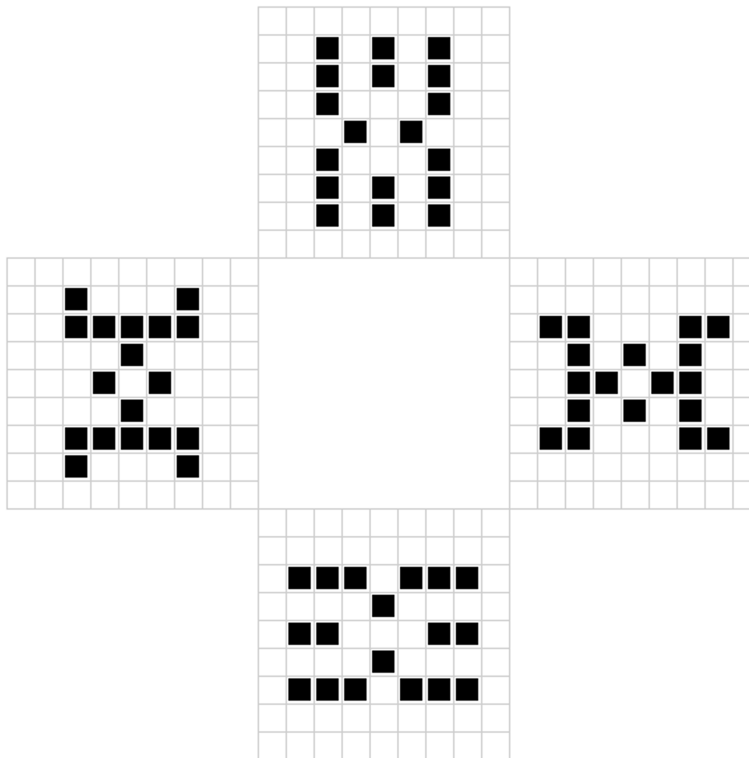
UNIVERSITEIT VAN AMSTERDAM

A Medley for Computational Complexity. With Applications of Information Theory, Learning Theory, and Ketan Mulmuley's Parametric Complexity Technique

B. Serra Loff Barreto

A Medley for Computational Complexity

With applications of Information Theory, Learning Theory,
and Ketan Mulmuley's Parametric Complexity Technique



Bruno Loff

Abstract

This thesis contains four parts. Each part studies a topic within computational complexity by applying techniques from other fields in theoretical computer science.

In **Chapter 1** we will use Kolmogorov complexity to study probabilistic polynomial-time algorithms. Let R denote the set of Kolmogorov-random strings, which are those strings x whose Kolmogorov complexity $K(x)$ is at least as large as their length $|x|$. There are two main results. First, we show that any bounded-error probabilistic polynomial-time algorithm can be simulated by a deterministic polynomial-time algorithm that is allowed to make non-adaptive queries to R. Second, we show that for a time-bounded analogue of R (defined using time-bounded Kolmogorov complexity), it holds that any polynomial-time algorithm that makes non-adaptive queries to R can be simulated both in polynomial space and by circuits of polynomial size.

This indicates that we are near to an alternative characterization of probabilistic polynomial-time as being exactly deterministic polynomial-time with non-adaptive queries to R. Such characterizations ultimately aim at using techniques from Kolmogorov complexity and computability to study the relationship between different complexity classes. As can be expected, the proofs in this chapter make essential use of such techniques.

In **Chapter 2** we make an effort at extending Mahaney's theorem [7] to more general reductions, or — seen another way — strengthening the Karp-Lipton theorem [5] to a stronger collapse of the polynomial-time hierarchy. Mahaney's theorem states that if SAT is m-reducible to a sparse set, then $\mathrm{P} = \mathrm{NP}$, and the Karp-Lipton theorem (more precisely, the strengthened version of Cai [2]) says that if SAT is Turing-reducible to a sparse set, then $\mathrm{PH} \subseteq \mathrm{ZPP}^{\mathrm{NP}}$.

We prove that if a class of functions $\mathcal{C}$ has a polynomial-time learning algorithm in Angluin's bounded error learning model, then if SAT is m-reducible to $\mathcal{C}$, it follows that $\mathrm{PH} \subseteq \mathrm{P}^{\mathrm{NP}}$.

Then from the existence of such an algorithm for linear-threshold functions, we conclude that if SAT is m-reducible to a linear-threshold function, then $\mathrm{PH} \subseteq \mathrm{P}^{\mathrm{NP}}$. It will be seen that both disjunctive and majority truth-table (non-adaptive) reductions to sparse sets are a special case of m-reductions to linear-threshold functions, and hence our results hold also for these kinds of reductions.

We also prove a somewhat stronger result of independent interest. For such a class of functions $\mathcal{C}$, it holds that if SAT m-reduces to $\mathcal{C}$, then we can answer any number of SAT-queries of length n by asking only n (larger) queries to SAT.

There are two main results in **Chapter 3**. First, we prove a more refined NP-hardness result for knapsack and related problems. We will construct a reduction from the satisfiability of fan-in-2 circuits of size S with k input bits to an instance of the subset-sum problem having bit-length $O(S+k)$. A corollary of this is a simple proof that there is no approximation algorithm for the knapsack problem which gives a better-than-inverse-polynomial approximation ratio, unless the exponential-time hypothesis of Impagliazzo and Paturi [3] fails to hold.

Secondly, we will use the technique we just developed, together with Ketan Mulmuley's parametric complexity technique, in order to prove an unconditional lower bound in Mulmuley's parallel semi-algebraic PRAM model [8]. We will show that, in that model, there is no algorithm for solving the knapsack problem in time $o(x^{1/4})$ using $2^{o(n^{1/4})}$ processors, even when the bit-length of the weights is restricted to n. The significance of this result follows from the fact that pretty much every known parallel algorithm can be implemented in this model.

In **Chapter 4**, we turn to communication complexity and information complexity [1]. We prove several theorems: (1) we show a "Reverse Newman's Theorem", stating that a private-coin q-round protocol that reveals I bits of information can be simulated by a public-coin q-round protocol that reveals $I+\tilde{O}(q)$ bits of information; (2) we show that public-coin protocols that reveal I bits of information can be simulated by protocols that communicate only $\tilde{O}(I)$ bits (but possibly use many more rounds); (3) we prove a constant-round two-way variant of the Slepian–Wolf theorem, and use it to show that q-round public-coin protocols that reveal I bits of information can be simulated by protocols that communicate only $O(I)+\tilde{O}(q)$ bits on average, and make use of $O(q)$ rounds, also on average; and (4) as a consequence of (1) and (3) we show a direct-sum theorem for bounded-round protocols, which states that, for any function f which needs C bits of (average) communication to be computed by randomized protocols in $O(q)$-average-rounds, computing k copies of the function using a q-round protocol requires $\Omega(kC)-\tilde{O}(q)$ bits of communication.

Samenvatting

Dit proefschrift is opgebouwd uit vier delen. In elk deel wordt een onderwerp binnen de computationele complexiteitstheorie bestudeerd door het toepassen van technieken uit andere gebieden van de theoretische informatica.

In **Hoofdstuk 1** zullen wij Kolmogorov-complexiteit gebruiken om probabilistische polynomiale-tijd algoritmen te bestuderen. Neem R als de verzameling van Kolmogorov-willekeurige binaire sequenties, dat zijn die sequenties x waarvan de Kolmogorov complexiteit $K(x)$ minstens zo groot is als hun lengte $|x|$. De twee belangrijkste resultaten in dit hoofdstuk zijn de volgende. In de eerste plaats laten we zien dat elk probabilistisch polynomiale-tijd algoritme met begrensde foutkans gesimuleerd kan worden door een deterministisch polynomiale-tijd algoritme dat niet-adaptieve vragen aan R kan stellen. Ten tweede laten we zien dat wanneer we een tijdsbegrensde variant van R gebruiken (gedefinieerd met behulp van tijdsbegrensde Kolmogorov-complexiteit), elk polynomiale-tijd algoritme dat niet-adaptieve vragen stelt aan R gesimuleerd kan worden door zowel Turing machines met polynomiale ruimte als door circuits van polynomiale grootte.

Deze resultaten geven aan dat we in de buurt komen van een alternatieve beschrijving van probabilistische polynomiale tijd, namelijk precies als deterministische polynomiale-tijd berekeningen met niet-adaptieve vragen naar R. Zulke beschrijvingen mikken uiteindelijk op het gebruik van technieken uit de Kolmogorov-complexiteit en berenbaarheidstheorie om de onderlinge verhoudingen van verschillende complexiteitsklassen te bestuderen. Zoals te verwachten maken de bewijzen in dit hoofdstuk sterk gebruik van zulke technieken.

In **Hoofdstuk 2** doen we een aanzet om Mahaney's stelling [7] uit te breiden naar algemenere reducties, of — anders bekeken — de ineenstorting van de polynomiale-tijd hierarchie in de Karp-Lipton stelling [5] te versterken. Mahaney's stelling zegt dat als SAT m-reduceerbaar is naar een ijle verzameling, dan volgt P = NP, en de Karp-Lipton stelling (of preciezer, de sterkere variant van Cai [2]) vertelt ons dat gegeven dat SAT Turing-reduceerbaar is naar een ijle verzameling, er dan geldt dat $\mathrm{PH} \subseteq \mathrm{ZPP}^{\mathrm{NP}}$.

We bewijzen dat wanneer een functie-klasse $\mathcal{C}$ een polynomiale-tijd leeralgoritme heeft in Angluin's *bounded error learning* model, dan krijgen we SAT is m-reduceerbaar naar $\mathcal{C}$ impliceert dat $\mathrm{PH} \subseteq \mathrm{P}^{\mathrm{NP}}$.

Vanuit het bestaan van zo een algoritme voor linear-threshold functies, con-

cluderen we dat als SAT m-reduceerbaar is naar een linear-threshold functie, dan volgt PH $\subseteq \mathrm{P}^{\mathrm{NP}}$. Het zal te zien dat dan zowel disjunctieve als majority truth-table (niet-adaptieve) reducties naar ijle verzamelingen een specifieke instantie van m-reducties naar linear-threshold functies zijn, en vandaar gelden onze resultaten ook voor deze typen reducties.

Ook bewijzen we het volgende sterkere resultaat, wat van zelfstandig belang is. Voor een klasse functies $\mathcal{C}$ geldt het dat wanneer SAT m-reduceerbaar is naar $\mathcal{C}$, dan kunnen we elk aantal SAT-vragen van lengte n beantwoorden door slechts n (grotere) vragen aan SAT te stellen.

De twee voornaamste resultaten te vinden in **Hoofdstuk 3** zijn de volgende. Ten eerste bewijzen we een verfijnder NP-moeilijkheidsresultaat voor het knapsakprobleem en gerelateerde problemen. We construeren een reductie vanaf vervulbaarheid van fan-in-2 circuits van grootte S met k invoer-bits naar een instantie van het subset-sum probleem met bit-lengte $O(S+k)$. Als corollarium hiervan krijgen we een simpel bewijs dat er geen benaderings-algoritme voor het knapzakprobleem is dat een beter-dan-reciproke-polynome benaderingsratio geeft, tenzij de exponentiële-tijd hypothese van Impagliazzo en Paturi [3] niet waar is.

Ten tweede zullen we de zojuist ontwikkelde methode gebruiken, samen met Ketan Mulmuley's parametrische complexiteitstechniek, om een onvoorwaardelijke ondergrens te bewijzen in Mulmuley's parallelle semi-algebraïsche PRAM model [8]. We zullen laten zien dat, in het betreffende model, er geen algoritme bestaat wat het knapzakprobleem oplost in tijd $o(x^{1/4})$, gebruik makend van $2^{o(n^{1/4})}$ processoren, zelfs wanneer de bit-lengte van de gewichten begrensd is tot n. De significantie van dit resultaat komt voort uit het feit dat vrijwel elk bekend parallel algoritme geïmplementeerd kan worden in dit model.

In **Hoofdstuk 4** richten we ons op communicatie-complexiteit en informatie-complexiteit [1]. We bewijzen verscheidene stellingen: (1) we geven een "Reverse Newman's Theorem" (omgekeerde Newman's stelling), deze stelt dat een private-coin q-ronde protocol welke I bits aan informatie openbaart, gesimuleerd kan worden door een public-coin q-ronde protocol welke $I + \tilde{O}(q)$ bits aan informatie openbaart; (2) we laten zien dat public-coin protocollen welke I bits aan informatie tonen gesimuleerd kunnen worden door procollen die slechts $\tilde{O}(I)$ bits communiceren (in mogelijk meer rondes); (3) we bewijzen een constante-ronde two-way variant van de Slepian–Wolf stelling, en gebruiken deze om te laten zien dat q-ronde public-coin protocollen die I bits informatie openbaren, gesimuleerd kunnen worden door protocollen die slechts gemiddeld $O(I) + \tilde{O}(q)$ bits communiceren, en gemiddeld $O(q)$ rondes gebruiken; en (4) gebruik makend van (1) en (3) laten we een direct-sum stelling zien voor protocollen met een begrensd aantal rondes, welke zegt dat, voor elke functie f die gemiddeld C bits communicatie nodig heeft om berekend te worden door gerandomiseerde protocollen in gemiddeld $O(q)$ rondes, er $\Omega(kC) - \tilde{O}(q)$ bits aan communicatie nodig zijn om k kopieën van deze functie te berekenen met een protocol wat q rondes gebruikt.

Introduction for a general audience

Computational complexity, also known as complexity theory, is the study of how much of a given resource is necessary to solve a given mathematical problem. An example that everyone is likely to know is the following: suppose we wish to multiply two integers x and y. We are given the digits of x and y in decimal notation — suppose both x and y have n digits — and we wish to obtain the digits of the number $z = x \times y$. The typical primary-school multiplication method works as follows: we place the number x on top of y, aligning the digits to the right, and then we proceed by multiplying the rightmost digit y_1 of y with x, and writing it down, then multiplying the second rightmost digit y_2 of y with x, and writing it below the previous number while shifting all its digits one position to the left, and so on, for all n digits of y; this gives us n numbers $z^{(1)}, \ldots, z^{(n)}$, where each $z^{(i)} = y_i \times x$ is the i-th digit of y multiplied by x, a number which could have up to $n+1$ digits. To end the multiplication, we sum all of the $z^{(i)}$'s, and this gives us the digits of $z = x \times y$. When we apply this method using paper divided into squares, like in a child's arithmetic book, we get something that looks like this:

					x_n	$\ldots$	x_1	
				$\times$	y_n	$\ldots$	y_1	
			$z^{(1)}_{n+1}$	$z^{(1)}_n$	$\ldots$	$z^{(1)}_1$		
		$z^{(2)}_{n+1}$	$z^{(2)}_n$	$\ldots$	$z^{(2)}_1$			
	$\iddots$	$\iddots$	$\iddots$	$\iddots$				
$+$	$z^{(n)}_{n+1}$	$z^{(n)}_n$	$\ldots$	$z^{(n)}_1$				
	z_{2n}	z_{2n-1}	$\ldots$	$\ldots$	$\ldots$	z_1		

Now let us look at the following resource: how many squares are we using in order to multiply the two numbers? In our tally, let us ignore the squares that we absolutely must use, namely the squares that hold the digits of x, the digits of y, and those for the result $x \times y$.[1] How many additional squares do we need to use?

[1] We do this because we are interested in comparing different methods for multiplication, with respect to the use of squares... but any method for multiplication is forced to use at least those squares required to write the input and the output. Hence counting these squares would be superfluous.

For instance, the primary-school method which we have just seen uses $n+1$ squares for each $z^{(i)}$, and there are n such $z^{(i)}$. So, in total, the number of squares used by this method is $n \times (n+1) = n^2 + n$.

For the problem of multiplying two numbers, and for this particular resource, a complexity theorist is primarily interested in answering the following question: What is the smallest number of squares that we need to use, in order multiply two n-digit numbers x and y? The answer to this question depends on n, and is given by a function $f(n)$, which a complexity theorist would call *the space-complexity of multiplication*. We have just shown that $f(n) \leq n^2 + n$. But this is not the minimum! If we use a slightly more clever method, we can show that $f(n) \leq 2n + 4$ (can the reader figure out how? the method requires the use of an erasor, in order to re-use the squares). The best method we know achieves $f(n) \leq c \log n$ for some fixed constant c that does not depend on n,[2] and it can be shown that this is the best possible[3].

Other problems can be studied. For instance, how much space do we need in order to find the shortest path between two cities on a map having n cities in total? How much space do we need to compute the number of different paths that can be taken between two cities? Other resources can be studied as well, such as time, or communication needed to solve a given problem. Interaction between resources is also interesting: If we are allowed to use more space than the strict minimum, can we do with less time? If we are given many computers to do calculations in parallel, can we solve the problem using less time? Given a method to *recognize* a solution to a problem using little time, can we devise a procedure that *finds* such a solution in a comparable amount of time?[4] ... and so on.

This thesis is a collection of answers to a few questions in computational complexity. We use a varied palette of mathematical techniques in order to answer these questions. The general theme is that of solving a problem in computational complexity by applying tools from a different field in theoretical computer science, or at least from a different context within computational complexity.

Chapter 1. In this chapter, we use techniques from Kolmogorov Complexity and Computability in order to study the time complexity of randomized algorithms.

Randomized algorithms are algorithms that are allowed to toss a fair coin during their execution, and have their behavior depend on the outcome of these coin tosses. Such an algorithm is said to solve some problem if it

[2] See wikipedia [9] for the algorithm. The quantity $\log n$ is approximately the number of digits needed to write the number n in decimal notation. The number n itself is the number of digits needed to write x and y.

[3] This is proven by means of a pumping lemma, as in [6, Lemma 3].

[4] For instance, it is easy to quickly recognize if a filled sudoku puzzle on n^2-by-n^2 boards of n-by-n blocks is correct, just by going through each line, each row and each block making sure that there are no repeated numbers (this takes time roughly n^4). But is it possible to actually solve such a sudoku puzzle — find a solution to a partially filled sudoku — in time n^c for some constant c? Answering this question amounts to solving the famous P versus NP problem.

gives the correct answer with high (99%) probability. A normal (deterministic, non-randomized) algorithm can be thought of as a randomized algorithm that always gives the correct answer (with 100% probability), so, in principle, it could be the case that randomized algorithms can solve problems faster than deterministic algorithms, and whether this is the case or not is a longstanding open question in computational complexity.

Kolmogorov Complexity measures how complex a string of symbols is by how large a program to output said string needs to be. A string of symbols x is called *Kolmogorov-random* if there is no program that outputs x and has a length smaller than the length of x. Computability is the study of what can and cannot be solved by computers, regardless of the amount of resources used, and an example of an uncomputable problem is that of knowing whether a given string is Kolmogorov-random. We give some evidence showing that a problem can be solved by a randomized algorithm in polynomial-time if and only if it can be solved by a *deterministic* polynomial-time algorithm that has an unusual extra ability: the ability to know which strings have high Kolmogorov complexity, among a list of strings which it produces after seeing the input (for a precise and technical description of what this means see the technical overview, and the introductory section of this chapter).

Chapter 2. In this chapter we use computational learning theory to prove some results on circuit-size complexity.

Circuit-size complexity measures how big a circuit must be in order to solve a given problem.[5] It is easy to show that most functions are hard to compute by small circuits, and it is widely believed that certain natural problems are hard to solve by small circuits. However, thus far there is no known natural example of such a hard problem, and coming up with such an example — and proving that it cannot be computed by small circuits — is regarded as the holy grail of the field. Complexity theorists have tried to circumvent this difficult problem by showing conditional results, of the form "if problem X can be solved by small circuits, then such and such unlikely consequence must follow"; such results can be thought of as evidence (but not proof) that problem X cannot be solved by small circuits. The result presented in this chapter is a conditional result of this form.

The proof makes use of computational learning theory. This field studies when and how computer programs can be taught to classify data. Typically, the learning algorithm is fed a number examples (x, y) with $y = f(x)$, where f belongs to a restricted class $\mathcal{F}$ of functions, and has to learn which function f was used in generating them. If this can be done efficiently (for instance, if the number of counter-examples needed is small, and/or if the learning algorithm is a polynomial-time algorithm),

[5] By circuit we mean a boolean circuit, which is a mathematical model of the digital logic circuits used in modern computer hardware. For a precise description see `http://en.wikipedia.org/wiki/Boolean_circuit`. We are interested in the size of circuits because if the circuit is too big, then building it will be impossible, or too expensive.

then we say that $\mathcal{F}$ is "learnable". In this chapter we show that if $\mathcal{F}$ is "learnable," then certain natural problems are unlikely to be solvable by certain kinds of circuits that make use of functions from $\mathcal{F}$ (again, see the technical overview for precision).

Chapter 3. Here we use a mix of techniques to show that a problem — called the knapsack problem — cannot be solved by certain kinds of circuits, i.e., that it has high complexity for a certain model of computation.

The model of computation in question is Ketan Mulmuley's parallel PRAM without bit operations. In this model there are several processors executing instructions on a shared memory, which is somewhat similar to a modern GPU (a Graphics Processing Unit, used in modern graphics boards). In some ways, the model is much more powerful than a modern GPU, because it is a multiple-instruction multiple-data (MIMD) model where access to shared memory is instantaneous and conflict-free, and in other ways it is less powerful, because the model is not allowed to operate on the individual bits of the memory, and instead is only allowed to treat numbers stored in memory as self-contained units to which the processors apply given operations (such as addition and multiplication).

The knapsack problem is the problem of, when given a set of n (unbreakable) bars of gold of various sizes, having weights $w_1, \ldots, w_n$, determine the maximum amount of gold that we can carry in a bag which can hold at most W units of weight.

We show that this problem cannot be solved with less than $n^{1/4}$ time and $2^{n^{1/4}}$ processors, even when the given numbers have no more than n bits. Because there is no known parallel algorithm that makes crucial use of bit-access, we believe that our impossibility result also holds for general parallel algorithms.

Chapter 4. In this chapter we use a mix of Information Theory and combinatorial properties of random graphs in order to transform private-coin protocols into public coin protocols that reveal the same amount of information, and prove new direct-sum results in communication complexity.

Communication complexity concerns itself with the following scenario: two parties — customarily called Alice and Bob — are each given two strings of length n as input — respectively x and y — and wish to compute a joint function of this input, i.e., they wish to know $f(x, y)$, for some given f. The function f may depend on both inputs, and so Alice and Bob are required to communicate with each other, which they do according to a predefined set of rules called a *protocol*. We then define the *communication complexity* of f (for length n) as the minimum number of bits of communication required for any protocol that allows the two parties to compute f (on every input of length n). This scenario is ubiquitous, we can think of Alice and Bob as two computers communicating over a network, or as two pieces of hardware communicating over a memory bus, *etc.*

A natural question in this setting is the following *direct-sum question*: suppose that the communication complexity of f is C, i.e., that I need to communicate C bits in order to jointly compute f on all inputs of length n, then what is the communication complexity of computing k copies of f simultaneously? Is it necessarily at least kC, or can we do something smarter?

In a related subject, called "information complexity," we have the same scenario, Alice and Bob wanting to compute a function of their joint input, but now instead of measuring the amount of bits that they need to exchange, we measure the amount of information that those bits reveal about their input. It could happen, for instance, that Alice's part in the protocol requires her to send to Bob the outcome of many random coin tosses, and in this case Bob will receive many bits, but they are completely uncorrelated with Alice's input and hence reveal no information about it; this is an example of (part of) a protocol that has high communication but reveals no information. The *information complexity* of a function f is the minimum number of bits of information that the parties must necessarily reveal to each other about their inputs, when executing any protocol that computes f.

At the meeting of these two subjects, we arrive at the following *protocol compression question*: can we take a protocol that reveals little information about the player's inputs and convert it ("compress it") into a protocol that uses little communication?

When Alice and Bob are allowed to make private coin tosses, it sometimes happens that they are able to use the outcome of these tosses to somehow "obfuscate" their inputs and compute f without revealing a lot of information. However, if they are forced to share the outcome of their coin tosses, it is not clear how to avoid revealing more information. In the literature on the subject, the following question had been asked [4]: whether or not private coin tosses confer any advantage over public coin tosses in this setting.

We will show essentially that these last two questions are intimately related: modulo certain insignificant factors, protocol compression is possible if and only if any private-coin protocol can be simulated by a public-coin protocol that reveals no additional information. We then show that such a public-coin simulation is possible, in the case of bounded-round protocols, and as a consequence we prove a direct-sum property for the communication complexity of bounded-round protocols.

Bibliography

[1] Mark Braverman. Interactive information complexity. In *Proceedings of the 45th STOC*, pages 505–524, 2012.

[2] Jin-Yi Cai. $S_2^p \subseteq \mathrm{ZPP}^{\mathrm{NP}}$. *Journal of Computer and System Sciences*, 73 (1):25–35, 2002.

[3] Russel Impagliazzo and Ramamohan Paturi. The complexity of k-SAT. In *Proceedings of the 14th CCC*, pages 237–240, 1999.

[4] Rahul Jain and Ashwin Nayak. The space complexity of recognizing well-parenthesized expressions in the streaming model: the Index function revisited, 2010. URL `http://arxiv.org/abs/1004.3165`.

[5] Richard Karp and Richard Lipton. Some connections between nonuniform and uniform complexity classes. In *Proceedings of the 12th STOC*, pages 302–309, 1980.

[6] Maciej Liśkiewicz and Rüdiger Reischuk. The complexity world below logarithmic space. In *Proceedings of the 9th CCC (then known as Structures in Complexity Theory)*, pages 64–78, 1994.

[7] Stephen Mahaney. Sparse complete sets for NP: Solution of a conjecture of Berman and Hartmanis. *Journal of Computer and System Sciences*, 25(2): 130–143, 1982.

[8] Ketan Mulmuley. Lower bounds in a parallel model without bit operations. *SIAM Journal on Computing*, 28(4):1460–1509, 1999.

[9] Wikipedia. Multiplication algorithm. URL `http://en.wikipedia.org/wiki/Multiplication_algorithm`.