Universitat de Barcelona
Universitat Politècnica de Catalunya

Master in Pure and Applied Logic

Master's Thesis

A Formalization of Kannan's Circuit Lower Bound in Bounded Arithmetic

Author:
Carlos Cantero

Supervisor:
Albert Atserias

September 11, 2024

Universitat de Barcelona

Universitat Politècnica de Catalunya
BarcelonaTech

To Ari

Abstract

The aim of this work is to formalize the circuit-size lower bound showed by Kannan in 1982 in a weak theory for feasible computations. In particular, we will work with theories of bounded arithmetic, which are subtheories of Peano Arithmetic that weaken its induction axiom scheme by restricting it to formulas in which the quantifiers are bounded. Kannan's circuit lower bound states that for every fixed polynomial size of circuits, there is a language in the second level of the polynomial hierarchy that cannot be decided by circuits of that size. We note that the essential ingredient in this proof is a key use of the weak pigeonhole principle, which is available in bounded arithmetic. Instrumental in the proof of Kannan's Theorem is the celebrated Karp-Lipton's Theorem, stating that if the satisfiability problem for propositional formulas can be decided by polynomial-size circuits then the polynomial hierarchy collapses to its second level, which we also formalize in the same theory.

Acknowledgements

I would like to thank every person that has helped me with the elaboration of this work. Most of all, I want to express my gratitude to my advisor, Albert Atserias, for helping me with every step of the process and for solving a myriad of doubts and problems. This thesis would not have been remotely possible without him.

I would also like to thank my parents, for supporting me during this whole Master's Degree; I owe them a lot. Finally, I would like to thank my classmates, for their invaluable help and their stimulating conversations. This Master's would not have been the same without them.

Contents

Chapter 1

Introduction

This work revolves around a result proven in 1982 by Ravindran Kannan. Theorem 2 of [Kan82] shows that for every $c > 0$, there is a language in one of the lowest levels of the polynomial hierarchy that does not have circuits of size a polynomial with exponent c. More precisely, Kannan's result is that for every $c > 0$, there exists a language K^c in Σ_2^{P} such that:

$$K^c \notin \mathrm{SIZE}(n^c). \tag{1.1}$$

In the same article, Kannan also proved a superpolynomial circuit lower bound for a single language L^* in the complexity class Σ_2^{E}, thus showing the following today well-known separation of complexity classes:

$$\Sigma_2^{\mathrm{E}} \not\subseteq \mathrm{P/poly}. \tag{1.2}$$

Our purpose is to adapt the proof of (1.1) so it can be proven by a weak theory. In particular, we will work with theories of bounded arithmetic, which are subtheories of Peano Arithmetic that weaken its induction axiom scheme by restricting it to formulas in which the quantifiers are bounded.

The study of the class P/poly and of circuit lower bounds has been pursued since the 80's, but the fundamental questions of this field (for example, NP $\subseteq^?$ P/poly) remain unanswered, as do the fundamental questions of uniform complexity (such as P $=^?$ NP). In part with the purpose of showing the underlying principles of known proofs to shed some light on our knowledge of complexity theory so we may be closer to answer those questions, a program dedicated to proving known results in weak theories emerged. This program, whose principles can be read in [Kra95], [CN10] or [Kra19], is the motivation for this work.

In particular, the program uses the bounded arithmetic theories defined by Sam Buss in [Bus86]. Some recent references that show the consistency of *conjectured* circuit lower bounds with certain weak theories of bounded arithmetic are [KO17] and [ABM23]. Since any true theory is consistent with any true statement, the theories studied in these works are, of course, also consistent with the *known* circuit lower bounds, such as Kannan's result. What the line of work started by those articles did not clarify is whether these weak theories were strong enough to *prove* the known circuit lower bounds. Our motivation for this work is to show that, at least for Kannan's result, they are. This is why we will formalize the proof of (1.1) in a weak bounded arithmetic theory: concretely, we will show T_2 proves (1.1).

We will also present a detailed proof of (1.2), but we will not formalize it in T_2 because it involves second order theories of bounded arithmetic and concepts that are out of the scope of this thesis. Nevertheless, we want to lay the foundations so the proof can be fully adapted on a further work; we believe the full proof of (1.2) is formalizable in the theory V_2^0.

In Chapter 2, we present the definitions and results of complexity theory needed to understand the rest of the material, as well as the concepts of bounded arithmetic that we will use in our formalizations. We have tried to make this work as self-contained as possible. Even though we will not work with second order bounded arithmetic theories, we define them, both for completeness and to make it easier to develop our formalization in the future.

Originally proven in [KL80], the Karp-Lipton Theorem is essential to the proof of (1.1), which is why we dedicate Chapter 3 to presenting the proof of Karp-Lipton in detail. In Chapter 4 we begin by presenting a version of the original proof of (1.2), which is unable to prove (1.1) due to its use of truth tables of exponential size, making it also impossible to be formalized in first-order bounded arithmetic. This is why in Section 4.3 we adapt the latter proof using more restricted means: in this way, we end up proving both (1.1) and (1.2). This adapted proof of (1.1) will be formalizable in T_2.

Chapters 5 and 6 are where the formalization in bounded arithmetic is carried out. As a sort of warm up and to present some core concepts that will be used in the final chapter, in Chapter 5 we formalize the Karp-Lipton Theorem. Finally, in Chapter 6, we formalize our adaptation of the proof of (1.1), Kannan's result. As announced, we leave out the formalization of the proof of (1.2), since it is not formalizable in first-order bounded arithmetic.

Chapter 2

Preliminaries

2.1 Notation and Conventions

When we talk about numbers, unless otherwise stated we will be referring to natural numbers (including 0). We will normally use:

The letters n, m, i, j, a, b, c, d (and variations of them — e.g. $n_1, n_2, n_3, \ldots$) to denote numbers.

The letters x, y, z, u, v, w to denote binary strings.

The letter L to denote sets of binary strings, or languages.

The letter M to denote Turing Machines.

The letters C, D to denote Boolean circuits.

The letters f, g, h to denote functions.

The letters t, s to denote terms.

Greek letters $\varphi, \psi, \chi, \ldots$ to denote formulas. The term *propositional formula* will denote exclusively quantifier-free propositional formulas, otherwise we will say *quantified formula*. The same stands for sentences.

The overlined letters $\overline{x}, \overline{y}, \overline{z}$ to denote sequences of propositional variables. Sometimes, (variations of) the letter x will be used to denote single propositional variables, but such a use will be made explicit by using the notation when it can be ambiguous. $\varphi(\overline{x})$ tacitly assumes that all the free variables in φ are all included in $\overline{x}$.

Definition 1. $\{0,1\}^*$ represents the set of all binary strings. For every $n > 0$, the set of all strings of length n is represented by $\{0,1\}^n$.

Definition 2. When applied to a string, the symbol $|\cdot|$ will represent the length of the string. For example, if $x \in \{0,1\}^n$, then $|x| = n$.

Definition 3. When applied to a number of strings, the symbol $\langle\cdot\rangle$ represents the application of a suitable coding of tuples. For example, $\langle x, y\rangle$ returns a unique string representing the pair (x, y). For example, we may take $\langle x, y\rangle = 1^{|x|}0xy$.

Definition 4. Given $i > 0$, we define $Q_i = \forall$ if i is even and $Q_i = \exists$ if i is odd. Conversely, $\overline{Q}_i = \exists$ if i is even and $\overline{Q}_i = \forall$ if i is odd.

Definition 5. For $i > 0$, we will say φ is a Σ_i-formula if it is of the following form:

$$\exists \overline{x}_1 \forall \overline{x}_2 \ldots Q_i \overline{x}_i \psi(\overline{x}_1 \ldots \overline{x}_i),$$

where ψ is a propositional formula. Similarly, we will say ϕ is a Π_i-formula if it is of the following form:

$$\forall \overline{x}_1 \exists \overline{x}_2 \ldots \overline{Q}_i \overline{x}_i \psi(\overline{x}_1 \ldots \overline{x}_i),$$

where ψ is a propositional formula. If a formula does not have quantifiers we will say it is quantifier-free, or QF.

Definition 6. If $\varphi(\overline{x}_1, \overline{x}_2)$ is a quantified formula and $v \in \{0,1\}^{|\overline{x}_1|}$, then $\varphi(\overline{x}_1/v, \overline{x}_2)$ will denote the formula obtained by substituting every occurrence of the propositional variables $\overline{x}_1$ in $\varphi(\overline{x}_1, \overline{x}_2)$ by the corresponding bits in the binary string v. Sometimes, we will just write $\varphi(v, \overline{x}_2)$ if it is clear what we are substituting.

Definition 7. Let $f : \mathbb{N} \to \mathbb{N}$ and $g : \mathbb{N} \to \mathbb{N}$ be functions. We say $f = \mathcal{O}(g)$ if there exist $c, n_0 \in \mathbb{N}$ such that for every $n > n_0$, we have $f(n) \leq c \cdot g(n)$.

2.2 Computational Complexity

In this section, we will present the main definitions and results of Computational Complexity that we will use during the rest of the work. Most of the definitions are adapted from [AB09].

2.2.1 Turing Machines

We will work with single-tape single-head Turing Machines.

Definition 8. A Turing Machine (TM) M is defined by a triple (Γ, Q, δ), where:

Γ is a finite set of symbols the tape of M can contain. It is known as the *alphabet* of M. We assume it only contains a blank symbol and the symbols 0 and 1.

Q is the set of possible states M can be in. We assume it contains two designated states: q_{start} and q_{halt}.

$\delta : Q \times \Gamma \to Q \times \Gamma \times \{L, S, R\}$ is the transition function, a function describing the behaviour of M: given the state M is in and the symbol its head is reading, δ defines the next state of M, the symbol its head will overwrite in the tile it is currently in and whether the head will move left, stay still or move right.

The *input* of a TM M is the binary string written on its tape in the beginning of the computation, when M's state is q_{start} and the head points to its first symbol.

The *output* of a TM M on a given input is the longest binary string written on its tape to the left of its head when the machine halts, namely when M enters state q_{halt} (if M does not halt, then the output is not defined). If the output of M on input $x \in \{0,1\}^*$ is $y \in \{0,1\}^*$, we use the notation $M(x) = y$. Note sometimes we will use the tuple coding implicitly: for example, $M(x_1, x_2, x_3) = y$ will represent $M(\langle x_1, x_2, x_3 \rangle) = y$.

We say M *accepts* on input $x \in \{0,1\}^*$ if it enters q_{halt} and outputs 1, or $M(x) = 1$. We say M *rejects* on input x when $M(x) = 0$.

Definition 9. Let $f : \{0,1\}^* \to \{0,1\}^*$ and $g : \mathbb{N} \to \mathbb{N}$ be functions, let M be a TM and let L be a language. We say M *computes* f if for every $x \in \{0,1\}^*$, we have the computation of M on input x enters state q_{halt} and $M(x) = f(x)$. Similarly, we say M *decides* L if for every $x \in \{0,1\}^*$, we have the computation of M on input x enters state q_{halt} and $x \in L$ iff $M(x) = 1$.

We say M runs in $g(n)$-time if on every input $x \in \{0,1\}^*$ it reaches the state q_{halt} in at most $g(|x|)$ steps. If M runs in $h(n)$-time for some $h : \mathbb{N} \to \mathbb{N}$ such that $h = \mathcal{O}(g)$, we can say M runs in $\mathcal{O}(g(n))$-time. In particular, if M runs in $\mathcal{O}(n^c)$ time for some $c > 0$, we say that M runs in polynomial time, or that M is a polynomial-time TM.

Definition 10. Let $f : \mathbb{N} \to \mathbb{N}$ be a function. Then, DTIME($f(n)$) is a class of languages defined in the following way: $L \in$ DTIME($f(n)$) iff there is some TM M such that L is decided by M and M runs in $\mathcal{O}(f(n))$-time.

Definition 11.

$$\mathrm{P} \stackrel{def}{=} \bigcup_{c>0} \mathrm{DTIME}(n^c).$$

$$\mathrm{E} \stackrel{def}{=} \bigcup_{c>0} \mathrm{DTIME}(2^{cn}).$$

$$\mathrm{EXP} \stackrel{def}{=} \bigcup_{c>0} \mathrm{DTIME}(2^{n^c}).$$

Definition 12. Given two languages A and B, the *disjoint union* $A \oplus B$ is defined as follows: for all $x \in \{0,1\}^*$, we have

$$x \in A \iff \langle x, 0\rangle \in A \oplus B$$
$$x \in B \iff \langle x, 1\rangle \in A \oplus B.$$

Definition 13. A language A is polynomial-time many-one reducible to a language B (denoted by $A \leq_{\mathrm{m}}^{\mathrm{P}} B$) when there exists a polynomial-time computable function $f : \{0,1\}^* \to \{0,1\}^*$ such that for all $x \in \{0,1\}^*$, we have $x \in A$ iff $f(x) \in B$. We say f witnesses $A \leq_{\mathrm{m}}^{\mathrm{P}} B$, or $f : A \leq_{\mathrm{m}}^{\mathrm{P}} B$.

Given a class of languages $\mathcal{C}$ and a language L, we say L is *$\mathcal{C}$-hard* when for all $L' \in \mathcal{C}$, we have $L' \leq_{\mathrm{m}}^{\mathrm{P}} L$. If L is $\mathcal{C}$-hard and $L \in \mathcal{C}$, we say L is *$\mathcal{C}$-complete*.

2.2.2 Nondeterministic Turing Machines

Definition 14. A Nondeterministic Turing Machine (NDTM) M is defined as a TM, with the exception that instead of a single transition function δ, the machine M has two different transition functions δ_0 and δ_1: in each step of a given computation, M chooses arbitrarily which one of the two transition functions to apply. Furthermore, instead of q_{halt}, the set of states of M has two halting states: q_{accept} and q_{reject}.

The *input* of an NDTM is defined as in the case of a TM. We say an NDTM M *accepts* on input x when there exists a sequence of nondeterministic choices (decisions between the transition functions) such that M enters the state q_{accept}, and we denote that by $M(x) = 1$. We say an NDTM M *rejects* on input x when for every sequence of nondeterministic choices, M enters the state q_{reject}, and we denote that by $M(x) = 0$. Note we have not defined the *output* of a NDTM.

Definition 15. Let $f : \{0,1\}^* \to \{0,1\}^*$ be a function, let M be a NDTM and let L be a language. We say M *decides* L if for every $x \in \{0,1\}^*$, we have that every computation of

M on input x enters a halting state and that $x \in L$ iff $M(x) = 1$.

We say M runs in $f(n)$-time if on every input $x \in \{0,1\}^*$, for every sequence of nondeterministic choices, M reaches either q_{accept} or q_{reject} in at most $f(|x|)$ steps. If M runs in $h(n)$-time for some $h : \mathbb{N} \to \mathbb{N}$ such that $h = \mathcal{O}(f)$, we can say M runs in $\mathcal{O}(f(n))$-time. In particular, if M runs in $\mathcal{O}(n^c)$ time for some $c > 0$, we say that M runs in polynomial time, or that M is a polynomial-time NDTM.

Definition 16. Let $f : \mathbb{N} \to \mathbb{N}$ be a function. Then, NTIME($f(n)$) is a class of languages defined in the following way: $L \in \text{NTIME}(f(n))$ iff there is some NDTM M such that L is decided by M and M runs in $\mathcal{O}(f(n))$-time.

Definition 17.

$$\text{NP} \stackrel{def}{=} \bigcup_{c>0} \text{NTIME}(n^c).$$

$$\text{NE} \stackrel{def}{=} \bigcup_{c>0} \text{NTIME}(2^{cn}).$$

$$\text{NEXP} \stackrel{def}{=} \bigcup_{c>0} \text{NTIME}(2^{n^c}).$$

2.2.3 Oracle Turing Machines

Definition 18. Given a language O, a TM M with *oracle* O is defined as a TM but with an additional tape (called oracle tape) and three selected states q_{query}, q_{yes} and q_{no}. When M enters the state q_{query}, in one step it reads the contents of the oracle tape and moves to state q_{yes} if $x \in O$, or to state q_{no} if $x \notin O$.

Oracle NDTMs are defined in a similar way.

Definition 19. Given a language O, the class of languages NP^O is defined in the following way: $L \in \text{NP}^O$ iff there is a polynomial-time NDTM M with oracle O deciding L. In case O is $\mathcal{C}$-complete, we will denote NP^O by $\text{NP}^{\mathcal{C}}$.

This allows us to define the following complexity classes: NP^{NP}, $\text{NP}^{\text{NP}^{\text{NP}}}$, $\text{NP}^{\text{NP}^{\text{NP}^{\text{NP}}}}$, ...

2.2.4 The Polynomial Hierarchy

Definition 20. For $i > 0$, the class of languages Σ_i^{P} is defined in the following way: $L \in \Sigma_i^{\mathrm{P}}$ iff there exists a polynomial-time TM M and a polynomial q such that

$$x \in L \iff \exists v_1 \in \{0,1\}^{q(|x|)} \forall v_2 \in \{0,1\}^{q(|x|)} \ldots Q_i v_i \in \{0,1\}^{q(|x|)} M(x, v_1, \ldots, v_i) = 1,$$

where $M(x, v_1, \ldots, v_i)$ abbreviates $M(\langle x, v_1, \ldots, v_i \rangle)$ (we will omit $\langle \cdot \rangle$ in similar cases).

Similarly, Π_i^{P} is defined as follows: $L \in \Pi_i^{\mathrm{P}}$ iff there exists a polynomial-time TM M and a polynomial q such that

$$x \in L \iff \forall v_1 \in \{0,1\}^{q(|x|)} \exists v_2 \in \{0,1\}^{q(|x|)} \ldots \overline{Q}_i v_i \in \{0,1\}^{q(|x|)} M(x, v_1, \ldots, v_i) = 1.$$

Furthermore, we define $\Sigma_0^{\mathrm{P}} = \Pi_0^{\mathrm{P}} = \mathrm{P}$.

Then, the *Polynomial Hierarchy* is defined in the following way:

$$\mathrm{PH} \stackrel{def}{=} \bigcup_{i>0} \Sigma_i^{\mathrm{P}} = \bigcup_{i>0} \Pi_i^{\mathrm{P}}.$$

Theorem 1. *For $i > 0$, we have:*

$$\Sigma_i^{\mathrm{P}} = \mathrm{NP}^{\mathrm{NP}^{\cdot^{\cdot^{\cdot^{\mathrm{NP}}}}}},$$

where $\mathrm{NP}^{\cdot^{\cdot^{\cdot^{\mathrm{NP}}}}}$ *represents an exponential tower of $i-1$ NPs. That is,* $\Sigma_1^{\mathrm{P}} = \mathrm{NP}$, $\Sigma_2^{\mathrm{P}} = \mathrm{NP}^{\mathrm{NP}}$, $\Sigma_3^{\mathrm{P}} = \mathrm{NP}^{\mathrm{NP}^{\mathrm{NP}}}$, $\Sigma_4^{\mathrm{P}} = \mathrm{NP}^{\mathrm{NP}^{\mathrm{NP}^{\mathrm{NP}}}}$, *etc.*

2.2.5 The Exponential Hierarchy

Definition 21. For $i > 0$, the class of languages Σ_i^{E} is defined in the following way: $L \in \Sigma_i^{\mathrm{E}}$ iff there exists a polynomial-time TM M and a $c > 0$ such that

$$x \in L \iff \exists v_1 \in \{0,1\}^{2^{c|x|}} \forall v_2 \in \{0,1\}^{2^{c|x|}} \ldots Q_i v_i \in \{0,1\}^{2^{c|x|}} M(x, v_1, \ldots, v_i) = 1,$$

And Π_i^{E} is defined as follows: $L \in \Pi_i^{\mathrm{E}}$ iff there exists a polynomial-time TM M and a $c > 0$ such that

$$x \in L \iff \forall v_1 \in \{0,1\}^{2^{c|x|}} \exists v_2 \in \{0,1\}^{2^{c|x|}} \ldots \overline{Q}_i v_i \in \{0,1\}^{2^{c|x|}} M(x, v_1, \ldots, v_i) = 1.$$

Additionally, we define $\Sigma_0^{\mathrm{E}} = \Pi_0^{\mathrm{E}} = \mathrm{E}$.

Then, the *Exponential Hierarchy* is defined in the following way:

$$\mathrm{EH} \stackrel{def}{=} \bigcup_{i>0} \Sigma_i^{\mathrm{E}} = \bigcup_{i>0} \Pi_i^{\mathrm{E}}.$$

Definition 22. Given a language O, the class of languages NE^O is defined in the following way: $L \in \mathrm{NE}^O$ iff there is a NDTM M with oracle O such that M decides L and M runs in time $\mathcal{O}(2^{cn})$ for some $c > 0$. As before, in case O is $\mathcal{C}$-complete, we will denote NE^O by $\mathrm{NE}^{\mathcal{C}}$.

This allows us to define the following complexity classes: $\mathrm{NE}^{\mathrm{NP}}$, $\mathrm{NE}^{\mathrm{NP}^{\mathrm{NP}}}$, $\mathrm{NE}^{\mathrm{NP}^{\mathrm{NP}^{\mathrm{NP}}}}$, ...

And as before, we have a Theorem relating both definition schemes:

Theorem 2. *For $i > 0$, we have:*

$$\Sigma_i^{\mathrm{E}} = \mathrm{NE}^{\mathrm{NP}^{\cdot^{\cdot^{\cdot^{\mathrm{NP}}}}}},$$

where $\mathrm{NP}^{\cdot^{\cdot^{\cdot^{\mathrm{NP}}}}}$ *represents an exponential tower of $i-1$* NP*s. That is,* $\Sigma_1^{\mathrm{E}} = \mathrm{NE}$, $\Sigma_2^{\mathrm{E}} = \mathrm{NE}^{\mathrm{NP}}$, $\Sigma_3^{\mathrm{E}} = \mathrm{NE}^{\mathrm{NP}^{\mathrm{NP}}}$, $\Sigma_4^{\mathrm{E}} = \mathrm{NE}^{\mathrm{NP}^{\mathrm{NP}^{\mathrm{NP}}}}$, *etc.*

2.2.6 Boolean Circuits

Definition 23. A *circuit* is a directed acyclic graph, the nodes of which are either input nodes, with fan-in 0, or nodes representing the boolean connectives (called gates), with fan-in 2 in the case of AND and OR, and fan-in 1 in the case of NOT. Some of the gates are declared output nodes. A *n-input circuit* is a circuit with exactly n input nodes.

The *output* of a n-input circuit C on an input x of length n, denoted by $C(x)$, is computed in the usual way (AND gates output 1 iff its two inputs are 1, etc.).

Definition 24. The *size* of a circuit is the number of nodes of the circuit. For $f : \mathbb{N} \to \mathbb{N}$, a *$f(n)$-size circuit family* is a sequence $\{C_n\}_{n\in\mathbb{N}}$ such that the size of C_n is at most $f(n)$, where C_n is a n-input circuit.

Definition 25. For $f : \mathbb{N} \to \mathbb{N}$, we have $\mathrm{SIZE}(f(n))$ is a class of languages, where $L \in \mathrm{SIZE}(f(n))$ iff there exists a function $g : \mathbb{N} \to \mathbb{N}$ such that $g = \mathcal{O}(f(n))$ and there is a $g(n)$-size circuit family $\{C_n\}_{n\in\mathbb{N}}$ such that for every $n \in \mathbb{N}$ and every $x \in \{0,1\}^n$, we have $x \in L \Leftrightarrow C_n(x) = 1$. In this case, we say $\{C_n\}_{n\in\mathbb{N}}$ is an $\mathcal{O}(f(n))$-size circuit family deciding L.

The following is only one of the possible characterizations of P/poly.

Definition 26.

$$\mathrm{P/poly} \stackrel{def}{=} \bigcup_{c \in \mathbb{N}} \mathrm{SIZE}(n^c).$$

We say a language L has small circuits when $L \in \mathrm{P/poly}$.

Definition 27. A language A is polynomial circuit many-one reducible to a language B (denoted by $A \leq_{\mathrm{m}}^{\mathrm{P/poly}} B$) when there exists a function $f : \{0,1\}^* \to \{0,1\}^*$ such that for all $x \in \{0,1\}^*$, we have $x \in A$ iff $f(x) \in B$ and there exists c such that there is an $\mathcal{O}(n^c)$-size circuit family $\{C_n\}_{n \in \mathbb{N}}$ computing f (that is, for every n and for every $x \in \{0,1\}^n$ we have $C_n(x) = f(x)$). We say f witnesses $A \leq_{\mathrm{m}}^{\mathrm{P/poly}} B$, or $f : A \leq_{\mathrm{m}}^{\mathrm{P/poly}} B$.

The following Lemma follows from Theorem 2.8 of [Vol99]:

Lemma 1. *Let $f : \{0,1\}^* \to \{0,1\}^*$ be a function. If f is computable by a TM M that runs in $\mathcal{O}(n^c)$-time, then f is computable by a $\mathcal{O}(n^{2c})$-size circuit family.*

2.2.7 The Cook-Levin Theorem

The following theorem is a generalization of the well-known Cook-Levin Theorem that follows from the same proof. It will be of importance in many places of this work.

Theorem 3. *There are two polynomially computable functions f and g such that for every Turing Machine M, every $k > 0$, all $n_1, \ldots, n_k > 0$ and every $t > 0$, we have:*

$f(M, 1^{n_1}, \ldots, 1^{n_k}, 1^t)$ *outputs a propositional formula* $\varphi(\overline{x}_1, \ldots, \overline{x}_k, \overline{z})$,

$g(M, 1^{n_1}, \ldots, 1^{n_k}, 1^t)$ *outputs a propositional formula* $\psi(\overline{x}_1, \ldots, \overline{x}_k, \overline{z})$,

where $|\overline{x}_i| = n_i$ for $1 \leq i \leq k$ and $|\overline{z}| = p(|M|, n_1, \ldots, n_k, t)$ for some polynomial p, such that for every TM M, every k, all $n_1, ..., n_k$ and all $x_1 \in \{0,1\}^{n_1}, ..., x_k \in \{0,1\}^{n_k}$, we have that the following statements are equivalent:

1. *$M(x_1, \ldots, x_k)$ accepts in t steps.*
2. *$\exists z \in \{0,1\}^{|\overline{z}|} \varphi(\overline{x}_1/x_1, \ldots, \overline{x}_k/x_k, \overline{z}/z)$.*
3. *$\forall z \in \{0,1\}^{|\overline{z}|} \psi(\overline{x}_1/x_1, \ldots, \overline{x}_k/x_k, \overline{z}/z)$.*

The usual statement of the Cook-Levin Theorem states that SAT, the satisfiability problem for propositional formulas, is NP-complete. This is a special case of Theorem 3 by noting that if L is any language in NP defined for every x by

$$x \in L \iff \exists y \in \{0,1\}^{q(|x|)} M(x,y) = 1,$$

where p is a polynomial and M is a TM that runs in $p(|x|)$ steps on all inputs $\langle x, y\rangle$ with $|y| = q(|x|)$, then the map $x \mapsto f(M, 1^{|x|}, 1^{q(|x|)}, 1^{p(|x|)})$ witnesses that $L \leq_{\mathrm{m}}^{\mathrm{P}} \mathrm{SAT}$.

2.3 Bounded Arithmetic

In this section, we will present the definitions of the bounded arithmetic theories T_2^i, S_2^i and V_2^i (for $i \in \mathbb{N}$) initially given in [Bus86]. We will follow the the definitions given in [ABM23], which differ from the original one in that they enrich the language of bounded arithmetic theories with polynomial-time computable functions (this will be explain in more detail later). First, we will present the background needed to understand one-sorted theories and define T_2^i and S_2^i; next, we will give background for two-sorted theories, define the two-sorted versions of T_2^i and S_2^i, and define V_2^i; finally, we will comment on some differences between the way V_2^i is defined on [ABM23] and in the main references on proof complexity.

2.3.1 One-sorted Theories

Definition 28. L_{BA}, the language of bounded arithmetic, is defined as follows:

$$L_{\mathrm{BA}} \stackrel{def}{=} \{0, 1, \leq, +, \cdot, \lfloor /2 \rfloor, \#, |\ |\},$$

where 0 and 1 are constant symbols, $\leq$ is a binary relation symbol; $+$, $\cdot$ and $\#$ are binary function symbols, and $\lfloor /2 \rfloor$ and $|\ |$ are unary function symbols. Its standard interpretation is the usual interpretation of those symbols over $\mathbb{N}$, where $|x|$ is the bit-length of x and $x\#y$ means $2^{|x|\cdot|y|}$. We will use the notation $2^{|x|}$ as an abbreviation for $x\#1$. L_{BA} is a first-order-logic language, so it contains built-in equality.

In this subsection, let t denote an L_{BA}-term and φ, ψ denote L_{BA}-formulas.

Definition 29. BASIC is the set of the universal closures of the following formulas:

1. $x \leq y \to x \leq y + 1$
2. $x \neq x + 1$

3. $0 \leq x$
4. $(x \leq y \wedge x \neq y) \rightarrow x + 1 \leq y$
5. $x \neq 0 \rightarrow 2x \neq 0$
6. $x \leq y \vee y \leq x$
7. $(x \leq y \wedge y \leq x) \rightarrow x = y$
8. $(x \leq y \wedge y \leq z) \rightarrow x \leq z$
9. $|0| = 0$
10. $x \neq 0 \rightarrow$
    $(|2x| = |x| + 1 \wedge |2x + 1| = |x| + 1)$
11. $|1| = 1$
12. $x \leq y \rightarrow |x| \leq |y|$
13. $|x\#y| = (|x| \cdot |y|) + 1$
14. $0\#x = 1$
15. $x \neq 0 \rightarrow (1\#(2x) = 2(1\#x) \wedge$
    $1\#(2x + 1) = 2(1\#x))$
16. $x\#y = y\#x$
17. $|x| = |y| \rightarrow x\#z = y\#z$
18. $|x| = |y| + |z| \rightarrow x\#u = (y\#u) \cdot (z\#u)$
19. $x \leq x + y$
20. $(x \leq y \wedge x \neq y) \rightarrow (2x+1 \leq 2y \wedge 2x+1 \neq$
    $2y)$
21. $x + y = y + x$
22. $x + 0 = x$
23. $x + (y + 1) = (x + y) + 1$
24. $(x + y) + z = x + (y + z)$
25. $x + y \leq x + z \rightarrow y \leq z$
26. $x \cdot 0 = 0$
27. $x \cdot (y + 1) = (x \cdot y) + x$
28. $x \cdot y = y \cdot x$
29. $x \cdot (y + z) = (x \cdot y) + (x \cdot z)$
30. $1 \leq x \rightarrow (x \cdot y \leq x \cdot z \leftrightarrow y \leq z)$
31. $x \neq 0 \rightarrow |x| = |\lfloor x/2 \rfloor| + 1$
32. $x = \lfloor y/2 \rfloor \leftrightarrow (2x = y \vee 2x + 1 = y)$,

where $2t := (1 + 1) \cdot t$.

Definition 30. We define the following notation:

$$\exists x \leq t \ \varphi \stackrel{def}{=} \ \exists x(x \leq t \wedge \varphi),$$

$$\forall x \leq t \ \varphi \stackrel{def}{=} \ \forall x(x \leq t \rightarrow \varphi).$$

Quantifiers that appear in the form $\exists x \leq t$ or $\forall x \leq t$ are called *bounded quantifiers*. When $t = |s|$ for some term s, they are called *sharply bounded quantifiers*.

Let $\exists x < t \ \varphi := \exists x(x \leq t \wedge x \neq t \wedge \varphi)$ and $\forall x < t \ \varphi := \forall x(x \leq t \wedge x \neq t \rightarrow \varphi)$.

Definition 31. The sets of formulas Σ^{b}_i and Π^{b}_i are defined by counting alternations of bounded quantifiers and ignoring sharply bounded quantifiers. More precisely:

$\Sigma^{\mathrm{b}}_0 = \Pi^{\mathrm{b}}_0$ is the set of all formulas in which all quantifiers are sharply bounded.

$\Sigma^{\mathrm{b}}_{i+1}$ is the smallest set of formulas such that:

1. $\Sigma^{\mathrm{b}}_i \cup \Pi^{\mathrm{b}}_i \subseteq \Sigma^{\mathrm{b}}_{i+1}$
2. $\varphi, \psi \in \Sigma^{\mathrm{b}}_{i+1} \implies \varphi \wedge \psi \in \Sigma^{\mathrm{b}}_{i+1}$[1]
3. $\varphi \in \Pi^{\mathrm{b}}_{i+1} \implies \neg\varphi \in \Sigma^{\mathrm{b}}_{i+1}$
4. $\varphi \in \Sigma^{\mathrm{b}}_{i+1} \implies \exists x \leq |t|\ \varphi \in \Sigma^{\mathrm{b}}_{i+1}$ and $\forall x \leq |t|\ \varphi \in \Sigma^{\mathrm{b}}_{i+1}$
5. $\varphi \in \Sigma^{\mathrm{b}}_{i+1} \implies \exists x \leq t\ \varphi \in \Sigma^{\mathrm{b}}_{i+1}$

Π^{b}_{i+1} is the smallest set of formulas such that:

1. $\Sigma^{\mathrm{b}}_i \cup \Pi^{\mathrm{b}}_i \subseteq \Pi^{\mathrm{b}}_{i+1}$
2. $\varphi, \psi \in \Pi^{\mathrm{b}}_{i+1} \implies \varphi \wedge \psi \in \Pi^{\mathrm{b}}_{i+1}$
3. $\varphi \in \Sigma^{\mathrm{b}}_{i+1} \implies \neg\varphi \in \Pi^{\mathrm{b}}_{i+1}$
4. $\varphi \in \Pi^{\mathrm{b}}_{i+1} \implies \exists x \leq |t|\ \varphi \in \Pi^{\mathrm{b}}_{i+1}$ and $\forall x \leq |t|\ \varphi \in \Pi^{\mathrm{b}}_{i+1}$
5. $\varphi \in \Pi^{\mathrm{b}}_{i+1} \implies \forall x \leq t\ \varphi \in \Pi^{\mathrm{b}}_{i+1}$

$\Sigma^{\mathrm{b}}_\infty = \Pi^{\mathrm{b}}_\infty \stackrel{def}{=} \bigcup_{i\in\omega} \Sigma^{\mathrm{b}}_i = \bigcup_{i\in\omega} \Pi^{\mathrm{b}}_i$

Given a set of formulas Φ, by a Φ-formula φ we mean a formula $\varphi \in \Phi$.

Definition 32. Given a set of formulas Φ, the scheme of bounded Φ-induction (Φ-IND) is the set of the universal closures of formulas of the following form:

$$\Phi\text{-IND}(N, \varphi) \stackrel{def}{=} \ \varphi(0) \wedge \forall y < N\ (\varphi(y) \to \varphi(y+1)) \to \varphi(N),$$

where $\varphi \in \Phi$ may have additional parameters z.

The scheme of bounded Φ-length-induction (Φ-LIND) is the set of the universal closures of formulas of the following form:

$$\Phi\text{-LIND}(N, \varphi) \stackrel{def}{=} \ \varphi(0) \wedge \forall y < |N|\ (\varphi(y) \to \varphi(y+1)) \to \varphi(|N|),$$

where $\varphi \in \Phi$ may have additional parameters z.

[1]For concision, let us consider only $\wedge$ and $\neg$ as primitive Boolean symbols and define $\vee$ and $\to$ in the usual way.

Definition 33. For each $i \geq 0$, we have:

- The theory T_2^i is axiomatized by BASIC + Σ_i^{b}-IND.
- The theory S_2^i is axiomatized by BASIC + Σ_i^{b}-LIND.

And full bounded arithmetic is defined as follows:

$$T_2 \stackrel{def}{=} \bigcup_{i \geq 0} T_2^i.$$

The 2 in the subscript indicates the presence of the symbol # in the language.

Definition 34. Δ_1^{b} is a set of formulas defined in the following way: $\varphi \in \Delta_1^{\mathrm{b}}$ iff $S_2^1 \vdash \varphi \leftrightarrow \psi$ and $S_2^1 \vdash \varphi \leftrightarrow \chi$ for some $\psi \in \Sigma_1^{\mathrm{b}}$ and $\chi \in \Pi_1^{\mathrm{b}}$.

Normally we would define Δ_i^{b} with respect to a theory T, but in this work we will only use Δ_1^{b} with respect to S_2^1.

As we announced at the beginning of the section, we enrich L_{BA} with Δ_1^{b}-definable functions. That is:

Definition 35. For each function that is definable by a Δ_1^{b}-formula, we will assume L_{BA} has a term representing that function. It is usual to denote this enriched bounded arithmetic language by $L_{\mathrm{BA}}(\mathrm{PV})$, and the corresponding theories by $T_2^i(\mathrm{PV})$ and $S_2^i(\mathrm{PV})$: that is, for each function f whose graph is definable by a Δ_1^{b}-formula $\varphi(\overline{x}, y)$, we will assume that the language $L_{\mathrm{BA}}(\mathrm{PV})$ has a term representing $f(\overline{x}) = y$ and the corresponding theories have an axiom stating $f(\overline{x}) = y \leftrightarrow \varphi(\overline{x}, y)$.

To gain some concision (following [ABM23]), in this work we will omit the (PV). Thus, when we write T_2, what we will really be referring to is the $L_{\mathrm{BA}}(\mathrm{PV})$-theory $T_2(\mathrm{PV})$, and when we talk about L_{BA}, we will really be talking about $L_{\mathrm{BA}}(\mathrm{PV})$. Since by Buss' Theorem [Bus86] the Δ_1^{b}-definable functions are precisely the polynomial-time computable functions, this will allow us to freely define polynomial-time computable functions and predicates by the polynomial-time TM that computes them and assume that the language $L_{\mathrm{BA}}(\mathrm{PV})$ contains symbols for them.

A problem that might arise when doing so is that the properties that we would want the symbols defined this way to have may not be provable in T_2. A way around this problem

is to restrict our use of this feature to very simple functions and predicates, so that it is elementary to prove in T_2 the symbols that define them fulfill the required properties. For example, this is how we will define the projection function:

$\mathrm{pr}(x,i) \stackrel{def}{=}$ if x represents a string of length at most i, returns the i-th bit of x; otherwise, returns 0. We will use the notation $(x)_i := \mathrm{pr}(x,i)$.

Since the function that defines pr() is polynomial-time computable, pr() is a symbol of $L_{\mathrm{BA}}(\mathrm{PV})$, and since the relevant properties of the function that defined pr() are elementary (it returns the same outputs on the same indices given the same string, it returns different outputs on some index given different strings of the same length, etc.), we can assume that the defining Δ_1^{b} L_{BA}-formula $\varphi(\overline{x}, y)$ is such that T_2 proves the desired properties of pr(). Continuing with the example, if the defining formula for $\mathrm{pr}(x,i) = b$ is $\varphi(x,i,b)$, then T_2 proves $\varphi(x,i,b) \wedge \varphi(x,i,b') \rightarrow b = b'$.

Definition 36. We define the following notation, similar to bounded quantification:

$$\exists n \in \mathrm{Log}_{>1}\ \varphi \stackrel{def}{=}\ \exists N(|N| > 1 \wedge \varphi(n/|N|)),$$

$$\forall n \in \mathrm{Log}_{>1}\ \varphi \stackrel{def}{=}\ \forall N(|N| > 1 \rightarrow \varphi(n/|N|)).$$

Log is intended to represent the set of small numbers; that is, the set of numbers n such that $n = |N|$ for some $N > n$. And $\mathrm{Log}_{>1}$ is the set of small numbers larger than one.

Proof System

Definition 37. LKB+ is the traditional Gentzen proof system extended by rules for bounded quantification and for full bounded induction. Formally, LKB+ consists on the following rules:

$$\text{Weakening} \qquad \textbf{left}\ \frac{\Gamma \longrightarrow \Delta}{A, \Gamma \longrightarrow \Delta} \qquad \textbf{right}\ \frac{\Gamma \longrightarrow \Delta}{\Gamma \longrightarrow \Delta, A}$$

$$\text{Exchange} \qquad \textbf{left}\ \frac{\Gamma_1, A, B, \Gamma_2 \longrightarrow \Delta}{\Gamma_1, B, A, \Gamma_2 \longrightarrow \Delta} \qquad \textbf{right}\ \frac{\Gamma \longrightarrow \Delta_1, A, B, \Delta_2}{\Gamma \longrightarrow \Delta_1, B, A, \Delta_2}$$

$$\text{Contraction} \qquad \textbf{left}\ \frac{\Gamma_1, A, A, \Gamma_2 \longrightarrow \Delta}{\Gamma_1, A, \Gamma_2 \longrightarrow \Delta} \qquad \textbf{right}\ \frac{\Gamma \longrightarrow \Delta_1, A, A, \Delta_2}{\Gamma \longrightarrow \Delta_1, A, \Delta_2}$$

$$\neg : \text{introduction} \qquad \textbf{left}\ \frac{\Gamma \longrightarrow \Delta, A}{\neg A, \Gamma \longrightarrow \Delta} \qquad \textbf{right}\ \frac{A, \Gamma \longrightarrow \Delta}{\Gamma \longrightarrow \Delta, \neg A}$$

$\wedge$: introduction $\quad \mathbf{left}_1 \dfrac{A, \Gamma \longrightarrow \Delta}{A \wedge B, \Gamma \longrightarrow \Delta} \quad \mathbf{left}_2 \dfrac{A, \Gamma \longrightarrow \Delta}{B \wedge A, \Gamma \longrightarrow \Delta}$

$$\mathbf{right} \frac{\Gamma \longrightarrow \Delta, A \qquad \Gamma \longrightarrow \Delta, B}{\Gamma \longrightarrow \Delta, A \wedge B}$$

$\vee$: introduction $\quad \mathbf{left} \dfrac{A, \Gamma \longrightarrow \Delta \qquad B, \Gamma \longrightarrow \Delta}{A \vee B, \Gamma \longrightarrow \Delta}$

$$\mathbf{right}_1 \frac{\Gamma \longrightarrow \Delta, A}{\Gamma \longrightarrow \Delta, A \vee B} \qquad \mathbf{right}_2 \frac{\Gamma \longrightarrow \Delta, A}{\Gamma \longrightarrow \Delta, B \vee A}$$

$$\text{Cut} \quad \frac{\Gamma \longrightarrow \Delta, A \qquad A, \Gamma \longrightarrow \Delta}{\Gamma \longrightarrow \Delta}$$

$\forall$: introduction $\quad \mathbf{left} \dfrac{A(t), \Gamma \longrightarrow \Delta}{\forall x A(x), \Gamma \longrightarrow \Delta} \quad \mathbf{right} \dfrac{\Gamma \longrightarrow \Delta, A(a)}{\Gamma \longrightarrow \Delta, \forall x A(x)}$

$\exists$: introduction $\quad \mathbf{left} \dfrac{A(a), \Gamma \longrightarrow \Delta}{\exists x A(x), \Gamma \longrightarrow \Delta} \quad \mathbf{right} \dfrac{\Gamma \longrightarrow \Delta, A(t)}{\Gamma \longrightarrow \Delta, \exists x A(x)}$

$\forall \leq$: introduction $\quad \mathbf{left} \dfrac{A(t), \Gamma \longrightarrow \Delta}{t \leq s, (\forall x \leq s) A(x), \Gamma \longrightarrow \Delta}$

$$\mathbf{right} \frac{a \leq t, \Gamma \longrightarrow \Delta, A(a)}{\Gamma \longrightarrow \Delta, (\forall x \leq t) A(x)}$$

$\exists \leq$: introduction $\quad \mathbf{left} \dfrac{a \leq t, A(a), \Gamma \longrightarrow \Delta}{(\exists x \leq t) A(x), \Gamma \longrightarrow \Delta}$

$$\mathbf{right} \frac{\Gamma \longrightarrow \Delta, A(t)}{t \leq s, \Gamma \longrightarrow \Delta, (\exists x \leq s) A(x)}$$

where A and B are L_{BA}-formulas, t and a are L_{BA}-terms and $\Gamma \longrightarrow \Delta$ are sequents, so Γ and Δ are sets of L_{BA}-formulas. LKB+ also implements the following rule (hence the +), the IND-rule:

$$\frac{\Gamma, A(a) \longrightarrow A(a+1), \Delta}{\Gamma, A(0) \longrightarrow A(t), \Delta}$$

where a and t are L_{BA}-terms and A is a bounded formula.

Then, the following lemma follows from Lemma 7.1.3 of [Kra95].

Lemma 2. *Let* BASIC_{seq} *be* BASIC *in the form of sequents: that is, let* $\longrightarrow \varphi \in \mathrm{BASIC}_{seq}$ *iff* $\varphi \in$ BASIC*, for* φ *a* L_{BA}*-formula and* $\longrightarrow \varphi$ *a sequent. Then, for any* L_{BA}*-formula* φ*, we have* $T_2 \vdash \varphi$ *iff the sequent* $\longrightarrow \varphi$ *is derivable in LKB+ from* BASIC_{seq}*.*

What this means is that when we reason in T_2, we can use the classic derivation rules, as well as full bounded induction. In Chapters 5 and 6 we will use this lemma implicitly every time we say that we reason in T_2.

Strings and Numbers

We define the following notation so we can, in a way, talk about strings in our bounded arithmetic formalization, which will make it easier to adapt the intended proofs.

Definition 38.

$$\exists x \in \{0,1\}^{|t|}\varphi \stackrel{def}{=} \exists X < 2^{|t|}\ \varphi(x/(2^{|t|}+X)),$$
$$\forall x \in \{0,1\}^{|t|}\varphi \stackrel{def}{=} \forall X < 2^{|t|}\ \varphi(x/(2^{|t|}+X)).$$

where t is an L_{BA}-term.

Note that for every $n > 0$, there is a bijection between $\{0,1\}^n$ and $\{2^n, 2^n+1, ..., 2^{n+1}-1\}$: namely, $x \mapsto 2^n + \mathrm{bin}(x)$, where $\mathrm{bin}(x)$ returns the binary number represented by the string x, ignoring left zeros. The inverse of this bijection would be $y \mapsto \mathrm{str}(y - 2^n, n)$, where $\mathrm{str}(m,n)$ returns the binary string of length $n \in \mathrm{Log}_{>1}$ that represents the number m in binary (assuming $m < 2^n$). Thus, Definition 38 is well defined. In general, when in a L_{BA}-formula we use x as a string of length n, it is notation for the number $2^n + \mathrm{bin}(x)$.

In Chapters 5 and 6 we will often define formulas named "$x \in L$", where L is destined to represent a particular language, like SAT. We understand those formulas are only applied to $x \in \{0,1\}^n$ for a particular $n \in \mathrm{Log}_{>1}$, so that in the formula x is substituted by $2^n + X$ for a suitable X.

2.3.2 Two-sorted Theories

The goal of two-sorted bounded arithmetic is to allow us to talk about sets of numbers without abandoning first-order logic. For this purpose, structures for two-sorted theories will

have a universe consisting of two sets, and the elements of the second set will represent sets of the elements of the first one: we will denote this kind of elements with capital letters $(X, Y, Z \ldots)$.

Definition 39. $L_{\mathrm{BA}}(\alpha)$, the language of two-sorted bounded arithmetic, is defined as follows:

$$L_{\mathrm{BA}}(\alpha) \stackrel{def}{=} L_{\mathrm{BA}} \cup \{\in\},$$

where $\in$ is a binary relation symbol whose standard meaning is actual element-hood.

In this subsection, let φ be an $L_{\mathrm{BA}}(\alpha)$-formula and t be an $L_{\mathrm{BA}}(\alpha)$-term without set-sort variables.

Definition 40. $L_{\mathrm{BA}}(\alpha)$-structures for two-sorted theories have a universe of the form $(M, \mathcal{X})$: M is called the number-sort and $\mathcal{X}$ is called the set-sort; $\in$ is interpreted by a subset of $M \times \mathcal{X}$, and the symbols in L_{BA} are interpreted only in the number-sort. As L_{BA}, we have that $L_{\mathrm{BA}}(\alpha)$ is a first-order-logic language, so we have built-in equality, but we have to distinguish between equality and quantification over number-sort elements and equality and quantification over set-sort elements: these functions work in the standard way in both sorts, but they are considered different functions depending on the sort.

The standard structure of two-sorted bounded arithmetic has $(\mathbb{N}, [\mathbb{N}]^{<\omega})$ as its universe (where $[\mathbb{N}]^{<\omega}$ is the set of all finite subsets of $\mathbb{N}$), $\in$ interpreted as actual element-hood, and the rest of symbols interpreted in the usual way.

Definition 41. Let $X \leq x := \forall y(y \in X \to y \leq x)$. Then, let the set BASIC(α) be defined as BASIC plus the universal closures of the two following formulas:

$\exists x(X \leq x)$ *(Set-boundedness)*

$X \leq x \wedge Y \leq x \wedge \forall y \leq x\ (y \in X \leftrightarrow y \in Y) \to X = Y$ *(Extensionality).*

Definition 42. Given a set Φ, let the scheme of bounded Φ-comprehension (Φ-COMP) be the set of the universal closures of formulas of the following form:

$$\exists Y(Y \leq z \wedge \forall y \leq z\ (y \in Y \leftrightarrow \varphi(\overline{X}, \overline{x}, y)),$$

where $\varphi \in \Phi$ and Y is not in $\overline{X}$.

Definition 43. The sets $\Sigma^{\mathrm{b}}_i(\alpha)$, $\Pi^{\mathrm{b}}_i(\alpha)$ and $\Sigma^{\mathrm{b}}_\infty(\alpha)$ are defined as Σ^{b}_i, Π^{b}_i and $\Sigma^{\mathrm{b}}_\infty$, but in $L_{\mathrm{BA}}(\alpha)$ and considering only number-sort equality and quantifiers. More precisely,

$\Sigma^{\mathrm{b}}_0(\alpha) = \Pi^{\mathrm{b}}_0(\alpha)$ is the set of all $L_{\mathrm{BA}}(\alpha)$-formulas without set-sort equality or set-sort quantifiers in which all number-sort quantifiers are sharply bounded.

$\Sigma^{\mathrm{b}}_{i+1}(\alpha)$ is the smallest set of formulas such that:

1. $\Sigma^{\mathrm{b}}_i(\alpha) \cup \Pi^{\mathrm{b}}_i(\alpha) \subseteq \Sigma^{\mathrm{b}}_{i+1}(\alpha)$
2. $\varphi, \psi \in \Sigma^{\mathrm{b}}_{i+1}(\alpha) \implies \varphi \wedge \psi \in \Sigma^{\mathrm{b}}_{i+1}(\alpha)$
3. $\varphi \in \Pi^{\mathrm{b}}_{i+1}(\alpha) \implies \neg\varphi \in \Sigma^{\mathrm{b}}_{i+1}(\alpha)$
4. $\varphi \in \Sigma^{\mathrm{b}}_{i+1}(\alpha) \implies \exists x \leq |t|\ \varphi, \forall x \leq |t|\ \varphi \in \Sigma^{\mathrm{b}}_{i+1}(\alpha)$
5. $\varphi \in \Sigma^{\mathrm{b}}_{i+1}(\alpha) \implies \exists x \leq t\ \varphi \in \Sigma^{\mathrm{b}}_{i+1}(\alpha)$

$\Pi^{\mathrm{b}}_{i+1}(\alpha)$ is the smallest set of formulas such that:

1. $\Sigma^{\mathrm{b}}_i(\alpha) \cup \Pi^{\mathrm{b}}_i(\alpha) \subseteq \Pi^{\mathrm{b}}_{i+1}(\alpha)$
2. $\varphi, \psi \in \Pi^{\mathrm{b}}_{i+1}(\alpha) \implies \varphi \wedge \psi \in \Pi^{\mathrm{b}}_{i+1}(\alpha)$
3. $\varphi \in \Sigma^{\mathrm{b}}_{i+1}(\alpha) \implies \neg\varphi \in \Pi^{\mathrm{b}}_{i+1}(\alpha)$
4. $\varphi \in \Pi^{\mathrm{b}}_{i+1}(\alpha) \implies \exists x \leq |t|\ \varphi, \forall x \leq |t|\ \varphi \in \Pi^{\mathrm{b}}_{i+1}(\alpha)$
5. $\varphi \in \Pi^{\mathrm{b}}_{i+1}(\alpha) \implies \forall x \leq t\ \varphi \in \Pi^{\mathrm{b}}_{i+1}(\alpha)$

$$\Sigma^{\mathrm{b}}_\infty(\alpha) = \Pi^{\mathrm{b}}_\infty(\alpha) \stackrel{def}{=} \bigcup_{i\in\omega} \Sigma^{\mathrm{b}}_i(\alpha) = \bigcup_{i\in\omega} \Pi^{\mathrm{b}}_i(\alpha)$$

Definition 44. Let T be the theory axiomatized by BASIC + $\Sigma^{\mathrm{b}}_1(\alpha)$-LIND, and let $\Delta^{\mathrm{b}}_1(\alpha)$ be the set of formulas φ such that $T \vdash \varphi \leftrightarrow \psi$ and $T \vdash \varphi \leftrightarrow \chi$ for some $\psi \in \Sigma^{\mathrm{b}}_1(\alpha)$ and $\chi \in \Pi^{\mathrm{b}}_1(\alpha)$. Note that since $L_{\mathrm{BA}} \subseteq L_{\mathrm{BA}}(\alpha)$, BASIC is a set of $L_{\mathrm{BA}}(\alpha)$-formulas.

Definition 45. For each $i \geq 0$, we have:

- The theory $T^i_2(\alpha)$ is axiomatized by BASIC(α) + $\Sigma^{\mathrm{b}}_i(\alpha)$-IND + $\Delta^{\mathrm{b}}_1(\alpha)$-COMP.
- The theory $S^i_2(\alpha)$ is axiomatized by BASIC(α) + $\Sigma^{\mathrm{b}}_i(\alpha)$-LIND + $\Delta^{\mathrm{b}}_1(\alpha)$-COMP.

And we define:

$$T_2(\alpha) \stackrel{def}{=} \bigcup_{i\in\omega} T^i_2(\alpha).$$

Definition 46. The sets of formulas $\Sigma^{1,\mathrm{b}}_i$ and $\Pi^{1,\mathrm{b}}_i$ are defined by counting alternations of set-sort quantifiers and ignoring bounded quantifiers (again, excluding set-sort equality). More precisely:

$\Sigma_0^{1,\mathrm{b}} = \Pi_0^{1,\mathrm{b}} \stackrel{def}{=} \Sigma_\infty^{\mathrm{b}}(\alpha)$

$\Sigma_{i+1}^{1,\mathrm{b}}$ is the smallest set of formulas such that:

1. $\Sigma_i^{1,\mathrm{b}} \cup \Pi_i^{1,\mathrm{b}} \subseteq \Sigma_{i+1}^{1,\mathrm{b}}$
2. $\varphi, \psi \in \Sigma_{i+1}^{1,\mathrm{b}} \implies \varphi \wedge \psi \in \Sigma_{i+1}^{1,\mathrm{b}}$
3. $\varphi \in \Pi_{i+1}^{1,\mathrm{b}} \implies \neg\varphi \in \Sigma_{i+1}^{1,\mathrm{b}}$
4. $\varphi \in \Sigma_{i+1}^{1,\mathrm{b}} \implies \exists x \leq t\ \varphi, \forall x \leq t\ \varphi \in \Sigma_{i+1}^{1,\mathrm{b}}$
5. $\varphi \in \Sigma_{i+1}^{1,\mathrm{b}} \implies \exists X \varphi \in \Sigma_{i+1}^{1,\mathrm{b}}$

$\Pi_{i+1}^{1,\mathrm{b}}$ is the smallest set of formulas such that:

1. $\Sigma_i^{1,\mathrm{b}} \cup \Pi_i^{1,\mathrm{b}} \subseteq \Pi_{i+1}^{1,\mathrm{b}}$
2. $\varphi, \psi \in \Pi_{i+1}^{1,\mathrm{b}} \implies \varphi \wedge \psi \in \Pi_{i+1}^{1,\mathrm{b}}$
3. $\varphi \in \Sigma_{i+1}^{1,\mathrm{b}} \implies \neg\varphi \in \Pi_{i+1}^{1,\mathrm{b}}$
4. $\varphi \in \Pi_{i+1}^{1,\mathrm{b}} \implies \exists x \leq t\ \varphi, \forall x \leq t\ \varphi \in \Pi_{i+1}^{1,\mathrm{b}}$
5. $\varphi \in \Pi_{i+1}^{1,\mathrm{b}} \implies \forall X \varphi \in \Pi_{i+1}^{1,\mathrm{b}}$

$\Sigma_\infty^{1,\mathrm{b}} = \Pi_\infty^{1,\mathrm{b}} \stackrel{def}{=} \bigcup_{i \in \omega} \Sigma_i^{1,\mathrm{b}} = \bigcup_{i \in \omega} \Pi_i^{1,\mathrm{b}}$

Definition 47. For every $i \geq 0$, the theory V_2^i is axiomatized by BASIC(α) + $\Sigma_i^{1,\mathrm{b}}$-COMP.

A Comparison

V_2^i was first introduced by Buss in [Bus86, p.167], where it receives the name $\widetilde{V}_2^i$. It is axiomatized over $L_{\mathrm{BA}}(\alpha)$ by BASIC(α) + $\Sigma_0^{1,\mathrm{b}}$-COMP + $\Sigma_i^{1,\mathrm{b}}$-IND, so it differs from our definition roughly by replacing $\Sigma_i^{1,\mathrm{b}}$-COMP by $\Sigma_i^{1,\mathrm{b}}$-IND. Confusingly enough, Buss also defines a theory named V_2^i, which is defined as $\widetilde{V}_2^i$ plus an axiom scheme bounding all functions by terms, but this difference is not meaningful because his V_2^i is shown to be a conservative extension of $\widetilde{V}_2^i$. Ten years later, in [Kra95, p.87], Krajíček "slightly abuses Buss' notation"(p.84) and renames Buss' $\widetilde{V}_2^i$ as V_2^i, notation that has become standard.

In [CN10, p.95], Cook and Nguyen define the theory V^i, which is defined as V_2^i but without the symbols # and $\lfloor /2 \rfloor$ in the language (nor the relevant axioms) and $|\ |$ applied to set-sort variables instead of number-sort ones (it represents the least upper bound of a set). Note

that V^i is axiomatized by $\Sigma_i^{1,b}$-COMP, like in our case, and not by $\Sigma_i^{1,b}$-IND, like in the case of Buss and Krajíček. More recently, in [Kra19, pp.193-194], Krajíček distinguishes between V_2^i, which he defines as in [Kra95], and V_1^i, defined as V_2^i but excluding # from the language (and the relevant axioms)[2]. Note the language of V_1^i still includes $\lfloor /2 \rfloor$ and the relevant axioms.

But it turns out a theory with Φ-COMP can prove Φ-IND [CN10, p.98], so our definition of V_2^i is at least as strong than all the preceding ones.

The definition of BASIC can be found on [Kra95], and the definitions of Σ_i^b, Π_i^b and the related sets are based on the ones given in [Kra19]. The rest of definitions can be found in [ABM23], perhaps less detailed.

2.3.3 The Pigeonhole Principle and Minimization

The Pigeonhole Principle is a well-known principle with many links to bounded arithmetic. Here, we define the following version of the Weak Pigeonhole Principle, which will be essential to the bounded proof of Kannan's result in Chapter 6:

Definition 48. The Weak Pigeonhole Principle (WPHP) is the following formula:

$$\begin{aligned} \mathrm{WPHP}(M, N, \varphi) \stackrel{def}{=} \quad & M \geq N^2 \wedge \forall x < M \; \exists y < N \; \varphi(x, y) \rightarrow \\ & \exists x < M \; \exists x' < M \; \exists y < N \; (x \neq x' \wedge \varphi(x, y) \wedge \varphi(x', y)), \end{aligned}$$

where M and N are variables and $\varphi(x, y)$ is an L_{BA}-formula that may have additional parameters z.

The following version of the Minimization Principle will also be of importance in Chapter 6:

Definition 49. The Minimization Principle (MIN) is the following formula:

$$\mathrm{MIN}(M, \varphi) \stackrel{def}{=} \exists x < M \; \varphi(x) \rightarrow \exists x < M \; \forall z < x \; (\varphi(x) \wedge \neg\varphi(z)),$$

where M is a variable and $\varphi(x)$ is an L_{BA}-formula that may have additional parameters z.

The following result follows from Theorem 14 of [MPW02]:

Lemma 3. *For every bounded formula φ, we have:*

$$T_2 \vdash \mathrm{WPHP}(M, N, \varphi).$$

[2]Actually, in [Kra19], Krajíček only defines V_1^i and V_2^i for $i \leq 1$, but it's easy to infer the general case.

And the following result follows from Lemma 5.2.7 of [Kra95]:

Lemma 4. *For every bounded formula φ, we have:*

$$T_2 \vdash \mathrm{MIN}(M, \varphi).$$

Chapter 3

The Karp-Lipton Theorem

In this chapter, we will prove The Karp-Lipton Theorem, originally shown by R. M. Karp and R. J. Lipton in [KL80]. We will follow the exposition of this theorem presented in [AB09].

3.1 Auxiliary Lemmas

First, we present two Lemmas that will make the proof of the Theorem much easier. The second one (Lemma 6) will actually be used directly in Section 4.3, which does not need to use the whole Karp-Lipton Theorem.

Lemma 5 (Hierarchy Collapse)**.** *For every $i > 0$ and every $j \geq i$, if $\Sigma_i^{\mathrm{P}} = \Pi_i^{\mathrm{P}}$, then $\Sigma_j^{\mathrm{P}} \cup \Pi_j^{\mathrm{P}} \subseteq \Sigma_i^{\mathrm{P}} \cap \Pi_i^{\mathrm{P}}$.*

Proof. Let $i > 0$, and assume $\Sigma_i^{\mathrm{P}} = \Pi_i^{\mathrm{P}}$. We proceed by induction on $j \geq i$:

Base case: if $j = i$, since $\Sigma_i^{\mathrm{P}} = \Pi_i^{\mathrm{P}}$, we clearly have $\Sigma_j^{\mathrm{P}} \cup \Pi_j^{\mathrm{P}} \subseteq \Sigma_i^{\mathrm{P}} \cap \Pi_i^{\mathrm{P}}$.

Inductive case: suppose $\Sigma_j^{\mathrm{P}} \cup \Pi_j^{\mathrm{P}} \subseteq \Sigma_i^{\mathrm{P}} \cap \Pi_i^{\mathrm{P}}$. We want to prove $\Sigma_{j+1}^{\mathrm{P}} \cup \Pi_{j+1}^{\mathrm{P}} \subseteq \Sigma_i^{\mathrm{P}} \cap \Pi_i^{\mathrm{P}}$; let us start showing $\Sigma_{j+1}^{\mathrm{P}} \subseteq \Sigma_i^{\mathrm{P}} \cap \Pi_i^{\mathrm{P}}$. We know $L \in \Sigma_{j+1}^{\mathrm{P}}$ if and only if

$$x \in L \iff \exists v_1 \in \{0,1\}^{q(|x|)} \forall v_2 \in \{0,1\}^{q(|x|)} \ldots Q_{j+1} v_{j+1} \in \{0,1\}^{q(|x|)} M(x, v_1, \ldots, v_{j+1}) = 1$$

where Q_n is $\forall$ if n is even and $\exists$ if n is odd, q is a polynomial and M a polytime TM. Now, we define the language L' as follows:

$$\langle x, v_1 \rangle \in L' \iff \forall v_2 \in \{0,1\}^{q(|x|)} \ldots Q_{j+1} v_{j+1} \in \{0,1\}^{q(|x|)} M(x, v_1, \ldots, v_{j+1}) = 1$$

And we have $L' \in \Pi_j^{\mathrm{P}}$, so that $L' \in \Sigma_j^{\mathrm{P}}$ per our assumption. Then, there is a polynomial r and a polytime TM M' such that

$$\langle x, v_1 \rangle \in L' \iff \exists u_1 \in \{0,1\}^{r(|x|)} \dots Q_j u_j \in \{0,1\}^{r(|x|)} M'(x, v_1, u_1, \dots, u_j) = 1$$

But then,

$$\begin{aligned} x \in L &\iff \exists v_1 \in \{0,1\}^{q(|x|)} \langle x, v_1 \rangle \in L' \\ &\iff \exists v_1 \in \{0,1\}^{q(|x|)} \exists u_1 \in \{0,1\}^{r(|x|)} \dots Q_j u_j \in \{0,1\}^{r(|x|)} M'(x, v_1, u_1, \dots, u_j) = 1 \\ &\iff \exists u_1 \in \{0,1\}^{s(|x|)} \dots Q_j u_j \in \{0,1\}^{s(|x|)} M''(x, u_1, \dots, u_j) = 1 \end{aligned}$$

where M'' works as M' but interpreting u_1 so that now it also encodes v_1, and $s(n)$ is a polynomial large enough to allow the codification of the pair $\langle u_1, v_1 \rangle$ (for example, $s(n) = 2(p(n) + q(n) + 1)$). This means $L \in \Sigma_j^{\mathrm{P}}$, so that $L \in \Sigma_i^{\mathrm{P}}$ by the inductive hypothesis. Since L was arbitrary, we get $\Sigma_{j+1}^{\mathrm{P}} \subseteq \Sigma_i^P = \Sigma_i^{\mathrm{P}} \cap \Pi_i^{\mathrm{P}}$, as we wanted. A symmetrical argument (with $\Pi, \forall$ in place of $\Sigma, \exists$) can be used to show $\Pi_{j+1}^{\mathrm{P}} \subseteq \Pi_i^{\mathrm{P}} \cap \Sigma_i^{\mathrm{P}}$. □

Lemma 6 (Self-reducibility of SAT)**.** *For all $c > 0$ there exists $c' > 0$ such that if* $\mathrm{SAT} \in \mathrm{SIZE}(n^c)$*, then there is a $\mathcal{O}(n^{c'})$-size circuit family $\{C'_n\}_{n \in \mathbb{N}}$ that outputs a satisfying assignment on input a satisfiable formula.*

Proof. Fix $c > 0$, and suppose $\mathrm{SAT} \in \mathrm{SIZE}(n^c)$. Let $\{C_n\}_{n \in \mathbb{N}}$ be a $\mathcal{O}(n^c)$-size circuit family deciding SAT. Given φ representing a quantified formula, we have $C_{|\varphi|}(\varphi) = 1$ iff (the formula represented by) φ is satisfiable. We want $c' > 0$ such that given C_n we have a circuit C'_n of size at most $n^{c'}$ and on input $\varphi(x_1, \dots, x_m)$ (where $|\varphi| = n$ and $x_1, \dots, x_m$ are single variables) we have $C'_n(\varphi) = y_1 \dots y_m$, where if φ is satisfiable, $y_1 \dots y_n$ is a satisfying assignment, and if it is not, $y_i = 0$ for $1 \leq i \leq m$.

To achieve this, given C_n, define the following algorithm: on input $\varphi(x_1, \dots, x_m)$,

Set $\varphi_0 = \varphi$
For $i = 1, \dots, m$:
    Set φ_i to be as φ_{i-1} but with $x_i = 1$
    If $C_n(\varphi_i) = 1$, set $y_i = 1$
    Else, set φ_i to be as φ_{i-1} but with $x_i = 0$ and set $y_i = 0$
Output $y_1 \dots y_m$

This algorithm loops $m \leq n$ times, and in the i-th loop it computes $C_n(\varphi_i)$, updates φ_{i+1} and sets y_i to 1 or to 0, all of which are computable by polynomial-sized circuits. Then, there must be a $c' > 0$ big enough so that the algorithm is computable by a $\mathcal{O}(n^{c'})$-size circuit. Let C'_n be such a circuit. □

3.2 The Theorem

Theorem 4 (Karp-Lipton)**.** *For every $c > 0$, if* $\text{SAT} \in \text{SIZE}(n^c)$ *then* $\text{PH} \subseteq \Sigma_2^{\text{P}} \cap \Pi_2^{\text{P}}$.

Proof. Fix $c > 0$, and assume $\text{SAT} \in \text{SIZE}(n^c)$. By Lemma 5, we only need to show $\Sigma_2^{\text{P}} = \Pi_2^{\text{P}}$. We start by showing $\Pi_2^{\text{P}} \subseteq \Sigma_2^{\text{P}}$, and we do so by proving the Π_2^{P}-complete language Π_2-TQBF is in Σ_2^{P}. The language Π_2-TQBF is the set of all (strings representing) true quantified formulas of the following form:

$$\forall \overline{x} \exists \overline{y} \varphi(\overline{x}, \overline{y}),$$

where $\varphi(x_1, \ldots, x_{2n})$ is a QF formula and $x_1, \ldots, x_{2n}$ are (single) propositional variables. By $\text{SAT} \in \text{SIZE}(n^c)$, there is an $\mathcal{O}(n^c)$-size circuit family $\{C_n\}_{n \in \mathbb{N}}$ deciding SAT: then, by Lemma 6, there exists $c' > 0$ such that there is an $\mathcal{O}(n^{c'})$-size circuit family $\{C'_n\}_{n \in \mathbb{N}}$ so that for any formula $\varphi(x_1, \ldots, x_n)$, we have $C'_{|\varphi|}(\varphi) = y_1 \ldots y_n$, where if φ is satisfiable, then the string $y_1 \ldots y_n$ is an assignment satisfying φ. Now, consider the set A of all Π_2-formulas $\forall \overline{x} \exists \overline{y} \varphi(\overline{x}, \overline{y})$ such that the following statement is true:

$$\exists C' \in \{0,1\}^{8n^{c'} \log(n^{c'})} \forall u \in \{0,1\}^{|\overline{x}|} \varphi(u, C'(\varphi(u, \overline{y}))).$$

Due to the existence of $\{C'_n\}_{n \in \mathbb{N}}$, we have $\psi \in A$ iff $\psi \in \Pi_2$-TQBF, so that $A = \Pi_2$-TQBF. But $A \in \Sigma_2^{\text{P}}$, since the evaluation of circuits of the family $\{C'_n\}_{n \in \mathbb{N}}$ is doable in polynomial time, so we conclude $\Pi_2\text{-TQBF} \in \Sigma_2^{\text{P}}$, as we wanted.

$\Sigma_2^{\text{P}} \subseteq \Pi_2^{\text{P}}$ is proven in a similar way. Σ_2-TQBF is the set of all true quantified formulas of the following form:

$$\exists \overline{x} \forall \overline{y} \varphi(\overline{x}, \overline{y}),$$

for φ a QF formula. Given a Σ_2-formula $\exists \overline{x} \forall \overline{y} \varphi(\overline{x}, \overline{y})$, by propositional equivalences we know it is true iff the Π_2-formula $\forall \overline{x} \exists \overline{y} \neg \varphi(\overline{x}, \overline{y})$ is false, and by the previous argument we know $\forall \overline{x} \exists \overline{y} \neg \varphi(\overline{x}, \overline{y})$ is false iff the following statement is false:

$$\exists C' \in \{0,1\}^{8n^{c'} \log(n^{c'})} \forall u \in \{0,1\}^{|\overline{x}|} \neg \varphi(u, C'(\neg \varphi(u, \overline{y}))).$$

Then, consider the set B of all Σ_2-formulas $\exists\overline{x}\forall\overline{y}\varphi(\overline{x},\overline{y})$ such that following statement is true:

$$\forall C' \in \{0,1\}^{8n^{c'}\log(n^{c'})}\exists u \in \{0,1\}^{|\overline{x}|}\varphi(u, C'(\varphi(u,\overline{y}))).$$

By the preceding argument and some logical equivalences, we get $B = \Sigma_2$-TQBF. Since $B \in \Pi_2^{\mathrm{P}}$, because the evaluation of circuits of the family $\{C'_n\}_{n\in\mathbb{N}}$ is doable in polynomial time, we get Σ_2-TQBF $\in \Pi_2^{\mathrm{P}}$, as we wanted. □

Chapter 4

Kannan's Result

Recall (1.1) and (1.2) from Chapter 1:

$$K^c \notin \mathrm{SIZE}(n^c). \tag{1.1}$$

$$\Sigma_2^{\mathrm{E}} \not\subseteq \mathrm{P/poly}. \tag{1.2}$$

The purpose of this chapter is to present a detailed proof of Kannan's result (1.1) and its consequence (1.2). Although it is not our main goal, we will begin by presenting a proof of (1.2), since it introduces many concepts that are essential for the proof of (1.1); then, we will adapt this proof, proving both (1.1) and (1.2). We will loosely follow the original proof by Kannan.

We will prove the following statement: for every $c > 0$, we have $L^* \notin \mathrm{SIZE}(n^c)$, where L^* is the following standard $\mathrm{NE}^{\mathrm{NP}}$-complete language:

$$L^* \stackrel{def}{=} \{\langle M, x, 1^t\rangle : \exists y \in \{0,1\}^{2^t} \forall z \in \{0,1\}^{2^t} (M(x,y,z) = 1)\}. \tag{4.1}$$

Of course, this implies $L^* \notin \mathrm{P/poly}$, which entails $\mathrm{NE}^{\mathrm{NP}} \not\subseteq \mathrm{P/poly}$, which is in turn equivalent to $\Sigma_2^{\mathrm{E}} \not\subseteq \mathrm{P/poly}$ by Theorem 2.

To prove this, we will follow the next strategy: first, we will define a language $L(S^*) \in \mathrm{NE}^{\mathrm{NP}^{\mathrm{NP}}}$ that doesn't have small circuits (by virtue of a simple counting argument), and then we will distinguish two cases, depending on whether NP is included on P/poly or not; if it isn't, the claim easily follows, and if it is, we will use the Karp-Lipton Theorem and padding to show $L(S^*) \in \mathrm{NE}^{\mathrm{NP}}$, from which our claim will follow.

4.1 Counting argument

Consider a n-input function $f : \{0,1\}^n \to \{0,1\}$: the *truth-table* of f is a binary string S of length 2^n, where the bit on the m-th position of S (m being the number represented by x in binary), denoted by $S(x)$, is $f(x)$. There are 2^{2^n} possible truth-tables of n-input functions. A circuit of size at most n can be represented by a binary string of length at most $8n \log n$,[1] so there are at most $2^{8n \log n}$ circuits of size at most n.

Now, consider circuits of size at most $2^n/(9(n+1))$: there are at most $2^{8(2^n/(9(n+1)))\log(2^n/(9(n+1)))}$ of them. But

$$8\frac{2^n}{9(n+1)} \log\left(\frac{2^n}{9(n+1)}\right) \leq \frac{8 \cdot 2^n \log(2^n)}{9(n+1)} = \frac{8 \cdot 2^n(n+1)}{9(n+1)} = \frac{8}{9}2^n < 2^n$$

so that $2^{8(2^n/(9(n+1)))\log(2^n/(9(n+1)))} < 2^{2^n}$, which means there must be a n-input function which is not computed by any circuit of size at most $2^n/(9(n+1))$. This means we can talk about the lexicographically minimal truth-table of a n-input function not computed by any circuit of size at most $2^n/(9(n+1))$: call it S^*_n. If we call $L(S^*)$ the characteristic language of $\bigcup_{n\in\mathbb{N}} S^*_n$ (i.e., for an input x of length n, we have $x \in L(S^*) \Leftrightarrow S^*_n(x) = 1$), then $L(S^*) \notin \bigcup_{n\in\mathbb{N}} \mathrm{SIZE}(2^n/(9(n+1)))$; in particular, $L(S^*) \notin \mathrm{P/poly}$.

Now, we want to see $L(S^*) \in \mathrm{NE}^{\mathrm{NP}^{\mathrm{NP}}}$. First of all, define

$$\begin{aligned}
\mathrm{Computes}_n(C,S) &\stackrel{def}{=} \forall y \in \{0,1\}^n (C(y) = S(y)) \\
\mathrm{Smallereq}_n(S,S') &\stackrel{def}{=} \forall x \in \{0,1\}^n \exists z \in \{0,1\}^n ((S(x) \leq S'(x)) \vee \\
&\qquad (\mathrm{bin}(x) < \mathrm{bin}(z) \wedge S(z) < S'(z))).
\end{aligned}$$

where C is (a binary string representing) a n-input circuit, while S and S' are truth tables of length 2^n; recall $\mathrm{bin}(x)$ outputs the number represented by a string x in binary (ignoring leftmost zeros). Note since the output of a circuit or a truth table on a certain input can be computed in time polynomial on the length of said circuit or truth table and there are exactly 2^n inputs of length n, both $\mathrm{Computes}_n$ and $\mathrm{Smallereq}_n$ can be solved by brute force in time polynomial on inputs of length 2^n. We also define $\mathrm{Size}(C)$, a function computing the size of (the circuit represented by) C, which is always polynomial on the length of C.

Given n, let L'_n and L''_n be defined as follows:

$$L'_n \stackrel{def}{=} \{S \in \{0,1\}^{2^n} : \forall C \in \{0,1\}^{\frac{8}{9}2^n} (\mathrm{Size}(C) \leq \frac{2^n}{9(n+1)} \to \neg \mathrm{Computes}_n(C,S))\}$$

[1] Where $\log n$ stands for $\lceil \log_2(n+1) \rceil$.

$$L''_n \stackrel{def}{=} \{S{\in}\{0,1\}^{2^n} : \forall S'{\in}\{0,1\}^{2^n}[\exists C'{\in}\{0,1\}^{\frac{8}{9}2^n}(\mathrm{Size}(C') \leq \frac{2^n}{9(n+1)} \wedge \mathrm{Computes}_n(C',S'))$$
$$\vee \mathrm{Smallereq}_n(S,S')]\}$$

For a 2^n-length truth table S, we have $S \in L'_n$ means S is not computed by any circuit of size at most $2^n/(9(n+1))$ and $S \in L''_n$ means S is the lexicographically smallest truth table fulfilling that property, so we can express $L(S^*)$ as follows:

$$L(S^*) = \{x : \exists S \in \{0,1\}^{2^{|x|}}(S \in L'_{|x|} \wedge S \in L''_{|x|} \wedge S(x) = 1)\}$$

Let $n = |x|$ in the following formulas. Unfolding the definitions of L'_n and L''_n we obtain:

$$L(S^*) = \{x : \exists S{\in}\{0,1\}^{2^n}[\forall C{\in}\{0,1\}^{\frac{8}{9}2^n}(\mathrm{Size}(C) \leq \frac{2^n}{9(n+1)} \rightarrow \neg\mathrm{Computes}_n(C,S))$$
$$\wedge \forall S'{\in}\{0,1\}^{2^n}(\exists C'{\in}\{0,1\}^{\frac{8}{9}2^n}(\mathrm{Size}(C') \leq \frac{2^n}{9(n+1)} \wedge \mathrm{Computes}_n(C',S')) \vee \mathrm{Smallereq}_n(S,S'))$$
$$\wedge S(x) = 1]\}$$
$$= \{x : \exists S{\in}\{0,1\}^{2^n}\forall C{\in}\{0,1\}^{\frac{8}{9}2^n}\forall S'{\in}\{0,1\}^{2^n}\exists C'{\in}\{0,1\}^{\frac{8}{9}2^n}[(\mathrm{Size}(C) \leq \frac{2^n}{9(n+1)} \rightarrow$$
$$\neg\mathrm{Computes}_n(C,S)) \wedge (\mathrm{Size}(C') \leq \frac{2^n}{9(n+1)} \wedge \mathrm{Computes}_n(C',S') \vee \mathrm{Smallereq}_n(S,S')) \wedge S(x) = 1]\}.$$

Note the two consecutive universal quantifiers $\forall C{\in}\{0,1\}^{\frac{8}{9}2^n}$ and $\forall S'{\in}\{0,1\}^{2^n}$ can be expressed as a single universal quantifier ranging over $\{0,1\}^{2^n+1}$ (the extra bit will convey which one of the two universal quantifiers it represents in each occurrence) by modifying the occurrences of C and S' in the matrix of the formula in a suitable way, so that $L(S^*)$ is definable by a formula with $\exists\forall\exists$ prenex and a polynomial-time computable matrix. Since in that formula all quantifiers range over strings of length bounded exponentially by the length of the input, we have $L(S^*) \in \Sigma_3^{\mathrm{E}} = \mathrm{NE}^{\mathrm{NP}^{\mathrm{NP}}}$, using Theorem 2.

4.2 Case distinction

Now, we prove for all c, we have $L^* \notin \mathrm{SIZE}(n^c)$ by case distinction: either $\mathrm{NP} \not\subseteq \mathrm{P/poly}$ or $\mathrm{NP} \subseteq \mathrm{P/poly}$.

4.2.1 $\mathrm{NP} \not\subseteq \mathrm{P/poly}$

This means, in particular, $\mathrm{SAT} \notin \mathrm{P/poly}$. Since $\mathrm{NP} \subseteq \mathrm{NE}^{\mathrm{NP}}$ and L^* is $\mathrm{NE}^{\mathrm{NP}}$-complete, $\mathrm{SAT} \leq_p L^*$. Let f be a polynomial-time function witnessing that reduction: by Lemma 1,

polynomial-time functions have small circuits, so let k be such that $\{C'_n\}_{n\in\mathbb{N}}$ is a $\mathcal{O}(n^k)$-size circuit family computing f. Then, suppose towards contradiction there is a c such that $L^* \in \text{SIZE}(n^c)$, and let $\{C_n\}_{n\in\mathbb{N}}$ be a $\mathcal{O}(n^c)$-size circuit family witnessing that. Now, define a $\mathcal{O}(n^{c+k})$-size circuit family $\{C''_n\}_{n\in\mathbb{N}}$, where C''_n is built composing C_n with C'_n: the output of C is fed into the input of C'. It is easy to see $\{C''_n\}_{n\in\mathbb{N}}$ solves SAT,[2] witnessing SAT $\in$ P/poly, a contradiction.

4.2.2 NP $\subseteq$ P/poly

In this case, we can use Karp-Lipton's Theorem 4, which implies PH $\subseteq \text{NP}^{\text{NP}}$ by Theorem 1. This allows us to prove $\text{NE}^{\text{NP}^{\text{NP}}} \subseteq \text{NE}^{\text{NP}}$ by padding: let $L \in \text{NE}^{\text{NP}^{\text{NP}}}$, and let t be such that M is a NDTM with oracle in NP^{NP} deciding L in time $\mathcal{O}(2^{n^t})$. We define

$$L_{\text{pad}} \stackrel{def}{=} \{\langle x, 1^{2^{|x|^t}}\rangle : x \in L\}$$

$L_{\text{pad}} \in \text{NP}^{\text{NP}^{\text{NP}}}$, since we can define a NDTM M' with oracle in NP^{NP} that on a n-length input y accepts iff there is a z such that $y = \langle z, 1^{2^{|z|^t}}\rangle$ and $M(z) = 1$ (a simulation that can be done in time linear on the length of y) $\Leftrightarrow z \in L \Leftrightarrow y \in L_{\text{pad}}$. Because $\text{NP}^{\text{NP}^{\text{NP}}} \subseteq \text{NP}^{\text{NP}}$ (by Karp-Lipton), we have $L_{\text{pad}} \in \text{NP}^{\text{NP}}$; let M'' be a NDTM with oracle in NP witnessing that. Then, we can define another NDTM M''' with oracle in NP that decides L in time exponential on the length of the input: given x, simply pad the input (getting $\langle x, 1^{2^{|x|^t}}\rangle$) and then simulate M''. We get $L \in \text{NE}^{\text{NP}}$; since L was arbitrary, $\text{NE}^{\text{NP}^{\text{NP}}} \subseteq \text{NE}^{\text{NP}}$. But then, $L(S^*) \in \text{NE}^{\text{NP}}$, and we are in the same place as in Subsection 4.2.1, but with $L(S^*)$ in place of SAT: with a similar argument, we get that for all c, we have $L^* \notin \text{SIZE}(n^c)$, as we wanted.

4.3 Adapting Kannan's Result

As announced on the Introduction, this argument is unable to conclude

$$K^c \notin \text{SIZE}(n^c) \tag{1.1}$$

due to the use of truth tables of exponential size. It is also impossible to be formalized in first-order bounded arithmetic for the same reason, since the existence of exponential functions is not guaranteed in first-order bounded-arithmetic theories. Thus, our strategy is to adapt the proof using smaller truth tables: we will use partial truth tables, which will have only

[2] For any n-length input x, the C'_n component of C''_n will transform x to an input y such that $x \in$ SAT $\Leftrightarrow y \in L^* \Leftrightarrow C''_n = 1$, the last equivalence due to the C_n component of C''_n.

polynomial size. As we will see, by doing this we will end up with a different proof, and in particular we will not need to use the whole Karp-Lipton Theorem, only the Self-Reducibility Lemma, Lemma 6.

Given two functions $f : \{0,1\}^n \to \{0,1\}$ and $g : \mathbb{N} \to \mathbb{N}$ such that $g(n) \leq 2^n$ for all n, let $S \in \{0,1\}^{g(n)}$ be a g-partial truth table (g-ptt) for f iff for all $x \in \{0,1\}^n$ such that $\text{bin}(x) < g(n)$ we have $S(x) = f(x)$, where $\text{bin}(x)$ is the number represented by x in binary and $S(x)$ is as previously defined (that is, a g-ptt for f is a truth table for f that considers only the first $g(n)$ inputs of f).

Given n and $c > 0$, consider n^{5c}-ptts over functions $f : \{0,1\}^n \to \{0,1\}$ (note this only makes sense when $n^{5c} \leq 2^n$, which happens for big enough n): there are $2^{n^{5c}}$-many of them. There are at most $2^{8 \cdot n^{c+1} \cdot \log(n^{c+1})}$ circuits of size at most n^{c+1}, but for $n > 8$,

$$8 \cdot n^{c+1} \cdot \log(n^{c+1}) \leq 8n^{2c+2} < n^{2c+3} \leq n^{5c}, \tag{4.2}$$

so that (for $n > 8$) we have:

$$2^{8 \cdot n^{c+1} \cdot \log(n^{c+1})} < 2^{n^{5c}}. \tag{4.3}$$

Let $n_c > 8$ be the smallest number such that $n_c^{5c} \leq 2^{n_c}$. Then, for $n \geq n_c$, there is at least one n^{5c}-ptt not computable by any circuit of size at most n^{c+1}: for $n \geq n_c$, let S_n^c be the lexicographically minimal n^{5c}-ptt fulfilling such a property; otherwise, if $n < n_c$, let S_n^c be a n^{5c}-length string composed entirely of zeros. Let L^c be the characteristic language of $\bigcup_{n \in \mathbb{N}} S_n^c$:

$$L^c \stackrel{def}{=} \{x : \text{bin}(x) < |x|^{5c} \wedge S^c_{|x|}(x) = 1\} \tag{4.4}$$

Since every function in $\mathcal{O}(n^c)$ is bounded by n^{c+1} for large enough n, by the previous argument we have $L^c \notin \text{SIZE}(n^c)$ for all $c > 0$ (we will reason this in more detail later). Now, define:

$$\begin{aligned}
\text{Computesptt}_n^{g(n)}(C,S) &\stackrel{def}{=} \forall x \in \{0,1\}^n(\text{bin}(x) < g(n) \to C(x) = S(x)) \\
\text{Smallereqptt}_n^{g(n)}(S,S') &\stackrel{def}{=} \forall x \in \{0,1\}^n \exists z \in \{0,1\}^n(\text{bin}(x) < g(n) \to \\
&\qquad S(x) \leq S'(x) \vee (\text{bin}(x) < \text{bin}(z) < g(n) \wedge S(z) < S'(z))).
\end{aligned}$$

Note if $g(n)$ is a polynomial, then for every n we have $\text{Computesptt}_n^{g(n)}$ and $\text{Smallereqptt}_n^{g(n)}$ are computable in polynomial time, since to compute them we can loop only over the $g(n)$ strings $x \in \{0,1\}^n$ such that $\text{bin}(x) < g(n)$ (in the case of $\text{Smallereqptt}_n^{g(n)}$, in each loop we may also loop over the $g(n) - \text{bin}(x) - 1$ strings $z \in \{0,1\}^n$ such that $\text{bin}(x) < \text{bin}(z) < g(n)$).

By (4.3), for $n > 8$ (in particular, for $n \geq n_c$), there are at most $2^{n^{5c}}$ circuits of size at most n^{c+1}, which allows us to bound the length of a string C representing a circuit of size at most n^{c+1} by $C \in \{0,1\}^{n^{5c}}$, instead of the usual $C \in \{0,1\}^{8n^{c+1}\log(n^{c+1})}$. For convenience, we will do so in what follows. Then, we can define:

$$L'_{c,n} \stackrel{def}{=} \{S \in \{0,1\}^{n^{5c}} : \forall C \in \{0,1\}^{n^{5c}}(\text{Size}(C) \leq n^{c+1} \to \neg\text{Computesptt}_n^{n^{5c}}(C,S))\}$$
$$L''_{c,n} \stackrel{def}{=} \{S \in \{0,1\}^{n^{5c}} : \forall S' \in \{0,1\}^{n^{5c}}(S' \notin L'_{c,n} \vee \text{Smallereqptt}_n^{n^{5c}}(S,S'))\}.$$

We have $S \in L'_{c,n}$ iff S is a n^{5c}-ptt not computed by any circuit of size at most n^{c+1} and $S \in L''_{c,n}$ iff S is smaller or equal than any n^{5c}-ptt fulfilling that property, so we can express L^c as follows:

$$L^c = \{x : |x| \geq n_c \wedge \exists S \in \{0,1\}^{|x|^{5c}}(S \in L'_{c,|x|} \wedge S \in L''_{c,|x|} \wedge S(x) = 1)\}$$

Let $n = |x|$ in the following formulas. Unfolding the definitions of $L'_{c,n}$, of $L''_{c,n}$, we obtain:

$$\begin{aligned}
L^c \;=\; \{x :\;& n \geq n_c \wedge \exists S \in \{0,1\}^{n^{5c}} \\
& (\forall C \in \{0,1\}^{n^{5c}}(\text{Size}(C) \leq n^{c+1} \to \neg\text{Computesptt}_n^{n^{5c}}(C,S)) \wedge \\
& \forall S' \in \{0,1\}^{n^{5c}}(\exists C' \in \{0,1\}^{n^{5c}}(\text{Size}(C') \leq n^{c+1} \wedge \text{Computesptt}_n^{n^{5c}}(C',S')) \\
& \vee \text{Smallereqptt}_n^{n^{5c}}(S,S')) \wedge S(x) = 1)\} \\
\\
=\; \{x :\;& \exists S \in \{0,1\}^{n^{5c}} \forall C \in \{0,1\}^{n^{5c}} \forall S' \in \{0,1\}^{n^{5c}} \exists C' \in \{0,1\}^{n^{5c}} \\
& (n \geq n_c \wedge (\text{Size}(C) \leq n^{c+1} \to \neg\text{Computesptt}_n^{n^{5c}}(C,S)) \wedge \\
& ((\text{Size}(C') \leq n^{c+1} \wedge \text{Computesptt}_n^{n^{5c}}(C',S')) \vee \text{Smallereqptt}_n^{n^{5c}}(S,S')) \wedge \\
& S(x) = 1)\}.
\end{aligned} \tag{4.5}$$

Since tuples of equal quantifiers can be expressed by single quantifiers in an equivalent formula, we see L^c is definable by a formula with $\exists\forall\exists$ prenex and a polynomial-time computable matrix (because $\text{Computesptt}_n^{n^{5c}}$ and $\text{Smallereqptt}_n^{n^{5c}}$ are polynomial-time computable). And since on this formula all quantifiers range over strings of length bounded polynomially on the length of the input, we conclude $L^c \in \Sigma_3^{\text{P}}$. To summarize, we have the following theorem:

Theorem 5. *For every $c > 0$, we have $L^c \in \Sigma_3^{\text{P}}$ and $L^c \notin \text{SIZE}(n^c)$.*

Proof. Fix $c > 0$. We have already argued $L^c \in \Sigma_3^{\text{P}}$, so let us assume $L^c \in \text{SIZE}(n^c)$ towards a contradiction. Let $\{C_n\}_{n\in\mathbb{N}}$ be an $\mathcal{O}(n^c)$-size circuit family deciding L^c; concretely, let k and m_c be such that $\text{Size}(C_n) \leq kn^c$ for all $n \geq m_c$. Then, let $n_{\max} = \max(n_c, m_c, k)$: for all

$n \geq n_{\max}$, the following holds:

$$\begin{array}{ll} n > 8 & (\text{Because } n_{\max} \geq n_c) \\ n^{5c} \leq 2^n & (\text{Because } n_{\max} \geq n_c) \\ \text{Size}(C_n) \leq kn^c & (\text{Because } n_{\max} \geq m_c) \\ kn^c \leq n^{c+1} & (\text{Because } n_{\max} \geq k). \end{array}$$

Thus, fix $n > n_{\max}$. Let $S \in \{0,1\}^{n^{5c}}$ be the lexicographically minimal n^{5c}-ptt not computable by any circuit of size at most n^{c+1}, which exists by the counting argument surrounding (4.3): then, there exists $x \in \{0,1\}^n$ such that $\text{bin}(x) < n^{5c}$ and $S(x) \neq C_n(x)$. By (4.4), we have $x \in L^c$ iff $S(x) = 1$, which, together with the fact $\{C_n\}_{n\in\mathbb{N}}$ decides L^c, means we have:

$$C_n(x) = 1 \Leftrightarrow x \in L^c \Leftrightarrow S(x) = 1 \Leftrightarrow C_n(x) \neq 1,$$

a contradiction. □

Before proceeding, let us prove an auxiliary lemma.

Lemma 7. *Let A and B be languages, let $c, d > 0$ and let $f : \{0,1\}^* \to \{0,1\}^*$ be a function computable by an $\mathcal{O}(n^d)$-size circuit family. If $f : A \leq_m^{\text{P/poly}} B$ and $B \in \text{SIZE}(n^c)$, then $A \in \text{SIZE}(n^{\max(c,d)})$.*

Proof. Fix $c, d > 0$ and let A, B and f be as in the statement. Let $\{C_n\}_{n\in\mathbb{N}}$ be an $\mathcal{O}(n^c)$-size circuit family deciding B and let $\{C'_n\}_{n\in\mathbb{N}}$ be an $\mathcal{O}(n^d)$-size circuit family computing f. Now, we simply build an $\mathcal{O}(n^{\max(c,d)})$-size circuit family $\{C''_n\}_{n\in\mathbb{N}}$, where we get C''_n by composing C'_n with C_n. Since $\{C''_n\}_{n\in\mathbb{N}}$ decides A, we get $A \in \text{SIZE}(n^{\max(c,d)})$.

□

Finally, we are in a position to prove (1.1), Kannan's result: for every $c > 0$, there is a language $K^c \in \Sigma_2^{\text{P}}$ such that $K^c \notin \text{SIZE}(n^c)$. Let us proceed to define K^c.

Fix $c > 0$. Below, let $n = |x|$. Since L^c belongs to Σ_3^{P}, there is a polynomial-time TM M_c and a $d_c > 0$ such that

$$L^c = \{x : \exists w_1 \in \{0,1\}^{n^{d_c}} \forall w_2 \in \{0,1\}^{n^{d_c}} \exists w_3 \in \{0,1\}^{n^{d_c}} M_c(x, w_1, w_2, w_3) = 1\}. \tag{4.6}$$

We will use M_c and d_c to build K^c. The idea is to invoke a circuit that outputs a satisfying assignment on input a satisfiable formula to eliminate the innermost existential quantifier of

the definition (4.6) of L^c; such a circuit will have polynomial size in case SAT $\in$ P/poly by Lemma 6. Let e_c be such that, on every input $\langle x, w_1, w_2, w_3\rangle$ such that $w_1, w_2, w_3 \in \{0,1\}^{n^{d_c}}$, we have M_c runs in n^{e_c} time.

Let f be the polynomial-time computable function guaranteed by Theorem 3. Given $n > 0$, we define:

$$\varphi_n^c(\overline{x}_1, \overline{x}_2, \overline{x}_3, \overline{x}_4, \overline{z}) \stackrel{def}{=} f(M_c, 1^n, 1^{n^{d_c}}, 1^{n^{d_c}}, 1^{n^{d_c}}, 1^{n^{e_c}});$$

that is, for every $x \in \{0,1\}^n$ and all $w_1, w_2, w_3 \in \{0,1\}^{n^{d_c}}$, we have:

$$M_c(x, w_1, w_2, w_3) \text{ accepts in } n^{e_c} \text{ steps } \iff \exists z \in \{0,1\}^{|\overline{z}|}\varphi_n^c(\overline{x}_1/x, \overline{x}_2/w_1, \overline{x}_3/w_2, \overline{x}_4/w_3, \overline{z}/z). \tag{4.7}$$

Let c' be as guaranteed by Lemma 6. Let a_c be such that for all $n > 0$, we have:

$$8|\varphi_n^c|^{c'} \log(|\varphi_n^c|^{c'}) \leq a_c n^{a_c}.$$

This allows us to define:

$$\begin{aligned} K_1^c \stackrel{def}{=} \{x : \; & \exists D \in \{0,1\}^{a_c n^{a_c}} \exists w_1 \in \{0,1\}^{n^{d_c}} \forall w_2 \in \{0,1\}^{n^{d_c}} \\ & \varphi_n^c(x, w_1, w_2, D(\varphi_n^c(x, w_1, w_2, \overline{x}_4, \overline{z})))\}, \end{aligned} \tag{4.8}$$

where the idea is that D represents a polynomial-size circuit that outputs a satisfying assignment on input a satisfiable formula, which allows us to erase both the innermost existential quantifier from the definition (4.6) and the existential quantifier of the right hand side of (4.7). We can see $K_1^c \in \Sigma_2^{\mathrm{P}}$, since φ_n^c is computable on polynomial time on the length of x (this polynomial will depend on d_c and e_c, ultimately on c). Then, we can define:

$$K^c \stackrel{def}{=} K_1^c \oplus \mathrm{SAT},$$

and we can prove:

Theorem 6. *For every $c > 0$, we have $K^c \in \Sigma_2^{\mathrm{P}}$ and $K^c \notin \mathrm{SIZE}(n^c)$.*

Proof. Fix $c > 0$. We know $K_1^c \in \Sigma_2^{\mathrm{P}}$ by (4.8); since SAT $\in \Sigma_1^{\mathrm{P}} \subseteq \Sigma_2^{\mathrm{P}}$ and Σ_2^{P} is closed under disjoint union, we get $K^c \in \Sigma_2^{\mathrm{P}}$. Now, we will show $K^c \notin \mathrm{SIZE}(n^c)$ by case distinction.

> Case SAT $\notin \mathrm{SIZE}(n^c)$. Assume $K^c \in \mathrm{SIZE}(n^c)$ towards a contradiction. Since the function $f : x \mapsto \langle x, 1\rangle$ is computable by a $\mathcal{O}(n)$-size circuit family, we have $f :$ SAT $\leq_{\mathrm{m}}^{\mathrm{P/poly}} K^c$, so that we get SAT $\in \mathrm{SIZE}(n^c)$ by Lemma 7: contradiction.
>
> Case SAT $\in \mathrm{SIZE}(n^c)$. In this case, we have $L^c = K_1^c$:

$\boxed{K_1^c \subseteq L^c}$ Let $x \in K_1^c$. By (4.8), we have some $D \in \{0,1\}^{a_c|x|^{a_c}}$ and some $w_1 \in \{0,1\}^{|x|^{d_c}}$ such that for every $w_2 \in \{0,1\}^{|x|^{d_c}}$, we have:

$$\varphi_n^c(x, w_1, w_2, D(\varphi_n^c(x, w_1, w_2, \overline{x}_4, \overline{z}))). \tag{4.9}$$

Fix $w_2 \in \{0,1\}^{|x|^{d_c}}$ and let w_3 represent the first $|x|^{d_c}$ bits of $D(\varphi_n^c(x, w_1, w_2, \overline{x}_4, \overline{z}))$. Using (4.7) and (4.9), we get:

$$M_c(x, w_1, w_2, w_3) = 1,$$

so that, by (4.6), w_1 witnesses $x \in L^c$.

$\boxed{L^c \subseteq K_1^c}$ Let $x \in L^c$. Since SAT $\in$ SIZE(n^c), by Lemma 6 there is some $D \in \{0,1\}^{a_c|x|^{a_c}}$ representing a $|\varphi_{|x|}^c|$-input circuit that outputs a satisfying assignment on input a satisfiable formula. Since $x \in L^c$, let $w_1 \in \{0,1\}^{|x|^{d_c}}$ be as guaranteed by (4.6). Fix $w_2 \in \{0,1\}^{|x|^{d_c}}$: we have

$$\exists w_3 \in \{0,1\}^{|x|^{d_c}} M_c(x, w_1, w_2, w_3) = 1,$$

so that, by (4.7), we get

$$\exists w_3 \in \{0,1\}^{|x|^{d_c}} \exists z \in \{0,1\}^{|\overline{z}|} \varphi_n^c(x, w_1, w_2, w_3, z),$$

which, by definition of D, implies

$$\varphi_n^c(x, w_1, w_2, D(\varphi_n^c(x, w_1, w_2, \overline{x}_4, \overline{z}))),$$

which, by (4.8), means D and w_1 witness $x \in K_1^c$.

To complete the argument, assume towards a contradiction that $K^c \in$ SIZE(n^c). Since the function $f : x \mapsto \langle x, 0\rangle$ is computable by a $\mathcal{O}(n)$-size circuit family, we have $f : K_1^c \leq_{\mathrm{m}}^{\mathrm{P/poly}} K^c$, so that we get $K_1^c \in$ SIZE(n^c) by Lemma 7. Since $K_1^c = L^c$, this means $L^c \in$ SIZE(n^c), which contradicts Theorem 5.

□

Using Theorem 6, we can also prove (1.2). Recall (4.1), the definition of L^*:

$$L^* \stackrel{def}{=} \{\langle M, x, 1^t\rangle : \exists y \in \{0,1\}^{2^t} \forall z \in \{0,1\}^{2^t} (M(x, y, z) = 1)\}. \tag{4.1}$$

We have:

Theorem 7. *For all $c > 0$, we have $L^* \notin$ SIZE(n^c).*

Proof. It suffices to prove it for $c > 1$, so fix $c > 1$. Suppose $L^* \in \mathrm{SIZE}(n^c)$ towards a contradiction. Since $K^c \in \Sigma_2^{\mathrm{P}}$, let M' be a polynomial-time TM and $d' > 0$ be such that for all $x \in \{0,1\}^*$, we have:

$$x \in K^c \iff \exists y \in \{0,1\}^{|x|^{d'}} \forall z \in \{0,1\}^{|x|^{d'}} M'(x,y,z) = 1.$$

Now, we want to define a linear-time reduction from K^c to L^*. For this purpose, we define a TM M'' that acts as M' on inputs padded in the appropriate way. More precisely: on input $\langle x, y, z\rangle$,

1. If $|y| \neq 2^{d'|x|}$ or $|z| \neq 2^{d'|x|}$, then $M''(x,y,z) = 0$.
2. Else, if there are no $y' \in \{0,1\}^{|x|^{d'}}$ such that $y = y'0^m$ for $m = 2^{d'|x|} - |x|^{d'}$, then $M''(x,y,z) = 0$.
3. Else, if there are no $z' \in \{0,1\}^{|x|^{d'}}$ such that $z = z'0^m$ for $m = 2^{d'|x|} - |x|^{d'}$, then $M''(x,y,z) = 1$.
4. Else, let $y', z' \in \{0,1\}^{|x|^{d'}}$ be such that $y'0^m = y$ and $z'0^m = z$ for $m = 2^{d'|x|} - |x|^{d'}$. Then, $M''(x,y,z) = M'(x,y',z')$.

We can now define $f : x \mapsto \langle M'', x, 1^{d'|x|}\rangle$. By the definition of M'' and the fact $n^{d'} \leq 2^{d'n}$ for all $n > 0$, we have $x \in K^c$ iff $f(x) \in L^*$ for all $x \in \{0,1\}^*$. Since we have f is linear-time computable (because the length of the string M'' depends only on d', and d' depends only on c, we have M'' is computable in time independent on the length of x), f must be computable by a $\mathcal{O}(n^2)$-size circuit family by Lemma 1, and we have $f : K^c \leq_{\mathrm{m}}^{\mathrm{P/poly}} L^*$. By Lemma 7 and $L^* \in \mathrm{SIZE}(n^c)$, we get $K^c \in \mathrm{SIZE}(n^{\max(c,2)})$; since $c > 1$, we have $K^c \in \mathrm{SIZE}(n^c)$, which contradicts Theorem 6. □

Chapter 5

Bounded Karp-Lipton

In this chapter, we formalize in bounded arithmetic the proof of the Karp-Lipton Theorem developed in Chapter 3; more precisely, in the bounded arithmetic Theory T_2. First, we define a number of L_{BA}-formulas that we will use in the following formalization, and then we adapt the three results presented in Chapter 3. As we will see, proofs in bounded arithmetic are unavoidably longer and more detailed than their natural counterparts, so sometimes it will be useful to prove additional lemmas to simplify the formalization.

5.1 Definitions

First of all, we define some functions:

$\mathrm{evaltm}_r(M, x_1, \ldots, x_r, k) \stackrel{def}{=}$ if M represents a TM and k represents a string of length t, returns the output of running M on input $\langle x_1, \ldots, x_r \rangle$ during t steps; otherwise, returns 0.

$\mathrm{subs}(\varphi, v) \stackrel{def}{=}$ if φ represents a quantified formula $\psi(\overline{z})$ such that $\overline{z}$ is a sequence of m propositional variables and v represents a string of length m', then if $m' < m$, returns a string representing $\psi(\overline{z}_1/v, \overline{z}_2)$, where $\overline{z}_1$ represents the first m' propositional variables in $\overline{z}$ and $\overline{z}_2$ represents the rest of variables; and if $m' \geq m$, returns a string representing $\psi(\overline{z}/u)$, where u represents the first m bits of v; otherwise, returns 0.

When φ represents such a formula we will use the notation $\varphi(\overline{z}/v) := \mathrm{subs}(\varphi, v)$, or just $\varphi(v) := \mathrm{subs}(\varphi, v)$ when it is clear which variables are being replaced.

$\text{subs-ex}(\varphi, v) \stackrel{def}{=}$ if φ represents an existential quantified sentence of the form $\exists \overline{z} \psi(\overline{z})$

such that ψ is not an existential formula, returns a string representing $\psi(\overline{z}/v)$; otherwise, returns 0.

subs-fa$(\varphi, v) \stackrel{def}{=}$ if φ represents a universal quantified sentence of the form $\forall \overline{z}\psi(\overline{z})$ such that ψ is not a universal formula, returns a string representing $\psi(\overline{z}/v)$; otherwise, returns 0.

$\mathrm{pr}(x, i) \stackrel{def}{=}$ if x represents a string of length at most i, returns the i-th bit of x; otherwise, returns 0. We will use the notation $(x)_i := \mathrm{pr}(x, i)$.

$\mathrm{size}(C) \stackrel{def}{=}$ if C represents a circuit, returns the size of C; otherwise, returns 0.

$\mathrm{evalc}(C, x) \stackrel{def}{=}$ if C represents a n-input circuit, where n is such that $x \in \{0,1\}^n$, returns the output of C on input x; otherwise, returns 0. We will use the notation $C(x) := \mathrm{evalc}(C, x)$.

And some predicates (note there is a difference between $\doteq$, a symbol that we will use in some of our defined formulas to denote equality, and $=$, the equality of first-order logic):

$C(x) \doteq y \stackrel{def}{=}$ "C represents a n-input m-output circuit, where n is such that $x \in \{0,1\}^n$ and m is such that $y \in \{0,1\}^m$ and $\mathrm{evalc}(C, x) = y$".

$x \in \Sigma_0$-TQBF $\stackrel{def}{=}$ "x represents a true propositional sentence".

$x \in \Pi_0$-TQBF $\stackrel{def}{=}$ "x represents a true propositional sentence".

$\mathrm{clock}_r(M, x_1, \ldots, x_r, k) \stackrel{def}{=}$ "M represents a TM, k represents a string and M halts in at most t steps on input $\langle x_1, ..., x_r \rangle$, where t is the length of k".

It is easy to see all of them are polynomial-time computable, so that they are expressible by L_{BA}-formulas (as explained on Definition 35). The same happens with the following formulas ($x \in \Sigma_i$-TQBF and $x \in \Pi_i$-TQBF are defined recursively). For every $n \in \mathrm{Log}_{>1}$ and every x representing an n-length string, we have:

$x \in \Sigma_{i+1}$-TQBF $\stackrel{def}{=}$ $\exists v \in \{0,1\}^n(\text{subs-ex}(x, v) \in \Pi_i\text{-TQBF})$.

$x \in \Pi_{i+1}$-TQBF $\stackrel{def}{=}$ $\forall v \in \{0,1\}^n(\text{subs-fa}(x, v) \in \Sigma_i\text{-TQBF})$.

$x \in \mathrm{SAT} \stackrel{def}{=}$ $\exists v \in \{0,1\}^n(\mathrm{subs}(x, v) \in \Sigma_0\text{-TQBF})$.

$\mathrm{evalbf}(\varphi, u) \stackrel{def}{=}$ $\varphi(u) \in \Sigma_0$-TQBF. When there is no danger of ambiguity, we will simply use the notation $\varphi(u) := \mathrm{evalbf}(\varphi, u)$.

$$M(x_1,\ldots,x_r,k) \doteq y \stackrel{def}{=} \ \mathrm{clock}_r(M,x_1,\ldots,x_r,k) \wedge \mathrm{evaltm}_r(M,x_1,\ldots,x_r,k) = y.$$

$$\begin{aligned} L \in \Sigma_i\text{-TIME}(n^c) \ \stackrel{def}{=} \ & \exists M \in \mathrm{Log}_{>1}\ \exists k \in \mathrm{Log}_{>1}\ \exists l \in \mathrm{Log}_{>1}\ \exists n_0 \in \mathrm{Log}_{>1} \\ & \forall n \in \mathrm{Log}_{>1}\ \forall x \in \{0,1\}^n (n > n_0 \rightarrow \\ & (x \in L \leftrightarrow \exists v_1 \in \{0,1\}^{ln^c} \cdots Q_i v_i \in \{0,1\}^{ln^c} M(x,v_1,\ldots,v_i,1^{kn^c}) \doteq 1)) \end{aligned}$$

$$\begin{aligned} L \in \Pi_i\text{-TIME}(n^c) \ \stackrel{def}{=} \ & \exists M \in \mathrm{Log}_{>1}\ \exists k \in \mathrm{Log}_{>1}\ \exists l \in \mathrm{Log}_{>1}\ \exists n_0 \in \mathrm{Log}_{>1} \\ & \forall n \in \mathrm{Log}_{>1}\ \forall x \in \{0,1\}^n (n > n_0 \rightarrow \\ & (x \in L \leftrightarrow \forall v_1 \in \{0,1\}^{ln^c} \cdots \overline{Q}_i v_i \in \{0,1\}^{ln^c} M(x,v_1,\ldots,v_i,1^{kn^c}) \doteq 1)) \end{aligned}$$

$$\begin{aligned} L \in \mathrm{SIZE}(n^c) \ \stackrel{def}{=} \ & \exists k \in \mathrm{Log}_{>1}\ \exists n_0 \in \mathrm{Log}_{>1}\ \forall n \in \mathrm{Log}_{>1}\ \exists C \in \{0,1\}^{8kn^c|kn^c|} \forall x \in \{0,1\}^n \\ & (n > n_0 \rightarrow (\mathrm{size}(C) \leq kn^c \wedge (x \in L \rightarrow C(x) \doteq 1) \wedge (x \notin L \rightarrow C(x) \doteq 0))). \end{aligned}$$

As usual, in the above definitions $A \notin B$ is notation for $\neg(A \in B)$.

5.2 Bounded Hierarchy Collapse

We want to prove the Hierarchy Collapse Lemma in T_2, but first, we prove some useful lemmas.

Lemma 8. *For every $i > 0$, if $x \in L$ is defined by an Σ_i^b L_{BA}-formula, then there exist $k, c > 0$ and a TM M such that we have:*

$$\begin{aligned} T_2 \ \vdash \ & \forall n \in \mathrm{Log}_{>1}\ \forall x \in \{0,1\}^n \\ & (x \in L \leftrightarrow \exists v_1 \in \{0,1\}^{n^c} \cdots Q_i v_i \in \{0,1\}^{n^c} M(x,v_1,\ldots,v_i,1^{kn^c}) \doteq 1). \end{aligned}$$

Proof. Fix $i > 0$, and let $x \in L$ be defined by an Σ_i^{b} L_{BA}-formula. By standard manipulation of quantifiers we can assume that this formula has the special form indicated below. For all $n \in \mathrm{Log}_{>1}$ and $x \in \{0,1\}^n$, assume that

$$x \in L \leftrightarrow \exists u_1 \in \{0,1\}^{p_1(n)} \cdots Q_i u_i \in \{0,1\}^{p_i(n)} \varphi(x,u_1,\ldots,u_i),$$

where $p_1,\ldots,p_i$ are polynomials and φ is a QF L_{BA}-formula. We work in T_2.

Let M work as follows: on input $\langle x, v_1, \ldots, v_i \rangle$, the machine simply checks $\varphi(x, v_1', \ldots, v_i')$ is true, where v_j' represents the first $p_j(n)$-bits of v_j. We can see M runs in polynomial time, so let $c', k > 0$ be such that M runs in $kn^{c'}$-time. Let $c \geq c'$ be large enough to bound all

polynomials $p_1, \ldots, p_i$; that is, large enough that $m^c \geq p_j(m)$ for every $m > 1$ and every $j \in \{1, \ldots, i\}$. Fix $n \in \mathrm{Log}_{>1}$ and $x \in \{0,1\}^n$. We have:

$$x \in L \leftrightarrow \exists v_1 \in \{0,1\}^{n^c} \cdots Q_i v_i \in \{0,1\}^{n^c} M(x, v_1, \ldots, v_i, 1^{kn^c}) \doteq 1,$$

and the conclusion follows. □

Lemma 9. *For every $i > 0$, if $x \in L$ is defined by a Σ_i^b L_{BA}-formula and $x \in L'$ is defined by a Π_i^b L_{BA}-formula, then for some $c > 0$ we have:*

$$T_2 \vdash L \in \Sigma_i\text{-TIME}(n^c) \wedge L' \in \Pi_i\text{-TIME}(n^c).$$

Proof. Fix $i > 0$. We work in T_2.

Given $x \in L$ defined by a Σ_i^{b} L_{BA}-formula and $x \in L'$ defined by a Π_i^{b} L_{BA}-formula, we have $L \in \Sigma_i\text{-TIME}(n^{c'})$ for some $c' > 0$ follows from Lemma 8, and $L' \in \Pi_i\text{-TIME}(n^{c''})$ for some $c'' > 0$ is proven by a symmetrical argument. The conclusion follows for $c = \max(c', c'')$. □

Lemma 10. *For all $i > 0$ there exists $c > 0$ such that*

$$T_2 \vdash \Sigma_i\text{-TQBF} \in \Sigma_i\text{-TIME}(n^c) \wedge \Pi_i\text{-TQBF} \in \Pi_i\text{-TIME}(n^c).$$

Proof. Fix $i > 0$. If we develop the definitions of Σ_i-TQBF and Π_i-TQBF, we see they are defined respectively by a Σ_i^{b} L_{BA}-formula and by a Π_i^{b} L_{BA}-formula, so that the conclusion follows from Lemma 9. □

Lemma 11. *For all $i > 0$, for all $j > 0$ and for all $c > 0$, there exists $c' > 0$ such that:*

$$T_2 \vdash \begin{array}{l} \Sigma_i\text{-TQBF} \in \Pi_j\text{-TIME}(n^c) \rightarrow \Pi_i\text{-TQBF} \in \Sigma_j\text{-TIME}(n^{c'}) \quad \wedge \\ \Pi_i\text{-TQBF} \in \Sigma_j\text{-TIME}(n^c) \rightarrow \Sigma_i\text{-TQBF} \in \Pi_j\text{-TIME}(n^{c'}). \end{array}$$

Proof. Fix i, j, $c > 0$. We reason in T_2. We only prove

$$\Sigma_i\text{-TQBF} \in \Pi_j\text{-TIME}(n^c) \rightarrow \Pi_i\text{-TQBF} \in \Sigma_j\text{-TIME}(n^{c'}),$$

since the other implication is proved by a symmetric argument.

Suppose Σ_i-TQBF $\in \Pi_j$-TIME(n^c) to conclude Π_i-TQBF $\in \Sigma_j$-TIME$(n^{c'})$ for some $c' > 0$. Let M and l be as witnessed by Σ_i-TQBF $\in \Pi_j$-TIME(n^c). We define another TM M' as follows: on input $\langle y, u_1, \ldots, u_i \rangle$, the machine M' checks y represents a propositional Π_i-sentence, then runs M on input $\langle \tilde{y}, u_1', \ldots, u_i' \rangle$, where $\tilde{y}$ represents $\neg y$ with the negation

"pushed inside"(so that it represents a propositional Σ_i-sentence), $\tilde{m}$ is the length of $\tilde{y}$, and u'_k represents the first $l\tilde{m}^c$ bits of the string represented by u_k, and accepts iff M rejects. It is easy to see M' is a polynomial-time TM, so let $c' \geq c$ be such that M' halts in $c'n^{c'}$-time. Fix $n \in \mathrm{Log}_{>1}$ and $x \in \{0,1\}^n$. We have:

$$x \in \Pi_i\text{-TQBF} \leftrightarrow \exists u_1 \in \{0,1\}^{n^{c'}} \cdots Q_j u_j \in \{0,1\}^{n^{c'}} M(x, u_1, \ldots, u_j, 1^{c'n^{c'}}) \doteq 1,$$

so that M', $k = c'$, and $l = n_0 = 1$ witness $\Pi_i\text{-TQBF} \in \Sigma_j\text{-TIME}(n^{c'})$. □

Lemma 12. *Let $c, i, j > 0$. For every $c' > c$ and every L, we have:*

$$T_2 \vdash \begin{array}{l} L \in \Sigma_j\text{-TIME}(n^c) \rightarrow L \in \Sigma_j\text{-TIME}(n^{c'}) \quad \wedge \\ L \in \Pi_j\text{-TIME}(n^c) \rightarrow L \in \Pi_j\text{-TIME}(n^{c'}). \end{array}$$

Proof. Fix $c, i, j > 0$, and fix $c' > c$ and L. We reason in T_2.

Assume $L \in \Sigma_j\text{-TIME}(n^c)$, and let M, k, l and n_0 witness it. We build a TM M' as follows: on input $\langle y, v_1, \ldots, v_j \rangle$, let y represent a string of length m; then, M' runs M on input $\langle y, v'_1, \ldots, v'_j \rangle$, where v'_l represents the first lm^c bits of v_l. Let k' be such that M' runs in $k'n^{c'}$-time. Fix $n \in \mathrm{Log}_{>1}$ such that $n > n_0$ and $x \in \{0,1\}^n$. We have:

$$x \in L \leftrightarrow \exists v_1 \in \{0,1\}^{n^{c'}} \cdots Q_j v_1 \in \{0,1\}^{n^{c'}} M'(x, v_1, \ldots, v_j, 1^{k'n^{c'}}) \doteq 1,$$

so that M', k', $l = 1$ and n_0 witness $L \in \Sigma_j\text{-TIME}(n^{c'})$. We can prove $L \in \Pi_j\text{-TIME}(n^c) \rightarrow L \in \Pi_j\text{-TIME}(n^{c'})$ in a similar way. □

Now, we can adapt the Hierarchy Collapse Lemma:

Lemma 13 (Bounded Hierarchy Collapse)**.** *For all $i > 0$, for all $c > 0$ and for all $j \geq i$, there exists $c' > 0$ such that:*

$$T_2 \vdash \begin{array}{l} \Sigma_i\text{-TQBF} \in \Pi_i\text{-TIME}(n^c) \wedge \\ \Pi_i\text{-TQBF} \in \Sigma_i\text{-TIME}(n^c) \end{array} \rightarrow \begin{array}{l} \Sigma_j\text{-TQBF} \in \Sigma_i\text{-TIME}(n^{c'}) \wedge \Sigma_j\text{-TQBF} \in \Pi_i\text{-TIME}(n^{c'}) \wedge \\ \Pi_j\text{-TQBF} \in \Sigma_i\text{-TIME}(n^{c'}) \wedge \Pi_j\text{-TQBF} \in \Pi_i\text{-TIME}(n^{c'}) \end{array}$$

Proof. Fix $i > 0$ and $c > 0$. We reason by induction on $j \geq i$ (this induction is done outside T_2).

The case $j = i$ is easy: reasoning in T_2, assume

$$\Sigma_i\text{-TQBF} \in \Pi_i\text{-TIME}(n^c) \wedge \Pi_i\text{-TQBF} \in \Sigma_i\text{-TIME}(n^c).$$

Let $c' = \max(c, c_0)$, where c_0 is given by Lemma 10. Then,

$$\Sigma_i\text{-TQBF} \in \Sigma_i\text{-TIME}(n^{c'}) \wedge \Sigma_i\text{-TQBF} \in \Pi_i\text{-TIME}(n^{c'}) \wedge$$
$$\Pi_i\text{-TQBF} \in \Sigma_i\text{-TIME}(n^{c'}) \wedge \Pi_i\text{-TQBF} \in \Pi_i\text{-TIME}(n^{c'})$$

follows from Lemma 12. Now, outside T_2, let us assume the property holds for $j \geq i$ and prove it holds for $j+1$. From now on, we reason in T_2. Assume

$$\Sigma_i\text{-TQBF} \in \Pi_i\text{-TIME}(n^c) \wedge \Pi_i\text{-TQBF} \in \Sigma_i\text{-TIME}(n^c).$$

Our inductive hypothesis implies that, for some $c' > 0$ (fix it),

$$\begin{aligned} &\Sigma_j\text{-TQBF} \in \Sigma_i\text{-TIME}(n^{c'}) \wedge \Sigma_j\text{-TQBF} \in \Pi_i\text{-TIME}(n^{c'}) \wedge \\ &\Pi_j\text{-TQBF} \in \Sigma_i\text{-TIME}(n^{c'}) \wedge \Pi_j\text{-TQBF} \in \Pi_i\text{-TIME}(n^{c'}). \end{aligned} \tag{5.1}$$

We want $c'' > 0$ such that:

$$\Sigma_{j+1}\text{-TQBF} \in \Sigma_i\text{-TIME}(n^{c''}) \wedge \Sigma_{j+1}\text{-TQBF} \in \Pi_i\text{-TIME}(n^{c''}) \wedge$$
$$\Pi_{j+1}\text{-TQBF} \in \Sigma_i\text{-TIME}(n^{c''}) \wedge \Pi_{j+1}\text{-TQBF} \in \Pi_i\text{-TIME}(n^{c''}).$$

We begin by showing $\Sigma_{j+1}\text{-TQBF} \in \Sigma_i\text{-TIME}(n^{c''})$ for some $c'' > 0$. At this point, we want to define M' and c'', but we need to introduce some notation first. If y represents a quantified formula

$$\exists \overline{u}_1 \forall \overline{u}_2 \cdots Q_{j+1} \overline{u}_{j+1} \varphi(\overline{u}_1, \ldots, \overline{u}_{j+1}),$$

with φ a propositional formula, and v represents a m_1-length binary string, for m_1 the length of $\overline{u}_1$, then let $y(v)$ represent the quantified formula $\forall \overline{u}_2 \cdots Q_{j+1} \overline{u}_{j+1} \varphi(v, \overline{u}_2, \ldots, \overline{u}_{j+1})$; else, let $y(v)$ represent a fixed contradiction. Let $l(y, v)$ be the length of $y(v)$. Fix $n \in \text{Log}_{>1}$ and $x \in \{0,1\}^n$. We have:

$$x \in \Sigma_{j+1}\text{-TQBF} \leftrightarrow \exists u_1 \in \{0,1\}^{n^{c'}} (x(u_1) \in \Pi_j\text{-TQBF}) \tag{5.2}$$

where the length of u_1 can be bounded by $n^{c'}$ because it cannot in fact exceed n, the length of x. By (5.1) (in particular, by $\Pi_j\text{-TQBF} \in \Sigma_i\text{-TIME}(n^{c'})$), we know there is a TM M and some l, k such that

$$\begin{aligned} &\exists u_1 \in \{0,1\}^{n^{c'}} (x(u_1) \in \Pi_j\text{-TQBF}) \leftrightarrow \\ &\exists u_1 \in \{0,1\}^{ln^{c'}} \exists v_1 \in \{0,1\}^{ln^{c'}} \cdots Q_i v_i \in \{0,1\}^{ln^{c'}} M(x(u_1), v_1, \ldots, v_i, 1^{kl(x,u_1)^{c'}}) \doteq 1. \end{aligned} \tag{5.3}$$

Now we're ready to define M': let M' be a TM that on input $\langle y, w_1, \ldots, w_i \rangle$, where y represents a string of length n_1, checks w_1 represents a pair $\langle u_1, v_1 \rangle$ such that $u_1, v_1 \in \{0,1\}^{ln_1^{c'}}$,

checks y represents a propositional Σ_{j+1}-sentence with exactly k_1-many variables in the scope of the first existential quantifier, for k_1 the length of u_1, and runs M on input $\langle y(u_1), v_1, w_2', \ldots, w_i' \rangle$, where w_k' represents the first $ln_1^{c'}$ bits of the string represented by w_k. It is easy to see M' is a polytime TM, so let c'' be big enough so that M' halts on $c''m^{c''}$-time, and also so that for all m, a $m^{c''}$-length string can codify a pair of $m^{c'}$-length strings; fix c'' until the end of the proof. We have:

$$\exists u_1 \in \{0,1\}^{ln^{c'}} \exists v_1 \in \{0,1\}^{ln^{c'}} \cdots Q_i v_i \in \{0,1\}^{ln^{c'}} M(x(u_1), v_1, \ldots, v_i, 1^{kl(x,u_1)^{c'}}) \doteq 1 \leftrightarrow$$
$$\exists w_1 \in \{0,1\}^{n^{c''}} \cdots Q_i w_i \in \{0,1\}^{n^{c''}} M'(x, w_1, \ldots, w_i, 1^{c''n^{c''}}) \doteq 1$$

So, using (5.2) and (5.3), we get:

$$x \in \Sigma_{j+1}\text{-TQBF} \leftrightarrow \exists w_1 \in \{0,1\}^{n^{c''}} \cdots Q_i w_i \in \{0,1\}^{n^{c''}} M'(x, w_1, \ldots, w_i, 1^{c''n^{c''}}) \doteq 1, \quad (5.4)$$

and M', $k = c'$, $l = 1$ and $n_0 = 0$ witness Σ_{j+1}-TQBF $\in \Sigma_i$-TIME$(n^{c''})$. Fix M' until the end of the proof. Π_{j+1}-TQBF $\in \Pi_i$-TIME$(n^{c''})$ is proven in a symmetric way.

Now, we want to see Σ_{j+1}-TQBF $\in \Pi_i$-TIME$(n^{c'''})$ for some $c''' > 0$. As before, we want to define M'' and c''', but we need to introduce some definitions first. Fix $n \in \text{Log}_{>1}$, and $x \in \{0,1\}^n$. Since, by assumption, Σ_i-TQBF $\in \Pi_i$-TIME(n^c), there is a TM M and some l, k such that:

$$x \in \Sigma_i\text{-TQBF} \leftrightarrow \forall v_1 \in \{0,1\}^{ln^c} \cdots \overline{Q}_i v_i \in \{0,1\}^{ln^c} M(x, v_1, \ldots, v_i, 1^{kn^c}) \doteq 1 \quad (5.5)$$

Now, a small digression. For a string y of length m, consider the following statement:

$$\exists w_1 \in \{0,1\}^{m^{c''}} \cdots Q_i w_i \in \{0,1\}^{m^{c''}} M'(y, w_1, \ldots, w_i, 1^{c''m^{c''}}) \doteq 1.$$

By the Bounded version of Theorem 3 (which is provable in T_2), there are two polynomial time computable functions f and g such that for each string y, we have

$$f(M', m, m_1^{c''}, \ldots, m_i^{c''}, 1^{c''m^{c''}}) = \varphi_y(\overline{x}, \overline{x}_1, \ldots, \overline{x}_i, \overline{z})$$
$$g(M', m, m_1^{c''}, \ldots, m_i^{c''}, 1^{c''m^{c''}}) = \psi_y(\overline{x}, \overline{x}_1, \ldots, \overline{x}_i, \overline{z}),$$

where φ_y and ψ_y are propositional formulas, the length of $\overline{x}$ is m and the length of $\overline{x}_k$ is $m^{c''}$ for every $k \in \{1, \ldots, i\}$, and for all $w_1, \ldots, w_i \in \{0,1\}^{m^{c''}}$, we have:

$$M''(y, w_1, \ldots, w_i) \text{ accepts in } c''m^{c''} \text{ steps } \leftrightarrow \exists z \in \{0,1\}^l \varphi_y(\overline{x}/y, \overline{x}_1/w_1, \ldots, \overline{x}_i/w_i, \overline{z}/z)$$
$$M''(y, w_1, \ldots, w_i) \text{ accepts in } c''m^{c''} \text{ steps } \leftrightarrow \forall z \in \{0,1\}^l \psi_y(\overline{x}/y, \overline{x}_1/w_1, \ldots, \overline{x}_i/w_i, \overline{z}/z),$$

where l is the length of $\overline{z}$. Thus, given y, let $\widehat{y}$ represent $\exists w_1 \cdots Q_i w_i \exists z \varphi_y$ if $Q_i = \exists$ and $\exists w_1 \cdots Q_i w_i \forall z \psi_y$ if $Q_i = \forall$: for every y, we have $\widehat{y}$ is a propositional Σ_i-sentence which is true iff

$$\exists w_1 \in \{0,1\}^{m^{c''}} \cdots Q_i w_i \in \{0,1\}^{m^{c''}} M'(y, w_1, \ldots, w_i, 1^{c'' m^{c''}}) \doteq 1.$$

By the latter argument, $\widehat{y}$ is computable in time polynomial on the length of y. Using (5.4), we get:

$$x \in \Sigma_{j+1}\text{-TQBF} \leftrightarrow \widehat{x} \in \Sigma_i\text{-TQBF}. \tag{5.6}$$

We are now ready to define M''. Let M'' work as follows: on input $\langle z, u_1, \ldots, u_i \rangle$, let m and $\widehat{m}$ be such that $z \in \{0,1\}^m$ and $\hat{z} \in \{0,1\}^{\widehat{m}}$ and run M on input $\langle \widehat{z}, u'_1, \ldots, u'_i \rangle$ (where u'_k represents the first $\widehat{m}^c$ bits of the string represented by u_k). It is clear M'' runs on polynomial time, so let c''' be big enough that M'' halts on $c''' n^{c'''}$-time and that

$$\widehat{m}^c \leq m^{c'''}$$

for any $m > 1$ and any $z \in \{0,1\}^m$. We have:

$$\begin{aligned}
&\forall v_1 \in \{0,1\}^{ln^c} \cdots \overline{Q}_i v_i \in \{0,1\}^{ln^c} M(\widehat{x}, v_1, \ldots, v_i, 1^{kn^c}) \doteq 1 \leftrightarrow \\
&\forall u_1 \in \{0,1\}^{n^{c'''}} \cdots \overline{Q}_i u_i \in \{0,1\}^{n^{c'''}} M''(x, u_1, \ldots, u_i, 1^{c''' n^{c'''}}) \doteq 1,
\end{aligned}$$

so, by (5.5) and (5.6), we get:

$$x \in \Sigma_{j+1}\text{-TQBF} \leftrightarrow \forall u_1 \in \{0,1\}^{n^{c'''}} \cdots \overline{Q}_i u_i \in \{0,1\}^{n^{c'''}} M''(x, u_1, \ldots, u_i, 1^{c''' n^{c'''}}) \doteq 1.$$

And M'', $k = c'''$, $l = 1$ and $n_0 = 0$ witness Σ_{j+1}-TQBF $\in \Pi_i$-TIME$(n^{c'''})$, which entails Π_{j+1}-TQBF $\in \Sigma_i$-TIME$(n^{c''''})$ for some $c'''' > 0$ by Lemma 11. Let $c^{\mathrm{v}} = \max(c'', c''', c'''')$. Using Lemma 12, we have:

$$\begin{aligned}
&\Sigma_{j+1}\text{-TQBF} \in \Sigma_i\text{-TIME}(n^{c^{\mathrm{v}}}) \wedge \Sigma_{j+1}\text{-TQBF} \in \Pi_i\text{-TIME}(n^{c^{\mathrm{v}}}) \wedge \\
&\Pi_{j+1}\text{-TQBF} \in \Sigma_i\text{-TIME}(n^{c^{\mathrm{v}}}) \wedge \Pi_{j+1}\text{-TQBF} \in \Pi_i\text{-TIME}(n^{c^{\mathrm{v}}}),
\end{aligned}$$

as we wanted. □

5.3 Bounded Self-reducibility of SAT

In the same fashion, we adapt the Self-reducibility of SAT Lemma. As we can see, the resulting proof is very similar to the original.

Lemma 14 (Bounded Self-reducibility of SAT)**.** *For all $c > 0$, there exists $c' > 0$ such that:*

$$T_2 \vdash \ \mathrm{SAT} \in \mathrm{SIZE}(n^c) \ \rightarrow \quad \begin{array}{l} \forall n \in \mathrm{Log}_{>1}\ \exists C' \in \{0,1\}^{8n^{c'}|n^{c'}|} \forall x \in \{0,1\}^n \forall u \in \{0,1\}^n \\ (\mathrm{size}(C') \leq c'n^{c'} \wedge (\mathrm{evalbf}(x,u) \rightarrow \mathrm{evalbf}(x, C'(x)))) \end{array}$$

Proof. Fix $c > 0$. We reason in T_2.

Assume SAT $\in$ SIZE(n^c), and fix $n \in \mathrm{Log}_{>1}$. Let k and C be as asserted by SAT $\in$ SIZE(n^c). Then, let C' represent a circuit implementing the following algorithm: on input $x \in \{0,1\}^n$, it checks x represents a propositional formula $\varphi(u_1, \ldots, u_m)$, with exactly m free variables ($u_1, \ldots, u_m$ represent single variables). After that, the circuit sets $\varphi_0 = \varphi$ and, for $i = 1, \ldots, m$, does the following:

> Set φ_i to be as φ_{i-1} but with $u_i = 1$.
>
> If $C(\varphi_i) \doteq 1$, set $y_i = 1$.
>
> Else, set φ_i to be as φ_{i-1} but with $u_i = 0$ and set $y_i = 0$.

Finally, the circuit represented by C' outputs $y_1, \ldots, y_m$.

For every $x \in \{0,1\}^n$ representing a quantified formula $\varphi(u_1, \ldots, u_m)$, the algorithm described iterates $m \leq n$ times and in each iteration it computes φ_i and sets y_i to 0 or 1, and also evaluates the circuit represented by C on input φ_i, all of which can be computed by polynomial-sized circuits. Then, there must be c' such that the size of C' is bounded by $c'n^{c'}$. This proves $\mathrm{size}(C') \leq c'n^{c'}$.

To argue that for all $x \in \{0,1\}^n$ and all $u \in \{0,1\}^n$ we have that $\mathrm{evalbf}(x,u)$ implies $\mathrm{evalbf}(x, C'(x))$, we fix $x \in \{0,1\}^n$ and $u \in \{0,1\}^n$ such that $\mathrm{evalbf}(x,u)$ holds and prove that $\mathrm{evalbf}(x, C'(x))$ holds. Since $\mathrm{evalbf}(x,u)$ holds, in particular x represents a propositional formula $\varphi(z_1, \ldots, z_m)$ with $m \leq n$ variables $z_1, \ldots, z_m$. We prove the following statement by length Σ^b_1-induction on $i \leq n$: if $v_i \in \{0,1\}^i$ represents the assignment constructed by the algorithm after i iterations on input φ, then

$$\exists w \in \{0,1\}^{n-i}(\mathrm{evalbf}(\mathrm{subs}(\varphi, v_i), w)).$$

Note that φ and v_i are the outputs of corresponding PV-functions on input (C', x, i) and hence the statement we want to prove is a Σ^b_1-statement of i, with C' and x as parameters. In the following, let φ_i abbreviate $\mathrm{subs}(\varphi, v_i)$.

The base case $i = 0$ of the induction is obvious since v_0 is the empty string, so $\varphi_0 = \text{subs}(\varphi, v_0) = \varphi = x$, and we can take $w = u$ to make $\text{evalbf}(\varphi_0, w)$ hold. For the inductive case, fix $i < n$ and assume that there exists $w \in \{0,1\}^{n-i}$ such that $\text{evalbf}(\varphi_i, w)$ holds. We need to find $w' \in \{0,1\}^{n-i-1}$ such that $\text{evalbf}(\varphi_{i+1}, w')$ holds. If $i \geq m$, then φ_i does not have any variables and then $\varphi_{i+1} = \varphi_i$ so we can let w' be any string, say 1^{n-i-1}, and we have $\text{evalbf}(\varphi_{i+1}, w')$ from the assumption that $\text{evalbf}(\varphi_i, w)$ holds. If $i < m$, then from the assumption that $\text{evalbf}(\varphi_i, w)$ holds we have that φ_i is satisfiable, and from the assumption that C solves SAT we have that either $C(\text{subs}(\varphi, v_i 1)) = 1$ and then $v_{i+1} = v_i 1$ and φ_{i+1} is satisfiable, or else $C(\text{subs}(\varphi, v_i 0)) = 1$ and then $v_{i+1} = v_i 0$ and φ_{i+1} is again satisfiable. In either case there exists $w' \in \{0,1\}^{n-i-1}$ such that $\text{evalbf}(\varphi_{i+1}, w')$ as was to be shown. To conclude the proof note that $v_n = v_m = C'(x)$ and $x = \varphi$, so $\text{evalbf}(x, C'(x))$ holds. □

5.4 Bounded Karp-Lipton

And finally, we are in position of adapting the Karp-Lipton Theorem to Bounded Arithmetic.

Theorem 8 (Bounded Karp-Lipton)**.** *For every $c > 0$ and every $i > 1$, there exists $c' > 0$ such that:*

$$T_2 \vdash \text{SAT} \in \text{SIZE}(n^c) \to \Sigma_i\text{-TQBF} \in \Sigma_2\text{-TIME}(n^{c'}) \wedge \Sigma_i\text{-TQBF} \in \Pi_2\text{-TIME}(n^{c'}).$$

Proof. Fix $c > 0$ and $i > 1$, and let $c'' > 0$ be as asserted by Lemma 14. We reason in T_2.

Assume $\text{SAT} \in \text{SIZE}(n^c)$. By Lemma 11 and Lemma 13, we only need to show $\Pi_2\text{-TQBF} \in \Sigma_2\text{-TIME}(n^{c'})$ for some $c' > 0$. Let M be a TM that works as follows: on input $\langle x, C, v\rangle$, checks x represents a propositional Π_2-sentence $\forall \overline{u}_1 \exists \overline{u}_2 \varphi(\overline{u}_1, \overline{u}_2)$. Let m_1 be the length of $\overline{u}_1$ and y represent a string of length m which represents the quantified formula $\varphi(v', \overline{u}_2)$, where v' represents the string formed by the first m_1 bits of v. Then, M checks C represents a m-input circuit of size $m^{c''}$, and runs the circuit represented by C on input y. Afterwards, it checks $C(y)$ is a m_2-length string (call it w), where m_2 is the length of $\overline{u}_2$. Finally, M accepts iff $\varphi(v', w)$ is true.

Since the size of C is polynomially bounded, every operation performed by M can be executed in polynomial time, so that M halts on kn^k time for some $k > 0$. Take $c' \geq k$ long enough so that $n^{c'} \geq 8n^{c''}|n^{c''}|$ for all $n > 1$, and fix $n \in \text{Log}_{>1}$ and $x \in \{0,1\}^n$. By Lemma 14, we

have:

$$x \in \Pi_2\text{-TQBF} \leftrightarrow \exists C \in \{0,1\}^{n^{c'}} \forall v \in \{0,1\}^{n^{c'}} M(x, C, v, 1^{c' n^{c'}}) \doteq 1,$$

so that M, $k = c'$, and $k = n_0 = 1$ witness Π_2-TQBF $\in \Sigma_2$-TIME$(n^{c'})$, as we wanted. $\square$

Chapter 6

Bounded Kannan

In this chapter, we formalize the adapted proof of Kannan's result that we developed in Section 4.3. First, we will define some new L_{BA}-formulas, and then we will formalize every result from Section 4.3 except Theorem 7. As previously announced, the formalization of this last theorem requires the use of second order bounded arithmetic, which is out of the scope of this work.

6.1 Definitions

Let us introduce some new definitions. First, we define some functions:

$\mathrm{bin}(x) \stackrel{def}{=}$ if x represents a binary string, return the number represented by x in binary (ignoring left zeros); otherwise, return 0.

$\mathrm{str}(i, n) \stackrel{def}{=}$ if $i < 2^n$, return the binary string of length n that represents the number i in binary; otherwise, return 0.

$\mathrm{first}(p) \stackrel{def}{=}$ if p represents a pair $\langle x, y\rangle$, returns x, the first element of the pair; otherwise, returns 0.

$\mathrm{second}(p) \stackrel{def}{=}$ if p represents a pair $\langle x, y\rangle$, returns y, the second element of the pair; otherwise, returns 0.

Note if we quantify over strings of a fixed length, bin() becomes injective. Then (for each $c > 0$) we define the following predicates:

$\text{computesptt}_c(N, C, S) \stackrel{def}{=}$ "N represents a string of length n, we have that $S \in \{0,1\}^{n^c}$ and C represents an n-input circuit, and for all $i = 0, \ldots, n^c - 1$, we have $C(\text{str}(i, n)) \doteq (S)_i$".

$\text{smallereqptt}_c(N, S, S') \stackrel{def}{=}$ "N represents a string of length n, we have that $S, S' \in \{0,1\}^{n^c}$, and for all $i = 0, \ldots, n^c - 1$, we have that either $(S)_i \leq (S')_i$, or for some $j \in \{i+1, \ldots, n^c - 1\}$ we have $(S)_j < (S')_j$".

Finally, we define the following formulas for every $n > 0$ and every x representing a string of length n:

$$\begin{aligned} x \in L^c \stackrel{def}{=}\ & \exists S{\in}\{0,1\}^{n^{5c}} \forall C{\in}\{0,1\}^{n^{5c}} \forall S'{\in}\{0,1\}^{n^{5c}} \exists C'{\in}\{0,1\}^{n^{5c}} \\ & n \geq 16 \wedge n \geq 5c \wedge ((\text{size}(C) \leq n^{c+1} \rightarrow \neg\text{computesptt}_{5c}(x, C, S)) \\ & \wedge((\text{size}(C') \leq n^{c+1} \wedge \text{computesptt}_{5c}(x, C', S'))\vee \\ & \text{smallereqptt}_{5c}(x, S, S')) \wedge (S)_{\text{bin}(x)} = 1) \end{aligned}$$

$$x \in L \oplus L' \stackrel{def}{=} (\text{second}(x) = 0 \wedge \text{first}(x) \in L) \vee (\text{second}(x) = 1 \wedge \text{first}(x) \in L')$$

$$\begin{aligned} A \leq_{\text{m}}^{\text{SIZE}(n^d)} B \stackrel{def}{=}\ & \exists k \in \text{Log}_{>1}\ \exists n_0 \in \text{Log}_{>1}\ \forall n \in \text{Log}_{>1}\ \exists C \in \{0,1\}^{8kn^d|kn^d|} \forall x \in \{0,1\}^n \\ & (\text{size}(C) \leq kn^d) \wedge (x \in A \leftrightarrow C(x) \in B)). \end{aligned}$$

Again, since all the previous formulas are built from predicates and formulas computable in polynomial time, they are expressible by L_{BA}-formulas. Note, in particular, $x \in L^c$ is defined by a Σ_3^{b} L_{BA}-formula. Also note for all $n \geq \max(16, 5c)$ we have $n^{5c} \leq 2^n$, so that this inequality holds in the definition of $x \in L^c$.

6.2 $L^c \notin \text{SIZE}(n^c)$

The following lemma shows T_2 proves for every $c > 0$ there is a lexicographically minimal n^{5c}-ptt not computed by any circuit of size at most n^{c+1}. Notice for every $c > 0$, in the following statement the bounds of the circuits are smaller than the corresponding bounds in the definition of $x \in L^c$. This doesn't matter because of our choice of n and the inequalities on (4.2), and we need a smaller bound to apply the Weak Pigeonhole Principle.

Lemma 15. *For every $c > 0$, we have:*

$$
\begin{aligned}
T_2 \vdash \quad & \exists n_0 \in \mathrm{Log}_{>1}\ \forall n \in \mathrm{Log}_{>1}\ n > n_0 \to \exists S \in \{0,1\}^{n^{5c}} \\
& (\forall C \in \{0,1\}^{8n^{2c+2}} (\mathrm{size}(C) \leq n^{c+1} \to \neg \mathrm{computesptt}_{5c}(1^n, C, S)) \\
& \wedge\ \forall S' \in \{0,1\}^{n^{5c}} \exists C' \in \{0,1\}^{8n^{2c+2}} \\
& (((\mathrm{size}(C') \leq n^{c+1} \wedge \mathrm{computesptt}_{5c}(1^n, C', S')) \vee \mathrm{smallereqptt}_{5c}(1^n, S, S')))).
\end{aligned}
$$

Proof. For this proof, we will use the Weak Pigeonhole Principle and the Minimization Principle defined in Chapter 2, (definitions 48 and 49). Fix $c > 0$. We will work in T_2. Let $n_0 = 16$, and fix $n \in \mathrm{Log}_{>1}$ such that $n > n_0$: first, we will prove the first part of the remaining conjunction (including the existence of S), and then we will use it to prove the whole statement.

To prove the first part of the conjunction, we will assume that the following formula is true towards contradiction:

$$
\forall S \in \{0,1\}^{n^{5c}} \exists C \in \{0,1\}^{8n^{2c+2}} (\mathrm{size}(C) \leq n^{c+1} \wedge \mathrm{computesptt}_{5c}(1^n, C, S)). \tag{6.1}
$$

Consider the following formula:

$$
\varphi(\overline{x}, \overline{y}) \stackrel{def}{=} \ \mathrm{size}(\mathrm{str}(\overline{y}, 8n^{2c+2})) \leq n^{c+1} \wedge \mathrm{computesptt}_{5c}(1^n, \mathrm{str}(\overline{y}, 8n^{2c+2}), \mathrm{str}(\overline{x}, n^{5c})),
$$

where we use n as a parameter of φ. Since φ is quantifier-free, we have $\varphi \in \Sigma^{\mathrm{b}}_0$, so that it is a bounded formula. Then, by Lemma 3, we have $\mathrm{WPHP}(M, N, \varphi)$ for all M, N:

$$
\begin{aligned}
\mathrm{WPHP}(M, N, \varphi) \ = \ & M \geq N^2 \wedge \forall x < M\ \exists y < N\ \varphi(x, y) \to \\
& \exists x < M\ \exists x' < M\ \exists y < N\ (x \neq x' \wedge \varphi(x, y) \wedge \varphi(x', y)).
\end{aligned} \tag{6.2}
$$

Choose $M = 2^{n^{5c}}$ and $N = 2^{8n^{2c+2}}$. Since $n > n_0$, we have $16n^{2c+2} < n^{5c}$ holds, which entails $M \geq N^2$. Furthermore, (6.1) shows $\forall x < M\ \exists y < N\ \varphi(x, y)$. But then, using (6.2) and modus ponens, we get the following formula:

$$
\exists x < M\ \exists x' < M\ \exists y < N\ (x \neq x' \wedge \varphi(x, y) \wedge \varphi(x', y)). \tag{6.3}
$$

Let $x, x' \in \{0,1\}^{n^{5c}}$ and $y \in \{0,1\}^{8n^{2c+2}}$ be as witnessed by (6.3). We have $x \neq x'$, so there must be a bit where the two strings differ, call it the i-th bit: that is, we have $(x)_i \neq (x')_i$. And we also have $\varphi(x, y)$ and $\varphi(x', y)$, so that in particular we have $\mathrm{computesptt}_{5c}(1^{n_1}, y, x)$ and $\mathrm{computesptt}_{5c}(1^{n_1}, y, x')$ for some $n_1 < n^{5c}$, which in particular implies $y(\mathrm{str}(i)) = (x)_i$ and $y(\mathrm{str}(i)) = (x')_i$, and we get $(x)_i = (x')_i$, against the choice of i. This contradiction

proves that the assumption was false.

To prove the second part of the conjunction, we will use the Minimization Principle and the fact we have proven:

$$\exists S \in \{0,1\}^{n^{5c}} \forall C \in \{0,1\}^{8n^{2c+2}} (\mathrm{size}(C) \leq n^{c+1} \to \neg \mathrm{computesptt}_{5c}(1^n, C, S)). \tag{6.4}$$

Consider the following formula:

$$\psi(\overline{x}) \stackrel{def}{=} \ \forall C \in \{0,1\}^{8n^{2c+2}} (\mathrm{size}(C) \leq n^{c+1} \to \neg \mathrm{computesptt}_{5c}(1^n, C, \mathrm{str}(\overline{x}, n^{5c}))),$$

where we use n as a parameter of ψ. Since ψ has only a universal quantifier, it is a bounded formula. Then, by Lemma 4, we have $\mathrm{MIN}(M, \psi)$ for every M:

$$\mathrm{MIN}(M, \psi) = \exists x < M \ \psi(x) \to \exists x < M \ \forall z < x \ (\psi(x) \wedge \neg \psi(z)). \tag{6.5}$$

Choose $M = 2^{n^{5c}}$. We have (6.4) shows $\exists x < M \ \psi(x)$, so using (6.5) and modus ponens we can prove the following formula:

$$\exists x < M \ \forall z < x \ (\psi(x) \wedge \neg \psi(z)), \tag{6.6}$$

meaning there exists a n^{5c}-ptt not computed by any circuit of size at most n^{c+1} which is smaller or equal than any n^{5c}-ptt not computed by any circuit of size at most n^{c+1}. From (6.6), the whole statement follows. □

Theorem 9. *For all $c > 0$, there exists $c' > 0$ such that we have:*

$$T_2 \vdash L^c \in \Sigma_3\text{-TIME}(n^{c'}) \wedge \ L^c \notin \mathrm{SIZE}(n^c).$$

Proof. Fix $c > 0$. We work in T_2.

Since $x \in L^c$ is defined by a Σ^{b}_3 L_{BA}-formula, we know there exists $c' > 0$ such that $L^c \in \Sigma_3\text{-TIME}(n^{c'})$ by Lemma 9.

Then, suppose $L^c \in \mathrm{SIZE}(n^c)$ towards contradiction, and let $k, n_0' \in \mathrm{Log}_{>1}$ be as witnessed by it. Let n_0 be as witnessed by Lemma 15, and let $n > \max(n_0, n_0', 16, 5c, k)$. By the choice of n_0, there exists $S \in \{0,1\}^{n^{5c}}$ representing the lexicographically minimal n^{5c}-ptt not computed by any circuit of size at most n^{c+1}. Let $C \in \{0,1\}^{8kn^c|kn^c|}$ be as witnessed by $L^c \in \mathrm{SIZE}(n^c)$ given n. Since $n \geq k$, we have $\mathrm{size}(C) \leq kn^c \leq n^{c+1}$ so that $C(y) \neq (S)_{\mathrm{bin}(y)}$ holds for some $y \in \{0,1\}^n$ such that $\mathrm{bin}(y) < n^{5c}$. Fix that y. We will show this leads to contradiction by case distinction.

Case $y \in L^c$: by choice of n_0', we have $C(y) \doteq 1$, and by the definition of $y \in L^c$ and the fact there can only be a single lexicographically minimal n^{5c}-ptt not computed by any circuit of size at most n^{c+1}, we get $(S)_{\mathrm{bin}(y)} = 1$. But then, $C(y) \doteq (S)_{\mathrm{bin}(y)}$: contradiction.

Case $y \notin L^c$: by choice of n_0', we have $C(y) \doteq 0$. Since S does indeed represent the lexicographically minimal n^{5c}-ptt not computed by any circuit of size at most n^{c+1}, for $y \notin L^c$ to hold it must happen that $(S)_{\mathrm{bin}(y)} = 0$. But then, $C(y) \doteq (S)_{\mathrm{bin}(y)}$, a contradiction.

□

6.3 $K^c \notin \mathrm{SIZE}(n^c)$

Before defining K^c, we will prove an auxiliary lemma:

Lemma 16. *Let A and B be languages and let $c, d > 0$. We have:*

$$T_2 \vdash A \leq_m^{\mathrm{SIZE}(n^d)} B \wedge B \in \mathrm{SIZE}(n^c) \to A \in \mathrm{SIZE}(n^{\max(c,d)}).$$

Proof. Let A, B be languages, and fix $c, d > 0$. We work in T_2.

Assume $A \leq_{\mathrm{m}}^{\mathrm{SIZE}(n^d)} B$ and let k and n_0 be as witnessed by it. Assume $B \in \mathrm{SIZE}(n^c)$ and let k' and n_0' be as witnessed by it. Let $n_0'' = \max(n_0, n_0')$ and fix $n > n_0''$. By choice of k, there exists $C \in \{0,1\}^{8kn^d|kn^d|}$ such that $\mathrm{size}(C) \leq kn^d$ and for every $x \in \{0,1\}^n$, we have

$$x \in A \leftrightarrow C(x) \in B.$$

Similarly, by choice of k', there exists $C' \in \{0,1\}^{8k'n^c|k'n^c|}$ such that $\mathrm{size}(C') \leq k'n^c$ and for every $x \in \{0,1\}^n$, we have

$$x \in B \leftrightarrow C'(x) \doteq 1.$$

Let $k'' = k + k'$, and let $C'' \in \{0,1\}^{8k''n^{\max(c,d)}|k''n^{\max(c,d)}|}$ represent the circuit obtained by composing C with C': the output of C is fed into the input of C'. We have $\mathrm{size}(C'') \leq k''n^{\max(c,d)}$, and for every $x \in \{0,1\}^n$, we get:

$$x \in A \leftrightarrow C''(x) \doteq 1.$$

Since $n > n_0''$ was arbitrary, k'' and n_0'' witness $A \in \mathrm{SIZE}(n^{\max(c,d)})$. □

As we will see, the definition of K^c for a given $c > 0$ in bounded arithmetic is very similar to the definition we gave in Chapter 4.

Fix $c > 0$. By Lemma 9, we know $L^c \in \Sigma_3\text{-TIME}(n^{c'})$ for some $c' > 0$; for convenience, define $d_c = c'$. Let M_c be the TM witnessing $L^c \in \Sigma_3\text{-TIME}(n^{d_c})$. Let f be the polynomial-time computable function guaranteed by Theorem 3. For $n > 0$, we define:

$$\varphi_n^c(\overline{x}_1, \overline{x}_2, \overline{x}_3, \overline{x}_4, \overline{z}) \stackrel{def}{=} \quad f(M_c, 1^n, 1^{n^{d_c}}, 1^{n^{d_c}}, 1^{n^{d_c}}, 1^{d_c n^{d_c}}).$$

For every $n \in \text{Log}_{>1}$, every x representing a string of length n and all $v_1, v_2, v_3 \in \{0,1\}^{n^{d_c}}$, we have:

$$M_c(x, v_1, v_2, v_3, 1^{d_c n^{d_c}}) \doteq 1 \iff \exists z \in \{0,1\}^l \varphi_n^c(\overline{x}_1/x, \overline{x}_2/v_1, \overline{x}_3/v_2, \overline{x}_4/v_3, \overline{z}/z), \tag{6.7}$$

where l is the length of $\overline{z}$. Note since for any given $n > 0$ we have φ_n^c is a propositional formula, it can be expressed by a QF L_{BA}-formula; from now on, we will assume φ_n^c represents an L_{BA}-formula. Let $c' > 0$ be as in the statement of Lemma 14. Let a_c be such that for every $n \in Log_{>}1$, if m_n^c is the length of the string that represents φ_n^c, then

$$T_2 \vdash 8(m_n^c)^{c'} |(m_n^c)^{c'}| \leq a_c n^{a_c}. \tag{6.8}$$

Then, we can define, for every n and every x representing a string of length n:

$$\begin{aligned} x \in K_1^c \stackrel{def}{=} \quad & \exists D \in \{0,1\}^{a_c n^{a_c}} \exists v_1 \in \{0,1\}^{n^{d_c}} \forall v_2 \in \{0,1\}^{n^{d_c}} \\ & \varphi_n^c(x, v_1, v_2, D(\varphi_n^c(x, v_1, v_2, \overline{x}_4, \overline{z}))). \end{aligned} \tag{6.9}$$

Which allows us to define:

$$x \in K^c \stackrel{def}{=} \quad x \in K_1^c \oplus \text{SAT}.$$

And finally, the formalization of Kannan's result.

Theorem 10. *For all $c > 0$, there exists $c' > 0$ such that we have:*

$$T_2 \vdash K^c \in \Sigma_2\text{-TIME}(n^{c'}) \wedge \ K^c \notin \text{SIZE}(n^c).$$

Proof. Fix $c > 0$. We work in T_2.

Since $x \in K_1^c$ is defined by an Σ_2^{b} L_{BA}-formula (see 6.9) and $x \in \text{SAT}$ is defined by a Σ_1^{b} L_{BA}-formula, we have K^c is defined by a Σ_2^{b} L_{BA}-formula, so that there exists $c' > 0$ such that $K^c \in \Sigma_2\text{-TIME}(n^{c'})$ by Lemma 9. We will show $K^c \notin \text{SIZE}(n^c)$ by case distinction.

Case SAT $\notin$ SIZE(n^c). Assume $K^c \in$ SIZE(n^c) towards a contradiction. Fix $n \in \text{Log}_{>1}$. Consider a circuit that on input $x \in \{0,1\}^n$ returns $\langle x, 0\rangle$: such a circuit can be built to have size at most 3 times its input length, so let $C \in \{0,1\}^{8 \cdot 3n|3n|}$ represent the described circuit with size(C) $\leq 3n$. For all $x \in \{0,1\}^n$, we have $x \in \text{SAT} \leftrightarrow C(x) \in K^c$. Since $n \in \text{Log}_{>1}$ was arbitrary, $k = 3$ and $n_0 = 0$ witness SAT $\leq_{\text{m}}^{\text{SIZE}(n)} K^c$. We get SAT $\in$ SIZE(n^c) by Lemma 16, in contradiction with the assumption.

Case SAT $\in$ SIZE(n^c). In this case, we will see $x \in L^c \leftrightarrow x \in K_1^c$ for every $n \in \text{Log}_{>1}$ and every $x \in \{0,1\}^n$. Fix $n \in \text{Log}_{>1}$ and $x \in \{0,1\}^n$:

$\boxed{x \in K_1^c \rightarrow x \in L^c}$

Let $x \in K_1^c$. By (6.9), we have some $D \in \{0,1\}^{a_c n^{a_c}}$ and some $v_1 \in \{0,1\}^{n^{d_c}}$ such that for every $v_2 \in \{0,1\}^{n^{d_c}}$, we have:

$$\varphi_n^c(x, v_1, v_2, D(\varphi_n^c(x, v_1, v_2, \overline{x}_4, \overline{z}))). \tag{6.10}$$

Fix $v_2 \in \{0,1\}^{n^{d_c}}$ and let v_3 represent the first n^{d_c} bits of $D(\varphi_n^c(x, v_1, v_2, \overline{x}_4, \overline{z}))$. Using (6.7) and (6.10), we get:

$$M_c(x, v_1, v_2, v_3, 1^{d_c n^{d_c}}) = 1,$$

so that, by the choice of M_c, we get v_1 witnesses $x \in L^c$.

$\boxed{x \in L^c \rightarrow x \in K_1^c}$

Since SAT $\in$ SIZE(n^c), let $c' > 0$ be as guaranteed by Lemma 14, so that there is some $D \in \{0,1\}^{a_c n^{a_c}}$ representing a m_n^c-input circuit that outputs a satisfying assignment on input a satisfiable formula, for m_n^c the length of φ_n^c, and a_c has the property (6.8). Now, let $x \in L^c$, and let $v_1 \in \{0,1\}^{n^{d_c}}$ be as guaranteed by $L^c \in \Sigma_3$-TIME(n^c). Fix $v_2 \in \{0,1\}^{n^{d_c}}$: we have

$$\exists v_3 \in \{0,1\}^{n^{d_c}} M_c(x, v_1, v_2, v_3, 1^{d_c^{d_c}}) = 1,$$

so that, by (6.7), we get

$$\exists v_3 \in \{0,1\}^{d_c} \exists z \in \{0,1\}^l \varphi_n^c(x, v_1, v_2, v_3, z),$$

which, by definition of D, implies

$$\varphi_n^c(x, v_1, v_2, D(\varphi_n^c(x, v_1, v_2, \overline{x}_4, \overline{z}))),$$

which, by (6.9), means D and v_1 witness $x \in K_1^c$.

To complete the argument, assume $K^c \in \mathrm{SIZE}(n^c)$ towards a contradiction. Fix $n \in \mathrm{Log}_{>1}$. Consider a circuit that on input $x \in \{0,1\}^n$ returns $\langle x, 1\rangle$: such a circuit can be built to have size at most 3 times its input length, so let $C' \in \{0,1\}^{8 \cdot 3n|3n|}$ represent the described circuit with $\mathrm{size}(C') \leq 3n$. For all $x \in \{0,1\}^n$, we have:

$$x \in L^c \leftrightarrow x \in K_1^c \leftrightarrow C(x) \in K^c.$$

Since $n \in \mathrm{Log}_{>1}$ was arbitrary, $k = 3$ and $n_0 = 0$ witness $L^c \leq_{\mathrm{m}}^{\mathrm{SIZE}(n)} K^c$. We get $L^c \in \mathrm{SIZE}(n^c)$ by Lemma 16, which contradicts Theorem 9.

□

Bibliography

[AB09] S. Arora and B. Barak. *Computational complexity: a modern approach*. Cambridge University Press, 2009.

[ABM23] A. Atserias, S. Buss, and M. Müller. "On the Consistency of Circuit Lower Bounds for Non-Deterministic Time". In: *arXiv preprint arXiv:2303.01016* (2023).

[BDG95] J.L. Balcazar, J. Diaz, and J. Gabarro. *Structural Complexity I*. Second Edition. Springer-Verlag, Berlin, 1995.

[Bus86] S. R. Buss. *Bounded Arithmetic*. University of California, 1986.

[CN10] S. Cook and P. Nguyen. *Logical foundations of proof complexity*. Vol. 11. Cambridge University Press, 2010.

[Kan82] R. Kannan. "Circuit-size lower bounds and non-reducibility to sparse sets". In: *Information and control* 55.1-3 (1982), pp. 40–56.

[KL80] R. M. Karp and R. J. Lipton. "Some connections between nonuniform and uniform complexity classes". In: *Proceedings of the twelfth annual ACM symposium on Theory of computing*. 1980, pp. 302–309.

[KO17] J. Krajicek and I. C. Oliveira. "Unprovability of circuit upper bounds in Cook's theory PV". In: *Logical Methods in Computer Science* 13 (2017).

[Kra95] J. Krajíček. *Bounded arithmetic, propositional logic and complexity theory*. Cambridge University Press, 1995.

[Kra19] J. Krajíček. *Proof complexity*. Cambridge University Press, 2019.

[MPW02] A. Maciel, T. Pitassi, and A.R. Woods. "A New Proof of the Weak Pigeonhole Principle". In: *Journal of Computer and System Sciences* 64.4 (2002), pp. 843–872.

[Vol99] Heribert Vollmer. *Introduction to circuit complexity: a uniform approach*. Springer Science & Business Media, 1999.