Iddo Tzameret

Introduction to Concrete Complexity

Lecture Notes

Imperial College London
Dept. of Computing

December 25, 2024

Contents

0.1 Summary of Module

We are interested in approaches to the fundamental hardness questions in computational complexity.

Computational complexity: the study of which problems can be efficiently computed and which cannot.

Efficiency: we understand efficiency as Polynomial Time computability. A Boolean function $f : \{0,1\}^n \to \{0,1\}$ is said to be efficiently computable if there is a polynomial-time Turing Machine M that, on input x in $\{0,1\}^n$ outputs $f(x)$ and runs in time n^c for some constant c. Note that n^c is a polynomial in the input length n (the exponent c does not depend on the input length n).

The arguably main question of the theory of computation, the one that subsumes in some sense many problems in other parts of computing, once is the P vs NP question: Can we separate P from NP, namely, is there a language in NP that is not in P? In other words, can we prove that SAT (Boolean satisfiability problem) cannot be solved in polynomial time? And yet again, roughly, can problems whose solutions once given can be verified efficiently, can be solved efficiently?

We are interested in *concrete* approaches, namely, considering a simple, usually combinatorial-looking, model of computation, such as a Boolean circuit, and establishing lower bounds against the size of circuits required to prove certain specific functions that are given to us concretely (usually, these functions also possess some straightforward combinatorial properties; e.g., they represented specific graph problems). In this sense, the question is concrete because the result is unconditional (namely, it does not depend on unproved assumptions, such as $\mathsf{P} \neq \mathsf{NP}$), and the model itself is concrete: it is a (primarily combinatorial) object of which its size we lower bound in precise terms (e.g., circuit C computing function $f(x)$ must have size $2^{|x|}$, where $|x|$ is the bit-size of the input x).

Three main concrete approaches to the fundamental hardness questions are the following:

1. Circuit Complexity
2. Proof Complexity
3. Algebraic Complexity

We shall see a bit from each, mainly circuit complexity and some basic proof complexity while commenting briefly on algebraic complexity.

Other approaches to the fundamental hardness questions are usually more intrinsic to complexity theory. In that respect, the whole of computational complexity theory could be viewed as “approaching" the fundamental hardness questions through complexity class, reductions, concrete lower bounds and the relation between these notions and results. One intriguing approach that makes this attempt in particular is the “Meta Complexity" approach. We are not going to touch on this in this course.

Chapter 1
Introduction to Circuit Complexity

1.1 Basic Circuit Complexity

Definition 1.1 (Boolean Circuit) Let $n \in \mathbb{N}$ and $x_1, \ldots, x_n$ be n variables. A Boolean Circuit C with n inputs is a directed acyclic graph. It contains n nodes with no incoming edges, called the *input nodes* and a single node with no outgoing edges, called the *output node*. All other nodes are *internal nodes* or *gates*, and are labelled by the logical gates $\vee$, $\wedge$, $\neg$ (i.e., logical OR, AND, NOT, resp.). The $\vee$, $\wedge$ nodes have fan-in (i.e., number of incoming nodes) 2, and $\neg$ has fan-in 1. The *size of* C, denoted $|C|$, is the number of nodes in the underlying graph. C is called a *formula* if each node has at most one outgoing edge (i.e., the underlying graph is a tree),

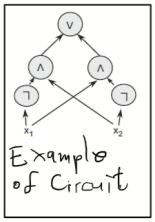
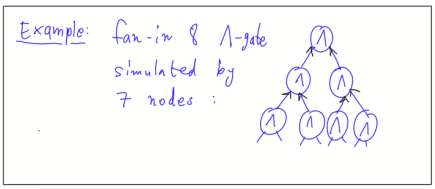


Fig. 1.1 Example of a simple Boolean circuit.

Comment

Fan-in $d > 2$ can be simulated by a tree of $d - 1$ nodes:



? What is the function computed by the circuit below?

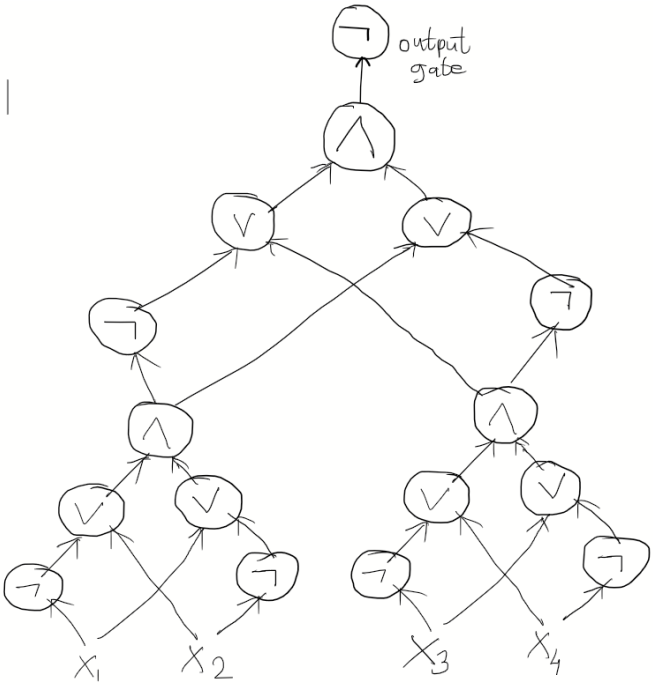


Fig. 1.2 Question

Answer

PARITY on 4 input bits. It outputs 1 if the number of 1's in the input is odd. I.e., it computes the XOR of $x_1, \ldots, x_4$. In other words, it computes the function $x_1 + x_2 + x_3 + x_4 \bmod 2$. (Understand why that is. Hint: recall that $\neg x_1 \vee x_2$ is

logically equivalent to $x_1 \rightarrow x_2$. And notice that this structure repeats throughout the circuit.)

1.2 Circuit families, language recognition, and function computation

Let $\mathbb{N}$ denote the set of natural numbers starting from 1.

A *language* is a set of (finite) strings. Note that the language can contain infinite many strings, only that each string is finite. A *string* is an ordered sequence of symbols from a fixed constant size alphabet. We shall use mainly strings over the alphabet consisting of two symbols $\{0,1\}$. Hence, a language is simply a set $L \subseteq \{0,1\}^*$ (recall, that $\{0,1\}^*$ is the set of all finite 0-1 strings, including the empty string).

Definition 1.2 Let $T : \mathbb{N} \rightarrow \mathbb{N}$ be a function. A $T(n)$-***sized circuit family*** is a sequence $\{C_n\}_{n\in\mathbb{N}}$ of Boolean circuits, where C_n has n input-gates (i.e., n inputs-bits) and a single output-bit such that $|C_n| \leq T(n), \forall n \in \mathbb{N}$.[1]

A language $L \subseteq \{0,1\}^*$ is said to be *in* $\mathsf{SIZE}(T(n))$, namely,

$$L \in \mathsf{SIZE}(T(n)),$$

if there is a $T(n)$-size circuit family $\{C_n\}_{n\in\mathbb{N}}$ s.t. (such that)

$$\forall n \in \mathbb{N}\ \forall x \in \{0,1\}^n : x \in L \Leftrightarrow C_n(x) = 1.$$

In this case, we say that the family $\{C_n\}_{n\in\mathbb{N}}$ **decides** the language L.

Similarly, for a *function* $f : \{0,1\}^* \rightarrow \{0,1\}$, $f \in \mathsf{SIZE}(T(n))$ if there exists $T(n)$-size circuit family $\{C_n\}_{n\in\mathbb{N}}$ such that $\forall n \in \mathbb{N} \quad \forall x \in \{0,1\}^n$, $f(x) = 1 \Leftrightarrow C_n(x) = 1$. In this case, we say that the family $\{C_n\}_{n\in\mathbb{N}}$ **computes** the function f.

Slice of function. Let $f : \{0,1\}^* \rightarrow \{0,1\}$ be a Boolean function. We can consider the *slice of size n* of f to be the function $f_n : \{0,1\}^n \rightarrow \{0,1\}$, which is f restricted to inputs of length precisely n.

Similarly, we can consider $\{f_n\}_{n\in\mathbb{N}}$ to be the *family* of Boolean functions $f_n : \{0,1\}^n \rightarrow \{0,1\}$, for all n at least 1 (so, this is a family of all slices of f). In other words, the family $\{f_n\}_{n\in\mathbb{N}}$ is the sane as $f : \{0,1\}^* \rightarrow \{0,1\}$.

STOPPED polish Here

1.3 Quick recap: growth functions, O-notation and run-times

Be sure to recall the following basic definitions of (though we are not going to use them concretely in this course).

- Time bounds for Turing Machines (TMs)
- The class P (aka PTIME)
- The class NP, verifiers, short certificates
- SAT is NP complete: Cook-Levin theorem

Time Complexity

We mainly consider the worst-case time for a problem to be solved by a given TM.

Definition 1.3 Let M be a TM that halts on every input. Let $f : \mathbb{N} \to \mathbb{N}$ be a function, s.t., $f(n)$ is the max number of steps it takes M to halt on inputs of length n. We say f is the running time of M. And that M runs in time $f(n)$.

O-Notation

We usually use n for the length of inputs
Recall:

1. $f(n) = O(g(n))$ if $\exists$ constant c and $n_0 \in \mathbb{N}$ st. $\forall n \geqslant n_0 \quad f(n) \leqslant cg(n)$ (for two functions $f, g : \mathbb{N} \to \mathbb{R}^+$).
2. $f(n) = o(g(n))$ if $\lim_{n\to\infty} \frac{f(n)}{g(n)} = 0$

Basic Growth Rates

1. When $f = O(\log n)$ the base b of $\log_b n$ is irrelevant.
2. $2^{O(n)}$ means $\leqslant 2^{cn}$ for some c, const.
3. *Polynomial growth rate* means $n^{O(1)} = 2^{O(\log n)} = n^c$, for some constant c, namely c is independent of n (while n grows, c stays fixed).
4. *Exponential growth rate* (similarly, *exponential bound*) means for us 2^{n^δ}, for a real constant $\delta > 0$. (Some texts insist that "exponential growth" should only refer to 2^{cn}, for a constant $c > 0$. Note the difference between this and our definition!)

Definition 1.4 (P/poly; polynomial-size circuits) The class P/poly is the class of languages that are decidable by polynomial-size circuit families. That is, $\mathsf{P/poly} = \bigcup_{c\in\mathbb{N}} \mathsf{SIZE}(n^c)$.

Theorem 1.1 $\mathsf{P} \subseteq \mathsf{P/poly}$.

Proof Assignment/tutorial. [See for a sketch in Arora-Barak'10, Sec. 6.1.1. page 110; [1]]. □

1.4 Our main focus: lower bounds

Lower bounds, hard functions. We say that we have a *super-polynomial lower bound against* P/poly, namely a super-polynomial lower bound against (polynomial-size) Boolean circuits, if the following occurs: there exists a Boolean function $f : \{0,1\}^* \to \{0,1\}$ such that no polynomial-size circuit family $\{C_n\}_n$ computes f. In other words, f is not in $\mathsf{P/poly}$. If we show that $f \notin \mathsf{SIZE}(T(n))$, we say that we have *proven a* $T(n)$*-lower bound for* f (against Boolean circuits). We say that this f is *hard* for $\mathsf{SIZE}(T(n))$.

Motivation for Circuit Complexity

1) Non-uniformity: maybe for every $l \in \mathbb{N}$ there's a small circuit for SAT that can solve it in quadratic time? (Karp-Lipton: PH doesn't collapse $\Rightarrow NP \subseteq P/$ poly).

2) Mathematically *cleaner* than Turing Machines; so might have better hope for lower bounds.

1.4.1 Most functions are hard: Shannon lower bound

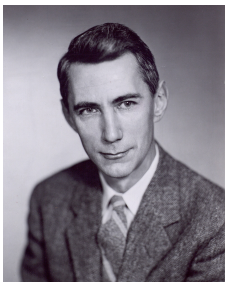


Fig. 1.3 C.E. Shannon. Source: By Unknown author - Tekniska Museet, CC BY 2.0, https://commons.wikimedia.org/w/index.php?curid=144716444

Open: NEXP $\stackrel{?}{\subseteq} P/poly$ (note that by the time hierarchy theorem EXP $\subsetneq P$; but in "NEXP $\stackrel{?}{\subseteq} P/poly$" we consider NEXP which is *uniform*, and P/poly which is *non*-uniform).

Theorem 1.2 (Shannon lower bound) *For every $n > 1$ exists a function $f : \{0,1\}^n \to \{0,1\}$ that cannot be computed by a circuit of size $\leq 2^n/10n$.*

Proof (by counting method.)

Claim We can encode a boolean circuit of size S with $9 \cdot S \log S$ bits. □

Proof (Claim) Using adjacency List:

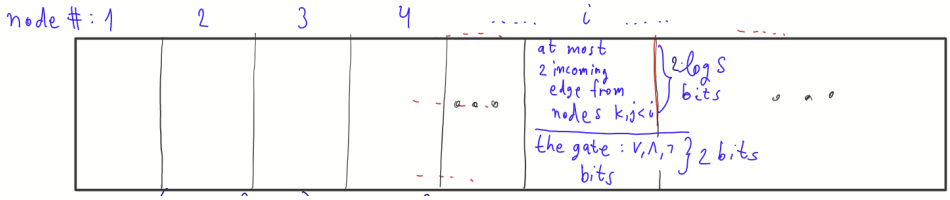


Fig. 1.4 Enter Caption

We have:

$$S \cdot (2 + 2\log S) \leq 9S \log S.$$

(Actually, we have: $2S(\log S + 1) = 2S(2 \log S) \leq 4S \log S$.) □

We now show that the number of circuits of size $\leq 2^n/10n$ is smaller than that of possible Boolean functions w/ n inputs. Thus, by the Pigeonhole Principle, there exists a Boolean function that cannot be computed by a circuit of size $\leq 2^n/10n$.

The number of circuits of size S is bounded from above by the number of distinct codes of circuits of size S, which is

$$\begin{aligned} &\leq 2^{9 \cdot S \log S} \\ &= 2^{\left(9 \cdot 2^n/10n \cdot \log(2^n/10n)\right)} = 2^{(9 \cdot 2^n/10n \cdot (n - \log 10n))} \\ &\leq 2^{\frac{9n}{10n} \cdot 2^n} = 2^{\frac{9}{10} \cdot 2^n}. \end{aligned}$$

But this is less than 2^{2^n} the number of distinct Boolean functions $f : \{0,1\}^n \to \{0,1\}$. □

Note: the reason we need to lower bound $2^n/10n$ circuits (and not $2^n/c$), is that encoding a circuit is super linear (i.e., $\Omega(\log n)$).

Chapter 2
Monotone Circuit Lower Bounds

We've seen that proving that SAT is not in P/poly, i.e., can't be solved by polynomial-size circuits, implies that $\mathsf{P} \neq \mathsf{NP}$. Due to the notorious difficulty of these questions, we are interested in proving *weaker* lower bounds, namely, some lower bounds against restricted classes of circuits. Here, we study such a restricted circuit class: A boolean circuit without negation gates, i.e., monotone circuits.

Definition 2.1 (Monotone circuit) A Monotone circuit is a Boolean circuit that contains fan-in two gates AND and OR, but has *no* NOT gates.

This means in particular that monotone circuits can compute only monotone functions: a Boolean function is said to be monotone if increasing the number of ones in the input cannot flip the value of the function from 1 to 0.

More precisely, for $\bar{x}, \bar{y} \in \{0,1\}^n$, write $\bar{x} \geqslant \bar{y}$ iff $\forall i \in [n], x_i \geqslant y_i$, where $[n]$ denotes $\{1, \ldots, n\}$. (Here, $x_i \geq y_i$ for Boolean x_i, y_i means simply that $1 \geq 0$ and $0 \geq 0$, $1 \geq 1$, while $0 \not\geq 1$.)

Definition 2.2 (Monotone function) A Boolean function $f : \{0,1\}^n \to \{0,1\}$ is said to be *monotone* if $\forall \bar{x} \geq \bar{y}$, $f(\bar{x}) \geqslant f(y)$.

Many NP problems are monotone, like CLIQUE:

Given an undirected graph $G = (V, E)$ with n nodes, a k-clique in G is a set $U \subseteq V$ of size k, st. every pair of nodes $u_1, u_2 \in U$ is connected by an edge (in E):

$$\forall u_1 \in u \forall u_2 \in u (u \neq u_2 \Rightarrow (u_1, u_2) \in E).$$

Recall that a computational (decision) problem is a *language*, namely an infinite set of finite strings over a finite alphabet (usually the alphabet $\{0,1\}$). Here, our language consists of all the strings that encode (in some natural way) an accepted graph, i.e., a k-clique with n nodes. The natural way to encode a graph in our case is this: a graph $G = (V, E)$ with n nodes, is encoded by $\binom{n}{2}$ input variables x_{ij}, where the semantic of the encoding is: $x_{ij} = 1$ iff $(i, j) \in E$. In other words, if the input variable $x_{ij} = 1$, our input graph contains the edge (i, j), and otherwise it does not.

We are interested in CLIQUE(k, n) for a fixed k, as the following Boolean function:

The computational problem **CLIQUE**(k, n):
Input: Undirected graph $G = (V, E)$ with n nodes, and a number k (given in unary, i.e., 1^k).
Accept: if the graph G contains a k-clique.
Reject: otherwise.

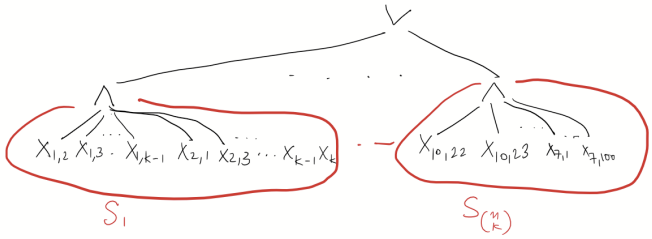
Note that CLIQUE(k, n) is a monotone function: If we add 1's to the input, we only *increase* the chance it has a k-clique!

It is known that CLIQUE(k, n) is NP-complete (see standard complexity textbooks; e.g., Papadimitriou 1994).

Since CLIQUE(n, k) is a monotone (Boolean) function we can compute it by a monotone Boolean circuit.

Example of a monotone circuit computing CLIQUE(n, k)

"Run" over all $\binom{n}{k}$ k-sub-graphs in G, and check if at least one of those is a clique:



$S_1, S_2, \ldots, S_{\binom{n}{k}}$ are the $\binom{n}{k}$ subgraphs in G each of size k. Size of this circuit: $O\left(k^2 \cdot \binom{n}{k}\right)$. We call such a circuit for computing CLIQUE(n, k) consisting of a big $\vee$ of all S_i's, each computed by $\wedge$'s of edges, a ***crude-circuit*** for CLIQUE(n, k).

Notation: $CC\left(S_1, \ldots, S_{\binom{n}{k}}\right)$ is the crude circuit computing the $\vee$ of all subgraphs $S_1 \ldots S_{\binom{n}{k}}$. In general, we shall use different subgraphs: $CC(x_1, \ldots, x_m)$ for $x_i \subseteq V$ not necessarily of size k.

Note: When $k = w(\log n)$, $CC\left(S_1, \ldots, S_{(k_k)}\right)$ is of exponential size. The following theorem shows this naive monotone circuit cannot be improved much:

Theorem 2.1 *[Razborov] Let $k = \sqrt[4]{n}$. Then, every monotone circuit computing CLIQUE(n, k) has size $2^{\Omega(\sqrt[8]{n})}$.*

Fig. 2.1 Alexander Razborov

That is, exists a constant c s.t. for large enough $n \in \mathbb{N}$, if C_n computes CLIQUE(n, k) then $|C_n| \geqslant 2^{c \cdot \sqrt[8]{n}}$.

Approximation Method

Here we provide an overview of the approach we take to prove Theorem 2.1 which is called *the approximation method*. Our exposition is taken from Papadimitriou's textbook [2]. We shall describe a way of approximating any **monotone** circuit for CLIQUE(n, k) by a crude circuit, namely a big OR of cliques:

1. Given a monotone circuit C , we shall construct a crude circuit $CC(X1, \ldots, Xm)$ for some m and $|X_i| \leq l$ (for some l, all $i = 1, \ldots, m$), that approximates CLIQUE(n, k) with **precision** that is dependent on the number of gates in C.
2. I.e., if the precision is not good, namely the crude circuit $CC(X1, \ldots, Xm)$ for CLIQUE (n,k) we end up with makes *many errors* on the CLIQUE (n, k) function (i.e., says "NO" on an input that has a k-clique, and "YES" on an input that has no k-clique), it means that the circuit C has *many gates*, and vice versa.
3. We show that every crude circuit $CC(X1, \ldots, Xm)(|X_i| \leq l$ for some l, for all $i = 1, \ldots, m$), ought to make *exponentially many errors* on the function CLIQUE(n, k). From 2 above we conclude that the number of gates in C was exponential.

The **approximation** (i.e., construction of a crude circuit for CLIQUE(n, k) given the circuit C) will proceed in steps, one step for each gate of the monotone circuit:

1. If C is a monotone circuit computing CLIQUE(n, k) we can *approximate* any gate OR or AND in C with a crude circuit.
2. Each such approximation step introduces rather few errors (false positives and false negatives).

2.1 Proof of monotone circuit lower bounds

Parameters & notation

Recall we want to compute CLIQUE(n, k) with n the number of nodes in the graph and k the size of a clique within the graph. We set:

$$k = \sqrt[4]{n}.$$

Goal: Show that every monotone circuit computing CLIQUE(n, k) has size at least $2^{c\sqrt[8]{n}}$ for some constant c (for sufficiently large n).

$$\begin{aligned} l &= \sqrt[8]{n} \\ p &\approx \sqrt[8]{n} \\ M = (p-1)^l \cdot l! \quad &\approx (\sqrt[8]{n}-1)^{\sqrt[8]{x}} \cdot (\sqrt[8]{n})! \\ &\leq (\sqrt[4]{n})^{\sqrt[8]{n}} \end{aligned}$$

Each crude-circuit we use in the approximation is:

$$CC(x_1, \ldots, x_m)$$

for $m \leqslant M$ and $|X_i| \leqslant l, \forall i \in [m]$.

- The approximation of the monotone circuit C that computes CLIQUE(n, k) is done by induction on the size of C, ie., number of $\vee, \wedge$ gates in C.
- Comment: Such induction is also called "Induction on the structure of C".
- Such induction proceeds as follows:

Base Case

$|C| = 1$, ie., C consists of only a single input gate g_{ij}. Recall g_{ij} is an input gate denoting whether $(i, j) \in E$, for $i, j \in V$. That is, if there is an edge between i and j in the input graph G.

This is an easy case: We need to show a crude cat $\mathrm{CC}(X_1, \ldots, X_m)$ w/ $m \leqslant M$ and $|X_i| \leqslant l \quad \forall i \in [m]$ that approximates g_{ii} (without introducing too many errors; We shall count precisely the number of potential errors later). But the circuit $CC(\{i, j\}) = g_{ij}$ by definition. (Hence, no errors here!)

Induction Step

Given two crude circuits $\mathrm{CC}(\mathcal{X})$ and $\mathrm{CC}(\mathcal{Y})$, with $\mathcal{X} = \{x_1, \ldots, x_m\}$, $\mathcal{Y} = \{y_1, \ldots, y_{m'}\}$, $m \leq M$, and $|x_i| \leq \ell$, for all i, $m' \leq M$ and $|y_i| \leq \ell$, for all i.

We wish to construct another crude circuit for computing $CC(\mathcal{X}) \vee CC(\mathcal{Y})$, and $\mathrm{CC}(\mathcal{X}) \wedge \mathrm{CC}(\mathcal{Y})$.

Case 1: $\vee$-gate.

Naive attempt: $\mathrm{CC}(\mathcal{X}) \vee \mathrm{CC}(\mathcal{Y})$ is approximated by $\mathrm{CC}(\mathcal{X} \cup \mathcal{Y})$. That is, $\mathrm{CC}(x_1, \ldots, x_m, y_1, \ldots, y_m)$. At first glance this is a good solution because it does not introduce any errors (why?). But there is a problem: what if $m + m' > M$?

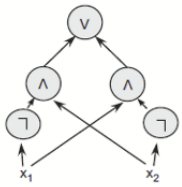
Solution: We need to cleverly *reduce* the number of sets $x_1, \ldots, x_m, y_1, \ldots, y_{m'}$. To do this we use a combinatorial lemma called The Sunflower Lemma.

Chapter 3
Constant Depth Circuit Lower Bounds

3.1 Defining constant depth circuits

Recall that the depth of a circuit is the maximal length of a path from a leaf to the output node.

Here is an example of a circuit of depth 3:



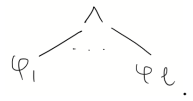
Recall that we are interested in the asymptotic study of circuit families, not a single circuit. That is, we want to consider how circuit size grows when the number of inputs n to a family of Boolean functions grows.

We can consider circuits of constant depth. This means that the depth of the circuit is *independent of the number of inputs* n. In other words, while n the number of input gates can grow to infinity in the Boolean circuit family $\{C_n\}_{n=1}^{\infty}$, the depth of each circuit C_n stays the same!

Note: If the depth of the circuit is constant & the fan-in of gates is at most 2, then the number of functions we can compute w/ such constant-depth circuits is constant. For example, the number of variables (appearing as leaves, i.e., input nodes) is constant that way, so for a large enough number of inputs n, we will not be able to compute functions that read all the n inputs. Thus, this model is *not* complete: for a constant d, a depth-d circuit cannot compute all Boolean functions over n inputs for every $n \in \mathbb{N}$.

Corollary: To make the model of const. depth meaningful we allow *unbounded* fan-in gates:

Unbounded fan-in AND ($\varphi_1 \wedge \varphi_2 \wedge \ldots \wedge \varphi_\ell$):



Unbounded fan-in OR ($\varphi \vee \cdots \vee \varphi_\ell$): Where φ_i's are circuits (of constant depth)

by themselves, with possibly *joint* nodes.

Important: When speaking about constant-depth circuits, we assume by default the fan-in of $\vee, \wedge$ is *unbounded*. ($\neg$ has fan-in one as always.)

Chapter 4
Introduction

4.1 Section Heading

Use the template *chapter.tex* together with the document class SVMono (monograph-type books) or SVMult (edited books) to style the various elements of your chapter content conformable to the Springer Nature layout.

4.2 Section Heading

Instead of simply listing headings of different levels we recommend to let every heading be followed by at least a short passage of text. Furtheron please use the LaTeX automatism for all your cross-references and citations.

Please note that the first line of text that follows a heading is not indented, whereas the first lines of all subsequent paragraphs are.

Use the standard `equation` environment to typeset your equations, e.g.

$$a \times b = c \,, \tag{4.1}$$

however, for multiline equations we recommend to use the `eqnarray` environment[1].

$$\left|\nabla U_\alpha^\mu(y)\right| \le \frac{1}{d-\alpha} \int \left|\nabla \frac{1}{|\xi - y|^{d-\alpha}}\right| d\mu(\xi) = \int \frac{1}{|\xi - y|^{d-\alpha+1}}\, d\mu(\xi) \tag{4.2}$$

$$= (d-\alpha+1) \int_{d(y)}^{\infty} \frac{\mu(B(y,r))}{r^{d-\alpha+2}}\, dr \le (d-\alpha+1) \int_{d(y)}^{\infty} \frac{r^{d-\alpha}}{r^{d-\alpha+2}}\, dr \tag{4.3}$$

4.2.1 Subsection Heading

Instead of simply listing headings of different levels we recommend to let every heading be followed by at least a short passage of text. Further on please use the LaTeX automatism for all your cross-references and citations as has already been described in Sect. 4.2.

> Please do not use quotation marks when quoting texts! Simply use the `quotation` environment – it will automatically be rendered in the preferred layout.

4.2.1.1 Subsubsection Heading

Instead of simply listing headings of different levels we recommend to let every heading be followed by at least a short passage of text. Furtheron please use the LaTeX automatism for all your cross-references and citations as has already been described in Sect. 4.2.1, see also Fig. 4.1[2]

Please note that the first line of text that follows a heading is not indented, whereas the first lines of all subsequent paragraphs are.

Paragraph Heading

Instead of simply listing headings of different levels we recommend to let every heading be followed by at least a short passage of text. Furtheron please use the LaTeX automatism for all your cross-references and citations as has already been described in Sect. 4.2.

---

[1] In physics texts please activate the class option `vecphys` to depict your vectors in ***boldface-italic*** type - as is customary for a wide range of physical subjects.

[2] If you copy text passages, figures, or tables from other works, you must obtain *permission* from the copyright holder (usually the original publisher). Please enclose the signed permission with the manucript. The sources must be acknowledged either in the captions, as footnotes or in a separate section of the book.

Please note that the first line of text that follows a heading is not indented, whereas the first lines of all subsequent paragraphs are.

For typesetting numbered lists we recommend to use the `enumerate` environment – it will automatically render Springer's preferred layout.

1. Livelihood and survival mobility are oftentimes coutcomes of uneven socioeconomic development.
   a. Livelihood and survival mobility are oftentimes coutcomes of uneven socioeconomic development.
   b. Livelihood and survival mobility are oftentimes coutcomes of uneven socioeconomic development.
2. Livelihood and survival mobility are oftentimes coutcomes of uneven socioeconomic development.

Subparagraph Heading

In order to avoid simply listing headings of different levels we recommend to let every heading be followed by at least a short passage of text. Use the LaTeX automatism for all your cross-references and citations as has already been described in Sect. 4.2, see also Fig. 4.2.

Please note that the first line of text that follows a heading is not indented, whereas the first lines of all subsequent paragraphs are.

For unnumbered list we recommend to use the `itemize` environment – it will automatically render Springer's preferred layout.

- Livelihood and survival mobility are oftentimes coutcomes of uneven socioeconomic development, cf. Table 4.1.
  – Livelihood and survival mobility are oftentimes coutcomes of uneven socioeconomic development.
  – Livelihood and survival mobility are oftentimes coutcomes of uneven socioeconomic development.

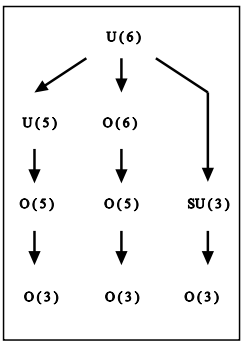


Fig. 4.1 If the width of the figure is less than 7.8 cm use the `sidecapion` command to flush the caption on the left side of the page. If the figure is positioned at the top of the page, align the sidecaption with the top of the figure – to achieve this you simply need to use the optional argument `[t]` with the `sidecaption` command

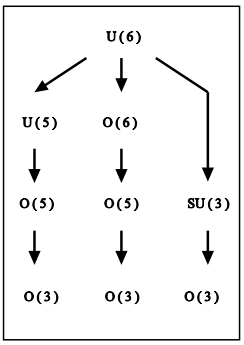


Fig. 4.2 Please write your figure caption here

Table 4.1 Please write your table caption here

Classes	Subclass	Length	Action Mechanism
Translation	mRNA[a]	22 (19–25)	Translation repression, mRNA cleavage
Translation	mRNA cleavage	21	mRNA cleavage
Translation	mRNA	21–22	mRNA cleavage
Translation	mRNA	24–26	Histone and DNA Modification

[a] Table foot note (with superscript)

- Livelihood and survival mobility are oftentimes coutcomes of uneven socioeconomic development.

Run-in Heading Boldface Version Use the LaTeX automatism for all your cross-references and citations as has already been described in Sect. 4.2.

Run-in Heading Boldface and Italic Version Use the LaTeX automatism for all your cross-references and citations as has already been described in Sect. 4.2.

Run-in Heading Displayed Version
Use the LaTeX automatism for all your cross-references and citations as has already been described in Sect. 4.2.

4.3 Section Heading

Instead of simply listing headings of different levels we recommend to let every heading be followed by at least a short passage of text. Furtheron please use the LaTeX automatism for all your cross-references and citations as has already been described in Sect. 4.2.

Please note that the first line of text that follows a heading is not indented, whereas the first lines of all subsequent paragraphs are.

If you want to list definitions or the like we recommend to use the Springer-enhanced `description` environment – it will automatically render Springer's preferred layout.

Type 1 That addresses central themes pertainng to migration, health, and disease. In Sect. 4.1, Wilson discusses the role of human migration in infectious disease distributions and patterns.

Type 2 That addresses central themes pertainng to migration, health, and disease. In Sect. 4.2.1, Wilson discusses the role of human migration in infectious disease distributions and patterns.

4.3.1 Subsection Heading

In order to avoid simply listing headings of different levels we recommend to let every heading be followed by at least a short passage of text. Use the LaTeX automatism for all your cross-references and citations citations as has already been described in Sect. 4.2.

Please note that the first line of text that follows a heading is not indented, whereas the first lines of all subsequent paragraphs are.

If you want to emphasize complete paragraphs of texts we recommend to use the newly defined Springer class option `graybox` and the newly defined environment `svgraybox`. This will produce a 15 percent screened box 'behind' your text.

If you want to emphasize complete paragraphs of texts we recommend to use the newly defined Springer class option and environment `svgraybox`. This will produce a 15 percent screened box 'behind' your text.

4.3.1.1 Subsubsection Heading

Instead of simply listing headings of different levels we recommend to let every heading be followed by at least a short passage of text. Furtheron please use the LaTeX automatism for all your cross-references and citations as has already been described in Sect. 4.2.

Please note that the first line of text that follows a heading is not indented, whereas the first lines of all subsequent paragraphs are.

Theorem 4.1 *Theorem text goes here.*

Definition 4.1 Definition text goes here.

Proof Proof text goes here. □

Paragraph Heading

Instead of simply listing headings of different levels we recommend to let every heading be followed by at least a short passage of text. Furtheron please use the LaTeX automatism for all your cross-references and citations as has already been described in Sect. 4.2.

Note that the first line of text that follows a heading is not indented, whereas the first lines of all subsequent paragraphs are.

Theorem 4.2 *Theorem text goes here.*

Definition 4.2 Definition text goes here.

Proof Proof text goes here. □

Trailer Head

If you want to emphasize complete paragraphs of texts in an `Trailer Head` we recommend to use

```
\begin{trailer}{Trailer Head}
...
\end{trailer}
```

? Questions

If you want to emphasize complete paragraphs of texts in an `Questions` we recommend to use

```
\begin{question}{Questions}
...
\end{question}
```

> Important

If you want to emphasize complete paragraphs of texts in an `Important` we recommend to use

```
\begin{important}{Important}
...
\end{important}
```

Attention

If you want to emphasize complete paragraphs of texts in an `Attention` we recommend to use

```
\begin{warning}{Attention}
...
\end{warning}
```

Program Code

If you want to emphasize complete paragraphs of texts in an `Program Code` we recommend to use

```
\begin{programcode}{Program Code}
\begin{verbatim}...\end{verbatim}
\end{programcode}
```

Tips

If you want to emphasize complete paragraphs of texts in an `Tips` we recommend to use

```
\begin{tips}{Tips}
...
\end{tips}
```

Overview

If you want to emphasize complete paragraphs of texts in an `Overview` we recommend to use

```
\begin{overview}{Overview}
...
\end{overview}
```

Background Information

If you want to emphasize complete paragraphs of texts in an `Background Information` we recommend to use

```
\begin{backgroundinformation}{Background Information}
...
\end{backgroundinformation}
```

Legal Text

If you want to emphasize complete paragraphs of texts in an `Legal Text` we recommend to use

```
\begin{legaltext}{Legal Text}
...
\end{legaltext}
```

Acknowledgements If you want to include acknowledgments of assistance and the like at the end of an individual chapter please use the `acknowledgement` environment – it will automatically render Springer's preferred layout.

Appendix

When placed at the end of a chapter or contribution (as opposed to at the end of the book), the numbering of tables, figures, and equations in the appendix section continues on from that in the main text. Hence please *do not* use the `appendix` command when writing an appendix at the end of your chapter or contribution. If there is only one the appendix is designated "Appendix", or "Appendix 1", or "Appendix 2", etc. if there is more than one.

$$a \times b = c \tag{4.4}$$

Problems

4.1 A given problem or Excercise is described here. The problem is described here. The problem is described here.

4.2 Problem Heading
(a) The first part of the problem is described here.
(b) The second part of the problem is described here.

References

1. S. Arora and B. Barak, *Computational Complexity: A Modern Approach*, Cambridge University Press, New York, NY, USA, 2009.
2. C. H. Papadimitriou, *Computational Complexity*, Addison-Wesley, Reading, MA, USA, 1994.
3. A. A. Razborov, "Lower bounds on the monotone complexity of some Boolean functions," *Mathematics of the USSR-Izvestiya*, vol. 31, no. 2, pp. 354-362, 1988. (Note: this is the translated version of the original 1985 paper that appeared in the journal *Mathematics of the USSR-Izvestiya*.)