some of the algorithms involved are in the subclass DET of polynomial time, where DET is the closure of $\#L$ (and also of the integer determinant function Det) under AC^0 reductions. As mentioned in Section 2.1.3 the subtheory $\overline{V\#L}$ of VPV can define all functions in DET, but it is open whether $\overline{V\#L}$ can prove properties of Det such as the expansion by minors. However the theory $\overline{V\#L} + CH$ can prove such properties [58], where CH is an axiom stating the Cayley-Hamilton Theorem. Thus in the statements of Lemma 59 and Theorem 62 we could have replaced $(VPV \vdash)$ by $(V\#L + CH \vdash)$. We could have done the same for Theorem 63 if we changed the argument $\vec{W}$ of the function H to M, where M is a permutation matrix encoding a perfect matching for the underlying bipartite graph. This modified statement of the theorem still proves the interesting direction of the correctness of the bipartite perfect matching algorithm in Section 4.2.3, since the function H is used only to bound the error assuming that G does have a perfect matching.

We leave open the question of whether any of the other correctness proofs can be formalized in $\overline{V\#L} + CH$.

In Chapter 5 we used Moser's technique in [44] to show that if every clause of a k-CNF formula F shares a variable with at most $2^k/8$ other clauses, then $VPV + \mathsf{sWPHP}(\mathcal{L}_{\mathsf{FP}})$ proves the existence of a satisfying assignment for F. Messner and Thierauf [41] generalized Moser's elegant proof to instances of k-SAT with neighbourhood size up to $2^k/e$. Thus it will be an interesting future work to generalize the result in Chapter 5 to k-SAT instances with neighbourhood size up to $2^k/e$. However, it is not clear at all how to formalize the full proof in [45], which uses properties of *infinite branching processes*. Thus we leave open the question of whether the constructive proof of the LLL by Moser-Tardos [45] formalizable in $VPV + \mathsf{sWPHP}(\mathsf{FP})$.

In Chapter 6 we extended Jeřábek's framework to formalize the Goldreich-Levin Theorem in the conservative extension $HARD^A$ of $VPV + \mathsf{sWPHP}(\mathsf{FP})$. However we leave open of how to formalize the construction of pseudorandom generators from one-way permutations.

I believe that Jeřábek's framework deserves to be studied in greater depth since it helps us to understand better the connection between probabilistic reasoning and weak systems of bounded arithmetic. Even with all the work mentioned in this thesis, the most important goal of bounded reverse mathematics in this context is to classify theorems that use probabilistic reasoning in their proofs according to the theories needed to prove them. Abundant resources of theorems with probabilistic flavor, which can easily be found in textbooks on the probabilistic method [3], randomized algorithms [46, 42] and cryptography theory [22, 23], are still waiting for us to be explored.

Appendix A

The correctness proof of the Hungarian algorithm

Before proceeding with the proof of Theorem 65 which implies the correctness of the Hungarian algorithm, we need to formalize the two most fundamental theorems for bipartite matching: Berge's Theorem and Hall's Theorem.

A.1 Formalizing Berge's Theorem and the augmenting-path algorithm

Let $G = (X \uplus Y, E)$ be a bipartite graph, where $X = \{x_i \mid 1 \leq i \leq n\}$ and $Y = \{y_i \mid 1 \leq i \leq n\}$. Formally to make sure that X and Y are disjoint, we can let $x_i := i$ and $y_i := n + i$. We encode the edge relation E of G by a matrix $E_{n \times n}$, where $E(i,j) = 1$ iff x_i is adjacent to y_j. Note that we often abuse notation and write $\{u,v\} \in E$ to denote that u and v are adjacent in G, which formally means either

$$u \in X \ \wedge \ v \in Y \wedge E(u, v - n), \text{ or}$$
$$v \in X \ \wedge \ u \in Y \wedge E(v, u - n).$$

This complication is due to the usual convention of using an $n \times n$ matrix to encode the edge relation of a bipartite graph with $2n$ vertices.

An $n \times n$ matrix M encodes a matching of G iff M is a permutation matrix satisfying

$$\forall i, j \in [n], M(i,j) \to E(i,j).$$

We represent a path by a sequence of vertices $\langle v_1, \ldots, v_k \rangle$ with $\{v_i, v_{i+1}\} \in E$ for all $i \in [k]$.

Given a matching M, a vertex v is *M-saturated* if v is incident with an edge in M. We will say v is *M-unsaturated* if it is not M-saturated. A path $P = \langle v_1, \ldots, v_k \rangle$ is an *M-alternating path* if P alternates between edges in M and edges in $E \setminus M$. More formally, P is an M-alternating path if either of the following two conditions holds:

- For every $i \in \{1, \ldots, k-1\}$, $\{v_i, v_{i+1}\} \in E \setminus M$ if i is odd, and $\{v_i, v_{i+1}\} \in M$ if i is even.

- For every $i \in \{1, \dots, k-1\}$, $\{v_i, v_{i+1}\} \in E \setminus M$ if i is even, and $\{v_i, v_{i+1}\} \in M$ if i is odd.

An M-alternating path $\langle v_1, \dots, v_k \rangle$ is an *M-augmenting path* if the vertices v_1 and v_k are M-unsaturated.

Theorem 116 (Berge's Theorem). *($VPV \vdash$) Let $G = (X \uplus Y, E)$ be a bipartite graph. A matching M is maximum iff there is no M-augmenting path in G.*

Proof. ($\Rightarrow$): Assume that all matchings N of E satisfy $|N| \leq |M|$. Suppose for a contradiction that there is an M-augmenting path P. Let $M \oplus P$ denote the symmetric difference of two sets of edges M and P. Then $M' = M \oplus P$ is a matching greater than M, a contradiction.

($\Leftarrow$): We will prove the contrapositive. Assume there is another matching M' satisfying $|M'| > |M|$. We want to construct an M-augmenting path in G.

Consider $Q = M' \oplus M$. Since $|M'| > |M|$, it follows that $|M' \setminus M| > |M \setminus M'|$, and thus

$$|Q \cap M'| > |Q \cap M| \tag{A.1}$$

Note that we can compute cardinalities of the sets directly here since all the sets we are considering here are small. Now let H be the graph whose edge relation is Q and whose vertices are simply the vertices of G. We then observe the following properties of H:

- Since Q is constructed from two matchings M and M', every vertex of H can only be incident with at most two edges: one from M and another from M'. So every vertex of H has degree at most 2.

- Any path of H must alternate between the edges of M and M'.

We will provide a polytime algorithm to extract from the graph H an augmenting path with respect to M, which gives us the contradiction.

```
1: Initialize $K = H$ and $i = 1$
2: **while** $K \neq \varnothing$ **do**
3:    Pick the least vertex $v \in K$
4:    Compute the connected component $C_i$ containing $v$
5:    **if** $C_i$ is an $M$-augmenting path **then**
6:       return $C_i$ and halt.
7:    **end if**
8:    Update $K = K \setminus C_i$ and $i = i + 1$.
9: **end while**
```

Note that since H has n vertices, the while loop can only iterate at most n times. It only remains to show the following.

Claim: The algorithm returns an M-augmenting path assuming $|M'| > |M|$.

Suppose for a contradiction that the algorithm would never produce any M-augmenting path. Since H has degree at most two, in every iteration of the while loop, we know that the connected component C_i is

- either a cycle, which means $|C_i \cap M| = |C_i \cap M'|$, or

- a path but not an M-augmenting path, which implies that $|C_i \cap M| \geq |C_i \cap M'|$.

Since $Q = M' \oplus M = \bigcup_i C_i$ and all C_i are disjoint, we have

$$|Q \cap M| = |\bigcup_i (C_i \cap M)| \geq |\bigcup_i (C_i \cap M')| = |Q \cap M'|.$$

But this contradicts (A.1). □

Algorithm 117 (The augmenting-path algorithm)**.** As a corollary of Berge's Theorem, we have the following simple algorithm for finding a maximum matching of a bipartite graph G. We start from any matching M of G, say empty matching. Repeatedly locate an M-augmenting path P and augment M along P and replace M by the resulting matching. Stop when there is no M-augmenting path. Then we know that M is maximum. Thus, it remains to show how to search for an M-augmenting path given a matching M of G.

Algorithm 118 (The augmenting-path search algorithm)**.** First, from G and M we construct a directed graph H, where the vertices V_H of H are exactly the vertices $X \uplus Y$ of G, and the edge relation E_H of H is a $2n \times 2n$ matrix defined as follows:

$$E_H := \{(x,y) \in X \times Y \mid \{x,y\} \in E \setminus M\} \cup \{(y,x) \in Y \times X \mid \{y,x\} \in M\}.$$

The key observation is that t is reachable from s by an M-alternating path in the bipartite graph G iff t is reachable from s in the directed graph H.

After constructing the graph H, we can search for an M-augmenting path using the *breadth first search* algorithm as follows. Let s be an M-unsaturated vertex in X. We construct two $2n \times 2n$ matrices S and T as follows.

1: The row $\mathsf{Row}(1,S)$ of S encodes the set $\{s\}$, the starting point of our search.

2: From $\mathsf{Row}(i,S)$, the set $\mathsf{Row}(i+1,S)$ is defined as follows: $j \in \mathsf{Row}(i+1,S) \leftrightarrow$ there exists some $k \in [n]$,

$$\mathsf{Row}(i,S)(k) \ \wedge \ E_H(k,j) \ \wedge \forall \ell \in [i], \neg \mathsf{Row}(\ell,S)(k).$$

After finish constructing $\mathsf{Row}(i+1,S)$, we can update T by setting $T(j,k)$ to 1 for every $k \in \mathsf{Row}(i,S)$ and $j \in \mathsf{Row}(i+1,S)$ satisfying $E_H(k,j)$.

Intuitively, $\mathsf{Row}(i,S)$ encodes the set of vertices that are of distance $i-1$ from s, and T is the auxiliary matrix that can be used to recover a path from s to any vertex $j \in \mathsf{Row}(i,S)$ for all $i \in [2n]$.

Remark 119. Let $R = \bigcup_{i=2}^{n} \mathsf{Row}(i,S)$, then R the set of vertices reachable from s by an M-alternating path. This follows from the fact that our construction mimics the construction of the formula δ_{CONN}, which was used to define the theory *VNL* for NL in [14, Chapter 9]. The matrix T is constructed to help us trace a path for every vertex $v \in R$ to s.

By induction on i, we can prove that $\mathsf{Row}(i,S) \subseteq X$ for every odd $i \in [2n]$, and $\mathsf{Row}(i,S) \subseteq Y$ for every even $i \in [2n]$. Thus, after having S and T, we choose the largest even i^* such that $\mathsf{Row}(i^*,S) \neq \varnothing$, and then search the set $\mathsf{Row}(i^*,S) \cap Y$ for an M-unsaturated vertex t. If such vertex t exists, then we use T to trace back a path to s. This path will be our M-augmenting path. If no such vertex t exists, we report that there is no M-augmenting path.

A.2 Formalizing Hall's Theorem

Theorem 120 (Hall's Theorem). *(VPV $\vdash$) Let $G = (X \uplus Y, E)$ be a bipartite graph. Then G has a perfect matching if and only if for every subset $S \subseteq X$,*

$$|S| \leq |N(S)|,$$

where $N(S)$ denotes the neighborhood of S in G.

The condition $\forall S \subseteq X, |S| \leq |N(S)|$, which is necessary and sufficient for a bipartite graph to have a perfect matching, is called Hall's condition. We encode a set $S \subseteq X$ in the theorem as a binary string of length n, where $S(i) = 1$ iff $x_i \in S$. Similarly, we encode the neighborhood $N(S)$ as a binary string of length n, and we define

$$N(S) := \bigcup \{\mathsf{Row}(i, E) \mid x_i \in S\},$$

where the union can be computed by taking the disjunction of all binary vectors in the set

$$\{\mathsf{Row}(i, E) \mid x_i \in S\}$$

componentwise. Note that we can compute the cardinalities of S and $N(S)$ directly since both of these sets are subsets of small sets X and Y.

Proof. ($\Rightarrow$): Assume M is a perfect matching of G. Given a subset $S \subseteq X$, the vertices in S are matched to the vertices in some subset $T \subseteq Y$ by the perfect matching M, where $|S| = |T|$. Since $T \subseteq N(S)$, we have $|N(S)| \geq |T| = |S|$.

($\Leftarrow$): We will prove the contrapositive. Assume G does not have a perfect matching, we want to construct a subset $S \subseteq X$ such that $|N(S)| < |S|$.

Let M be a maximum but not perfect matching constructed by the augmenting-path algorithm. Since M is not a perfect matching, there is some M-unsaturated vertex $s \in X$. Let S and T be the result of running the "augmenting-path search" algorithm from s, then $R := \bigcup_{i=2}^{n} \mathsf{Row}(i, S)$ is the set of all vertices reachable from s by an M-alternating path. Since there is no M-augmenting path, all the vertices in R are M-saturated. We want to show the following two claims.

Claim 1: The vertices in $R \cap X$ are all matched to the vertices in $R \cap Y$ by M, and

$$|R \cap X| = |R \cap Y|.$$

Suppose for a contradiction that some vertex $v \in R$ is not matched to any vertex $u \in R$ by M. Since we already know that all vertices in R are M-saturated, v is matched by some vertex $w \notin R$ by M. But this is a contradiction since w must be reachable from s by an alternating path, and so the augmenting-path search algorithm must already have added w to R. Thus, the vertices in $R \cap X$ are all matched to the vertices in $R \cap Y$ by M, which implies that $|R \cap X| = |R \cap Y|$.

Claim 2: $N(R \cap X) = R \cap Y$.

Since $R \cap X$ are matched to $R \cap Y$, we know $N(R \cap X) \supseteq R \cap Y$. Suppose for a contradiction that $N(R \cap X) \supset R \cap Y$. Let $v \in N(R \cap X) \setminus R \cap Y$, and $u \in R \cap X$ be the vertex adjacent to v. Since u is reachable from s by an M-alternating path P, we can extend P to get an M-alternating path from s to v, which contradicts that v is not added to R.

We note that $N(\{s\} \cup (R \cap X)) = R \cap Y$. Then $S = \{s\} \cup (R \cap X)$ is the desired set since

$$|N(S)| = |R \cap Y| = |R \cap X| < |S|.$$

□

From the proof of Hall's Theorem, we have the following corollary saying that if a bipartite graph does not have a perfect matching, then we can find in polytime a subset of vertices violating Hall's condition.

Corollary 121. *($VPV \vdash$) There is a VPV function that, on input a bipartite graph G that does not have a perfect matching, outputs a subset $S \subseteq X$ such that $|S| > |N(S)|$.*

A.3 Proof of Theorem 65

Let $H = H_{\vec{u},\vec{v}}$ be the equality subgraph for the weight cover $(\vec{u}, \vec{v})$, and let M be a maximum cardinality matching of H. Recall Theorem 65 wants us to show that VPV proves equivalence of the following three statements:

1. $w(M) = \mathsf{cost}(\vec{u}, \vec{v})$

2. M is a maximum-weight matching and the cover $(\vec{u}, \vec{v})$ is a minimum-weight cover of G

3. M is a perfect matching of H

Proof of Theorem 65. (1)⇒(2): Assume that $\mathsf{cost}(\vec{u}, \vec{v}) = w(M)$. By Lemma 64, no matching has weight greater than $\mathsf{cost}(\vec{u}, \vec{v})$, and no cover with weight less than $w(M)$.

(2)⇒(3): Assume M is a maximum-weight matching and $(\vec{u}, \vec{v})$ is a minimum-weight cover of G. Suppose for a contradiction that the maximum matching M is not a perfect matching of H. We will construct a weight cover whose cost is strictly less than $\mathsf{cost}(\vec{u}, \vec{v})$, which contradicts that $(\vec{u}, \vec{v})$ is a minimum-weight cover.

Since the maximum matching M is not a perfect matching of H, by Corollary 121, we can construct in polytime a subset $S \subseteq X$ satisfying

$$|N(S)| < |S|.$$

Then we calculate the quantity

$$\delta = \min\{u_i + v_j - w_{i,j} \mid x_i \in S \ \wedge \ y_j \notin N(S)\}.$$

Note that $\delta > 0$ since H is the equality subgraph. Next we construct a pair of sequences $\vec{u}' = \langle u_i' \rangle_{i=1}^{n}$ and $\vec{v}' = \langle v_i' \rangle_{i=1}^{n}$, as follows:

$$u_i' = \begin{cases} u_i - \delta & \text{if } x_i \in S \\ u_i & \text{if } x_i \notin S \end{cases} \qquad v_i' = \begin{cases} v_j + \delta & \text{if } y_j \in N(S) \\ v_j & \text{if } y_j \notin N(S) \end{cases}$$

We claim that $(\vec{u}', \vec{v}')$ is again a weight cover. The condition $w_{i,j} \leq u_i' + v_j'$ might only be violated for $x_i \in S$ and $y_i \notin N(S)$. But since we chose $\delta \leq u_i + v_j - w_{i,j}$, it follows that

$$w_{i,j} \leq (u_i - \delta) + v_j = u_i' + v_j'.$$

Since

$$\begin{aligned} \mathsf{cost}(\vec{u}', \vec{v}') &= \textstyle\sum_{i=1}^{n} (u_i' + v_i') \\ &= \underbrace{\textstyle\sum_{i=1}^{n} (u_i + v_i)}_{=\mathsf{cost}(\vec{u}, \vec{v})} + \underbrace{\delta |N(S)| - \delta |S|}_{<0}, \end{aligned}$$

it follows that $\mathsf{cost}(\vec{u}', \vec{v}') < \mathsf{cost}(\vec{u}, \vec{v})$.

(3)⇒(1): Suppose M is a perfect matching of H. Then $w_{i,j} = u_i + v_j$ holds for all edges in M. Summing equalities $w_{i,j} = u_i + v_j$ over all edges of M yields the equality $\mathsf{cost}(\vec{u}, \vec{v}) = w(M)$. □

Bibliography

[1] M. Agrawal and S. Biswas. Primality and identity testing via chinese remaindering. *Journal of the ACM*, 50(4):429–443, 2003.

[2] N. Alon. A parallel algorithmic version of the local lemma. *Random Structures and Algorithms*, 2(4):367–378, 1991.

[3] N. Alon and J. H. Spencer. *The Probabilistic Method*. Wiley Series in Discrete Mathematics and Optimization. Wiley-Interscience, 2008.

[4] S. Arora, C. Lund, R. Motwani, M. Sudan, and M. Szegedy. Proof verification and the hardness of approximation problems. *Journal of the ACM*, 45(3):501–555, 1998.

[5] S. Arora and S. Safra. Probabilistic checking of proofs: A new characterization of NP. *Journal of the ACM*, 45(1):122, 1998.

[6] David A. Mix Barrington, Neil Immerman, and Howard Straubing. On uniformity within NC^1. *Journal of Computer and System Sciences*, 41(3):274 – 306, 1990.

[7] K. E. Batcher. Sorting networks and their applications. In *Proceedings of the AFIPS Spring Joint Computer Conference 32*, pages 307–314. ACM, 1968.

[8] J. Beck. An algorithmic approach to the Lovász Local Lemma. *Random Structures and Algorithms*, 2(4):343–365, 1991.

[9] S. J. Berkowitz. On computing the determinant in small parallel time using a small number of processors. *Information Processing Letters*, 18(3):147–150, 1984.

[10] S. R. Buss. *Bounded Arithmetic*. Bibliopolis, Naples, 1986.

[11] S. R. Buss. The Boolean Formula Value Problem is in ALOGTIME. In *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing*, STOC '87, pages 123–131, New York, NY, USA, 1987. ACM.

[12] A. Cobham. The Intrinsic Computational Difficulty of Functions. In *Proceedings of the 1965 International Congress for Logic, Methodology and Philosophy of Science*, pages 24–30. North-Holland, 1965.

[13] S. Cook and L. Fontes. Formal Theories for Linear Algebra. In *Computer Science Logic*, pages 245–259. Springer, 2010.

[14] S. Cook and P. Nguyen. *Logical foundations of proof complexity*. Cambridge University Press, 2010.

[15] S. A. Cook. Feasibly constructive proofs and the propositional calculus (preliminary version). In *Proceedings of the seventh annual ACM symposium on Theory of computing*, pages 83–97. ACM, 1975.

[16] S. A. Cook, Y. Filmus, and D. T. M. Lê. The complexity of the comparator circuit value problem. *CoRR*, abs/1208.2721, 2012.

[17] S. A. Cook and R. A. Reckhow. The Relative Efficiency of Propositional Proof Systems. *The Journal of Symbolic Logic*, 44(1):36–50, 1979.

[18] J. Edmonds. Systems of distinct representatives and linear algebra. *J. Res. Natl. Bureau of Standards*, 71A:241–245, 1967.

[19] P. Erdős and L. Lovász. Problems and results on 3-chromatic hypergraphs and some related questions. In A. Hajnal, R. Rado, and V. T. Sós, editors, *Infinite and Finite Sets (to Paul Erdős on his 60th birthday)*, volume II, pages 609–627. North-Holland, 1975.

[20] D. Gale and L. S. Shapley. College admissions and the stability of marriage. *The American Mathematical Monthly*, 69(1):9–15, 1962.

[21] W. I. Gasarch and B. Haeupler. Lower Bounds on van der Waerden Numbers: Randomized- and Deterministic-Constructive. *Electr. J. Comb.*, 18(1), 2011.

[22] O. Goldreich. *Foundations of cryptography - Volume I (Basic tools)*. Cambridge University Press, 2001.

[23] O. Goldreich. *Foundations of cryptography - Volume II (Basic applications)*. Cambridge University Press, 2004.

[24] O. Goldreich and L.A. Levin. A hard-core predicate for all one-way functions. In *Proceedings of the twenty-first annual ACM symposium on Theory of computing*, pages 25–32. ACM, 1989.

[25] R. Greenlaw and S. Kantabutra. On the parallel complexity of hierarchical clustering and CC-complete problems. *Complexity*, 14(2):18–28, 2008.

[26] O. H. Ibarra and S. Moran. Probabilistic Algorithms for Deciding Equivalence of Straight-Line Programs. *Journal of the ACM (JACM)*, 30(1):217–228, 1983.

[27] R. Impagliazzo and A. Wigderson. P=BPP if E requires exponential circuits: Derandomizing the XOR lemma. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 220–229. ACM, 1997.

[28] E. Jeřábek. Dual weak pigeonhole principle, Boolean complexity, and derandomization. *Annals of Pure and Applied Logic*, 129(1–3):1–37, 2004.

[29] E. Jeřábek. *Weak pigeonhole principle, and randomized computation*. PhD thesis, Faculty of Mathematics and Physics, Charles University, Prague, 2005.

[30] E. Jeřábek. Approximate counting in bounded arithmetic. *Journal of Symbolic Logic*, 72(3):959–993, 2007.

[31] J. Katz and Y. Lindell. *Introduction to modern cryptography*. CRC Press, 2008.

[32] B.H. Korte and J. Vygen. *Combinatorial optimization: theory and algorithms*. Springer Verlag, 2008.

[33] J. Krajíček. *Bounded arithmetic, propositional logic, and complexity theory*. Cambridge University Press, 1995.

[34] J. Krajíček, P. Pudlák, and G. Takeuti. Bounded Arithmetic and the Polynomial Hierarchy. *Annals of Pure and Applied Logic*, 52:143–153, 1991.

[35] H. W. Kuhn. The Hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955.

[36] D. T. M. Lê and S. A. Cook. Formalizing randomized matching algorithms. In *Symposium on Logic in Computer Science*, pages 185–194. IEEE Computer Society, 2011.

[37] D. T. M. Lê and S. A. Cook. Formalizing randomized matching algorithms. *Logical Methods in Computer Science*, 8(3), 2012.

[38] D. T. M. Lê, S. A. Cook, and Y. Ye. A Formal Theory for the Complexity Class Associated with the Stable Marriage Problem. In *Computer Science Logic, 20th Annual Conference of the EACSL, CSL 2011, September 12-15, 2011, Bergen, Norway, Proceedings*, pages 381–395. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2011.

[39] L. Lovász. On determinants, matchings and random algorithms. In L. Budach, editor, *Fundamentals of Computation Theory*, pages 565–574. Akademia-Verlag, 1979.

[40] E. W. Mayr and A. Subramanian. The complexity of circuit value and network stability. *Journal of Computer and System Sciences*, 44(2):302–323, 1992.

[41] J. Messner and T. Thierauf. A Kolmogorov Complexity Proof of the Lovász Local Lemma for Satisfiability. In Bin Fu and Ding-Zhu Du, editors, *Computing and Combinatorics*, volume 6842 of *Lecture Notes in Computer Science*, pages 168–179. Springer Berlin / Heidelberg, 2011.

[42] M. Mitzenmacher and E. Upfal. *Probability and computing: Randomized algorithms and probabilistic analysis*. Cambridge University Press, 2005.

[43] C. Moore and J. Machta. Internal Diffusion-Limited Aggregation: Parallel Algorithms and Complexity. *Journal of Statistical Physics*, 99:661–690, 2000.

[44] R. A. Moser. A constructive proof of the Lovász Local Lemma. In *Proceedings of the 41st annual ACM symposium on Theory of computing*, pages 343–350. ACM, 2009.

[45] R. A. Moser and G. Tardos. A constructive proof of the general Lovász Local Lemma. *Journal of the ACM (JACM)*, 57(2):1–15, 2010.

[46] R. Motwani and P. Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

[47] K. Mulmuley, U. V. Vazirani, and V. V. Vazirani. Matching is as easy as matrix inversion. *Combinatorica*, 7(1):105–113, 1987.

[48] J. Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial and Applied Mathematics*, pages 32–38, 1957.

[49] P. Nguyen. *Bounded Reverse Mathematics*. PhD thesis, University of Toronto, 2008.

[50] N. Nisan and A. Wigderson. Hardness vs randomness. *Journal of Computer and System Sciences*, 49(2):149–167, 1994.

[51] R. Parikh. Existence and feasibility in arithmetic. *Journal of Symbolic Logic*, 36(3):494–508, 1971.

[52] J. B. Paris, A. J. Wilkie, and A. R. Woods. Provability of the pigeonhole principle and the existence of infinitely many primes. *Journal of Symbolic Logic*, 53(4):1235–1244, 1988.

[53] P. Pudlák. Ramsey's theorem in bounded arithmetic. In *Computer Science Logic*, pages 308–317. Springer, 1991.

[54] V. Ramachandran and L.-C. Wang. Parallel algorithm and complexity results for telephone link simulation. In *Proceedings of the Third IEEE Symposium on Parallel and Distributed Processing*, pages 378 –385, 1991.

[55] J. T. Schwartz. Fast probabilistic algorithms for verification of polynomial identities. *Journal of the ACM (JACM)*, 27(4):717, 1980.

[56] A. Shamir. IP = PSPACE. *Journal of the ACM*, 39(4):869–877, 1992.

[57] S. Simpson. *Subsystems of second order arithmetic*. Cambridge University Press; 2 edition, 2009.

[58] M. Soltys and S. Cook. The proof complexity of linear algebra. *Annals of Pure and Applied Logic*, 130:277–323, 2004.

[59] Aravind Srinivasan. Improved algorithmic versions of the Lovász Local Lemma. In *Proceedings of the nineteenth annual ACM-SIAM symposium on Discrete algorithms*, SODA '08, pages 611–620, Philadelphia, PA, USA, 2008. Society for Industrial and Applied Mathematics.

[60] A. Subramanian. *The computational complexity of the circuit value and network stability problems*. PhD thesis, Dept. of Computer Science, Stanford University, 1990.

[61] A. Subramanian. A new approach to stable matching problems. *SIAM Journal on Computing*, 23(4):671–700, 1994.

[62] N. Thapen. *The weak pigeonhole principle in models of bounded arithmetic*. PhD thesis, University of Oxford, 2002.

[63] S. Toda. PP is as Hard as the Polynomial-Time Hierarchy. *SIAM Journal on Computing*, 20(5):865–877, 1991.

[64] H. Wasserman and M. Blum. Software reliability via run-time result-checking. *Journal of the ACM*, 44(6):826–849, 1997.

[65] R. Zippel. An explicit separation of relativised random polynomial time and relativised deterministic polynomial time. *Information Processing Letters*, 33(4):207–212, 1989.