1

Introduction

Ten years ago I had the wonderful opportunity to attend a series of lectures given by Jeff Paris in Prague on his and Alec Wilkie's work on bounded arithmetic and its relations to complexity theory. Their work produced fundamental information about the strength and properties of these weak systems, and they developed a variety of basic methods and extracted inspiring problems.

At that time Pavel Pudlák studied sequential theories and proved interesting results about the finitistic consistency statements and interpretability (Pudlák 1985,1986,1987). A couple of years later Sam Buss's Ph.D. thesis (Buss 1986) came out with an elegant proof-theoretic characterization of the polynomial time computations. Then I learned about Cook (1975), predating the above developments and containing fundamental ideas about the relation of weak systems of arithmetic, propositional logic, and feasible computations. These ideas were developed already in the late 70s by some of his students but unfortunately remained, to a large extent, unavailable to a general audience. New connections and opportunities opened up with Miki Ajtai's entrance with powerful combinatorics applied earlier in Boolean complexity (Ajtai 1988).

The work of these people attracted other researchers and allowed, quite recently, further fundamental results.

It appears to me that with a growing interest in the field a text surveying some basic knowledge could be helpful. The following is an outline of the book.

Chapter 2 lists notions and results from logic and complexity theory the reader is expected to have heard about. Chapter 3 overviews basic Boolean complexity and basic facts about predicates definable by bounded arithmetic formulas. Sketches of a few proofs are offered there, but mostly I refer the reader to other survey texts. All later chapters contain all necessary proofs.

Chapters 4 and 5 present basic information about the main propositional proof systems and complexity of proofs issues, and about the main first order systems of bounded arithmetic and their strength and mutual relations.

Chapters 6 and 7 survey the characterizations of definable functions in the systems of arithmetic known as *witnessing theorems*. Chapter 8 treats the second order systems of bounded arithmetic using *RSUV isomorphism* and transfers some results of the previous three chapters to these systems.

Chapter 9 defines and studies propositional translations of arithmetic formulas and propositional simulations of arithmetic proofs with applications to polynomial simulation results.

Chapter 10 is devoted to the fundamental problem of whether bounded arithmetic S_2 is finitely axiomatizable, the central problem in the area. It surveys all relevant results known (to me) to date.

Chapter 11 studies direct combinatorial arguments allowing separation of the lowest relativized subsystems of bounded arithmetic.

Chapters 12 and 13 concern the central question of propositional logic, whether there is a proof system admitting polynomial size proofs of all tautologies, for *Frege* and extended *Frege* systems and for constant-depth systems. The main results are several exponential lower bounds for the constant-depth systems and a certain conceptual framework for the unrestricted system.

Chapter 14 presents finitistic consistency statements and studies the issue of hard tautologies and optimal proof systems.

The final chapter, Chapter 15, develops some combinatorics and Boolean complexity theory within bounded arithmetic and studies several model-theoretic constructions relevant to all the basic questions studied earlier in the book.

I have made an attempt to present the chosen material as completely and as up to date as possible but I did not try to compile a handbook of the whole field (hence the Bibliography also does not attempt to list the whole literature in the field). Open problems are occasionally mentioned in the text (see the Index), but I refer the reader to Clote and Krajíček (1993) for a comprehensive annotated list of open problems in the area.

In the main text I give explicit credit only for main ideas and results. The chapters end with a section of bibliographical and other remarks where complete bibliographical information is given, and where I briefly comment on related but not covered topics.

Finally I want to comment on material that is *not* covered in the book. This includes, in particular, intuitionistic versions of bounded arithmetic systems, functional interpretations of these theories (and the issue of feasible functionals in general), equational theories and machine-independent characterizations of various computational classes, and modifications of the basic systems relating them to a variety of subclasses of the polynomial time. Some of this material is omitted as I do not feel familiar with it; some is omitted because I think that it – although technically difficult and innovative – builds on basic ideas already apparent from earlier results presented in the book.

2

Preliminaries

In this chapter we briefly review the basic notions and facts from logic and complexity theory whose knowledge is assumed throughout the book. We shall always sketch important arguments, both from logic and from complexity theory, and so a determined reader can start with only a rough familiarity with the notions surveyed in the next two sections and pick the necessary material along the way.

For those readers who prefer to consult relevant textbooks we recommend the following books: The best introduction to logic are parts of Shoenfield (1967); for elements of structural complexity theory I recommend Balcalzár, Diáz, and Gabbarró (1988, 1990); for NP-completeness Garey and Johnson (1979); and for a Boolean complexity theory survey of lower bounds Boppana and Sipser (1990) or the comprehensive monograph Wegener (1987). A more advanced (but self-contained) text on logic of first order arithmetic theories is Hájek and Pudlák (1993).

2.1. Logic

We shall deal with first order and second order theories of arithmetic. The second order theories are, in fact, just two-sorted first order theories: One sort are numbers; the other are finite sets. This phrase means that the underlying logic is always the first order predicate calculus; in particular, no set-theoretic assumptions are a part of the underlying logic.

From basic theorems we shall use Gödel *completeness* and *incompleteness* theorems, Tarski's *undefinability of truth*, and, in arithmetic, constructions of *partial truth definitions*.

A prominent theory is *Peano arithmetic* (PA), in the *language of arithmetic* $L_{PA} = \{0, 1, +, \cdot, <, =\}$ axiomatized by *Robinson's arithmetic* Q

1. $a + 1 \neq 0$

2. $a + 1 = b + 1 \rightarrow a = b$
3. $a + 0 = a$
4. $a + (b + 1) = (a + b) + 1$
5. $a \cdot 0 = 0$
6. $a \cdot (b + 1) = (a \cdot b) + a$
7. $a \neq 0 \rightarrow \exists x, x + 1 = a$

see Tarski, Mostowski, and Robinson (1953), and by the *induction scheme* IND

$$\big(\phi(0, \overline{a}) \wedge \forall x(\phi(x, \overline{a}) \rightarrow \phi(x + 1, \overline{a}))\big) \rightarrow \forall x \phi(x, \overline{a})$$

for every formula $\phi(x, \overline{a})$ in the language L_{PA}.

We shall use the letters $x, y, z, \ldots$ mostly for bounded variables; the letters $a, b, c, \ldots$ will be reserved for free variables (also called parameters). Free variables in axioms are assumed to be universally quantified; for example, the first axiom given is equivalent to the formula $\forall x, x + 1 \neq 0$.

There are other schemes that can equivalently replace the induction scheme, for example, *the least number principle* LNP scheme

$$\phi(b, \overline{a}) \rightarrow \exists x \forall y\big(\phi(x, \overline{a}) \wedge (y < x \rightarrow \neg\phi(y, \overline{a}))\big).$$

The *standard model* $\boldsymbol{N}$ of PA is the set of natural numbers with the symbols of L_{PA} interpreted with the usual meaning. A crucial fact about PA is that there are *nonstandard* models (models not isomorphic with $\boldsymbol{N}$) of PA and indeed of the theory of $\boldsymbol{N}$, $\mathrm{Th}(N)$. Natural numbers $\boldsymbol{N}$ are isomorphic to a unique initial substructure of any nonstandard model M and we shall usually simply assume that $\boldsymbol{N} \subset M$.

A *cut* in a nonstandard model M is any nonempty $I \subseteq M$ satisfying

1. $a < b \wedge b \in I \rightarrow a \in I$, all $a, b \in M$
2. $a \in I \rightarrow a + 1 \in I$, all $a \in M$.

For example, $\boldsymbol{N}$ is a cut in every nonstandard model. Cuts in nonstandard models of PA closed under both addition and multiplication have special prominence as they are particular models of *bounded arithmetic* $I\Delta_0$: They satisfy induction for all *bounded arithmetic formulas* Δ_0, which are formulas in the language L_{PA} with all quantifiers bounded (Section 3.2 is devoted to bounded formulas).

Nonstandard models of PA and even of its proper subtheories are difficult to construct; it is a theorem of Tennenbaum (1959) that there are no countable recursive nonstandard models of PA (and, indeed, of a weak subtheory IE_1 with the induction just for bounded existential formulas, cf. Paris (1984). In particular, these results show that every nonstandard countable model of IE_1 has a nonstandard cut that is a model of whole PA; hence, in a sense, the model theory of bounded arithmetic is as complex as that of PA. Consult Hájek and Pudlák (1993), Kaye (1991), or Smorynski (1984) for the model theory of PA.

From proof theory we shall use theorems of Gentzen and Herbrand in various versions. The reader is advised to refer to Takeuti (1975) for Gentzen's sequent calculus.

We close this section with some remarks on notation. Logical connectives we shall use are the standard $\neg$, $\vee$, $\wedge$, $\rightarrow$, and $\equiv$ with the usual meaning – negation, disjunction, conjunction, implication, and equivalence – and the constants 1, 0 for *truth* and *falsity*.

The symbols $\subset$ and $\subseteq$ are used in the sense of *proper inclusion* and *inclusion*.

The symbols $f(n) = O(g(n))$, $f(n) = \Omega(g(n))$ and $f(n) = \Theta(g(n))$ denote that eventually $f(n) \leq cg(n)$, $f(n) \geq cg(n)$, and $c_1 g(n) \leq f(n) \leq c_2 g(n)$ where c, c_1, and c_2 are positive constants, and $f(n) = o(g(n))$ means that $f(n)/g(n) \rightarrow 0$.

2.2. Complexity theory

I assume that the reader is acquainted with such notions as *Turing machine, oracle Turing machine*, and *time* and *space* complexity measures. We adopt the multi-tape version of Turing machines with a read-only input tape and with a finite but arbitrarily large alphabet.

The basic relations between classes of languages $\mathrm{Time}(f)$ and $\mathrm{Space}(f)$ recognized by a deterministic Turing machine in time (respectively space) bounded by $f(n)$, n the length of the input, and their nondeterministic versions $\mathrm{NTime}(f)$ and $\mathrm{NSpace}(f)$ are

1. $\mathrm{Time}(f(n)) \subseteq \mathrm{NTime}(f(n)) \subseteq \mathrm{Space}(f(n))$
2. $\mathrm{Space}(f(n)) \subseteq \bigcup_c \mathrm{Time}(c^{f(n)})$
3. (Hartmanis and Stearns 1965) $\mathrm{Time}(f(n)) = \mathrm{Time}(c \cdot f(n))$ and $\mathrm{Space}(f(n)) = \mathrm{Space}(c \cdot f(n))$ whenever $n = o(f(n))$ and $n \leq f(n)$
4. (Hartmanis and Stearns 1965, Hartmanis, Lewis, and Stearns 1965)

$$\mathrm{Space}(f) \subset \mathrm{Space}(g)$$

and

$$\mathrm{Time}(f) \subset \mathrm{Time}(g \log(g))$$

whenever $f = o(g(n))$.
5. (Savitch 1970)

$$\mathrm{NSpace}(f) \subseteq \mathrm{Space}(f^2)$$

whenever $f(n)$ is itself computable in space $f(n)$
6. (Szelepcsényi 1987, Immerman 1988)

$$\mathrm{NSpace}(f) = \mathrm{coNSpace}(f)$$

for $f(n) \geq \log(n)$ and f itself computable in nondeterministic space $f(n)$.

7. (Hopcroft, Paul, and Valiant 1975) For $n \leq f(n)$

$$\text{Time}(f(n)) \subseteq \text{Space}\left(\frac{f(n)}{\log f(n)}\right)$$

Particular bounds to time or space define the usual complexity classes

$$\text{LinTime} = \bigcup_c \text{Time}(cn)$$

$$\text{P} = \bigcup_c \text{Time}(n^c)$$

$$\text{NP} = \bigcup_c \text{NTime}(n^c)$$

$$\text{L} = \text{Space}(\log(n))$$

$$\text{PSpace} = \bigcup_c \text{Space}(n^c)$$

$$\text{LinSpace} = \text{Space}(n)$$

$$\text{E} = \bigcup_c \text{Time}(c^n)$$

$$\text{EXP} = \bigcup_c \text{Time}(2^{n^c})$$

Oracle computations allow one to define hierarchies of languages, the most important of which are the *linear time hierarchy* LinH of Wrathall (1978)

$$\Sigma_0^{\text{lin}} = \text{LinTime and } \Sigma_{i+1}^{\text{lin}} = \text{NLinTime}^{\Sigma_i^{\text{lin}}}$$

and the *polynomial time hierarchy* PH of Stockmeyer (1977)

$$\Sigma_0^p = \text{P} \quad \text{and} \quad \Sigma_{i+1}^p = \text{NP}^{\Sigma_i^p}$$

The class of complements of languages from class X is denoted co$X$, and special classes of this form coΣ_i^{lin} and coΣ_i^p are denoted Π_i^{lin} and Π_i^p, respectively.

The class $\square_{i+1}^p$ is the class of functions computable by a polynomial-time machine with access to an oracle from the class Σ_i^p.

Some important facts about these classes include the following: $\Sigma_i^{\text{lin}} \subset \Sigma_i^p$ (and generally more resource in the "same" computational class properly increases the class; see Žák 1983 for a general diagonalization technique), and LinH contains L and is, in fact, equal to the class of *rudimentary predicates* as defined by Smullyan (1961) (cf. Wrathall 1978). It is also known that LinH also equals the class of predicates definable by Δ_0-formulas; we shall prove that in Section 3.2. Also note that either LinH $\neq$ PH or LinH does not collapse (i.e., LinH $\neq \Sigma_i^{\text{lin}}$ for all i).

The notion of NP-completeness, Cook's theorem, and the *P versus NP problem* are central to complexity theory, as well as to the connections with logic, and in Section 3.1 we shall review more basics, in particular some facts from circuit complexity.

Many interesting problems and notions arise in connection with *counting functions*. For $R(x, y)$ a binary predicate with the property that for every x there are only finitely many y's satisfying $R(x, y)$ defines the function

$$\#R(x) := \text{ the number of } y\text{'s such that } R(x, y)$$

Class #P consists of all functions $\#R(x)$ with the polynomial time computable relation $R(x, y)$ and satisfying the preceding finiteness property in a stronger form (cf. Valiant 1979):

$$R(x, y) \rightarrow |y| \leq |x|^{O(1)}$$

An important result of Toda (1989) is that every language in PH is polynomial–time reducible to a function in #P.

Nonuniform versions of the preceding classes are defined with the help of *advice functions*. *Polynomially bounded advice* is a function $f : \boldsymbol{N} \rightarrow \{0, 1\}^*$ such that:

$$|f(n)| = n^{O(1)}$$

The class P/poly, a nonuniform version of P, is the class of all sets A such that there are a set $B \in \mathrm{P}$ and a polynomially bounded advice function f for which it holds

$$x \in A \text{ iff } (x, f(|x|)) \in B$$

The classes NP/poly, L/poly, and so on, are defined analogously (see the paragraph after Theorem 3.1.4).

3

Basic complexity theory

We shall survey the basic notions and results of Boolean complexity (Section 3.1) and bounded formulas (Section 3.2) in this chapter. Most of the results in the first section are stated without a proof; some proofs appear later (Chapter 15, in particular) formalized in bounded arithmetic. In the second section, most proofs are at least sketched.

3.1. The P versus NP problem

The central problem in complexity theory, and a major problem of contemporary logic and mathematics, is whether the class P equals the class NP, the famous *P versus NP problem* (Cook 1971). By Cook's theorem the problem is equivalent to asking whether there is a polynomial time deterministic algorithm recognizing the set of satisfiable propositional formulas, or equivalently, such an algorithm recognizing the set of propositional tautologies.

One approach to this problem is via investigating the circuit-complexity of Boolean functions. Some interesting, although only preliminary, results were obtained in Boolean complexity.

Definition 3.1.1. *A* Boolean function *with n inputs and m outputs is a function*

$$f : \{0,1\}^n \mapsto \{0,1\}^m.$$

Examples of Boolean functions are obtained from any language $Z \subseteq \{0,1\}^*$: For any n, define the Boolean function $Z_n : \{0,1\}^n \mapsto \{0,1\}$ to be the characteristic function of $Z \cap \{0,1\}^n$.

On the other hand, a sequence of Boolean functions

$$f_n : \{0,1\}^n \mapsto \{0,1\},\ n = 0,1,\ldots$$

defines a language

$$\bigcup_n \{w \in \{0,1\}^n \mid f_n(w) = 1\}$$

Definition 3.1.2.

(a) *A Boolean* connective *is a Boolean function with one output. A* basis *is a finite set of connectives.*

(b) *A* Boolean circuit *with input variables* $x_1, \ldots, x_n$*; output variables* $y_1, \ldots, y_m$*; and basis of connectives* $\Omega = \{g_1, \ldots, g_k\}$ *is a labeled acyclic directed graph whose out-degree* 0 *nodes are labeled by* y_j*'s, in-degree* 0 *nodes are labeled by* x_i*'s or by constants from* Ω*, and whose in-degree* $\ell \geq 1$ *nodes are labeled by functions from* Ω *of arity* ℓ*.*

(c) *A* Boolean formula *is a Boolean circuit in which every node has the out-degree at most* 1*.*

We shall mostly consider the *de Morgan* basis $\Omega = \{0, 1, \neg, \vee, \wedge\}$.

A Boolean circuit with input variables $x_1, \ldots, x_n$ naturally computes a Boolean function with domain $\{0,1\}^n$: given input $\bar{\epsilon} \in \{0,1\}^n$ evaluate consecutively the nodes of the circuit by 0, 1 where a node gets the value computed by the connective labeling the node from the values at the incoming nodes. The requirement that a circuit is acyclic guarantees that this can be done consistently (and uniquely).

Definition 3.1.3.

(a) *The* size *of a circuit is the number of its nodes.*

(b) *The* depth *of a circuit is the maximum length of a directed path in the circuit.*

(c) *For a Boolean function* f*,* $C_\Omega(f)$ *denotes the minimal size of a circuit with basis* Ω *computing* f*,* $Depth_\Omega(f)$ *denotes the minimal depth of a circuit with basis* Ω *computing* f*, and* $L_\Omega(f)$ *denotes the minimal size of a formula with basis* Ω *computing* f*.*
When Ω *is the* de Morgan *basis then the index* Ω *is usually omitted.*

The following theorem is the stimulus for investigating the circuit complexity of Boolean functions.

Theorem 3.1.4. *Let* $Z \subseteq \{0,1\}^*$ *be a polynomial-time recognizable language* $Z \in$ P*. Then there exist a polynomial* $p(x)$ *and a sequence* $\{C_n\}_n$ *of circuits in* de Morgan *basis with one output such that for all* n

1. Z_n *is computed by* C_n
2. *the size of* C_n *is at most* $p(n)$*.*

In other words, $C(Z_n) \leq p(n)$*, for all* n*.*

Note that the languages Z with polynomially bounded $C(Z_n)$ are exactly those from the class P/poly: circuits C_n can act as advice for inputs of length n (as

the *evaluation* of a circuit can be performed by a polynomial time algorithm), and, for each n, an algorithm computing whether $(x, f(|x|)) \in B$ (see the end of Section 2.2) can be turned into a polynomial size circuit.

Corollary 3.1.5. *Assume that for some* $Z \in$ NP *the function* $C(Z_n)$ *is not bounded by any polynomial in* n.

Then P $\neq$ NP.

Hence the following problem is a fundamental one.

Fundamental problem. *Is there a language* $Z \in$ NP *with superpolynomial circuit-size complexity?*

The next theorem shows that even the unexpected negative answer has important corollaries.

Theorem 3.1.6 (Karp and Lipton 1982). *Assume that every* NP *language can be computed by a family of polynomial size circuits; that is,* NP $\subseteq$ P/poly.

Then the polynomial time hierarchy PH *collapses to its second level*

$$\text{PH} = \Sigma_2^p = \Pi_2^p.$$

Theorem 3.1.7 (Shannon 1949, Muller 1956). *For every* n *there are Boolean functions with* n *inputs and one output having the circuit complexity* $\Omega(2^{n-\log n})$.

Proof. There are 2^{2^n} Boolean functions with n unknowns and one output, whereas there are at most

$$(n+m)^{O(m)}$$

circuits of size $\leq m$, which is less than 2^{2^n} for $m \leq (2^n/c \cdot n)$ and c a sufficiently large constant. Q.E.D.

Next we give an account of the *method of approximations* of Razborov (1989), following to some extent an exposition of Karchmer (1993) but stressing the ultra-product interpretation of the construction. We include this material here because it is the only framework that can be, at least in principle, applied to unrestricted circuits.

Let $f : \{0,1\}^n \to \{0,1\}$ be a Boolean function of n inputs and one output, and $U := f^{-1}(0)$.

Let C be a circuit in n inputs and of size m. We shall denote the nodes of C by $z_1, \ldots, z_m$ where the first n nodes are labeled by $x_1, \ldots, x_n$ and the last one is the output node y. We identify node z_i with the Boolean function of n inputs computed by the subcircuit ending in z_i and we shall occasionally write x_i instead of z_i if $i \leq n$ and y if $i = m$.

The idea is to take the set of all computations of C on inputs $u \in U$ and produce by "ultraproduct" a new computation on some $w \notin U$. As all original