Integer Complexity, Addition Chains, and Well-Ordering

by

Harry J. Altman

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Mathematics)
in the University of Michigan
2014

Doctoral Committee:

Professor Jeffrey C. Lagarias, Chair
Professor Alexander Barvinok
Professor Andreas R. Blass
Associate Professor Kevin J. Compton
Professor Martin J. Strauss

For my grandparents

Acknowledgments

The author is grateful to Joshua Zelinsky, who originally suggested the subject of integer complexity, and together with whom much of the work in Chapter 2 was conducted.

The author is grateful to Juan Arias de Reyna for much helpful discussion regarding integer complexity – improving notation, clarifying statements, and providing shorter proofs than the author's of the lower bound in Proposition 3.6.3 and the $k = 1$ case of Lemma 2.4.5.

The author is grateful to his advisor Jeffrey C. Lagarias, for suggesting the additional topic of addition chains, for pointing out the relations to computational complexity issues, and for much helpful discussion.

The author thanks in addition the following people: J. Iraids and K. Podnieks for providing much helpful numerical data; Andreas Blass, Paul Pollack, and Mike Bennett for suggesting references; E. H. Brooks for discussion regarding some of the proofs in Chapter 5; J. Heidi Soderstrom for help translating references into English; and A. Mishchenko for providing his LaTeX files for his dissertation, from which the formatting of this dissertation has largely been copied.

Preface

A note to the reader: The bulk of this thesis, Chapters 2 through 5, were originally written as separate papers (the paper which became Chapter 2 was co-authored with J. Zelinsky). As such, each has its own individual abstract in addition to the overall abstract, and the initial few sections of each chapter repeat much information from previous chapters. Appendix A was originally an appendix to Chapter 3, and Appendices B and C were originally appendices to Chapter 5.

Contents

List of Tables

List of Figures

List of Appendices

Abstract

In this dissertation we consider two notions of the "complexity" of a natural number, the first being addition chain length, and the second known simply as "integer complexity".

The *integer complexity* of n, denoted $\|n\|$, is the smallest number of 1's needed to write n using an arbitrary combination of addition and multiplication. It is known that $\|n\| \geq 3\log_3 n$ for all n.

We consider the difference $\delta(n) := \|n\| - 3\log_3 n$, which we call the *defect* of n. We consider the set of all defects – the set

$$\mathscr{D} := \{\delta(n) : n \in \mathbb{N}\}.$$

We show that, as a subset of the real numbers, $\mathscr{D}$ is well-ordered, with order type ω^ω; we also show the same for several variants of this set. Moreover, we show that, for $k \geq 1$ a natural number, $\mathscr{D} \cap [0, k)$ has order type precisely ω^k.

We also use the defect to prove stabilization results about $\|n\|$. Specifically, for any n, there exists $K = K(n)$ such that for $k \geq K$, we have

$$\delta(3^k n) = \delta(3^K n).$$

We call $K(n)$ the *stabilization length* of n.

Finally, we provide a way of, given $r > 0$, computing all numbers n with $\delta(n) < r$. We use this to show that the stabilization length $K(n)$ is effectively computable. The algorithm is also, empirically, much faster than existing methods for computing $\|2^k\|$, and we use it to prove that $\|2^k 3^\ell\| = 2k + 3\ell$ for $0 \leq k \leq 48$ and $\ell \geq 0$, with k and ℓ not both 0.

In parallel to our results for integer complexity, we also consider addition chain length. An *addition chain* for n is defined to be a sequence $(a_0, a_1, \ldots, a_r)$ such that $a_0 = 1$, $a_r = n$, and, for any k with $1 \leq k \leq r$, there exist $0 \leq i, j < k$ such that $a_k = a_i + a_j$; the number r is called the length of the addition chain. The shortest

length among addition chains for n, called the *addition chain length* of n, is denoted $\ell(n)$. The number $\ell(n)$ is always at least $\log_2 n$.

We consider the difference $\delta^\ell(n) := \ell(n) - \log_2 n$, which we call the *addition-chain defect* of n, and the set of all addition-chain defects

$$\mathscr{D}^\ell := \{\delta^\ell(n) : n \in \mathbb{N}\}.$$

We show that $\mathscr{D}^\ell$ is also a well-ordered set with order type ω^ω. We also use the defect to prove stabilization results about $\ell(n)$; specifically, for any n, there exists $K' = K'(n)$ such that for $k \geq K'$, we have

$$\delta(2^k n) = \delta^\ell(2^{K'} n).$$

Chapter 1

Introduction

1.1 Notions of complexity for natural numbers

In this dissertation we will consider the complexity of computing natural numbers under some simple computational models. When we speak of computing a natural number, we mean building it up in some finite number of steps from the number 1, which is the most basic of all natural numbers and generates all the others. There are various models of computation we could turn our attention to, but we will focus on two: One is known as "integer complexity" (Section 1.1.1), and the other is that of addition chains (Section 1.1.2). Some of the others will be briefly discussed in Section 1.4.

In all these cases, we are discussing building up natural numbers from the number 1; what we vary is what tools are allowed. Of course, every natural number n can be written as the sum of n ones, and if we only allow the use of addition, it is impossible to do better, so this is not a very interesting model; something more is needed to allow shorter, less obvious ways of writing n.

Thus we define the *integer complexity* of a natural number n to be the least number of 1's needed to write it using any combination of addition and multiplication, with the order of the operations specified using parentheses grouped in any legal nesting. For instance, 11 has complexity of 8, since it can be written using 8 ones as

$$11 = (1 + 1 + 1)(1 + 1 + 1) + 1 + 1,$$

but not with any fewer. This notion was implicitly introduced in 1953 by Kurt Mahler and Jan Popken [38]; they actually considered an inverse function, the size of the largest number representable using k copies of the number 1. (More generally, they considered the same question for representations using k copies of a positive real number x.) Integer complexity was explicitly studied by John Selfridge, and was later

popularized by Richard Guy [29, 30]. Following J. Arias de Reyna [8] we will denote the complexity of n by $\|n\|$.

A second model of complexity is that of addition chains. An *addition chain* for the number n is defined to be a sequence $(a_0, a_1, \ldots, a_r)$ such that $a_0 = 1$, $a_r = n$, and, for any k with $1 \leq k \leq r$, there exist $0 \leq i, j < k$ such that $a_k = a_i + a_j$; the number r is called the length of the addition chain. The shortest length among addition chains for n, called the *addition chain length* of n, is denoted $\ell(n)$. Addition chains were introduced in 1894 by H. Dellac [22] and reintroduced in 1937 by A. Scholz [41], who raised a series of questions about them.

We could consider other similar notions, such as allowing addition, multiplication, *and* free reuse (see Section 1.4), but this dissertation will focus on these two, which share a number of similarities – some obvious, others less so.

Both integer complexity and addition chain length seem to be moderately hard to compute. Let us define the following computational problems:
INTEGER COMPLEXITY

- INSTANCE: Positive integers n and k, both encoded in binary.
- QUESTION: Is $\|n\| \leq k$?

ADDITION CHAIN LENGTH

- INSTANCE: Positive integers n and k, both encoded in binary.
- QUESTION: Is $\ell(n) \leq k$?

Both problems are known to be in the complexity class NP [8, 24] (and so in particular they are computable in exponential time), but neither is known to be in P (or even *co-*NP), nor is either known to be NP-complete. However, an extension of the problem ADDITION CHAIN LENGTH is known to be NP-complete, as will be discussed in Section 1.1.2. So both integer complexity and addition chain length have the property that to verify an upper bound is easy, but to verify a lower bound seems to be hard.

By better understanding the structure of integer complexity and addition chain length, we can find new ways of lower-bounding them. In this dissertation, we will explore the structure of integer complexity and addition chains, by making use of functions we call the *defect*; see Sections 1.2.1 and 1.3.1. We will find much regularity in the set of values that the defect takes on, and, in Chapter 5, we will use it to provide a new algorithm for computing $\|n\|$ (as well as more detailed information), one which,

unlike existing approaches, does not require first computing the complexities of all the numbers up to n.

Both addition chains and integer complexity have some specific outstanding open problems. For addition chain length, an open problem is the *Scholz-Brauer conjecture*, ([41, Question 3]), which asserts that

$$\ell(2^n - 1) \leq n + \ell(n) - 1;$$

see Section 1.1.2 for more on the history of this question. For integer complexity, an outstanding problem is that of whether, for all $k \geq 1$,

$$\|2^k\| = 2k;$$

it is clear that $\|2^k\| \leq 2k$, so the question is one of the lower bound.

It's worth noting here that this is not an isolated question, unconnected to the larger internal structure of integer complexity. A deeper outstanding question regarding integer complexity is understanding the values of the function $\frac{\|n\|}{\ln n}$, and, in particular, determining $\limsup_{n\to\infty} \frac{\|n\|}{\ln n}$; if indeed $\|2^k\| = 2k$ for $k \geq 1$, that would require this limit to be at least $\frac{2}{\ln 2}$. See Section 1.1.1 for more on this.

Now let us examine the particulars of integer complexity and addition chains in more detail.

1.1.1 Integer complexity

The *complexity* of a natural number n is the least number of 1's needed to write it using any combination of addition and multiplication, with the order of the operations specified using parentheses grouped in any legal nesting. Following J. Arias de Reyna [8] we will denote the complexity of n by $\|n\|$.

Notice that for any natural numbers n and m we will have

$$\|1\| = 1, \quad \|n + m\| \leq \|n\| + \|m\|, \quad \|nm\| \leq \|n\| + \|m\|,$$

More specifically, for any $n > 1$, we have

$$\|n\| = \min_{\substack{a,b<n\in\mathbb{N} \\ a+b=n \text{ or } ab=n}} (\|a\| + \|b\|).$$

This fact together with $\|1\| = 1$ allows one to compute $\|n\|$ recursively by dynamic

programming. If the equality $\|n\| = \|a\| + \|b\|$ holds, with either $n = a+b$ or $n = ab$, then we will say n can be written *most-efficiently* as $a + b$ or as ab, respectively.

Integer complexity is approximately logarithmic; it satisfies the bounds

$$3 \log_3 n \leq \|n\| \leq 3 \log_2 n, \qquad n > 1.$$

The upper bound can be obtained by writing n in binary and finding a representation using Horner's algorithm. The lower bound follows from results described below. The lower bound is known to be attained infinitely often, namely for all $n = 3^k$. The constant in the upper bound above can be improved further [52], and it is an open problem to determine the true asymptotic order of magnitude of the upper bound. At present even the possibility that an asymptotic formula $\|n\| \sim 3 \log_3 n$ might hold has not been ruled out.

Let $E(k)$ be the largest number writable with k ones, i.e., with complexity at most k. John Selfridge (see [29]) proved that $E(1) = 1$ and that the larger values depend on the residue class of k modulo 3, namely for $k = 3j + i \geq 2$,

$$\begin{aligned} E(3j) &= 3^j \\ E(3j+1) &= 4 \cdot 3^{j-1} \\ E(3j+2) &= 2 \cdot 3^j \end{aligned}$$

Observe that $E(k) \leq 3^{k/3}$ in all cases, and that equality holds for cases where 3 divides k. These formulas also show that $E(k) > E(k-1)$, a fact that implies that the integer $E(k)$ requires exactly k ones. This yields the following result:

Theorem 1.1.1. *For $k = 0, 1, 2$ and for all $\ell \geq 0$ with $k + \ell \geq 1$, one has*

$$\|2^k \cdot 3^\ell\| = 2k + 3\ell.$$

This suggests the following conjecture, which was originally formulated as a question in Guy [29].

Conjecture 1.1.2. *For all $k \geq 0$ and all $\ell \geq 0$ with $k + \ell \geq 1$ there holds*

$$\|2^k 3^\ell\| = 2k + 3\ell.$$

Note that this conjecture would in particular imply $\|2^k\| = 2k$, for all k. Selfridge raised this special case in a contrary form, asking the question whether there is some

k for which $\|2^k\| < 2k$ (see [29]). The truth of $\|2^k\| = 2k$ would also immediately imply the lower bound

$$\limsup_{n\to\infty} \frac{\|n\|}{\ln n} \geq \frac{2}{\ln 2}.$$

Computer experiments seem to agree with this prediction and even allow the possibility of equality; see Iraids et al [33]. Meanwhile, as mentioned above, if we define

$$C_{max} := \limsup_{n\to\infty} \frac{\|n\|}{\ln n},$$

the following question remains open:

Question 1.1.3. *Is $C_{max} = \frac{3}{\ln 3}$, or is $C_{max} > \frac{3}{\ln 3}$? That is to say, is $\|n\| \sim 3\log_3 n$, or not?*

As such, proving that $\|2^k\| = 2k$ for all $k \geq 1$, let alone Conjecture 1.1.2, would be a very strong result. And indeed others have suggested the opposite, that $\|2^k\| < 2k$ for some k [10], or even [29] that $\|n\| \sim 3\log_3 n$.

Proving that $C_{max} \leq \frac{2}{\ln 2}$ would probably also be quite difficult. At present, all known upper bounds on $\|n\|$ that hold for sufficiently large n also hold for all $n > 1$. (The largest value of $\frac{\|n\|}{\ln n}$ seen to occur so far is at $n = 1439$, $\|n\| = 26$, for a value of approximately 3.5755; this is the largest among all n with $n \leq 10^{12}$ [33].) However, better bounds are known if we do not insist the bound hold for all sufficiently large n, and only insist they hold on a set of density 1. Let us define

$$C_{most} := \inf\{C \in \mathbb{R} : \|n\| \leq C \ln n \text{ for } n \text{ on a set of density } 1\},$$

so that we have $\frac{3}{\ln 3} \leq C_{most} \leq C_{max} \leq \frac{3}{\ln 2}$. Then it was shown by J. Arias de Reyna and J. Van de Lune [9] that

$$C_{most} \leq \frac{41747875}{2^7 3^8 \ln(2^9 3^8)} < 3.309.$$

Notably, this is smaller than the largest seen value that is $\frac{26}{\log 1439}$, which the best bounds on C_{max} still are not.

Let us also take a moment to remark on computing $\|n\|$. The recursive definition permits computing $\|n\|$ by dynamic programming, but it requires knowing $\{\|k\| : 1 \leq k \leq n-1\}$, so takes exponential time in the input size of n measured in bits. In particular, a straightforward approach to computing $\|x\|$ requires on the order of n^2 steps. Srinivas and Shankar [44] obtained an improvement on this, running in time

$O(n^{\log_2 3})$, and Arias de Reyna and Van de Lune [9] further improved this to $O(n^{1.231})$. As we noted above, the problem "Given n and k in binary, is $\|n\| \leq k$?" is known to be in the complexity class NP [8], but it is neither known to be in P, nor known to be NP-complete.

Finally, we also mention the work of J. Arias de Reyna[8] in 2000 which formulated a series of conjectures about the structure of $\|\cdot\|$. Some of these are discussed in Appendix A.

1.1.2 Addition chains

An *addition chain* for the number n is defined to be a sequence $(a_0, a_1, \ldots, a_r)$ such that $a_0 = 1$, $a_r = n$, and, for any k with $1 \leq k \leq r$, there exist $0 \leq i, j < k$ such that $a_k = a_i + a_j$; the number r is called the length of the addition chain. The shortest length among addition chains for n, called the *addition chain length* of n, is denoted $\ell(n)$. Addition chains were introduced in 1894 by H. Dellac [22] and reintroduced in 1937 by A. Scholz [41], who raised a series of questions about them. They have been much studied in the context of computation of powers, since an addition chain for n of length r allows one to compute x^n from x using r multiplications. Extensive surveys on the topic can be found in Knuth [35, Section 4.6.3] and Subbarao [46]. It is common to restrict discussion to addition chains which are increasing, as among the shortest addition chains for a given number there necessarily can be found one with this property.

To put it another way, with integer complexity, we were allowed the use of both addition and multiplication, but had to pay the full cost of a number each time it was used. With addition chains, by contrast, we are not allowed multiplication, but we are allowed free reuse – once we construct a number, we may reuse it as often as we please at no additional cost.

Addition chain length is approximately logarithmic; it satisfies the bounds

$$\log_2 n \leq \ell(n) \leq \lfloor \log_2 n \rfloor + \nu_2(n) - 1,$$

in which $\nu_2(n)$ counts the number of 1's in the binary expansion of n. The lower bound follows from the observation that the largest number that can be made with an addition chain of k steps is 2^k, since each step can at most double the previous number. The upper bound follows from writing n using the "binary method", which can be defined recursively: The binary chain for $2n$ is the binary chain for n followed by $2n$, and the binary chain for $2n+1$ is the binary chain for $2n$ followed by $2n+1$;

this chain has length $\lfloor \log_2 n \rfloor + \nu_2(n) - 1$. In fact, A. Brauer [15] proved in 1939 that $\ell(n) \sim \log_2 n$. This stands in contrast to the case of integer complexity, as $\limsup_{n\to\infty} \frac{\|n\|}{\ln n}$ remains unknown.

In addition to being a measure of complexity that is logarithmic in growth, addition chain length also has a similarity to integer complexity in that it satisfies $\ell(nm) \leq \ell(n) + \ell(m)$, as if one has addition chains $(a_0, \ldots, a_r)$ and $(b_0, \ldots, b_s)$, one may make an addition chain $(a_0, \ldots, a_r, a_r b_1, a_r b_2, \ldots, a_r b_s)$. However, this similarity will mostly not be relevant here.

The addition chain complexity function $\ell(n)$ seems complicated and hard to compute. An outstanding open problem about it is the *Scholz-Brauer conjecture* ([41, Question 3]):

Conjecture 1.1.4 (Scholz, Brauer)**.** *For any $n \geq 0$,*

$$\ell(2^n - 1) \leq n + \ell(n) - 1.$$

Partly in order to investigate this conjecture, A. Brauer[15] introduced a restricted form of addition chains known as *star chains*. An addition chain $(a_0, \ldots, a_r)$ is called a star chain if for each nonzero $k \leq r$, there is some $i < k$ such that $a_k = a_{k-1} + a_i$. The length of the shortest star chain for n is denoted $\ell^*(n)$. Brauer showed that

$$\ell^*(2^n - 1) \leq n + \ell^*(n) - 1,$$

so if $\ell(n) = \ell^*(n)$, i.e. if there is a shortest addition chain for n which is a star chain, then the Scholz-Brauer conjecture holds for n. Such a number is known as a Brauer number. One might hope that all natural numbers are Brauer numbers; however, Hansen [31] showed that there are in fact infinitely many non-Brauer numbers.

In an attempt to salvage Brauer's conjecture, Hansen defined a generalization of the star chain, which he called an *ℓ^0-chain* and which is now also known as a *Hansen chain*. A Hansen chain is an addition chain $(a_0, \ldots, a_r)$ such that there some is subset $S \subseteq \{0, \ldots, r\}$ such that for each $0 < k \leq r$, there is some $i < k$ such that $a_k = a_i + a_j$, where j is the largest element of S that is less than k. So a star chain is simply a Hansen chain with $S = \{0, \ldots, r\}$. The length of the shortest Hansen chain for n is denoted $\ell^0(n)$, and one has for Hansen chains the inequality

$$\ell^0(2^n - 1) \leq n + \ell^0(n) - 1,$$

so if $\ell(n) = \ell^0(n)$ (in which case n is called a *Hansen number*), the Scholz-Brauer

conjecture holds for n. Computations of Clift [18] have verified that all $n < 5784689$ are Hansen numbers, but 5784689 is not; this remains the smallest n for which the Scholz-Brauer conjecture remains unknown.

Another difference worth noting between addition chains and integer complexity is that addition chain length cannot be computed via dynamic programming. Suppose we have a shortest addition chain $(a_0, \ldots, a_{r-1}, a_r)$ for n; one might hope that $(a_0, \ldots, a_{r-1})$ is a shortest addition chain for a_{r-1}, but this need not be the case. An example is provided by the addition chain $(1, 2, 3, 4, 7)$; this is a shortest addition chain for 7, but $(1, 2, 3, 4)$ is not a shortest addition chain for 4, as $(1, 2, 4)$ is shorter. Moreover, there is no way to assign to each natural number n a shortest addition chain $(a_0, \ldots, a_r)$ for n such that $(a_0, \ldots, a_{r-1})$ is the addition chain assigned to a_{r-1} [35]. This can be an obstacle both to computing addition chain length and to proving statements about addition chains. Despite this, there has been considerable work on algorithms in practice for computing addition chain length[12, 48].

As we noted above, the question "Given n and k in binary, is $\ell(n) \leq k$?" is known to be in the complexity class NP, but it is neither known to be in P, nor known to be NP-complete. However, an extension of the problem is NP-complete [24]:

Theorem 1.1.5 (P. Downey, B. Leong, R. Sethi)**.** *Consider the problem of, given numbers $n_1, \ldots, n_r$ and k, represented in binary, determining whether there is an addition chain of length at most k that includes all the numbers $n_1, \ldots, n_r$. This problem is NP-complete.*

1.2 Main results: Integer complexity

1.2.1 The defect and the defect set

In this dissertation, we will study integer complexity by subtracting off Selfridge's lower bound. Specifically:

Definition 1.2.1. The (integer complexity) *defect* of a natural number n is given by

$$\delta(n) = \|n\| - 3\log_3 n.$$

This might seem to be an unnatural object of study, compared to the ratio $\frac{\|n\|}{3\log_3 n}$. However, its set of values turns out to have very nice structure. First, let us introduce some notation for it.

Definition 1.2.2. The (integer complexity) *defect set*, denoted $\mathscr{D}$, is the set of all defects, $\{\delta(n) : n \in \mathbb{N}\}$.

Then we can state our first main result:

Theorem 1.2.3 (Well-ordering theorem for integer complexity)**.** *As a subset of $[0, \infty)$, the set $\mathscr{D}$ is a well-ordered set, with order type ω^ω. Moreover, for $k \geq 1$ an integer, the set $\mathscr{D} \cap [0, k)$ has order type precisely ω^k.*

This well-ordering of the defect set $\mathscr{D}$ reveals new fundamental structure in the interaction between addition and multiplication. Some of the tangledness of that interaction may be reflected in how the set $\mathscr{D}$ grows more complicated as its elements get larger.

1.2.2 Stabilization for integer complexity

Since $\|3^k\| = 3k$ for $k \geq 1$, while integer complexity of other other powers is hard to determine, one might hope that one has $\|3n\| = \|n\| + 3$ for all $n > 1$. Unfortunately, this is not the case; for instance, $\|107\| = 16$ while $\|321\| = 18$. Nonetheless, a weaker version of this does hold:

Theorem 1.2.4 (Stabilization theorem for integer complexity)**.** *For any natural number n, there exists a $K = K(n)$ such that for any $k \geq K$, one has*

$$\|3^k n\| = 3(k - K) + \|3^K n\|.$$

Moreover, $K(n)$ is effectively computable.

Proving that $K(n)$ is computable turns out to be substantially more difficult than proving it exists. We will prove that K exists in Chapter 2, and show that it is computable in Chapter 5; see also Section 1.2.5.

Because of this theorem, we will make the following defintion:

Definition 1.2.5. We say a natural number n is *stable* if, for all $k \geq 0$,

$$\|3^k n\| = 3k + \|n\|.$$

So then Theorem 1.2.4 says that for any n, there exists $K(n)$ such that $3^{K(n)}n$ is stable; the smallest such K will be called the *stabilization length* of n. We also make the following definition:

Definitions 1.2.6. The *stable complexity* of n, denoted $\|n\|_{st}$, is defined to be $\|3^k n\| - 3k$, where k is chosen such that $3^k n$ is stable. We also define $\Delta(n) = \|n\| - \|n\|_{st}$.

That is to say, the stable complexity of n is what the complexity of n would be "if n were stable"; it's equal to $\|n\|$ if and only if n is stable.

Empirically, it seems that most numbers are stable, but a positive fraction, around 3%, are unstable. Still, it is difficult to find examples that are "far from stable", in that either $K(n)$ or $\Delta(n)$ are large. Exactly computing $K(n)$ or $\Delta(n)$, while possible with the algorithms in Chapter 5, is slow, but from computations of $\|n\|$ we can at least put lower bounds on these quantities. Looking at $n \leq 3^{15}$, we find 17 numbers, such as 3643, which must have $K(n) \geq 5$ (and $\Delta(n) \geq 1$); no examples are presently known that demonstrate that K can ever be at least 6. We also find the example of $n = 4721323$, which has the surprising property $\|3n\| < \|n\|$, and so must have $\Delta(n) \geq 4$ (and $K(n) \geq 1$); no examples are presently known that demonstrate that Δ can ever be at least 5. Whether $K(n)$ and $\Delta(n)$ can be arbitrarily large, or whether they each have some uniform upper bound, remains unknown.

1.2.3 Variations on the defect set

We can also define some variations on the defect set. One thing we can do is restrict to defects of stable numbers. We will show in Chapter 3 that if $\delta(n) = \delta(m)$, then n is stable if and only if m is stable. Thus it makes sense to talk about "stable defects". Thus we can define $\mathscr{D}_{st}$, the set of stable (integer complexity) defects. We will see that this set too is well-ordered with order type ω^ω.

We can discriminate even further. We will show in Chapter 2 that if $\delta(n) = \delta(m)$, then $\|n\| \equiv \|m\| \pmod 3$. It follows that we can split the set of defects $\mathscr{D}$ into sets $\mathscr{D}^0$, $\mathscr{D}^1$, $\mathscr{D}^2$ according to these congruence classes modulo 3. We will prove in Chapter 3 that these sets too are well-ordered with order type ω^ω. We can combine this idea with that of restricting to stable defects, forming sets $\mathscr{D}^0_{st}$, $\mathscr{D}^1_{st}$, $\mathscr{D}^2_{st}$; these too will turn out to be well-ordered with order type ω^ω.

1.2.4 Numbers of small defect and low-defect polynomials

The basic method in all of this is to restrict the form that numbers of small defect can take. We will do this by showing that any number of small defect can be represented by substituting powers of 3 into certain multilinear polynomials we call *low-defect polynomials*.

In Chapter 2, we will provide a method where, if we know all the numbers of defect

less than some $\alpha < 1$, we can use this to successively determine all the numbers of defect less than $k\alpha$ for every k. Of course, by itself, this is useless; the method needs a starting point. But we will show, using a result of Rawsthorne [39], that the only numbers with defect less than $\delta(2) = 0.107\ldots$ are the powers of 3. With this, we can apply the method repeatedly until we know all the numbers with defect less than 1. Thus the method "pulls itself up by its own bootstraps", as once we know all numbers of defect less than 1, we can apply the method with any step size $\alpha < 1$. Once we know this, we will be able to use the method to demonstrate that numbers below a fixed defect are quite scarce:

Theorem 1.2.7 (Counting numbers of small defect)**.** *For real numbers r and x, let $A_r(x)$ denote the number of natural numbers $n \leq x$ such that $\delta(n) < r$. Then for any $r > 0$,*

$$A_r(x) = \Theta_r((\log x)^{\lfloor r \rfloor}).$$

However, much of the power of the method will not be demonstrated until Chapter 3, where we show that the output of this method has a tractable form. There we show that for any $s > 0$, there exists a finite set of low-defect polynomials $\mathcal{S}_s$ such that any number of defect less than s can be written as $f(3^{n_1}, \ldots, 3^{n_k})3^{n_{k+1}}$ for some $f \in \mathcal{S}_s$ and nonnegative $n_1, \ldots, n_{k+1}$. However, using solely the methods of Chapter 3, the low-defect polynomials may also produce extraneous numbers, with defect higher than intended. In Chapter 5 we will demonstrate how to modify this construction so that it no longer produces extraneous numbers; then we will be able to say that $\delta(N) < s$ if and only if N can be written as $f(3^{n_1}, \ldots, 3^{n_k})3^{n_{k+1}}$ for some $f \in \mathcal{S}_s$ and nonnegative $n_1, \ldots, n_{k+1}$. (We will say that N can be 3-*represented* by f; or, more properly, when $n_{k+1} > 0$, by $\hat{f}$, which we will define later.)

So with this approach, we can get at properties of the set of defects by examining properties of low-defect polynomials. For instance, as the defects involved get larger, the low-defect polynomials required get more complicated; one way in which this occurs is that they require more variables. In fact, we will see that to cover defects up to a real number s, one needs low-defect polynomials with up to $\lfloor s \rfloor$ variables. And it happens that if we have a low-defect polynomial f in k variables, and consider the numbers $f(3^{n_1}, \ldots, 3^{n_k})$, then the defects of the numbers obtained this way form a well-ordered set of order type at least ω^k and less than ω^{k+1}. It is this that leads us to Theorem 1.2.3, that for $k \geq 1$, the set $\mathscr{D} \cap [0, k)$ has order type precisely ω^k – and hence that the set $\mathscr{D}$ has order type ω^ω.

1.2.5 Algorithms and computational results

In Chapter 5 we present a series of algorithms for working with and computing low-defect polynomials and for extracting from them information about integer complexity. We describe here what these algorithms do and some of the computations we have performed with our implementation of them in a series of Haskell programs.

In Section 5.6 we will show that not only does a set $\mathcal{S}_s$ exist, but we can effectively compute it. Algorithm 1 provides the base case $\mathcal{S}_\alpha$; Algorithms 2 and 3 allow this to be built up to $\mathcal{S}_s$; and Algorithms 4, 5, and 6 allow one to additionally choose $\mathcal{S}_s$ so as not to generate extraneous numbers.

By making use of these, we then describe Algorithm 7 and Algorithm 8, which provide a method of computing the quantity $K(n)$ from Theorem 1.2.4 and thus prove that it is computable.

But these methods can be used not only to compute $K(n)$ but also to simultaneously compute $\|n\|$; Algorithm 9 computes both of these. Algorithm 10 is an optimized version for when n is a power of 2 – which, due to Conjecture 1.1.2, is a case of some interest. In general, using Algorithm 9 is not faster than computing $\|n\|$ by existing methods, but the special case of Algorithm 10 turns out to be much faster than previous methods. By using it, we are able to prove by the following theorem:

Theorem 1.2.8. *If* $0 \leq k \leq 48$ *and* $\ell \geq 0$ *with* $k + \ell > 0$, $\|2^k 3^\ell\| = 2k + 3\ell$.

This is a substantial improvement over the existing theorem that this is true for $1 \leq k \leq 2$, as well as over the results in this direction of Iraids et. al. [33], who, with a much longer computation, were able to verify that (for k and ℓ not both zero) $\|2^k 3^\ell\| = 2k + 3\ell$ whenever $2^k 3^\ell \leq 10^{12}$. (And so in particular $\|2^k\| = 2k$ for $1 \leq k \leq 39$.) Notably, compared to [33], we did not have to lower k or increase computation time in order to obtain that ℓ may be arbitrary, but in fact increased k and shortened computation time.

1.3 Main results: Addition chains

In this section, we present our main results regarding addition chains. These are largely parallel to our results regarding integer complexity mentioned above, even though in some cases entirely different methods of proof are needed.

1.3.1 The addition chain defect and its defect set

As with integer complexity, we will also study addition chains by subtracting off the corresponding lower bound. Specifically:

Definition 1.3.1. The *addition chain defect* of a natural number n is given by

$$\delta^\ell(n) = \ell(n) - \log_2 n.$$

With this we can then define:

Definitions 1.3.2. The *addition chain defect set*, denoted $\mathscr{D}^\ell$, is the set of all addition chain defects, $\{\delta^\ell(n) : n \in \mathbb{N}\}$.

Then, as in the case of integer complexity, we have:

Theorem 1.3.3 (Well-ordering theorem for addition chains)**.** *As a subset of* $[0, \infty)$*, the set* $\mathscr{D}^\ell$ *is a well-ordered set, with order type* ω^ω*.*

The addition chain version of the theorem is visibly weaker than the integer complexity version of the theorem; still, we may turn the analogy into a conjecture:

Conjecture 1.3.4. *For* $k \geq 1$ *an integer, the set* $\mathscr{D}^\ell \cap [0, k)$ *has order type precisely* ω^k*.*

We are able to recover the missing part of the theorem at least partly. To do so, let us make the following definition:

Definition 1.3.5. We define $f^\ell(k)$ to be the limit of the initial ω^k defects in $\mathscr{D}^\ell$.

Then while we cannot currently prove that $f^\ell(k) = k$ for all k, we can show:

Theorem 1.3.6 (Order type bounds for addition chains)**.** *For* k *a whole number, we have:*

1. *For* $0 \leq k \leq 2$*, we have* $f^\ell(k) = k$*.*
2. *For* $3 \leq k \leq 7$*, we have* $2 < f^\ell(k) \leq k$*.*
3. *For* $8 \leq k \leq 33$*, we have* $3 < f^\ell(k) \leq k$*.*
4. *For* $k \geq 34$*, we have* $\log_2(k+1) - 2.13 < f^\ell(k) \leq k$*.*

In fact, study of the defect for addition chains is not entirely new; other authors ([35, 27, 47, 50]), investigating the behavior of $\ell(n)$, and in particular the Scholz-Brauer conjecture, have studied the quantity $s(n) := \ell(n) - \lfloor \log_2 n \rfloor$, which Knuth [35] calls the number of *small steps* of n. This is plainly a rounded off version of the defect; it is related to our notion of defect by

$$s(n) = \lceil \delta^\ell(n) \rceil.$$

As Theorem 1.3.3 shows, however, quite a lot of structure is lost in rounding off the defect.

1.3.2 Stabilization for addition chains

We can also consider stabilization for addition chains. As $\|3^k\| = 3k$ for $k \geq 1$, so $\ell(2^k) = k$ for $k \geq 0$, and so one might hope that in general $\ell(2n) = \ell(n) + 1$; again, this is not so. Indeed, Thurber[50] showed that all numbers of the form $23 \cdot 2^k + 7$, for $k \geq 5$, are counterexamples to this. But once again, we do have a stabilization form of this:

Theorem 1.3.7 (Stabilization theorem for addition chains)**.** *For any natural number n, there exists a $K = K_\ell(n)$ such that for any $k \geq K$, one has*

$$\ell(2^k n) = k - K + \ell(2^K n).$$

Unlike in the integer complexity case, we do not know whether $K_\ell(n)$ is effectively computable; see Chapter 6.

1.3.3 Variations on the addition chain defect set

In Chapter 4, we will see that if $\delta^\ell(n) = \delta^\ell(m)$, then n is ℓ-stable if and only if m is ℓ-stable, parallel to the results mentioned in Section 1.2.3. Thus, as we can define $\mathscr{D}_{st}$ for integer complexity, we can define $\mathscr{D}^\ell_{st}$, the set of stable addition chain defects. We will see that this set too is well-ordered with order type ω^ω.

However, the decomposition of the set $\mathscr{D}$ into $\mathscr{D}^0$, $\mathscr{D}^1$, and $\mathscr{D}^2$ has no analogue for $\mathscr{D}^\ell$. For instance, for any whole number k one has $\ell(2^k) = k$, even though one always has $\delta^\ell(2^k) = 0$.

1.3.4 Numbers of small defect and binary digit sums

As with integer complexity, the basic method we present for dealing with addition chains is again to restrict the form that numbers of small defect can take. In the case of addition chains, unfortunately, we cannot at present get as good control as we can in the case of integer complexity. Nonetheless, by making use of the following theorem of Schönhage[42], we can recover quite a bit:

Theorem 1.3.8 (Schönhage)**.** *For any $n \geq 1$,*

$$\delta^{\ell}(n) \geq \log_2 \nu_2(n) - C_s,$$

where

$$C_s := \frac{2}{3} + \frac{2}{3}\log_2 3 - \frac{1}{\ln 2} - \log_2 \log \frac{4}{3} + \sum_{k=0}^{\infty} \log_2(1 + 2^{-6\cdot 2^k+1}) \leq 2.13.$$

Here, $\nu_2(n)$ denotes the number of 1's in the binary expansion of n. If we like, we can extend the analogy with integer complexity by thinking of this as saying that if $\delta^{\ell}(n) < r$, then n can be "2-represented" by one of the polynomials

$$((((x_1 + 1)x_2 + 1)\ldots)x_k + 1)x_{k+1},$$

where $0 \leq k \leq \lfloor 2^{r+C_s} \rfloor - 1$. However, unlike in the integer complexity case, there is no guarantee that such a representation need be "most-efficient" (which here would mean that $\ell(n) = \lfloor \log_2 n \rfloor + \nu_2(n) - 1$). This unfortunately limits what can be proven by these means. Still, it is enough to prove Theorem 1.3.3.

Note that while Theorem 1.3.8 is the best known bound of this type when $\nu_2(n)$ is large, there are better bounds known when $\nu_2(n)$ is small. We will give an accounting of these in Chapter 4. Theorem 1.3.6 is based on Theorem 1.3.8 when k is large, and these various other results when k is small.

1.4 Other notions of complexity

1.4.1 Addition-multiplication chains

As mentioned in Section 1.1.2, while addition chains, unlike integer complexity, do not allow for the use of multplication, they do allow for the free reuse of numbers already constructed. An obvious extension then is the notion of *addition-multiplication*

chains. An addition-multiplication chain is a sequence $(a_0, \ldots, a_r)$ where $a_0 = 1$, and for $k > 1$, each a_k can either be written as $a_i + a_j$ or $a_i \cdot a_j$ for some $0 \leq i, j < k$. We say that $(a_0, \ldots, a_r)$ is a chain for a_r, and r is its length; the length of the shortest addition-multiplication chain for n is denoted $\ell_{AM}(n)$. For $n \geq 2$, one has

$$\ell_{AM}(n) \geq \log_2 \log_2 n + 1;$$

however, no corresponding $\Theta(\log \log n)$ upper bound holds. W. De Melo and B. F. Svaiter showed [23] that, if $\tau(n)$ denotes the length of the shortest addition-multiplication-subtraction chain for n (see below), then for any $\varepsilon > 0$, one has, for n outside a set of density 0,

$$\tau(n) \geq \frac{\ln n}{(\ln \ln n)^{1+\varepsilon}},$$

and hence the same holds for ℓ_{AM}.

Despite this disanalogy regarding the bounds, it may be possible to prove results for addition-multiplication chains similar to the ones in this dissertation for addition chains and for integer complexity. If we define, for $n \geq 2$,

$$\delta_{AM}(n) = \ell_{AM}(n) - \log_2 \log_2 n - 1,$$

then it is plausible that the image of δ_{AM} could be well-ordered. At present the question of whether this is so remains unanswered.

1.4.2 Models allowing exponentiatoin

One could also increase the set of operations available beyond addition and multiplication. If one simply continues up the chain of hyper operations, the next step would be to allow exponentiation as well. Models allowing for exponentiation are also not good for studying defects, however, because the lower bound grows as $\log^* n$ (or slower), and it's not clear how to make a continuous version of this.

1.4.3 Models allowing subtraction or division

One could also expand in another direction and allow non-monotonic operations, such as subtraction. One could expand any of the above models by allowing subtraction, considering addition-subtraction chains [35, 42], or integer complexity allowing subtraction, or even addition-multiplication-subtraction chains. (Addition-multiplication-subtraction chains, it turns out, are related to the P vs. NP problem

for Blum-Shub-Smale machines [13].) One could even allow for the use of both subtraction and exponentiation, or one could go further than subtraction and allow forms of division [14].

Unfortunately, the non-monotonicity of subtraction breaks well-ordering. For instance, if we let $\ell^{\pm}(n)$ denote the length of the shortest addition-subtraction chain for n, then one can easily see that for $k \geq 3$,

$$\ell^{\pm}(2^k - 1) = k + 1.$$

Thus, if one were to define the addition-subtraction chain defect

$$\delta^{\pm}(n) := \ell^{\pm}(n) - \log_2 n,$$

then one would find that the image of this function contains the infinite decreasing sequence $1 - \log_2(1 - 2^{-k})$. It follows that the set of all addition-subtraction chain defects is not well ordered with respect to the usual ordering of the real line. Observe that well-ordering fails here even though the theorem of Schönhage we use to prove well-ordering for addition chain defects (see Section 1.3.4) has an analogue for addition-subtraction chains [42].

Similarly, if we were to define $\|n\|_-$ to be the complexity of n with subtraction allowed, it is easy to check that for $k \geq 3$, we have

$$\|3^k - 1\|_- = 3k + 1,$$

and so if we were to define

$$\delta_-(n) := \|n\|_- - 3\log_3 n,$$

then the image of this function would contain the infinite decreasing sequence $1 - 3\log_3(1 - 3^{-k})$, and so again would not be well-ordered.

1.4.4 Other complexity measures of trees

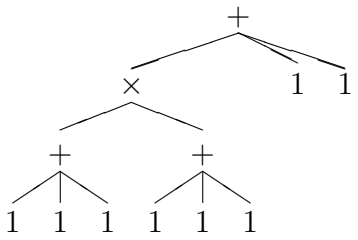
Finally, one could also consider addition chains or $+, \cdot, 1$-expressions as above, but measure their complexity in a different way. For instance, a $+, \cdot, 1$-expression can be visualized as a tree; see Figure 1.1. One could consider addition and multiplication as binary operations and only allow binary trees, or one could consider them as multiary operations and allow vertices of higher degree. Based on this, J. Iraids et al. [33]

considered a quantity they denoted $rank(n)$, which they defined to be the minimum height of a (not necessarily binary) tree corresponding to a shortest expression for n. (Note that we must restrict to a shortest expression for n in order for the notion to be non-trivial; otherwise the rank would always be 0.) They proved furthermore that $rank(n)$ is related to $\delta(n)$ by the inequality

$$\delta(n) \geq \left\lfloor \frac{rank(n)-1}{2} \right\rfloor \left(1 + 3\log_3 \frac{6}{7}\right);$$

it may be possible to do more in this direction, but we will not further consider this notion here.

Figure 1.1: A tree for $n = 11$ using 8 ones and of height 3, demonstrating $rank(11) = 3$.



1.5 Plan of this thesis

Chapter 2 presents the basic method of, given r, determining all numbers n with $\delta(n) < r$. In it we prove Theorem 1.2.7, prove part of Theorem 1.2.4, and verify that $\|2^k 3^\ell\| = 2k + 3\ell$ for $m \leq 21$ with k and ℓ not both zero. A version of Chapter 2 previously appeared in volume 12 of *Integers* [7].

Chapter 3 introduces low-defect polynomials and proves the existence of the set $\mathcal{S}_s$ discussed in Section 1.2.4. It then uses this to prove Theorem 1.2.3.

Chapter 4 discusses addition chains and proves Theorems 1.3.3 and 1.3.7.

Chapter 5 shows how, as discussed in Section 1.2.4, the set $\mathcal{S}_s$ may be chosen so as not to yield extraneous numbers. It then discusses computational issues, providing ten algorithms, which can be used to prove the rest of Theorem 1.2.4 as well as Theorem 1.2.8.

Chapter 6 discusses open problems and possible directions of future research regarding integer complexity, addition chains, and other notions of complexity for natural numbers.

Chapter 2

Numbers with Integer Complexity Close to the Lower Bound

Abstract: Define $\|n\|$ to be the *complexity* of n, the smallest number of 1's needed to write $\|n\|$ using an arbitrary combination of addition and multiplication. John Selfridge showed that $\|n\| \geq 3\log_3 n$ for all n. Define the *defect* of n, denoted $\delta(n)$, to be $\|n\| - 3\log_3 n$; in this chapter we present a method for classifying all n with $\delta(n) < r$ for a given r. From this, we derive several consequences. We prove that $\|2^m 3^k\| = 2m + 3k$ for $m \leq 21$ with m and k not both zero, and present a method that can, with more computation, potentially prove the same for larger m. Furthermore, defining $A_r(x)$ to be the number of n with $\delta(n) < r$ and $n \leq x$, we prove that $A_r(x) = \Theta_r((\log x)^{\lfloor r \rfloor + 1})$, allowing us to conclude that the values of $\|n\| - 3\log_3 n$ can be arbitrarily large.

2.1 Introduction

The complexity of a natural number n is the least number of 1's needed to write it using any combination of addition and multiplication, with the order of the operations specified using parentheses grouped in any legal nesting. For instance, 11 has complexity of 8, since it can be written using 8 ones as $(1+1+1)(1+1+1)+1+1$, but not with any fewer. This notion was introduced by Kurt Mahler and Jan Popken in 1953 [38]. It was later circulated by Richard Guy [29], who discusses it under problem F26 in his *Unsolved Problems in Number Theory* [30]. It has since been studied by a number of authors, e.g. Daniel Rawsthorne [39] and especially Juan Arias de Reyna [8].

Following Arias de Reyna [8] we will denote the complexity of n by $\|n\|$. Notice that for any natural numbers n and m we will have

$$\|1\| = 1, \quad \|n+m\| \leq \|n\| + \|m\|, \quad \|nm\| \leq \|n\| + \|m\|,$$

More specifically, for any $n > 1$, we have

$$\|n\| = \min_{\substack{a,b<n\in\mathbb{N} \\ a+b=n \text{ or } ab=n}} \|a\| + \|b\|.$$

This fact together with $\|1\| = 1$ allows one to compute $\|n\|$ recursively. If the equality $\|n\| = \|a\| + \|b\|$ holds, with either $n = a + b$ or $n = ab$, then we will say n can be written *most-efficiently* as $a + b$ or as ab, respectively.

Integer complexity is approximately logarithmic; it satisfies the bounds

$$3\log_3 n \leq \|n\| \leq 3\log_2 n, \qquad n > 1.$$

The upper bound can be obtained by writing n in binary and finding a representation using Horner's algorithm. The lower bound follows from results described below. The lower bound is known to be attained infinitely often, namely for all $n = 3^k$. The constant in the upper bound above can be improved further [52], and it is an open problem to determine the true asymptotic order of magnitude of the upper bound. At present even the possibility that an asymptotic formula $\|n\| \sim 3\log_3 n$ might hold has not been ruled out.

Let $E(k)$ be the largest number writable with k ones, i.e., with complexity at most k. John Selfridge (see [29]) proved that $E(1) = 1$ and that the larger values depend on the residue class of k modulo 3, namely for $k = 3j + i \geq 2$,

$$\begin{aligned} E(3j) &= 3^j \\ E(3j+1) &= 4 \cdot 3^{j-1} \\ E(3j+2) &= 2 \cdot 3^j \end{aligned}$$

Observe that $E(k) \leq 3^{k/3}$ in all cases, and that equality holds for cases where 3 divides k. These formulas also show that $E(k) > E(k-1)$, a fact that implies that the integer $E(k)$ requires exactly k ones. This yields the following result:

Theorem 2.1.1. *For $a = 0, 1, 2$ and for all $k \geq 0$ with $a + k \geq 1$, one has*

$$\|2^a \cdot 3^k\| = 2a + 3k.$$

Further results are known on the largest possible integers having a given complexity. We can generalize the notion of $E(k)$ with the following definition:

Definition 2.1.2. Define $E_r(k)$ to be the $(r+1)$-th largest number writable using k

ones, i.e. complexity at most k, so long as there are indeed $r+1$ or more distinct such numbers. Thus $E_r(k)$ is defined only for k sufficiently large depending on r. Here $E_0(k) = E(k)$.

Daniel A. Rawsthorne [39] determined a formula for $E_1(k)$, namely:

$$E_1(k) = \frac{8}{9}E(k), \qquad k \geq 8$$

Direct computation establishes that $E_1(k) \leq (8/9)E(k)$ holds for all k with $2 \leq k \leq 7$ (note that $E_1(1)$ is not defined). From this fact we deduce that, for $0 \leq a \leq 5$ and all $k \geq 0$ with $a + k > 0$,

$$\|2^a \cdot 3^k\| = 2a + 3k.$$

J. Iraids et al. [33] has verified that $\|2^a 3^k\| = 2a + 3k$ for $2 \leq 2^a \cdot 3^k \leq 10^{12}$, so in particular

$$\|2^a\| = 2a, \quad \text{for} \quad 1 \leq a \leq 39.$$

These results together with results given later in this thesis lend support to the following conjecture, which was originally formulated as a question in Guy [29].

Conjecture 2.1.3. *For all $a \geq 0$ and all $k \geq 0$ with $a + k \geq 1$ there holds*

$$||2^a \cdot 3^k|| = 2a + 3k.$$

This conjecture is presented as a convenient form for summarizing existing knowledge; there is limited evidence for its truth, and it may well be false. Indeed its truth would imply $\|2^a\| = 2a$, for all a. Selfridge raised this special case in a contrary form, asking the question whether there is some a for which $\|2^a\| < 2a$ (see [29]).

In this chapter, we will investigate these questions by looking at numbers n for which the difference $\delta(n) := \|n\| - 3\log_3 n$ is less than a given threshold; these sets we may call numbers with integer complexity close to the lower bound.

2.1.1 Main Results

The fundamental issue making the complexity of an integer a complicated quantity are: (1) It assumes the same value for many integers, because it is logarithmically small; (2) It is hard to determine lower bounds for a given value $\|n\|$, since the dynamic programming tree is exponentially large. The feature (1) implies there can be many tie values in going down the tree, requiring a very large search, to determine any specific complexity value.

We introduce a new invariant to study integer complexity.

Definition 2.1.4. The *defect* of a natural number n is given by

$$\delta(n) = \|n\| - 3\log_3 n$$

The introduction of the defect simplifies things in that it provides a more discriminating invariant: we show that $\delta(n) \geq 0$ and that it separates integers into quite small equivalence classes. In these equivalence classes powers of 3 play a special role. The following result establishes a conjecture of Arias de Reyna [8, Conjecture 1].

Theorem 2.1.5. *The following hold:*

1. *For a given value δ of the defect, the set $S(\delta) := \{m : \ \delta(m) = \delta\}$, is a chain $\{n \cdot 3^k : 0 \leq k \leq k(n)\}$ where $k(n)$ may be finite or infinite. The value n is called the leader of the chain.*

2. *The function $\delta(n \cdot 3^k)$ is non-increasing on the sequence $\{n \cdot 3^k : k \geq 0\}$. This sequence has a finite number of leaders culminating in a largest leader $n \cdot 3^L$, having the property that*

$$||n \cdot 3^k|| = ||n \cdot 3^L|| + 3(k - L), \text{ for all } k \geq L.$$

The set of integers $n \cdot 3^k$ for $k \geq L$ are termed *stable integers*, because their representation using 1's stabilizes into a predictable form for $k \geq L$. This result is proved in Section 2.2.1.

The main results of the chapter concern classifying integers having small values of the defect. The defect is compatible with the multiplication aspect of the dynamic programming definition of the integer complexity, but it does not fully respect the addition aspect. The main method underlying the results of this chapter is given in Theorem 2.4.4, which provides strong constraints on the dynamic programming recursion for classifying numbers of small defect. It allows construction of sets of integers including all integers of defect below a specified bound r, which may however include some additional integers. The method contains adjustable parameters, and with additional work they sometimes permit exact determination of these sets.

This main method has several applications. First, we use it to explictly classify all integers of defect below the bound $12\delta(2) \approx 1.286$. (Theorem 2.5.1). This requires pruning the sets found using Theorem 2.4.4 to determine the sets below $k\delta(2)$ for $1 \leq k \leq 12$.

Using this result we obtain an explicit classification of all integers having defect at most 1, as follows.

Theorem 2.1.6. *The numbers n satisfying $0 \leq \delta(n) < 1$ are precisely those that can be written in one of the following forms, and have the following complexities:*

1. 3^k *for* $k \geq 1$, *of complexity* $3k$
2. $2^a 3^k$ *for* $a \leq 9$, *of complexity* $2a + 3k$ *(for* a, k *not both zero)*
3. $5 \cdot 2^a 3^k$ *for* $a \leq 3$, *of complexity* $5 + 2a + 3k$
4. $7 \cdot 2^a 3^k$ *for* $a \leq 2$, *of complexity* $6 + 2a + 3k$
5. $19 \cdot 3^k$ *of complexity* $9 + 3k$
6. $13 \cdot 3^k$ *of complexity* $8 + 3k$
7. $(3^n + 1)3^k$ *of complexity* $1 + 3n + 3k$ *(for* $n \neq 0$*)*

Furthermore $n = 1$ *is the only number having defect exactly* 1.

This result is established in Section 2.6.2. Using a slightly more general result, which we present as Theorem 2.5.1, one can obtain a generalization of Rawsthorne's results, consisting of a description of all $E_r(k)$ for every finite $r \geq 0$, valid for all sufficiently large k, depending on r. This answer also depends on the congruence class of $k \pmod 3$. For example, one has

$$E_2(3k) = \frac{64}{81} E(3k),$$

$$E_2(3k+1) = \frac{5}{6} E(3k+1)$$

and

$$E_2(3k+2) = \frac{5}{6} E(3k+2),$$

all holding for $k \geq 4$. For $E_5(k)$ all three residue classes have different formulas, valid for $k \geq 5$. This generalization will be described elsewhere ([4]).

Secondly, the result can be used to obtain lower bounds on complexity of certain integers, by showing they are excluded from sets containing all integers of complexity at most r. This we use to prove Conjecture 2.1.3 for $a \leq 21$.

Theorem 2.1.7. *For all a and k with $0 \le a \le 21$, $k \ge 0$, and $a + k \ge 1$, there holds*

$$\|2^a 3^k\| = 2a + 3k.$$

This result is established in Section 2.6.3. It is possible to carry out computations establishing the Conjecture 2.1.3 for some larger values of a, as we shall describe in Chapter 5.

Thirdly, our main method can be used to estimate the magnitude of numbers below x having a given defect.

Theorem 2.1.8. *For any $r > 0$ the number $A_r(x)$ of numbers n smaller than x which have complexity $\delta(n) < r$ satisfies an upper bound, valid for all $x \ge 2$,*

$$A_r(x) \le C_r (\log x)^{\lfloor r \rfloor + 1},$$

where $C_r > 0$ is an effectively computable constant depending on r.

This result is proved in Section 2.6.4. It implies that the set of possible defect values is unbounded.

2.1.2 Discussion

We first remark on computing $\|n\|$. The recursive definition permits computing $\|n\|$ by dynamic programming, but it requires knowing $\{\|k\| : 1 \le k \le n-1\}$, so takes exponential time in the input size of n measured in bits. In particular, a straightforward approach to computing $\|x\|$ requires on the order of n^2 steps. Srinivas and Shankar [44] obtained an improvement on this, running in time $O(n^{\log_2 3})$.

We make some further remarks on Conjecture 2.1.3. Let's specialize to $k = 0$ and consider an analogous question for prime powers, concerning $\|p^m\|$ as m varies. It is clear that $\|p^m\| \le m \cdot \|p\|$, since we can concatenate by multiplication m copies of a good representation of p. For which primes p is it true that $\|p^m\| = m\|p\|$ holds for all $m \ge 1$? This is verified for $p = 3$ by $\|3^m\| = 3m$, and the truth of Conjecture 2.1.3 requires that it hold for $p = 2$, with $\|2^m\| = 2m$. However this question has a negative answer for powers of 5. Here while $\|5\| = 5$, one instead gets that $\|5^6\| = \|15625\| = 29 < 6 \cdot \|5\| = 30$, as

$$\begin{aligned} 15625 \;=\; & 1 + (1+1)(1+1)(1+1)(1+1+1)(1+1+1) \cdot \\ & (1 + (1+1)(1+1)(1+1)(1+1+1)(1+1+1)(1+1+1)) \end{aligned}$$

This encodes the identity $5^5 = 1 + 72 \cdot 217$, in which $72 = 2^3 \cdot 3^2$ and $217 = 1 + 2^3 \cdot 3^3$. This counterexample for powers of 5 leaves open the possibility that there might exist a (possibly far larger) counterexample for powers of 2, that has not yet been detected.

This discussion shows that Conjecture 2.1.3, if true, implies a kind of very strong arithmetic independence of powers of 2 and powers of 3. This would represent an important feature of the prime 2 in integer complexity. Conjecture 2.1.3 has implications about the number of nonzero digits in the expansion of 2^n in base 3 as a function of n; namely, if there existed a large power of 2 with a huge number of zero digits in its base 3 expansion, then this would give a (counter)-example achieving $\|2^k\| < 2k$. Problems similar to this very special subproblem already appear difficult (see Lagarias [37]). A result of C. L. Stewart [45] yields a lower bound on the number of nonzero digits appearing in the base 3 expansion of 2^n, but it is tiny, being only $\Omega(\frac{\log n}{\log \log n})$.

The truth of $\|2^n\| = 2n$ would also immediately imply the lower bound

$$\limsup_{n \to \infty} \frac{\|n\|}{\ln n} \geq \frac{2}{\ln 2}.$$

Computer experiments seem to agree with this prediction and even allow the possibility of equality, see Iraids et al [33].

There remain many interesting open questions concerning the classification of integers given by the defect. The first concerns the distribution of stable and unstable integers. How many are there of each kind? A second question concerns the function $M(n)$ that counts the number of distinct minimal decompositions into 1's that a given integer n has. How does this function behave?

Finally we remark that the set $\mathscr{D} := \{\delta(n) : n \geq 1\}$ of all defect values turns out to be a highly structured set. In Chapter 3, we shall show that it is a well-ordered set, of order type ω^ω, a fact related to some earlier conjectures of Juan Arias de Reyna [8].

2.2 Properties of the defect

The defect is the fundamental tool in this thesis; let us begin by noting some of its basic properties.

Proposition 2.2.1. *The following hold:*

1. *For all integers $a \geq 1$,*

$$\delta(a) \geq 0.$$

Here equality holds precisely for $a = 3^k$, $k \geq 1$.

2. *One has*
$$\delta(ab) \leq \delta(a) + \delta(b),$$
and equality holds if and only if $\|ab\| = \|a\| + \|b\|$.

3. *For $k \geq 1$,*
$$\delta(3^k \cdot n) \leq \delta(n)$$
and equality holds if and only if $\|3^k \cdot n\| = 3k + \|n\|$.

Proof. (1) This follows from the result of Selfridge. Since for $k \geq 1$, $\|3^k\| = 3k$, we have $\delta(3^k) = 0$ for $k \geq 1$, while $\delta(1) = 1$. For the converse, note that $3 \log_3 n$ is only an integer if n is a power of 3.

(2) This is a direct consequence of the definition.

(3) This follows from (2), from noting that $\delta(3^k) = 0$ for $k \geq 1$. □

Because $\|3^k\| = 3k$ for $k \geq 1$, one might hope that in general, $\|3n\| = 3 + \|n\|$ for $n > 1$. However, this is not so; for instance, $\|107\| = 16$, but $\|321\| = 18$.

The defect measures how far a given integer is from the upper bound $E(||n||)$, given in terms of the ratio $E(\|n\|)/n$:

Proposition 2.2.2. *We have $\delta(1) = 1$ and*

$$\delta(n) = \begin{cases} 3 \log_3 \frac{E(\|n\|)}{n} & \text{if} \quad \|n\| \equiv 0 \pmod 3, \\ 3 \log_3 \frac{E(\|n\|)}{n} + 2\,\delta(2) & \text{if} \quad \|n\| \equiv 1 \pmod 3, \text{ with } n > 1, \\ 3 \log_3 \frac{E(\|n\|)}{n} + \delta(2) & \text{if} \quad \|n\| \equiv 2 \pmod 3. \end{cases}$$

In particular $E(\|n\|)/n \geq 1$ for any $n \geq 1$.

Proof. The proof is a straightforward computation using Selfridge's formulas for $E(k)$, for $k = 3j + i$, $i = 0, 1, 2$. □

2.2.1 Stable Integers

The example of $\|107\|$ vs. $\|321\|$ motivates the following definition.

Definition 2.2.3. A number m is called *stable* if $\|3^k \cdot m\| = 3k + \|m\|$ holds for every $k \geq 1$. Otherwise it is called *unstable*.

We have the following criterion for stability.

Proposition 2.2.4. *The number m is stable if and only if $\delta(3^k \cdot m) = \delta(m)$ for all $k \geq 0$.*

Proof. This is immediate from Proposition 2.2.1(3). □

These results already suffice to prove the following result, conjectured by Juan Arias de Reyna [8].

Theorem 2.2.5. *The following hold:*

1. *For any $m \geq 1$, there exists a finite $K \geq 0$ such that $3^K m$ is stable.*

2. *If the defect $\delta(m)$ satisfies $0 \leq \delta(m) < 1$, then m itself is stable.*

Proof of Theorem 2.2.5. (1) From Proposition 2.2.1, we have that for any n, $\delta(3n) \leq \delta(n)$, with equality if and only if $\|3n\| = \|n\| + 3$. More generally, $\delta(3n) = \delta(n) - (\|n\| + 3 - \|3n\|)$, and so the difference $\delta(n) - \delta(3n)$ is always an integer. This means that the sequence $\delta(m), \delta(3m), \delta(9m), \ldots$ is non-increasing, nonnegative, and can only decrease in integral amounts; hence it must eventually stabilize. Applying Proposition 2.2.4 proves the theorem.

(2) If $\delta(m) < 1$, since all $\delta(n) \geq 0$ there is no room to remove any integral amount, so m must be stable. □

Note that while this proof shows that for any n there exists K such that $3^K n$ is stable, it yields no upper bound on such a K. We will give a more constructive proof and show how to compute such a K in Chapter 5.

The value of the defect separates the integers into small classes, whose members differ only by powers of 3.

Proposition 2.2.6. *Suppose that m and n are two positive integers, with $m > n$.*

(1) If $q := \delta(n) - \delta(m)$ is rational, then it is necessarily a nonnegative integer, and furthermore $m = n \cdot 3^k$ for some $k \geq 1$.

(2) If $\delta(n) = \delta(m)$ then $m = n \cdot 3^k$ for some $k \geq 1$ and furthermore

$$||n \cdot 3^j|| = 3j + ||n|| \qquad \text{for } 0 \leq j \leq k.$$

In particular $\delta(n) = \delta(m)$ implies $\|n\| \equiv \|m\| \pmod 3$.

Proof. (1) If $q = \delta(n) - \delta(m)$ is rational, then $k = \log_3(m/n) \in \mathbb{Q}$ is rational; since m/n is rational, the only way this can occur is if $\log_3(m/n)$ is an integer k, in which

case, since $m > n$, $m = n \cdot 3^k$ with $k \geq 1$. It then follows from the definition of defect that $q = \|n\| + 3k - \|m\|$.

(2) By (1) we know that $m = n \cdot 3^k$ for some $k \geq 1$. By Proposition 2.2.1 (3) we have $\delta(n \cdot 3^j) \leq \delta(n)$, for $j \geq 0$ and it also gives $\delta(m) = \delta(n \cdot 3^k) \leq \delta(n \cdot 3^j)$, for $0 \leq j \leq k$. Since $\delta(m) = \delta(n)$ by hypothesis, this gives $\delta(n \cdot 3^j) = \delta(n)$, so that $||n \cdot 3^j|| = 3j + ||n|| : 0 \leq j \leq k$. □

The results so far suffice to prove Theorem 2.1.5.

Proof of Theorem 2.1.5. (1) This follows from Proposition 2.2.6(2).

(2) The non-increasing assertion follows from Proposition 2.2.1(3). The finiteness of the number of leaders in a sequence $3^k \cdot n$ follows from Theorem 2.2.5 (1). □

2.2.2 Leaders

Again because $\|3n\|$ is not always equal to $3 + \|n\|$, it makes sense to introduce the following definition:

Definition 2.2.7. We call a natural number n a *leader* if it cannot be written most-efficiently as $3m$ for some m; i.e., if either $3 \nmid n$, or, if $3 \mid n$, then $\|n\| < 3 + \|n/3\|$.

For example, 107 is a leader since $3 \nmid 107$, and 321 is also a leader since

$$\|321\| = 18 < 3 + 16 = 3 + \|107\|.$$

However, 963 is not a leader, as

$$\|963\| = 21 = 3 + \|321\|.$$

Leaders can be stable or unstable. In this example 107 is unstable, but by Theorem 2.2.5 some multiple $3^K \cdot 107$ will be stable, and the smallest such multiple will be a stable leader.

We have the following alternate characterization of leaders:

Proposition 2.2.8. *The following hold:*

1. *A number n is a leader if and only if it is the smallest number having its given defect value.*

2. *For any natural number m, there is a unique leader $n \leq m$ such that $\delta(n) = \delta(m)$. For it $m = n \cdot 3^k$ for some $k \geq 0$.*

Proof. (1) If this were false, there would a leader n with some $n' < n$ with $\delta(n') = \delta(n)$. By Proposition 2.2.6 (2) $n = 3^k \cdot n'$ with $k \geq 1$ and $||n' \cdot 3^j|| = 3j + ||n'||$ for $0 \leq j \leq k$. But then $n/3 = n' \cdot 3^{k-1}$ is an integer and $||n/3|| = ||n'|| + 3k - 3 = ||n|| - 3$, which contradicts n being a leader.

Conversely, if n is the first number of its defect and is divisible by 3, then we cannot have $\|n\| = \|n/3\| + 3$, or else by Proposition 2.2.1 we would obtain $\delta(n) = \delta(n/3)$, contradicting minimality.

(2) Pick n to be the smallest number such that $\delta(n) = \delta(m)$; this is the unique leader satisfying $\delta(n) = \delta(m)$. Then $m = 3^k n$ for some $k \geq 0$ by Proposition 2.2.6. □

To summarize, if δ occurs as a defect, then the set of integers

$$N(\delta) := \{m : \delta(m) = \delta\},$$

having a given defect value δ has a smallest element that is a leader. If this leader n is unstable, then

$$N(\delta) = \{3^j \cdot n : 0 \leq j \leq j(\delta)\}.$$

If this leader is stable, then $N(\delta) = \{3^j \cdot n : j \geq 0\}$ is an infinite set. Furthermore if $3 \nmid n$ then n is a leader, and there is a unique $K = K(n) \geq 0$ such that $n' = 3^K n$ is a stable leader.

2.3 Good factorizations and solid numbers

Given a natural number $n > 1$, by the dynamic programming definition of complexity there are two numbers u and v, both smaller than n, such that either $n = u \cdot v$ and $\|n\| = \|u\| + \|v\|$, or such that $n = u + v$ and $\|n\| = \|u\| + \|v\|$. In the case u and v such that $n = u + v$, and $\|n\| = \|u\| + \|v\|$ we say n is *additively reducible*. In the case $n = u \cdot v$ and $\|n\| = \|u\| + \|v\|$ we say n is *multiplicatively reducible*. Some numbers n are reducible in both senses. For instance, $10 = 9 + 1$ with $\|10\| = \|9\| + \|1\|$, and $\|10\| = 2 \cdot 5$ with $\|10\| = \|2\| + \|5\|$.

2.3.1 Additive Irreducibility and Solid Numbers

We introduce terminology for numbers not being additively reducible.

Definition 2.3.1. We will say that a natural number n is *additively irreducible* if it cannot be written most-efficiently as a sum, i.e., for all u and v such that $n = u + v$, we have $\|n\| < \|u\| + \|v\|$. We call such values of n *solid numbers*.

The first few solid numbers are

$$\{1, 6, 8, 9, 12, 14, 15, 16, 18, 20, 21, 24, 26, 27, \ldots\}$$

It can be shown that 3^n is a solid number for $n \geq 2$, and so there are infinitely many solid numbers. Experimental evidence suggests that a positive fraction of integers below x are solid numbers, as $x \to \infty$.

2.3.2 Multiplicative Irreducibility and Good Factorizations

We introduce further terminology for factorizations that respect complexity.

Definition 2.3.2. A factorization $n = u_1 \cdot u_2 \cdots u_k$ is a *good factorization* of n if n can be written most-efficiently as $u_1 \cdot u_2 \cdots u_k$, i.e., if the following equality holds:

$$\|n\| = \|u_1\| + \|u_2\| + \ldots + \|u_k\|.$$

The factorization containing only one factor is automatically good; this will be called a *trivial good factorization*.

Proposition 2.3.3. *If $n = n_1 \cdot n_2 \cdot \ldots \cdot n_k$ is a good factorization then for any nonempty subset $I \subset \{1, 2, \ldots, k\}$ the product $m = \prod_{j \in I} n_j$ is a good factorization of m.*

Proof. If the factorization of m were not good, then we would have

$$\|m\| < \sum_{j \in I} \|n_j\|$$

But then

$$\|n\| = \left\| m \prod_{j \notin I} n_j \right\| < \sum_{j \in I} \|n_j\| + \sum_{j \notin I} \|n_j\| = \sum_{j=1}^{k} \|n_j\|$$

and the given factorization of n would not be a good factorization. □

Proposition 2.3.4. *The following hold:*

1. *If $n = n_1 \cdot n_2 \cdot \ldots \cdot n_k$ is a good factorization, and each $n_i = n_{i,1} \cdot \ldots \cdot n_{i,l_i}$ is a good factorizations, then so is $n = \prod_{i=1}^{k} \prod_{j=1}^{l_i} n_{i,j}$.*

2. *If $n = n_1 \cdot n_2 \cdot \ldots \cdot n_k$ is a good factorization, and $I_1, I_2, \ldots, I_l$ is a partition of $\{1, \ldots, k\}$, then letting $m_i = \prod_{j \in I_i} n_j$, we have that $n = \prod_{i=1}^{l} m_i$ is a good factorization.*

Proof. (1) We have that $\|n_i\| = \sum_{j=1}^{l_i} \|n_{i,j}\|$ and $\|n\| = \sum_{i=1}^{k} \|n_i\|$, so

$$\|n\| = \sum_{i=1}^{k} \sum_{j=1}^{l_i} \|n_{i,j}\|$$

and we are done.

(2) This follows from Proposition 2.3.3 together with (1). □

Definition 2.3.5. We will say that a natural number n is *multiplicatively irreducible* (abbreviated *m-irreducible*) if n has no nontrivial good factorizations.

Proposition 2.3.4(2) shows n is m-irreducible if and only if all nontrivial factorizations $n = uv$ have $\|n\| < \|u\| + \|v\|$. Thus a prime number p is automatically m-irreducible since the only factorization is $p = p \cdot 1$ and obviously we have $\|p\| < \|p\| + 1 = \|p\| + \|1\|$. However, the converse does not hold. For instance, 46 is a composite number which is m-irreducible.

Proposition 2.3.6. *Any natural number has a good factorization into m-irreducibles.*

Proof. We may apply induction and assume that any $m < n$ has a factorization into m-irreducibles. If n is m-irreducible, we are done. Otherwise, n has a good factorization $n = uv$. Observe that $n = n \cdot 1$ is never a good factorization, since $\|1\| = 1$; hence, $u, v < n$. Then the induction hypothesis implies that u and v have good factorizations into m-irreducibles. Multiplying these factorizations together and applying Proposition 2.3.4, we obtain a good factorization of n into m-irreducibles. □

Good factorizations into m-irreducibles need not be unique. For $4838 = 2 \cdot 41 \cdot 59$, we find that $2 \cdot (41 \cdot 59)$, $(2 \cdot 59) \cdot 41$ and $(2 \cdot 41) \cdot 59$ are all good factorizations, but the full factorization $2 \cdot 41 \cdot 59$ is not a good factorization. (Thanks to Juan Arias de Reyna for this example.) This is deducible from the following data:

$$\|2 \cdot 41 \cdot 59\| = 27,$$
$$\|2\| = 2, \quad \|41\| = 12, \quad \|59\| = 14.$$
$$\|2 \cdot 41\| = 13, \quad \|2 \cdot 59\| = 15, \quad \|41 \cdot 59\| = 25,$$

2.3.3 Good factorizations and leaders

The next two propositions show how the notion of good factorization interacts with leaders and stability.

Proposition 2.3.7. *Let $n = n_1 \cdot n_2 \cdots n_r$ be a good factorization. If n is a leader then each of the factors n_j is a leader.*

Proof. Suppose otherwise; without loss of generality, we may assume that n_1 is not a leader, so $3 \mid n_1$ and $\|n_1\| = 3 + \|n_1/3\|$. So $3 \mid n$ and

$$\|n/3\| = \|(n_1/3) \cdot n_2 \cdot \ldots \cdot n_r\| \leq \|n_1/3\| + \sum_{j=2}^{r} \|n_j\|$$
$$= \|n_1\| - 3 + \sum_{j=2}^{r} \|n_j\| = \|n\| - 3.$$

Since $\|n\| \leq 3 + \|n/3\|$, we have $\|n\| = 3 + \|n/3\|$, and thus n is not a leader. □

Proposition 2.3.8. *Let $n = n_1 \cdot n_2 \cdots n_r$ be a good factorization. If n is stable, then each of its factors n_j is stable.*

Proof. Suppose otherwise. Without loss of generality, we may assume that n_1 is unstable; say $\|3^k n_1\| < \|n_1\| + 3k$. So

$$\|3^k n\| = \|(3^k n_1) \cdot n_2 \cdot \ldots \cdot n_r\| \leq \|3^k n_1\| + \sum_{j=2}^{r} \|n_j\|$$
$$< \|n_1\| + 3k + \sum_{j=2}^{r} \|n_j\| = \|n\| + 3k.$$

and thus n is not stable. □

Assembling all these results we deduce that being a leader and being stable are both inherited properties for subfactorizations of good factorizations.

Proposition 2.3.9. *Let $n = n_1 \cdot n_2 \cdots n_r$ be a good factorization, and I be a nonempty subset of $\{1, \ldots, r\}$; let $m = \prod_{i \in I} n_i$. If n is a leader, then so is m. If n is stable, then so is m.*

Proof. This is immediate from Proposition 2.3.7, Proposition 2.3.8, and Proposition 2.3.4.(2). □

2.4 The Classification Method

Here, we state and prove a result (Theorem 2.4.4) that will be our primary tool for the rest of the chapter. By applying it repeatedly, for any $r > 0$, we can put restrictions on what integers n can satisfy $\delta(n) < r$.

Definition 2.4.1. (1) For any real $r \geq 0$, define A_r to be $\{n \in \mathbb{N} : \delta(n) < r\}$.

(2) Define B_r to be the set consisting of those elements of A_r that are leaders.

While A_r is our main object of interest, it turns out to be easier and more natural to deal with B_r. Note that knowing B_r is enough to determine A_r, as expressed in the following proposition:

Proposition 2.4.2.

$$A_r = \{3^k n : n \in B_r, k \geq 0\}$$

Proof. If $n \in B_r$, then $\delta(3^k n) \leq \delta(n) < r$, so $3^k n \in A_r$. Conversely, if $m \in A_r$, by Proposition 2.2.8(2) we can take $n \geq 1$ and $k \geq 0$ such that n is a leader, $m = 3^k n$, and $\delta(m) = \delta(n)$; then $n \in B_r$ and we are done. □

We now let $\alpha > 0$ be a real parameter, specifiable in advance. The main result puts constraints on the allowable forms of the dynamic programming recursion (most efficient representations) to compute integers in $B_{(k+1)\alpha}$ in terms of integers in $B_{j\alpha}$ for $1 \leq j \leq k$. However there are some exceptional cases that must be considered separately in the theorem; fortunately, for any $\alpha < 1$, there are only finitely many. We will collect these into a set we call T_α.

Definition 2.4.3. Define T_α to consist of 1 together with those m-irreducible numbers n which satisfy

$$\frac{1}{n-1} > 3^{\frac{1-\alpha}{3}} - 1$$

and do not satisfy $\|n\| = \|n - b\| + \|b\|$ for any solid numbers b with $1 < b \leq n/2$.

Observe that for $0 < \alpha < 1$, the above inequality is equivalent to

$$n < (3^{\frac{1-\alpha}{3}} - 1)^{-1} + 1$$

and hence T_α is a finite set. For $\alpha \geq 1$, the inequality is trivially satisfied and so $T_\alpha = T_1$. We do not know whether T_1 is a finite or an infinite set. However in our computations we will always choose values $0 < \alpha < 1$.

We can now state the main classification result, which puts strong constraints on the form of most-efficient decompositions of numbers in sets $B_{(k+1)\alpha}$.

Theorem 2.4.4. *Suppose $0 < \alpha < 1$ and that $k \geq 1$. Then any $n \in B_{(k+1)\alpha}$ can be most-efficiently represented in (at least) one of the following forms:*

1. *For $k = 1$, there is either a good factorization $n = u \cdot v$ where $u, v \in B_\alpha$, or a good factorization $n = u \cdot v \cdot w$ with $u, v, w \in B_\alpha$;*
   For $k \geq 2$, there is a good factorization $n = u \cdot v$ where $u \in B_{i\alpha}$, $v \in B_{j\alpha}$ with $i + j = k + 2$ and $2 \leq i, j \leq k$.

2. *$n = a + b$ with $b \leq a$ and $\|n\| = \|a\| + \|b\|$, where $a \in A_{k\alpha}$, b is a solid number and*
$$\delta(a) + \|b\| < (k+1)\alpha + 3\log_3 2.$$

3. *There is a good factorization $n = (a + b)v$ with $v \in B_\alpha$ and a and b satisfying the conditions in the case (2) above.*

4. *$n \in T_\alpha$, a finite set (and thus in particular either $n = 1$ or $\|n\| = \|n - 1\| + 1$.)*

5. *There is a good factorization $n = u \cdot v$ with $u \in T_\alpha$ and $v \in B_\alpha$.*

We will prove Theorem 2.4.4 in Section 2.4.2, after establishing a preliminary combinatorial lemma in Section 2.4.1.

To apply Theorem 2.4.4, one recursively constructs from given sets $B^*_{j\alpha}$, $A^*_{j\alpha}$ for $1 \leq j \leq k - 1$ which contain $B_{j\alpha}, A_{j\alpha}$, respectively, the set of all n satisfying the relaxed conditions (1)-(5) obtained replacing $B_{j\alpha}$ by $B^*_{j\alpha}$ and $A_{j\alpha}$ by $A^*_{j\alpha}$. This new set $B^{**}_{(k+1)\alpha}$ contains the set $B_{(k+1)\alpha}$ we want. Sometimes we can, by other methods, prune some elements from $B^{**}_{(k+1)\alpha}$ that do not belong to $B_{(k+1)\alpha}$, to obtain a new approximation $B^*_{(k+1)\alpha}$. This then determines

$$A^*_{(k+1)\alpha} := \{3^k n : \ k \geq 0, n \in B^*_{(k+1)\alpha}\},$$

permitting continuation to the next level $k + 2$. We will present two applications of this construction:

1. To get an upper bound on the cardinality of the set $A_{(k+1)\alpha}(x)$ of numbers below a given bound x with defect less than $(k + 1)\alpha$.

2. To get a lower bound for the complexity $\|n\|$ of a number n by showing it does not belong to a given set $A^*_{k\alpha}$; this excludes it from $A_{k\alpha}$, whence $\|n\| \geq 3\log_3 n + k\alpha$.

In some circumstances we can obtain the exact sets $B_{k\alpha}$ and $A_{k\alpha}$ for $1 \leq k \leq k_0$, i.e. we recursively construct $B^*_{k\alpha}$ so that $B^*_{k\alpha} = B_{k\alpha}$. This requires a perfect pruning operation at each step. Here a good choice of the parameter α is helpful.

In applications we will typically not use the full strength of Theorem 2.4.4. Though the representations it yields are most efficient, the proofs will typically not use this fact. Also, in the addition case (2), the requirement that

$$\delta(a) + \|b\| < (k+1)\alpha + 3\log_3 2$$

implies the weaker requirement that just

$$\|b\| < (k+1)\alpha + 3\log_3 2.$$

The latter relaxed condition is easier to check, but it does enlarge the initial set $B^{**}_{(k+1)\alpha}$ to be pruned.

2.4.1 A Combinatorial Lemma

We establish a combinatorial lemma regarding decomposing a sum of real numbers into blocks.

Lemma 2.4.5. *Let $x_1, x_2, \ldots, x_r > 0$ be real numbers such that $\sum_{i=1}^{r} x_i < k+1$, where $k \geq 1$ is a natural number.*

(1) If $k \geq 2$ then either there is some i with $x_i \geq k$, or else we may find a partition $A \cup B$ of the set $\{1, 2, \ldots, r\}$ such that

$$\sum_{i \in A} x_i < k, \qquad \sum_{i \in B} x_i < k.$$

(2) If $k = 1$ then either there is some i with $x_i \geq 1$, or else we may find a partition $A \cup B \cup C$ of the set $\{1, 2, \ldots, r\}$ such that

$$\sum_{i \in A} x_i < 1, \qquad \sum_{i \in B} x_i < 1, \qquad \sum_{i \in C} x_i < 1.$$

Proof. (1) Suppose $k \geq 2$. Let us abbreviate $\sum_{i \in S} x_i$ by $\sum S$. Among all partitions $A \cup B$ of $\{1, \ldots, r\}$, take one that minimizes $|\sum A - \sum B|$, with $\sum A \geq \sum B$. Suppose that $\sum A \geq k$; then since $\sum A + \sum B < k+1$, we have $\sum B < 1$, and so $\sum A - \sum B > k-1$. So pick $x_i \in A$ and let $A' = A \setminus \{i\}$, $B' = B \cup \{i\}$. If $\sum A' > \sum B'$, then

$$|\sum A' - \sum B'| = \sum A - \sum B - 2x_i < \sum A - \sum B,$$

contradicting minimality, so $\sum A' \leq \sum B'$. So $\sum B' - \sum A' \geq \sum A - \sum B$, i.e.,

$$x_i \geq \sum A - \sum B > k - 1.$$

Now i was an arbitrary element of A; this means that A can have at most one element, since otherwise, if $j \neq i \in A$, we would have $\sum A \geq x_i + x_j$ and hence

$$x_j \leq \sum A - x_i \leq \sum B < 1,$$

but also $x_j > k - 1$, contradicting $k \geq 2$. Thus $A = \{i\}$ and so $x_i \geq k$.

(2) Here $k = 1$. Assume that $x_1 \geq x_2 \geq \cdots \geq x_r$. If $x_1 \geq 1$ we are done. Otherwise, if $r \leq 3$, we can partition $\{1, \ldots, r\}$ into singletons.

For $r \geq 4$, assume by induction the lemma is true for all sets of numbers with strictly less than r elements. Let $y = x_{r-1} + x_r$. We must have $y < 1$ because otherwise $x_{r-3} + x_{r-2} \geq x_{r-1} + x_r \geq 1$ and we get $\sum_{i=1}^{r} x_i \geq 2$ in contradiction to the hypothesis. Hence, if we define $x'_1 = x_1$, $\ldots$, $x'_{r-2} = x_{r-2}$, $x'_{r-1} = y$, we have

$$\sum_{i=1}^{r-1} x'_i = \sum_{i=1}^{r} x_i < 2,$$

and $x'_i < 1$ for all i. By the inductive hypothesis, then, there exists a paritition

$$A' \cup B' \cup C' = \{1, \ldots, r-1\}$$

with

$$\sum_{i \in A'} x'_i < 1, \quad \sum_{i \in B'} x'_i < 1, \quad \sum_{i \in C'} x'_i < 1.$$

Replacing x'_{r-1} with x_{r-1} and x_r, we get the required partition of $\{1, \ldots, r\}$. □

For $k = 1$ the example taking $\{x_1, x_2, x_3\} = \{3/5, 3/5, 3/5\}$ shows that a partition into three sets is sometimes necessary.

2.4.2 Proof of the Classification Method

Proof of Theorem 2.4.4. Suppose $n \in B_{(k+1)\alpha}$; take a most-efficient representation of n, which is either ab, $a + b$, or 1. If $n = 1$, then $n \in T_\alpha$ and we are in case (4). So suppose $n > 1$.

If n is m-irreducible, we will pick a way of writing $n = a + b$ with $\|n\| = \|a\| + \|b\|$, $a \geq b$, and b is solid. There is necessarily a way to do this, since one way to do so is

to write $n = a+b$ with $\|n\| = \|a\| + \|b\|$ and b minimal. Since this is possible, then, if there is a way to choose a and b to have $b > 1$, do so; otherwise, we must pick $b = 1$. In either case,

$$\|a\| + \|b\| = \|n\| < 3\log_3(a+b) + (k+1)\alpha \leq 3\log_3(2a) + (k+1)\alpha,$$

so $\delta(a) + \|b\| < (k+1)\alpha + 3\log_3 2$.

If $a \in A_{k\alpha}$, we are in case (2). Otherwise, we have

$$3\log_3 a + k\alpha + \|b\| \leq \|a\| + \|b\| = \|n\| <$$
$$3\log_3(a+b) + (k+1)\alpha \leq 3\log_3(2a) + (k+1)\alpha,$$

so $\|b\| < 3\log_3 2 + \alpha$; since $\alpha < 1$, we have $\|b\| \leq 2$ and thus $b \leq 2$. Because b is solid, we have $b = 1$. By assumption, we only picked $b = 1$ if this choice was forced upon us, so in this case, we must have that n does not satisfy

$$\|n\| = \|n - b\| + \|b\|$$

for any solid b with $1 < b \leq n/2$.

Since $b = \|b\| = 1$ we have $3\log_3 a + k\alpha + 1 < 3\log_3(a+1) + (k+1)\alpha$; since $\alpha < 1$, solving for a, we find that

$$\frac{1}{n-1} = \frac{1}{a} > 3^{\frac{1-\alpha}{3}} - 1.$$

Thus, $n \in T_\alpha$ and we are in case (4).

Now we consider the case when n is not m-irreducible. Choose a good factorization of n into m-irreducible numbers, $n = \prod_{i=1}^{r} m_i$; since n is not m-irreducible, we have $r \geq 2$. Then we have $\sum_{i=1}^{r} \delta(m_i) = \delta(n) < (k+1)\alpha$. Note that since we assumed n is a leader, every product of a nonempty subset of the m_i is also a leader by Proposition 2.3.9. We now have two cases.

Case 1. $k \geq 2$.

Now by Lemma 2.4.5(1), either there exists an i with $\delta(m_i) \geq k\alpha$, or else we can partition the $\delta(m_i)$ into two sets each with sum less than $k\alpha$.

In the latter case, we may also assume these sets are nonempty, as if one is empty, this implies that $\delta(n) < k\alpha$, and hence any partition of the $\delta(m_i)$ will work; since $r \geq 2$, we can take both these sets to be nonempty. In this case, call the products of these two sets u and v, so that $n = uv$ is a good factorization of n. Then $\delta(u) + \delta(v) < (k+1)\alpha$, so if we let $(i-1)\alpha$ be the largest integral multiple of α which

is at most $\delta(u)$, then letting $j = k + 2 - i$, we have $\delta(v) < j\alpha$. So $i + j = k + 2$; furthermore, since $i\alpha$ is the smallest integral multiple of α which is greater than $\delta(u)$, and $\delta(u) < k\alpha$, we have $i \leq k$, so $j \geq 2$. If also $i \geq 2$ then $j \leq k$, and so we are in case (1). If instead $i = 1$, then we have $u \in B_\alpha \subseteq B_{2\alpha}$, and $v \in B_{k\alpha}$ (since $\delta(v) < k\alpha$), so we are again in case (1) if we take $i = 2$ and $j = k$.

If such a partition is not possible, then let u be an m_i with $\delta(m_i) \geq k\alpha$, and let v be the product of the other m_i, so that once again $n = uv$ is a good factorization of n. Since $\delta(u) + \delta(v) = \delta(n)$, we have $\delta(v) < \alpha$, and so $v \in B_\alpha$. Finally, since u is m-irreducible and an element of $B_{(k+1)\alpha}$, it satisfies the conditions of either case (2) or case (4), and so n satisfies the conditions of either case (3) or case (5).

Case 2. $k = 1$.

Now by Lemma 2.4.5(2), either there exists an i with $\delta(m_i) \geq \alpha$, or else we can partition the $\delta(m_i)$ into three sets each with sum less than α.

In the latter case, we may also assume at least two of these sets are nonempty, as otherwise $\delta(n) < \alpha$, and hence any partition of the $\delta(m_i)$ will work. If there are two nonempty sets, call the products of these two sets u and v, so that $n = uv$ is a good factorization of n. If there are three nonempty sets, call their products u, v, w, so that $n = uvw$ is a good factorization of n. Thus we are in case (1) for $k = 1$.

If such a partition is not possible, then we repeat the argument in Case 1 above, determining that n satisfies one of the conditions of cases (3) or (5). □

2.5 Determination of all elements of defect below a given bound r

In this section we determine all elements of A_r for certain small r, using Theorem 2.4.4 together with a pruning operation.

2.5.1 Classification of numbers of small defect

We will now choose as our parameter

$$\alpha := \delta(2) = 2 - 3\log_3 2 \approx 0.107.$$

The choice of this parameter is motivated by Theorem 2.5.2 below. We use above method to inductively compute $A_{k\delta(2)}$ and $B_{k\delta(2)}$ for $0 \leq k \leq 12$. Numerically, $1.286 < 12\delta(2) < 1.287$. The following result classifies all integers in $A_{12\delta(2)}$.

Theorem 2.5.1. (Classification Theorem) *The numbers n satisfying $\delta(n) < 12\delta(2)$ are precisely those that can be written in at least one of the following forms, which have the indicated complexities:*

1. 3^k *of complexity* $3k$ *(for* $k \geq 1$*)*
2. $2^a 3^k$ *for* $a \leq 11$*, of complexity* $2a + 3k$ *(for* a*,* k *not both zero)*
3. $5 \cdot 2^a 3^k$ *for* $a \leq 6$*, of complexity* $5 + 2a + 3k$
4. $7 \cdot 2^a 3^k$ *for* $a \leq 5$*, of complexity* $6 + 2a + 3k$
5. $19 \cdot 2^a 3^k$ *for* $a \leq 3$*, of complexity* $9 + 2a + 3k$
6. $13 \cdot 2^a 3^k$ *for* $a \leq 2$*, of complexity* $8 + 2a + 3k$
7. $2^a(2^b 3^l + 1)3^k$ *for* $a + b \leq 2$*, of complexity* $2(a + b) + 3(l + k) + 1$ *(for* b*,* l *not both zero).*
8. 1*, of complexity* 1
9. $55 \cdot 2^a 3^k$ *for* $a \leq 2$*, of complexity* $12 + 2a + 3k$
10. $37 \cdot 2^a 3^k$ *for* $a \leq 1$*, of complexity* $11 + 2a + 3k$
11. $25 \cdot 3^k$ *of complexity* $10 + 3k$
12. $17 \cdot 3^k$ *of complexity* $9 + 3k$
13. $73 \cdot 3^k$ *of complexity* $13 + 3k$

In particular, all numbers $n > 1$ with $\delta(n) < 12\delta(2)$ are stable.

This list is redundant; for example list (7) with $a = 0, b = 1, l = 1$ gives $7 \cdot 3^k$, which overlaps list (4) with $a = 0$. But the given form is convenient for later purposes. In the next section we will give several applications of this result. They can be derived knowing only the statement of this theorem, without its proof, though one will also require Theorem 2.4.4.

The detailed proof of this theorem is given in the rest of this section. The proof recursively determines all the sets $A_{k\delta(2)}$ and $B_{k\delta(2)}$ for $1 \leq k \leq 12$. It is possible to extend this method to values $k\delta(2)$ with $k > 12$ but it is tedious. In Chapter 5, we will present a method for automating these computations.

2.5.2 Base case

The use of $\delta(2)$ may initially seem like an odd choice of step size. Its significance is shown by the following base case, which is proved using Rawsthorne's result that $E_1(k) \leq (8/9)E(k)$ (with equality for $k \geq 8$).

Theorem 2.5.2. *If $\delta(n) \neq 0$, then $\delta(n) \geq \delta(2)$. Equivalently, if n is not a power of 3, then $\delta(n) \geq \delta(2)$.*

Proof. We apply Proposition 2.2.2. There are four cases.

Case 1. If $n = 1$, then $\delta(n) = 1 \geq \delta(2)$.

Case 2. If $\|n\| \equiv 2 \pmod 3$, then

$$\delta(n) = \delta(2) + 3\log_3 \frac{E(\|n\|)}{n} \geq \delta(2).$$

Case 3. If $\|n\| \equiv 1 \pmod 3$ and $n > 1$, then

$$\delta(n) = 2\delta(2) + 3\log_3 \frac{E(\|n\|)}{n} \geq 2\delta(2) \geq \delta(2).$$

Case 4. If $\|n\| \equiv 0 \pmod 3$, then

$$\delta(n) = 3\log_3(E(\|n\|)/n).$$

We know that in this case $n = E(\|n\|)$ if and only if n is a power of 3 if and only if $\delta(n) = 0$. So if $\delta(n) \neq 0$, then $n \leq E_1(\|n\|)$. But $E_1(\|n\|) \leq (8/9)E(\|n\|)$, so $E(\|n\|)/n \geq 9/8$, so $\delta(n) \geq 3\log_3 \frac{9}{8} = 3\delta(2) \geq \delta(2)$. □

The proof above also establishes:

Proposition 2.5.3. *We have $B_0 = \emptyset$, and $B_{\delta(2)} = \{3\}$.*

To prove Theorem 2.5.1 we will use Theorem 2.4.4 for the "inductive step". However, while Theorem 2.4.4 allows us to place restrictions on what A_r can contain, if we want to determine A_r itself, we need a way to certify membership in it. To certify inclusion in A_r we need an upper bound on the defect, which translates to an upper bound on complexity, which is relatively easy to do. However we also need to discard n that do not belong to A_r, i.e. pruning the set we are starting with. This requires establishing lower bounds on their defects, certifying they are r or larger, and for this we need lower bounds on their complexities.

2.5.3 Two pruning lemmas

To find lower bounds on complexities, we typically use the following technique. Say we want to show that $\|n\| \geq l$ ($l \in \mathbb{N}$); since $\|n\|$ is always an integer, it suffices to show $\|n\| > l - 1$. We do this by using our current knowledge of A_r for various r; by showing that if $\|n\| \leq l - 1$ held, then it would put n in some A_r which we have already determined and know it's not in. The following two lemmas, both examples of this principle, are useful for this purpose.

Lemma 2.5.4. *If $\alpha \leq 1/2$, $i + j = k + 2$, and a and b are natural numbers then*

$$a \in A_{i\alpha}, \quad b \in A_{j\alpha}, \quad ab \notin A_{k\alpha} \quad \Longrightarrow \quad \|ab\| = \|a\| + \|b\|.$$

Proof. Note

$$\|ab\| \geq 3\log_3(ab) + k\alpha = 3\log_3 a + 3\log_3 b + (i + j - 2)\alpha > \|a\| + \|b\| - 1$$

so $\|ab\| \geq \|a\| + \|b\|$. □

Lemma 2.5.5. *For natural numbers a, k, and $m \geq 0$ we have*

$$a \in A_{k\alpha}, \quad 3^m(a + 1) \notin A_{k\alpha} \quad \Longrightarrow \quad \|3^m(a + 1)\| = \|a\| + 3m + 1.$$

Proof. Note

$$\|3^m(a + 1)\| \geq 3\log_3(a + 1) + 3m + k\alpha > \|a\| + 3m$$

so $\|3^m(a + 1)\| \geq 3m + \|a\| + 1$. □

In applying the lemmas to verify that a given n does not lie in a given A_r, one must check that n is not in some other A_s. In our applications, we will have $s < r$, and A_s will already be known, allowing the required check. In the following subsection we will typically not indicate these checks explicitly, using the fact that in our cases one can always check whether $n \in A_s$ by looking at the base-3 expansion of n.

2.5.4 Proof of Theorem 2.5.1: Inductive Steps

We prove Theorem 2.5.1 by repeatedly applying Theorem 2.4.4, to go from k to $k+1$ for $0 \leq k \leq 12$. We will use a step size $\alpha = \delta(2)$, so let us first determine $T_{\delta(2)}$. We compute that

$$3 < (3^{\frac{1-\delta(2)}{3}} - 1)^{-1} + 1 < 4,$$

and so $T_{\delta(2)} = \{1, 2, 3\}$. We note that in all cases of attempting to determine $B_{(k+1)\alpha}$ we are considering, we will have $(k+1)\alpha \leq 12\delta(2)$, and so if

$$\|b\| < (k+1)\alpha + 3\log_3 2,$$

then

$$\|b\| < 12\delta(2) + 3\log_3 2 = 3.179\ldots,$$

so $\|b\| \leq 3$, which for b solid implies $b = 1$.

The base cases $B_0 = \emptyset$ and $B_{\delta(2)} = \{3\}$ were handled in Proposition 2.5.3. We now treat the $B_{k\delta(2)}$ in increasing order.

Proposition 2.5.6.

$$B_{2\delta(2)} = B_{\delta(2)} \cup \{2\},$$

and the elements of $A_{2\delta(2)}$ have the complexities listed in Theorem 2.5.1.

Proof. By the main theorem,

$$\begin{aligned} B_{2\delta(2)} \setminus B_{\delta(2)} \quad \subseteq \quad & \{1, 2, 6, 9, 27\} \cup \\ & \{3 \cdot 3^n + 1 : n \geq 0\} \cup \{3(3 \cdot 3^n + 1) : n \geq 0\}. \end{aligned}$$

We can exclude 1 because $\delta(1) = 1$, and we can exclude 6, 9, and 27 as they are not leaders. For $3^{n+1} + 1$, Lemma 2.5.5 shows $\|3^{n+1} + 1\| = 3(n+1) + 1$, and thus $\delta(3^{n+1} + 1) = 1 - 3\log_3(1 + 3^{-(n+1)})$, which allows us to check that none of these lie in $A_{2\delta(2)}$. We can exclude $3(3^{n+1} + 1)$ since Lemma 2.5.5 shows it has the same defect as $3^{n+1} + 1$ (and so therefore also is not a leader). Finally, checking the complexity of $2 \cdot 3^k$ can be done with Lemma 2.5.4. □

To make later computations easier, let us observe here that

$$\delta(3^1 + 1) = \delta(4) = 2\delta(2),$$

$$6\delta(2) < \delta(3^2 + 1) = \delta(10) < 7\delta(2),$$

$$8\delta(2) < \delta(3^3 + 1) = \delta(28) < 9\delta(2),$$

and that for $n \geq 4$,

$$9\delta(2) < \delta(3^n + 1) < 10\delta(2).$$

In the above, for illustration, we explicitly considered and excluded 3, 6, 9, 27, and $3(3^{n+1} + 1)$, but henceforth we will simply not mention any multiplications by 3.

If $n = 3a$ is a good factorization, n cannot be a leader (by definition), but if it is not a good factorization, we can by Theorem 2.4.4 ignore it.

Proposition 2.5.7.

$$B_{3\delta(2)} = B_{2\delta(2)} \cup \{4\},$$

and the elements of $A_{3\delta(2)}$ have the complexities listed in Theorem 2.5.1.

Proof. By the main theorem,

$$\begin{aligned} B_{3\delta(2)} \setminus B_{2\delta(2)} \quad \subseteq \quad & \{1, 4\} \cup \\ & \{3 \cdot 3^n + 1 : n \geq 0\} \cup \{2 \cdot 3^n + 1 : n \geq 0\}. \end{aligned}$$

Again, $\delta(1) = 1$. By the above computation, the only number of the form $3^{n+1} + 1$ occuring in $A_{3\delta(2)}$ is 4. Lemma 2.5.5 shows that $\|2 \cdot 3^n + 1\| = 3 + 3n$ for $n > 0$, and hence $\delta(2 \cdot 3^n + 1) = 3 - 3\log_3(2 + 3^{-n})$, which allows us to check that none of these lie in $A_{3\delta(2)}$. Finally, checking the complexity of $4 \cdot 3^k$ can be done with Lemma 2.5.4. $\square$

To make later computations easier, let us observe here that

$$6\delta(2) < \delta(2 \cdot 3^1 + 1) = \delta(7) < 7\delta(2),$$

$$8\delta(2) < \delta(2 \cdot 3^2 + 1) = \delta(19) < 9\delta(2),$$

$$9\delta(2) < \delta(2 \cdot 3^3 + 1) = \delta(55) < 10\delta(2),$$

and that for $n \geq 4$,

$$10\delta(2) < \delta(2 \cdot 3^n + 1) < 11\delta(2).$$

We will henceforth stop explicitly considering and then excluding 1, since we know that $9\delta(2) < \delta(1) = 1 < 10\delta(2)$.

Proposition 2.5.8.

$$B_{4\delta(2)} = B_{3\delta(2)} \cup \{8\},$$

and the elements of $A_{4\delta(2)}$ have the complexities listed in Theorem 2.5.1.

Proof. By the main theorem,

$$\begin{aligned} B_{4\delta(2)} \setminus B_{3\delta(2)} \quad \subseteq \quad & \{8\} \cup \{3 \cdot 3^n + 1 : n \geq 0\} \cup \\ & \{2 \cdot 3^n + 1 : n \geq 0\} \cup \{4 \cdot 3^n + 1 : n \geq 0\}. \end{aligned}$$

By the above computation, no numbers of the form $3^{n+1}+1$ or $2\cdot 3^n+1$ occur in $A_{4\delta(2)} \setminus A_{3\delta(2)}$. Lemma 2.5.5 shows $\|4\cdot 3^n+1\| = 5+3n$ and hence

$$\delta(4\cdot 3^n+1) = 5 - 3\log_3(4+3^{-n}),$$

which allows us to check that none of these lie in $A_{4\delta(2)}$. Finally, checking the complexity of $8\cdot 3^k$ can be done with Lemma 2.5.4. □

To make later computations easier, let us observe here that

$$5\delta(2) < \delta(4\cdot 3^0+1) = \delta(5) < 6\delta(2),$$

$$9\delta(2) < \delta(4\cdot 3^1+1) = \delta(13) < 10\delta(2),$$

$$10\delta(2) < \delta(4\cdot 3^2+1) = \delta(37) < 11\delta(2),$$

and that for $n \geq 3$,

$$11\delta(2) < \delta(4\cdot 3^n+1) < 12\delta(2).$$

Proposition 2.5.9.

$$B_{5\delta(2)} = B_{4\delta(2)} \cup \{16\},$$

and the elements of $A_{5\delta(2)}$ have the complexities listed in Theorem 2.5.1.

Proof. By the main theorem,

$$\begin{aligned} B_{5\delta(2)} \setminus B_{4\delta(2)} \quad \subseteq \quad & \{16\} \cup \{3\cdot 3^n+1 : n \geq 0\} \cup \{2\cdot 3^n+1 : n \geq 0\} \cup \\ & \{4\cdot 3^n+1 : n \geq 0\} \cup \{8\cdot 3^n+1 : n \geq 0\}. \end{aligned}$$

By the above computation, no numbers of the form $3^{n+1}+1$, $2\cdot 3^n+1$, or $4\cdot 3^n+1$ occur in $A_{5\delta(2)} \setminus A_{4\delta(2)}$. Lemma 2.5.5 shows that $\|8\cdot 3^n+1\| = 7+3n$ for $n>0$, and hence $\delta(8\cdot 3^n+1) = 7-3\log_3(8+3^{-n})$, which allows us to check that none of these lie in $A_{5\delta(2)}$. Finally, checking the complexity of $16\cdot 3^k$ can be done with Lemma 2.5.4. □

To make later computations easier, let us observe here that

$$11\delta(2) < \delta(8\cdot 3^1+1) = \delta(25) < \delta(8\cdot 3^2+1) = \delta(73) < 12\delta(2),$$

and that for $n \geq 3$,

$$\delta(8\cdot 3^n+1) > 12\delta(2).$$

Proposition 2.5.10.

$$B_{6\delta(2)} = B_{5\delta(2)} \cup \{32, 5\},$$

and the elements of $A_{6\delta(2)}$ have the complexities listed in Theorem 2.5.1.

Proof. By the main theorem,

$$\begin{aligned} B_{6\delta(2)} \setminus B_{5\delta(2)} \quad \subseteq \quad & \{32\} \cup \{3 \cdot 3^n + 1 : n \geq 0\} \cup \{2 \cdot 3^n + 1 : n \geq 0\} \cup \\ & \{4 \cdot 3^n + 1 : n \geq 0\} \cup \{8 \cdot 3^n + 1 : n \geq 0\} \cup \\ & \{16 \cdot 3^n + 1 : n \geq 0\}. \end{aligned}$$

By the above computations, the number of any of the forms $3^{n+1} + 1$, $2 \cdot 3^n + 1$, $4 \cdot 3^n + 1$, or $8 \cdot 3^n + 1$ occurring in $A_{5\delta(2)} \setminus A_{4\delta(2)}$ is $5 = 4 \cdot 3^0 + 1$. Lemma 2.5.5 shows that $\|16 \cdot 3^n + 1\| = 9 + 3n$, and hence $\delta(16 \cdot 3^n + 1) = 9 - 3\log_3(16 + 3^{-n})$, which allows us to check that none of these lie in $A_{6\delta(2)}$. Finally, checking the complexity of $32 \cdot 3^k$ can be done with Lemma 2.5.4, and checking the complexity of $5 \cdot 3^k$ can be done with Lemma 2.5.5. □

To make later computations easier, let us observe here that

$$11\delta(2) < \delta(16 \cdot 3^0 + 1) = \delta(17) < 12\delta(2),$$

and that for $n \geq 1$,

$$\delta(16 \cdot 3^n + 1) > 12\delta(2).$$

In the above, for illustration, we explicitly considered and excluded numbers of the form $3 \cdot 3^n + 1$, $2 \cdot 3^n + 1$, etc., for large n, despite having already computed their complexities earlier. Henceforth, to save space, we will simply not consider a number if we have already computed its defect and seen it to be too high. E.g., in the above proof, we would have simply said, "By the main theorem and the above computations, $B_{6\delta(2)} \setminus B_{5\delta(2)} \subseteq \{32, 5\} \cup \{8 \cdot 3^n + 1 : n \geq 0\}$".

Proposition 2.5.11.

$$B_{7\delta(2)} = B_{6\delta(2)} \cup \{64, 7, 10\},$$

and the elements of $A_{7\delta(2)}$ have the complexities listed in Theorem 2.5.1.

Proof. By the main theorem and the above computations,

$$B_{7\delta(2)} \setminus B_{6\delta(2)} \subseteq \{64, 7, 10\} \cup \{32 \cdot 3^n + 1 : n \geq 0\} \cup \{5 \cdot 3^n + 1 : n \geq 0\}.$$

Lemma 2.5.5 shows that $\|32 \cdot 3^n + 1\| = 11 + 3n$ and, for $n \geq 2$, $\|5 \cdot 3^n + 1\| = 6 + 3n$. Hence $\delta(32 \cdot 3^n + 1) = 11 - 3\log_3(32 + 3^{-n})$, and, for $n \geq 2$, $\delta(5 \cdot 3^n + 1) = 6 - 3\log_3(5 + 3^{-n})$ which allows us to check that none of these lie in $A_{7\delta(2)}$. Finally, checking the complexities of $64 \cdot 3^k$, $7 \cdot 3^k$, and $10 \cdot 3^k$ can be done via Lemma 2.5.4 (for 64 and 10) and Lemma 2.5.5 (for 7 and 10). □

To make later computations easier, let us observe here that

$$\delta(32 \cdot 3^n + 1) > 12\delta(2) \quad \text{for all } n,$$

and that for $n \geq 2$,

$$\delta(5 \cdot 3^n + 1) > 12\delta(2)$$

as well. Indeed, as we will see, from this point on, no new examples of multiplying by a power of 3 and then adding 1 will ever have complexity less than $12\delta(2)$.

Proposition 2.5.12.

$$B_{8\delta(2)} = B_{7\delta(2)} \cup \{128, 14, 20\},$$

and the elements of $A_{8\delta(2)}$ have the complexities listed in Theorem 2.5.1.

Proof. By the main theorem and the above computations,

$$\begin{aligned} B_{8\delta(2)} \setminus B_{7\delta(2)} \quad \subseteq \quad & \{128, 14, 20\} \cup \{64 \cdot 3^n + 1 : n \geq 0\} \cup \\ & \{7 \cdot 3^n + 1 : n \geq 0\} \cup \{10 \cdot 3^n + 1 : n \geq 0\}. \end{aligned}$$

Lemma 2.5.5 shows that $\|64 \cdot 3^n + 1\| = 13 + 3n$, $\|10 \cdot 3^n + 1\| = 8 + 3n$, and, for $n \neq 0, 2$, $\|7 \cdot 3^n + 1\| = 7 + 3n$. Using this to check their defects, we see that none of these lie in $A_{8\delta(2)}$, or even $A_{12\delta(2)}$. Finally, checking the complexities of $128 \cdot 3^k$, $14 \cdot 3^k$, and $20 \cdot 3^k$ can be done with Lemma 2.5.4. □

Proposition 2.5.13.

$$B_{9\delta(2)} = B_{8\delta(2)} \cup \{256, 28, 40, 19\},$$

and the elements of $A_{9\delta(2)}$ have the complexities listed in Theorem 2.5.1.

Proof. By the main theorem and the above computations,

$$\begin{aligned} B_{9\delta(2)} \setminus B_{8\delta(2)} \quad \subseteq \quad & \{256, 28, 40, 19\} \cup \{128 \cdot 3^n + 1 : n \geq 0\} \cup \\ & \{14 \cdot 3^n + 1 : n \geq 0\} \cup \{20 \cdot 3^n + 1 : n \geq 0\}. \end{aligned}$$

Lemma 2.5.5 shows that $\|128 \cdot 3^n + 1\| = 15 + 3n$, and for $n \geq 1$, $\|14 \cdot 3^n + 1\| = 9 + 3n$ and $\|20 \cdot 3^n + 1\| = 10 + 3n$. Using this to check their defects, we see that none of these lie in $A_{8\delta(2)}$, or even $A_{12\delta(2)}$. Finally, checking the complexities of $256 \cdot 3^k$, $28 \cdot 3^k$, and $40 \cdot 3^k$, and $19 \cdot 3^k$ can be done via Lemma 2.5.4 (for 256, 28, and 40) and Lemma 2.5.5 (for 28 and 19). □

Proposition 2.5.14.

$$B_{10\delta(2)} = B_{9\delta(2)} \cup \{512, 13, 1, 56, 80, 55, 38\} \cup \{3 \cdot 3^n + 1 : n \geq 3\},$$

and the elements of $A_{10\delta(2)}$ have the complexities listed in Theorem 2.5.1.

Proof. By the main theorem and the above computations,

$$\begin{aligned} B_{10\delta(2)} \setminus B_{9\delta(2)} \quad \subseteq \quad & \{512, 13, 1, 56, 80, 55, 38\} \cup \{3 \cdot 3^n + 1 : n \geq 3\} \cup \\ & \{256 \cdot 3^n + 1 : n \geq 0\} \cup \{28 \cdot 3^n + 1 : n \geq 0\} \cup \\ & \{40 \cdot 3^n + 1 : n \geq 0\} \cup \{19 \cdot 3^n + 1 : n \geq 0\}. \end{aligned}$$

We know $\delta(1) = 1$. Lemma 2.5.5 shows that $\|256 \cdot 3^n + 1\| = 17 + 3n$, $\|28 \cdot 3^n + 1\| = 11 + 3n$, $\|40 \cdot 3^n + 1\| = 12 + 3n$, and for $n \geq 1$, $\|19 \cdot 3^n + 1\| = 10 + 3n$. Using this to check their defects, we see that none of these lie in $A_{10\delta(2)}$, or even $A_{12\delta(2)}$. Finally, checking the complexities of $512 \cdot 3^k$, $13 \cdot 3^k$, $56 \cdot 3^k$, $80 \cdot 3^k$, $55 \cdot 3^k$, $38 \cdot 3^k$, and $(3^{n+1} + 1)3^k$ can be done via Lemma 2.5.4 (for 512, 56, 80, and 38) and Lemma 2.5.5 (for 13, 55 and $3^{n+1} + 1$). □

Proposition 2.5.15. *We have*

$$\begin{aligned} B_{11\delta(2)} \quad = \quad & B_{10\delta(2)} \cup \{1024, 26, 112, 37, 160, 110, 76\} \cup \\ & \{2(3 \cdot 3^n + 1) : n \geq 3\} \cup \{2 \cdot 3^n + 1 : n \geq 4\}, \end{aligned}$$

and the elements of $A_{11\delta(2)}$ have the complexities listed in Theorem 2.5.1.

Proof. By the main theorem and the above computations,

$$\begin{aligned} B_{11\delta(2)} \setminus B_{10\delta(2)} \subseteq\ & \{1024, 26, 112, 37, 160, 110, 76, 25\} \cup \\ & \{2(3 \cdot 3^n + 1) : n \geq 3\} \cup \{2 \cdot 3^n + 1 : n \geq 4\} \cup \\ & \{512 \cdot 3^n + 1 : n \geq 0\} \cup \{13 \cdot 3^n + 1 : n \geq 0\} \cup \\ & \{56 \cdot 3^n + 1 : n \geq 0\} \cup \{80 \cdot 3^n + 1 : n \geq 0\} \cup \\ & \{55 \cdot 3^n + 1 : n \geq 0\} \cup \{38 \cdot 3^n + 1 : n \geq 0\} \cup \\ & \{(3 \cdot 3^n + 1)3^m + 1 : n \geq 3, m \geq 0\} \end{aligned}$$

Lemma 2.5.5 shows that for $m \geq 3$, $\|(3^{m+1}+1)3^n+1\| = 2+3(m+1)+3n$, and that for $n \geq 1$, $\|512 \cdot 3^n + 1\| = 19 + 3n$, $\|56 \cdot 3^n + 1\| = 13 + 3n$, $\|80 \cdot 3^n + 1\| = 14 + 3n$, $\|55 \cdot 3^n + 1\| = 13 + 3n$, $\|38 \cdot 3^n + 1\| = 12 + 3n$, and that for $n \geq 2$, $\|13 \cdot 3^n + 1\| = 9 + 3n$. Using this to check their defects, we see that none of these lie in $A_{11\delta(2)}$, or even $A_{12\delta(2)}$. We checked earlier that $\delta(25) > 11\delta(2)$. Finally, checking the complexities of $1024 \cdot 3^k$, $26 \cdot 3^k$, $112 \cdot 3^k$, $37 \cdot 3^k$, $160 \cdot 3^k$, $110 \cdot 3^k$, $76 \cdot 3^k$, $2(3^{n+1} + 1)3^k$, and $(2 \cdot 3^n + 1)3^k$ can be done via Lemma 2.5.4 (for 1024, 26, 112, 160, 110, 76, and $2(3^{n+1} + 1)$) and Lemma 2.5.5 (for 37 and $2 \cdot 3^n + 1$). □

Proposition 2.5.16.

$$\begin{aligned} B_{12\delta(2)} =\ & B_{11\delta(2)} \cup \{2048, 25, 52, 224, 74, 320, 17, 220, 152, 73\} \cup \\ & \{4(3 \cdot 3^n + 1) : n \geq 3\} \cup \{2(2 \cdot 3^n + 1) : n \geq 4\} \cup \\ & \{4 \cdot 3^n + 1 : n \geq 3\} \end{aligned}$$

and the elements of $A_{12\delta(2)}$ have the complexities listed in Theorem 2.5.1.

Proof. By the main theorem and the above computations,

$$\begin{aligned} B_{12\delta(2)} \setminus B_{11\delta(2)} \subseteq\ & \{2048, 25, 52, 224, 74, 320, 17, 220, 152, 73, 35\} \cup \\ & \{4(3 \cdot 3^n + 1) : n \geq 3\} \cup \{2(2 \cdot 3^n + 1) : n \geq 4\} \cup \\ & \{4 \cdot 3^n + 1 : n \geq 3\} \cup \{1024 \cdot 3^n + 1 : n \geq 0\} \cup \\ & \{26 \cdot 3^n + 1 : n \geq 0\} \cup \{112 \cdot 3^n + 1 : n \geq 0\} \cup \\ & \{37 \cdot 3^n + 1 : n \geq 0\} \cup \{160 \cdot 3^n + 1 : n \geq 0\} \cup \\ & \{110 \cdot 3^n + 1 : n \geq 0\} \cup \{76 \cdot 3^n + 1 : n \geq 0\} \cup \\ & \{2(3 \cdot 3^n + 1)3^m + 1 : n \geq 3, m \geq 0\} \cup \\ & \{(2 \cdot 3^n + 1)3^m + 1 : n \geq 4, m \geq 0\} \end{aligned}$$

Lemma 2.5.5 shows that for $m \geq 3$ and $n \geq 1$,

$$\|2(3^{m+1}+1)3^n+1\| = 4+3(m+1)+3n,$$

that for $m \geq 4$ and $n \geq 1$,

$$\|(2\cdot 3^m+1)3^n+1\| = 4+3m+3n,$$

and that

$$\|1024\cdot 3^n+1\| = 21+3n, \quad \|112\cdot 3^n+1\| = 15+3n,$$
$$\|160\cdot 3^n+1\| = 16+3n, \quad \|76\cdot 3^n+1\| = 14+3n,$$

and that for $n \geq 1$,

$$\|26\cdot 3^n+1\| = 11+3n, \qquad \|110\cdot 3^n+1\| = 15+3n$$

, and that for $n \geq 2$,

$$\|37\cdot 3^n+1\| = 12+3n.$$

Using this to check their defects, we see that none of these lie in $A_{12\delta(2)}$. We can then check that $\delta(35) > 12\delta(2)$. Finally, checking the complexities of $2048\cdot 3^k$, $25\cdot 3^k$, $52\cdot 3^k$, $224\cdot 3^k$, $74\cdot 3^k$, $320\cdot 3^k$, $220\cdot 3^k$, $152\cdot 3^k$, $73\cdot 3^k$, $4(3^{n+1}+1)3^k$, $2(2\cdot 3^n+1)3^k$, and $(4\cdot 3^n+1)3^k$ can be done via Lemma 2.5.4 (for 2048, 25, 52, 224, 74, 320, 220, 152, $4(3^{n+1}+1)$, and $2(2\cdot 3^n+1)$) and Lemma 2.5.5 (for 25, 17, 73, and $4\cdot 3^n+1$). □

Combining all these propositions establishes Theorem 2.5.1.

2.6 Applications

We now present several applications of the classification obtained in Section 2.5. These are: (i) Stability of numbers $n > 1$ of defect less than $12\delta(2)+1$; (ii) Classification of all integers n having defect $0 \leq \delta(n) \leq 1$ and finiteness of B_r for all $r < 1$; (iii) Determination of complexities $\|2^a\cdot 3^k\|$ for $a \leq 21$ and all k; (iv) Upper bounds on the number of integers $n \leq x$ having complexity $\delta(n) < r$, for any fixed $r > 0$.

2.6.1 Stability of numbers of low defect

We have already noted in Theorem 2.5.1 that numbers $n > 1$ of defect less than $12\delta(2)$ are stable. In fact, we can conclude something stronger.

Theorem 2.6.1. *If $n > 1$ and $\delta(n) < 12\delta(2) + 1 = 2.2865\ldots$, then n is stable.*

Proof. From Theorem 2.5.1, we can check that if $\delta(3n) < 12\delta(2)$, then

$$\delta(n) < 12\delta(2).$$

So suppose the theorem were false, and we have unstable $n > 1$ with

$$\delta(n) < 12\delta(2) + 1.$$

Then for some K, $\delta(3^K n) \leq \delta(n) - 1 < 12\delta(2)$. So by above, we have $\delta(n) < 12\delta(2)$, and thus, as noted in Theorem 2.5.1, n is stable unless $n = 1$. □

In fact, if $n > 1$ and $\delta(n) < \delta(107) = 3.2398\ldots$, then n is stable, as we will prove in Chapter 5.

2.6.2 Classifying the integers of defect at most 1

Using Theorem 2.5.1 we can classify all the numbers with defect less than 1, as follows:

Theorem 2.6.2. *The natural numbers n satisfying $\delta(n) < 1$ are precisely those that can be written in one of the following forms, and have the following complexities:*

1. 3^k *for* $k \geq 1$, *of complexity* $3k$
2. $2^a 3^k$ *for* $a \leq 9$, *of complexity* $2a + 3k$ *(for a, k not both zero)*
3. $5 \cdot 2^a 3^k$ *for* $a \leq 3$, *of complexity* $5 + 2a + 3k$
4. $7 \cdot 2^a 3^k$ *for* $a \leq 2$, *of complexity* $6 + 2a + 3k$
5. $19 \cdot 3^k$ *of complexity* $9 + 3k$
6. $13 \cdot 3^k$ *of complexity* $8 + 3k$
7. $(3^n + 1)3^k$ *of complexity* $1 + 3n + 3k$ *(for $n \neq 0$)*

Furthermore $n = 1$ is the only number having defect exactly 1.

Proof. This list includes all numbers in $A_{9\delta(2)}$, and some numbers in $A_{10\delta(2)}$. These in turn are determined by the corresponding lists for $B_{9\delta(2)}, B_{10\delta(2)}$, in the latter case (Proposition 2.5.14) checking the complexities to exclude the leaders $\{56, 80, 55, 38\}$. □

Using this list one may deduce the following important fact.

Theorem 2.6.3. *For every $\alpha \in (0,1)$, the set of leaders B_α is a finite set. For every $\alpha \geq 1$, the set B_α is an infinite set.*

Proof. The first part follows from the fact that each of the categories above has a finite set of leaders, and that the final list (7) has a finite number of sublists with defect smaller than $1 - \epsilon$, for any epsilon. The defects

$$\delta((3^n+1)3^k) = (3n+1) - 3\log_3(3^n+1) = 1 - 3\log_3(1+\frac{1}{3^n})$$

approach 1 from below as n approaches infinity. This also establishes that B_1 is an infinite set, giving the second part. □

2.6.3 The complexity of 2^m3^k for small m

The determination of A_r in Theorem 2.5.1 allows us to put lower bounds on the complexities of any numbers not in it. Thus for instance we have the following result.

Lemma 2.6.4. *Let n be a natural number and suppose that there is no k such that $2^{n+9}3^k \in A_{n\delta(2)}$. Then for any $m \leq n+9$ and any k (with m and k not both zero), $\|2^m3^k\| = 2m+3k$.*

Proof. It suffices to show that $\|2^{n+9}3^k\| > 2n+3k+17$, but by assumption,

$$\|2^{n+9}3^k\| \geq (n+9)3\log_3 2 + 3k + n\delta(2) = 2n+3k+27\log_3 2 > 2n+3k+17,$$

and we are done. □

This lemma immediately establishes Conjecture 2.1.3 for $a \leq 21$.

Proof of Theorem 2.1.7. From our classification, it is straightforward to check that $2^{21}3^k$ does not lie in $A_{12\delta(2)}$ for any k, so we can conclude that for $m \leq 21$ and any k, with m and k not both zero, $\|2^m3^k\| = 2m+3k$. □

2.6.4 Counting the integers below x having defect at most r

In our computations in Section 2.5, we used a small step size $\alpha = \delta(2)$, and kept our superset of A_r small by using a pruning step. In what follows, we will use a different trick to keep our supersets of A_r from getting too large. Instead of pruning, we will use step sizes arbitrarily close to 1.

Proposition 2.6.5. *Given any $\alpha \in (0,1)$ and any $k \geq 1$, we have that*

$$B_{k\alpha}(x) = O_{k\alpha}((\log x)^{k-1}) \quad \textit{and} \quad A_{k\alpha}(x) = O_{k\alpha}((\log x)^k),$$

where $S(x)$ denotes the number of elements of S that are less than x.

Proof. We induct on k. Suppose $k = 1$; by Corollary 2.6.3, then $B_{k\alpha} = B_\alpha$ is a finite set, so $B_{k\alpha}(x) = O_{k\alpha}(1)$. Also, for any r, $A_r(x) \leq B_r(x)(\log_3 x)$; in particular, $A_{k\alpha}(x) = O_{k\alpha}(\log x)$.

So suppose it is true for k and we want to prove it for $k+1$; we apply Proposition 2.4.4 with step size α. For convenience, let S_r denote the set of solid numbers b satisfying $\|b\| < r + 3\log_3 2$, as mentioned in the discussion after Theorem 2.4.4; for any r, this is a finite set.

In the case $k+1 = 2$,

$$\begin{aligned} B_{2\alpha}(x) &\leq B_\alpha(x)^3 + (A_\alpha(x)|S_{2\alpha}| + |T_\alpha|)(|B_\alpha| + 1) \\ &= O_\alpha(1)^3 + O_\alpha(\log x) + O_\alpha(1) \\ &= O_{(k+1)\alpha}(\log x). \end{aligned}$$

In the case $k+1 > 2$,

$$\begin{aligned} B_{(k+1)\alpha}(x) &\leq \sum_{\substack{i+j=k+2 \\ i,j\geq 2}} B_{i\alpha}(x)B_{j\alpha}(x) + (A_{k\alpha}(x)|S_{(k+1)\alpha}| + |T_\alpha|)(|B_\alpha| + 1) \\ &= \sum_{\substack{i+j=k+2 \\ i,j\geq 2}} O_{i\alpha}((\log x)^{i-1})O_{j\alpha}((\log x)^{j-1}) + O_{(k+1)\alpha}((\log x)^k) + O_\alpha(1) \\ &= O_{k\alpha}((\log x)^k). \end{aligned}$$

In either case, we also have $A_{(k+1)\alpha}(x) = O_{(k+1)\alpha}((\log x)^{k+1})$. This completes the proof.

□

Using this result we conclude:

Theorem 2.6.6. *For any number $r > 0$,*

$$B_r(x) = \Theta_r((\log x)^{\lfloor r \rfloor}) \quad \textit{and} \quad A_r(x) = \Theta_r((\log x)^{\lfloor r \rfloor + 1}).$$

Proof. For the upper bound, it suffices to note that $r = (\lfloor r \rfloor + 1)\frac{r}{\lfloor r \rfloor + 1}$, and that $\frac{r}{\lfloor r \rfloor + 1} < 1$, and apply Proposition 2.6.5.

For the lower bound, let $k = \lfloor r \rfloor$, and consider numbers of the form

$$N = ((\cdots((3 \cdot 3^{n_k} + 1)3^{n_{k-1}} + 1)\cdots)3^{n_1} + 1)3^{n_0}.$$

Then

$$\|N\| \leq 3(n_0 + \cdots + n_k + 1) + k$$

and since $\log_3 N \geq n_0 + \cdots + n_k + 1$, this means $\delta(N) \leq k$. Furthermore, if $n_0 = 0$ and $n_1 > 0$ then N is not divisible by 3 and so is a leader. It is then easy to count that there are at least

$$\binom{\lfloor \log_3 x \rfloor}{k+1} \gtrsim \frac{1}{(k+1)!}(\log_3 x)^{k+1}$$

such N less than a given x, and at least

$$\binom{\lfloor \log_3 x \rfloor}{k} \gtrsim \frac{1}{k!}(\log_3 x)^{k}$$

if we insist that N be a leader. □

An immediate consequence of Theorem 2.6.6 is Theorem 2.1.8 in the introduction.

Proof of Theorem 2.1.8. The existence of numbers of arbitrarily large defect follows from the fact that the set of integers of defect $< r$ has density zero. □

This result is a long way from proving a bound of the type $\|n\| \nsim 3 \log_3 n$.

Chapter 3

Integer Complexity and Well-Ordering

Abstract: Define $\|n\|$ to be the *complexity* of n, the smallest number of ones needed to write n using an arbitrary combination of addition and multiplication. John Selfridge showed that $\|n\| \geq 3\log_3 n$ for all n. Define the *defect* of n, denoted $\delta(n)$, to be $\|n\| - 3\log_3 n$. In this chapter, we consider the set $\mathscr{D} := \{\delta(n) : n \geq 1\}$ of all defects. We show that as a subset of the real numbers, the set $\mathscr{D}$ is well-ordered, of order type ω^ω. More specifically, for $k \geq 1$ an integer, $\mathscr{D} \cap [0, k)$ has order type ω^k. We also consider some other sets related to $\mathscr{D}$, and show that these too are well-ordered and have order type ω^ω.

3.1 Introduction

The *complexity* of a natural number n is the least number of 1's needed to write it using any combination of addition and multiplication, with the order of the operations specified using parentheses grouped in any legal nesting. For instance, $n = 11$ has a complexity of 8, since it can be written using 8 ones as $(1+1+1)(1+1+1)+1+1$, but not with any fewer. This notion was implicitly introduced in 1953 by Kurt Mahler and Jan Popken [38]; they actually considered the inverse function of the size of the largest number representable using k copies of the number 1. (More generally, they considered the same question for representations using k copies of a positive real number x.) Integer complexity was explicitly studied by John Selfridge, and was later popularized by Richard Guy [29, 30]. Following J. Arias de Reyna [8] we will denote the complexity of n by $\|n\|$.

Integer complexity is approximately logarithmic; it satisfies the bounds

$$3\log_3 n = \frac{3}{\ln 3}\ln n \leq \|n\| \leq \frac{3}{\ln 2}\ln n, \qquad n > 1. \tag{3.1}$$

The lower bound can be deduced from the result of Mahler and Popken, and was explicitly proved by John Selfridge [29]. It is attained with equality for $n = 3^k$ for

all $k \geq 1$. The upper bound can be obtained by writing n in binary and finding a representation using Horner's algorithm. It is not sharp, and the constant $\frac{3}{\ln 2}$ can be improved for large n [52].

The notion of integer complexity is similar in spirit but different in detail from the better known measure of *addition chain length*, which has application to computation of powers, and which is discussed in detail in Knuth [35, Sect. 4.6.3]. One important difference between the two notions is that integer complexity can be computed by dynamic programming, while this does not seem to be the case for addition chain length. Specifically, integer complexity is computable via the dynamic programming recursion, for any $n > 1$,

$$\|n\| = \min_{\substack{a,b<n\in\mathbb{N} \\ a+b=n \text{ or } ab=n}} \|a\| + \|b\|.$$

There are many mysteries about $\|n\|$. For powers one has

$$\|n^k\| \leq k\|n\|$$

and it is known that $\|3^k\| = 3k$ for all $k \geq 1$. However other values have a more complicated behavior. For instance, powers of 5 do not work nicely, as $\|5^6\| = 29 < 30 = 6\|5\|$. The behavior of powers of 2 remains unknown; it has been verified that

$$\|2^k\| = k\|2\| = 2k \text{ for } 1 \leq k \leq 39;$$

see [33].

3.1.1 Main Result

In Chapter 2, we introduced the notion the *defect* of an integer n, denoted $\delta(n)$, by

$$\delta(n) := \|n\| - 3\log_3 n.$$

This is a rescaled version of integer complexity, which, given n, contains equivalent information to $||n||$. In view of the lower bound 3.1 above it satisfies $\delta(n) \geq 0$. Chapter 2 exploited patterns in the dynamic programming structure of integer complexity to classify the structure of all integers with small values of the defect. In particular it classifies all integers with $\delta(n) \leq 1$.

The defect encodes interesting structure about integer complexity. In this chapter, we will consider the image of this defect function in the general case:

Definition 3.1.1. The *defect set* $\mathscr{D} \subseteq [0, \infty)$ is the set of all defect values $\{\delta(n) : n \in \mathbb{N}\}$.

Addition and multiplication tend to interact badly and unpredictably when placed on an equal footing. So one might not expect to find any particular sort of structure in the values of $\delta(n)$, even though its definition is based on powers of 3 which give the extremal case. In this chapter we will prove the following striking result:

Theorem 3.1.2. *The set $\mathscr{D}$ is a well-ordered subset of $\mathbb{R}$, of order type ω^ω. Furthermore, for $k \geq 1$ an integer, the set $\mathscr{D} \cap [0, k)$ has order type ω^k.*

This well-ordering of the defect set $\mathscr{D}$ reveals new fundamental structure in the interaction between addition and multiplication. Some of the tangledness of that interaction may be reflected in how the set $\mathscr{D}$ grows more complicated as its elements get larger. In fact the structure of $\mathscr{D}$ has even more regularity than what Theorem 3.1.2 describes, which we plan to discuss in a future paper[6].

In Section 3.7, we will also prove that Theorem 3.1.2 still holds even if we replace $\mathscr{D}$ with any of several other closely-related sets.

Theorem 3.1.2 is closely related to conjectures of J. Arias de Reyna [8] about integer complexity. We discuss these conjectures and use our results to prove modified versions of some of them in Appendix A.

In contrast to Theorem 3.1.2, little is known about the set of values of $\frac{\|n\|}{3 \log_3 n}$, even though that might appear to be a more natural object of study. An open question is to determine the value

$$C_{max} := \limsup_{n \to \infty} \frac{\|n\|}{3 \log_3 n}.$$

The bounds 3.1 imply $1 \leq C_{max} \leq \log_2 3$. It is an open problem to decide whether $C_{max} = 1$ or $C_{max} > 1$ holds.

3.1.2 Low-Defect Polynomials

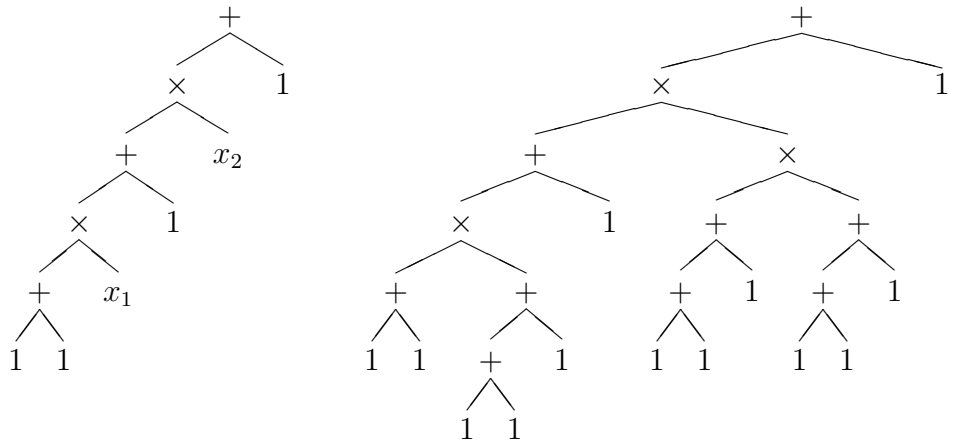
The strategy to prove the main theorem is to build up the set $\mathscr{D}$ by inductively building up the sets $\mathscr{D} \cap [0, s)$ for real numbers $s > 0$. The proof of Theorem 3.1.2 makes use of the methods of Chapter 2 classifying numbers of low defect. Chapter 2 gave a method to list families of such integers, and explicitly listed all integers of defect $\delta(n) < 1$. The innovation made here is that instead of treating the output of this method as an undifferentiated blob, we group it into tractable families.

We introduce a family of multilinear polynomials that we call *low-defect polynomials*. We show that for any $s > 0$, there exists a finite set of low-defect polynomials $\mathcal{S}_s$

such that any number of defect less than s can be written as $f(3^{n_1}, \ldots, 3^{n_k})3^{n_{k+1}}$ for some $f \in \mathcal{S}_s$ and nonnegative $n_1, \ldots, n_{k+1}$. Indeed, stronger statements are true; see Theorem 3.4.10 and Theorem 3.4.15. Note, however, that the low-defect polynomials may also produce extraneous numbers, with defect higher than intended; examples of these are given after Theorem 3.4.10. We will remedy this deficiency in Chapter 5.

To state this another way, these low-defect polynomials provide forms into which powers of 3 can be substituted to obtain all the numbers below the specified defect. As the defects get larger, the low-defect polynomials and the families of numbers we get this way become more complicated. And just as we can visualize expressions in $+$, $\times$, and 1 as trees, we can also visualize low-defect polynomials – or the expressions that generate them – as trees, with open slots where powers of 3 can be plugged in. By attaching trees corresponding to powers of 3, we obtain trees for the numbers we get this way. This is illustrated in Figure 3.1 with the polynomial $(2x_1+1)x_2+1$. (Note, however, that this picture is not quite correct when we plug in 3^0; see Figure 3.2 in section 3.4).

Figure 3.1: A tree corresponding to the polynomial $(2x_1 + 1)x_2 + 1$, and the same tree after making the substitution $x_1 = 3^1$, $x_2 = 3^2$.



So with this approach, we can get at properties of the set of defects by examining properties of low-defect polynomials. For instance, as mentioned above, as the defects involved get larger, the low-defect polynomials required get more complicated; one way in which this occurs is that they require more variables. In fact, we will see (Theorem 3.4.10) that to cover defects up to a real number s, one needs low-defect polynomials with up to $\lfloor s \rfloor$ variables. And it happens that if we have a low-defect polynomial f in k variables, and consider the numbers $f(3^{n_1}, \ldots, 3^{n_k})$, then the defects

of the numbers obtained this way form a well-ordered set of order type at least ω^k and less than ω^{k+1} (Proposition 3.6.3). It is this that leads us to Theorem 3.1.2, that for $k \geq 1$, the set $\mathscr{D} \cap [0, k)$ has order type precisely ω^k. In future papers we will draw more detailed conclusions by examining the structure of low-defect polynomials more closely.

3.1.3 Variant Results

We also prove analogues of the main theorem for several other sets. Chapter 2 showed that given the value of $\delta(n)$, one can determine the value of $\|n\|$ modulo 3; see Theorem 3.2.1(6) below. It follows that one can split the set of defects $\mathscr{D}$ into sets $\mathscr{D}^0$, $\mathscr{D}^1$, and $\mathscr{D}^2$ according to these congruence classes modulo 3; see Definition 3.2.4. In Section 3.7 we prove analogues of the main theorem for each set $\mathscr{D}^a$ separately; see Theorem 3.7.4.

Chapter 2 also introduced a notion of stable numbers; a number n is said to be *stable* if $\|3^k n\| = 3k + \|n\|$ for all $k \geq 0$; equivalently, if $\delta(3^k n) = \delta(n)$ for all $k \geq 0$. In Section 3.3 we show that given $\delta(n)$, one can determine whether or not a given number n is stable, and thus we can consider the set of "stable defects", $\mathscr{D}_{st}$, which are the defect values for all stable numbers.

We can combine this notion with splitting based on the value of $\|n\|$ modulo 3 to define sets $\mathscr{D}^0_{st}$, $\mathscr{D}^1_{st}$, and $\mathscr{D}^2_{st}$. In Section 3.7 we prove each of these sets is well-ordered of type ω^ω, as are the closures of all these sets. All these well-ordering results are collected in Theorem 3.7.4.

3.1.4 Computability Questions

Integer complexity captures part of the complicated interaction of addition and multiplication, where subtraction is not allowed; the underlying algebraic structure is that of a commutative semiring $(\mathbb{N}, +, \times)$. It is a very simple computational model, but already exhibits difficult issues.

The model of computation treated in this thesis could be considered as taking number inputs other than 1. Mahler and Popken [38] considered constructing numbers starting with copies of any fixed positive real number x. Note that as x varies the ordering of computed quantities on the positive real line will change. One feature of complexity for $x = 1$ (or for $x = k$, an integer) is that multiple ties can occur in doing the computations, which complicates determination of the structure of the minimal computation tree. For a generic (transcendental) x, the complexity issue

simplifies to viewing the computation tree as computing a univariate polynomial with positive integer coefficients, having a zero constant term. One can assign a complexity to the problem of computing such polynomials. Study of this simplified problem might be fruitful. Allowing multiple indeterminates as inputs, we can consider the complexity of computing multivariate polynomials, which is a much-studied topic. The model of computation allowing + and × above can compute all multivariate polynomials with nonnegative integer coefficients, but is restricted in that it does not allow free reuse of polynomials already constructed. The complexity of computation in this restricted model can be compared to that in other computational models which allow additional operations beyond addition and multiplication, or allow free reuse of already computed polynomials (straight-line computation). It is much easier to compute polynomials in models with subtraction [51] or division [25] than with only addition and multiplication [16, 28, 34, 40]. Indeed, similar phenomena occur in the computation of integers as well as that of polynomials [14].

We can also ask about the computational complexity of integer complexity itself, or related notions, viewed in the polynomial hierarchy of complexity theory (see Garey and Johnson [26, Sect. 7.2]). An open question concerns the computational complexity of computing $\|n\|$. Consider the problem:

INTEGER COMPLEXITY

- INSTANCE: Positive integers n and k, both encoded in binary.
- QUESTION: Is $\|n\| \leq k$?

This problem is known to be in the complexity class NP (Arias de Reyna [8]), but it is not known to be either in P or in co-NP, nor is it known to be NP-complete.

This thesis introduces the ordering of defects as an object of investigation. Hence we can also consider the problem:

DEFECT ORDERING

- INSTANCE: Positive integers n_1 and n_2, both encoded in binary.
- QUESTION: Is $\delta(n_1) \leq \delta(n_2)$?

This problem, of computing the defect ordering is not known to be in the complexity class NP. If one could answer INTEGER COMPLEXITY in polynomial time, then

one could also answer DEFECT ORDERING in polynomial time. To show this, observe that the inequality $\delta(n_1) \leq \delta(n_2)$ is equivalent to

$$3^{\|n_1\|}(n_2)^3 \leq 3^{\|n_2\|}(n_1)^3,$$

and since $\|n\|$ is logarithmically small, this could be computed in polynomial time if one knew $\|n\|$. This argument shows that DEFECT ORDERING belongs to the complexity class $P^{NP} = \Delta_2^P$.

Another question related to the defect is that of computing a set $\mathcal{S}_s$ of low-defect polynomials sufficient to describe all integers of defect $\delta(n) < s$, i.e., a set $\mathcal{S}_s$ satisfying the conditions of Theorem 3.4.10. What is the minimal cardinality of such a set, as a function of s? What is the complexity of computing one (say for s integral, or rational)? The proof of Theorem 3.4.10 does give a construction of one such set $\mathcal{S}_s$; however there exist other such sets $\mathcal{S}_s$, perhaps some smaller or computable more quickly than the one constructed.

3.2 Properties of the defect

We begin by reviewing the relevant properties of integer complexity and the defect from Chapter 2. They can be summed up in the following theorem:

Theorem 3.2.1. *We have:*

1. *For all n, $\delta(n) \geq 0$.*
2. *For $k \geq 0$, $\delta(3^k n) \leq \delta(n)$, with equality if and only if $\|3^k n\| = 3k + \|n\|$. The difference $\delta(n) - \delta(3^k n)$ is a nonnegative integer.*
3. *If the difference $\delta(n) - \delta(m)$ is rational, then $n = m3^k$ for some integer k (and so $\delta(n) - \delta(m) \in \mathbb{Z}$).*
4. *Given any n, there exists L such that for all $k \geq L$, $\delta(3^k n) = \delta(3^L n)$. That is to say, $\|3^k n\| = \|3^L n\| + 3(k - L)$.*
5. *For a given defect α, the set $\{m : \delta(m) = \alpha\}$ has either the form $\{n3^k : 0 \leq k \leq L\}$ for some n and L, or the form $\{n3^k : 0 \leq k\}$ for some n. This latter occurs if and only if α is the smallest defect among $\delta(3^k n)$ for $k \in \mathbb{Z}$.*
6. *If $\delta(n) = \delta(m)$, then $\|n\| = \|m\| \pmod 3$.*
7. *$\delta(1) = 1$, and for $k \geq 1$, $\delta(3^k) = 0$. No other integers occur as $\delta(n)$ for any n.*

Proof. Part (1) is just Selfridge's lower bound [29]. The first statement in part (2) is Proposition 2.2.1(3); the second statement follows from the computation $\delta(n) - \delta(3^k n) = \|n\| - \|3^k n\| + 3k$. Part (3) is Proposition 2.2.6(1). Parts (4) and (5) are Theorem 2.1.5. Part (6) is part of Proposition 2.2.6(2). For part (7), the fact that $\delta(1) = 1$ is immediate. The fact that $\delta(3^k) = 0$ for $k \geq 1$ is the same as the fact that $\|3^k\| = 3k$ for $k \geq 1$; that $\|3^k\| \leq 3k$ is obvious, and that $\|3^k\| \geq 3k$ follows from Selfridge's lower bound [29]. Finally, that no other integers occur as $\delta(n)$ for any n follows from part (3). □

We also recall the definitions made for discussing the above:

Definition 3.2.2. A number m is called *stable* if $\|3^k m\| = 3k + \|m\|$ holds for every $k \geq 1$, or equivalently if $\delta(3^k m) = \delta(m)$ for every $k \geq 1$. Otherwise it is called *unstable*.

Definition 3.2.3. A natural number n is called a *leader* if it is the smallest number with a given defect. By part (5) of Theorem 3.2.1, this is equivalent to saying that either $3 \nmid n$, or, if $3 \mid n$, then $\delta(n) < \delta(n/3)$, i.e., $\|n\| < 3 + \|n/3\|$.

Also, because of part (6) of Theorem 3.2.1, we can make the following definitions:

Definition 3.2.4. For a a congruence class modulo 3, we define

$$\mathscr{D}^a = \{\delta(n) : \|n\| \equiv a \pmod 3,\ n \neq 1\}$$

We explicitly exclude the number 1 here as it is dissimilar to other numbers whose complexity is congruent to 1 modulo 3. This is because, unlike other numbers which are 1 modulo 3, the number 1 cannot be written as $3j + 4$ for some j, and so the largest number that can be made with a single 1 is simply 1, rather than $4 \cdot 3^j$ (see Appendix A). For this reason, numbers of complexity 1 do not really go together with other numbers whose complexity is congruent to 1 modulo 3; however, the only such number is 1, so we simply explicitly exclude it. So $\mathscr{D}$ is the disjoint union of $\mathscr{D}^0$, $\mathscr{D}^1$, $\mathscr{D}^2$, and $\{1\}$.

Of course, we care not just about small defects, but about the numbers giving rise to those small defects; so we recall the following definitions:

Definition 3.2.5. For any real $r \geq 0$, define the set of *r-defect numbers* A_r to be

$$A_r := \{n \in \mathbb{N} : \delta(n) < r\}.$$

Define the set of *r-defect leaders* B_r to be

$$B_r := \{n \in A_r : \ n \text{ is a leader}\}.$$

These sets are related by:

Proposition 3.2.6. *For every $n \in A_r$, there exist a unique $m \in B_r$ and $k \geq 0$ such that $n = 3^k m$ and $\delta(n) = \delta(m)$; then $\|n\| = \|m\| + 3k$.*

Proof. The first part of this is Proposition 2.2.8(2). The second part follows as then

$$\|n\| = \delta(n) + 3\log_3(3^k m) = 3k + \delta(m) + 3\log_3 m = \|m\| + 3k.$$

□

3.2.1 Inductive covering of B_r and A_r

In addition to the above properties of the defect, there are two substantive theorems we will need from Chapter 2. They allow us to inductively build up the sets A_r and B_r, or at least coverings of these. The first provides the base case:

Theorem 3.2.7. *For every α with $0 < \alpha < 1$, the set of leaders B_α is a finite set.*

The other theorem provides the inductive step, telling us how to build up $B_{(k+1)\alpha}$ from previous $B_{i\alpha}$. In order to state it we'll first need some definitions.

Definitions 3.2.8. We say n is *most-efficiently* represented as ab if $n = ab$ and $\|n\| = \|a\| + \|b\|$, or as $a + b$ if $n = a + b$ and $\|n\| = \|a\| + \|b\|$. In the former case we will also say that $n = ab$ is a *good factorization* of n. We say n is *solid* if it cannot be written most-efficiently as $a+b$ for any a and b. We say n is *m-irreducible* if it cannot be written most-efficiently as ab for any a and b. And for a real number $\alpha \in (0, 1)$, we define the set T_α to consist of 1 together with those m-irreducible numbers n which satisfy

$$\frac{1}{n-1} > 3^{\frac{1-\alpha}{3}} - 1$$

and do not satisfy $\|n\| = \|n - b\| + \|b\|$ for any solid numbers b with $1 < b \leq n/2$.

Note that for any $\alpha \in (0, 1)$, the set T_α is a finite set, due to the upper bound on the size of numbers $n \in T_\alpha$.

Now we can state the theorem. The theorem provides fives possibilities; three "generic cases" (1 through 3), and two "exceptional cases" (4 and 5).

Theorem 3.2.9. *Suppose that $0 < \alpha < 1$ and that $k \geq 1$. Then any $n \in B_{(k+1)\alpha}$ can be most-efficiently represented in (at least) one of the following forms:*

1. *For $k = 1$, there is either a good factorization $n = u \cdot v$ where $u, v \in B_\alpha$, or a good factorization $n = u \cdot v \cdot w$ with $u, v, w \in B_\alpha$;*
   For $k \geq 2$, there is a good factorization $n = u \cdot v$ where $u \in B_{i\alpha}$, $v \in B_{j\alpha}$ with $i + j = k + 2$ and $2 \leq i, j \leq k$.

2. *$n = a + b$ with $\|n\| = \|a\| + \|b\|$, $a \in A_{k\alpha}$, $b \leq a$ a solid number and*

$$\delta(a) + \|b\| < (k+1)\alpha + 3\log_3 2.$$

3. *There is a good factorization $n = (a + b)v$ with $v \in B_\alpha$, $a + b$ being a most-efficient representation, and a and b satisfying the conditions in the case (2) above.*

4. *$n \in T_\alpha$ (and thus in particular either $n = 1$ or $\|n\| = \|n - 1\| + 1$.)*

5. *There is a good factorization $n = u \cdot v$ with $u \in T_\alpha$ and $v \in B_\alpha$.*

By applying these two theorems, we can inductively build up the sets B_r and A_r; in a sense they form the engine of our proof. However, without additional tools, it can be hard to say anything about just what these theorems output. In Section 3.4, we will show how to group the output of these theorems into tractable families, allowing us to go beyond the earlier work of Chapter 2 and prove the main theorem.

3.3 Stable defects and stable complexity

It will also be useful here to introduce the notion of "stable defect" and "stable complexity". First, let us discuss the defects of stable numbers.

Proposition 3.3.1. *If $\delta(n) = \delta(m)$ and n is stable, then so is m.*

Proof. Suppose $\delta(n) = \delta(m)$ and n is stable. Then we can write $m = 3^k n$ for some $k \in \mathbb{Z}$. Now, a number a is stable if and only if $\delta(3^\ell a) = \delta(a)$ for all $\ell \geq 0$; so if $k \geq 0$, then m is stable. If, on the other hand, $k < 0$, then consider $\ell \geq 0$. If $\ell \geq -k$, then $\delta(3^\ell m) = \delta(3^{\ell+k} n) = \delta(n)$, while if $\ell \leq -k$, then $\delta(n) \leq \delta(3^\ell m) \leq \delta(m)$, so $\delta(3^\ell m) = \delta(m)$; hence m is stable. □

Because of this proposition, it makes sense to make the following definition:

Definition 3.3.2. We define a *stable defect* to be the defect of a stable number, and define $\mathscr{D}_{st}$ to be the set of all stable defects. Also, for a a congruence class modulo 3, we define $\mathscr{D}_{st}^a = \mathscr{D}^a \cap \mathscr{D}_{st}$.

Note that the integer 1 is not stable, and so its defect, which is also 1, would be excluded from $\mathscr{D}_{st}^1$ even if we had not explicitly excluded it in the definition of $\mathscr{D}^1$.

This double use of the word "stable" could potentially be ambiguous if we had a positive integer n which were also a defect. However, the only positive integer which is also a defect is 1, which is not stable in either sense.

Proposition 3.3.3. *A defect α is stable if and only if it is the smallest $\beta \in \mathscr{D}$ such that $\beta \equiv \alpha \pmod 1$.*

Proof. This follows from parts (2), (3), and (5) of Theorem 3.2.1. □

Definition 3.3.4. For a positive integer n, define the *stable defect of n*, denoted $\delta_{st}(n)$, to be $\delta(3^k n)$ for any k such that $3^k n$ is stable. (This is well-defined as if $3^k n$ and $3^\ell n$ are stable, then $k \geq \ell$ implies $\delta(3^k n) = \delta(3^\ell n)$, and so does $\ell \geq k$.)

Here are two equivalent characterizations:

Proposition 3.3.5. *The number $\delta_{st}(n)$ can be characterized by:*

1. $\delta_{st}(n) = \min_{k \geq 0} \delta(3^k n)$
2. *$\delta_{st}(n)$ is the smallest $\alpha \in \mathscr{D}$ such that $\alpha \equiv \delta(n) \pmod 1$.*

Proof. Part (1) follows from part (2) Theorem 3.2.1 and the fact that m is stable if and only if $\delta(3^k m) = \delta(m)$ for all $k \geq 0$. To prove part (2), take k such that $3^k n$ is stable. Then $\delta(3^k n) \equiv \delta(n) \pmod 1$, and it is the smallest such by Proposition 3.3.3. □

So we can think about $\mathscr{D}_{st}$ either as the subset of $\mathscr{D}$ consisting of the stable defects, or we can think about it as the image of δ_{st}. (This latter way of thinking doesn't work so well for the $\mathscr{D}_{st}^a$, however.)

Just as we can talk about the stable defect of a number n, we can also talk about its *stable complexity* – what the complexity would be "if n were stable".

Definition 3.3.6. For a positive integer n, we define the *stable complexity of n*, denoted $\|n\|_{st}$, to be $\|3^k n\| - 3k$ for any k such that $3^k n$ is stable. This is well-defined; if $3^k n$ and $3^\ell n$ are both stable, say with $k \leq \ell$, then

$$\|3^k n\| - 3k = 3(k - \ell) + \|3^\ell n\| - 3k = \|3^\ell n\| - 3\ell.$$

Proposition 3.3.7. *We have:*

1. $\|n\|_{st} = \min_{k \geq 0}(\|3^k n\| - 3k)$
2. $\delta_{st}(n) = \|n\|_{st} - 3\log_3 n$

Proof. To prove part (1), observe that $\|3^k n\| - 3k$ is nonincreasing in k, since $\|3m\| \leq 3 + \|m\|$. So a minimum is achieved if and only if for all ℓ,

$$\|3^{k+\ell} n\| - 3(k+\ell) = \|3^k n\| - 3k,$$

i.e., for all ℓ, $\|3^{k+\ell} n\| = \|3^k n\| + 3\ell$, i.e., $3^k n$ is stable.

To prove part (2), take k such that $3^k n$ is stable. Then

$$\delta_{st}(n) = \delta(3^k n) = \|3^k n\| - 3\log_3(3^k n) = \|3^k n\| - 3k - 3\log_3 n = \|n\|_{st} - 3\log_3 n.$$

□

Proposition 3.3.8. *We have:*

1. $\delta_{st}(n) \leq \delta(n)$*, with equality if and only if* n *is stable.*
2. $\|n\|_{st} \leq \|n\|$*, with equality if and only if* n *is stable.*

Proof. The inequality in part (1) follows from Proposition 3.3.5. Also, if n is stable, then for any $k \geq 1$, we have $\delta(3^k n) = \delta(n)$, so $\delta_{st}(n) = \delta(n)$. Conversely, if $\delta_{st}(n) = \delta(n)$, then by Proposition 3.3.5, for any $k \geq 1$, we have $\delta(3^k n) \geq \delta(n)$. But also $\delta(3^k n) \leq \delta(n)$ by part (2) of Theorem 3.2.1, and so $\delta(3^k n) = \delta(n)$ and n is stable.

Part (2) follows from part (1) along with part (2) of Proposition 3.3.7. □

We will write more about the properties of $\|n\|_{st}$ in Chapter 5.

3.4 Low-defect polynomials

The primary tool we will use to prove the main theorem is to group the numbers produced by the main theorem of Chapter 2 into families. Each of these families will be expressed via a multilinear polynomial in $\mathbb{Z}[x_1, x_2, \ldots]$, which we will call a *low-defect polynomial*. We will associate these with a "base complexity" to form a *low-defect pair*. Formally:

Definition 3.4.1. We define the set $\mathscr{P}$ of *low-defect pairs* as the smallest subset of $\mathbb{Z}[x_1, x_2, \ldots] \times \mathbb{N}$ such that:

1. For any constant polynomial $k \in \mathbb{N} \subseteq \mathbb{Z}[x_1, x_2, \ldots]$ and any $C \geq \|k\|$, we have $(k, C) \in \mathscr{P}$.

2. Given (f_1, C_1) and (f_2, C_2) in $\mathscr{P}$, we have $(f_1 \otimes f_2, C_1 + C_2) \in \mathscr{P}$, where, if f_1 is in r_1 variables and f_2 is in r_2 variables,

$$(f_1 \otimes f_2)(x_1, \ldots, x_{r_1+r_2}) := f_1(x_1, \ldots, x_{r_1}) f_2(x_{r_1+1}, \ldots, x_{r_1+r_2}).$$

3. Given $(f, C) \in \mathscr{P}$, $c \in \mathbb{N}$, and $D \geq \|c\|$, we have $(f \otimes x_1 + c, C + D) \in \mathscr{P}$ where $\otimes$ is as above.

The polynomials obtained this way will be referred to as *low-defect polynomials*. If (f, C) is a low-defect pair, C will be called its *base complexity*. If f is a low-defect polynomial, we will define its *absolute base complexity*, denoted $\|f\|$, to be the smallest C such that (f, C) is a low-defect pair.

Note that the degree of a low-defect polynomial is also equal to the number of variables it uses; see Proposition 3.4.2. We will often refer to the "degree" of a low-defect pair (f, C); this refers to the degree of f.

Note that we do not really care about what variables a low-defect polynomial (or pair) is in – if we permute the variables of a low-defect polynomial or replace them with others, we will still regard the result as a low-defect polynomial. From this perspective, the meaning of $f \otimes g$ could be simply regarded as "relabel the variables of f and g so that they do not share any, then multiply f and g". Helpfully, the $\otimes$ operator is associative not only with this more abstract way of thinking about it, but also in the concrete way it was defined above.

One can actually get additional information by looking at not just the pair (f, C) but the process by how it was built-up and the underlying *low-defect expression* represented by the polynomial f; we will not need this at present, though. We will take this approach in Chapter 5, however; see Section 5.4 for more information on low-defect expressions.

3.4.1 Properties of low-defect polynomials

Let us begin by stating some structural properties of low-defect polynomials.

Proposition 3.4.2. *Suppose f is a low-defect polynomial of degree r. Then f is a polynomial in the variables $x_1, \ldots, x_r$, and it is a multilinear polynomial, i.e., it has degree 1 in each of its variables. The coefficients are non-negative integers. The*

constant term is nonzero, and so is the coefficient of $x_1 \dots x_r$, which we will call the leading coefficient *of f.*

Proof. We prove the statement by structural induction.

If the low-defect polynomial f is just a constant n, it has no variables and the leading coefficient and constant term are both n, which is positive.

If $f = g \otimes h$, say $f(x_1, \dots, x_r) = g(x_1, \dots, x_s)h(x_{s+1}, \dots, x_r)$, then by the inductive hypothesis f is a product of two polynomials whose coefficients are nonnegative integers, and thus so is f. To see that f is multilinear, consider a variable x_i; if $1 \leq i \leq s$, then x_i has degree 1 in $g(x_1, \dots, x_s)$ and degree 0 in $h(x_{s+1}, \dots, x_r)$, while if $r + 1 \leq i \leq s$, the reverse is true. Either way, x_i has degree 1 in f.

The coefficient of $x_1 \dots x_r$ in f is the product of the coefficient of $x_1 \dots x_s$ in g and the coefficient of $x_1 \dots x_{r-s}$ in h and so does not vanish, and the constant term of f is the product of the constant terms of g and h and so does not vanish.

Finally, if $f = g \otimes x_1 + c$, say $f(x_1, \dots, x_r) = g(x_1, \dots, x_{r-1})x_r + c$, then since g has coefficients that are nonnegative integers, so does f. To see that f is multilinear, consider a variable x_i; for $1 \leq i \leq r - 1$, the variable x_i has degree 1 in g and hence so does in f, while x_r has degree 0 in g and hence has degree 1 in f as well. Finally, the coefficient of $x_1 \dots x_r$ in f is the same as the coefficient of $x_1 \dots x_{r-1}$ in g and hence does not vanish, while the constant term of f is c, which is positive. □

We will also need the following lemma in Section 3.6:

Lemma 3.4.3. *For any low-defect polynomial f of degree $k > 0$, there exist low-defect polynomials g and h and a positive integer c such that $f = h \otimes (g \otimes x_1 + c)$.*

Proof. We apply structural induction. Since f has degree greater than zero, it is not a constant. Hence it can be written either as $f_1 \otimes f_2$ (in which case at least one of these has degree greater than zero) or as $g \otimes x_1 + c$. In the latter case we are done, writing $f = 1 \otimes (g \otimes x_1 + c)$.

In the former case, without loss of generality, say f_2 has degree $r > 0$. (Since if f_2 is a constant, we have $f_1 \otimes f_2 = f_2 \otimes f_1$.) Then by the inductive hypothesis, there are low-defect polynomials g_2 and h_2 and a positive integer c_2 such that $f_2 = h_2 \otimes (g_2 \otimes x_1 + c)$, so $f = (f_1 \otimes h_2) \otimes (g_2 \otimes x_1 + c)$, as needed. □

There is more that can be said about the structure of low-defect polynomials, as we will show in Chapter 5.

3.4.2 Numbers 3-represented by low-defect polynomials

We will obtain actual numbers from these polynomials by substituting in powers of 3 as mentioned in Section 3.1. Let us state here the following obvious but useful lemma:

Lemma 3.4.4. *For any a, b, and n, $\|ab^n\| \leq \|a\| + n\|b\|$.*

Proof. If $n \geq 1$, then $\|ab^n\| \leq \|a\| + \|b^n\| \leq \|a\| + n\|b\|$. Whereas if $n = 0$, then $\|ab^n\| = \|a\| = \|a\| + n\|b\|$. □

This provides an upper bound on the complexities of the outputs of these polynomials:

Proposition 3.4.5. *If (f, C) is a low-defect pair of degree r, then*

$$\|f(3^{n_1}, \ldots, 3^{n_r})\| \leq C + 3(n_1 + \ldots + n_r).$$

Proof. We prove the statement by structural induction. If f is a constant k, then $C \geq \|k\|$, and we are done.

If there are low-defect pairs (g_1, D_1) and (g_2, D_2) (say of degrees s_1 and s_2) such that $f = g_1 \otimes g_2$ and $C = D_1 + D_2$, then

$$\begin{aligned}\|f(3^{n_1}, \ldots, 3^{n_r})\| &\leq \|g_1(3^{n_1}, \ldots, 3^{n_{s_1}})\| + \|g_2(3^{n_{s_1+1}}, \ldots, 3^{n_r})\| \\ &\leq D_1 + D_2 + 3(n_1 + \ldots + n_r) = C + 3(n_1 + \ldots + n_r).\end{aligned}$$

In the last case, if there is a low-defect pair (g, D) and a constant c with $C \geq D + \|c\|$ such that $f = g \otimes x_1 + c$, we apply Lemma 3.4.4:

$$\begin{aligned}\|f(3^{n_1}, \ldots, 3^{n_r})\| &\leq \|g(3^{n_1}, \ldots, 3^{n_{r-1}})\| + 3n_r + \|c\| \\ \leq D + \|c\| + 3(n_1 + \ldots + n_r) &\leq C + 3(n_1 + \ldots + n_r).\end{aligned}$$

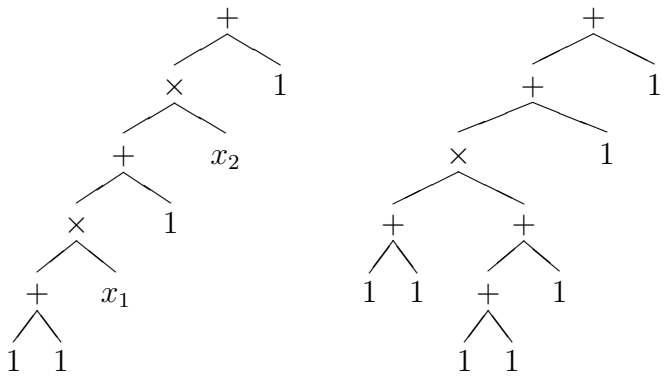
□

Note that because of the two cases in the proof of Lemma 3.4.4, the picture in Figure 3.1 is slightly inaccurate; this is only the picture when 3^k is plugged in for $k \geq 1$. See Figure 3.2 for an illustration of what happens when we plug in 3^0.

Because of Proposition 3.4.5, we define:

Definition 3.4.6. Given a low-defect pair (f, C) (say of degree r) and a number N, we will say that (f, C) *efficiently* 3*-represents* N if there exist nonnegative integers

Figure 3.2: A tree corresponding to the polynomial $(2x_1+1)x_2+1$, and the same tree after making the substitution $x_1 = 3^1$, $x_2 = 3^0$; observe how the top multiplication node disappears.



$n_1, \ldots, n_r$ such that $N = f(3^{n_1}, \ldots, 3^{n_r})$ and $\|N\| = C + 3(n_1 + \ldots + n_r)$. More generally, we will also say f 3-represents N if there exist nonnegative integers $n_1, \ldots, n_r$ such that $N = f(3^{n_1}, \ldots, 3^{n_r})$.

Note that if (f, C) efficiently 3-represents N, then $(f, \|f\|)$ efficiently 3-represents N, which means that in order for (f, C) to 3-represent anything efficiently at all, we must have $C = \|f\|$. However it is still worth using low-defect pairs rather than just low-defect polynomials since we may not always know $\|f\|$. This chapter will not be concerned with these sorts of computational issues, but in Chapter 5 we will discuss how to refine the theorems here to allow for computation.

For this reason it makes sense to use "f efficiently 3-represents N" to mean "some (f, C) efficiently 3-represents N" or equivalently "$(f, \|f\|)$ efficiently 3-reperesents N".

In keeping with the name, the numbers 3-represented by a low-defect polynomial have bounded defect. First let us make two definitions:

Definition 3.4.7. Given a low-defect pair (f, C), we define $\delta(f, C)$, the defect of (f, C), to be $C - 3\log_3 a$, where a is the leading coefficient of f. When we are not concerned with keeping track of base complexities, we will use $\delta(f)$ to mean $\delta(f, \|f\|)$.

Definition 3.4.8. Given a low-defect pair (f, C) of degree r, we define

$$\delta_{f,C}(n_1, \ldots, n_r) = C + 3(n_1 + \ldots + n_r) - 3\log_3 f(3^{n_1}, \ldots, 3^{n_r}).$$

We will also define δ_f to mean $\delta_{f,\|f\|}$ when we are not concerned with keeping track

of base complexities.

Then we have:

Proposition 3.4.9. *Let (f, C) be a low-defect pair of degree r, and let $n_1, \ldots, n_r$ be nonnegative integers.*

1. *We have*
$$\delta(f(3^{n_1}, \ldots, 3^{n_r})) \leq \delta_{f,C}(n_1, \ldots, n_r)$$
and the difference is an integer.

2. *We have*
$$\delta_{f,C}(n_1, \ldots, n_r) \leq \delta(f, C)$$
and if $r \geq 1$, this inequality is strict.

Proof. For part (1), observe that this inequality is just Proposition 3.4.5 with the quantity $3\log_3(f(3^{n_1}, \ldots, 3^{n_r})$ subtracted off both sides. And since Proposition 3.4.5 is an inequality of integers, the difference is an integer.

For part (2), let a denote the leading coefficient of f. Then by Proposition 3.4.2,

$$f(3^{n_1}, \ldots, 3^{n_r}) \geq a \cdot 3^{n_1 + \ldots + n_r},$$

and this inequality is strict if $r \geq 1$ (since the constant term of f does not vanish). So

$$\begin{aligned}\delta_{f,C}(n_1, \ldots, n_r) &= C + 3(n_1 + \ldots + n_r) - 3\log_3 f(3^{n_1}, \ldots, 3^{n_r}) \\ &\leq C + 3(n_1 + \ldots + n_r) - 3\log_3(a) - 3(n_1 + \ldots + n_r) \\ &= C - 3\log_3(a) = \delta(f, C),\end{aligned}$$

and this inequality is strict if $r \geq 1$. □

3.4.3 Low-defect polynomials give all leaders of small defect

The reason these polynomials are relevant is as follows:

Theorem 3.4.10. *For any real $r \geq 0$, there exists a finite set $\mathcal{S}_r$ of low-defect pairs satisfying the following conditions:*

1. *Each $(f, C) \in \mathcal{S}_r$ has degree at most $\lfloor r \rfloor$;*

2. For every $N \in B_r$, there exists some $(f, C) \in \mathcal{S}_r$ that efficiently 3-represents N.

Proof. We prove this statement in the following form: For any real $\alpha \in (0, 1)$ and any integer $k \geq 1$, there exists a finite set $\mathcal{S}_{k,\alpha}$ of low-defect pairs, each of degree at most $k-1$, such that for every $N \in B_{k\alpha}$ there exists some $(f, C) \in \mathcal{S}_{\alpha,r}$ that efficiently 3-represents N. Once we have this, the result will follow by taking $\mathcal{S}_r = \mathcal{S}_{k,\alpha}$ for $k = \lfloor r \rfloor + 1$ and $\alpha = \frac{r}{\lfloor r \rfloor + 1}$.

We prove this by induction on k. If $k = 1$, then B_α is finite by Theorem 3.2.7, so we can take $\mathcal{S}_{1,\alpha} = \{(N, \|N\|) : N \in B_\alpha\}$. Now suppose the statement is true for k, and we want to prove it for $k+1$, so we have already constructed sets $\mathcal{S}_{i,\alpha}$ for $i \leq k$.

We will define the set $\mathcal{S}_{k+1,\alpha}$ to consist of the following:

1. If $k + 1 > 2$, then for $(f, C) \in \mathcal{S}_{i,\alpha}$ and $(g, D) \in \mathcal{S}_{j,\alpha}$ with $2 \leq i, j \leq k$ and $i + j = k + 2$ we include $(f \otimes g, C + D)$ in $\mathcal{S}_{k+1,\alpha}$;
   while if $k + 1 = 2$, then for $(f_1, C_1), (f_2, C_2), (f_3, C_3) \in \mathcal{S}_{1,\alpha}$, we include $(f_1 \otimes f_2, C_1 + C_2)$ and $(f_1 \otimes f_2 \otimes f_3, C_1 + C_2 + C_3)$ in $\mathcal{S}_{2,\alpha}$.

2. For $(f, C) \in \mathcal{S}_{k,\alpha}$ and any solid number b with $\|b\| < (k+1)\alpha + 3\log_3 2$, we include $(f \otimes x_1 + b, C + \|b\|)$ in $\mathcal{S}_{k+1,\alpha}$.

3. For $(f, C) \in \mathcal{S}_{k,\alpha}$, any solid number b with $\|b\| < (k+1)\alpha + 3\log_3 2$, and any $v \in B_\alpha$, we include $(v(f \otimes x_1 + b), C + \|b\| + \|v\|)$ in $\mathcal{S}_{k+1,\alpha}$.

4. For all $n \in T_\alpha$, we include $(n, \|n\|)$ in $\mathcal{S}_{k+1,\alpha}$.

5. For all $n \in T_\alpha$ and $v \in B_\alpha$, we include $(vn, \|vn\|)$ in $\mathcal{S}_{k+1,\alpha}$.

This is a finite set, as the $\mathcal{S}_i$ for $i \leq k$ are all finite, B_α is finite, T_α is finite, and there are only finitely many b satisfying $\|b\| < (k+1)\alpha + 3\log_3 2$, as this implies that

$$3\log_3 b < (k+1)\alpha + 3\log_3 2.$$

Also, all elements of $\mathcal{S}_{k+1,\alpha}$ have degree at most k: In case (1), if $k + 1 > 2$, f and g have degree at most $i - 1$ and and $j - 1$ respectively, so $f \otimes g$ has degree at most $i + j - 2 = k$, while if $k + 1 = 2$, then f_1, f_2, and f_3 all have degree 0, so $f_1 \otimes f_2$ and $f_1 \otimes f_2 \otimes f_3$ also have degree 0. In cases (2) and (3), f has degree at most $k - 1$, so $f \otimes x_1 + b$ has degree at most k. Finally, in cases (4) and (5), we are adding low-defect pairs of degree 0.

So suppose that $N \in B_{(k+1)\alpha}$; we apply Theorem 3.2.9.

In case (1) of Theorem 3.2.9, if $k+1 > 2$, then there is a good factorization $N = uv$ where $u \in B_{i\alpha}$, $v \in B_{j\alpha}$ with $i + j = k + 2$ and $2 \leq i, j \leq k$. So by the inductive hypothesis, we can take $(f, C) \in \mathcal{S}_{i,\alpha}$ and $(g, D) \in \mathcal{S}_{j,\alpha}$ such that (f, C) efficiently 3-represents u and (g, D) efficiently 3-represents v. Since the factorization $N = uv$ is good, it follows that $(f \otimes g, C + D)$ efficiently represents N. If $k + 1 = 2$, there is either a good factorization $n = u_1u_2$ or a good factorization $n = u_1u_2u_3$ with all $u_\ell \in B_\alpha$. So take $(f_\ell, C_\ell) \in \mathcal{S}_{1,\alpha}$ such that (f_ℓ, C_ℓ) efficiently 3-represents u_l; then either $(f_1 \otimes f_2, C_1 + C_2)$ or $(f_1 \otimes f_2 \otimes f_3, C_1 + C_2 + C_3)$ efficiently 3-represents N, as appropriate.

In case (2) of Theorem 3.2.9, there are a and b with $N = a + b$, $\|N\| = \|a\| + \|b\|$, $a \in A_{k\alpha}$, $b \leq a$ a solid number, and

$$\delta(a) + \|b\| < (k + 1)\alpha + 3 \log_3 2.$$

In particular, we have $\|b\| < (k + 1)\alpha + 3 \log_3 2$. Write $a = a'3^\ell$ with a' a leader and $\|a\| = \|a'\| + 3\ell$, so $a' \in B_{k\alpha}$, and pick $(f, C) \in \mathcal{S}_{k,\alpha}$ that efficiently 3-represents a'. Then $(f \otimes x_1 + b, C + \|b\|)$ is in $\mathcal{S}_{k+1,\alpha}$ and efficiently 3-represents N. In case (3) of Theorem 3.2.9, there is a good factorization $n = (a + b)v$ with $v \in B_\alpha$ and a and b satisfying the conditions in the case (2) of Theorem 3.2.9, so the proof is similar; if we write $a = a'3^\ell$ with a' a leader and $\|a\| = \|a'\| + 3\ell$ and pick $(f, C) \in \mathcal{S}_{k,\alpha}$ efficiently 3-representing a', then $(v(f \otimes x_1 + b), C + \|b\| + \|v\|)$ efficiently 3-represents N.

Finally, in cases (4) and (5) of Theorem 3.2.9, the pair $(N, \|N\|)$ is itself in $\mathcal{S}_{k+1,\alpha}$, by cases (4) and (5) above. This proves the theorem. □

Note that while this theorem produces a covering of B_r, there is no guarantee that for $f \in \mathcal{S}_r$, all the numbers 3-represented by f will have defect less than r; and in general this will not be the case. For instance, if we use the method of the proof of Theorem 3.4.10 to produce the set $\mathcal{S}_1$, it will contain the polynomial $16x_1 + 1$, which 3-represents the number 17, which has defect greater than 1. This deficiency will be remedied in Chapter 5, where it will be shown how to choose the $\mathcal{S}_r$ to get this additional property. There is also no guarantee that the numbers 3-represented by f will be leaders; for instance, if we use this method to produce the set $\mathcal{S}_1$, it will also contain the constant polynomials 9 and 27.

3.4.4 Augmented low-defect polynomials

Theorem 3.4.10 gives us a representation of the leaders with defect less than a fixed r, but we want to consider all numbers with defect less than r. However, by Proposi-

tion 3.2.6, any number can be written most-efficiently as $3^k m$ for some $k \geq 0$ and some leader m. To account for this, we introduce the notion of an augmented low-defect polynomial:

Definition 3.4.11. For any low-defect polynomial f, we define $\hat{f} = f \otimes x$. The polynomial $\hat{f}$ will be called an *augmented low-defect polynomial*. For a low-defect pair (f, C), the pair $(\hat{f}, C)$ will be called an *augmented low-defect pair*.

Note that augmented low-defect polynomials are never low-defect polynomials; by Proposition 3.4.2, low-defect polynomials always have nonzero constant term, while an augmented low-defect polynomial always has zero constant term.

We can then make the following observations and definitions, parallel to the contents of Subsections 3.4.2 and 3.4.3:

Corollary 3.4.12. *If (f, C) is a low-defect pair of degree r, then*

$$\|\hat{f}(3^{n_1}, \ldots, 3^{n_{r+1}})\| \leq C + 3(n_1 + \ldots + n_{r+1}).$$

Proof. This is immediate from Proposition 3.4.5 and Lemma 3.4.4. □

Definition 3.4.13. Given a low-defect pair (f, C) (say of degree r) and a number N, we will say $(\hat{f}, C)$ efficiently 3-represents N if there exist $n_1, \ldots, n_{r+1}$ such that $N = \hat{f}(3^{n_1}, \ldots, 3^{n_{r+1}})$ and $\|N\| = C + 3(n_1 + \ldots + n_{r+1})$. More generally, we will also say $\hat{f}$ 3-represents N if there exist $n_1, \ldots, n_{r+1}$ such that $N = \hat{f}(3^{n_1}, \ldots, 3^{n_{r+1}})$.

Corollary 3.4.14. *Let (f, C) be a low-defect pair of degree r, and let $n_1, \ldots, n_r$ be nonnegative integers. Then*

$$\delta(\hat{f}(3^{n_1}, \ldots, 3^{n_{r+1}})) \leq \delta_{f,C}(n_1, \ldots, n_r)$$

and the difference is an integer.

Proof. This inequality is just Corollary 3.4.12 with $3 \log_3 \hat{f}(3^{n_1}, \ldots, 3^{n_{r+1}})$ subtracted off both sides. And since Corollary 3.4.12 is an inequality of integers, the difference is an integer. □

Theorem 3.4.15. *For any real $r \geq 0$, there exists a finite set $\mathcal{S}_r$ of low-defect pairs satisfying the following conditions:*

1. *Each $(f, C) \in \mathcal{S}_r$ has degree at most $\lfloor r \rfloor$;*

2. *For every $N \in A_r$, there exists some $(f, C) \in \mathcal{S}_r$ such that $(\hat{f}, C)$ that efficiently 3-represents N.*

Proof. This is immediate from Theorem 3.4.10 and Proposition 3.2.6. □

3.5 Facts from order theory and topology

This section collects facts about well orderings and partial orderings needed to prove the main result. Recall that a *well partial order* is a partial order which is well-founded (has no infinite descending chains) and has no infinite antichains. Any totally-ordered extension of a well partial order is well-ordered. Given a well partial order X, we can consider the set of order types of well-orders obtained by extending the ordering on X. It was proved by D.H.J. De Jongh and R. Parikh [21, Theorem 2.13] that for any well partial order X, the set of ordinals obtained this way has a maximum; this maximum is denoted $o(X)$. They further proved [21, Theorem 3.4, Theorem 3.5]:

Theorem 3.5.1. *Let X and Y be two well partial orders. Then $X \amalg Y$ and $X \times Y$ are well partial orders, and $o(X \amalg Y) = o(X) \oplus o(Y)$, and $o(X \times Y) = o(X) \otimes o(Y)$, where $\oplus$ and $\otimes$ are the operations of natural sum and natural product (also known as the Hessenberg sum and Hessenberg product).*

The natural sum and natural product are defined as follows [21]:

Definition 3.5.2. The *natural sum* (also known as the *Hessenberg sum*) of two ordinals α and β, here denoted $\alpha \oplus \beta$, is defined by simply adding up their Cantor normal forms as if they were "polynomials in ω". That is to say, if there are ordinals $\gamma_0 < \ldots < \gamma_n$ and whole numbers $a_0, \ldots, a_n$ and $b_0, \ldots, b_n$ such that $\alpha = \omega^{\gamma_n} a_n + \ldots + \omega^{\gamma_0} a_0$ and $\beta = \omega^{\gamma_n} b_n + \ldots + \omega^{\gamma_0} b_0$, then

$$\alpha \oplus \beta = \omega^{\gamma_n}(a_n + b_n) + \ldots + \omega^{\gamma_0}(a_0 + b_0).$$

Similarly, the *natural product* (also known as the *Hessenberg product*) of α and β, here denoted $\alpha \otimes \beta$, is defined by multiplying their Cantor normal forms as if they were "polynomials in ω", using the natural sum to add the exponents. That is to say, if we write $\alpha = \omega^{\gamma_n} a_n + \ldots + \omega^{\gamma_0} a_0$ and $\beta = \omega^{\delta_m} b_m + \ldots + \omega^{\delta_0} b_0$ with $\gamma_0 < \ldots < \gamma_0$ and $\delta_0 < \ldots < \delta_m$ ordinals and the a_i and b_i whole numbers, then

$$\alpha \otimes \beta = \bigoplus_{\substack{0 \leq i \leq n \\ 0 \leq j \leq m}} \omega^{\gamma_i \oplus \delta_j} a_i b_j.$$

These operations are commutative and associative, and $\otimes$ distributes over $\oplus$. The expression $\alpha \oplus \beta$ is strictly increasing in α and β; and $\alpha \otimes \beta$ is strictly increasing in β so long as $\alpha \neq 0$, and vice versa [17].

There are other definitions of these operations. Given ordinals α and β, $\alpha \oplus \beta$ is sometimes defined as $o(\alpha \amalg \beta)$, and $\alpha \otimes \beta$ as $o(\alpha \times \beta)$, where for this definition we consider α and β as partial orders). As noted above, De Jongh and Parikh showed the stronger statement Theorem 3.5.1, from which it follows that

$$\begin{aligned} o(\alpha_1 \amalg \ldots \amalg \alpha_n) &= \alpha_1 \oplus \ldots \oplus \alpha_n \\ o(\alpha_1 \times \ldots \times \alpha_n) &= \alpha_1 \otimes \ldots \otimes \alpha_n \end{aligned}$$

There is also a recursive definition [19].

Note also the following statements about well partial orderings:

Proposition 3.5.3. *Suppose that X is a well partially ordered set, S a totally ordered set, and $f : X \to S$ is monotonic. Then $f(X)$ is well-ordered, and has order type at most $o(X)$.*

Proof. Pick a well-ordering extending the ordering $\leq$ on X; call it $\preceq$. Define another total ordering on X, call it $\leq'$, by $a <' b$ if either $f(a) < f(b)$ or $f(a) = f(b)$ and $a \prec b$. Observe that $\leq'$ is an extension of $\leq$ as f is monotonic, so it is a well-ordering and has order type at most $o(X)$. Since f is clearly also monotonic when we instead use the ordering $\leq'$ on the domain, its image is therefore also well-ordered and of order type at most $o(X)$. □

Note in particular that if X is the union of $X_1, \ldots, X_n$, then $o(X) \leq o(X_1) \oplus \ldots \oplus o(X_n)$ as X is a monotonic image of $X_1 \amalg \ldots \amalg X_n$. So we have:

Proposition 3.5.4. *We have:*

1. *If S is a well-ordered set and $S = S_1 \cup \ldots \cup S_n$, and S_1 through S_n all have order type less than ω^k, then so does S.*

2. *If S is a well-ordered set of order type ω^k and $S = S_1 \cup \ldots \cup S_n$, then at least one of S_1 through S_n also has order type ω^k.*

Proof. For (1), observe that the order type of S is at most the natural sum of those of $S_1, \ldots, S_n$, and the natural sum of ordinals less than ω^k is again less than ω^k.

For (2), by (1), if $S_1, \ldots, S_k$ all had order type less than ω^k, so would S; so at least one has order type at least ω^k, and it necessarily also has order type at most ω^k, being a subset of S. □

For the proof of the main result we will also need some facts about well-ordered sets sitting inside the real numbers. In particular, we need results about closures and limit points of such sets, with the ambient space carrying the order topology. Since we have not found all the following results in the literature, we supply proofs.

Proposition 3.5.5. *Let X be a totally ordered set, and let S be a well-ordered subset of order type α. Then $\overline{S}$ is also well-ordered, and has order type either α or $\alpha + 1$. If $\alpha = \gamma + k$ where γ is a limit ordinal and k is finite, then $\overline{S}$ has order type $\alpha + 1$ if and only if the initial segment of S of order type γ has a supremum in X which is not in S.*

Proof. We induct on α. If $\alpha = 0$, S is empty and thus so is $\overline{S}$.

If $\alpha = \beta + 1$, say x is the maximum element of S and $T = S \setminus \{x\}$. Then $\overline{S} = \overline{T} \cup \{x\}$, and x is the maximum element of $\overline{S}$. If $x \in \overline{T}$, then $\overline{S} = \overline{T}$; otherwise its order type is 1 greater. So as $\overline{T}$ has order type either β or $\beta + 1$ by the inductive hypothesis, $\overline{S}$ has order type β, $\beta + 1 = \alpha$, or $\beta + 2 = \alpha + 1$. Of course, the first of these is impossible, as its order type must be at least α, since it contains S, so the order type is either α or $\alpha + 1$.

Furthermore, if $\beta = \gamma + k$ where γ is a limit ordinal, we can let R be the initial segment of T (equivalently, of S) of order type γ. Then by the inductive hypothesis, $\overline{T}$ has order type $\beta + 1$ if and only if R has a supremum in X which is not in T. In the case where $x \notin \overline{T}$, then $x \notin \overline{R}$ and so x cannot be a supremum of R in X. Hence, in this case, $\overline{T}$ has order type $\beta + 1$ if and only if R has a supremum in X which is not in S, and so $\overline{S}$ has order type $\beta + 2 = \alpha + 1$ if and only if R has a supremum in X which is not in S.

In the case where $x \in \overline{T}$, it must be that x is a supremum of T in X. Since x is not itself in T, this requires that β be a limit ordinal, and hence that $\beta = \gamma$, i.e. $T = R$, since γ is the largest limit ordinal smaller than S. So R has a supremum which is not in T, namely, x; and so by the inductive hypothesis $\overline{T}$ has order type $\beta + 1$. As $\overline{S} = \overline{T}$ in this case, it too has order type $\beta + 1 = \alpha$. Furthermore, R has a supremum, x, but this supremum is in S; thus the theorem is true in this case.

Finally we have the case where α is a limit ordinal. If $x \in \overline{S}$, either x is an upper bound of S or it is not; we will first consider R, the subset of $\overline{S}$ consisting of those elements which are not upper bounds of S. For any $x \in R$, there is some $y \in S$ with $y > x$, and so $x \in (-\infty, y) \cap \overline{S}$. Since the former is an open set, this means $x \in \overline{S \cap (-\infty, y)}$. As $S \cap (-\infty, y)$ is a proper initial segment of S, by the inductive hypothesis, its closure is well-ordered. Note that for varying y, the sets $\overline{S \cap (-\infty, y)}$

form a chain under inclusion of well-ordered sets, with smaller ones being initial segments of larger ones. So as R is the union of these, it is well-ordered, and its order type is equal to their supremum. Now clearly the order type of R is at least α, since R includes S; and by the inductive hypothesis, it is at most $\lim_{\beta<\alpha}(\beta+1) = \alpha$. So R has order type α.

This leaves the question of elements of $\overline{S}$ that are upper bounds of S (and hence R). The only way such an element can exist is if it is the supremum of S. Hence, if S has a supremum in X, and this supremum is not already in S, then $\overline{S}$ has order type $\alpha+1$, and otherwise it has order type α. □

Proposition 3.5.6. *Suppose X is a totally ordered set, S a subset of X, and T an initial segment of S. Then $\overline{T}$ is an intial segment of $\overline{S}$.*

Proof. Suppose $x \in \overline{T}$, $y \in \overline{S}$, and $y < x$; we want to show $y \in \overline{T}$. The set (y, ∞) is an open subset of X and contains $x \in \overline{T}$, thus it also contains some $t \in \overline{T}$. That is to say, there is some $t \in T$ with $t > y$.

Now say U is any open neighborhood of y; then $U \cap (-\infty, t)$ is again an open neighborhood of y, and since $y \in \overline{S}$, there must exist some $s \in S \cap U \cap (-\infty, t)$. But then $s \in S$, $s < t$, and $t \in T$, so $s \in T$ as well as we assumed that T was an initial segment of S. Thus each neighborhood U of y contains some element of T, that is to say, $y \in \overline{T}$. □

Corollary 3.5.7. *Let X be a totally ordered set with the least upper bound property, and S a well-ordered subset of X of order type α. Then if $\beta < \alpha$ is a limit ordinal, the β'th element of $\overline{S}$ is the supremum (limit) of the initial β elements of S.*

Proof. Let T be the intial segment of S of order type β. Since $\beta < \alpha$, T is bounded above in S, and thus in X, and thus it has a supremum s. This supremum s is not in T as T has order type β, a limit ordinal, and thus has no maximum. So $\overline{T}$, by Proposition 3.5.5, has order type $\beta + 1$, and s is clearly its final element. So by Proposition 3.5.6, it is the β'th element of $\overline{S}$ as well, and by definition it is the supremum of the initial β elements of S. □

Proposition 3.5.8. *If S is a well-ordered set of order type $\alpha < \omega^{n+1}$ with n finite, then S', the set of limit points of S (in the order topology) has order type strictly less than ω^n.*

Proof. Since we are considering S purely as a totally-ordered set and not embedded in anything else, we may assume it is an ordinal. Let β be the order type of S'. The

elements of S' consist of the limit ordinals less than α. If $n = 0$, then α is finite and so $\beta = 0 < \omega^0$.

Otherwise, $\alpha < \omega^{n+1}$ so say $\alpha \leq \omega^n k$. An ordinal γ is a limit ordinal if and only if it can be written as $\omega\gamma'$ for some $\gamma' > 0$. Since, assuming $n > 0$, $\omega\gamma' < \omega^n k$ if and only if $\gamma' < \omega^{n-1}k$, the order type of the set of limit ordinals less than $\omega^n k$ is easily seen to be $\omega^{n-1}k - 1$ (where the 1 is subtracted off the beginning; this only makes a difference if $n = 1$). So the order type of β is at most $\omega^{n-1}k - 1 < \omega^n$. □

It is not too hard to write down a general formula for the order type of S' in terms of the order type of S (even without the restriction that $\alpha < \omega^\omega$), but we will not need such detail here. See [43, Theorem 8.6.6] for more on this.

Proposition 3.5.9. *Let T be a totally-ordered set and S a well-ordered subset. If S' (in the order topology on T) has order type at least ω^n with n finite, then S has order type at least ω^{n+1}.*

Proof. Suppose S has order type less than ω^{n+1}. Then by Proposition 3.5.5, so does $\overline{S}$. Since $\overline{S}' = S'$, we can just consider $\overline{S}$. And we can consider the order topology on $\overline{S}$ instead of the subspace topology, since the former is coarser and thus $\overline{S}$ has more limit points under it. But by Proposition 3.5.8, the order type of $\overline{S}'$ in the order topology on $\overline{S}$ is less than ω^n. Hence $\overline{S}'$ under the subspace topology also has order type less than ω^n, and hence S' has order type less than ω^n. So if S' has order type at least ω^n, then S has order type at least ω^{n+1}. □

3.6 Well-ordering of defects

We now begin proving well-ordering theorems about defects.

Proposition 3.6.1. *Let (f, C) be a low-defect pair; then the function $\delta_{f,C}$ is strictly increasing in each variable.*

Proof. Suppose f has degree r. We can define g, the reverse polynomial of f:

$$g(x_1, \ldots, x_r) = x_1 \ldots x_r f(x_1^{-1}, \ldots, x_r^{-1}).$$

So g is a multilinear polynomial in $x_1, \ldots, x_r$, with the coefficient of $\prod_{i \in S} x_i$ in g being the coefficient of $\prod_{i \notin S} x_i$ in f. By Proposition 3.4.2, f has nonnegative coefficients, so so does g; since the constant term of f does not vanish, the $x_1 \ldots x_r$ term of g does not vanish. Hence g is strictly increasing in each variable.

Then

$$\begin{aligned}\delta_{f,C}(n_1,\ldots,n_r) &= C + 3(n_1+\ldots+n_r) - 3\log_3 f(3^{n_1},\ldots,3^{n_r}) \\ &= C - 3\log_3 \frac{f(3^{n_1},\ldots,3^{n_r})}{3^{n_1+\ldots+n_r}} = C - 3\log_3 g(3^{-n_1},\ldots,3^{-n_r})\end{aligned}$$

which is strictly increasing in each variable, as claimed. □

Proposition 3.6.2. *Let (f, C) be a low-defect pair of degree r; then the image of $\delta_{f,C}$ is a well-ordered subset of $\mathbb{R}$, with order type ω^r.*

Proof. By Proposition 3.6.1, $\delta_{f,C}$ is a monotonic function from $\mathbb{Z}^r_{\geq 0}$ to $\mathbb{R}$, and $\mathbb{R}$ is totally ordered, so by Proposition 3.5.3 and Theorem 3.5.1 its image is a well-ordered set of order type at most ω^r.

For the lower bound, we induct on r. Let S denote the image of $\delta_{f,C}$. If $r = 0$, $\delta_{f,C}$ is a constant and so S has order type $1 = \omega^0$. Now suppose $r \geq 1$ and that this is true for $r-1$. By Lemma 3.4.3, we can write $f = h \otimes (g \otimes x_1 + c)$ where c is a positive integer and g and h are low-defect polynomials. Unpacking this statement, if s is the degree of h, we have $f(x_1,\ldots,x_r) = h(x_1,\ldots,x_s)(g(x_{s+1},\ldots,x_{r-1})x_r + c)$. Then

$$\begin{aligned}\delta_{f,C}(n_1,\ldots,n_r) &= (C - \|h\|) + \delta_h(n_1,\ldots,n_s) + \\ &\quad 3(n_{s+1}+\ldots+n_{r-1}) - 3\log_3(g(3^{n_{s+1}},\ldots,3^{n_{r-1}}) + c3^{-n_r}).\end{aligned}$$

Thus,

$$\begin{aligned}\lim_{n_r\to\infty} \delta_{f,C}(n_1,\ldots,n_r) &= C - \|h\| + \delta_h(n_1,\ldots,n_s) + \\ &\quad 3(n_{s+1}+\ldots+n_{r-1}) - 3\log_3(g(3^{n_{s+1}},\ldots,3^{n_{r-1}})) \\ &= C - \|h\| - \|g\| + \delta_h(n_1,\ldots,n_s) + \delta_g(n_{s+1},\ldots,n_{r-1}) \\ &= C - 3\log_3(h(n_1,\ldots,n_s)g(n_{s+1},\ldots,n_{r-1})) \\ &= C - \|g\otimes h\| + \delta_{g\otimes h}(n_1,\ldots,n_{r-1}).\end{aligned}$$

And since $\delta_{f,C}$ is increasing in n_r, this means that this is in fact a limit point of S. So we see that S' contains a translate of the image of $\delta_{g\otimes h}$. The degree of $g \otimes h$ is $r-1$, so by the inductive hypothesis, this image has order type at least ω^{r-1}. Thus S' has order type at least ω^{r-1}, and so by Proposition 3.5.9, this means that S has order type at least ω^r. □

Proposition 3.6.3. *Let (f, C) be a low-defect pair of degree r; then the set of $\delta(n)$ for all n 3-represented by the augmented low-defect polynomial $\hat{f}$ is a well-ordered subset of $\mathbb{R}$, with order type at least ω^r and at most $\omega^r(\lfloor\delta(f, C)\rfloor + 1) < \omega^{r+1}$. The same is true if f is used instead of the augmented version $\hat{f}$.*

Proof. Let S be the set of all $\delta(n)$ for all n that are 3-represented by $\hat{f}$, and let T be the image of $\delta_{f,C}$. By Proposition 3.6.2, T is a well-ordered subset of $\mathbb{R}$, of order type ω^r. Suppose $n = \hat{f}(3^{m_1}, \ldots, 3^{m_{r+1}})$. Then by Corollary 3.4.14,

$$\delta(n) = \delta_{f,C}(m_1, \ldots, m_{r+1}) - k$$

for some $k \geq 0$. But $\delta_{f,C}(m_1, \ldots, m_{r+1}) \leq \delta(f, C)$ by Proposition 3.4.9, and since $\delta(n) \geq 0$, this implies $k \leq \delta(f, C)$. As k is an integer, this implies

$$k \in \{0, \ldots, \lfloor\delta(f, C)\rfloor\},$$

which is a finite set. Let ℓ refer to the number $\lfloor\delta(f, C)\rfloor$.

Thus, S is covered by finitely many translates of T; more specifically, we can partition T into T_0 through T_ℓ such that

$$S = T_0 \cup (T_1 - 1) \cup \ldots \cup (T_\ell - \ell).$$

Then the T_i all have order type at most ω^r, and by Proposition 3.5.4 at least one has order type ω^r. Hence S is well-ordered of order type at most $\omega^r(\lfloor\delta(f, C)\rfloor + 1) < \omega^{r+1}$ by Propositions 3.5.1 and 3.5.3. And by the above reasoning, it also has order type at least ω^r.

The proof for f instead of $\hat{f}$ is similar. □

Proposition 3.6.4. *For any $s > 0$, the set $\mathscr{D} \cap [0, s)$ is a well-ordered subset of $\mathbb{R}$ with order type at least $\omega^{\lfloor s\rfloor}$ and less than $\omega^{\lfloor s\rfloor+1}$.*

Proof. By Theorem 3.4.15, there exists a finite set $\mathcal{S}_s$ of low-defect polynomials of degree at most $\lfloor s\rfloor$ such that each $n \in A_s$ can be 3-represented by $\hat{f}$ for some $f \in \mathcal{S}_s$. By Proposition 3.6.3, for each $f \in \mathcal{S}$, the set of defects of numbers 3-represented by $\hat{f}$ is a well-ordered set of order type less than $\omega^{\lfloor s\rfloor+1}$. Since $\mathscr{D} \cap [0, s)$ is covered by a finite union of these, it is also well-ordered of order type less than $\omega^{\lfloor s\rfloor+1}$ by Proposition 3.5.4.

For the lower bound on the order type, if $0 < s < 1$, observe that $0 \in \mathscr{D} \cap [0, s)$.

Otherwise, let $k = \lfloor s \rfloor$ and consider the low-defect polynomial

$$f = (\ldots(((3x_1 + 1)x_2 + 1)x_3 + 1)\ldots)x_k + 1.$$

We have $\|f\| \leq 3 + k$, so $\delta(f) \leq k \leq s$. And since $k \geq 1$, by Propostion 3.4.9 the set of $\delta(n)$ for n that are 3-represented by f is contained in $\mathscr{D} \cap [0, s)$; while by Proposition 3.6.3, it has order type at least ω^k, proving the claim. □

We can thus conclude:

Theorem 3.6.5. *The set $\mathscr{D}$ is a well-ordered subset of $\mathbb{R}$, of order type ω^ω.*

Proof. By Proposition 3.6.4, we see that each initial segment of $\mathscr{D}$ is well-ordered, and with order type less than ω^ω; hence $\mathscr{D}$ is well-ordered, and has order type at most ω^ω. Also by Proposition 3.6.4, we can find initial segments of $\mathscr{D}$ with order type at least ω^n for any $n \in \mathbb{N}$, so $\mathscr{D}$ has order type at least ω^ω. □

We have now determined the order type of $\mathscr{D}$. However, we have not fully determined the order types of $\mathscr{D} \cap [0, s]$ for real numbers s. Of course in general determining this is complicated, but we can answer the question when s is an integer:

Theorem 3.6.6. *For any whole number $k \neq 1$, $\mathscr{D} \cap [0, k]$ is a well-ordered subset of $\mathbb{R}$ with order type ω^k, while $\mathscr{D} \cap [0, 1]$ has order type $\omega + 1$.*

Proof. The order type of $\mathscr{D} \cap [0, k]$ is either the same as that of $\mathscr{D} \cap [0, k)$, or that same order type plus 1, depending on whether or not $k \in \mathscr{D}$. By Theorem 3.2.1, the only integral elements of $\mathscr{D}$ are 0 and 1, so what remains is to determine the order type of $\mathscr{D} \cap [0, k)$. For $k = 0$ this is clearly $1 = \omega^0$, making the statement true for $k = 0$, so assume $k \geq 1$.

By Proposition 3.6.4, $\mathscr{D} \cap [0, k)$ is well-ordered and has order type at least ω^k. However its order type is also equal to the supremum of the order types of $\mathscr{D} \cap [0, r)$ for $r < k$, and by Proposition 3.6.4, since k is an integer, these are all less than ω^k. Hence its order type is also at most ω^k, and thus exactly ω^k. Thus for $k \geq 1$, the order type of $\mathscr{D} \cap [0, k]$ is exactly ω^k, unless $k = 1$, in which case it is $\omega + 1$. □

Putting these together, we have the main theorem:

Proof of Theorem 3.1.2. The first part is Theorem 3.6.5. The second part follows from the proof of Theorem 3.6.6, or from Theorem 3.6.6 and the fact that 1 is the only nonzero defect which is also an integer. □

We will further discuss the order type of $\mathscr{D} \cap [0, s]$ when s is not an integer in a future paper [4].

3.7 Variants of the main theorem

In this section, we prove several variants of the main theorem, all showing ω^ω well-ordering for various related sets.

We begin with proving the well ordering holds for the closure $\overline{\mathscr{D}}$ of the defect set in $\mathbb{R}$.

Proposition 3.7.1. *The set $\overline{\mathscr{D}}$, the closure of the defect set, is well-ordered, with order type ω^ω. Furthermore, for an integer $k \geq 1$, the order type of $\overline{\mathscr{D}} \cap [0, k]$ is $\omega^k + 1$. (And $k \in \overline{\mathscr{D}}$, so k is the ω^k'th element of $\overline{\mathscr{D}}$).*

Proof. By Proposition 3.5.5, the set $\overline{\mathscr{D}}$ is well-ordered, and its order type is ω^ω since $\mathscr{D}$ is unbounded in $\mathbb{R}$. For the set $\overline{\mathscr{D}} \cap [0, k]$, observe that this set is is the same as the closure of $\mathscr{D} \cap [0, k]$ within $[0, k]$, so Proposition 3.5.5 implies this has order type $\omega^k + 1$ since $[0, k]$ has the least-upper-bound property. And since by Proposition 3.5.5, for $r < k$ the set $\overline{\mathscr{D}} \cap [0, r]$ has order type less than ω^k, the ω^k'th element must be k itself. □

The other variants of the main result include considering defect sets for integers n whose complexity $\|n\|$ falls in individual congruence classes modulo 3 and, in a separate direction, restricting to stable defects. Furthermore results in both directions can be combined. These defect sets are all well-ordered by virtue of being contained in $\overline{\mathscr{D}}$, and the issue is to show they have the appropriate order type.

To prove the main theorem, we needed to know that given a low-defect pair (f, C) of degree k, we have $\|f(3^{n_1}, \ldots, 3^{n_k})\| \leq C + 3(n_1 + \ldots + n_k)$. In order to prove these more detailed versions, as a preliminary result we demonstrate that for certain low-defect pairs (f, C), equality holds for "most" choices of $(n_1, \ldots, n_k)$. Indeed, we'll need an even stronger statement: Since $\|f(3^{n_1}, \ldots, 3^{n_k})\| \leq C + 3(n_1 + \ldots + n_k)$, it follows that also

$$\|f(3^{n_1}, \ldots, 3^{n_k})\|_{st} \leq C + 3(n_1 + \ldots + n_k),$$

and it's equality in this form that we'll need for "most" $(n_1, \ldots, n_k)$.

Proposition 3.7.2. *Let (f, C) be a low-defect pair of degree k with $\delta(f, C) < k + 1$. Define its "exceptional set" to be*

$$S := \{(n_1, \ldots, n_k) : \|f(3^{n_1}, \ldots, 3^{n_k})\|_{st} < C + 3(n_1 + \ldots + n_k)\}$$

Then the set $\{\delta(f(3^{n_1},\ldots,3^{n_k})) : (n_1,\ldots,n_k) \in S\}$ *has order type less than* ω^k*. In particular, the set* $\{\delta(f(3^{n_1},\ldots,3^{n_k})) : (n_1,\ldots,n_k) \notin S\}$ *has order type at least* ω^k*, and thus so does the set*

$$\{\delta(f(3^{n_1},\ldots,3^{n_k})) : (n_1,\ldots,n_k) \in \mathbb{Z}^k_{\geq 0}\} \cap \mathscr{D}^C_{st}.$$

Proof. The set S can be equivalently written as

$$\{(n_1,\ldots,n_k) : \|f(3^{n_1},\ldots,3^{n_k})\|_{st} \leq C + 3(n_1+\ldots+n_k) - 1\}$$

and hence as

$$\{(n_1,\ldots,n_k) : \delta_{st}(f(3^{n_1},\ldots,3^{n_k})) \leq \delta_{f,C}(n_1,\ldots,n_k) - 1\}.$$

Hence for $(n_1,\ldots,n_k) \in S$, we have

$$\delta_{st}(f(3^{n_1},\ldots,3^{n_k})) \leq \delta(f,C) - 1 < k,$$

and thus by Proposition 3.6.4, the set of these stable defects has order type less than ω^k.

Equivalently, applying Proposition 3.5.4, the set

$$\{\delta(f(3^{n_1},\ldots,3^{n_k})) : (n_1,\ldots,n_k) \in S\}$$

and the set $\delta_{f,C}(S)$ have order type less than ω^k, since each is a finite union of translates of subsets of the set $\{\delta(f(3^{n_1},\ldots,3^{n_k})) : (n_1,\ldots,n_k) \in S\}$.

So consider the set

$$\{\delta(f(3^{n_1},\ldots,3^{n_k})) : (n_1,\ldots,n_k) \notin S\},$$

which can equivalently be written as

$$\{\delta_{st}(f(3^{n_1},\ldots,3^{n_k})) : (n_1,\ldots,n_k) \notin S\},$$

since for $(n_1,\ldots,n_k) \notin S$, the number $f(3^{n_1},\ldots,3^{n_k})$ is stable. This set must have order type at least ω^k by Proposition 3.6.3 and Proposition 3.5.4. Since for $(n_1,\ldots,n_k) \notin S$, we have that $f(3^{n_1},\ldots,3^{n_k})$ is stable and

$$\|f(3^{n_1},\ldots,3^{n_k})\| = C + 3(n_1+\ldots+n_k) \equiv C \pmod 3,$$

this implies that the set

$$\{\delta(f(3^{n_1},\ldots,3^{n_k})) : (n_1,\ldots,n_k) \in \mathbb{Z}^k_{\geq 0}\} \cap \mathscr{D}^C_{st},$$

being a superset of the above, has order type at least ω^k. □

Recall that $\mathscr{D}^a_{st}$ denotes the set of defect values $\delta(n)$ taken by stable numbers n having complexity $\|n\| \equiv a \pmod 3$. Using the Proposition above, we can now prove:

Theorem 3.7.3. *For $a = 0, 1, 2$, the stable defect sets $\mathscr{D}^a_{st}$ are well-ordered, with order type ω^ω. Furthermore, if $k \equiv a \pmod 3$, then the set $\mathscr{D}^a_{st} \cap [0, k]$ has order type ω^k.*

Proof. Each of these sets is a subset of $\mathscr{D}$ and so they are well-ordered with order type at most ω^ω. To check that it is in fact exactly ω^ω, consider the following low-defect polynomial:

$$f_{a,k} := (\ldots(((ax_1 + 1)x_2 + 1)x_3 + 1)\ldots)x_k + 1.$$

Specifically, consider the low-defect pair $(f_{a,k}, \|a\| + k)$, for $a = 2, 3, 4$. Observe that $\delta(f_{a,k}, \|a\| + k) = \delta(a) + k$, and for these choices of a, we have $\delta(a) < 1$. Thus for $a = 2, 3, 4$, $f_{a,k}$ satisfies the conditions of Proposition 3.7.2. Thus for $a = 2, 3, 4$ and $k \geq 0$, $\mathscr{D}^{a+k}_{st}$ has order type at least ω^k. Since regardless of k, the set $\{2+k, 3+k, 4+k\}$ is a complete system of residues modulo 3, it follows that for $a = 0, 1, 2$ and any k, the set $\mathscr{D}^a_{st}$ has order type at least ω^k. Hence $\mathscr{D}^a_{st}$ has order type at least ω^ω and hence exactly ω^ω.

Now suppose we take $k \equiv a \pmod 3$. We know, if $k \neq 1$, that $\mathscr{D}^a_{st} \cap [0, k]$ has order type at most ω^k by Theorem 3.6.6. (If $k = 1$, we know this because $1 \notin \mathscr{D}_{st}$.) To see that it is at least ω^k, we consider the low-defect pair $(f_{3,k}, 3+k)$. Observe that $\delta(f_{3,k}, 3 + k) = k$, and so (by Proposition 3.7.2) the set $\mathscr{D}^{3+k}_{st} \cap [0, k]$ has order type at least ω^k. Since $3 + k \equiv a \pmod 3$, this is the same as the set $\mathscr{D}^a_{st} \cap [0, k]$, proving the claim. □

With this result in hand, we can now prove:

Theorem 3.7.4. *We have:*

1. *The defect set $\mathscr{D}$ and stable defect set $\mathscr{D}_{st}$ are both well-ordered, both with order type ω^ω. Furthermore, the set $\mathscr{D}_{st} \cap [0, k]$ has order type ω^k, and for $k \neq 1$, so does $\mathscr{D} \cap [0, k]$.*

2. *The sets $\overline{\mathscr{D}_{st}}$ and $\overline{\mathscr{D}}$ are well-ordered, both with order type ω^ω. Furthermore, for $k \geq 1$, the sets $\overline{\mathscr{D}_{st}} \cap [0, k]$ and $\overline{\mathscr{D}_{st}} \cap [0, k]$ have order type $\omega^k + 1$ (and both contain k, so k is the ω^k'th element of both).*

3. *For $a = 0, 1, 2$, the sets $\mathscr{D}^a$ and $\mathscr{D}^a_{st}$ are all well-ordered, each with order type ω^ω. Furthermore, if $a \equiv k \pmod 3$, then $\mathscr{D}^a \cap [0, k]$ and $\mathscr{D}^a_{st} \cap [0, k]$ have order type ω^k*

4. *For $a = 0, 1, 2$, the sets $\overline{\mathscr{D}^a}$ and $\overline{\mathscr{D}^a_{st}}$ are well-ordered with order type ω^ω. Furthermore, if $k \geq 1$ and $a \equiv k \pmod 3$, then $\overline{\mathscr{D}^a} \cap [0, k]$ and $\overline{\mathscr{D}^a_{st}} \cap [0, k]$ have order type $\omega^k + 1$ (and each contains k, so k is the ω^k'th element).*

Proof. The part of (1) for $\mathscr{D}$ is just Theorem 3.6.6. To prove the rest, observe that the order type of $\mathscr{D}_{st}$ is ω^ω because it is contained in $\mathscr{D}$ and contains, e.g., $\mathscr{D}^0_{st}$. For $k \neq 1$, we can see that the order type of $\mathscr{D}_{st} \cap [0, k]$ is at most ω^k because it is contained in $\mathscr{D} \cap [0, k]$. For $k = 1$, we need to additionally note that $1 \notin \mathscr{D}_{st}$. Finally, the order type of $\mathscr{D}_{st} \cap [0, k]$ is at least ω^k because it contains $\mathscr{D}^k_{st} \cap [0, k]$.

The part of (2) for $\overline{\mathscr{D}}$ is Proposition 3.7.1. To prove the rest, note that by (1), $\mathscr{D}_{st}$ is unbounded in $\mathbb{R}$, and so Proposition 3.5.5 implies that $\overline{\mathscr{D}_{st}}$ is well-ordered with order type ω^ω. For $\overline{\mathscr{D}_{st}} \cap [0, k]$, (1) together with Proposition 3.5.5 implies this has order $\omega^k + 1$. And since by Proposition 3.5.5, for $r < k$ the set $\overline{\mathscr{D}_{st}} \cap [0, r]$ has order type less than ω^k, the ω^k'th element must be k itself.

The part of (3) for $\overline{\mathscr{D}^a}$ is just Theorem 3.7.3. To prove the rest, observe that the sets $\mathscr{D}^a$ are well-ordered with order type ω^ω because they contain $\mathscr{D}^a_{st}$ and are contained in $\mathscr{D}$. Furthermore, if $a \equiv k \pmod 3$, then $\mathscr{D}^a \cap [0, k]$ has order type at least ω^k by Theorem 3.7.3. If $k \neq 1$, then Theorem 3.6.6 shows it has order type at most ω^k; for $k = 1$, we need to additionally note that $1 \notin \mathscr{D}^a$.

Finally, to prove (4), note that by Theorem 3.7.3 and (3), $\mathscr{D}^a$ and $\mathscr{D}^a_{st}$ are unbounded in $\mathbb{R}$, and so Proposition 3.5.5 implies $\overline{\mathscr{D}^a_{st}}$ and $\overline{\mathscr{D}^a}$ are well-ordered with order type ω^ω. For $\overline{\mathscr{D}^a_{st}} \cap [0, k]$ and $\overline{\mathscr{D}^a} \cap [0, k]$, Theorem 3.7.3 and (3) together with Proposition 3.5.5 imply these have order type $\omega^k + 1$. And since by Proposition 3.5.5, for $r < k$ the sets $\overline{\mathscr{D}^a_{st}} \cap [0, r]$ and $\overline{\mathscr{D}^a} \cap [0, r]$ has order type less than ω^k, the ω^k'th element must be k itself. □

We can also re state this result in the following way:

Corollary 3.7.5. *We have:*

1. *For $k \geq 1$, the ω^k'th elements of $\overline{\mathscr{D}}$ and $\overline{\mathscr{D}_{st}}$ are both k. If $a \equiv k \pmod 3$, this is also true of $\overline{\mathscr{D}^a}$ and $\overline{\mathscr{D}^a_{st}}$.*

2. *For $k \geq 0$, the supremum of the initial ω^k elements of $\mathscr{D}$ is k, and so is that of the initial ω^k elements of $\mathscr{D}_{st}$. If $a \equiv k \pmod 3$, then this is also true of $\mathscr{D}^a$ and $\mathscr{D}^a_{st}$.*

Proof. Part (1) is just Theorem 3.7.4. Part (2), for $k \geq 1$, is Theorem 3.7.4 and Corollary 3.5.7. For $k = 0$, this is just the observation that 0 is the intial element of $\mathscr{D}$ and so also of $\mathscr{D}_{st}$, $\mathscr{D}^0$, and $\mathscr{D}^0_{st}$ (since these all contain 0). □

So we have now exhibited sixteen particular sets of defects that are well-ordered with order type ω^ω: $\mathscr{D}$, $\mathscr{D}_{st}$, the closures of these sets, and for $a = 0, 1, 2$, the sets $\mathscr{D}^a$, $\mathscr{D}^a_{st}$, and their closures. We leave it for future work to resolve which of these sets are distinct.

Chapter 4

Addition Chains and Well-Ordering

Abstract: An *addition chain* for n is defined to be a sequence $(a_0, a_1, \ldots, a_r)$ such that $a_0 = 1$, and, for any k with $1 \leq k \leq r$, there exist $0 \leq i, j < k$ such that $a_k = a_i + a_j$; the number r is called the length of the addition chain. The shortest length among addition chains for n, called the addition chain length of n, is denoted $\ell(n)$. The number $\ell(n)$ is always at least $\log_2 n$; in this chapter we consider the difference $\delta^\ell(n) := \ell(n) - \log_2 n$, which we call the addition chain defect. First we use this notion to show that for any n, there exists K such that for any $k \geq K$, we have $\ell(2^k n) = \ell(2^K n) + (k - K)$. The main result is that the set of values of δ^ℓ is a well-ordered subset of $[0, \infty)$, with order type ω^ω. The results obtained here are analogous to the results for integer complexity obtained in Chapters 2 and 3. We also prove similar well-ordering results for restricted forms of addition chain length, such as star chain length and Hansen chain length.

4.1 Introduction

An *addition chain* for n is defined to be a sequence $(a_0, a_1, \ldots, a_r)$ such that $a_0 = 1$, and, for any k with $1 \leq k \leq r$, there exist $0 \leq i, j < k$ such that $a_k = a_i + a_j$; the number r is called the length of the addition chain. The shortest length among addition chains for n, called the *addition chain length* of n, is denoted $\ell(n)$. Addition chains were introduced in 1894 by H. Dellac [22] and reintroduced in 1937 by A. Scholz [41], who raised a series of questions about them. They have been much studied in the context of computation of powers, since an addition chain for n of length r allows one to compute x^n from x using r multiplications. Extensive surveys on the topic can be found in Knuth [35, Section 4.6.3] and Subbarao [46].

Addition chain length is approximately logarithmic; it satisfies the bounds

$$\log_2 n \leq \ell(n) \leq \lfloor \log_2 n \rfloor + \nu_2(n) - 1,$$

in which $\nu_2(n)$ counts the number of 1's in the binary expansion of n. The lower bound follows from the observation that the largest number that can be made with an addition chain of k steps is 2^k, since each step can at most double the previous number. The upper bound follows from writing n using the "binary method", which can be defined recursively: The binary chain for $2n$ is the binary chain for n followed by $2n$, and the binary chain for $2n+1$ is the binary chain for $2n$ followed by $2n+1$; this chain has length $\lfloor \log_2 n \rfloor + \nu_2(n) - 1$. In fact, A. Brauer [15] proved in 1939 that $\ell(n) \sim \log_2 n$.

The addition chain complexity function $\ell(n)$ seems complicated and hard to compute. An outstanding open problem about it is the *Scholz-Brauer conjecture* ([41, Question 3]), which asserts that

$$\ell(2^n - 1) \leq n + \ell(n) - 1.$$

To investigate it Brauer [15] introduced a restricted type of addition chain called a *star chain*, and later authors introduced other restricted types of addition chains, such as *Hansen chains*, discussed in Section 4.1.3. Later Knuth [35] introduced the quantity $s(n) := \ell(n) - \lfloor \log_2 n \rfloor$, which he called the number of *small steps* of n. This notion was subsequently used by other authors ([27, 47, 50]) investigating the general behavior of $\ell(n)$ and the Scholz-Brauer conjecture. The Scholz-Brauer conjecture has been verified to hold for $n < 5784689$, by computations of Clift [18].

In this chapter we introduce and study an invariant of addition chain length related to small steps, where instead of rounding off we subtract off the exact logarithm $\log_2 n$. Formally:

Definition 4.1.1. The *addition chain defect* $\delta^{\ell}(n)$ of n is

$$\delta^{\ell}(n) := \ell(n) - \log_2 n.$$

This quantity is related to the number of small steps of n by the equation

$$s(n) = \lceil \delta^{\ell}(n) \rceil.$$

The lower bound result above shows that

$$\delta^{\ell}(n) \geq 0,$$

with equality holding for $n = 2^k$ for $k \geq 0$. The object of this chapter is to show that

the addition chain defect encodes a subtle structural regularity of the addition chain length function.

4.1.1 Main Results

We prove results about the structure of the set of integers having a given defect value α, and about the set of all defect values, called $\mathscr{D}^\ell$ below.

Our first result concerns determination of the set of integers having a given value α of the addition chain defect. We will show that If $\delta^\ell(n_1) = \delta^\ell(n_2) = \alpha$ with $n_1 \neq n_2$ then it is necessary (but not always sufficient) that $n_1 = 2^k n_2$ for some (positive or negative) integer k.

It is always the case that $\ell(2n) \leq \ell(n) + 1$, and the equality $\ell(2n) = \ell(n) + 1$ corresponds to $\delta^\ell(2n) = \delta^\ell(n)$. One might hope that we always have $\ell(2n) = \ell(n) + 1$, but this is not the case; sometimes $\delta^\ell(2n) < \delta^\ell(n)$. In fact, infinitely many counterexamples are known (Thurber [50]). However infinitely many integers n have this property, which is a stabilization phenomenon. We make the following definition.

Definition 4.1.2. A number m is called *ℓ-stable* if

$$\ell(2^k m) = \ell(m) + k, \quad \text{for all} \quad k \geq 0.$$

Otherwise it is called *ℓ-unstable*.

Using the defect, we will prove:

Theorem 4.1.3. (ℓ-stability theorem) *We have:*

1. *If α is a value of δ^ℓ, and*

$$S(\alpha) := \{m : \ \delta^\ell(m) = \alpha\}$$

   then there is a unique integer n such that $S(\alpha)$ has either the form $\{n \cdot 2^k : 0 \leq k \leq K\}$ for some finite K or else the form $\{n \cdot 2^k : k \geq 0\}$. The integer n will be called the leader *of $S(\alpha)$.*

2. *The set $S(\alpha)$ is infinite if and only if α is the smallest defect occurring among all defects $\delta^\ell(2^k n)$ for $k \geq 0$, where n is the leader of $S(\alpha)$.*

3. *For a fixed odd integer n, the sequence $\{\delta^\ell(n \cdot 2^k) : \ k \geq 0\}$ is non-increasing. This sequence takes on finitely many values, all differing by integers, culminat-*

ing in a smallest value α such that if $\delta^\ell(m) = \alpha$ and $k \geq 0$, then

$$\ell(m \cdot 2^k) = \ell(m) + k.$$

That is to say, while doubling a number n may not increase its addition chain length by precisely 1, if one starts with a fixed n and begins doubling, eventually one will reach a point where the length goes up by 1 each time. This result is easy to prove and is established in Section 4.3.

We use Theorem 4.1.3 to define in Section 4.3.2 a notion of the "stable defect" and "stable length" of a number n – these notions measure what the defect and the addition chain length would be "if n were stable".

The main results of the chapter concern the structure of the set of all addition chain defect values.

Definition 4.1.4. We define $\mathscr{D}^\ell$ to be the set of all addition chain defects:

$$\mathscr{D}^\ell = \{\delta^\ell(n) : n \in \mathbb{N}\}.$$

The main result of this chapter is the following well-ordering theorem.

Theorem 4.1.5. (ℓ-defect well-ordering theorem) *The set $\mathscr{D}^\ell$ is a well-ordered subset of $\mathbb{R}$, of order type ω^ω.*

The two main results of this chapter above are analogues for addition chains of results we previously showed for a another notion called integer complexity, (see Chapters 2 and 3) which has its own measure of defect. In Section 4.2 we discuss integer complexity, define its associated notion of defect $\delta(n)$, and compare it with addition chain complexity. Integer complexity has the feature that it is definable by a dynamic programming recursion, and this feature played an important role in the proof of well-ordering for defect values in Chapter 3. In contrast addition chain complexity is apparently not definable by dynamic programming recursion, and the proofs here require some new ideas.

As we will describe below, the proof of the main result for addition chains works in much greater generality, and we will obtain Theorem 4.1.5 as a special case of Theorem 4.1.13 below.

4.1.2 Methods

A key result which substitutes for dynamic programming and allows well ordering to the proved in the addition chain case is the following result of Schönhage [42]:

Theorem 4.1.6 (Schönhage)**.** *For any $n \geq 1$,*

$$\delta^\ell(n) \geq \log_2 \nu_2(n) - C_s,$$

where

$$C_s := \frac{2}{3} + \frac{2}{3}\log_2 3 - \frac{1}{\ln 2} - \log_2 \log \frac{4}{3} + \sum_{k=0}^{\infty} \log_2(1 + 2^{-6\cdot 2^k+1}) \leq 2.13.$$

Our proof of Theorem 4.1.13 (and hence of Theorem 4.1.5) requires only the assertion that $\delta^\ell(n)$ can be bounded below by some increasing unbounded function of $\nu_2(n)$ – in fact, similar but weaker inequalities were proven earlier by E. G. Thurber [49] and A. Cottrell [20]. However we can use Schönhage's inequality to prove more detailed information; see Theorem 4.6.4 and Corollary 4.6.7.

The idea of the proof is to consider initial segments of $\mathscr{D}^\ell$, say $\mathscr{D}^\ell \cap [0, r]$. By Theorem 4.1.6, numbers of bounded defect have boundedly many 1's in their binary expansion. But as we will show in Proposition 4.6.3, the set of defects arising from numbers with exactly k occurrences of 1 in their binary expansion is well-ordered and has order type at least ω^{k-1} but less than ω^k. From this fact we can conclude (Theorem 4.6.4) that $\mathscr{D}^\ell \cap [0, r]$ is well-ordered and has order type less than ω^ω, and thence that $\mathscr{D}^\ell$ itself is well-ordered with order type at most ω^ω. To get the lower bound on the order type, we note that $\mathscr{D}^\ell$ includes, for every k, the set of defects arising from numbers with exactly k occurrences of 1 in their binary expansion; by above, this means its order type must be at least ω^k for every natural k, and hence at least ω^ω.

4.1.3 Extensions and variations of the main theorem

In the discussion above we treated the addition chain length of n, but the theorems can be proved more generally for other, similar notions of addition chain complexity that put restrictions on the allowed set A of addition chains. A common variation on the notion of addition chains is the notion of the *star chain*; a star chain is an addition chain $(a_0, \ldots, a_r)$ with the additional restriction that for any $k \geq 1$, there exists $i < k$ such that $a_k = a_{k-1} + a_i$. The length of the shortest star chain for n,

called the *star chain length* of n, is denoted by $\ell^*(n)$. Naturally $\ell^*(n) \geq \ell(n)$, and it is known that $\ell^*(n) \sim \log_2 n$. We will see below that the results of this chapter apply to star chain length as well as addition length. Indeed, we can generalize much further.

Let A be a fixed set of addition chains, such as the set of all addition chains or the set of star chains. We will be considering the length of the shortest addition chain in A for a number n; we denote this length by $\ell^A(n)$. However we will not allow A to be an arbitrary set of addition chains, but require it to satisfy the following admissibility condition.

Definition 4.1.7. We define a set A of addition chains to be *admissible* if

1. For any n, there is an addition chain in A for n of length at most $\lfloor \log_2 n \rfloor + \nu_2(n) - 1$. That is to say, $\ell^A(n)$ is defined and is at most $\lfloor \log_2 n \rfloor + \nu_2(n) - 1$.

2. For any n, $\ell^A(2n) \leq \ell^A(n) + 1$.

The first of these conditions says that for any n, there are chains in A for n which are at least as short as those produced by the binary method. So, for instance, if A includes all chains produced by the binary method, it satisfies the first condition. The meaning of the second condition is straightforward. It is is satisfied if, for instance, given any chain in A for n, appending $2n$ again yields a chain in A, or if given any chain in A for n, doubling all the entries and prepending 1 again yields a chain in A.

Interesting examples of admissible sets of addition chains include:

1. the set of all addition chains;

2. the set of star chains;

3. the set of Hansen chains (also known as ℓ^0-chains, see Hansen [31], also [35, 46]);

4. the set of chains which are star or quasi-star (see Subbarao [46]).

Of course, there are trivial examples as well. For instance, one could let be A be just the set of addition chains produced by the binary method; then one would always have $\ell^A(n) = \lfloor \log_2 n \rfloor + \nu_2(n) - 1$. But the particular set of addition chains chosen will mostly not matter so long as it satisfies those two conditions.

One interesting set of addition chains that has been studied but which is not admissible is the set of *Lucas chains*, also known as *LUC chains*; they satisfy the second condition but not the first. (For instance, the shortest Lucas chain for 17 has length 6.) See Kutz [36] for more information on these.

Unless stated otherwise, we assume throughout that A is an admissible set of addition chains. We can now make definitions analogous to those above with ℓ^A replacing ℓ:

Definition 4.1.8. For an admissible set A of addition chains, we define the *A-defect*

$$\delta^A(n) := \ell^A(n) - \log_2 n.$$

If A is the set of all addition chains, we just write $\delta^\ell(n)$. If A is the set of star chains, we write $\delta^*(n)$.

Definition 4.1.9. For an admissible set A of addition chains, we define

$$\mathscr{D}^A = \{\delta^A(n) : n \in \mathbb{N}\}.$$

If A is the set of all addition chains, we just write $\mathscr{D}^\ell$. If A is the set of star chains, we write $\mathscr{D}^*$.

With these, we can once again define:

Definition 4.1.10. A number m is called *A-stable* if $\ell^A(2^k m) = k + \ell^A(m)$ holds for every $k \geq 0$. Otherwise it is called *A-unstable*. If A is the set of all addition chains, we write *ℓ-stable*. If A is the set of star chains, we write *$*$-stable*.

And with these, we once again get:

Theorem 4.1.11. (A-stability theorem) *Fix an admissible set A of addition chains. Then we have:*

1. *If α is a value of δ^A, and*

$$S(\alpha) := \{m : \ \delta^A(m) = \alpha\}$$

   then there is a unique integer n such that $S(\alpha)$ has either the form $\{n \cdot 2^k : 0 \leq k \leq K\}$ for some finite K or else the form $\{n \cdot 2^k : k \geq 0\}$. The integer n will be called the leader *of $S(\alpha)$.*

2. *The set $S(\alpha)$ is infinite if and only if α is the smallest defect occurring among all defects $\delta^\ell(2^k n)$ for $k \geq 0$, where n is the leader of $S(\alpha)$.*

3. *For a fixed odd integer n, the sequence $\{\delta^A(n \cdot 2^k) : k \geq 0\}$ is non-increasing. This sequence takes on finitely many values, all differing by integers, culminating in a smallest value α such that if $\delta^A(m) = \alpha$ and $k \geq 0$, then*

$$\ell^A(m \cdot 2^k) = \ell^A(m) + k.$$

Another interesting variation on the set $\mathscr{D}^\ell$ or $\mathscr{D}^A$ is to restrict to defects of stable numbers. We make the following definition:

Definition 4.1.12. We define an *A-stable defect* to be the defect of an A-stable number, and define $\mathscr{D}^A_{st}$ to be the set of all A-stable defects.

This double use of the word "stable" could potentially be ambiguous if we had a positive integer n which were also a defect. However, we will see (Corollary 4.3.5) that only integer which occurs as a defect is 0, and so this does not occur.

With these definitions, we obtain:

Theorem 4.1.13. (A-defect well ordering theorem) *For any admissible set A of addition chains, the sets $\mathscr{D}^A$ and $\mathscr{D}^A_{st}$ are well-ordered subsets of $\mathbb{R}$, of order type ω^ω. In particular, the sets $\mathscr{D}^\ell$, $\mathscr{D}^*$, $\mathscr{D}^\ell_{st}$, and $\mathscr{D}^*_{st}$ are well-ordered, with order type ω^ω.*

We remark that Schönhage's lower bound theorem plays the same role in establishing these well-ordering results as it does in the special case of all addition chains, since $\delta^A(n) \geq \delta^\ell(n)$.

4.1.4 Further remarks

To conclude this introduction, we add a few additional remarks.

First, a natural generalization of addition chains is *addition-subtraction chains*, where subtraction of two elements is permitted as an elementary operation. Here Schönhage [42] has proved a lower bound for addition-subtraction chains analogous to that in Theorem 4.1.6. However, our well-ordering result given in Theorem 4.1.5 does not generalize to addition-subtraction chains. Indeed, one can verify that for $k \geq 3$,

$$\ell^\pm(2^k - 1) = k + 1;$$

thus, if one were to define the *addition-subtraction chain defect*

$$\delta^\pm(n) := \ell^\pm(n) - \log_2 n,$$

then one would find that the image of this function contains the infinite decreasing sequence $1 - \log_2(1 - 2^{-k})$. It follows that the set of all addition-subtraction chain defects is not well ordered with respect to the usual ordering of the real line.

Secondly, our proof of the well ordering in Theorem 4.1.5 does not currently enable us to determine all the the cutoff values c_k such that the set of defect values $\mathscr{D}^\ell \cap [1, c_k)$ is of order type ω^k. In Section 4.7 we use the known classification of numbers with $s(n) = 1$ due to Gioia et al. [27] and of numbers with $s(n) = 2$ due to Knuth [35] in order to determine the cutoff values for $k = 1$ and $k = 2$ to be $c_1 = 1, c_2 = 2$ respectively. (Recall that $s(n)$ denotes $\lceil \delta^\ell(n) \rceil$.) In Remark 4.4.7 we discuss problems with determining values of c_k for higher k.

Thirdly, in the integer complexity case there exists an effectively computable algorithm for determining whether a given integer n is stable (see Chapter 5). We do not currently know of such an algorithm in the addition chain case, and hope to return to this question at a future time.

4.2 Comparison of addition chain complexity and integer complexity

The main results in this chapter are analogues for addition chains of results recently established for integer complexity. The *(integer) complexity* of a natural number n is the least number of 1's needed to write n using any combination of addition and multiplication, with the order of the operations specified using parentheses grouped in any legal nesting. For instance, $n = 11$ has a complexity of 8, since it can be written using 8 ones as $(1 + 1 + 1)(1 + 1 + 1) + 1 + 1$, but not with any fewer. This notion was implicitly introduced in 1953 by Kurt Mahler and Jan Popken [38], and later popularized by Richard Guy [29]. We denote the complexity of n by $\|n\|$.

The parallel results for integer complexity stem from a series of conjectures formulated in 2000 by J. Arias de Reyna [8]. They include a conjecture on stability for integer complexity, subsequently proved in 2012 by the author with J. Zelinsky [7], which is included here as Chapter 2. That paper introduced a notion of *(integer complexity) defect*

$$\delta(n) := ||n|| - 3 \log_3 n,$$

and proved stability using that notion. Some of Arias de Reyna's other conjectures were reformulated by the author in terms of a well-ordering of the values of the defect $\delta(n)$ for integer complexity, and a theorem establishing the well-ordering of the range of the defect function was recently proved by the author in [2], which is included here

as Chapter 3.

In this section we expand on this analogy between integer complexity and addition chain length. These notions have obvious similarities; each is in some sense a measure of the resources are required to build up the number n starting from 1. Both allow the use of addition, but integer complexity supplements this by allowing the use of multiplication, while addition chain length supplements this by allowing the reuse of any number at no additional cost once it has been constructed. Furthermore, both measures are approximately logarithmic; integer complexity satisfies the bounds

$$3\log_3 n = \frac{3}{\ln 3}\ln n \leq \|n\| \leq \frac{3}{\ln 2}\ln n, \qquad n > 1.$$

However, a difference worth noting is that while $\ell(n)$ is known to be asymptotic to $\log_2 n$ as mentioned above, the function $\|n\|$ is not known to be asymptotic to $3\log_3 n$; the value of the quantity $\limsup_{n\to\infty} \frac{\|n\|}{\ln n}$ remains unknown. Guy [29] has asked whether $\|2^k\| = 2k$ for $k \geq 1$; if true, it would make this quantity at least $\frac{2}{\ln 2}$. It is known that $\|2^k\| = 2k$ does hold for $1 \leq k \leq 48$; see Chapter 5.

Another difference worth noting between the two notions is that integer complexity, unlike addition chain length, can be computed via dynamic programming. Specifically, for any $n > 1$,

$$\|n\| = \min_{\substack{a,b<n\in\mathbb{N} \\ a+b=n \text{ or } ab=n}} \|a\| + \|b\|.$$

By contrast, addition chain length is harder to compute. Suppose we have a shortest addition chain $(a_0, \ldots, a_{r-1}, a_r)$ for n; one might hope that $(a_0, \ldots, a_{r-1})$ is a shortest addition chain for a_{r-1}, but this need not be the case. An example is provided by the addition chain $(1, 2, 3, 4, 7)$; this is a shortest addition chain for 7, but $(1, 2, 3, 4)$ is not a shortest addition chain for 4, as $(1, 2, 4)$ is shorter. Moreover, there is no way to assign to each natural number n a shortest addition chain $(a_0, \ldots, a_r)$ for n such that $(a_0, \ldots, a_{r-1})$ is the addition chain assigned to a_{r-1} [35]. This can be an obstacle both to computing addition chain length and proving statements about addition chains.

However, when one examines certain particular aspects of integer complexity and addition chains, similarities once again appear. The stabilization result Theorem 4.1.11 is analogous to Theorem 2.1.5. Meanwhile, the well-ordering result Theorem 4.1.13 is analogous to part of Theorem 3.1.2. Unfortunately, it is substantially weaker than a direct analogue of Theorem 3.1.2, since it does not tell us where the supremum of the initial ω^k defects occurs. We prove some bounds on this at the end

of Section 4.6 and in Section 4.7. We suspect that the supremum of the initial ω^k defects is at least k, for addition chains and for star chains; see Conjecture 4.8.1 and Question 4.8.2.

4.3 The A-defect and A-stabilization

We will give proofs in this chapter for an arbitrary admissible set A of addition chains.

4.3.1 A-defect

The A-defect is the basic object of study in this chapter.

Proposition 4.3.1. *Let A be an admissible set of addition chains. We have*

1. *For all integers $a \geq 1$,*
$$\delta^A(a) \geq 0.$$
Here equality holds precisely when $a = 2^k$ for some $k \geq 0$.

2. *For $k \geq 0$,*
$$\delta^A(2^k n) \leq \delta^A(n).$$
The difference is an integer, and equality holds if and only if
$$\ell^A(2^k n) = \ell^A(n) + k.$$

Proof. The first statement in part (1) is just the lower bound $\ell^A(n) \geq \log_2 n$. And for $n = 2^k$, we know that $\ell^A(n) = k$, so $\delta^A(n) = 0$. For the converse, note that $\log_2 n$ is only an integer if n is a power of 2.

For part (2), note that by the requirements on A we have

$$\ell^A(2^k n) \leq k + \ell^A(n). \tag{4.1}$$

Subtracting $k + \log_2 n$ from both sides yields the stated inequality. Furthermore, since (4.1) is an inequality of integers, the difference is an integer; and we have equality in the result if and only if we had equality in (4.1). □

As was noted in Section 4.1.1, though one might hope that $\ell(2n) = \ell(n) + 1$ in general, infinitely many counterexamples are known [50]. Still, based on this idea, we defined in Section 4.1.1 the notions of an *ℓ-stable number* and in Section 4.1.3 the notion of an *A-stable number*.

This can be alternately characterized as follows:

Proposition 4.3.2. *The number m is A-stable if and only if $\delta^A(2^k m) = \delta^A(m)$ for all $k \geq 0$.*

Proof. This is immediate from Proposition 4.3.1(2). □

This is already enough to prove the following:

Theorem 4.3.3. *We have*

1. *For any $m \geq 1$, there exists a finite $K \geq 0$ such that $2^K m$ is A-stable.*
2. *If the defect $\delta^A(m)$ satisfies $0 \leq \delta^A(m) < 1$, then m itself is A-stable.*

Proof. (1) From Proposition 4.3.1, we have that for any n, $\delta^A(2n) \leq \delta^A(n)$, with equality if and only if $\ell^A(2n) = \ell^A(n) + 1$. More generally,

$$\delta^A(n) - \delta^A(2n) = \ell^A(n) + 1 - \ell^A(2n),$$

and so the difference $\delta^A(n) - \delta^A(2n)$ is always an integer. This means that the sequence $\delta^A(m), \delta^A(2m), \delta^A(4m), \ldots$ is non-increasing, nonnegative, and can only decrease in integral amounts; hence it must eventually stabilize. Applying Proposition 4.3.2 proves the theorem.

(2) If $\delta^A(m) < 1$, since all $\delta^A(n) \geq 0$ there is no room to remove any integral amount, so m must be A-stable. □

Note that while this proof shows that for any n there is some K such that $2^K n$ is A-stable (in particular, ℓ-stable or $*$-stable), it does not give any upper bound on K.

Because we use the actual logarithm, the value of the defect is enough to determine a number up to a power of 2:

Proposition 4.3.4. *Suppose that m and n are two positive integers, with $m \geq n$. If $q := \delta^A(n) - \delta^A(m)$ is rational, then it is necessarily a nonnegative integer, and furthermore $m = n \cdot 2^k$ for some $k \geq 0$. In particular this holds if $\delta^A(n) = \delta^A(m)$.*

Proof. If $q = \delta^A(n) - \delta^A(m)$ is rational, then $\log_2(m/n)$ is rational; since m/n is rational, the only way this can occur is if $\log_2(m/n)$ is an integer k, in which case, since $m > n$, $m = n \cdot 2^k$ with $k \geq 0$. It then follows from the definition of defect that $q = \ell^A(n) + k - \ell^A(m)$. □

Corollary 4.3.5. *No nonzero integer occurs as $\delta^A(n)$ for any n.*

Proof. If $\delta^A(n) \in \mathbb{Z}$, then $n = 2^k$ for some $k \geq 0$ by Proposition 4.3.4; but then $\delta^A(n) = 0$. □

We can now prove Theorems 4.1.11 and 4.1.3:

Proof of Theorem 4.1.11. For part (3), the non-increasing assertion follows from part (2) of Proposition 4.3.1. Also, part (1) of Theorem 4.3.3 implies that eventually the sequence stabilize; hence it can take only finitely many values.

For part (1), the assertion about the form of $S(\alpha)$ follows from Proposition 4.3.4. The rest, and part (2), follows from the fact that $\delta^A(2^k n)$ is nonincreasing as a function of k. □

Proof of Theorem 4.1.3. This is just Theorem 4.1.11 in the case when A is the set of all addition chains. □

4.3.2 A-stable defects and A-stable length

Knowing the defect of a number also tells us whether or not that number is stable:

Proposition 4.3.6. *If $\delta^A(n) = \delta^A(m)$ and n is A-stable, then so is m.*

Proof. Suppose $\delta^A(n) = \delta^A(m)$ and n is A-stable. Then we can write $m = 2^k n$ for some $k \in \mathbb{Z}$. Now, a number a is A-stable if and only if $\delta^A(2^j a) = \delta^A(n)$ for all $j \geq 0$; so if $k \geq 0$, then m is A-stable. While if $k < 0$, then consider $j \geq 0$; if $j \geq -k$, then $\delta^A(2^j m) = \delta^A(2^{j+k} n) = \delta^A(n)$, while if $j \leq -k$, then $\delta^A(n) \leq \delta^A(2^j m) \leq \delta^A(m)$, so $\delta^A(2^j m) = \delta^A(m)$; hence m is A-stable. □

Because of this proposition, Definition 4.1.12 makes more sense; a stable defect is not just the defect of a stable number, but one for which all numbers with that defect are stable.

Proposition 4.3.7. *A defect α is A-stable if and only if it is the smallest $\beta \in \mathscr{D}^A$ such that $\beta \equiv \alpha \pmod 1$.*

Proof. This follows from part (2) of Proposition 4.3.1, Proposition 4.3.4, and part (1) of Theorem 4.1.11. □

Definition 4.3.8. For a positive integer n, define the *stable defect of n with regard to A*, denoted $\delta^A_{st}(n)$, to be $\delta^A(2^k n)$ for any k such that $2^k n$ is A-stable. (This is well-defined as if $2^k n$ and $2^j n$ are A-stable, then $k \geq j$ implies $\delta^A(2^k n) = \delta^A(2^j n)$, and so does $j \geq k$.)

Here are two equivalent characterizations of stable defect:

Proposition 4.3.9. *The number $\delta^A_{st}(n)$ can be characterized by:*

1. $\delta^A_{st}(n) = \min_{k\geq 0} \delta^A(2^k n)$
2. $\delta^A_{st}(n)$ *is the smallest* $\alpha \in \mathscr{D}^A$ *such that* $\alpha \equiv \delta(n) \pmod 1$.

Proof. Part (1) follows from part (2) of Theorem 4.3.1 and the fact that m is A-stable if and only if $\delta^A(2^k m) = \delta^A(m)$ for all $k \geq 0$. To prove part (2), take k such that $2^k n$ is A-stable. Then $\delta^A(2^k n) \equiv \delta^A(n) \pmod 1$, and it is the smallest such by Proposition 4.3.7. □

So we can think about $\mathscr{D}^A_{st}$ either as the subset of $\mathscr{D}^A$ consisting of the A-stable defects, or we can think of it as the image of δ^A_{st}. This double characterization will be useful in Section 4.6.

Just as we can talk about the stable defect of a number n, we can also talk about its *stable length* – what the length of n would be "if n were stable".

Definition 4.3.10. For a positive integer n, we define the *stable length of n with regard to A*, denoted $\ell^A_{st}(n)$, to be $\ell^A(2^k n) - k$ for any k such that $2^k n$ is A-stable. This is well-defined; if $2^k n$ and $2^j n$ are both stable, say with $k \leq j$, then

$$\ell^A(2^k n) - k = k - j + \ell^A(2^j n) - k = \ell^A(2^j n) - j.$$

Proposition 4.3.11. *We have:*

1. $\ell^A_{st}(n) = \min_{k\geq 0}(\ell^A(2^k n) - k)$
2. $\delta^A_{st}(n) = \ell^A_{st}(n) - \log_2 n$

Proof. To prove part (1), observe that $\ell^A(2^k n) - k$ is nonincreasing in k, since $\ell^A(2m) \leq 1 + \ell^A(m)$. So a minimum is achieved if and only if for all j,

$$\ell^A(2^{k+j} n) - (k + j) = \ell^A(2^k n) - k,$$

i.e., for all j, $\ell^A(2^{k+j} n) = \ell^A(2^k n) + j$, i.e., $2^k n$ is A-stable.

To prove part (2), take k such that $2^k n$ is A-stable. Then

$$\delta^A_{st}(n) = \delta^A(2^k n) = \ell^A(2^k n) - \log_2(2^k n) = \ell^A(2^k n) - k - \log_2 n = \ell^A_{st}(n) - \log_2 n.$$

□

Proposition 4.3.12. *We have:*

1. $\delta_{st}^A(n) \leq \delta^A(n)$, *with equality if and only if* n *is* A*-stable.*

2. $\ell_{st}^A(n) \leq \ell^A(n)$, *with equality if and only if* n *is* A*-stable.*

Proof. The inequality in part (1) follows from Proposition 4.3.9. Also, if n is A-stable, then for any $k \geq 0$, $\delta^A(2^k n) = \delta(n)$, so $\delta_{st}^A(n) = \delta^A(n)$. Conversely, if $\delta_{st}^A(n) = \delta^A(n)$, then by Proposition 4.3.9, for any $k \geq 0$, $\delta^A(2^k n) \geq \delta^A(n)$. But also $\delta^A(2^k n) \leq \delta^A(n)$ by part (2) of Theorem 4.3.1, and so $\delta^A(2^k n) = \delta^A(n)$ and n is A-stable.

Part (2) follows from part (1) along with part (2) of Proposition 4.3.11. □

4.4 Bit-counting in numbers of small defect

Schönhage's Theorem, Theorem 4.1.6, implies that for any real $r \geq 0$, there is an upper bound on how many 1's can appear in the binary expansion of a number with addition chain defect at most r. Because of this, we define:

Definition 4.4.1. We define a function $q : [0, \infty) \to \mathbb{N}$ by

$$q(r) = \max_{\delta^\ell(n) \leq r} \nu_2(n).$$

More generally, for an admissible set of addition chains A, we can define

$$q^A(r) = \max_{\delta^A(n) \leq r} \nu_2(n).$$

Then in this language, Theorem 4.1.6 says the following:

Proposition 4.4.2. *For* $r \geq 0$,

$$q(r) \leq \lfloor 2^{r+C_s} \rfloor.$$

Proof. Solving Theorem 4.1.6 for $\nu_2(n)$ yields the inequality $\nu_2(n) \leq 2^{\delta^\ell(n)+C_s}$; since $\nu_2(n)$ is an integer, it follows that $\nu_2(n) \leq \lfloor 2^{\delta^\ell(n)+C_s} \rfloor$. Hence, $q(r) \leq \lfloor 2^{r+C_s} \rfloor$. □

Note, by the way, the following properties of $q^A(r)$:

Proposition 4.4.3. *Let* A *and* B *be admissible sets of addition chains. We have:*

1. *The function* $q^A(r)$ *is nondecreasing in real* $r \geq 0$.

2. *For* $B \subseteq A$ *and any* r, $q^B(r) \leq q^A(r)$. *In particular,* $q^A(r) \leq q(r)$.

Proof. To prove part (1), observe that as r increases, the set $\{n : \delta^A(n) \leq r\}$ gets larger, and hence so does $q^A(r)$ as it is a maximum taken over that set. To prove part (2), note that for any n, $\delta^A(n) \leq \delta^B(n)$ and so the set $\{n : \delta^B(n) \leq r\}$ is contained in the set $\{n : \delta^A(n) \leq r\}$; thus $q^A(r)$ is at least as large as it is a maximum over a superset. □

Schönhage was not the first to investigate the relation between $\nu(n)$ and $\delta^\ell(n)$ – or rather, between $\nu(n)$ and $s(n)$, since $s(n)$ rather than $\delta^\ell(n)$ has been the primary object of study of previous authors. Schönhage's theorem is a partial result towards the following conjecture of Knuth and Stolarsky [35, 47, 46]:

Conjecture 4.4.4 (Knuth, Stolarsky)**.** *For all n, $s(n) \geq \log_2 \nu_2(n)$.*

The Knuth-Stolarsky conjecture is known to be true for $0 \leq s(n) \leq 3$. The case $s(n) = 0$ is trivial; the case $s(n) = 1$ was proved by Gioia et al. [27]; the case $s(n) = 2$ was proved by Knuth [35]; and the case $s(n) = 3$ was proved by Thurber [50]. In fact, Knuth proved a more detailed theorem about the case $s(n) = 2$; we will make use of this in Section 4.7.2. We summarize these results formally here:

Theorem 4.4.5 (Gioia et al., Knuth, Thurber)**.** *We have:*

1. *For a natural number n, $s(n) = 0$ if and only if $\nu_2(n) = 1$.*
2. *For a natural number n, $s(n) = 1$ if and only if $\nu_2(n) = 2$.*
3. *For a natural number n, if $s(n) = 2$, then $\nu_2(n) = 3$ or $\nu_2(n) = 4$.*
4. *For a natural number n, if $s(n) = 3$, then $\nu_2(n) \leq 8$.*

This theorem yields:

Proposition 4.4.6. *For k an integer with $0 \leq k \leq 3$, $q(k) = 2^k$.*

Proof. For $0 \leq k \leq 3$ an integer, if $\delta^\ell(n) \leq k$, then $\nu_2(n) \leq 2^k$ by Theorem 4.4.5. That is to say, $q(k) \leq 2^k$. For the converse, observe that $s(1) = 0$ and $\nu_2(1) = 1$, so $q(0) \geq 1$; $s(3) = 1$ and $\nu_2(3) = 2$, so $q(1) \geq 2$; $s(15) = 2$ and $\nu_2(15) = 4$, so $q(2) \geq 4$; and $s(255) = 3$ and $\nu_2(255) = 8$, so $q(3) \geq 8$. □

So while Schönhage's theorem yields the best known result for large r, these results settle the matter for small r.

Remark 4.4.7. In Section 4.6, we will give an upper bound on the order type of $\mathscr{D}^\ell \cap [0, r]$ in terms of $q(r)$. So while in this chapter we state concrete bounds proved using Theorem 4.1.6, any improvement in the upper bounds on $q(r)$ – for instance, a proof of the Knuth-Stolarsky conjecture – would improve these bounds. (Note that if one wants merely to prove Theorem 4.1.5, it suffices to know that $q(r)$ is well-defined; one does not even need to know any bounds on it at all.) However, this does not mean that one is limited to bounds based on $q(r)$; in Section 4.7.2, we will demonstrate an example of a bound that goes beyond what one can learn from study of $q(r)$ alone.

4.5 Cutting and pasting well-ordered sets

We pause to recall some external facts dealing with the cutting and pasting of well-ordered sets. We begin with the following theorem of P. W. Carruth [17]:

Theorem 4.5.1. *Let S be a well-ordered set and suppose $S = S_1 \cup S_2$. Then the order type of S is at most the* natural sum *of the order types of S_1 and S_2.*

The natural sum is defined as follows [17]:

Definition 4.5.2. The *natural sum* (also known as the *Hessenberg sum*) [21] of two ordinals α and β, here denoted $\alpha \oplus \beta$, is defined by simply adding up their Cantor normal forms as if they were "polynomials in ω". That is to say, if there are ordinals $\gamma_0 < \ldots < \gamma_n$ and whole numbers $a_0, \ldots, a_n$ and $b_0, \ldots, b_n$ such that $\alpha = \omega^{\gamma_n} a_n + \ldots + \omega^{\gamma_0} a_0$ and $\beta = \omega^{\gamma_n} b_n + \ldots + \omega^{\gamma_0} b_0$, then

$$\alpha \oplus \beta = \omega^{\gamma_n}(a_n + b_n) + \ldots + \omega^{\gamma_0}(a_0 + b_0).$$

Theorem 4.5.1 is sometimes used as the definition of the natural sum [17]. There is also a recursive definition [19]. There is also a similar natural product [17, 21], but we will not be using it here. See [21] for generalizations of this theorem.

From this we can then conclude:

Proposition 4.5.3. *For any ordinal α:*

1. *If S is a well-ordered set and $S = S_1 \cup \ldots \cup S_n$, and S_1 through S_n all have order type less than ω^α, then so does S.*

2. *If S is a well-ordered set of order type ω^α and $S = S_1 \cup \ldots \cup S_n$, then at least one of S_1 through S_n also has order type ω^α.*

Proof. For (1), observe that the order type of S is at most the natural sum of those of $S_1, \ldots, S_n$, and the natural sum of ordinals less than ω^α is again less than ω^α.

For (2), by (1), if $S_1, \ldots, S_k$ all had order type less than ω^α, so would S; so at least one has order type at least ω^α, and it necessarily also has order type at most ω^α, being a subset of S. □

We can say more when the sets are interleaved with each other:

Proposition 4.5.4. *Suppose α is an ordinal and S is a well-ordered set which can be written as a finite union $S_1 \cup \ldots \cup S_k$ such that:*

1. *The S_i all have order types at most ω^α*
2. *If a set S_i has order type ω^α, it is cofinal in S.*

Then the order type of S is at most ω^α. In particular, if at least one of the S_i has order type ω^α, S has order type ω^α.

Proof. Consider a proper initial segment of S; call it T. Let x be the smallest element of $S \setminus T$. Let $\mathcal{A}$ be the set of S_i of order type ω^α. Since each element of $\mathcal{A}$ is cofinal in S, each contains some element that is at least x, and thus not in T. That is, for $S_i \in \mathcal{A}$, $T \cap S_i$ is always a proper initial segment of S_i. Thus T is a finite union of proper initial segments of the elements of $\mathcal{A}$ and possibly improper initial segments of the S_i not in $\mathcal{A}$. But any set of either of these types has order type strictly less than ω^α, and so by Proposition 4.5.3, so would T. Since each proper initial segment of S has order type less than ω^α, it follows that S has order type at most ω^α. If furthermore some S_i has order type ω^α, then S also has order type at least ω^α and thus exactly ω^α. □

We'll be applying these propositions to take apart and put together sets of defects in the subsequent sections.

Also worth noting is the following fact.

Proposition 4.5.5. *Let X be a totally ordered set with the least upper bound property, and S a well-ordered subset of X of order type α. Then $\overline{S}$ is a well-ordered subset of S of order type either α or $\alpha + 1$, and if $\beta < \alpha$ is a limit ordinal, the β'th element of $\overline{S}$ is the supremum (limit) of the initial β elements of S.*

Proof. This result was proved earlier as Corollary 3.5.7. □

4.6 Well-ordering of defects

Now we are prepared to prove that the set of defects is well-ordered.

4.6.1 Well-ordering of defect sets for n with $\nu_2(n) \leq k$

First we observe:

Proposition 4.6.1. *For any n, $\delta^A_{st}(n) \leq \delta^A(n) \leq \nu_2(n) - 1$.*

Proof. We know $\delta^A_{st}(n) \leq \delta^A(n)$ by Proposition 4.3.12, and the rest is immediate as

$$\delta^A(n) = \ell^A(n) - \log_2 n \leq \lfloor \log_2 n \rfloor - \log_2 n + \nu_2(n) - 1 \leq \nu_2(n) - 1.$$

□

Next we show that, applied to numbers with a fixed number of 1's in the binary expansion, the binary method produces a well-ordered set of defects.

Proposition 4.6.2. *Let $k \geq 1$ be a natural number, and define the set S_k to be*

$$\{k - 1 + \lfloor \log_2 n \rfloor - \log_2 n : \nu_2(n) = k\}.$$

Then S_k is a well-ordered set, with order type ω^{k-1}.

Proof. If $\nu(n) = k$, write $n = 2^{a_0} + \ldots + 2^{a_{k-1}}$. Then $\lfloor \log_2 n \rfloor = a_0$ and

$$k - 1 + \lfloor \log_2 n \rfloor - \log_2 n = k - 1 - \log_2(1 + 2^{a_1 - a_0} + \ldots + 2^{a_{k-1} - a_0}).$$

We observe then that S_k can also be written as

$$\{k - 1 - \log_2(1 + 2^{-b_1} + \ldots + 2^{-b_{k-1}}) : 0 < b_1 < b_2 < \ldots < b_{k-1} \in \mathbb{Z}\}.$$

This set contains S_k as $a_0 > a_i$ for $i > 0$ and the sequence of a_i is decreasing, and the converse holds as, given $b_1, \ldots, b_{k-1}$, we can pick $a_0 = \sum_{i=1}^{k-1} b_i$ and $a_i = a_0 - b_i$ for $i > 0$. Now we can write down an order-preserving bijection $\phi : \omega^{k-1} \to S_k$. Define $\phi(c_1, \ldots, c_{k-1}) = k - 1 - \log_2(1 + 2^{-b_1} + \ldots + 2^{-b_{k-1}})$, where

$$b_i = i + \sum_{j=0}^{i} c_j.$$

This is a bijection as, since an element of S_k is identified by its sequence of $b_1, \ldots, b_{k-1}$, it has inverse given by

$$c_i = b_i - b_{i-1} - 1$$

(where we take $b_0 = 0$). To see this is order-preserving, take $(c_1, \ldots, c_{k-1}) < (c'_1, \ldots, c'_{k-1})$; say $c_1 = c'_1, \ldots, c_i = c'_i, c_{i+1} < c'_{i+1}$. Then $b_j = b'_j$ for $1 \leq j \leq i$ and $b'_{i+1} > b_{i+1}$.

$$2^{-b_1} + \ldots + 2^{-b_{k-1}} > 2^{-b'_1} + \ldots + 2^{-b'_{k-1}}$$

as they have the same binary expansion up to 2^{-b_i} place, but the former's next 1 occurs at $2^{-b_{i+1}}$, and the latter's next 1 occurs at $2^{-b'_{i+1}}$, and $b'_{i+1} > b_{i+1}$. Since $k - 1 - \log_2(1 + 2^{-b_1} + \ldots + 2^{-b_{k-1}})$ is an order-reversing function of $2^{-b_1} + \ldots + 2^{-b_{k-1}}$, this implies $\phi(c_1, \ldots, c_{k-1}) < \phi(c'_1, \ldots, c'_{k-1})$, proving the claim. □

Next we see that this is true even when chains may be shorter than those produced by the binary method:

Proposition 4.6.3. *For $k \geq 1$, the set $\{\delta^A(n) : \nu_2(n) = k\}$ is a well-ordered subset of the real numbers, with order type at least ω^{k-1} and at most $\omega^{k-1}k < \omega^k$. The same is true of the set $\{\delta^A_{st}(n) : \nu_2(n) = k\}$.*

Proof. We prove it here for the set $\{\delta^A(n) : \nu_2(n) = k\}$; the proof for the set $\{\delta^A_{st}(n) : \nu_2(n) = k\}$ is analogous.

Say $\nu_2(n) = k$, and write $n = 2^{a_0} + \ldots + 2^{a_{k-1}}$. Then $\ell^A(n) \leq k - 1 + a_0$, i.e., $\ell^A(n) = k - 1 + a_0 - m$ for some integer $m \geq 0$. So also

$$\delta^A(n) = k - 1 + a_0 - m - \log_2 n \leq k - 1 - m.$$

But also $\delta^A(n) \geq 0$, so $m \leq k - 1$. As m is an integer, this means $m \in \{0, \ldots, k-1\}$, a finite set.

So if we fix k and let T be the set $\{\delta^A(n) : \nu_2(n) = k\}$ and U be the set $\{k - 1 - \log_2 n : \nu_2(n) = k\}$, then we see that T is covered by finitely many translates of S_k from Proposition 4.6.2; more specifically, we can partition S_k into $U_0, \ldots, U_{k-1}$ such that

$$T = U_0 \cup U_1 - 1 \cup \ldots \cup U_{k-1} - (k-1).$$

But by Proposition 4.6.2, S_k has order type ω^{k-1}. So the U_i all have order type at most ω^{k-1}, and by Proposition 4.5.3 at least one has order type ω^{k-1}. Hence T is well-ordered of order type at most $\omega^{k-1}k < \omega^k$ by Proposition 4.5.1, and by above it also has order type at least ω^{k-1}. □

4.6.2 Well-ordering of initial segment of A-defect set

Finally we apply the existence of an upper bound on ν_2 in terms of δ^ℓ to prove the theorem:

Theorem 4.6.4. (Well-ordering of intial segments of A-defect set) *Let A be an admissible set of addition chains, and let $r \geq 0$ be a real number. Then $\mathscr{D}^A \cap [0, r]$ is a well-ordered subset of the real numbers with order type at least $\omega^{\lfloor r \rfloor}$ and at most*

$$\omega^{q^A(r)-1} q^A(r) + \ldots + \omega^2 3 + \omega 2 + 1,$$

which is less than $\omega^{q^A(r)-1}(q^A(r)+1)$ and hence less than $\omega^{q^A(r)}$. The same is true of $\mathscr{D}^A_{st} \cap [0, r]$.

Proof. Say n is a number with $\delta^A(n) \leq r$; then $\nu_2(n) \leq q^A(r)$. So $\mathscr{D}^A \cap [0, r]$ can be covered by the sets $\{\delta^A(n) : \nu_2(n) = k\}$ for $k = 1, 2, \ldots, q^A(r)$. By Proposition 4.6.3, each of these sets is well-ordered, with order type at most $\omega^{k-1}k$. Hence by Proposition 4.5.1, $\mathscr{D}^A \cap [0, r]$ is well-ordered with order type at most

$$\omega^{q^A(r)-1} q^A(r) + \ldots + \omega^2 3 + \omega 2 + 1,$$

which is less than $\omega^{q^A(r)-1}(q^A(r)+1)$ and hence less than $\omega^{q^A(r)}$. Since $\mathscr{D}^A_{st} \cap [0, r]$ is a subset of $\mathscr{D}^A \cap [0, r]$, this upper bound applies to it as well.

For the lower bound, observe that the set $\{\delta^A_{st}(n) : \nu_2(n) = \lfloor r \rfloor + 1\}$ is, by Proposition 4.6.1, entirely contained within $\mathscr{D}^A_{st} \cap [0, r]$, and by Proposition 4.6.3 it has order type at least $\omega^{\lfloor r \rfloor}$, and thus so does $\mathscr{D}^A_{st} \cap [0, r]$, and so also does $\mathscr{D}^A \cap [0, r]$. □

If we plug in Theorem 4.4.2, we get an explicit version of this. We can also plug in the other bounds in Section 4.4 to yield explicit versions of this that will be worse for large r but sometimes better for small r; see Section 4.7 for more on this.

We can now prove Theorem 4.1.13.

Proof of Theorem 4.1.13. We prove it for $\mathscr{D}^A$; the proof for $\mathscr{D}^A_{st}$ is analogous. Take an initial segment of $\mathscr{D}^A$, say $\mathscr{D}^A \cap [0, r)$. Then this is contained in $\mathscr{D}^A \cap [0, r]$ and so well-ordered with order type less than $\omega^{q^A(r)}$ by Theorem 4.6.4. Hence $\mathscr{D}^A$ is well-ordered with order type at most ω^ω, as all its initial segments are well-ordered with order type less than ω^ω. Furthermore, for any whole number k, $\mathscr{D}^A \cap [0, k]$ is well-ordered with order type at least ω^k by Theorem 4.6.4, so $\mathscr{D}^A$ must have order type at least ω^ω as well. □

Proof of Theorem 4.1.5. This follows immediately from Theorem 4.1.13 by taking A to be the set of all addition chains. □

4.6.3 Cutoff values $f^A(k)$ for ω^k-limit points

We can turn the well-ordering question around and consider, what is the supremum (limit) of the initial ω^k defects? This is of course essentially the same question, but it is also a helpful way of thinking about the question, so we note the results here.

Definition 4.6.5. We define $f^A(k)$ to be the limit of the initial ω^k defects in $\mathscr{D}^A$, and $f^A_{st}(k)$ to be the limit of the initial ω^k defects in $\mathscr{D}^A_{st}$. Note that by Proposition 4.5.5, if $k \geq 1$, this is the same as the ω^k'th element of $\overline{\mathscr{D}^A}$ (or $\overline{\mathscr{D}^A_{st}}$), while if $k = 0$, this is the same as the 0'th element of $\overline{\mathscr{D}^A}$ (or $\overline{\mathscr{D}^A_{st}}$). If A is the set of all addition chains we will write f^ℓ; if A is the set of star chains we will write f^*.

Proposition 4.6.6. *For any k, we have $f^A(k) \leq f^A_{st}(k)$.*

Proof. The set $\mathscr{D}^A_{st}$ is a subset of $\mathscr{D}^A$; hence for $\alpha < \omega^\omega$, the α'th element of $\mathscr{D}^A_{st}$ is at least the α'th element of $\mathscr{D}^A$. Taking limits, $f^A_{st}(k) \geq f^A(k)$. □

We now have the following corollary of Theorem 4.6.4:

Corollary 4.6.7. *We have*

$$\log_2(k+1) - 2.13 < \log_2(k+1) - C_s < f^A(k) \leq f^A_{st}(k) \leq k.$$

Proof. For the upper bound, observe that by Theorem 4.6.4, the order type of $\mathscr{D}^A_{st} \cap [0,k]$ is at least ω^k, so $\mathscr{D}^A_{st}(\omega^k) \leq k$.

For the lower bound, consider $\mathscr{D}^A \cap [0,r]$ with $r < \log_2(k+1) - C_s$. Then $2^{r+C_s} < k+1$, so $\lfloor 2^{r+C_s} \rfloor \leq k$. Since $q^A(r) \leq \lfloor 2^{r+C_s} \rfloor \leq k$ by Theorem 4.4.2 and Proposition 4.4.3, by Theorem 4.6.4, $\mathscr{D}^A \cap [0,r]$ has order type less than $\omega^{k-1}(k+1)$. Hence, if we consider $\mathscr{D}^A \cap [0, \log_2(k+1) - C_s)$, all its proper initial segments have order type less than $\omega^{k-1}(k+1)$, and so it has order type at most $\omega^{k-1}(k+1) < \omega^k$. Thus we must have $f^A(k) > \log_2(k+1) - C_s$. □

We will examine this question further in the next section, where we will improve this for small k.

4.7 Bounds on order type for small A-defect values

In the previous section, we proved bounds on the order types of $\mathscr{D}^A \cap [0, r]$ and $\mathscr{D}^A_{st} \cap [0, r]$. However, as was noted in Section 4.4, we can say more when r is small. First, we note the implications of the theorems in Section 4.4 regarding the functions f^A and f^A_{st} defined in the previous section. Then we will perform a more detailed examination of the case $r \leq 2$ using a theorem of Knuth. Then we compile these results to present bounds on $f^A(k)$ and $f^A_{st}(k)$ when k is small. We also make some notes on stability of numbers of small defect.

4.7.1 Bound for A-defect $r < 1$

The case of $r \leq 1$ can be handled with part (2) of Theorem 4.4.5, that was proved by Gioia et al.

Theorem 4.7.1. *The order type of $\mathscr{D}^A \cap [0, 1]$ is ω, while for any $r < 1$, $\mathscr{D}^A \cap [0, 1]$ is finite. Furthermore, all defects in $\mathscr{D}^A \cap [0, 1]$ are A-stable, and so the order type of $\mathscr{D}^A_{st} \cap [0, 1]$ is ω.*

Proof. Suppose that $\lceil \delta^A(n) \rceil = 1$. Then $\delta^\ell(n) \leq \delta^A(n) \leq 1$, so $\lceil \delta^\ell(n) \rceil = 1$ unless n is a power of 2, and n cannot be a power of 2, as then we would have $\delta^A(n) = 0$. So we can apply Theorem 4.4.5 to conclude that n can be written as $2^a + 2^b$ for some $b > a$. Conversely, if $n = 2^a + 2^b$ with $b > a$, then $\ell^A(n) \leq b + 1$ by the assumption that A is admissible, and we cannot have $\ell^A(n) \leq b$ as otherwise we would have $\delta^A(n) < 0$; so $\ell^A(n) = b + 1$.

Thus the set $\mathscr{D}^A \cap [0, 1]$ is precisely $\{0\} \cup S_2$, where S_2 is as in Proposition 4.6.2. Thus by that same proposition it has order type ω. Also it is easily seen to have a supremum of 1, so for $r < 1$, the set $\mathscr{D}^A \cap [0, r]$ is a proper initial segment of $\mathscr{D}^A \cap [0, 1]$ and so has strictly smaller order type.

Furthermore, if $n = 2^a + 2^b$ with $b > a$, then $2^k n = 2^{a+k} + 2^{b+k}$, and so $\ell^A(2^k n) = b + k + 1 = k + \ell^A(n)$, and so n is A-stable. And if $n = 2^b$, then $\ell^A(2^k n) = b + k = k + \ell^A(n)$, and so again n is A-stable. This proves the stability part of the theorem. □

4.7.2 Bounds for A-defect $r < 2$.

For the case $k = 2$, we will need to go beyond what is in Theorem 4.4.5. We state here the full theorem regarding numbers with 2 small steps, as proved by Knuth [35]:

Theorem 4.7.2 (Knuth)**.** *For a positive integer n, $s(n) = 2$ if and only if n can be written in one of the following forms:*

1. $2^a + 2^b + 2^c$ *for* $0 \leq a < b < c$

2. $2^a + 2^{a+1} + 2^{a+2} + 2^{a+7}$ *for* $a \geq 0$

3. $2^a + 2^{a+1} + 2^b + 2^{b+3}$ *for* $b > a + 1$, $a \geq 0$

4. $2^a + 2^b + 2^c + 2^{b+c-a}$ *for* $0 \leq a < b < c$

5. $2^a + 2^b + 2^c + 2^{b+c-a+1}$ *for* $0 \leq a < b < c$

With this, we can handle the case of $\mathscr{D}^A \cap [0, 2]$ with an argument which is similar to that of Theorem 4.7.1 but slightly more involved:

Theorem 4.7.3. *The order type of $\mathscr{D}^A \cap [0, 2]$ is ω^2, while for $r < 2$, $\mathscr{D}^A \cap [0, r]$ has order type strictly less than ω^2. Furthermore, all defects in $\mathscr{D}^A \cap [0, 2]$ are A-stable, and so the order type of $\mathscr{D}^A_{st} \cap [0, 2]$ is ω^2.*

Proof. Suppose that $\lceil \delta^A(n) \rceil = 2$. Then $\delta^\ell(n) \leq \delta^A(n) \leq 2$, so by Theorem 4.4.5, $\lceil \delta^\ell(n) \rceil = 2$ unless $\nu_2(n) \leq 2$, and this cannot occur, as then we would have $\delta^A(n) \leq 1$. So we can apply Theorem 4.7.2 to conclude that n can be written in one of the forms listed there.

Conversely, suppose we have a number n of one of the forms listed in Theorem 4.7.2. Since $\nu_2(n) > 2$, $\ell^A(n) > \lfloor \log_2 n \rfloor + 1$. And if $\ell^A(n) \geq \lfloor \log_2 n \rfloor + 3$, then

$$\delta^A(n) = \lfloor \log_2 n \rfloor + 3 - \log_2 n > 2.$$

Thus, $\mathscr{D}^A \cap (1, 2]$ is a subset of

$$T := \{2 + \lfloor \log_2 n \rfloor - \log_2 n : n \text{ satisfies the conclusion of Theorem 4.7.2}\}.$$

Let T_i denote the set

$$\{2 + \lfloor \log_2 n \rfloor - \log_2 n : n \text{ falls under case } i \text{ of Theorem 4.7.2}\},$$

so that T is the union of T_1 through T_5. We will examine each of these sets in turn.

The set T_1 is the same as the set S_3 from Proposition 4.6.2, and so has order type ω^2. In fact, if $n = 2^a + 2^b + 2^c$, $c > b > a$, then $\ell^A(n) \leq c + 2$ by the assumption that A is admissible, and so $\ell^A(n) = c + 2$ and $\delta^A(n) < 2$, meaning that all of T_1, rather than just a subset, is contained in $\mathscr{D}^A \cap [0, 2]$. As was noted earlier, we can rewrite S_3 as the set

$$\{2 - \log_2(1 + 2^{-a} + 2^{-b}) : 0 < a < b \in \mathbb{Z}\}.$$

As a and b go to infinity, this expression goes to 2, and so we see that $\sup T_1 = 2$, and thus T_1 must be cofinal in $T \subseteq [0, 2)$.

The set T_2 is easily seen to be equal to the set $\{9 - \log_2 135\}$, which has order type $1 = \omega^0$. This number is also strictly less than 2 and so T_2 is not cofinal in T.

The set T_3 is equal to the set $\{5 - \log_2(9 + 3 \cdot 2^{-a}) : a \geq 2\}$, which is a monotonic image of $\mathbb{N}$ and so has order type ω. It is also bounded above by $5 - \log_2 9 < 2$ and so not cofinal in T.

Finally, we consider the sets T_4 and T_5; we claim that both are order isomorphic to S_3 and hence to ω^2, and both are cofinal in T. We will only explicitly treat the case of T_4, as T_5 is similar. First observe that T_4 is equal to the set

$$\{2 - \log_2(1 + 2^{-a} + 2^{-b} + 2^{-a-b}) : 0 < a < b \in \mathbb{Z}\}.$$

As a and b go to infinity, this expression approaches 2, so T_4 is cofinal in T. To see that it has order type ω^2, consider the map

$$2 - \log_2(1 + 2^{-a} + 2^{-b}) \mapsto 2 - \log_2(1 + 2^{-a} + 2^{-b} + 2^{-a-b})$$

(where here $b > a > 0$). Let $f(a, b)$ denote $2 - \log_2(1 + 2^{-a} + 2^{-b})$ and $g(a, b)$ denote $2 - \log_2(1 + 2^{-a} + 2^{-b} + 2^{-a-b})$. Then it is straightforward to check that $f(a_1, b_1) > f(a_2, b_2)$ if and only if $(a_1, b_1) > (a_2, b_2)$ lexicographically, which also is true if and only if $g(a_1, b_1) > g(a_2, b_2)$. Hence the map above, which sends $f(a, b)$ to $g(a, b)$, is an order isomorphism, proving the claim. As mentioned above, the case of T_5 is similar.

Thus, by Proposition 4.5.4, T has order type ω^2. And so $\mathscr{D}^A \cap (1, 2]$ has order type at most ω^2, and so $\mathscr{D}^A \cap [0, 2]$ has order type at most $\omega + \omega^2 = \omega^2$. We also already know it has order type at least ω^2, so it has order type exactly ω^2.

Also, the supremum of $\mathscr{D}^A \cap [0, 2]$ is 2, so for any $r < 2$, the set $\mathscr{D}^A \cap [0, r]$ is a proper initial segment and so has order type strictly less than ω^2.

Finally, note that if $\lceil \delta^A(n) \rceil = 2$, then n must be A-stable, since otherwise, there would be some k with $\delta^A(2^k n) < 1$; but $\nu_2(n) \geq 3$ and $\nu_2(2^k n) \leq 2$, so this is impossible. By Theorem 4.7.1, all defects in $\mathscr{D}^A \cap [0, 1]$ are stable, and by the above, all defects in $\mathscr{D}^A \cap (1, 2]$ are stable, so the stability part of the theorem follows. □

4.7.3 Summing up: Lower bounds

So we can now sum up the lower bounds on $f^A(k)$ and $f^A_{st}(k)$ as follows:

Theorem 4.7.4. *For k a whole number, we have:*

1. *For $0 \leq k \leq 2$, we have $f^A(k) = f^A_{st}(k) = k$.*
2. *For $3 \leq k \leq 7$, we have $2 < f^A(k) \leq f^A_{st}(k) \leq k$.*
3. *For $8 \leq k \leq 33$, we have $3 < f^A(k) \leq f^A_{st}(k) \leq k$.*
4. *For $k \geq 34$, we have $\log_2(k+1) - C_s < f^A(k) \leq f^A_{st}(k) \leq k$.*

Proof. The upper bounds are just Corollary 4.6.7, so we focus on the lower bounds.

For $k = 0$, this follows as $0 \in \mathscr{D}^\ell$. For $k = 1$, this is immediate from Theorem 4.7.1. For $k = 2$, this is immediate from Theorem 4.7.3. Part (2) then follows as f^A is strictly increasing.

For part (3), observe that by Theorem 4.6.4 and Proposition 4.4.6, the order type of $\mathscr{D}^A \cap [0, 3]$ is less than ω^8, and so $f^A(8) > 3$; the rest then follows as f^A is increasing. Finally, part (4) is just Corollary 4.6.7. □

4.8 Concluding Remarks

In future papers we hope to prove better bounds on $f^\ell(k)$, $f^*(k)$, and their stable versions. Meanwhile we conjecture:

Conjecture 4.8.1. *(1) For $k \geq 0$, $f^\ell(k) = f^\ell_{st}(k) = k$.*
(2) For $k \geq 0$, $f^*(k) = f^*_{st}(k) = k$.

We can say for a fact that there are certain sets of addition chains A for which we know an analogue of Conjecture 4.8.1 holds; we could take A to be the set of addition chains generated by the binary method. Then we would have $\mathscr{D}^A = \mathscr{D}^A_{st} = \bigcup_{k \geq 1} S_k$, where S_k is as in Proposition 4.6.2. It is then easy to check that, for $k \geq 2$, we have $S_k \subseteq (k-2, k-1)$ and then conclude that $f^A(k) = k$. But this example is a triviality and tells us nothing about the structure of addition chains.

So we ask:

Question 4.8.2. *Assuming that Conjecture 4.8.1 holds, what conditions on A are needed to ensure that Conjecture 4.8.1 holds when $\mathscr{D}^A$ is used in place of $\mathscr{D}^\ell$ or $\mathscr{D}^*$? Does it hold when A is the set of Hansen chains, or the set of chains which are star or quasi-star?*

Chapter 5

Integer Complexity: Computational Methods and Results

Abstract: Define $\|n\|$ to be the *complexity* of n, the smallest number of ones needed to write n using an arbitrary combination of addition and multiplication. Define n to be *stable* if for all $k \geq 0$, we have $\|3^k n\| = \|n\| + 3k$. In Chapter 2, we Zelinsky showed that for any n, there exists some $K = K(n)$ such that $3^K n$ is stable; however, the proof there provided no upper bound on $K(n)$ or any way of computing it. In this chapter, we describe an algorithm for computing $K(n)$, and thereby also show that the set of stable numbers is a computable set. The algorithm is based on considering the *defect* of a number, defined by $\delta(n) := \|n\| - 3\log_3 n$, building on the methods presented in Chapter 3. As a side benefit, this algorithm also happens to allow fast evaluation of the complexities of powers of 2; we use it to verify that $\|2^k 3^\ell\| = 2k+3\ell$ whenever k and ℓ are not both zero and $k \leq 48$, providing more evidence for the conjecture that $\|2^k 3^\ell\| = 2k + 3\ell$ whenever k and ℓ are not both zero.

5.1 Introduction

The *complexity* of a natural number n is the least number of 1's needed to write it using any combination of addition and multiplication, with the order of the operations specified using parentheses grouped in any legal nesting. For instance, $n = 11$ has a complexity of 8, since it can be written using 8 ones as $(1+1+1)(1+1+1)+1+1$, but not with any fewer. This notion was implicitly introduced in 1953 by Kurt Mahler and Jan Popken [38]; they actually considered an inverse function, the size of the largest number representable using k copies of the number 1. (More generally, they considered the same question for representations using k copies of a positive real number x.) Integer complexity was explicitly studied by John Selfridge, and was later popularized by Richard Guy [29, 30]. Following J. Arias de Reyna [8] we will denote the complexity of n by $\|n\|$.

Integer complexity is approximately logarithmic; it satisfies the bounds

$$3\log_3 n = \frac{3}{\ln 3}\ln n \leq \|n\| \leq \frac{3}{\ln 2}\ln n, \qquad n > 1.$$

The lower bound can be deduced from the result of Mahler and Popken, and was explicitly proved by John Selfridge [29]. It is attained with equality for $n = 3^k$ for all $k \geq 1$. The upper bound can be obtained by writing n in binary and finding a representation using Horner's algorithm. It is not sharp, and the constant $\frac{3}{\ln 2}$ can be improved for large n [52].

One can compute $\|n\|$ via dynamic programming, since $\|1\| = 1$, and for $n > 1$, one has

$$\|n\| = \min_{\substack{a,b<n\in\mathbb{N} \\ a+b=n \text{ or } ab=n}} \|a\| + \|b\|.$$

This yields an algorithm for computing $\|n\|$ that runs in time $\Theta(n^2)$; in the multiplication case, one needs to check $a \leq \sqrt{n}$, and, naïvely, in the addition case, one needs to check $a \leq n/2$. However, Srinivas and Shankar[44] showed that the upper bound on the addition case case be improved, lowering the time required to $O(n^{\log_2 3})$, by taking advantage of the inequality $\|n\| \geq 3\log_3 n$ to rule out cases when a is too large. Arias de Reyna and Van de Lune[9] took this further and showed that it could be computed in time $O(n^\alpha)$, where

$$\alpha = \frac{\ln(3^6 2^{-10}(30557189 + 21079056\sqrt[3]{3} + 14571397\sqrt[3]{9}))}{\ln(2^{10}3^7)} < 1.231;$$

this remains the best known algorithm for computing $\|n\|$ for general n.

The notion of integer complexity is similar in spirit but different in detail from the better known measure of *addition chain length*, which has application to computation of powers, and which is discussed in detail in Knuth [35, Sect. 4.6.3]. See also Chapter 4 for some interesting analogies between them; we will discuss this further in Section 5.1.3.

An obvious question about $\|n\|$ is that of the complexity of powers. For $k \geq 1$ it is true that

$$\|n^k\| \leq k\|n\|,$$

and for the case of $n = 3$ we even have equality, i.e., it is known that $\|3^k\| = 3k$ for all $k \geq 1$. However other values have a more complicated behavior. For instance, powers of 5 do not work nicely, as $\|5^6\| = 29 < 30 = 6 \cdot \|5\|$. The behavior of powers

of 2 remains unknown; it has previously been verified [33] that

$$\|2^k\| = k\|2\| = 2k \text{ for } 1 \leq k \leq 39.$$

One may combine the known fact that $\|3^k\| = 3k$ for $k \geq 1$, and the hope that $\|2^k\| = 2k$ for $k \geq 1$, into the following conjecture:

Conjecture 5.1.1. *For $k, \ell \geq 0$ and not both equal to 0,*

$$\|2^k 3^\ell\| = 2k + 3\ell.$$

Such a conjecture is quite far from being proven; after all, it would require that $\|2^k\| = 2k$ for all $k \geq 1$, which would in turn imply that

$$\limsup_{n\to\infty} \frac{\|n\|}{\ln n} \geq \frac{2}{\ln 2};$$

at present, it is not even known that this limit is any greater than $\frac{3}{\ln 3}$, i.e., that $\|n\| \nsim 3\log_3 n$. Indeed, some have suggested that $\|n\|$ may indeed just be asymptotic to $3\log_3 n$; see [29].

Nonetheless, in this chapter we provide some more evidence for this conjecture, by proving:

Theorem 5.1.2. *For $k \leq 48$, and k and ℓ not both zero,*

$$\|2^k 3^\ell\| = 2k + 3\ell.$$

This extends the results of [33] regarding numbers of the form $2^k 3^\ell$, as well as the results of Chapter 2, which showed this for $k \leq 21$. We prove this not by careful hand analysis, as was done in Chapter 2, but by demonstrating, based on the methods of Chapters 2 and 3, a new algorithm (Algorithm 10) for computing $\|2^k\|$ – which not only runs much faster than existing algorithms, but is also capable of in addition telling us whether or not, for the given k, $\|2^k 3^\ell\| = 2k + 3\ell$ holds for all $\ell \geq 0$. Before we go into its workings, however, let us take a moment to say a bit more about just what it is that it does.

5.1.1 Stability and main result

The fact that $\|3^k\| = 3k$ holds for all $k \geq 1$ might prompt one to ask whether in general it is true that $\|3n\| = \|n\| + 3$. This is false for $n = 1$, but it does not seem

an unreasonable guess for $n > 1$. Nonetheless, this does not hold; the next smallest counterexample is $n = 107$, where $\|107\| = 16$ but $\|321\| = 18$. Indeed, not only do there exist n for which $\|3n\| < \|n\| + 3$, there are even n for which $\|3n\| < \|n\|$; one example is $n = 4721323$. Still, this guess can be rescued. Let us first make a definition:

Definition 5.1.3. A number m is called *stable* if $\|3^k m\| = 3k + \|m\|$ holds for every $k \geq 0$. Otherwise it is called *unstable*.

In Chapter 2, we showed:

Theorem 5.1.4. *For any natural number n, there exists $K \geq 0$ such that $3^K n$ is stable. That is to say, there exists a minimal $K := K(n)$ such that for any $k \geq K$,*

$$\|3^k n\| = 3(k - K) + \|3^K n\|.$$

This can be seen as a "rescue" of the incorrect guess that $\|3n\| = \|n\| + 3$ always. With this theorem, it makes sense to define:

Definition 5.1.5. Given $n \in \mathbb{N}$, define $K(n)$, the *stabilization length* of n, to be the smallest k such that $3^k n$ is stable.

We can also define the notion of the *stable complexity* of n (see Chapter 3), which is, intuitively, what the complexity of n would be "if n were stable":

Definition 5.1.6. For a positive integer n, we define the *stable complexity* of n, denoted $\|n\|_{st}$, to be $\|3^k n\| - 3k$ for any k such that $3^k n$ is stable. This is well-defined; if $3^k n$ and $3^\ell n$ are both stable, say with $k \leq \ell$, then

$$\|3^k n\| - 3k = 3(k - \ell) + \|3^\ell n\| - 3k = \|3^\ell n\| - 3\ell.$$

Unfortunately, Chapter 2, while proving the existence of $K(n)$, gave no upper bound on $K(n)$ or indeed any way of computing it. Certainly one cannot compute whether or not n is stable simply by computing for all k the complexity of $3^k n$; one can guarantee that n is unstable by such computations, but never that it is stable. And it's not clear that $\|n\|_{st}$, though it has been a useful object of study in Chapter 3, can actually be computed. But in this chapter we prove:

Theorem 5.1.7. *We have:*

1. *The function $K(n)$, the stabilization length of n, is a computable function of n.*

2. *The function $\|n\|_{st}$, the stable complexity of n, is a computable function of n.*

3. *The set of stable numbers is a computable set.*

It's worth observing here that, strictly speaking, all three parts of this theorem are equivalent. If one has an algorithm for computing $K(n)$, then one may check whether n is stable by checking whether $K(n) = 0$, and one may compute $\|n\|_{st}$ by computing $\|3^{K(n)}n\|$ by the usual methods and observing that

$$\|n\|_{st} = \|3^{K(n)}n\| - 3K(n).$$

Similarly, if one has an algorithm for computing $\|n\|_{st}$, one may compute whether n is stable by checking if $\|n\|_{st} = \|n\|$. Finally, if one has an algorithm for telling if n is stable, one may determine $K(n)$ by simply applying this algorithm to n, $3n$, $9n$, ..., until it returns a positive result, which must eventually occur. Such methods for converting between $K(n)$ and $\|n\|_{st}$ may be quite slow, however. Fortunately, the algorithm described here (Algorithm 8) will yield both $K(n)$ and $\|n\|_{st}$ at once, averting such issues; and if one has $K(n)$, checking whether n is stable is a one-step process.

5.1.2 The defect, low-defect polynomials, truncation, and the algorithm

Let us now turn our attention to the inner workings of these algorithms. Proving the statement $\|n\| = k$ has two parts; showing that $\|n\| \leq k$, and showing that $\|n\| \geq k$. The former is, comparatively, the easy part, as it consists of just finding an expression for n that uses at most k ones; the latter requires ruling out shorter expressions. The simplest method for this is simply exhaustive search, which, as has been mentioned, takes time $\Theta(n^2)$, or time $O(n^{1.24625})$ once some possibilities have been eliminated from the addition case.

In this chapter, we take a different approach to lower bounding the quantity $\|n\|$, one used earlier in Chapter 2; however, we make a number of improvements to the method of Chapter 2 that both turn this method into an actual algorithm, and frequently allow it to run in a reasonable time. The method is based on considering the *defect* of n:

Definition 5.1.8. The *defect* of n, denoted $\delta(n)$ is defined by

$$\delta(n) := \|n\| - 3\log_3 n.$$

Let us further define:

Definition 5.1.9. For a real number $s \geq 0$, the set A_s is the set of all natural numbers with defect less than s.

Chapter 2 provided a method of, for any choice of $\alpha \in (0, 1)$, recursively building up covering sets for the sets $A_\alpha, A_{2\alpha}, A_{3\alpha}, \ldots$; then, if for some n and k we can use this to demonstrate that $n \notin A_{k\alpha}$, then we have determined a lower bound on $\|n\|$.

Chapter 3 improved on this by showing that the covering sets generated this way have a tractable form. It showed that for any $s \geq 0$, there is a finite set $\mathcal{S}_s$ of multilinear polynomials, of a particular form called *low-defect polynomials*, such that if $\delta(n) < s$ then n can be written as $f(3^{k_1}, \ldots, 3^{k_r})3^{k_{r+1}}$ for some $f \in \mathcal{S}_s$ and some $k_1, \ldots, k_{r+1} \geq 0$. However, extraneous numbers could also be generated this way; not every number represented this way would necessarily have defect less than s. This makes it harder to demonstrate that $n \notin A_{k\alpha}$. (It is technically possible to prove Theorem 5.1.7 based only on the methods of Chapter 3, without the new methods here; however, empirically, the algorithm obtained this way is too slow to be practical, and the new methods here are of independent interest regardless. See Section 5.1.3 and Appendix C.)

The innovation in this chapter is that we introduce a way of "truncating" a low-defect polynomial f to a given defect s, though this replaces the one polynomial f by a finite set of low-defect polynomials $\{g_1, \ldots, g_k\}$. If we truncate every polynomial in the set $\mathcal{S}_s$ to the defect s, we obtain a set $\mathcal{T}_s$ of low-defect polynomials so that for any natural number n, $\delta(n) < s$ if and only if $n = f(3^{k_1}, \ldots, 3^{k_r})3^{k_{r+1}}$ for some $f \in \mathcal{T}_s$ and some $k_1, \ldots, k_{r+1}$. That is to say, we are no longer merely covering the set A_r, but representing it exactly. Indeed, stronger statements are true; see Theorem 5.5.9. This remedies many of the deficiencies of attempting to apply the methods of Chapter 3 directly, and also leads to algorithms which can be applied in practice (e.g., to prove Theorem 5.1.2).

In brief, the algorithm works as follows: First, we choose a step size α. We start with a set of low-defect polynomials representing A_α, and apply the method of Chapter 3 to build up sets representing $A_{2\alpha}, A_{3\alpha}, \ldots$; at each step, we use truncation to ensure we are representing the set $A_{i\alpha}$ exactly and not including extraneous elements. Then we check whether or not $n \in A_{i\alpha}$; if it is not, we continue on to $A_{(i+1)\alpha}$. If it is, then we have a representation $n = f(3^{k_1}, \ldots, 3^{k_r})3^{k_{r+1}}$, and this gives us an upper bound on $\|n\|$ – indeed, by Theorem 5.5.9, we can find a shortest representation for n in this way, and so it gives us $\|n\|$ exactly.

This is, strictly speaking, a little different than what was described above, in that it does not involve directly getting a lower bound on $\|n\|$ from the fact that $n \notin A_{i\alpha}$. However, this can be used too, so long as we know in advance an upper bound on $\|n\|$. For instance, this is quite useful when $n = 2^k$ (for $k \geq 1$), as then we know that $\|n\| \leq 2k$, and hence that $\delta(n) \leq k\delta(2)$. So we can use the method of the above paragraph, but stop early, once we have covered defects up to $k\delta(2) - 1$. If we get a hit within that time, then we have found a shortest representation for $n = 2^k$. Conversely, if n is not detected, then we know that we must have

$$\delta(2^k) > k\delta(2) - 1,$$

and hence that

$$\|2^k\| > 2k - 1,$$

i.e., $\|2^k\| = 2k$, thus verifying that the obvious representation is the best possible. Again, though we have illustrated it here with powers of 2, this method can be used whenever we know in advance an upper bound on $\|n\|$; see Appendix C.

Now, so far we've discussed using these methods to compute $\|n\|$, but the more interesting application is Theorem 5.1.7, i.e., using them to compute $K(n)$ and $\|n\|_{st}$. In this case, at each step, instead of checking whether there is some $f \in \mathcal{T}_{i\alpha}$ such that $n = f(3^{k_1}, \ldots, 3^{k_r})3^{k_{r+1}}$, we check whether is some $f \in \mathcal{T}_{i\alpha}$ and some ℓ such that

$$3^\ell n = f(3^{k_1}, \ldots, 3^{k_r})3^{k_{r+1}}.$$

It is not immediately obvious that this is possible, since naïvely we would need to check arbitrarily large ℓ, but Lemma 5.7.1 allows us to do this while checking only finitely many ℓ. Once we have such a detection, we can use the value of ℓ to determine $K(n)$, and the representation of 3^ℓ obtained this way to determine $\|3^{K(n)}n\|$ and hence $\|n\|_{st}$. In addition, if we know in advance $\|n\| \leq k$, we can use the same trick as above to sometimes cut the computation short and conclude not only that $\|n\| = k$ but also that n is stable.

The algorithms here can be used for more purposes as well; see Theorem 5.8.2 for a further application of them.

5.1.3 Discussion

Many of the algorithms described here are parameterized, in that they require a choice of a "step size" $\alpha \in (0, 1)$. In the author's implementation, α is always taken

to be $\delta(2) = 0.107\ldots$, and some precomputations have been made based on this choice. See Appendix C for more on this. Below, when we discuss the computational complexity of the algorithms given here, we are assuming a fixed choice of α. It is possible that the value of α affects the time complexity of these algorithms. One could also consider what happens when α is considered as an input to the algorithm, so that one cannot do pre-computations based on the choice of α. (In this case we should really restrict the form of α so that the question makes sense, for instance to $\alpha = p - q \log_3 n$, with n a natural number and $p, q \in \mathbb{Q}$.) But we will avoid these issues for now, and assume for the rest of this section that $\alpha = \delta(2)$ unless otherwise specified. Two of the algorithms here also require first choosing an upper bound L on the input $\|n\|$, which may be ∞. We will assume here the simplest case, where we always pick $L = \infty$.

Note that we do not actually conduct here a formal analysis of the time complexity of Algorithm 8 or Algorithm 10. The statement that Algorithm 10 is much faster than existing methods for computing $\|2^k\|$ is an empirical one. The speedup is a dramatic one, though; for instance, J. Iraids's computation of $\|10^{12}\|$ required about 3 weeks on a supercomputer, although he used the $\Theta(n^2)$-time algorithm rather than any of the improvements [32]; whereas computing $\|2^{48}\|$ via Algorithm 10 required only around 20 hours on the author's laptop computer. Of course, Iraids did not just compute $\|10^{12}\|$, but in fact computed $\|n\|$ for all $n \leq 10^{12}$; but the comparison is not an unfair one, as all previous methods for computing $\|n\|$ require computing $\|m\|$ for all $m \leq n$.

Still, it is worth noting that, empirically, it seems that increasing k by one approximately doubles the run time of Algorithm 10. This suggests that perhaps Algorithm 10 runs in time $O(2^k)$, which would be better than the $O(2^{1.231k})$ bound coming from applying existing methods [9] to compute the complexity of $\|2^k\|$.

Note that for Algorithm 8, the run time seems to be determined more by the size of $\delta_{st}(n)$ (or $\delta(n)$ for Algorithm 9), rather than the size of n, since it seems that most of the work is in building the sets of low-defect polynomials, rather than checking if n is 3-represented. For this reason, computing $\|n\|$ via Algorithm 9 is frequently much slower than using existing methods, even though it is much faster for powers of 2. Note that strictly speaking, $\delta(n)$ can be bounded in terms of n, since

$$\delta(n) \leq 3 \log_2 n - 3 \log_3 n,$$

but as mentioned earlier, this may be a substantial overestimate. So it is worth asking the question:

Question 5.1.10. *What is the time complexity of Algorithm 10, for computing $K(2^k)$ and $\|2^k\|_{st}$? Of Algorithm 8 (with $L = \infty$), for computing $K(n)$ and $\|n\|_{st}$? Of Algorithm 9 (with $L = \infty$), for computing the values of $\|3^k n\|$ for a given n and all $k \geq 0$? What if L may be finite? How do these depend on the parameter α? What if α is an input?*

Further worth noting is that although we have now given a means to compute $K(n)$, we have not provided any explicit upper bound on it. The same is true for the quantity

$$\Delta(n) := \|n\| - \|n\|_{st},$$

which is another way of measuring "how unstable" the number n is, and which is also now computable due to Theorem 5.1.7. Nor do we have any reliable method of generating unstable numbers with which to demonstrate lower bounds.

Indeed, empirically, large instabilities – measured either by $K(n)$ or by $\Delta(n)$ – seem to be rare. Be warned, this statement is not based on running Algorithm 8 on many numbers to determine their stability, as that is quite slow in general, but rather on simply computing $\|n\|$ for $n \leq 3^{15}$ and then checking $\|n\|$, $\|3n\|$, $\|9n\|$,..., and guessing that n is stable if no instability is detected before the data runs out, a method that can only ever put lower bounds on $K(n)$ and $\Delta(n)$, never upper bounds. Still, numbers that are detectably unstable at all seem to be somewhat rare, although they still seem to make up a positive fraction of all natural numbers; namely, around 3%. Numbers that are more than merely unstable – having $K(n) \geq 2$ or $\Delta(n) \geq 2$ – are yet rarer.

The largest lower bounds on $K(n)$ or $\Delta(n)$ for a given n encountered based on these computations are $n = 4721323$, which, as mentioned earlier, has $\|3n\| < \|n\|$ and thus $\Delta(n) \geq 4$; and 17 numbers, the smallest of which is $n = 3643$, which have $\|3^5 n\| < \|3^4 n\| + 3$ and thus $K(n) \geq 5$. Finding n where both $K(n)$ and $\Delta(n)$ are decently large is hard; for instance, these computations did not turn up any n for which it could be seen that both $K(n) \geq 3$ and $\Delta(n) \geq 3$. (See Table 5.1 for more.) It's not even clear whether $K(n)$ or $\Delta(n)$ can get arbitrarily large, or are bounded by some finite constant, although there's no clear reason why the latter would be so. Still, this is worth pointing out as a question:

Question 5.1.11. *What is the natural density of the set of unstable numbers? What is an explicit upper bound on $K(n)$, or on $\Delta(n)$? Can $K(n)$ and $\Delta(n)$ get arbitrarily large, or are they bounded?*

Table 5.1: Numbers that seem to have unusual drop patterns. Here, the "drop pattern" of n is the list of values $\delta(3^k n) - \delta(3^{k+1} n)$, or equivalently $\|3^k n\| - \|3^{k+1} n\| + 3$, up until the point where this is always zero. This table is empirical, based on a computation of $\|n\|$ for $n \leq 3^{15}$; it's possible these numbers have later drops further on. Numbers which are divisible by 3 are not listed.

Drop pattern	Numbers with this pattern
4	4721323
$1, 2$	1081079
$2, 1$	203999, 1328219
$1, 0, 0, 1$	$153071, 169199$

Further questions along these lines suggest themselves, but these questions seem difficult enough, so we will stop this line of inquiry there for now.

It is worth noting here that, strictly speaking, it is possible to prove Theorem 5.1.7 using algorithms based purely on the methods of Chapter 3, without actually using this chapter's main innovation, the method of truncation. Of course, one cannot simply remove the truncation step from the algorithms here and get correct answers; other checks are necessary to compensate. See Appendix C for a brief discussion of this. However, while this is sufficient to prove Theorem 5.1.7, the algorithms obtained this way are simply too slow to be of use in practice. And without the method of truncation, one cannot prove Theorem 5.5.9 or write Algorithm 6, which are interesting and useful in their own right. For instance, proving Theorem 5.8.2 would be quite difficult without Algorithm 6. We will demonstrate further applications of the Theorem 5.8.2 and the method of truncation in future papers [4, 6].

We can also ask about the computational complexity of computing these functions in general, rather than just the specific algorithms here. As noted above, the best known algorithm for computing $\|n\|$ takes time $O(n^{1.24625})$. It is also known[8] that the problem "Given n and k in binary, is $\|n\| \leq k$?" is in the class NP, because the size of a witness is $O(\log n)$. (This problem is not known to be NP-complete.) However, it's not clear whether the problem "Given n and k in binary, is $\|n\|_{st} \leq k$?" is in the class NP, because there's no obvious bound on the size of a witness. It is quite possible that it could be proven to be in NP, however, if an explicit upper bound could be obtained on $K(n)$.

We can also consider the problem of computing the defect ordering, i.e., "Given n_1 and n_2 in binary, is $\delta(n_1) \leq \delta(n_2)$?"; the significance of this problem is that the set of all defects is in fact a well-ordered set with order type ω^ω, as detailed in Chapter 3.

This problem lies in Δ_2^P in the polynomial hierarchy (see Section 3.1.4. In Chapter 3 we also defined the *stable defect* of n:

Definition 5.1.12. The *stable defect* of n, denoted $\delta_{st}(n)$, is

$$\delta_{st}(n) := \|n\|_{st} - 3\log_3 n.$$

(We will review the stable defect and its properties in Section 5.2.1.) Thus we get the problem of, "Given n_1 and n_2 in binary, is $\delta_{st}(n_1) \leq \delta_{st}(n_2)$?" The image of δ_{st} is also well-ordered with order type ω^ω, but until now it was not known that this problem is computable. But Theorem 5.1.7 shows that it is, and so we can ask about its complexity. Again, due to a lack of bounds on $K(n)$, it's not clear that this lies in Δ_2^P.

We can also ask about the complexity of computing $K(n)$, or $\Delta(n)$ (which, conceivably, could be easier than $\|n\|$ or $\|n\|_{st}$, though this seems unlikely), or, perhaps most importantly, of computing a set $\mathcal{T}_s$ for a given $s \geq 0$. Note that in this last case, it need not be the set $\mathcal{T}_s$ found by Algorithm 6 here; we just want any set satisfying the conclusions of Theorem 5.5.9 – a *good covering* of B_s, as we call it here (see Definition 5.5.10). Of course, we must make a restriction on the input for this last question, as one cannot actually take arbitrary real numbers as input; perhaps it would be appropriate to restrict to s of the form

$$s \in \{p - q\log_3 n : p, q \in \mathbb{Q}, n \in \mathbb{N}\}.$$

We summarize:

Question 5.1.13. *What is the complexity of computing $\|n\|$? Of $\|n\|_{st}$? Of the difference $\Delta(n)$? Of the defect ordering $\delta(n_1) \leq \delta(n_2)$? Of the stable defect ordering $\delta_{st}(n_1) \leq \delta_{st}(n_2)$? Of the stabilization length $K(n)$?*

Question 5.1.14. *Given $s = p - q\log_3 n$, with $p, q \in \mathbb{Q}$ and $n \in \mathbb{N}$, what is the complexity of computing a good covering $\mathcal{T}_s$ of B_s?*

5.2 The defect, stability, and low-defect polynomials

In this section we review the results of Chapters 2 and 3 regarding the defect $\delta(n)$, the stable complexity $\|n\|_{st}$ and stable defect $\delta_{st}(n)$, and low-defect polynomials.

5.2.1 The defect and stability

First, some basic facts about the defect:

Theorem 5.2.1. *We have:*

1. *For all n, $\delta(n) \geq 0$.*
2. *For $k \geq 0$, $\delta(3^k n) \leq \delta(n)$, with equality if and only if $\|3^k n\| = 3k + \|n\|$. The difference $\delta(n) - \delta(3^k n)$ is a nonnegative integer.*
3. *A number n is stable if and only if for any $k \geq 0$, $\delta(3^k n) = \delta(n)$.*
4. *If the difference $\delta(n) - \delta(m)$ is rational, then $n = m3^k$ for some integer k (and so $\delta(n) - \delta(m) \in \mathbb{Z}$).*
5. *Given any n, there exists k such that $3^k n$ is stable.*
6. *For a given defect α, the set $\{m : \delta(m) = \alpha\}$ has either the form $\{n3^k : 0 \leq k \leq L\}$ for some n and L, or the form $\{n3^k : 0 \leq k\}$ for some n. This latter occurs if and only if α is the smallest defect among $\delta(3^k n)$ for $k \in \mathbb{Z}$.*
7. *If $\delta(n) = \delta(m)$, then $\|n\| = \|m\| \pmod 3$.*
8. *$\delta(1) = 1$, and for $k \geq 1$, $\delta(3^k) = 0$. No other integers occur as $\delta(n)$ for any n.*
9. *If $\delta(n) = \delta(m)$ and n is stable, then so is m.*

Proof. Parts (1) through (8), excepting part (3), are just Theorem 3.2.1. Part (3) is Proposition 2.2.4, and part (9) is Proposition 3.3.1. □

Also, although it will not be a focus of this chapter, we will sometimes want to consider the set of all defects:

Definition 5.2.2. We define the *defect set* $\mathscr{D}$ to be $\{\delta(n) : n \in \mathbb{N}\}$, the set of all defects.

In Chapter 3 we also defined the notion of a *stable defect*:

Definition 5.2.3. We define a *stable defect* to be the defect of a stable number.

Because of part (9) of Theorem 5.2.1, this definition makes sense; a stable defect α is not just one that is the defect of some stable number, but one for which any n with $\delta(n) = \alpha$ is stable. Stable defects can also be characterized by the following proposition from Chapter 3:

Proposition 5.2.4. *A defect α is stable if and only if it is the smallest $\beta \in \mathscr{D}$ such that $\beta \equiv \alpha \pmod 1$.*

We can also define the *stable defect* of a given number, which we denote $\delta_{st}(n)$. (We actually already defined this in Definition 5.1.12, but let us disregard that for now and give a different definition; we will see momentarily that they are equivalent.)

Definition 5.2.5. For a positive integer n, define the *stable defect* of n, denoted $\delta_{st}(n)$, to be $\delta(3^k n)$ for any k such that $3^k n$ is stable. (This is well-defined as if $3^k n$ and $3^\ell n$ are stable, then $k \geq \ell$ implies $\delta(3^k n) = \delta(3^\ell n)$, and so does $\ell \geq k$.)

Note that the statement "α is a stable defect", which earlier we were thinking of as "$\alpha = \delta(n)$ for some stable n", can also be read as the equivalent statement "$\alpha = \delta_{st}(n)$ for some n".

We then have the following facts relating the notions of $\|n\|$, $\delta(n)$, $\|n\|_{st}$, and $\delta_{st}(n)$:

Proposition 5.2.6. *We have:*

1. $\delta_{st}(n) = \min_{k \geq 0} \delta(3^k n)$
2. $\delta_{st}(n)$ *is the smallest* $\alpha \in \mathscr{D}$ *such that* $\alpha \equiv \delta(n) \pmod 1$.
3. $\|n\|_{st} = \min_{k \geq 0}(\|3^k n\| - 3k)$
4. $\delta_{st}(n) = \|n\|_{st} - 3 \log_3 n$
5. $\delta_{st}(n) \leq \delta(n)$, *with equality if and only if* n *is stable.*
6. $\|n\|_{st} \leq \|n\|$, *with equality if and only if* n *is stable.*

Proof. These are just Propositions 3.3.5, 3.3.7, and 3.3.8. □

5.2.2 Low-defect polynomials and low-defect pairs

As has been mentioned in Section 5.1.2, we are going to represent the set A_r by substituting in powers of 3 into certain multilinear polynomials we call *low-defect polynomials*. We will associate with each one a "base complexity" to from a *low-defect pair*. In this section we will review the basic properties of these polynomials. First, their definition:

Definition 5.2.7. We define the set $\mathscr{P}$ of *low-defect pairs* as the smallest subset of $\mathbb{Z}[x_1, x_2, \ldots] \times \mathbb{N}$ such that:

1. For any constant polynomial $k \in \mathbb{N} \subseteq \mathbb{Z}[x_1, x_2, \ldots]$ and any $C \geq \|k\|$, we have $(k, C) \in \mathscr{P}$.

2. Given (f_1, C_1) and (f_2, C_2) in $\mathscr{P}$, we have $(f_1 \otimes f_2, C_1 + C_2) \in \mathscr{P}$, where, if f_1 is in r_1 variables and f_2 is in r_2 variables,

$$(f_1 \otimes f_2)(x_1, \ldots, x_{r_1+r_2}) := f_1(x_1, \ldots, x_{r_1}) f_2(x_{r_1+1}, \ldots, x_{r_1+r_2}).$$

3. Given $(f, C) \in \mathscr{P}$, $c \in \mathbb{N}$, and $D \geq \|c\|$, we have $(f \otimes x_1 + c, C + D) \in \mathscr{P}$ where $\otimes$ is as above.

The polynomials obtained this way will be referred to as *low-defect polynomials*. If (f, C) is a low-defect pair, C will be called its *base complexity*. If f is a low-defect polynomial, we will define its *absolute base complexity*, denoted $\|f\|$, to be the smallest C such that (f, C) is a low-defect pair. We will also associate to a low-defect polynomial f the *augmented low-defect polynomial*

$$\hat{f} = f \otimes x_1$$

Note that the degree of a low-defect polynomial is also equal to the number of variables it uses; see Proposition 5.2.8. We will often refer to the "degree" of a low-defect pair (f, C); this refers to the degree of f. Also note that augmented low-defect polynomials are never low-defect polynomials; as we will see in a moment (Proposition 5.2.8), low-defect polynomials always have nonzero constant term, whereas augmented low-defect polynomials always have zero constant term.

Note that we do not really care about what variables a low-defect polynomial (or pair) is in – if we permute the variables of a low-defect polynomial or replace them with others, we will still regard the result as a low-defect polynomial. From this perspective, the meaning of $f \otimes g$ could be simply regarded as "relabel the variables of f and g so that they do not share any, then multiply f and g". Helpfully, the $\otimes$ operator is associative not only with this more abstract way of thinking about it, but also in the concrete way it was defined above.

In Chapter 3 we proved the following propositions about low-defect pairs:

Proposition 5.2.8. *Suppose f is a low-defect polynomial of degree r. Then f is a polynomial in the variables $x_1, \ldots, x_r$, and it is a multilinear polynomial, i.e., it has degree 1 in each of its variables. The coefficients are non-negative integers. The constant term is nonzero, and so is the coefficient of $x_1 \ldots x_r$, which we will call the* leading coefficient *of f.*

Proposition 5.2.9. *If (f, C) is a low-defect pair of degree r, then*

$$\|f(3^{n_1}, \ldots, 3^{n_r})\| \leq C + 3(n_1 + \ldots + n_r).$$

and

$$\|\hat{f}(3^{n_1}, \ldots, 3^{n_{r+1}})\| \leq C + 3(n_1 + \ldots + n_{r+1}).$$

Proof. This is a combination of Proposition 3.4.5 and Corollary 3.4.12. □

Because of this, it makes sense to define:

Definition 5.2.10. Given a low-defect pair (f, C) (say of degree r) and a number N, we will say that (f, C) *efficiently* 3*-represents* N if there exist nonnegative integers $n_1, \ldots, n_r$ such that

$$N = f(3^{n_1}, \ldots, 3^{n_r}) \text{ and } \|N\| = C + 3(n_1 + \ldots + n_r).$$

We will say $(\hat{f}, C)$ efficiently 3-represents N if there exist $n_1, \ldots, n_{r+1}$ such that

$$N = \hat{f}(3^{n_1}, \ldots, 3^{n_{r+1}}) \text{ and } \|N\| = C + 3(n_1 + \ldots + n_{r+1}).$$

More generally, we will also say f 3-represents N if there exist nonnegative integers $n_1, \ldots, n_r$ such that $N = f(3^{n_1}, \ldots, 3^{n_r})$. and similarly with $\hat{f}$.

Note that if (f, C) (or $(\hat{f}, C)$) efficiently 3-represents N, then $(f, \|f\|)$ (respectively, $(\hat{f}, \|f\|)$ efficiently 3-represents N, which means that in order for (f, C) (or $(\hat{f}, C)$ to 3-represent anything efficiently at all, we must have $C = \|f\|$. However it is still worth using low-defect pairs rather than just low-defect polynomials since we may not always know $\|f\|$. In our applications here, where we want to compute things, taking the time to compute $\|f\|$, rather than just making do with an upper bound, may not be desirable.

For this reason it makes sense to use "f efficiently 3-represents N" to mean "some (f, C) efficiently 3-represents N" or equivalently "$(f, \|f\|)$ efficiently 3-reperesents N". Similarly with $\hat{f}$.

In keeping with the name, numbers 3-represented by low-defect polynomials, or their augmented versions, have bounded defect. Let us make some definitions first:

Definition 5.2.11. Given a low-defect pair (f, C), we define $\delta(f, C)$, the defect of (f, C), to be $C - 3\log_3 a$, where a is the leading coefficient of f. When we are not concerned with keeping track of base complexities, we will use $\delta(f)$ to mean $\delta(f, \|f\|)$.

Definition 5.2.12. Given a low-defect pair (f, C) of degree r, we define

$$\delta_{f,C}(n_1, \ldots, n_r) = C + 3(n_1 + \ldots + n_r) - 3\log_3 f(3^{n_1}, \ldots, 3^{n_r}).$$

Then we have:

Proposition 5.2.13. *Let (f, C) be a low-defect pair of degree r, and let $n_1, \ldots, n_{r+1}$ be nonnegative integers.*

1. *We have*
$$\delta(\hat{f}(3^{n_1}, \ldots, 3^{n_{r+1}})) \leq \delta_{f,C}(n_1, \ldots, n_r)$$
and the difference is an integer.

2. *We have*
$$\delta_{f,C}(n_1, \ldots, n_r) \leq \delta(f, C)$$
and if $r \geq 1$, this inequality is strict.

Proof. This is a combination of Proposition 3.4.9 and Corollary 3.4.14. □

In fact, not only is $\delta(f, C)$ an upper bound on the values of $\delta_{f,C}$, it is the least upper bound:

Proposition 5.2.14. *Let (f, C) be a low-defect pair, say of degree r. Then $\delta_{f,C}$ is a strictly increasing function in each variable, and*

$$\delta(f, C) = \sup_{k_1, \ldots, k_r} \delta_{f,C}(k_1, \ldots, k_r).$$

Proof. We can define g, the reverse polynomial of f:

$$g(x_1, \ldots, x_r) = x_1 \ldots x_r f(x_1^{-1}, \ldots, x_r^{-1}).$$

So g is a multilinear polynomial in $x_1, \ldots, x_r$, with the coefficient of $\prod_{i \in S} x_i$ in g being the coefficient of $\prod_{i \notin S} x_i$ in f. By Proposition 5.2.8, f has nonnegative coefficients, so so does g; since the constant term of f does not vanish, the $x_1 \ldots x_r$ term of g does not vanish. Hence g is strictly increasing in each variable.

Then

$$\begin{aligned}\delta_{f,C}(k_1, \ldots, k_r) &= C + 3(k_1 + \ldots + k_r) - 3\log_3 f(3^{k_1}, \ldots, 3^{k_r}) \\ &= C - 3\log_3 \frac{f(3^{k_1}, \ldots, 3^{k_r})}{3^{k_1 + \ldots + k_r}} = C - 3\log_3 g(3^{-k_1}, \ldots, 3^{-k_r})\end{aligned}$$

which is strictly increasing in each variable, as claimed. Furthermore, if a is the leading coefficient of f, then it is also the constant term of g, and so

$$\inf_{k_1,\ldots,k_r} g(3^{-k_1},\ldots,3^{-k_r}) = a.$$

Thus

$$\sup_{k_1,\ldots,k_r} \delta_{f,C}(k_1,\ldots,k_r) = C - 3\log_3 a = \delta(f,C).$$

□

With this, we have the basic properties of low-defect polynomials.

5.2.3 Inductively building up numbers of small defect

Now let us discuss the "building-up" method from Chapters 2 and 3 that forms one-half the core of the algorithm. The new "filtering-down" half, truncation, will have to wait for Section 5.5.

First, we will need the idea of a *leader*:

Definition 5.2.15. A natural number n is called a *leader* if it is the smallest number with a given defect. By part (6) of Theorem 5.2.1, this is equivalent to saying that either $3 \nmid n$, or, if $3 \mid n$, then $\delta(n) < \delta(n/3)$, i.e., $\|n\| < 3 + \|n/3\|$.

Let us also define:

Definition 5.2.16. For any real $r \geq 0$, define the set of *r-defect numbers* A_r to be

$$A_r := \{n \in \mathbb{N} : \delta(n) < r\}.$$

Define the set of *r-defect leaders* B_r to be

$$B_r := \{n \in A_r : \ n \text{ is a leader}\}.$$

These sets are related by the following proposition from Chapter 3:

Proposition 5.2.17. *For every $n \in A_r$, there exists a unique $m \in B_r$ and $k \geq 0$ such that $n = 3^k m$ and $\delta(n) = \delta(m)$; then $\|n\| = \|m\| + 3k$.*

Because of this, we can focus on building up B_r, and derive A_r from it.

In order to inductively build up the sets A_r and B_r, we need a base case. Fortunately, Chapter 2 provides one in the form of the following theorem, determining all numbers with defect less than 1 and their complexities:

Theorem 5.2.18. *For every α with $0 < \alpha < 1$, the set of leaders B_α is a finite set. More specifically, the list of n with $\delta(n) < 1$ is as follows:*

1. 3^ℓ *for* $\ell \geq 1$, *of complexity* 3ℓ *and defect* 0
2. $2^k 3^\ell$ *for* $1 \leq k \leq 9$, *of complexity* $2k + 3\ell$ *and defect* $k\delta(2)$
3. $5 \cdot 2^k 3^\ell$ *for* $k \leq 3$, *of complexity* $5 + 2k + 3\ell$ *and defect* $\delta(5) + k\delta(2)$
4. $7 \cdot 2^k 3^\ell$ *for* $k \leq 2$, *of complexity* $6 + 2k + 3\ell$ *and defect* $\delta(7) + k\delta(2)$
5. $19 \cdot 3^\ell$ *of complexity* $9 + 3\ell$ *and defect* $\delta(19)$
6. $13 \cdot 3^\ell$ *of complexity* $8 + 3\ell$ *and defect* $\delta(13)$
7. $(3^k + 1)3^\ell$ *for* $k > 0$, *of complexity* $1 + 3k + 3\ell$ *and defect* $1 - 3\log_3(1 + 3^{-k})$

Strictly speaking, we do not necessarily need this theorem to the same extent as Chapter 3; we only need it if we want to be able to choose step sizes α with α arbitrarily close to 1. In Chapter 3, this was necessary to keep small the degrees of the polynomials; larger steps translates into fewer steps, which translates into lower degree. However, with the method of truncation, we can limit the degree without needing large steps – see Corollary 5.4.24 – allowing us to keep α small if we so choose. For instance, in the author's implementation, we always use $\alpha = \delta(2)$. Nonetheless, one may wish to use larger α, so this proposition is worth noting.

Now that we have the base case, we need the inductive step. In order to state it, we'll first need some definitions:

Definitions 5.2.19. We say n is *most-efficiently* represented as ab if $n = ab$ and $\|n\| = \|a\| + \|b\|$, or as $a + b$ if $n = a + b$ and $\|n\| = \|a\| + \|b\|$. In the former case we will also say that $n = ab$ is a *good factorization* of n. We say n is *solid* if it cannot be written most-efficiently as $a + b$ for any a and b. We say n is *m-irreducible* if it cannot be written most-efficiently as ab for any a and b. And for a real number $\alpha \in (0, 1)$, we define the set T_α to consist of 1 together with those m-irreducible numbers n which satisfy

$$\frac{1}{n-1} > 3^{\frac{1-\alpha}{3}} - 1$$

and do not satisfy $\|n\| = \|n - b\| + \|b\|$ for any solid numbers b with $1 < b \leq n/2$.

Note that for any $\alpha \in (0, 1)$, the set T_α is a finite set, due to the upper bound on the size of numbers $n \in T_\alpha$.

Let us make one more definition:

Definition 5.2.20. For $r \geq 0$, a finite set $\mathcal{S}$ of low-defect pairs will be called a *covering set* for B_r if every $n \in B_r$ can be efficiently 3-represented by some pair in $\mathcal{S}$. (And hence every $n \in A_r$ can be efficiently represented by some $(\hat{f}, C)$ with $(f, C) \in \mathcal{S}$.)

Now we can state the theorem. The theorem provides fives possibilities; three "generic cases" (1 through 3), and two "exceptional cases" (4 and 5).

Theorem 5.2.21. *Suppose that $0 < \alpha < 1$ and that $k \geq 1$. Further suppose that $\mathcal{S}_{1,\alpha}, \mathcal{S}_{2,\alpha}, \ldots, \mathcal{S}_{k,\alpha}$ are covering sets for B_α, $B_{2\alpha}, \ldots, \mathcal{S}_{k\alpha}$, respectively. Then we can build a covering set $\mathcal{S}_{k+1,\alpha}$ for $B_{(k+1)\alpha}$ as follows:*

1. *If $k+1 > 2$, then for $(f, C) \in \mathcal{S}_{i,\alpha}$ and $(g, D) \in \mathcal{S}_{j,\alpha}$ with $2 \leq i, j \leq k$ and $i + j = k + 2$ we include $(f \otimes g, C + D)$ in $\mathcal{S}_{k+1,\alpha}$;*
   while if $k + 1 = 2$, then for $(f_1, C_1), (f_2, C_2), (f_3, C_3) \in \mathcal{S}_{1,\alpha}$, we include $(f_1 \otimes f_2, C_1 + C_2)$ and $(f_1 \otimes f_2 \otimes f_3, C_1 + C_2 + C_3)$ in $\mathcal{S}_{2,\alpha}$.

2. *For $(f, C) \in \mathcal{S}_{k,\alpha}$ and any solid number b with $\|b\| < (k+1)\alpha + 3\log_3 2$, we include $(f \otimes x_1 + b, C + \|b\|)$ in $\mathcal{S}_{k+1,\alpha}$.*

3. *For $(f, C) \in \mathcal{S}_{k,\alpha}$, any solid number b with $\|b\| < (k+1)\alpha + 3\log_3 2$, and any $v \in B_\alpha$, we include $(v(f \otimes x_1 + b), C + \|b\| + \|v\|)$ in $\mathcal{S}_{k+1,\alpha}$.*

4. *For all $n \in T_\alpha$, we include $(n, \|n\|)$ in $\mathcal{S}_{k+1,\alpha}$.*

5. *For all $n \in T_\alpha$ and $v \in B_\alpha$, we include $(vn, \|vn\|)$ in $\mathcal{S}_{k+1,\alpha}$.*

Proof. While this did not appear as an explicit theorem in Chapter 3, it is proved in the proof of Theorem 3.4.10. □

So by applying this and Theorem 5.2.18, we can inductively build up a covering set for any B_r. By choosing α arbitrarily close to 1, in Chapter 3 we obtained the following result:

Theorem 5.2.22. *For any real $r \geq 0$, there exists a finite covering set $\mathcal{S}_r$ for B_r. Furthermore, we can choose $\mathcal{S}_r$ such that each $(f, C) \in \mathcal{S}_r$ has degree at most $\lfloor r \rfloor$.*

However, as has been noted earlier, in this chapter we will show that with the method of truncation, one can obtain the condition on degrees without needing to take α arbitrarily close to 1.

5.3 Further notes on stabilization and stable complexity

Before we continue, let's make a few more notes on the stabilization length $K(n)$ and the stable complexity $\|n\|_{st}$, now that we have the ability to compute them. We begin with the following inequality:

Proposition 5.3.1. *For natural numbers n_1 and n_2, $\|n_1 n_2\|_{st} \leq \|n_1\|_{st} + \|n_2\|_{st}$.*

Proof. Choose k_1, k_2, and K such that $k_1 + k_2 = K$, both $3^{k_i} n_i$ are stable, and $3^K n_1 n_2$ is also stable. Then

$$\|n_1 n_2\|_{st} = \|3^K n_1 n_2\| - 3K \leq \|3^{k_1} n_1\| + \|3^{k_2} n_2\| - 3(k_1 + k_2) = \|n_1\|_{st} + \|n_2\|_{st}.$$

□

Unfortunately, the analogous inequality for addition does not hold; for instance,

$$\|2\|_{st} = 2 > 0 = \|1\|_{st} + \|1\|_{st};$$

more examples can easily be found.

As was mentioned in Section 5.1.3, we can measure the instability of the number n by the quantity $\Delta(n)$, defined as

$$\Delta(n) = \|n\| - \|n\|_{st} = \delta(n) - \delta_{st}(n).$$

We've also already introduced the term "good factorization" to mean a factorization $N = n_1 \cdot \ldots \cdot n_k$ with $\|N\| = \|n_1\| + \ldots + \|n_k\|$. We can thus introduce a measure of how bad a factorization is – and, due to Proposition 5.3.1, a stabilized version:

Definitions 5.3.2. Let $n_1, \ldots, n_k$ be positive integers, and let N be their product. We define $\kappa(n_1, \ldots, n_k)$ to be the difference $\|n_1\| + \ldots + \|n_r\| - \|N\|$. Similarly we define $\kappa_{st}(n_1, \ldots, n_k)$ to be the difference $\|n_1\|_{st} + \ldots + \|n_k\|_{st} - \|N\|_{st}$.

If $\kappa(n_1, \ldots, n_k) = 0$, we will say that the factorization $N = n_1 \cdot \ldots \cdot n_k$ is a *good factorization*. If $\kappa_{st}(n_1, \ldots, n_k) = 0$, we will say that the factorization $N = n_1 \cdot \ldots \cdot n_k$ is a *stably good factorization*.

These definitions lead to the following easily-proved but useful equation:

Proposition 5.3.3. *Let $n_1, \ldots, n_k$ be natural numbers with product N. Then*

$$\Delta(N) + \kappa(n_1, \ldots, n_k) = \sum_{i=1}^{k} \Delta(n_i) + \kappa_{st}(n_1, \ldots, n_k).$$

Proof. Both sides are equal to the difference $\sum_{i=1}^{k} \|n_i\| - \|N\|_{st}$. □

The usefulness of this equation comes from the fact that all the summands are nonnegative integers. For instance, we can obtain the following implications from it:

Corollary 5.3.4. *Let $n_1, \ldots, n_k$ be natural numbers with product N; consider the factorization $N = n_1 \cdot \ldots \cdot n_k$. Then:*

1. *If N is stable and the factorization is good, then the n_i are stable.*
2. *If the n_i are stable and the factorization is stably good, then N is stable.*
3. *If the factorization is stably good, then $K(N) \leq \sum_i K(n_i)$.*

(Part (1) of this proposition is the same as the earlier Proposition 2.3.7.)

Proof. For part (1), by Proposition 5.3.3, if $\Delta(N) = \kappa(n_1, \ldots, n_k) = 0$, then we must have that $\Delta(n_i) = 0$ for all i, i.e., the n_i are all stable. For part (2), again by Proposition 5.3.3, if $\kappa_{st}(n_1, \ldots, n_k) = 0$ and $\Delta(n_i) = 0$ for all i, then we must have $\Delta(N) = 0$, i.e., N is stable. Finally, for part (3) let $K_i = K(N_i)$, and let $K = K_1 + \ldots + K_r$. Then $\prod_i (3^{K_i} n_i) = 3^K n$. Now by hypothesis,

$$\kappa_{st}(3^{K_1} n_1, \ldots, 3^{K_r} n_r) = \kappa_{st}(n_1, \ldots, n_r) = 0,$$

and furthermore each $3^{K_i} n_i$ is stable. Hence by part (2), we must also have that $3^K N$ is stable, that is, that $K(N) \leq K = K(N_1) + \ldots + K(N_r)$. □

Having noted this, let us continue on towards the method of truncation. First, though, we will need to better understand the structure of low-defect polynomials.

5.4 Low-defect expressions, the nesting ordering, and structure of low-defect polynomials

In this section we will go further into the structure of low-defect polynomials. In order to do this, we will investigate the expressions that give rise to them. That is to say, if we have a low-defect polynomial f, it was constructed according to rules (1)–(3) in Definition 5.2.7; each of these rules though gives a way not just of building up a polynomial, but an expression. For instance, we can build up the polynomial $4x+2$ by using rule (1) to make 2, then using rule (3) to make $2x+1$, then using rule (2) to make $2(2x+1) = 4x+2$. The polynomial $4x+2$ itself does not remember its history, of course; but perhaps we want to remember its history – in which we do not

want to consider the *polynomial* $4x+2$, but rather the *expression* $2(2x+1)$, which is different from the expression $4x+2$, which has a different history.

Strictly speaking, it is possible to prove many of the theorems about low-defect polynomials in this and the next section purely by structural induction, using just the rules (1)–(3) in Definition 5.2.7. But introducing low-defect expressions is more enlightening; it makes it clear why, for instance, the nesting ordering (see Definition 5.4.11) takes the form of a forest.

So, with that, we define:

Definition 5.4.1. A *low-defect expression* is defined to be a an expression in positive integer constants, $+$, $\cdot$, and some number of variables, constructed according to the following rules:

1. Any positive integer constant by itself forms a low-defect expression.

2. Given two low-defect expressions using disjoint sets of variables, their product is a low-defect expression. If E_1 and E_2 are low-defect expressions, we will use $E_1 \otimes E_2$ to denote the low-defect expression obtained by first relabeling their variables so that E_1 and E_2 have no variables in common and then multiplying the resulting expressions.

3. Given a low-defect expression E, a positive integer constant c, and a variable x not used in E, the expression $E \cdot x + c$ is a low-defect expression. (We can write $E \otimes x + c$ if we do not know in advance that x is not used in E.)

And, naturally, we also define:

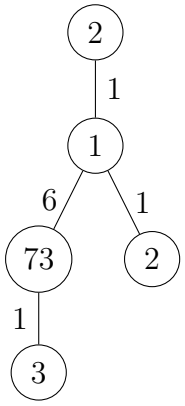
Definition 5.4.2. We define an *augmented low-defect expression* to be an expression of the form $E \cdot x$, where E is a low-defect expression and x is a variable not appearing in E. If E is a low-defect expression, we also denote the augmented low-defect expression $E \otimes x$ by $\hat{E}$.

It is clear from the definitions that evaluating a low-defect expression yields a low-defect polynomial, and that evaluating an augmented low-defect expression yields an augmented low-defect polynomial.

5.4.1 Equivalence and the tree representation

We can helpfully represent a low-defect expression by a rooted tree, with the vertices and edges both labeled by positive integers. Note, some information is lost in this

Figure 5.1: Low-defect tree for the expression $2((73(3x_1+1)x_2+6)(2x_3+1)x_4+1)$.



representation – but, as it happens, nothing we will care about; it turns out that while knowing some of the history of a low-defect polynomial is helpful, knowing the full expression it originated from is more than is necessary. The tree representation is frequently more convenient to work with than an expression, as it does away with such problems as, for instance, 4 and $2 \cdot 2$ being separate expressions. In addition, trees can be treated more easily combinatorially; in a sequel paper[5], we will take advantage of this to estimate how many elements of A_r lie below a given bound x. So we define:

Definition 5.4.3. Given a low-defect expression E, we define a corresponding *low-defect tree* T, which is a rooted tree where both edges and vertices are labeled with positive integers. We build this tree as follows:

1. If E is a constant n, T consists of a single vertex labeled with n.

2. If $E = E' \cdot x + c$, with T' the tree for E, T consists of T' with a new root attached to the root of T'. The new root is labeled with a 1, and the new edge is labeled with c.

3. If $E = E_1 \cdot E_2$, with T_1 and T_2 the trees for E_1 and E_2 respectively, we construct E by "merging" the roots of E_1 and E_2 – that is to say, we remove the roots of E_1 and E_2 and add a new root, with edges to all the vertices adjacent to either of the old roots; the new edge labels are equal to the old edge labels. The label of the new root is equal to the product of the labels of the old roots.

See Figure 5.1 for an example illustrating this construction.

We can use these trees to define a notion of equivalence for expressions:

Definition 5.4.4. Two low-defect expressions are said to be equivalent if their corresponding trees are isomorphic. (Here isomorphism must preserve both the root and all labels.)

Furthermore, every such tree occurs in this way:

Proposition 5.4.5. *Every rooted tree, with vertices and edges labeled by positive integers, occurs (up to isomorphism) as the tree for some low-defect expression.*

Proof. Call the tree T. We prove this by induction on the number of vertices. If T has only one vertex, the root, labeled n, it occurs as the tree for the low-defect expression n. Otherwise, the tree has more than one vertex, i.e., the root has at least one child.

If the root has only one child, let T' be the tree obtained by deleting the root of T, and let E' be a low-defect expression that yields it. If the root is labeled n and the unique edge off of it is labeled c, and x is a variable not appearing in E', then the expression $n(E' \cdot x + c)$ is a low-defect expression that yields T. (If $n = 1$, we may omit the multiplication by n.)

Finally, the root could have more than one child; call its children $v_1, \ldots, v_r$, and call its label n. Then for $1 \leq i \leq r$, let T_i be the tree obtained by removing all vertices except the root and the descendants of v_i, and relabeling the root to have a label of 1. Then for each i we can pick a low-defect expression E_i that yields T_i; then the expression $n \cdot E_1 \cdot \ldots \cdot E_r$ (with the multiplications performed in any order) is a low-defect expression that yields T. (Again, if $n = 1$, we may omit the multiplication by n.) □

Because of Proposition 5.4.5, we can use the term "low-defect tree" to simply refer to a rooted tree with vertices and edges labeled by positive integers. Also, among the various expressions in an equivalence class (i.e., that yield the same tree), the one constructed by Proposition 5.4.5 is one we'd like to pick out:

Definition 5.4.6. Given a low-defect tree T, a low-defect expression for it generated by the method of Proposition 5.4.5 (with multiplications by 1 omitted) will be called a *reduced* low-defect expression for T.

As mentioned above, passing from an expression E to its tree T loses a little bit of information, but not very much. We can, in fact, completely characterize when two expressions will yield the same tree:

Proposition 5.4.7. *Two low-defect expressions E and E' are equivalent if and only if one can get from E to the E' by applying the following transformations to subexpressions:*

1. *For low-defect expressions E_1 and E_2, one may replace $E_1 \cdot E_2$ by $E_2 \cdot E_1$.*
2. *For low-defect expressions E_1, E_2, and E_3, one may replace $(E_1 \cdot E_2) \cdot E_3$ by $E_1 \cdot (E_2 \cdot E_3)$, and vice versa.*
3. *For integer constants n and m, one may replace $n \cdot m$ by the constant nm; and for an integer constant k with $k = mn$, one may replace k by $m \cdot n$. This latter rule may only be applied if k does not appear as an addend in a larger expression.*
4. *For a low-defect expression E_1, one may replace $1 \cdot E_1$ by E_1, and vice versa.*
5. *One may rename all the variables in E, so long as distinct variables remain distinct. (This transformation can only be applied to E as a whole, not subexpressions.)*

Proof. It's clear that all these moves do not change the tree. The problem is proving that all equivalences come about this way.

Suppose T is the tree for E, T' is the tree for E', and $\phi : T \to T'$ is an isomorphism. We induct on the number of vertices of T, the label of the root, and the structure of E and E'.

First we consider the case where either E or E' is a product. In this case, we decompose E and E' until we have written each as a product of low-defect expressions which themselves are not products. Each of these factors can either be written as $F \cdot x + c$ for some low-defect expression F, some x not appearing in F, and some c; or as a natural number constant. Say $E = E_1 \cdot \ldots \cdot E_r \cdot n_1 \cdot \ldots \cdot n_s$ and $E' = E'_1 \cdot \ldots \cdot E'_{r'} \cdot n'_1 \cdot \ldots \cdot n'_{s'}$, where the E_i and E'_i have the former form and the n_i are constants. (Due to rules (1) and (2), we do not need to worry about parenthesization or the order of the factors.) Note that by assumption, $r + s, r' + s' \geq 1$, and at least one of them is at least 2.

Let T_i denote the tree of E_i and T'_i denote the tree of E'_i. Then we can conclude that the root of T has r children, and that T_i can be formed from T by removing, along with all their descendants, all the children of the root except child i, and changing the label of the root to 1. Similarly with T'_i and the r' children of its root. Similarly, if we let N denote the product of the n_i, and N' the product of the n'_i, we see that

N is the label of the root of T, and N' the label of the root of T'. Since T and T' are isomorphic, then, we have $N = N'$, $r = r'$, and ϕ maps the children of the root of T to the children of the root of T'. This allows us to construct isomorphisms $\phi_i : T_i \to T'_{\sigma(i)}$, where σ is a fixed permutation in the group S_r. By the inductive hypothesis, then, each E_i can be turned into $E'_{\sigma(i)}$ by use of moves of type (1)-(5); we can then use rules (1) and (2) to put these back in the original order. (Note that rule (5) should be applied all at once, at the end, so as to ensure that no two distinct variables are ever turned into the same variable.)

Meanwhile, the product $n_1 \cdot \ldots \cdot n_s$ may be turned into the product N by moves of type (3) and (4) (type (4) is necessary if $s = 0$; note that in this case we cannot have $r = 0$). But $N = N'$, which can be turned back into the product $n'_1 \cdot \ldots \cdot n'_{s'}$ by moves of type (3) and (4) as well. This concludes the case where either E or E' is a product.

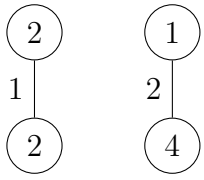
In the case where neither E nor E' is a product, E can either be an integer constant n, or it can be of the form $F \cdot x + c$, where F is a low-defect expression, x is a variable not appearing in F, and c is an integer constant. In the former case, T has no non-root vertices, so neither does T'; since we assumed E' is not a product, this means it too is an integer constant n'. However, n is the label of the unique vertex of T, and n' that of T', and since $T \cong T'$, this implies $n = n'$. Thus E and E' are simply equal, and no moves need be applied.

Finally, we have the case where $E = F \cdot x + c$ as above. In this case, we must also be able to similarly write $E' = F' \cdot x' + c'$, as if E' were a constant, E would be as well by the above argument. Let U and U' denote the trees of F and F', respectively. Then T consists of U together with a new root adjoined with a label of 1, with the unique edge off of it labeled c; and the relation between T', U', and c' is the same. Then since $T \cong T'$, we conclude that $c = c'$ and $U \cong U'$. By the inductive hypothesis, then, U may be transformed into U' by moves of type (1)-(5); this transforms T from $F \cdot x + c$ to $F' \cdot y + c$, where y is some variable not appearing in F'. (Since when applying rule (5), one may have to rename x if one changes one of the variables of F to x.) One may then apply rule (5) again to replace y by x', completing the transformation into E'. This proves the proposition. □

This tells us also:

Corollary 5.4.8. *If E_1 and E_2 are equivalent low-defect expressions that both yield the tree T, they also yield the same low-defect polynomial f, up to renaming of the variables. That is to say, up to renaming of the variables, it is possible to determine*

Figure 5.2: Two different trees yielding the polynomial $4x + 2$



f from T.

Proof. With the exception of renaming the variables, all of the moves allowed in Proposition 5.4.7 consist of replacing subexpressions with other subexpressions that evaluate to the same thing. This proves the claim. □

Note that inequivalent expressions (distinct trees) can also give rise to the same polynomial; for instance, $2(2x+1)$ and $4x+2$ are inequivalent expressions both yielding the polynomial $4x + 2$ (see Figure 5.2). However we will see in Section 5.4.2 that from the polynomial f we can recover at least the "shape" of T, i.e., the isomorphism class of the rooted but unlabeled tree underlying T.

Now, the non-root vertices of the tree correspond to the variables of the original expression:

Definition 5.4.9. Let E be a low-defect expression and T the corresponding tree. We recursively define a bijection between the variables of E and the non-root vertices of T as follows:

1. If E is an integer constant n, then it has no variables, and T has no non-root vertices, and the bijection is the trivial one.

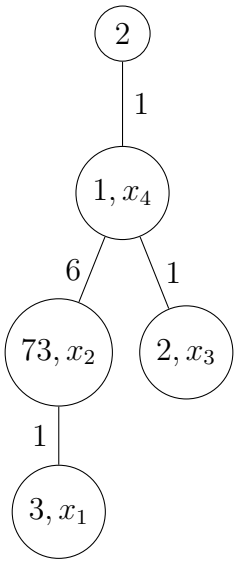
2. If $E = E' \cdot x + c$, with T' the tree for E, then we use the correspondence between variables of E' and the non-root vertices of T' to associate variables of E' with vertices of $T' \subseteq T$; and we assign the root of T' to correspond to the variable x.

3. If $E = E_1 \cdot E_2$, with T_1 and T_2 the trees for E_1 and E_2 respectively, then we use the correspondence between variables of E_1 and non-root vertices of T_1 to associate variables of E_1 with vertices of $T_1 \subseteq T$; and we do similarly with E_2 and T_2.

See Figure 5.3 for an illustration of this bijection.

Equivalently, each variable can be thought of as corresponding to an edge rather than to a non-root vertex; if the variable x corresponds to the vertex v, we can

Figure 5.3: Low-defect tree for the expression $2((73(3x_1+1)x_2+6)(2x_3+1)x_4+1)$; non-root vertices have been marked with corresponding variables in addition to their labels.



instead think of it as corresponding to the edge between v and the parent of v. If we think of variables as corresponding to vertices, however, then we can imagine the root as corresponding to the extra variable in the augmented low-defect expression $\hat{E}$, although this analogy is not perfect.

This bijection, placing the variables of E on the tree, shows us that the variables of a low-defect expression do not all play the same role. In Section 5.5, we will make extensive use of the variables corresponding to leaves. See also Remark 5.4.19 regarding the variables corresponding to the children of the root. In the following subsection, we will begin to lay out the details of how this works.

5.4.2 The nesting order, keys, and anti-keys

Given a low-defect expression, we will define a partial order, the nesting order, on its set of variables. First, let us make the following observation:

Proposition 5.4.10. *Let E be a low-defect expression. Each variable of E appears exactly once in E, and there is a smallest low-defect subexpression of E that contains it.*

Proof. By definition, a variable of E appears in E. A variable of E cannot appear twice in E, as no rule of constructing low-defect expressions allows this; rule (2) only

allows multiplying two low-defect expressions if their variables are disjoint, and rule (3) can only introduce a new variable different from the ones already in E.

For the second part, observe that rule (3) is the only rule that introduces new variables; so say x is some variable of E, it must have been introduced via rule (3). This means that it occurs in a subexpression of E of the form $E' \cdot x + c$, where E' is a low-defect expression and c is a positive integer constant. Since x itself is not a low-defect expression, and neither is $E' \cdot x$, the next-smallest subexpression containing x, i.e., $E' \cdot x + c$, is the smallest low-defect subexpression of E that contains x. □

Because of this, it makes sense to define:

Definition 5.4.11. Let E be a low-defect expression. Let x and y be variables appearing in E. We say that $x \preceq y$ under the nesting ordering for E if x appears in the smallest low-defect subexpression of E that contains y.

This is, in fact, a partial order:

Proposition 5.4.12. *The nesting ordering for a low-defect expression E is a partial order.*

Proof. We have $x \preceq x$ as x appears in any expression containing x. If $x \preceq y$ and $y \preceq z$, then the smallest low-defect expression containing z also contains y, and hence contains the smallest low-defect expression containing y, and hence contains x. And if $x \preceq y$ and $y \preceq x$, then the smallest low-defect expression containing each is contained in the other, i.e., the smallest low-defect expression containing x is the smallest low-defect expression containing y. Since the former has the form $E_1 \cdot x + c_1$, and the latter has the form $E_1 \cdot y + c_2$, we must have $x = y$. □

In fact, it's not just any partial order – it's a partial order that we've already sort of seen; it's the partial order coming from the bijection between variables of a low-defect expression E and non-root vertices of its tree T.

Proposition 5.4.13. *Let E be a low-defect expression, and let T be the corresponding tree. Then $x \preceq y$ under the nesting ordering if and only if the vertex in T corresponding to x is a descendant of the vertex in T corresponding to y.*

Proof. We prove this by structural induction on E. If E is an integer constant, then there are no variables and the statement is trivial.

In the case where $E = E' \cdot x + c$, say T' is the tree corresponding to E'. Suppose x_1 and x_2 are variables of E. If x_1 and x_2 are both variables of E', then by the

inductive hypothesis, $x_1 \preceq x_2$ in the nesting ordering in E' if and only if the vertex corresponding to x_1 in T' is a descendant of that corresponding to x_2. However, it is clear that $x_1 \preceq x_2$ in the nesting ordering of E' if and only if $x_1 \preceq x_2$ in the nesting ordering of E, since the smallest low-defect subexpression of E' containing x_2 is necessarily also the smallest low-defect subexpression of E containing x_2; and similarly with the corresponding vertices. Hence the proposition is proved in this case. Otherwise, we must have that one of the variables is x itself; say the variables are x and x'. But in that case we automatically have that $x' \preceq x$, and the vertex for x' is a descendant of that of x.

This leaves the case where $E = E_1 \cdot E_2$; say T_1 and T_2 are the trees corresponding to E_1 and E_2. If x_1 and x_2 are both variables of E_1, then by the inductive hypothesis, $x_1 \preceq x_2$ in the nesting ordering in E_1 if and only if the vertex corresponding to x_1 in T_1 is a descendant of that corresponding to x_2; but as above, it does not matter if we consider this in E_1 and T_1 or E and T. Similarly the statement holds if x_1 and x_2 are both variables of E_2. Finally, if x_1 is a variable of E_1 and x_2 is a variable of E_2, then x_1 and x_2 are incomparable in the nesting ordering, as the smallest low-defect subexpression containing x_1 is contained in E_1 and hence does not contain x_2, and vice versa; and, correspondingly, the corresponding vertices are incomparable in T. □

Now, we've already seen (Corollary 5.4.8) that it is possible to determine the low-defect polynomial f for a low-defect expression E from its tree T. In fact, not only is it possible to do so, but we can write down an explicit description of the terms of f in terms of T. Specifically:

Proposition 5.4.14. *Let T be a low-defect tree (say with root v_0) and f the corresponding low-defect polynomial after assigning variables to the non-root vertices of T; let x_v denote the variable corresponding to the vertex v. Then for a subset S of $V(T) \setminus \{v_0\}$, the monomial $\prod_{v \in S} x_v$ appears in f in and only if the subgraph induced by $S \cup v_0$ is a subtree of T. Furthermore, its coefficient is given by*

$$\left(\prod_{v \in S \cup \{v_0\}} w(v) \right) \left(\prod_{\substack{e \text{ has exactly one} \\ \text{vertex in } S \cup \{v_0\}}} w(e) \right).$$

The constant term corresponds to the subtree $\{v_0\}$, and the leading term is the term corresponding to all of T.

Proof. Let E be a low-defect expression giving rise to T; we use structual induction on E. If E is an integer constant n, then T consists of just a root labeled with n. So the only rooted subtree of T is T itself, containing no non-root vertices; and, correspondingly, f has a unique term, containing no variables, and with coefficient n, which matches the formula given.

If $E = E' \cdot x + c$, say T' and f' are the tree and the polynomial arising from E. Let v_x be the vertex of T corresponding to x, which is also the root of T'. Then a rooted subtree of T consists of either just v_0, or v_0 together with a rooted subtree of T'. Correspondingly, since $f = xf' + c$, a term of f is either x times a term of f, or just c. The subtree $\{v_0\}$ contains no non-root vertices and so corresponds to c; since the root is labeled with a 1 and the sole edge out of it is labeled with a c, the formula for the coefficient is correct. Any other rooted subtree X consists of v_0 together with a rooted subtree X' or T'; X' corresponds to some term m' of f'. Then we have a term xm' in f, which corresponds to X, since the old root of T' is also the vertex v_x. Furthermore, the coefficient matches that given by the formula, changing X' to X just means adding in the vertex v_0 and the edge $\{v_0, v_x\}$; however, v_0 has a label of 1, not changing the product, and the label of the edge $\{v_0, v_x\}$ is irrelevant as both vertices are in X. (Moreover, no edges drop out of the product, as the only new vertex is v_0, and its only edge is $\{v_0, v_x\}$.) And since every term of f is either c or of the form xm' for some term m' of f', every term arises in this way.

This leaves the case where $E = E_1 \cdot E_2$; say each E_i gives rise to a trees T_i and a polynomial f_i, and let v_i denote the root of T_i. Then a rooted subtree of T consists of $\{v_0\}$ together with subsets $X_1 \subseteq T_1$ and $X_2 \subseteq T_2$ such $X_i \cup \{v_i\}$ is a rooted subtree of T_i. Correspondingly, $f = f_1 f_2$, so each term of f is the product of a term of f_1 and a term of f_2; since f_1 and f_2 have no variables in common, terms $m_1 m_2$ are determined uniquely by the pair (m_1, m_2), which by the inductive hypothesis are in bijection with sets (X_1, X_2) as described above. It remains to check that the coefficients match. Say X_1 and X_2 are subsets as described above, with each X_i corresponding to a term m_i of f_i, so that the subtree $X_1 \cup X_2 \cup \{v_0\}$ corresponds to the term $m_1 m_2$. Then the product of the labels of vertices in $X_1 \cup X_2 \cup \{v_0\}$ is the product of the labels of vertices in $X_1 \cup X_2$ times $w(v_0)$, the latter of which is equal to $w(v_1)w(v_2)$, so this is the same as the product of the labels of vertices in $X_1 \cup \{v_1\}$ times the product of the labels of vertices in $X_2 \cup \{v_2\}$. Meanwhile, the product over the edges is also the product of both the previous ones, as the only edges that could change are those that connected X_1 to v_1 or X_2 to v_2, all of which were previously not in the product due to having both vertices in one of the $X_i \cup \{v_i\}$; but these now connect X_1 and X_2 to

v_0, with both vertices in $X_1 \cup X_2 \cup \{v_0\}$, so they still are not in the product.

Finally, the leading term corresponds to all of T as it contains all the variables, and the constant term corresponds to $\{v_0\}$ as it contains none of the variables. □

This yields the following corollary, which will be useful in Section 5.5:

Corollary 5.4.15. *Let E be a low-defect expression and f the corresponding low-defect polynomial. Any term of f other than the leading term must exclude at least one minimal variable.*

Proof. Consider the low-defect tree corresponding to E. Any subtree other than the whole tree must exclude at least one leaf, i.e., the corresponding term of f must exclude at least one minimal variable. □

It also, in particular, tells us the leading coefficient of f in terms of T, which we will use in Section 5.4.3:

Corollary 5.4.16. *Let T be a low-defect tree, and f be the corresponding low-defect polynomial. Then the leading coefficient of f is the product of the vertex labels of T.*

Proof. The leading term corresponds to the subtree consisting of all of T. This includes all the vertices; and no edge has exactly one vertex in it, as all edges have both vertices in it. □

Now, as we've already noted above, we cannot go backwards from f to determine T; the map from trees to polynomials is not one-to-one. However, we can go part of the way back – we can determine the "shape" of T, that is to say, the isomorphism class of the rooted but unlabeled tree underlying T; it is only the labels we cannot determine with certainty.

To do this, for a low-defect polynomial f, consider the set of monomials that appear in f, without their associated coefficients; ignoring the nesting ordering for a moment, these monomials can be partially ordered by divisibility. But we can, in fact, recover the nesting ordering (and thus the shape of T, without labels) from this partial ordering. First, a definition:

Definition 5.4.17. Let E be a low-defect expression yielding a low-defect tree T and a low-defect polynomial f; let x be a variable in E and v_x the corresponding vertex in T. We define the *key* of x in E to be the term of f corresponding to the subtree consisting of all ancestors of v_x. We define the *anti-key* of x in E to be the term of f corresponding to the subtree consisting of all non-descendants of v_x. So the key of

x is the smallest term of f containing x (under divisibility ignoring coefficients), and the anti-key of x is the largest term not containing x.

Both these operations, key and anti-key, are order-reversing:

Proposition 5.4.18. *Let E be a low-defect expression, and let x and y be variables appearing in E. Then $x \preceq y$ under the nesting ordering if and only if the key of y divides the key of x (ignoring coefficients), which also occurs if and only if the anti-key of y divides the anti-key of x.*

Proof. Let T be the low-defect tree determined by E, and let v_x and v_y be the vertices corresponding to x and y. By Proposition 5.4.13, $x \preceq y$ if and only if, v_x is a descendant of v_y. But if v_x is a descendant of v_y, then every ancestor of v_y is an ancestor of v_x, and so (ignoring coefficients), the key of y divides the key of x. Conversely, if the key of y divides the key of x, then y divides the key of x, and so v_y is an ancestor of v_x. Similarly, if v_x is a descendant of v_y, then every non-descendant of v_y is a non-descendant of v_x, and so the anti-key of y divides the anti-key of x (ignoring coefficients). Convesely, if the anti-key of y divides the anti-key of x, then every non-descendant of v_y is a non-descendant of v_x, i.e., every descendant of v_x is a descendant of v_y, i.e., v_x is a descendant of v_y and so $x \preceq y$. □

Thus, from f alone, the nesting ordering on the variables can be recovered; for as we saw above, it is possible from f alone to determine the key and the anti-key of some variable in f (so we can speak simply of "the key of x in f", or "the anti-key of x in f"). But by Proposition 5.4.18, if x and y are variables in f, and we know their keys or anti-keys, we can determine whether or not $x \preceq y$, without needing to know the tree or expression that f came from; it does not depend on those things. Thus it makes sense to simply talk about the nesting ordering on the variables of f. Furthermore this means we can also recover the shape of T from f alone; the vertex corresponding to x is a child of the vertex corresponding to y if and only if $x \preceq y$ and there are no other variables inbetween, and the vertex corresponding to x is a child of the root if and only if x is maximal in the nesting ordering.

Indeed, we can, given f, determine all trees T that yield it. By above, we know the shape, and which variables correspond to which vertices, and Proposition 5.4.14 constrains the vertex and edge labels – indeed, it not only constrains them, it bounds them (as every label divides at least one coefficient of f), making it possible to determine all T that yield f (and thus to determine $\|f\|$) via brute-force search. (One can also use this procedure to determine if f is a low-defect polynomial at all,

if one does not already know.) But this is rather more involved than what is needed to compute the complexity of a low-defect expression or tree!

Remark 5.4.19. It is the minimal variables of f will turn out to be quite important in Section 5.5, but it's worth noting that the maximal variables have a use too – in Chapter 3, the proposition was proved (Lemma 3.4.3) that if f is a low-defect polynomial of degree at least 1, there exists a variable x, low-defect polynomials g and h, and a positive integer c such that $f = h \cdot (g \cdot x + c)$. With this framework – if we allow for the use of commutativity and associativity – we can easily see that these x are precisely the maximal variables of f.

5.4.3 A lower bound on the complexity of a low-defect polynomial

In this section, we will discuss the notion of the complexity of a low-defect expression, tree, or polynomial, and use this to prove a lower bound on the complexity of a low-defect polynomial (Corollary 5.4.24). This lower bound is what allows us to show that truncation will keep the degrees of our polynomials low despite our use of small step sizes (see discussion in Section 5.2.3).

A low-defect expression has an associated base complexity:

Definition 5.4.20. We define the complexity of a low-defect expression E, denoted $\|E\|$, as follows:

1. If E is a positive integer constant n, we define $\|E\| = \|n\|$.
2. If E is of the form $E_1 \cdot E_2$, where E_1 and E_2 are low-defect expressions, we define $\|E\| = \|E_1\| + \|E_2\|$.
3. If E is of the form $E' \cdot x + c$, where E' is a low-defect expression, x is a variable, and c is a positive integer constant, we define $\|E\| = \|E'\| + \|c\|$.

In Section 5.2 we defined $\|f\|$, for a low-defect polynomial f, to be the smallest C such that (f, C) is a low-defect pair. Above, we also defined the notion of $\|E\|$ for E a low-defect expression. These are compatible as follows:

Proposition 5.4.21. *Let f be a low-defect polynomial. Then $\|f\|$ is the smallest value of $\|E\|$ among low-defect expressions E that evaluate to f.*

Proof. The rules for building up a low-defect pair (f, C) are exactly the same as the rules for building a low-defect expression E, and what these rules do to the base

complexity C is exactly the same as what they do to the complexity $\|E\|$ (except that they allow for increasing C further). So each low-defect pair (f, C) comes from some low-defect expression E yielding f with $\|E\| \leq C$, and any low-defect expression E yielding f yields a low-defect pair $(f, \|E\|)$. So the lowest possible value of C and of $\|E\|$ are the same. □

Indeed, though we will not use this formalism here, it may make sense to consider "low-defect expression pairs", pairs (E, C) where E is a low-defect expression and $C \geq \|E\|$. After all, the definition of $\|E\|$ assumes one knows the complexities of the integer constants appearing in $\|E\|$, but one may not know these exactly, but only have an upper bound on them. For instance, one might not be using low-defect expressions as we defined them here, but rather ones where, instead of integer constants, one has representations of integers in terms of 1, $+$, and $\cdot$. That is to say, perhaps one is not using expressions such as $2(2x + 1)$, but rather such as $(1 + 1)((1 + 1)x + 1)$. In this example, the expressions used for the integer constants were most-efficient, but this may not be the case in general. In this case, it would make sense to consider the complexity of the expression to be simply the number of 1's used, which would be an upper bound on the complexity of the low-defect expression it yields. This sort of only having an upper bound is, after all, the reason we consider pairs (f, C), and it may make sense in other contexts to do with expressions as we do here with polynomials. (Indeed, the author's implementation of the algorithms here does something like this; see Appendix C.)

Since we like to encode low-defect expressions as trees, it makes sense to define the complexity of these:

Definition 5.4.22. The complexity of a low-defect tree, $\|T\|$, is defined to be the smallest $\|E\|$ among all low-defect expressions yielding T.

Note that it follows from this definition that for a low-defect polynomial f, $\|f\|$ can be equivalently characterized as the smallest $\|T\|$ among all trees T yielding f. Again, it may make sense in other contexts to consider pairs (T, C) with $C \geq \|T\|$, for the same reasons discussed above. If, however, we do know the complexity of arbitrary natural numbers, then the complexities of expressions and of trees can be computed as follows:

Proposition 5.4.23. *We have:*

1. *Let E be a low-defect expression. Then $\|E\|$ is equal to the sum of the complexities of all the integer constants occurring in E.*

2. Let T be a low-defect tree. Then

$$\|T\| = \sum_{e \text{ an edge}} \|w(e)\| + \sum_{v \text{ a leaf}} \|w(v)\| + \sum_{\substack{v \text{ a non-leaf vertex} \\ w(v)>1}} \|w(v)\|,$$

where w denotes the label of the given vertex or edge.

Proof. The first statement is a straightforward structural induction. If E is a constant, its complexity is the complexity of that constant. If $E = E_1 \cdot E_2$, its complexity is $\|E_1\| + \|E_2\|$, which by the inductive hypothesis is the sum of the complexities of all the constants used in either. And if $E = E' \cdot x + c$, its complexity is $\|E'\| + \|c\|$, which by the inductive hypothesis is the sum of the complexities of the constants used in E' plus that of the new constant introduced.

For the second statement, consider a reduced low-defect expression E giving rise to T. Then the edge and vertex labels correspond exactly to the constants used in E, with the exception of labels of 1 on non-leaf vertices. As $\|T\| \leq \|E\|$, this shows that the formula above is an upper bound on $\|T\|$. For the lower bound, note that by Proposition 5.4.7, any other low-defect expression for T can be obtained by E by the listed moves. Moves of the form (1), (2), and (5) do not alter the complexity of an expression at all.

This leaves moves of type (3) and (4). Suppose (3) or (4) is going to be applied to a subexpression E'; consider E' as a product (possibly of one thing) and consider the largest product P containing the factors of E' as factors. That is to say, let P be the largest subexpression of the form $E_1 \cdot \ldots \cdot E_k$ (where due to (1) and (2), we do not need to worry about parenthesization or order) where the E_i cannot be written as products, and the factors of E' are among the E_i. Since (3) and (4), applied to factors of P, only alter things within P, and do not alter the internals of any E_i which can be written as a sum, we see that the least complexity is obtained by minimizing the complexity of each individual product P. But this is clearly done by multiplying together all constants and eliminating 1's where possible. This leaves us with an expression which is the same as E up to moves of the form (1), (2), and (5). Hence E has the lowest complexity among expressions for T, and so $\|T\| = \|E\|$, which as noted, is given by the formula.

□

With this, we now obtain our lower bound:

Proposition 5.4.24. *Let (f, C) be a low-defect pair of degree k, and suppose that a is the leading coefficient of f. Then $C \geq \|a\| + k$. Equivalently, if f is a low-defect polynomial of degree k with leading coefficient a, then $\|f\| \geq \|a\| + k$.*

Proof. Let T be a low-defect tree giving rise to f with $C \geq \|T\|$. Then

$$\begin{aligned}\|T\| &\geq \sum_{e \text{ an edge}} \|w(e)\| + \sum_{\substack{v \text{ a vertex} \\ w(v)>1}} \|w(v)\| \\ &\geq \left(\sum_{e \text{ an edge}} 1 \right) + \left\| \prod_{\substack{v \text{ a vertex} \\ w(v)>1}} w(v) \right\|.\end{aligned}$$

That is, by Corollary 5.4.16, it is at least the number of edges plus $\|a\|$. Since the number of edges is one less than the number of vertices, the number of edges is k. So $C \geq \|a\| + k$.

The second statement then follows as $\|f\|$ is by definition the smallest C among low-defect pairs (f, C). □

In particular, the degree of a polynomial is bounded by its defect:

Corollary 5.4.25. *Let (f, C) be a low-defect pair of degree k, and suppose that a is the leading coefficient of f. Then $\delta(f, C) \geq \delta(a) + k \geq k$. Equivalently, $\delta(f) \geq \delta(a) + k \geq k$.*

Proof. By definition, $\delta(f, C) = C - 3\log_3 a$. So

$$\delta(f, C) = C - 3\log_3 a \geq \|a\| + k - 3\log_3 a = \delta(a) + k,$$

and $\delta(a) + k \geq k$. The second statement then follows as $\delta(f)$ is just the smallest value of $\delta(f, C)$ among low-defect pairs (f, C). □

5.5 The truncation operation

Now, finally, we can describe the operation of truncating a low-defect polynomial (or expression, or tree) to a given defect – the "filtering-down" half of our method. The results here will be phrased in terms of low-defect pairs, but the analogues for low-defect expressions are clear.

5.5.1 Truncations and their properties

First we just describe truncating a low-defect polynomial in general:

Proposition 5.5.1. *Let (f, C) be a low-defect pair, say of degree r, and suppose x_i is a variable of f which is minimal with respect to the nesting ordering. Let $k \geq 0$ be an integer, and define*

$$g(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_r) := f(x_1, \ldots, x_{i-1}, 3^k, x_{i+1}, \ldots, x_r).$$

Then:

1. *The polynomial g is a low-defect polynomial, and $(g, C + 3k)$ is a low-defect pair.*
2. *If a is the leading coefficient of f, then the leading coefficient of g is strictly greater than $a3^k$, and so $\delta(g, C + 3k) < \delta(f, C)$.*
3. *The nesting order on the variables of g is the restriction of the nesting order on the variables of f to $\{x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_r\}$.*
4. *For any $k_1, \ldots, k_{i-1}, k_{i+1}, \ldots, k_r$, we have*

$$\delta_{g,C+3k}(k_1, \ldots, k_{i-1}, k_{i+1}, \ldots, k_r) = \delta_{f,C}(k_1, \ldots, k_{i-1}, k, k_{i+1}, \ldots, k_r).$$

Proof. Let E be a low-defect expression of complexity at most C giving rise to f; we apply structural induction to prove parts (1), (2), and (3). Note that E cannot be an integer constant as then it would have no variables.

If $E = E' \cdot x + c$, there are two cases; either E' has degree 0, or it has positive degree. In the former case, x is the unique minimal variable, so $x_i = x$; say E' evaluates to the constant n and has complexity at most $C' = C - \|c\|$. Then g is equal to the constant $n3^k + c$, which can be given by a low-defect expression. Furthermore, the complexity of this low-defect expression is at most $C' + 3k + \|c\| = C + 3k$, so $(g, C + 3k)$ is a low-defect pair. And whereas the leading coefficient of f was n, the leading coefficient of g is $n3^k + c > n3^k$. Finally, g has no variables, so part (3) is trivially true.

Otherwise, if E' has positive degree, then x is not minimal, and the minimal variables in E are precisely the minimal variables in E'. Assume without loss of generality that $x = x_r$. Say E' has complexity at most $C' = C - \|c\|$. Let f' be the polynomial coming from E', and

$$g'(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_{r-1}) := f'(x_1, \ldots, x_{i-1}, 3^k, x_{i+1}, \ldots, x_{r-1}).$$

Then by the inductive hypothesis, g' is a low-defect polynomial, coming from some low-defect expression E'' with complexity at most $C' + 3k$. So g is a low-defect polynomial as it comes from the low-defect expression $E'' \cdot x + c$, which has complexity at most

$$C' + 3k + \|c\| = C + 3k.$$

And if a is the leading coefficient of f, then it is also the leading coefficient of f', and so by the inductive hypothesis the leading coefficient of g' is greater than $a3^k$, but the leading coefficient of g is the same as that of g'. Finally, by the inductive hypothesis, the nesting order on the variables of g' is the restriction of the nesting order of the variables of f', and the nesting order on the variables of g is the same as that on the variables of g', but with x_r added as a new maximum element; since the same relation holds between the nesting order for f and the nesting order for f', part (3) is true in this case.

This leaves the case where $E = E_1 \cdot E_2$. In this case, a minimal variable of E is either a minimal variable of E_1 or a minimal variable of E_2. Suppose without loss of generality that

$$E(x_1, \ldots, x_r) = E_1(x_1, \ldots, x_s)E_2(x_{s+1}, \ldots, x_r)$$

and $i \leq s$. Say E_1 and E_2 give rise to polynomials f' and h, and E_1 has complexity at most $C' = C - \|E_2\|$. Then if we define

$$g'(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_s) := f'(x_1, \ldots, x_{i-1}, 3^k, x_{i+1}, \ldots, x_s),$$

by the inductive hypothesis, g' is a low-defect polynomial, coming from some low-defect expression E' with complexity at most $C' + 3k$. So g is a low-defect polynomial as it comes from the low-defect expression $E' \cdot E_2$, which has complexity at most $C' + 3k + \|E_2\| = C + 3k$. And if a_1 is the leading coefficient of f' and a_2 is the leading coefficient of h, then the leading coefficient of $f = f' \cdot h$ is a_1a_2, while by the inductive hypothesis, the leading coefficient of g is strictly greater than $3^k a_1$, and so the leading coefficient of $g = g' \cdot h$ is strictly greater than $3^k a_1 a_2$. Finally, the nesting order on the variables of g is just the disjoint union of the nesting order on the variables of g' and the nesting order on the variables of h, and the same relation holds between the nesting order for f and the nesting order for f'. By the inductive hypothesis, the nesting order for g' is just the restriction of that for f', so the same relation holds between g and f.

To prove the second statement in part (2), we note that if a is the leading coefficient of f and b is the leading coefficient of g, since $b > a3^k$,

$$\delta(g, C + 3k) = C + 3k - 3\log_3(b) = C - 3\log_3(b3^{-k}) < C - 3\log_3(a) = \delta(f, C).$$

Finally, part (4) follows as

$$\begin{aligned}
&\delta_{g,C+3k}(k_1, \ldots, k_{i-1}, k_{i+1}, \ldots, k_r) = \\
&C + 3k + 3(k_1 + \ldots + k_{i-1} + k_{i+1} + \ldots + k_r) - 3\log_3 g(3^{k_1}, \ldots, 3^{k_{i-1}}, 3^{k_{i+1}}, \ldots, 3^{k_r}) = \\
&C + 3(k_1 + \ldots + k_{i-1} + k + k_{i+1} + \ldots + k_r) - 3\log_3 f(3^{k_1}, \ldots, 3^{k_{i-1}}, 3^k, 3^{k_{i+1}}, \ldots, 3^{k_r}) = \\
&\qquad\qquad \delta_{f,C}(k_1, \ldots, k_{i-1}, k, k_{i+1}, \ldots, k_r).
\end{aligned}$$

□

Definition 5.5.2. Let (f, C) be a low-defect pair, and let (g, D) be obtained from it as in Proposition 5.5.1; we will call (g, D) a *direct truncation* of (f, C), and g an direct truncation of f.

Furthermore, we will define (g, D) to be a *truncation* of (f, C) if there are low-defect pairs $(f, C) = (f_0, C_0), (f_1, C_1), \ldots, (f_k, C_k) = (g, D)$ with (f_{i+1}, C_{i+1}) a direct truncation of (f_i, C_i). Similarly in this case we say g is a truncation of f.

Immediately we get:

Proposition 5.5.3. *Say (f, C) is a low-defect pair and (g, D) is a truncation of it. Then:*

1. *$\delta(g, D) < \delta(f, C)$.*
2. *The nesting order on the variables of g is the restriction of the nesting order on the variables of f.*

Proof. This follows immediately from iterating parts (2) and (3) of Proposition 5.5.1. □

So, when we truncate f, we are substituting powers of 3 into some of the variables, and leaving the other variables free. Say f has degree r, and consider the function

$$(k_1, \ldots, k_r) \mapsto f(3^{k_1}, \ldots, 3^{k_r})$$

from $\mathbb{Z}^r_{\geq 0}$ to $\mathbb{N}$; when we truncate f, we are fixing the values of some of the k_i. In a sense, we are restricting f to a subset of $\mathbb{Z}^r_{\geq 0}$ fo the form $S_1 \times \ldots \times S_r$, where each S_i is either a single point or all of $\mathbb{Z}_{\geq 0}$.

As such we will want a way of talking about such sets; we will represent them by elements of $(\mathbb{Z}_{\geq 0} \cup \{*\})^r$, where here $*$ is just an abstract symbol which is distinct from any whole number; it represents "this position can be any number", or the set $\mathbb{Z}_{\geq 0}$, where putting in an actual number n would represent "this position must be n", or the set $\{n\}$. Let us formally define our way of getting a set from such an object, how we can substitute these objects into low-defect polynomials:

Definitions 5.5.4. Given $(k_1, \ldots, k_r) \in (\mathbb{Z}_{\geq 0} \cup \{*\})^r$, we define $S(k_1, \ldots, k_r)$ to be the set

$$\{(\ell_1, \ldots, \ell_r) \in \mathbb{Z}^r_{\geq 0} : \ell_i = k_i \text{ for } k_i \neq *\}.$$

Furthermore, given $f \in \mathbb{Z}[x_1, \ldots, x_r]$, we define the 3-substitution of $(k_1, \ldots, k_r)$ into f to be the polynomial obtained by substituting 3^{k_i} for x_i whenever $k_i \neq *$. If (f, C) is a low-defect pair, we define the 3-substitution of $(k_1, \ldots, k_r)$ to be (g, D) where g is the 3-substitution of $(k_1, \ldots, k_r)$ into f, and $D = C + 3\sum_{k_i \neq *} k_i$.

Be warned that in general, 3-substituting into a low-defect pair may not yield a low-defect pair, if one substitutes into the wrong variables. For instance, if $(f, C) = ((3x_1 + 1)x_2 + 1, 5)$, then 3-substituting in $(*, 1)$ yields $(9x + 4, 8)$, which is not a low-defect pair. And if $(f, C) = ((3x_1 + 1)(3x_2 + 1)x_3 + 1, 9)$, and one 3-substitutes in $(*, *, 0)$, then one obtains $(9x_1x_2 + 3x_1 + 3x_2 + 2, 9)$, the first element of which is not a low-defect polynomial at all.

However, in what follows, we will only be using this notion in cases where it does, in fact, turn out to be a low-defect pair – specifically, in the following cases:

Proposition 5.5.5. *Let (f, C) be a low-defect pair, and let $(k_1, \ldots, k_r) \in (\mathbb{Z}_{\geq 0} \cup \{*\})^r$ be such that the set of i for which $k_i \neq *$ corresponds to a downward-closed subset of the variables of f. Let (g, D) denote the 3-substitution of $(k_1, \ldots, k_r)$ into (f, C). Then:*

1. *The pair (g, D) is a truncation of (f, C) (and hence a low-defect pair).*
2. *Let t be the number of i such that $k_i = *$, and let ι be the map from $\mathbb{Z}^t_{\geq 0}$ to $\mathbb{Z}^r_{\geq 0}$ given by inserting the arguments $(\ell_1, \ldots, \ell_t)$ into the coordinates of $(k_1, \ldots, k_r)$ where $k_i = *$. Then $\delta_{g,D} = \delta_{f,C} \circ \iota$.*

Furthermore, all truncations of (f, C) arise in this way.

Proof. We first prove part (1). Let $i_1, \ldots, i_s$ be the indices for which $k_i \neq *$, enumerated in an order such that if $x_{i_j} \preceq x_{i_{j'}}$ then $i_j \leq i_{j'}$. Let $(f_0, C_0) = (f, C)$. Now, for $1 \leq j \leq s$, given (f_{j-1}, C_{j-1}), we will take (f_j, C_j) to be the direct truncation of (f_{j-1}, C_{j-1}) where $3^{k_{i_j}}$ is substituted into x_{i_j}. Of course, in order for this to be a direct truncation, x_{i_j} must be minimal in f_{j-1}. But this follows due to the order we have enumerated the elements; by assumption, each x_{i_j} is minimal in $\{x_{i_j}, \ldots, x_{i_s}\}$, and since $\{x_{i_1}, \ldots, x_{i_s}\}$ is downwardly closed in $\{x_1, \ldots, x_r\}$, we have that $\{x_{i_j}, \ldots, x_{i_s}\}$ is downwardly closed in $\{x_1, \ldots, x_r\} \setminus \{x_{i_1}, \ldots, x_{i_{j-1}}\}$, and so x_{i_j} is minimal in $\{x_1, \ldots, x_r\} \setminus \{x_{i_1}, \ldots, x_{i_{j-1}}\}$. And by Proposition 5.5.3, this last set is precisely the set of variables of f_j, with the same nesting order. Thus this is indeed a truncation.

Part (2) follows by simply iterating part (4) of Proposition 5.5.1 in the above. Finally, we can see that every truncation arises in this way by inducting on the number of steps in the truncation. If there are no steps, then this is true with $(k_1, \ldots, k_r) = (*, \ldots, *)$. Otherwise, say that (f_s, C_s) is an s-step truncation of (f, C) and that (f_{s+1}, C_{s+1}) is a direct truncation of it; we assume by induction that (f_s, C_s) is the 3-substitution into (f, C) of some tuple $(k_1, \ldots, k_r) \in (\mathbb{Z}_{\geq 0} \cup \{*\})^r$. Then (f_{s+1}, C_{s+1}) is the 3-substitution into (f_s, C_s) of some tuple $(*, \ldots, *, \ell_j, *, \ldots, *) \in (\mathbb{Z}_{\geq 0} \cup \{*\})^{r-s}$ (here $\ell_j \neq *$). This makes it the 3-substitution into (f, C) of some tuple $(k'_1, \ldots, k'_r)$, where $k'_i = k_i$ when $k_i \neq *$, and $k_i = \ell_j$ for one particular i with $k_i = *$. □

So, in fact, we'll only be using the notion of 3-substitution in cases where it yields a truncation; or, really, we'll just be using it as another way of thinking about truncation.

5.5.2 Truncating a polynomial to a given defect

Having discussed truncation in general, we can now discuss how to truncate a low-defect polynomial to a given defect. Earlier, in Proposition 5.2.14, we showed that for a low-defect pair (f, C), the number $\delta(f, C)$ is the least upper bound of the values of $\delta_{f,C}$. Now we show that something stronger is true:

Proposition 5.5.6. *Let (f, C) be a low-defect pair of degree r. Say x_{i_j}, for $1 \leq j \leq s$, are the minimal variables of f. Then*

$$\lim_{k_{i_1}, \ldots, k_{i_s} \to \infty} \delta_{f,C}(k_1, \ldots, k_r) = \delta(f, C)$$

(where the other k_i remain fixed).

Proof. Consider once again g, the reverse polynomial of f:

$$g(x_1, \ldots, x_r) = x_1 \ldots x_r f(x_1^{-1}, \ldots, x_r^{-1}).$$

So g is a multilinear polynomial in $x_1, \ldots, x_r$, with the coefficient of $\prod_{i \in S} x_i$ in g being the coefficient of $\prod_{i \notin S} x_i$ in f. Let a denote the leading coefficient of f, which is also the constant term of g.

By Corollary 5.4.15, every non-leading term of f excludes some minimal variable. Hence every non-constant term of g includes some minimal variable. So if we once again write

$$\delta_{f,C}(k_1, \ldots, k_r) = C - 3 \log_3 g(3^{-k_1}, \ldots, 3^{-k_r}),$$

we see that as the minimal variables approach infinity, then each non-constant term of $g(3^{-k_1}, \ldots, 3^{-k_r})$ approaches 0, and so $g(3^{-k_1}, \ldots, 3^{-k_r})$ approaches a. So once again we have

$$\lim_{k_{i_1}, \ldots, k_{i_s} \to \infty} \delta_{f,C}(k_1, \ldots, k_r) = C - 3 \log_3 a = \delta(f, C).$$

□

One can obtain numerical versions of this proposition, but we do not bother to state them here. One can still write the algorithms this chapter describes without needing to make this numerical, and, as discussed in Appendix C, in the author's implementation that is exactly what we have done.

We can restate this proposition as follows:

Corollary 5.5.7. *Let (f, C) be a low-defect pair, say of degree $r > 0$, let $0 \leq s < \delta(f, C)$ be a real number. Then there exists a number K such that, whenever $\delta_{f,C}(k_1, \ldots, k_r) < s$, then $k_i \leq K$ for some i such that x_i is minimal in the nesting ordering for f.*

Proof. By Proposition 5.5.6, since $s < \delta(f, C)$, we can choose some K such that $\delta_{f,C}(k_1, \ldots, k_r) \geq s$, where $k_i = K + 1$ if x_i is minimal in the nesting ordering and $k_i = 0$ otherwise. Then if for some $\ell_1, \ldots, \ell_r$ we have $\delta_{f,C}(\ell_1, \ldots, \ell_r) < s$, then since $\delta_{f,C}$ is increasing in all variables, there is some i such that $\ell_i \leq K$. □

With this, we can now finally describe truncating a low-defect pair to a specified defect:

Theorem 5.5.8. *Let (f, C) be a low-defect pair, say of degree r, let $s \geq 0$ be a real number, and let $S = \{(k_1, \ldots, k_r) : \delta_{f,C}(k_1, \ldots, k_r) < s\}$. Then there exists a finite set $T \subseteq (\mathbb{Z}_{\geq 0} \cup \{*\})^r$ such that:*

1. *We have $S = \bigcup_{p \in T} S(p)$.*
2. *For each p in T, the set of i for which $k_i \neq *$ corresponds to a subset of the variables of f which is downward closed (under the nesting ordering); hence if (g, D) denotes the 3-substitution of p into (f, C), then (g, D) is a truncation of (f, C). Furthermore, we have $\delta(g, D) \leq s$, and hence $\deg g \leq \lfloor s \rfloor$; and if g has degree 0, the former inequality is strict.*

Proof. We prove the statement by induction on r.

Suppose $r = 0$, that is to say, f is a constant n. If $s > \delta(f, C)$, then we may take $T = \{()\}$, where here () indicates the unique element of $(\mathbb{Z}_{\geq 0} \cup \{*\})^0$. For $S() = \{()\}$, and $S = \{()\}$ as well, for $\delta_{f,C}() = C - 3 \log_3 n = \delta(f, C) < s$. So the first condition is satisfied. For the second condition, the set of indices used is the empty set, we have $(g, D) = (f, C)$ (hence (g, D) is trivially a truncation), and so $\delta(g, D) = \delta(f, C) < s$.

Otherwise, if $s \leq \delta(f, C)$, we take $T = \emptyset$, so $\bigcup_{p \in T} S(p) = \emptyset$. Since, as was noted above, $\delta_{f,C}() = \delta(f, C)$, we have $\delta_{f,C}() \geq s$, and hence $S = \emptyset$; thus the first condition is satisfied. The second condition is satisfied trivially.

Now suppose that $r > 0$. Once again, we have two cases. If $s \geq \delta(f, C)$, then we may take $T = \{(*, \ldots, *)\}$. By Proposition 5.2.13, for any $(k_1, \ldots, k_r) \in \mathbb{Z}^r_{\geq 0}$, we have $\delta_{f,C}(k_1, \ldots, k_r) < \delta(f, C) \leq s$, i.e. $S = \mathbb{Z}^r_{\geq 0} = S(*, \ldots, *)$, satisfying the first condition. For the second condition, we once again have that the set of indices used is the null set, so $(g, D) = (f, C)$, and so is trivially a truncation, and $\delta(g, D) = \delta(f, C) \leq s$.

This leaves the case where $r > 0$ and $s < \delta(f, C)$. In this case, we may apply Corollary 5.5.7, and choose a K such that whenever $\delta_{f,C}(k_1, \ldots, k_r) < s$, then $k_i \leq K$ for some i which is minimal in the nesting ordering. That is to say, if we define

$$T_0 := \{(*, \ldots, *, k_i, *, \ldots, *) : x_i \text{ minimal in nesting ordering, } k_i \leq K\},$$

then $S \subseteq \bigcup_{p \in T_0} S(p)$, and for each $p \in T$, the 3-substitution of p into (f, C) is a direct truncation of (f, C). However, we still do not necessarily have that $\delta(g, D) \leq s$, nor do we necessarily have equality in the first condition. This is where we apply the inductive hypothesis.

For each $p \in T_0$, let (g_p, D_p) be the 3-substitution of p into (f, C); this is a direct truncation of (f, C). Apply the inductive hypothesis to each (g_p, D_p) to obtain $T_p \subseteq (\mathbb{Z}_{\geq 0} \cup \{*\})^{r-1}$. We can then pull this back to $T'_p \subseteq (\mathbb{Z}_{\geq 0} \cup \{*\})^r$; since $p = (*, \ldots, *, k_i, *, \ldots)$ for some position i and some number k_i, we can pull back $q = (\ell_1, \ldots, \ell_{i-1}, \ell_{i+1}, \ldots, \ell_r) \in T_p$ (where here we may have $\ell_j = *$) to $q' := (\ell_1, \ldots, \ell_{i-1}, k_i, \ell_{i+1}, \ldots, \ell_r)$. Finally we can take $T = \bigcup_{p \in T_0} T'_p$.

It remains to show that T has the desired properties. Say we have an element of T; it is an element of some T'_p, i.e., with the notation above, it has the form q' for some $q \in T_p$. Say $p = (*, \ldots, *, k_i, *, \ldots)$. The indices used in q correspond to some downward closed subset of the variables of (g_p, D_p), i.e. to a downward closed subset of $\{x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_r\}$. Since x_i is minimal in $\{x_1, \ldots, x_r\}$, adding it in again results in a downward closed set.

Now we check that $S \subseteq \bigcup_{p \in T} S(p)$. Say $\delta_{f,C}(k_1, \ldots, k_r) < s$; then there is some i with x_i minimal and $k_i \leq K$. Let p be the corresponding element of T_0 and (g_p, D_p) as above. Then by Proposition 5.5.1, $\delta_{g_p, D_p}(k_1, \ldots, k_{i-1}, k_{i+1}, \ldots, k_r) = \delta_{f,C}(k_1, \ldots, k_r)$, and so $(k_1, \ldots, k_{i-1}, k_{i+1}, \ldots, k_r) \in T_p$, and so $(k_1, \ldots, k_r) \in T'_p \subseteq T$, as needed.

Suppose now that we take an element of T; write it as $q' \in T'_p$ for some p and some $q \in T_p$, using the notation above. Then $(g_{q'}, D_{q'})$ can also be obtained by 3-susbtituting q into (g_p, D_p); hence by the inductive hypothesis, $\delta(g_{q'}, D_{q'}) \leq s$, and this is strict if $\deg g_{q'} = 0$. This then proves as well that $S \supseteq \bigcup_{p \in T} S(p)$; say $q' \in T$, write $q' = (k_1, \ldots, k_r)$, and let $i_1, \ldots, i_s$ be the indices for which $k_i = *$. Then for $(\ell_1, \ldots, \ell_r) \in S(q')$, we may write $\delta_{f,C}(\ell_1, \ldots, \ell_r) = \delta_{g_{q'}, D_{q'}}(\ell_{i_1}, \ldots, \ell_{i_s})$, and this latter is less than s, since it is at most $\delta(g_{q'}, D_{q'})$, and strictly less than it if $\deg g_{q'} > 0$. This proves the theorem. □

And if we can truncate one low-defect polynomial to a given defect, we can truncate many low-defect polynomials to that same defect. Here then, at last, is what results from taking the "building-up" Theorem 5.2.21, and applying our new "filtering-down" step:

Theorem 5.5.9. *For any real $s \geq 0$, there exists a finite set $\mathcal{S}_s$ of low-defect pairs satisfying the following conditions:*

1. *For any $n \in B_s$, there is some low-defect pair in $\mathcal{S}_s$ that efficiently 3-represents n.*

2. *Each pair $(f, C) \in \mathcal{S}_s$ satisfies $\delta(f, C) \leq s$, and hence $\deg f \leq \lfloor s \rfloor$; and if f has degree 0, the former inequality is strict.*

Proof. By Theorem 5.2.22, there exists a finite set $\mathcal{T}_s$ of low-defect pairs such that for any $n \in B_s$, there is some low-defect pair in $\mathcal{T}_s$ that efficiently 3-represents s. (Indeed, by Theorem 5.2.22, we may even choose $\mathcal{T}_s$ to only consist of polynomials of degree at most $\lfloor s \rfloor$, but this is not needed.)

Now for each $(f, C) \in \mathcal{T}_s$, take $T_{f,C}$ as provided by Theorem 5.5.8; define $\mathcal{T}_{f,C}$ to be the set

$$\{(g, D) : (g, D) \text{ is a 3-substitution of } p \text{ into } (f, C),\ p \in \mathcal{T}_{f,C}\};$$

this is a set of low-defect pairs by condition (2) of Theorem 5.5.8. We can then define $\mathcal{S}_s$ to be the union of the $\mathcal{T}_{f,C}$. We see immediately that $\mathcal{S}$ satisfies condition (2) of the theorem, as this follows from condition (2) of Theorem 5.5.8.

To verify condition (1), say $n \in B_s$. Then there is some $(f, C) \in \mathcal{T}_s$ that efficiently 3-represents n; say $n = f(3^{\ell_1}, \ldots, 3^{\ell_r})$ with $\|n\| = C + 3(\ell_1 + \ldots + \ell_r)$, so $\delta_{f,C}(\ell_1, \ldots, \ell_r) = \delta(n) < s$. Then $(\ell_1, \ldots, \ell_r) \in S(p)$ for some $p \in T_{f,C}$. Say $p = (k_1, \ldots, k_r)$, and let $i_1, \ldots, i_s$ be the indices for which $k_i = *$. Then if we let (g, D) be the 3-substitution of p into (f, C), then $n = f(3^{\ell_1}, \ldots, 3^{\ell_r}) = g(3^{\ell_{i_1}}, \ldots, 3^{\ell_{i_s}})$, and $\|n\| = C + 3(\ell_1 + \ldots + \ell_r) = D + 3(\ell_{i_1} + \ldots + \ell_{i_s})$, so n is efficiently 3-represented by $(g, D) \in \mathcal{T}_{f,C} \subseteq \mathcal{S}_s$. □

The output of this proof is a useful enough concept that we will give it a name:

Definition 5.5.10. For $r \geq 0$, a finite set $\mathcal{S}$ of low-defect pairs will be called a *good covering* for B_r if every $n \in B_r$ can be efficiently 3-represented by some pair in $\mathcal{S}$ (and hence every $n \in A_r$ can be efficiently represented by some $(\hat{f}, C)$ with $(f, C) \in \mathcal{S}$); for every $(f, C) \in \mathcal{S}$, $\delta(f, C) \leq r$, with this being strict if $\deg f = 0$.

Note that although a good covering of B_r cannot produce extraneous numbers in the sense of 3-representing numbers whose defects are too high, it can still 3-represent numbers that are not leaders. For example, if we use Algorithm 6 from the next section with step size $\alpha = \delta(2)$ to produce a good covering of $B_{12\delta(2)}$, we will find that it includes the low-defect pair $(4x + 2, 5)$, which 3-represents the number 6, which is not a leader. Still, this is a minor deficiency; and so long as one cares primarily about A_r and not B_r, it is hardly a deficiency at all.

5.6 Algorithms: Computing good coverings

In this section we now turn the abstract results of the above sections into algorithms. However, the results of the above sections deal with real numbers, but real numbers

cannot be represented exactly in a computer. Hence, we will for the rest of this section fix a subset R of the real numbers on which we can do exact computation. For concreteness, we will define

Definition 5.6.1. The set R is the set of all real numbers of the form $q + r \log_3 n$, where q and r are rational and n is a natural number.

This will suffice for our purposes; however it is worth noting that all these algorithms will work just as well with a larger set of allowed numbers, so long as it supports all the required operations.

In this section we will present several algorithms for computing numbers of small defect and extracting information from them. Since these algorithms in some cases simply use the methods described in the proofs in this chapter, we will, in these cases, not give detailed proofs of correctness; we will simply direct the reader to the proof of the corresponding theorem. We will include proofs of correctness only where we are not directly following the proof of an earlier theorem.

5.6.1 Algorithm 1: Computing B_α, $0 < \alpha < 1$.

We begin with the algorithms for computing a good covering of a given B_r. First, we have Algorithm 1, found on page 160, for the base case, that of determining B_α, for $0 < \alpha < 1$.

Proof of correctness for Algorithm 1. The correctness of this algorithm is immediate from Theorem 5.2.18. □

5.6.2 Algorithm 2: Computing $B_{(k+1)\alpha}$.

Now we record Algorithm 2, found on page 161, for computing a covering set for $B_{(k+1)\alpha}$ if we have ones already for $B_\alpha, \ldots, B_{k\alpha}$. It is mostly just the same as Theorem 5.2.21, but we make a slight modification to avoid redundancy.

Proof of correctness for Algorithm 2. Algorithm 2 is, for the most part, exactly Theorem 5.2.21. The only difference is the removal of the pairs $(3, 3)$ and $(1, 1)$ from the possibilities of things to multiply by; this step needs additional justification. For $(1, 1)$, this is because no number n can be most-efficiently represented as $1 \cdot n$; if (f, C) is a low-defect pair, then the low-defect pair $(f, C + 1)$ cannot efficiently 3-represent anything, as anything it 3-represents is also 3-represented by the pair (f, C). For $(3, 3)$, there are two possibilities. If $3n$ is a number which is 3-represented by by

Algorithm 1 Determine the set B_α

Ensure: $\alpha \in (0,1) \cap R$
Require: $T = \{(n,k) : n \in B_\alpha, k = \|n\|\}$
  $T \leftarrow \{(3,3)\}$
  Determine largest integer k such that $k\delta(2) < \alpha$ and $k \leq 9$ {k may be 0, in which case the following loop never executes}
  for $i = 1$ **to** k **do**
    $T \leftarrow T \cup \{(2^i, 2i)\}$
  end for
  Determine largest integer k such that $\delta(5) + k\delta(2) < \alpha$ and $k \leq 3$ {k may be negative, in which case the following loop never executes}
  for $i = 0$ **to** k **do**
    $T \leftarrow T \cup \{(5 \cdot 2^i, 5 + 2i)\}$
  end for
  Determine largest integer k such that $\delta(7) + k\delta(2) < \alpha$ and $k \leq 2$ {k may be negative, in which case the following loop never executes}
  for $i = 0$ **to** k **do**
    $T \leftarrow T \cup \{(7 \cdot 2^i, 6 + 2i)\}$
  end for
  if $\alpha > \delta(19)$ **then**
    $T \leftarrow T \cup \{(19, 9)\}$
  end if
  if $\alpha > \delta(13)$ **then**
    $T \leftarrow T \cup \{(13, 8)\}$
  end if
  Determine largest integer k for which $1 - 3\log_3(1 + 3^{-k}) < \alpha$ {k may be 0, in which case the following loop never executes}
  for $i = 1$ **to** k **do**
    $T \leftarrow T \cup \{(3^i + 1, 1 + 3i)\}$
  end for
  return T

Algorithm 2 Compute a covering set $\mathcal{S}_{k+1}$ for $B_{(k+1)\alpha}$ from covering sets $\mathcal{S}_1, \ldots, \mathcal{S}_k$ for $B_\alpha, \ldots, B_{k\alpha}$

Require: $k \in \mathbb{N}$, $\alpha \in (0,1) \cap R$, $\mathcal{S}_i$ a covering set for $B_{i\alpha}$ for $1 \leq i \leq k$
Ensure: $\mathcal{S}_{k+1}$ a covering set for $B_{(k+1)\alpha}$
  for all $i = 1$ **to** k **do**
    $\mathcal{S}'_i \leftarrow \mathcal{S}_i \setminus \{(1,1),(3,3)\}$
  end for
  $\mathcal{S}_{k+1} \leftarrow \emptyset$
  Compute the set T_α, and the complexities of its elements; let U be the set $\{(n, \|n\|) : n \in T_\alpha\}$ {One may use instead a superset of T_α if determining T_α exactly takes too long}
  Compute the set $V_{k,\alpha} = \{n : \|n\| < (k+1)\alpha + 3\log_3 2\}$ {Again, one may use a superset}
  if $k = 1$ **then**
    $\mathcal{S}_{k+1} \leftarrow \mathcal{S}_{k+1} \cup \{(f_1 \otimes f_2 \otimes f_3, C_1 + C_2 + C_3) : (f_\ell, C_\ell) \in \mathcal{S}'_1\}$
    $\mathcal{S}_{k+1} \leftarrow \mathcal{S}_{k+1} \cup \{(f_1 \otimes f_2, C_1 + C_2) : (f_\ell, C_\ell) \in \mathcal{S}'_1\}$
  else
    $\mathcal{S}_{k+1} \leftarrow \mathcal{S}_{k+1} \cup \{(f \otimes g, C + D) : (f, C) \in \mathcal{S}'_i, (g, D) \in \mathcal{S}'_j, i + j = k + 2\}$
  end if
  $\mathcal{S}_{k+1} \leftarrow \mathcal{S}_{k+1} \cup \{(f \otimes x + b, C + \|b\|) : (f, C) \in \mathcal{S}_{k\alpha}, b \in V_{k,\alpha}\}$
  $\mathcal{S}_{k+1} \leftarrow \mathcal{S}_{k+1} \cup \{(g \otimes (f \otimes x + b), C + D + \|b\|) : (f, C) \in \mathcal{S}_{k\alpha}, b \in V_{k,\alpha}, (g, D) \in \mathcal{S}'_1\}$
  $\mathcal{S}_{k+1} \leftarrow \mathcal{S}_{k+1} \cup U$
  $\mathcal{S}_{k+1} \leftarrow \mathcal{S}_{k+1} \cup \{(f \otimes g, C \otimes D) : f \in U, g \in \mathcal{S}'_1\}$
  return $\mathcal{S}_{k+1}$

$(3f, C+3)$, then either the representation as $3 \cdot n$ is most-efficient or it is not. If it is, then $3n$ is not a leader, and so not in any $B_{i\alpha}$, and thus we do not need it to be 3-represented. If it is not, then it is not efficiently 3-represented by $(3f, C+3)$. So these particular pairs do not need to be multiplied by, and the algorithm still works. □

5.6.3 Algorithm 3: Computing a covering set for B_r.

We can now put the two of these together to form Algorithm 3, found on page 162, for computing a covering set for B_r. If we look ahead to Algorithm 5, we can turn it into a good covering.

Algorithm 3 Compute a covering set for B_r

Require: $r \in R, r \geq 0$
Ensure: S is a covering set for B_r
  Choose a step size $\alpha \in (0,1) \cap R$
  Let T_1 be the output of Algorithm 1 for α {This is a good covering of B_α}
  for $k = 1$ **to** $\lceil \frac{r}{\alpha} \rceil - 1$ **do**
    Use Algorithm 2 to compute a covering set T_{k+1} for $B_{(k+1)\alpha}$ from our covering sets T_i for $B_{i\alpha}$
    Optional step: Do other things to T_{k+1} that continue to keep it a covering set for $B_{(k+1)\alpha}$ while making it more practical to work with. For instance, one may use Algorithm 5 to turn it into a good covering of $B_{(k+1)\alpha}$, or one may remove elements of T_{k+1} that are redundant (i.e., if one has (f, C) and (g, D) in T_{k+1} such that any n which is efficiently 3-represented by (f, C) is also efficiently represented by (g, D), one may remove (f, C))
  end for
  $S \leftarrow T_{k+1}$
  return S

Proof of correctness for Algorithm 3. Assuming the correctness of Algorithm 1 and Algorithm 2, the correctness of Algorithm 3 follows immediately. Again, this is just making use of Theorem 5.2.21, following the proof of Theorem 5.2.22 (Theorem 3.4.10). □

5.6.4 Algorithm 4: Truncating a polynomial to a given defect.

That completes the "building-up" half of the method. For the "filtering-down" half, we start with Algorithm 4, found on page 163, for truncating a given polynomial to a given defect:

Algorithm 4 Truncate the low-defect pair (f, C) to the defect s

Require: (f, C) is a low-defect pair, $s \in R$

Ensure: T is a set of low-defect pairs, obtained by 3-substituting into (f, C) all the elements of a set S satisfying the conclusions of Theorem 5.5.8

```
if deg f = 0 then
  if δ(f, C) < s then
    T ← {(f, C)}
  else
    T ← ∅
  end if
else
  if δ(f, C) ≤ s then
    T ← {(f, C)}
  else
    Find the smallest K for which δ_{f,C}(k_1, ..., k_r) ≥ s, where k_i = K + 1 if x_i is
    minimal in the nesting ordering and x_i = 0 otherwise
    T ← ∅
    for all x_i a minimal variable, k ≤ K do
      Let (g, D) be the 3-substitution of (∗, ..., ∗, k_i, ∗, ..., ∗) into (f, C)
      Recursively apply Algorithm 4 to (g, D) and s to obtain a set T′
      T ← T ∪ T′
    end for
  end if
end if
return S
```

Proof of correctness for Algorithm 4. This algorithm is simply an algorithmic version of the method described in the proof of Theorem 5.5.8, except that instead of producing a set S of tuples which can be 3-substituted into (f, C), it directly produces the 3-substitutions; it performs the substitutions described in the inductive step of Theorem 5.5.8 and just leaves them substituted instead of pulling them back to find tuples which can be 3-substituted in to yield them. □

5.6.5 Algorithm 5: Truncating many polynomials to a given defect.

And, as before, if we can truncate one polynomial, we can truncate many of them (Algorithm 5):

Algorithm 5 Compute a good covering of B_r from a covering set for B_r

Require: $r \in R$, $r \geq 0$, $\mathcal{T}$ a covering set for B_r
Ensure: $\mathcal{S}$ is a good covering of B_r
  $\mathcal{S} \leftarrow \emptyset$
  for all $(f, C) \in \mathcal{T}$ **do**
    Use Algorithm 4 to truncate (f, C) to r; call the result $\mathcal{S}'$
    $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{S}'$
  end for
  return $\mathcal{S}$

Proof of correctness for Algorithm 5. Algorithm 2 is exactly the method described in the proof of Theorem 5.5.9. It can also be seen as an application of the correctness of Algorithms 2 and 4. □

5.6.6 Algorithm 6: Computing a good covering of B_r.

We can then put this all together into Algorithm 6, for computing a good covering of B_r:

Algorithm 6 Compute a good covering of B_r

Require: $r \in R$, $r \geq 0$
Ensure: $\mathcal{S}$ is a good covering of B_r
  Use Algorithm 3 to compute a covering set $\mathcal{T}$ for B_r
  Use Algorithm 5 to compute a good covering $\mathcal{S}$ for B_r from $\mathcal{T}$
  return $\mathcal{T}$

Proof of correctness for Algorithm 6. This follows immediately from the correctness of Algorithms 3 and 5. □

We've now described how to compute good coverings of B_r. But it still remains to show how to use this to compute other quantities of interest, such as $K(n)$ and $\|n\|_{st}$. We address this in the next section.

5.7 Algorithms: Computing stabilization length $K(n)$ and stable complexity $\|n\|_{st}$

In order to compute $K(n)$ and $\|n\|_{st}$, we're going to have to be able to tell algorithmically whether, given a low-defect polynomial f and a number n, there exists $k \geq 0$ such that f 3-represents $3^k n$. If we simply want to know whether f 3-represents n, this is easy; because

$$f(3^{k_1}, \ldots, 3^{k_r}) \geq 3^{k_1 + \ldots + k_r},$$

we have an upper bound on how large the k_i can be and we can solve this with brute force. However, if we want to check whether it represents $3^k n$ for any k, clearly this will not suffice, as there are infinitely many possibilities for k. We will need a lemma to narrow them down:

Lemma 5.7.1. *Let f be a polynomial in r variables with nonnegative integer coefficients and nonzero constant term; write*

$$f(x_1, \ldots, x_r) = \sum a_{i_1,\ldots,i_r} x_1^{i_1} \ldots x_r^{i_r}$$

with $a_{i_1,\ldots,i_r}$ positive integers and $a_{0,\ldots,0} > 0$. Let $b > 1$ be a natural number and let $v_b(n)$ denote the number of times n is divisible by b. Then for any $k_1, \ldots, k_r \in \mathbb{Z}_{\geq 0}$, we have

$$v_b(f(b^{k_1}, \ldots, b^{k_r})) \leq \sum_{a_{i_1,\ldots,i_r} > 0} (\lfloor \log_b a_{i_1,\ldots,i_r} \rfloor + 1) - 1.$$

In particular, this applies when f is a low-defect polynomial and $b = 3$.

Proof. The number $f(b^{k_1}, \ldots, b^{k_r})$ is the sum of the constant term $a_{0,\ldots,0}$ (call it simply A_0) and numbers of the form $A_i b^{\ell_i}$ where the A_i are simply the remaining $a_{i_1,\ldots,i_r}$ enumerated in some order (say $1 \leq i \leq s$). Since we can choose the order, assume that $v_b(A_1 b^{\ell_1}) \leq \ldots \leq v_b(A_s b^{\ell_s})$.

So consider forming the number $f(b^{k_1}, \ldots, b^{k_r})$ by starting with A_0 and adding in the numbers $A_i b^{\ell_i}$ one at a time. Let S_i denote the sum $\sum_{j=0}^{i} A_j b^{\ell_j}$, so $S_0 = A_0$ and

$S_s = f(b^{k_1}, \ldots, b^{k_r})$. We check that for any i, we have

$$v_b(S_i) \leq \sum_{j=0}^{i} (\lfloor \log_b A_j \rfloor + 1) - 1. \tag{5.1}$$

Before proceeding further, we observe that if for some i we have $v_b(A_{i+1}b^{\ell_{i+1}}) > v_b(S_i)$, then by assumption, for all $j > i$, $v_b(A_j b^{\ell_j}) \geq v_b(A_{i+1}b^{\ell_{i+1}}) > v_b(S_i)$. Now in general, if $v_b(n) < v_b(m)$, then $v_b(n+m) = v_b(n)$. So we can see by induction that for all $j \geq i$, $v_b(S_j) = v_b(S_i)$: This is true for $j = i$, and if it is true for j, then $v_b(S_j) = v_b(S_i) < v_b(A_j b^{\ell_j})$ and so $v_b(S_{j+1}) = v_b(S_i)$.

So let h be the smallest i such that $v_b(A_{i+1}b^{\ell_{i+1}}) > v_b(S_i)$. (If no such i exists, take $h = s$.) Then we first prove that (5.1) holds for $i \leq h$.

In the case that $i \leq h$, we will in fact prove the stronger statement that

$$\lfloor \log_b S_i \rfloor \leq \sum_{j=0}^{i} (\lfloor \log_b A_j \rfloor + 1) - 1;$$

this is stronger as in general it is true that $v_b(n) \leq \lfloor \log_b n \rfloor$. For $i = 0$ this is immediate. So suppose that this is true for i and we want to check it for $i+1$, with $i+1 \leq h$. Since $i+1 \leq h$, we have that $v_b(A_{i+1}b^{\ell_{i+1}}) \leq v_b(S_i)$. From this we can conclude the inequality

$$\begin{aligned} \lfloor \log_b(A_{i+1}b^{\ell_{i+1}}) \rfloor &= \ell_{i+1} + \lfloor \log_b A_{i+1} \rfloor \\ \leq v_b(A_{i+1}b^{\ell_{i+1}}) + \lfloor \log_b A_{i+1} \rfloor &\leq v_b(S_i) + \lfloor \log_b A_{i+1} \rfloor. \end{aligned}$$

Now, we also know that

$$\lfloor \log_b S_{i+1} \rfloor \leq \max\{\lfloor \log_b S_i \rfloor, \lfloor \log_b(A_{i+1}b^{\ell_{i+1}}) \rfloor\} + 1. \tag{5.2}$$

And we can observe using above that

$$\lfloor \log_b(A_{i+1}b^{\ell_{i+1}}) \rfloor + 1 \leq \lfloor \log_b S_i \rfloor + \lfloor \log_b A_{i+1} \rfloor + 1 \leq \sum_{j=0}^{i+1} (\lfloor \log_b A_j \rfloor + 1) - 1.$$

We also know that

$$\lfloor \log_b S_i \rfloor + 1 \leq \sum_{j=0}^{i} (\lfloor \log_b A_j \rfloor + 1) \leq \sum_{j=0}^{i+1} (\lfloor \log_b A_j \rfloor + 1) - 1,$$

as $\lfloor \log_b A_{i+1} \rfloor + 1 \geq 1$. So we can conclude using (5.2) that

$$\lfloor \log_b S_{i+1} \rfloor \leq \sum_{j=0}^{i+1} (\lfloor \log_b A_j \rfloor + 1) - 1,$$

as desired.

Having proved (5.1) for $i \leq h$, we then immediately obtain it for all i, as by the above, for $i \geq h$,

$$v_b(S_i) = v_b(S_h) \leq \sum_{j=0}^{h} (\lfloor \log_b A_j \rfloor + 1) - 1 \leq \sum_{j=0}^{s} (\lfloor \log_b A_j \rfloor + 1) - 1;$$

this proves the claim. □

5.7.1 Algorithm 7: Computing whether a polynomial 3-represents some $3^k n$.

With this in hand, we can now write down Algorithm 7, found on page 167, for determining if f 3-represents any $3^k n$:

Algorithm 7 Determine whether (f, C) 3-represents any $3^k n$ and with what complexities

---

Require: (f, C) a low-defect pair, n a natural number
Ensure: S is the set of (k, ℓ) such that there exist whole numbers $(k_1, \ldots, k_r)$ with $f(3^{k_1}, \ldots, 3^{k_r}) = 3^k n$ and $C + 3(k_1 + \ldots + k_r) = \ell$
  $S \leftarrow \emptyset$
  Determine v such that for any $k_1, \ldots, k_r$, one has $v_3(f(3^{k_1}, \ldots, 3^{k_r})) \leq v$ {one method is given by Lemma 5.7.1}
  for $i = 0$ **to** $v - v_3(n)$ **do**
    for all $(k_1, \ldots, k_r)$ such that $k_1 + \ldots + k_r \leq \lfloor \log_3 n \rfloor$ **do**
      if $f(3^{k_1}, \ldots, 3^{k_r}) = n$ **then**
        $S \leftarrow S \cup \{(k, C + 3(k_1 + \ldots + k_r))\}$
      end if
    end for
  end for
  return S

Proof of correctness for Algorithm 7. Once we have picked a v (which can be found using Lemma 5.7.1), it suffices to check if f represents $3^k n$ with $k + v_3(n) \leq v$. By

Proposition 5.2.8, for any $k_1, \ldots, k_r$, we have

$$f(3^{k_1}, \ldots, 3^{k_r}) \geq 3^{k_1+\ldots+k_r},$$

and so it suffices to check it for tuples $(k_1, \ldots, k_r)$ with $k_1 + \ldots + k_r \leq \lfloor \log_3 n \rfloor$. There are only finitely many of these and so this can be done by brute force, and this is exactly what the algorithm does. □

5.7.2 Algorithm 8: Algorithm to test stability and compute stable complexity

Now, at last, we can write down Algorithm 8, found on page 169, for computing $K(n)$ and $\|n\|_{st}$. We assume that in addition to n, we are given L, an upper bound on $\|n\|$, which may be ∞. Running Algorithm 8 with $L = \infty$ is always a valid choice; alternatively, one may compute $\|n\|$ or an upper bound on it before applying Algorithm 8.

Proof of correctness for Algorithm 8. This algorithm progressively builds up good covers $\mathcal{S}_i$ of $B_{i\alpha}$ until it finds some i such that there is some $(f, C) \in \mathcal{S}_i$ such that $\hat{f}$ 3-represents $3^k n$ for some $k \geq 0$. To see that this is indeed what it is doing, observe that if

$$f(3^{k_1}, \ldots, 3^{k_r})3^{k_{r+1}} = 3^k n,$$

then if $k \geq k_{r+1}$, we may write

$$f(3^{k_1}, \ldots, 3^{k_r}) = 3^{k-k_{r+1}} n$$

and so f itself 3-represents some $3^k n$, while if $k \leq k_{r+1}$, we may write

$$f(3^{k_1}, \ldots, 3^{k_r})3^{k_{r+1}-k} = n$$

and so $\hat{f}$ 3-represents n itself. And this is exactly what the inner loop does; it checks if f 3-represents any $3^k n$ using Algorithm 7, and it checks if $\hat{f}$ 3-represents n using brute force.

Now, if for a given i we obtain $U = \emptyset$, then that means that no $3^k n$ is 3-represented by any $(f, C) \in \mathcal{S}_i$, and so for any k, $\delta(3^k n) \geq i\alpha$, that is, $\delta_{st}(3^k n) \geq i\alpha$. Conversely, if for a given i we obtain U nonempty, then that means that some $3^k n$ is 3-represented by some $(f, C) \in \mathcal{S}_i$. Since for any (f, C) we have $\delta(f, C) \leq i\alpha$ (and this is strict if $\deg f = 0$), this means that $\delta(3^k n) < i\alpha$, and so $\delta_{st}(n) < i\alpha$.

Algorithm 8 Compute $K(n)$ and $\|n\|_{st}$

Require: n a natural number, $L \in \mathbb{N} \cup \{\infty\}$, $L \geq \|n\|$
Ensure: $(k, m) = (K(n), \|n\|_{st})$
  Choose a step size $\alpha \in (0, 1) \cap R$
  Let r be the smallest nonnegative integer, or ∞, such that $r\alpha > L - 3\log_3 n - 1$
  $i \leftarrow 1$
  $U \leftarrow \emptyset$
  while $U = \emptyset$ **and** $i \leq r$ **do**
    if $i = 1$ **then**
      Let $\mathcal{S}_1$ be the output of Algorithm 1 for α {This is a good covering of B_α}
    else
      Use Algorithm 2 to compute a covering $\mathcal{S}_i$ of $B_{i\alpha}$ from coverings $\mathcal{S}_j$ of $B_{j\alpha}$ for $1 \leq j < i$
      Use Algorithm 5 to turn $\mathcal{S}_i$ into a good covering
    end if
    Optional step: Remove redundancies from $\mathcal{S}_i$ as in Algorithm 2 {See "optional step" there}
    for all $(f, C) \in \mathcal{S}_i$ **do**
      Let U' be the output of Algorithm 7 on (f, C) and n {If r is finite and $i < r$ this whole loop may be skipped}
      for all $(k_1, \ldots, k_{r+1})$ such that $k_1 + \ldots + k_{r+1} \leq \lfloor \log_3 n \rfloor$ **do**
        if $\hat{f}(3^{k_1}, \ldots, 3^{k_{r+1}}) = n$ **then**
          $U' \leftarrow U' \cup \{(k, C + 3(k_1 + \ldots + k_{r+1}))\}$
        end if
      end for
      $U \leftarrow U \cup U'$
    end for
  end while
  if $U = \emptyset$ **then**
    $(k, m) = (0, L)$
  else
    Let V consist of the elements (k, ℓ) of U that minimize $\ell - 3k$
    Choose $(k, \ell) \in V$ that minimizes k
    $p \leftarrow \ell - 3k$
  end if
  return (k, m)

So we see that the algorithm if the algorithm exits the main loop with U nonempty, it does so once has found some i such that there exists k with $\delta(3^k n) < i\alpha$; equivalently, once it has found some i such that $\delta_{st}(n) < i\alpha$. Or, equivalently, once it has found some i such that $\delta(3^{K(n)}n) < i\alpha$. Furthermore, note that $3^{K(n)}n$ must be a leader if $K(n) > 0$, as otherwise $3^{K(n)-1}n$ would also be stable. So if $K(n) > 0$, then $3^{K(n)}n$ must be efficiently 3-represented by some $(f, C) \in \mathcal{S}_i$. Whereas if $K(n) = 0$, then we only know that it is efficiently 3-represented by some $(\hat{f}, C)$ for some $(f, C) \in \mathcal{S}_i$, but we also know $3^{K(n)}n = n$. That is to say, the ordered pair $(K(n), \|3^{K(n)}n\|)$ must be in the set U.

In this case, where U is nonempty, it remains to examine the set U and pick out the correct candidate. Each pair $(k, \ell) \in U$ consists of some k and some ℓ such that $\ell \geq \|3^k n\|$. This implies that

$$\delta_{st}(n) \leq \delta(3^k n) \leq \ell - 3k - 3\log_3 n,$$

and so the pair $(K(n), \|3^{K(n)}n\|)$ must be a pair (k, ℓ) for which the quantity $\ell - 3k - 3\log_3 n$, and hence the quantity $\ell - 3k$, is minimized; call this latter minimum p. So

$$\delta_{st}(n) = p - 3\log_3 n.$$

(Note that this means that $p = \|n\|_{st}$.) Then the elements of V are pairs $(k, p + 3k)$ with

$$\delta(3^k n) \leq p - 3\log_3 n,$$

but we know also that

$$\delta(3^k n) \geq \delta_{st}(n) = p - 3\log_3 n,$$

so we conclude that for such a pair, $\delta(3^k n) = \delta_{st}(n)$. But this means that $3^k n$ is stable, and so $k \geq K(n)$. But we know that $K(n)$ is among the set of k with $(k, p+3k) \in V$, and so it is their minimum. Thus, we can select the element $(k, \ell) \in V$ that minimizes k; then $k = K(n)$, and we can take $k - 3\ell$ to find $m = \|n\|_{st}$.

This leaves the case where U is empty. In this case, we must have that for all i with $1 \leq i \leq r$, and hence in particular for $i = r$, no (f, C) in $\mathcal{S}_i$ 3-represents any $n3^k$; i.e., no $n3^k$ lies in $B_{r\alpha}$, and hence, by Proposition 5.2.17, no $n3^k$ lies in $A_{r\alpha}$. That is to say, for any k, $\delta(n3^k) \geq r\alpha$, and so

$$\|n3^k\| \geq r\alpha + 3\log_3 n + 3k > L + 3k - 1.$$

Since $\|n3^k\| > L + 3k - 1$, and $\|n3^k\| \leq L + 3k$, we must have $\|n3^k\| = L + 3k$. Since this is true for all $k \geq 0$, we can conclude that n is a stable number. So, n is stable and $\|n\| = L$, that is to say, $K(n) = 0$ and $\|n\|_{st} = \|n\| = L$. □

We have now proven Theorem 5.1.7:

Proof of Theorem 5.1.7. Algorithm 8, run with $L = \infty$, gives us a way of computing $K(n)$ and $\|n\|_{st}$. Then, to check if n is stable, it suffices to check whether or not $K(n) = 0$. This proves the theorem. □

5.7.3 Algorithm 9: Determining leaders and the "drop pattern".

But we're not done; we can go further. As mentioned in Section 5.1.2, we can get more information if we go until we detect n, rather than stopping as soon as we detect some $3^k n$. We now record Algorithm 9, found on page 172, for not only determining $K(n)$ and $\|3^{K(n)}n\|$, but for determining all k such that either $k = 0$ or $3^k n$ is a leader, and the complexities $\|3^k n\|$. By Proposition 5.2.17, this is enough to determine $\|3^k n\|$ for all $k \geq 0$. One could also do this by using Algorithm 8 to determine $K(n)$ and then directly computing $\|3^k n\|$ for all $k \leq K(n)$, but Algorithm 9 will often be faster.

Proof of correctness for Algorithm 9. As in Algorithm 8, we are successively building up good coverings $\mathcal{S}_i$ of $B_{i\alpha}$, and for each one checking whether there is an $(f, C) \in \mathcal{S}_i$ and a $k \geq 0$ such that $(\hat{f}, C)$ 3-represents $3^k n$. However, the exit condition on the loop is different; ignoring for a moment the possibility of exiting due to $i > r$, the difference is that instead of stopping once some $3^k n$ is 3-represented, we do not stop until n itself is 3-represented, or equivalently, $\delta(n) < i\alpha$. We'll use i here to denote the value of i when the loop exits.

We want the set U to have two properties: Firstly, it should contain all the pairs (k, ℓ) we want to find. Secondly, for any $(k, \ell) \in U$, we should have $\|3^k n\| \leq \ell$. For the first property, observe that if $3^k n$ is a leader and $k > 1$, then

$$\delta(3^k n) \leq \delta(n) - 1 < L - 3\log_3 n - 1,$$

and so $\delta(3^k n) \leq r\alpha$; thus, $3^k n$ (being a leader) is efficiently 3-represented by some $(f, C) \in \mathcal{S}_r$, and so if the loop exits due to $i > r$, then $(k, \|3^k n\|) \in U$. Whereas if the loop exits due to $0 \in \pi_1(U)$, then note $\delta(3^k n) \leq \delta(n) < i\alpha$, and so $3^k n$ (again being a leader) is efficiently 3-represented by some $(f, C) \in S_i$, and so again $(k, \|3^k n\|) \in U$.

Algorithm 9 Compute information determining $\|3^k n\|$ for all $k \geq 0$

Require: n a natural number, $L \in \mathbb{N} \cup \{\infty\}$, $L \geq \|n\|$
Ensure: V the set of (k, ℓ) where either $k = 0$ or $k > 0$ and $3^k n$ is a leader, and $\ell = \|3^k n\|$
  Choose a step size $\alpha \in (0, 1) \cap R$
  Let r be the smallest nonnegative integer, or ∞, such that $r\alpha > L - 3\log_3 n - 1$
  $i \leftarrow 1$
  $U \leftarrow \emptyset$
  while $0 \notin \pi_1(U)$, where π_1 is projection onto the first coordinate, **and** $i \leq r$ **do**
    if $i = 1$ **then**
      Let $\mathcal{S}_1$ be the output of Algorithm 1 for α {This is a good covering of B_α}
    else
      Use Algorithm 2 to compute a covering $\mathcal{S}_i$ of $B_{i\alpha}$ from coverings $\mathcal{S}_j$ of $B_{j\alpha}$ for $1 \leq j < i$
      Use Algorithm 5 to turn $\mathcal{S}_i$ into a good covering
    end if
    Optional step: Remove redundancies from $\mathcal{S}_i$ as in Algorithm 2 {See "optional step" there}
    for all $(f, C) \in \mathcal{S}_i$ **do**
      Determine v such that for any $k_1, \ldots, k_r$, one has $v_3(f(3^{k_1}, \ldots, 3^{k_r})) \leq v$ {one method is given by Lemma 5.7.1} {If r is finite and $i < r$ this whole loop may be skipped}
      Let U' be the output of Algorithm 7 on (f, C) and n
      for all $(k_1, \ldots, k_{r+1})$ such that $k_1 + \ldots + k_{r+1} \leq \lfloor \log_3 n \rfloor$ **do**
        if $\hat{f}(3^{k_1}, \ldots, 3^{k_{r+1}}) = n$ **then**
          $U' \leftarrow U' \cup \{(k, C + 3(k_1 + \ldots + k_{r+1}))\}$
        end if
      end for
      $U \leftarrow U \cup U'$
    end for
  end while
  if $0 \notin \pi_1(U)$ **then**
    $U \leftarrow U \cup \{(0, L)\}$
  end if
  Let $V = \{(k, \ell - 3k) : (k, \ell) \in U\}$
  Let V_m consist of the minimal elements of V in the usual partial order
  Let $W = \{(k, p + 3k) : (k, p) \in V_m\}$
  return W

This leaves the case where $k = 0$. If the loop exits due to $0 \in \pi_1(U)$, then by choice of i, n is efficiently 3-represented by some $(\hat{f}, C)$ for some $(f, C) \in \mathcal{S}_i$, so $(0, \|n\|) \in U$. Whereas if the loop exits due to $i > r$, then this means that $\delta(n) \geq r\alpha$, and so

$$\|n\| \geq r\alpha + 3\log_3 n > L - 1;$$

since we know $\|n\| \leq L$, this implies $\|n\| = L$, and so including $(0, L)$ in U means $(0, \|n\|) \in U$.

For the second property, again, there are two ways a pair (k, ℓ) may end up in U. One is that some low-defect pair (f, C) 3-represents the number $3^k n$, which, as in the proof of correctness for Algorithm 8, means $\|3^k n\| \leq \ell$. The other is that $(k, \ell) = (0, L)$; but in this case, $\|n\| \leq L$ by assumption.

It then remains to isolate the pairs we want from the rest of U. We will show that they are in fact precisely the minimal elements of U under the partial order

$$(k_1, \ell_1) \leq (k_2, \ell_2) \iff k_1 \leq k_2 \text{ and } \ell_1 - 3k_1 \leq \ell_2 - 3k_2.$$

Say first that (k, ℓ) is one of the pairs we are looking for, i.e, either $k = 0$ or $3^k n$ is a leader, and $\ell = \|3^k n\|$. Now suppose that that $(k', \ell') \in U$ such that $k' \leq k$ and $\ell - 3k' \leq \ell - 3k$. Since $(k', \ell') \in U$, that means that $\|3^{k'} n\| \leq \ell'$. Since $k' \leq k$, we conclude that

$$\ell = \|3^k n\| \leq \ell' + 3(k - k') \tag{5.3}$$

and hence that $\ell - 3k \leq \ell' - 3k'$, so $\ell - 3k = \ell' - 3k'$. Now, if $k = 0$, then certainly $k \leq k'$ (and so $k = k'$); otherwise, $3^k n$ is a leader. Suppose we had $k' < k$; then since $3^k n$ is a leader, that would mean $\delta(3^k n) < \delta(3^{k'} n)$ and hence

$$\|3^k n\| < \|3^{k'} n\| + 3(k - k') = \ell + 3(k - k'),$$

contrary to 5.3. So we conclude $k' = k$, and so (k, ℓ) is indeed minimal.

Conversely, suppose that (k, ℓ) is a minimal element of U in this partial order. We must show that $\ell = \|3^k n\|$, and, if $k > 0$, that $3^k n$ is a leader. Choose $k' \leq k$ as large as possible with either $k' = 0$ or $3^{k'} n$ a leader, so that $\delta(3^{k'} n) = \delta(3^k n)$. Also, let $\ell' = \|3^{k'} n\|$; by above, $(k', \ell') \in U$. Since $(k, \ell) \in U$ and $\delta(3^{k'} n) = \delta(3^k n)$, we know that

$$\|3^{k'} n\| + 3(k - k') = \|3^k n\| \leq \ell$$

and hence $\ell' - 3k' \leq \ell - 3k$. Since by assumption we also have $k' \leq k$, by the

assumption of minimality we must have $(k', \ell') = (k, \ell)$. But this means exactly that either $k = 0$ or $3^k n$ is a leader, and that

$$\|3^k n\| = \|3^{k'} n\| = \ell' = \ell,$$

as needed. □

5.7.4 Algorithm 10: Stabilization length and stable complexity for $n = 2^k$.

Finally, before moving on to the results of applying these algorithms, we make note of one particular specialization of Algorithm 8, namely, the case where $n = 2^k$ and $\ell = 2k$. As was noted in Section 5.1.3, this turns out to be surprisingly fast as a method of computing $\|2^k\|$. We formalize it here:

Algorithm 10 Given $k \geq 1$, determine $K(2^k)$ and $\|2^k\|_{st}$

Require: $k \geq 1$ an integer
Ensure: $(h, p) = (K(2^k), \|2^k\|_{st})$
  Let (h, p) be the result of applying Algorithm 8 with $n = 2^k$ and $L = 2k$.
  return (h, p)

Proof of correctness for Algorithm 10. The correctness of Algorithm 10 follows from the correctness of Algorithm 8 and the fact that $\|2^k\| \leq 2k$ for $k \geq 1$. □

5.8 Results of computation

Armed with our suite of algorithms, we now proceed to the results of our computations. We can use Algorithm 10 to prove Theorem 5.1.2:

Proof of Theorem 5.1.2. Algorithm 10 was applied with $k = 48$, and it was determined that $K(2^{48}) = 0$ and $\|2^{48}\|_{st} = 96$, that is to say, that 2^{48} is stable and $\|2^{48}\| = 96$, that is to say, that $\|2^{48}3^\ell\| = 96 + 3\ell$ for all $\ell \geq 0$. This implies that $\|2^k 3^\ell\| = 2k + 3\ell$ for all $k \leq 48$ and $\ell \geq 0$ with k and ℓ not both zero, as if one instead had $\|2^k 3^\ell\| < 2k + 3\ell$, then writing $2^{48}3^\ell = 2^{48-k}(2^k 3^\ell)$, one would obtain $2^{48}3^\ell < 96 + 3\ell$. □

But we can do more with these algorithms than just straightforward computation of values of complexities and stable complexities. For instance, we can answer the question: What is the smallest unstable defect other than 1?

In Chapter 2 it was determined that

Theorem 5.8.1. *For any $n > 1$, if $\delta(n) < 12\delta(2)$, then n is stable.*

That is to say, with the exception of 1, all defects less than $12\delta(2)$ are stable. This naturally leads to the question, what is the smallest unstable defect (other than 1)? We might also ask, what is the smallest unstable number (other than 1)? Interestingly, among unstable numbers greater than 1, the number 107 turns out to be smallest both by magnitude and by defect. However, if we measure unstable numbers (other than 1) by their unstable defect, the smallest will instead turn out to be 683. We record this in the following theorem:

Theorem 5.8.2. *We have:*

1. *The number* 107 *is the smallest unstable number other than* 1*.*
2. *Other than* 1*, the number* 107 *is the unstable number with the smallest defect, and $\delta(107) = 3.2398\ldots$ is the smallest unstable defect other than* 1*.*
3. *Among nonzero values of $\delta_{st}(n)$ for unstable n, the number*

$$\delta_{st}(683) = \delta(2049) = 2.17798\ldots$$

is the smallest.

Proof. For part (1), it suffices to use Algorithm 8 to check the stability of all numbers from 2 to 106.

For parts (2) and (3), in order to find unstable numbers of small defect, we will search for leaders of small defect which are divisible by 3. (Since if n is unstable, then $3^{K(n)}n$ is a leader divisible by 3, and $\delta(3^{K(n)}n) < \delta(n)$). We use Algorithm 6 to compute a good covering S of $B_{21\delta(2)}$. Doing a careful examination of the low-defect polynomials that appear, we can determine all the multiples of 3 that each one can 3-represent; we omit this computation, but its results are that the following multiples of 3 can be 3-represented: 3, 6, 9, 12, 15, 18, 21, 24, 27, 30, 33, 36, 39, 42, 45, 48, 54, 57, 60, 63, 66, 72, 75, 78, 81, 84, 90, 96, 111, 114, 120, 126, 129, 132, 144, 162, 165, 168, 171, 180, 192, 225, 228, 231, 240, 252, 258, 264, 288, 321, 324, 330, 336, 360, 384, 480, 513, 516, 528, 576, 768, 1026, 1032, 1056, 1152, 1536, 2049, 2052, 2064, 2112, 2304, 3072, and, for $k \geq 0$, numbers of the forms $12 \cdot 3^k + 3$, $6 \cdot 3^k + 3$, $9 \cdot 3^k + 3$, $12 \cdot 3^k + 6$, and $18 \cdot 3^k + 6$.

For the individual leaders, we can easily check by computation that the only ones which are leaders are 3, 321, and 2049. This leaves the infinite families. For these, observe that if we divide them by 3, we get, respectively, $4 \cdot 3^k + 1$, $2 \cdot 3^k + 1$, $3 \cdot 3^k + 1$, $2(2 \cdot 3^k + 1)$, and $2(3 \cdot 3^k + 1)$, and it is easy to check that any number of any of those forms has defect less than $12\delta(2)$ and hence is stable by Theorem 5.8.1; thus, multiplying them by 3 cannot yield a leader.

So we conclude that the only leaders m with $\delta(m) < 21\delta(2)$ are 3, 321, and 2049. Therefore, the only unstable numbers n with $\delta_{st}(n) < 21\delta(2)$ are 1, 107, and 683. Note also that by the above computation, no power of 3 times any of 3, 321, or 2049 is a leader (as it would have to have smaller defect and would thus appear in the list), and thus the numbers 3, 321, and 2049 are not just leaders but in fact stable leaders. So to prove part (3), it suffices to note that, since $\delta_{st}(3) = 0$, among $\delta_{st}(107)$ (i.e. $\delta(321)$) and $\delta_{st}(683)$ (i.e. $\delta(2049)$), the latter is smaller.

This leaves part (2). Observe that $\delta(107) = \delta(321) + 1$. And if n is unstable, then $\delta_{st}(n) \leq \delta(n) - 1$. So if $n > 1$ is unstable and $\delta(n) < \delta(107)$, then $\delta_{st}(n) < \delta(321)$, which by the above forces $n = 683$. But in fact, although $\delta(2049) < \delta(107)$, we nonetheless have $\delta(683) > \delta(107)$ (because while $\delta(107) = \delta(321) + 1$, $\delta(683) = \delta(2049) + 2$). Thus $\delta(107)$ is the smallest unstable defect other than 1, i.e., 107 is (other than 1) the smallest unstable number by defect. □

These computational results provide a good demonstration of the power of the methods here.

Chapter 6

Open problems and future research

Having now demonstrated that numbers below a fixed defect are sparse; that the set of defects is well-ordered, with order type ω^ω; that an analogous statement holds for addition chains; and that the stable complexity $\|n\|_{st}$ is computable, we now turn our attention to the problems we have not solved. Some of the problems we discuss here have been mentioned already in previous chapters, but they are worth mentioning again.

6.1 Additional structure in the defect set

There seems to be additional structure in $\mathscr{D}$, the set of all defects, and its variants. The well-ordering theorems proved in Chapter 3 can be seen as slightly modified versions of conjectures made earlier by J. Arias de Reyna[8], which are discussed in Appendix A. Here, we present a reformulated version of Arias de Reyna's Conjecture 8 and some related statements.

Let us begin with the "related statements". In particular, the following conjecture:

Conjecture 6.1.1. *We have*

$$\overline{\mathscr{D}} = \overline{\mathscr{D}_{st}} = \mathscr{D} + \mathbb{Z}_{\geq 0} = \mathscr{D}_{st} + \mathbb{Z}_{\geq 0}.$$

This statement may seem a little opaque, so before we continue, let us discuss how these statement may be interpreted.

The new sets worth talking about here are the sets $\mathscr{D} + \mathbb{Z}_{\geq 0}$ and $\mathscr{D}_{st} + \mathbb{Z}_{\geq 0}$. It's easy to see that these two sets are equal, as by the propositions in Section 3.3, for any n the difference $\delta(n) - \delta_{st}(n)$ is an integer. But what do these sets mean? Let us focus on $\mathscr{D} + \mathbb{Z}_{\geq 0}$.

Suppose we have an element $\alpha \in \mathscr{D} + \mathbb{Z}_{\geq 0}$; write $\alpha = \delta(n) + k$, for some natural

number n and some integer $k \geq 0$. (This representation may not be unique.) Then

$$\alpha = (\|n\| + k) - 3\log_3 n.$$

Or, in other words,

$$\alpha = \ell - 3\log_3 n$$

for some $\ell \geq \|n\|$. But to say that $\ell \geq \|n\|$ is to say that ℓ is the number of 1's in some $+, \cdot, 1$-expression for n. So if we consider such expressions E, and denote by $\|E\|$ the number of 1's used in E, and denote by $V(E)$ the value of E, then we can write

$$\mathscr{D} + \mathbb{Z}_{\geq 0} = \{\|E\| - 3\log_3 V(E) : \ E \text{ a } +, \cdot, 1\text{-expression}\}.$$

That is to say, it is the set of defects of *expressions* rather than the set of defects of numbers. Thus, Conjecture 6.1.1 states that *the set of defects of expressions is equal to the closure of the set of defects of numbers.*

It also says furthermore that, in fact, it is equal to the closure of the set of defects of *stable* numbers; it's less clear why this should be so. Nonetheless, computations of good coverings of B_r for small r support this conjecture.

We can also state the following more specific conjecture:

Conjecture 6.1.2. *Moreover, if a is a congrunce class modulo 3, we have*

$$\begin{aligned}\overline{\mathscr{D}^a} = \overline{\mathscr{D}^a_{st}} &= (\mathscr{D}^a + 3\mathbb{Z}_{\geq 0}) \cup (\mathscr{D}^{a-1} + 3\mathbb{Z}_{\geq 0} + 1) \cup (\mathscr{D}^{a-2} + 3\mathbb{Z}_{\geq 0} + 2) \\ &= (\mathscr{D}^a_{st} + 3\mathbb{Z}_{\geq 0}) \cup (\mathscr{D}^{a-1}_{st} + 3\mathbb{Z}_{\geq 0} + 1) \cup (\mathscr{D}^{a-2}_{st} + 3\mathbb{Z}_{\geq 0} + 2)\end{aligned}$$

Conjecture 6.1.2 then says that *the set of defects of expressions with complexity congruent to a modulo 3, is equal to the closure of the set of defects of numbers with complexity congruent to a modulo 3*; this is the same phenomenon, just restricted to one particular congruence class. And, again, the set of stable defects in that class should suffice.

We now move on to a more straightforward reformulation of Arias de Reyna's Conjecture 8:

Conjecture 6.1.3 (Reformulated Arias de Reyna Conjecture 8)**.** *We have*

$$\overline{\mathscr{D}_{st}}' = \overline{\mathscr{D}_{st}} + 1.$$

Moreover, if a is a congruence class modulo 3,

$$\overline{\mathscr{D}_{st}^{a}}' = \overline{\mathscr{D}_{st}^{a-1}} + 1.$$

(Here, for a set S, S' denotes its set of limit points.)

The second of these statements implies the first, but both are worth stating. Let us focus on the first for now, because it is simpler. What does it mean? It tells us that the set $\overline{\mathscr{D}_{st}}$ (and hence $\overline{\mathscr{D}}$, if we believe Conjecture 6.1.1) has a sort of self-similarity property.

Say we look at $\overline{\mathscr{D}_{st}} \cap \{0\}$; it consists of the single point $\{0\}$. Now suppose we look at $\overline{\mathscr{D}_{st}} \cap (0, 1]$; it consists of the point $1 = 0 + 1$, plus a sequence of points α_n tending to 1, for an overall order type of $\omega + 1$. Now suppose we look at $\overline{\mathscr{D}_{st}} \cap (1, 2]$ – it consists of $2 = 1 + 1$, the points $\alpha_n + 1$ tending to 2 – and for each α_n, a sequence of points $\alpha_{n,m}$ tending to α_n as $m \to \infty$. This yields an overall order type of $\omega^2 + 1$.

Of course, we already know that $\overline{\mathscr{D}_{st}} \cap (k-1, k]$ has (for $k \geq 1$ an order type of ω^k, so what's the big deal? Well, this statement tells us something much stronger: that the $\overline{\mathscr{D}_{st}} \cap (k-1, k]$, is *exactly* the set $\overline{\mathscr{D}_{st}} \cap (k-2, k-1]$, except that, firstly, it's been shifted over by 1, and, secondly, each of the old elements has sprouted a new sequence of elements leading up to it! This tells us considerably more than just that the order type has been multiplied by ω; it's been multiplied by ω in a very specific way, by taking the existing elements and "pinning tails on them".

The second statement is similar except it accounts for congruences modulo 3. But notice that the congruence class on the right hand side of the equation, is different from that on the left hand side of the equation; it states not that one set is similar to itself, but that three sets are similar to each other in an intertwined manner. It does, of course, imply that

$$\overline{\mathscr{D}_{st}^{a}}''' = \overline{\mathscr{D}_{st}^{a}} + 3,$$

but the mechanism of this self-similarity passes through two other sets.

Our computations of good coverings for B_r for small values of r support all these conjectures; we may hope that the methods developed here may contribute to their solution.

One may also make analogous conjectures for addition chains:

Conjecture 6.1.4. *We have*

$$\overline{\mathscr{D}^{\ell}} = \overline{\mathscr{D}_{st}^{\ell}} = \mathscr{D}^{\ell} + \mathbb{Z}_{\geq 0} = \mathscr{D}_{st}^{\ell} + \mathbb{Z}_{\geq 0}.$$

Conjecture 6.1.5. *We have*

$$\overline{\mathscr{D}_{st}^{\ell}}' = \overline{\mathscr{D}^{\ell}} + 1.$$

These are a stronger version of part (a) of Conjecture 4.8.1, and seem to be supported by computations so far. Note that there is nothing here about congruence classes; as was noted in Section 1.2.3, those have no analogue for addition chains. One may naturally then make the same conjectures for star chains. As with Conjecture 4.8.1, it's unclear to what other admissible classes of addition chains this might extend.

6.2 Generalization to addition-multiplication chains

One generalization to other settings that has already been mentioned is that of addition-multiplication chains (see Section 1.4.1). Addition-multiplication chains have an an obvious disanalogy with integer complexity and addition chains; whereas $\|n\|$ and $\ell(n)$ are both $\Theta(\log n)$, the lower bound on $\ell_{AM}(n)$ is $\Theta(\log \log n)$ [11], and there is no corresponding $\Theta(\log \log n)$ upper bound [23], as discussed in Section 1.4.1. But then, perhaps this is no more a barrier than the fact that $\ell(n) \sim \log_2 n$, whereas we expect that $\|n\| \nsim 3 \log_3 n$. We can define, for $n > 1$,

$$\delta^{AM}(n) := \ell_{AM}(n) - \log_2 \log_2 n - 1$$

and then define

$$\mathscr{D}^{AM} := \{\delta^{AM}(n) : n \geq 2\}.$$

This then leads to the question:

Question 6.2.1. *Is $\mathscr{D}^{AM}$ well-ordered? Is its order type ω^{ω}? Is the order type of $\mathscr{D}^{AM} \cap [0, k)$ equal to ω^k for $k \geq 1$ an integer?*

Some limited evidence suggests that, at least, $\mathscr{D}^{AM} \cap [0, 1)$ may be well-ordered with order type ω.

6.3 Complexity based on a number other than 1

Another less obvious direction is to consider a generalization of integer complexity based on expressions with addition, multiplication, and a fixed positive real number x which is not necessarily 1. This is, after all, moving things back in the direction of the original Mahler and Popken paper[38]. Now, this is straying from the overall theme of this dissertation a bit, in that this doesn't necessarily yield a sensible notion

of complexity for natural numbers – if x is anything other than 1, one will either not be able to generate all natural numbers in this way, or else will be able to generate numbers that are not natural numbers at all. Nonetheless, it is possible to make this idea sensible by taking a different point of view – a point of view that was taken already in Section 6.1 in explaining the meaning of Conjecture 6.1.1.

Instead of considering the numbers that we are taking the complexity of as the primary thing, let's consider $+, \cdot, x$-expressions. For such an expression E, define $\|E\|$ to be the number of x's used in the expression, and $V(E)$ to be the actual value of the expression when evaluated. Then one can define, for such an expression E,

$$\delta_x(E) = \|E\| - \log_{\theta(x)} V(E)$$

where

$$\theta(x) := \max_{k \in \mathbb{N}} \sqrt[k]{kx}$$

as in [38]. Then one can define $\mathscr{D}_x$ to be the set of all $\delta_x(E)$, for all $+, \cdot, x$-expressions E. Note that $\mathscr{D}_x$ is analogous not to the set $\mathscr{D}$, but rather to the set $\mathscr{D} + \mathbb{Z}_{\geq 0}$, as described in Section 6.1. Then we can once again ask:

Question 6.3.1. *Is $\mathscr{D}_x$ well-ordered? Is its order type ω^ω? Is the order type of $\mathscr{D}_x \cap [0, k)$ equal to ω^k for $k \geq 1$ an integer?*

Some experimental evidence suggests the answer may be "yes", though it is difficult to tell, especially for $x < 1$.

6.4 Further stabilization hypotheses

There are two directions we can go in terms of "further stabilization hypotheses". The more obvious one is to consider what happens when we multiply by powers of numbers other than 3. For instance, does a similar stabilization phenomenon happen with powers of 2? That is to say, for any n, does there exist a K such that for all $k \geq K$, one has

$$\|2^k n\| = 2(k - K) + \|2^K n\|?$$

This question seems like it would be very difficult, though; an affirmative answer would suffice to imply

$$\|2^k\| = 2k$$

for $k \geq 1$, which is already out of reach.

A stabilization hypothesis which seems more approachable is J. Arias de Reyna's Conjecture 2 from [8]. The conjecture has some minor oversights as stated there, so we state a slightly different version, which we also strengthen slightly:

Conjecture 6.4.1. *Let a be a natural number, and suppose that a is stable. Then, for sufficiently large k,*

$$\|a3^k + 1\| = \|a\| + 3k + 1,$$

and $a3^k + 1$ is stable.

Moreover, let a and b be natural numbers, and suppose that ab is stable and that $\|ab\| = \|a\| + \|b\|$. Then, for sufficiently large k,

$$\|b(a3^k + 1)\| = \|a\| + \|b\| + 3k + 1,$$

and $b(a3^k + 1)$ is stable.

This conjecture is in line with our observations made in the process of computing good coverings. It seems to be related to Conjecture 6.1.3. It's possible there may be analogues for low-defect polynomials in more variables, but it's less clear how these would work.

6.5 Instability

Although in Chapter 5 we have given a means to compute $K(n)$ – the stabilization length for integer complexity – we have not provided any explicit upper bound on it. The same is true for the quantity

$$\Delta(n) := \|n\| - \|n\|_{st} = \delta(n) - \delta(n)_{st},$$

which is another way of measuring "how unstable" the number n is, and which, due to the results of Chapter 5, is also now computable. Nor do we have any reliable method of generating unstable numbers with which to demonstrate lower bounds.

Indeed, empirically, large instabilities – measured either by $K(n)$ or by $\Delta(n)$ – seem to be rare. Note that this statement is not based on running the algorithms from Chapter 5 on many numbers to determine their stability, as that is quite slow in general, but rather on simply computing $\|n\|$ for $n \leq 3^{15}$ and then checking $\|n\|$, $\|3n\|$, $\|9n\|$,..., and guessing that n is stable if no instability is detected before the data runs out, a method that can only ever put lower bounds on $K(n)$ and $\Delta(n)$, never upper bounds. Still, numbers that are detectably unstable at all seem to be

Table 6.1: Numbers that seem to have unusual drop patterns. Here, the "drop pattern" of n is the list of values $\delta(3^k n) - \delta(3^{k+1} n)$, or equivalently $\|3^k n\| - \|3^{k+1} n\| + 3$, up until the point where this is always zero. This table is empirical, based on a computation of $\|n\|$ for $n \leq 3^{15}$; it's possible these numbers have later drops further on. Numbers which are divisible by 3 are not listed.

Drop pattern	Numbers with this pattern
4	4721323
$1, 2$	1081079
$2, 1$	203999, 1328219
$1, 0, 0, 1$	$153071, 169199$

somewhat rare, although they still seem to make up a positive fraction of all natural numbers; namely, around 3%. Numbers that are more than merely unstable – having $K(n) \geq 2$ or $\Delta(n) \geq 2$ – are yet rarer.

The largest lower bounds on $K(n)$ or $\Delta(n)$ for a given n encountered based on these computations are $n = 4721323$, which, as mentioned earlier, has $\|3n\| < \|n\|$ and thus $\Delta(n) \geq 4$; and 17 numbers, the smallest of which is $n = 3643$, which have $\|3^5 n\| < \|3^4 n\| + 3$ and thus $K(n) \geq 5$. Finding n where both $K(n)$ and $\Delta(n)$ are decently large is hard; for instance, these computations did not turn up any n for which it could be seen that both $K(n) \geq 3$ and $\Delta(n) \geq 3$. (See Table 6.1 for more.) It's not even clear whether $K(n)$ or $\Delta(n)$ can get arbitrarily large, or are bounded by some finite constant, although there's no clear reason why the latter would be so. Still, this is worth pointing out as a question:

Question 6.5.1. *Is there a natural density of the set of unstable numbers? What is an explicit upper bound on $K(n)$, or on $\Delta(n)$? Can $K(n)$ and $\Delta(n)$ get arbitrarily large, or are they bounded?*

We can also ask the analogous questions regarding addition chain length. Let $K_\ell(n)$ denote the smallest k such that $2^k n$ is ℓ-stable, and define $\Delta_\ell(n)$ by

$$\Delta_\ell(n) := \ell(n) - \ell_{st}(n) = \delta^\ell(n) - \delta^\ell_{st}(n),$$

analogous to our measures of instability for integer complexity. Then we can again ask:

Question 6.5.2. *Is there a natural density of the set of ℓ-unstable numbers? What is an explicit upper bound on $K_\ell(n)$, or on $\Delta_\ell(n)$? Can $K_\ell(n)$ and $\Delta_\ell(n)$ get arbitrarily large, or are they bounded?*

Of course, we still have not even shown that $K_\ell(n)$ and $\Delta_\ell(n)$ are computable; see Section 6.7. Still, the questions make sense all the same. Meanwhile, computations by Neil Clift [18] have found that for $n = 30958077$ we have $\ell(n) = \ell(2n) = \ell(4n)$ (so that $K_\ell(n) \geq 2$ and $\Delta_\ell(n) \geq 2$), and that for $n = 375494703$ we have $\ell(2n) = \ell(n)-1$, so $\Delta_\ell(n) \geq 2$. These are, at present, the only known examples of n for which it is known that $K_\ell(n) \geq 2$ or $\Delta_\ell(n) \geq 2$. So while large instabilities seems to be rare for integer complexity, they may be even more rare for addition chains. For addition chains, however, in contrast to the case of integer complexity, it is known how to find infinitely many unstable numbers; E. G. Thurber [50] showed that for all $k \geq 5$, the number $n = 23 \cdot 2^k + 7$ has $\ell(2n) = \ell(n)$. We can also ask the analogous questions for star chains, Hansen chains, and other admissible sets of addition chains.

6.6 Counting problems

We consider two sorts of counting problems regarding integer complexity. The first is that of refining Theorem 2.6.6, getting better estimates for $A_r(x)$ and $B_r(x)$. Ideally we would like a theorem of the form

$$B_r(x) = C_r(\log_3)^{\lfloor r \rfloor} + O_r((\log x)^{\lfloor r \rfloor - 1})$$
$$A_r(x) = C'_r(\log_3)^{\lfloor r \rfloor + 1} + O_r((\log x)^{\lfloor r \rfloor})$$

for explicit constants C_r and C'_r. (Although it may be preferable to use $\overline{A}_r$ and $\overline{B}_r$ here; see Appdendix B for some examples of theorems that can be phrased more naturally in terms of $\overline{A}_r$ and $\overline{B}_r$ rather than A_r and B_r. In this case, making this switch would at least allow us to not have to exclude the case of $r = 0$.) J. Zelinsky has suggested that, if such estimates can be refined enough, it may be possible to use them to prove that $\|n\| \nsim 3 \log_3 n$.

The other sort of counting problem is that of counting particular types of numbers relevant to integer complexity – how common is it for numbers to be unstable? How common is it for numbers to be m-irreducible, or to be solid (additively irreducible)? Empirically, it seems that all these types of numbers make up a positive fraction of the natural numbers. Specifically, it seems that about 3% of numbers are unstable, about 63% of numbers are solid, and about 18% of numbrs are m-irreducible. On the addition chain side of things, we can also ask how common ℓ-unstable numbers are.

Question 6.6.1. *Is there a natural density of the set of unstable numbers? The set of solid numbers? The set of m-irreducible numbers?*

Question 6.6.2. *Is there a natural density of the set of ℓ-unstable numbers?*

We can also consider the same thing for star chains, Hansen chains, or other admissible sets of addition chains.

6.7 Computability and complexity-theoretic problems

There remains from Chapter 5 the problem of determining the computational complexity of the various functions considered there. As has been noted earlier, the best known algorithm for computing $\|n\|$ (due to Srinivas and Shankar [44]) takes time $O(n^{\log_2 3})$. It is also known[8] that the problem "Given n and k in binary, is $\|n\| \leq k$?" is in the class NP, because the size of a witness is $O(\log n)$. (This problem is not known to be NP-complete.) However, it's not clear whether the problem "Given n and k in binary, is $\|n\|_{st} \leq k$?" is in the class NP, because there's no obvious bound on the size of a witness. It is quite possible that it could be proven to be in NP, however, if an explicit upper bound could be obtained on $K(n)$.

We can also consider the problem of computing the defect ordering, i.e., "Given n_1 and n_2 in binary, is $\delta(n_1) \leq \delta(n_2)$?"; as noted in Chapter 3, this problem lies in Δ_2^P in the polynomial hierarchy. We can similarly consider the problem of the stable defect ordering – "Given n_1 and n_2 in binary, is $\delta_{st}(n_1) \leq \delta_{st}(n_2)$?" Again, due to a lack of bounds on $K(n)$, it's not clear that this lies in Δ_2^P.

We can also ask about the complexity of computing $K(n)$, or $\Delta(n)$ (which, conceivably, could be easier than $\|n\|$ or $\|n\|_{st}$, though this seems unlikely), or, perhaps most importantly, of computing a good covering of B_s for a given $s \geq 0$. Note that in this last case, it need not be the set $\mathcal{T}_s$ constructed by the methods of Chapter 5; we just want any good covering of B_s. Of course, we must make a restriction on the input for this last question, as one cannot actually take arbitrary real numbers as input; perhaps it would be appropriate to restrict to s of the form

$$s \in \{p - q \log_3 n : p, q \in \mathbb{Q}, n \in \mathbb{N}\}.$$

We summarize:

Question 6.7.1. *What is the complexity of computing $\|n\|$? Of $\|n\|_{st}$? Of the difference $\Delta(n)$? Of the defect ordering $\delta(n_1) \leq \delta(n_2)$? Of the stable defect ordering $\delta_{st}(n_1) \leq \delta_{st}(n_2)$? Of the stabilization length $K(n)$?*

Question 6.7.2. *Given $s = p - q \log_3 n$, with $p, q \in \mathbb{Q}$ and $n \in \mathbb{N}$, what is the complexity of computing a good covering $\mathcal{T}_s$ of B_s?*

Meanwhile, for addition chains, it still remains to be shown that $K_\ell(n)$ and $\ell_{st}(n)$ are computable.

6.8 Remaining computational problems

As has been mentioned above, there is still the problem of showing that $K_\ell(n)$ and $\ell_{st}(n)$ are effectively computable. But there are also remaining computational problems regarding integer complexity.

For instance, given a real number $r \geq 0$, can we compute the order type of $\mathscr{D} \cap [0, r)$? There is an "obvious" algorithm to do this – take a good covering $\mathcal{T}$ of $\overline{B}_r$; sort the pairs (f, C) in order of increasing $\delta(f, C)$; to each distinct value α of $\delta(f, C)$ assign the ordinal ω^k, where k is the largest degree of some (f, C) with $\delta(f, C) = \alpha$; then add up these ordinals, in order. But proving that this algorithm actually works is not so easy, because low-defect pairs do not, in general, efficiently 3-represent all the numbers that they 3-represent. So it remains to be determined whether, in fact, this algorithm might work regardless.

Similar to the above problem is that of whether the order isomorphism between $\mathscr{D}$ and ω^ω is computable. This is largely the same, though; if one has an algorithm to determine the order type of $\mathscr{D} \cap [0, r)$, it can be adapted to answer this question. And, of course, the same question can be asked for $\mathscr{D}^\ell$ and ω^ω. But this seems substantially further away from being solved.

With this, we conclude. Some of the problems above seem to be well out of reach at present. We can hope that the method of defects and low-defect polynomials will be able to resolve some of the easier ones.

Appendix A

Conjectures of J. Arias de Reyna

This appendix deals with the results of Chapter 3. In his paper "Complejidad de los números naturales," [8] Juan Arias de Reyna proposed a series of conjectures about integer complexity. These conjectures also proposed a structure to integer complexity described by ordinal numbers, using a different language. These conjectures make assertions similar in spirit to some of the results in Chapter 3. Below we prove modified versions of his conjectures 5 through 7.

The conjectures deal with the quantity $n3^{-\lfloor \|n\|/3 \rfloor}$, which is related to (in fact, determined by) the quantity $\delta(n)$. We recall first the formula for the largest number writable with k ones which was proved by Selfridge (see [29]).

Definition A.0.1. Let $E(k)$ denote the largest number writable with k ones, i.e., the largest number with complexity at most k.

Theorem A.0.2 (Selfridge)**.** *The number $E(k)$ is given by the following formulae:*

$$\begin{aligned} E(1) &= 1 \\ E(3j) &= 3^j \\ E(3j+2) &= 2\cdot 3^j \\ E(3j+4) &= 4\cdot 3^j \end{aligned}$$

Based on this, in Chapter 2, we showed:

Proposition A.0.3. *We have $\delta(1) = 1$ and*

$$\delta(n) = \begin{cases} 3\log_3 \frac{E(\|n\|)}{n} & \text{if} \quad \|n\| \equiv 0 \pmod 3, \\ 3\log_3 \frac{E(\|n\|)}{n} + 2\,\delta(2) & \text{if} \quad \|n\| \equiv 1 \pmod 3, \text{ with } n > 1, \\ 3\log_3 \frac{E(\|n\|)}{n} + \delta(2) & \text{if} \quad \|n\| \equiv 2 \pmod 3. \end{cases}$$

That is to say, for $n > 1$, given the congruence class of $\|n\|$ modulo 3, the quantity $nE(\|n\|)^{-1}$ is a one-to-one and order-reversing function of $\delta(n)$.

As noted above, whereas in Chapter 2 we considered $nE(\|n\|)^{-1}$, Arias de Reyna considered $n3^{-\lfloor \|n\|/3 \rfloor}$. However, this is much the same thing:

Proposition A.0.4. *For $k > 1$,*

$$E(k) = c3^{\lfloor \frac{k}{3} \rfloor}$$

where

$$c = \begin{cases} 1 & \text{if} \quad k \equiv 0 \pmod 3, \\ 4/3 & \text{if} \quad k \equiv 1 \pmod 3, \\ 2 & \text{if} \quad k \equiv 2 \pmod 3. \end{cases}$$

So for $n > 1$, within each congruence class of $\|n\|$ modulo 3, the quantity $n3^{-\lfloor \|n\|/3 \rfloor}$ is also a one-to-one and order-reversing function of $\delta(n)$, being the same as $nE(\|n\|)^{-1}$ up to a constant factor.

This allows us to conclude the following result, which is a modified version of what one gets if one combines Arias de Reyna's Conjectures 5, 6, and 7 with his Conjectures 3 and 4.

Theorem A.0.5. (Modified Arias de Reyna Conjectures 5, 6, 7)

For $a = 0, 1, 2$, the sets

$$\left\{ \frac{n}{3^{\lfloor \|n\|/3 \rfloor}} : \|n\| \equiv a \pmod 3, \ n \textit{ stable} \right\}$$

are reverse well-ordered, with reverse order type ω^ω.

Equivalently, for $a = 0, 1, 2$, so are the sets

$$\left\{ \frac{n}{E(\|n\|)} : \|n\| \equiv a \pmod 3, \ n \textit{ stable} \right\}.$$

Proof. By Propositions A.0.3 and A.0.4, each of these is the image of some $\mathscr{D}_{st}^a$ under an order-reversing function. □

Appendix B

Good coverings of closed intervals

The theorems in Chapters 2, 3, and 5 about A_r and B_r, and how to build up coverings for them, etc., are formulated in terms of A_r and B_r, which are defined by the strict inequality $\delta(n) < r$. In many contexts, however, it is more natural to consider the nonstrict inequality $\delta(n) \leq r$. So let us define:

Definition B.0.6. For a real number $r \geq 0$, the set $\overline{A}_r$ is the set $\{n \in \mathbb{N} : \delta(n) \leq r\}$. The set $\overline{B}_r$ is the set of all elements of $\overline{A}_r$ which are leaders.

We can then also define:

Definition B.0.7. A finite set $\mathcal{S}$ of low-defect pairs will be called a *covering set* for $\overline{B}_r$ if, for every $n \in \overline{B}_r$, there is some low-defect pair in $\mathcal{S}$ that efficiently 3-represents it. We will say $\mathcal{S}$ is a *good covering* of $\overline{B}_r$ if, in addition, every $(f, C) \in \mathcal{S}$ satisfies $\delta(f, C) \leq r$.

One can then write down theorems about $\overline{A}_r$ and $\overline{B}_r$ similar to those in Chapters 2, 3, and 5 about A_r and B_r. We will state them here without proof, as the proofs are the same except for the strictnesses of some of the inequalities.

Theorem B.0.8. *For any real $0 \leq \alpha < 1$, $\overline{B}_\alpha$ is a finite set.*

Lemma B.0.9. *Let $x_1, x_2, \ldots, x_r > 0$ be real numbers such that $\sum_{i=1}^{r} x_i \leq k + 1$, where $k \geq 1$ is a natural number.*

1. *If $k \geq 2$ then either there is some i with $x_i > k$, or else we may find a partition $A \cup B$ of the set $\{1, 2, \ldots, r\}$ such that*

$$\sum_{i \in A} x_i \leq k, \qquad \sum_{i \in B} x_i \leq k.$$

2. *If $k = 1$ then either there is some i with $x_i > 1$, or else we may find a partition $A \cup B \cup C$ of the set $\{1, 2, \ldots, r\}$ such that*

$$\sum_{i \in A} x_i \leq 1, \qquad \sum_{i \in B} x_i \leq 1, \qquad \sum_{i \in C} x_i \leq 1.$$

Theorem B.0.10. *Suppose that $0 < \alpha < 1$ and that $k \geq 1$. Then any $n \in \overline{B}_{(k+1)\alpha}$ can be most-efficiently represented in (at least) one of the following forms:*

1. *For $k = 1$, there is either a good factorization $n = u \cdot v$ where $u, v \in \overline{B}_\alpha$, or a good factorization $n = u \cdot v \cdot w$ with $u, v, w \in \overline{B}_\alpha$;*
   For $k \geq 2$, there is a good factorization $n = u \cdot v$ where $u \in \overline{B}_{i\alpha}$, $v \in \overline{B}_{j\alpha}$ with $i + j = k + 2$ and $2 \leq i, j \leq k$.

2. *$n = a + b$ with $\|n\| = \|a\| + \|b\|$, $a \in \overline{A}_{k\alpha}$, $b \leq a$ a solid number and*

$$\delta(a) + \|b\| \leq (k+1)\alpha + 3\log_3 2.$$

3. *There is a good factorization $n = (a + b)v$ with $v \in \overline{B}_\alpha$, $a + b$ being a most-efficient representation, and a and b satisfying the conditions in the case (2) above.*

4. *$n \in T_\alpha$ (and thus in particular either $n = 1$ or $\|n\| = \|n - 1\| + 1$.)*

5. *There is a good factorization $n = u \cdot v$ with $u \in T_\alpha$ and $v \in \overline{B}_\alpha$.*

(Note here that we do not need to change the definition of T_α.)

Theorem B.0.11. *Suppose that $0 < \alpha < 1$ and that $k \geq 1$. Further suppose that $\mathcal{S}_{1,\alpha}, \mathcal{S}_{2,\alpha}, \ldots, \mathcal{S}_{k,\alpha}$ are covering sets for B_α, $B_{2\alpha}, \ldots, \mathcal{S}_{k\alpha}$, respectively. Then we can build a covering set $\mathcal{S}_{k+1,\alpha}$ for $B_{(k+1)\alpha}$ as follows:*

1. *If $k + 1 > 2$, then for $(f, C) \in \mathcal{S}_{i,\alpha}$ and $(g, D) \in \mathcal{S}_{j,\alpha}$ with $2 \leq i, j \leq k$ and $i + j = k + 2$ we include $(f \otimes g, C + D)$ in $\mathcal{S}_{k+1,\alpha}$;*
   while if $k + 1 = 2$, then for $(f_1, C_1), (f_2, C_2), (f_3, C_3) \in \mathcal{S}_{1,\alpha}$, we include $(f_1 \otimes f_2, C_1 + C_2)$ and $(f_1 \otimes f_2 \otimes f_3, C_1 + C_2 + C_3)$ in $\mathcal{S}_{2,\alpha}$.

2. *For $(f, C) \in \mathcal{S}_{k,\alpha}$ and any solid number b with $\|b\| \leq (k+1)\alpha + 3\log_3 2$, we include $(f \otimes x_1 + b, C + \|b\|)$ in $\mathcal{S}_{k+1,\alpha}$.*

3. *For $(f, C) \in \mathcal{S}_{k,\alpha}$, any solid number b with $\|b\| \leq (k+1)\alpha + 3\log_3 2$, and any $v \in B_\alpha$, we include $(v(f \otimes x_1 + b), C + \|b\| + \|v\|)$ in $\mathcal{S}_{k+1,\alpha}$.*

4. *For all $n \in T_\alpha$, we include $(n, \|n\|)$ in $\mathcal{S}_{k+1,\alpha}$.*

5. *For all $n \in T_\alpha$ and $v \in B_\alpha$, we include $(vn, \|vn\|)$ in $\mathcal{S}_{k+1,\alpha}$.*

Theorem B.0.12. *For any real $r \geq 0$, there exists a finite covering set $\mathcal{S}_r$ for $\overline{B}_r$. Furthermore, we can choose $\mathcal{S}_r$ such that each $(f, C) \in \mathcal{S}_r$ has degree at most $\lfloor r \rfloor$.*

Theorem B.0.13. *Let (f, C) be a low-defect pair, say of degree r, let $s \geq 0$ be a real number, and let $S = \{(k_1, \ldots, k_r) : \delta_{f,C}(k_1, \ldots, k_r) \leq s\}$. Then there exists a finite set $T \subseteq (\mathbb{Z}_{\geq 0} \cup \{*\})^r$ such that:*

1. *We have $S = \bigcup_{p \in T} S(p)$.*

2. *For each p in T, the set of i for which $k_i \neq *$ corresponds to a subset of the variables of f which is downward closed (under the nesting ordering); hence if (g, D) denotes the 3-substitution of p into (f, C), then (g, D) is a truncation of (f, C). Furthermore, we have $\delta(g, D) \leq s$, and hence $\deg g \leq \lfloor s \rfloor$.*

Theorem B.0.14. *For any real $s \geq 0$, there exists a finite good covering $\mathcal{S}_s$ of $\overline{B}_s$.*

Appendix C

Implementation notes

In this appendix we make some notes about the author's implementation of the algorithms of Chapter 5 and other ways they could be implemented.

We have actually not implemented Algorithm 8 and Algorithm 9 in full generality, where L may be arbitrary; we have only implemented the case where $L = \infty$, the case where $L = \|n\|$ (computed beforehand), and the case of Algorithm 10.

As was mentioned in Section 5.1.3, the step size in the author's implementation has been fixed at $\alpha = \delta(2)$, with the sets B_α and T_α precomputed. Other integral multiples of $\delta(2)$ were tried, up to $9\delta(2)$ (since $10\delta(2) > 1$ and thus is not a valid step size), but these all seemed to be slower, contrary to the author's expectation. Another variation with a similar flavor is that one could write a version of this algorithm with nonstrict inequalities, computing numbers n with $\delta(n) \leq r$ for a given r, rather than $\delta(n) < r$, as discussed in Appendix B. This was not tried.

It is also worth noting that the check for whether a given polynomial f 3-represents a given number n can also be sped up. If f is a low-defect polynomial with leading coefficent a, maximum coefficient A, and N terms, then

$$a3^{k_1+\ldots+k_r} \leq f(3^{k_1}, \ldots, 3^{k_r}) \leq A3^{k_1+\ldots+k_r},$$

so we only need to search $(k_1, \ldots, k_r)$ with

$$\lceil \log_3 \frac{n}{NA} \rceil \leq k_1 + \ldots + k_r \leq \lfloor \log_3 \frac{n}{a} \rfloor,$$

a stricter condition than was described in the algorithms above. This improvement is, in fact, used in the author's implementation. It is also possible that there is a better way than brute force.

As was mentioned in Section 5.7, when running Algorithm 8 or Algorithm 9 with L finite, one can omit the 3-representation check at intermediate steps. We have only

implemented this variant for Algorithm 10.

It was mentioned in Section 5.4.3 that considering "low-defect expression pairs" (E, C) or "low-defect tree pairs" (T, C) (where E is a low-defect expression, T is a low-defect tree, and $C \geq \|E\|$ or $C \geq \|T\|$, as appropriate) may be useful. In fact, the author's implementation works with a tree representation essentially the same as low-defect trees and low-defect tree pairs. Among other things, this makes it easy to find the minimal variables to be substituted into. If one were actually representing low-defect polynomials as polynomials, this would take some work. There is a slight difference in that, rather than simply storing a base complexity $C \geq \|T\|$, it stores for each vertex or edge – say with label $\|n\|$ – a number k such that $k \geq \|n\|$, unless we are talking about a non-leaf vertex and $n = 1$, in which case $k = 0$. We can then determine a C by adding up the values of k (as per Proposition 5.4.23). That is to say, the complexity, rather than being attributed to the whole tree, is distributed among the parts of the tree responsible for it; this makes it easier to check for and remove redundant low-defect pairs.

It was also mentioned in Section 5.4.3 that one could use a representation similar to low-defect expressions, but with all the integer constants replaced with $+, \cdot, 1$-expressions for same. E.g., instead of $2(2x+1)$, one might have $(1+1)((1+1)x+1)$. We have not implemented this, but doing this woud have one concrete benefit: It would allow the algorithms above to not only determine the complexity of a given number n, but also to give a shortest representation. (And analogously with stable complexity.) The current implementation cannot consistently do this in a useful manner. For instance, suppose that we ran Algorithm 10 and found some k with $\|2^k\| = 2k - 1$. We might then look at the actual low-defect pair (f, C) that 3-represented it, to learn what this representation with only $2k - 1$ ones is. But it might turn out, on inspection, that f was simply the constant 2^k; this would not be very enlightening. Using $+, \cdot, 1$-expressions would remedy this, as would having low-defect pairs keep track of their "history" somehow.

As was mentioned in Section 5.5.2, it's possible to write numerical versions of Proposition 5.5.6, that say exactly how far out one has to go in order to get within a specified ε of the limit $\delta(f, C)$; one could use this in Algorithm 4 instead of simply searching larger and larger K until one works. This was tried but found to be slower.

Finally, it is worth expanding here on the remark in Section 5.1.3 that it is possible to write Algorithm 8 and Algorithm 8 without using truncation. Surprisingly little modification is required; the only extra step needed is that, in order to check if n (or any $3^k n$) has defect less than $i\alpha$, instead of just checking if a low-defect pair (f, C)

(or its augmented version) 3-represents n (or any $3^k n$), if one finds that indeed $n = f(3^{k_1}, \ldots, 3^{k_r})$ (or the appropriate equivalent), one must additionally check whether $\delta_{f,C}(k_1, \ldots, k_r) < i\alpha$, since this is no longer guaranteed in advance. We will not state a proof of correctness here; it is similar to the proofs above. Such no-truncation versions of the algorithms were tried, but found to be too slow to be practical, because of the time needed to check whether the resulting polynomials 3-represented a given number. Another possibility, in the case where one is using a cutoff, is to truncate only at the final step, and not at the intermediate steps; this has not been tried. If this is used, it should probably be combined with not checking whether n (or any $3^k n$) is 3-represented until the final step, for the reason just stated.

Appendix D

Leaders with defect at most 1

What follows is a table of all leaders of defect at most 1, sorted by defect.

Table D.1: Leaders of defect at most 1

Index	Leader	Decomposition	Complexity	Defect
0	3	$2+1$	3	0
1	2	$1+1$	2	$\delta(2) \approx 0.1072$
2	4	$2^2 = 3+1$	4	$2\delta(2) \approx 0.2144$
3	8	2^3	6	$3\delta(2) \approx 0.3216$
4	16	2^4	8	$4\delta(2) \approx 0.4288$
5	32	2^5	10	$5\delta(2) \approx 0.5361$
6	5	$4+1$	5	$\delta(5) \approx 0.6051$
7	64	2^6	12	$6\delta(2) \approx 0.6433$
8	7	$2 \cdot 3 + 1$	6	$\delta(7) \approx 0.6863$
9	10	$2 \cdot 5 = 3^2 + 1$	7	$\delta(5) + \delta(2) \approx 0.7123$
10	128	2^7	14	$7\delta(2) \approx 0.7505$
11	14	$2 \cdot 7$	8	$\delta(7) + \delta(2) \approx 0.7935$
12	20	$2 \cdot 10$	9	$\delta(5) + 2\delta(2) \approx 0.8195$
13	256	2^8	16	$8\delta(2) \approx 0.8577$
14	28	$2^2 \cdot 7 = 3^3 + 1$	10	$\delta(7) + 2\delta(2) \approx 0.9007$
15	40	$2 \cdot 20$	11	$\delta(5) + 3\delta(2) \approx 0.9267$
16	19	$2 \cdot 3^2 + 1$	9	$\delta(19) \approx 0.9596$
17	512	2^9	18	$9\delta(2) \approx 0.9649$
18	82	$3^4 + 1$	13	$\delta(82) \approx 0.9665$
19	244	$3^5 + 1$	16	$\delta(244) \approx 0.9888$
20	13	$4 \cdot 3 + 1$	8	$\delta(13) \approx 0.9958$
$21+k$	$3^{k+6}+1$	$3^{k+6}+1$	$3(k+6)+1$	$1 - 3\log_3(1+3^{-(k+6)})$
ω	1	1	1	1

References

[1] H. Altman, Addition Chains and Well-Ordering, in preparation.

[2] H. Altman, Integer Complexity and Well-Ordering, `arXiv:1310.2894`, 2013

[3] H. Altman, Integer Complexity: Computational Methods and Results, in preparation.

[4] H. Altman, Integer Complexity: The Integer Defect, in preparation.

[5] H. Altman, Refined Estimates for Counting Numbers of Low Defect, in preparation.

[6] H. Altman and J. Arias de Reyna, Integer Complexity, Stability, and Self-Similarity, in preparation

[7] H. Altman and J. Zelinsky, Numbers with Integer Complexity Close to the Lower Bound, *Integers* **12** (2012), no. 6, 1093–1125.

[8] J. Arias de Reyna, Complejidad de los números naturales, *Gaceta R. Soc. Mat. Esp.*, **3** (2000), 230–250.

[9] J. Arias de Reyna and J. Van de Lune, Algorithms for determining integer complexity, `arXiv:1404.2183`, 2014

[10] J. Arias de Reyna and J. Van de Lune, "How many 1's are needed?" revisted, `arXiv:1404.1850`, 2014

[11] H. M. Bahig, On a generalization of addition chains: Addition-multiplication chains, *Discrete Mathematics* **308** (2008), 611–616.

[12] H. M. Bahig and H. M. Bahig, A new strategy for generating shortest addition sequences, *Computing* 91 (2011), 285–306.

[13] L. Blum, F. Cucker, M. Shub, S. Smale, Algebraic Settings for the Problem "$P \neq NP$?",

[14] P. Borwein, J. Hobart, The Extraordinary Power of Division in Straight Line Programs, *American Mathematical Monthly* **119** (2012), 584–592.

[15] A. Brauer, On Addition Chains, *Bull. Amer. Math. Soc.*, **45** (1939), 736–739.

[16] P. Bürgisser, M. Clausen, and M. A. Shokrollahi, *Algebraic Complexity Theory*, Springer-Verlag, Berlin, 1997

[17] P. W. Carruth, Arithmetic of ordinals with applications to the theory of ordered abelian groups, *Bull. Amer. Math. Soc.* **48** (1942), 262–271.

[18] N. M. Clift, Calculating optimal addition chains, *Computing* **91** (2011), 265–284.

[19] J. H. Conway, *On Numbers and Games*, Second Edition, A K Peters, Ltd., Natick, Massachusetts, 2001, pp. 3–14.

[20] A. Cottrell, A lower bound for the Scholz-Brauer problem, PhD. Dissertation, University of California, Berkeley (1974)

[21] D. H. J. De Jongh and R. Parikh, Well-partial orderings and hierarchies, *Indag. Math.* **39** (1977), 195-206.

[22] H. Dellac, *Interméd. Math.* **1** (1894), 162–164.

[23] W. De Melo and B. F. Svaiter, The Cost of Computing Integers, *Proc. Amer. Math. Soc.* **124** (1996), 1377–1378.

[24] P. Downey, B. Leong, and R. Sethi, Computing Sequences with Addition Chains, *SIAM J. Comput.* **10** (1981), 638–646.

[25] S. Fomin, D. Grigoriev and G. Koshevoy, Subtraction-free complexity, cluster transformations and spanning trees, arXiv:1307.8425.

[26] M. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-completeness*, Freeman: San Francisco 1979

[27] A. A. Gioia, M. V. Subbarao, and M. Sugunamma, The Scholz-Brauer Problem in Addition Chains, *Duke Math. J.*, **29** (1962), 481–487.

[28] D. Grigoriev, Lower bounds in algebraic complexity, *J. Soviet Math.* **29** (1985), 1388–1425.

[29] R. K. Guy, Some suspiciously simple sequences, *Amer. Math. Monthly*, **93** (1986), 186–190; and see **94** (1987), 965 & **96** (1989), 905.

[30] R. K. Guy, *Unsolved Problems in Number Theory*, Third Edition, Springer-Verlag, New York, 2004, pp. 399–400.

[31] W. Hansen, Zum Scholz-Brauerschen Problem, *J. Reine Angew. Math.* **202** (1959), 129–136.

[32] J. Iraids, personal communication.

[33] J. Iraids, K. Balodis, J. Čerņenoks, M. Opmanis, R. Opmanis, K. Podnieks. Integer Complexity: Experimental and Analytical results, arXiv:1203.6462, 2012

[34] M. Jerrum and M. Snir, Some Exact Complexity Results for Straight-Line Computations over Semirings, *J. ACM* **29** (1982), 874–897.

[35] D. E. Knuth, *The Art of Computer Programming*, Vol. 2, Third Edition, Addison-Wesley, Reading, Massachusetts. [Section 4.6.3 is pp. 461–485.]

[36] M. Kutz, Lower Bounds for Lucas Chains, *SIAM J. Comput* **31** (2002), 1896–1908.

[37] J. C. Lagarias, On ternary expansions of powers of 2, *J. London Math. Soc.*, **79** (2009), 562–588. *MR* 2508867.

[38] K. Mahler & J. Popken, On a maximum problem in arithmetic (Dutch), *Nieuw Arch. Wiskunde*, (3) **1** (1953), 1–15; *MR* **14**, 852e.

[39] D. A. Rawsthorne, How many 1's are needed?, *Fibonacci Quart.*, **27** (1989), 14–17; *MR* **90b**:11008.

[40] C. P. Schnorr, A lower bound on the number of additions in monotone computations, *Theor. Comput. Sci.* **2**, (1976), 305–315.

[41] A. Scholz, Aufgabe 253, Jahresbericht der Deutschen Mathematikervereinigung, Vol. 47, Teil II, B. G. Teubner, Leipzig and Berlin, 1937, pp. 41–42.

[42] A. Schönhage, A Lower Bound for the Length of Addition Chains, *Theoretical Computer Science*, **1** (1975), 1–12.

[43] Z. Semadeni, *Banach Spaces of Continuous functions*, Vol. I, Monografie Matematyczne, Tom 55. PWN—Polish Scientific Publishers, Warsaw, 1971.

[44] V. V. Srinivas & B. R. Shankar, Integer Complexity: Breaking the $\Theta(n^2)$ barrier, *World Academy of Science*, **41** (2008), 690–691.

[45] C. L. Stewart, On the Representation of an Integer in Two Different Bases, *J. Reine Angew. Math.*, **319** (1980), 63–72.

[46] M. V. Subbarao, Addition Chains – Some Results and Problems, *Number Theory and Applications*, Editor R. A. Mollin, NATO Advanced Science Series: Series C, V. 265, Kluwer Academic Publisher Group, 1989, pp. 555–574.

[47] K. B. Stolarsky, A Lower Bound for the Scholz-Brauer Problem, *Canadian Journal of Mathematics*, **21** (1969), 675–683.

[48] E. G. Thurber, Efficient Generation of Minimal Length Addition Chains, *SIAM J. Comput.*, **28** (1999), 1247–1263.

[49] E. G. Thurber, On Addition Chains $l(mn) \leq l(n) - b$ and Lower Bounds for $c(r)$, *Duke Math. J.*, **40** (1973), 907–913.

[50] E. G. Thurber, The Scholz-Brauer Problem on Addition Chains, *Pacific Journal of Mathematics*, **49** (1973), 229–242.

[51] L. G. Valiant, Negation can be exponentially powerful, *Theor. Comput. Sci.* **12**, (1980), 303–314.

[52] J. Zelinsky, An Upper Bound on Integer Complexity, in preparation