Una perspectiva computacional sobre números normales

Heiber, Pablo Ariel

2014

UNIVERSIDAD DE BUENOS AIRES
FACULTAD DE CIENCIAS EXACTAS Y NATURALES
DEPARTAMENTO DE COMPUTACIÓN

Una perspectiva computacional sobre números normales

Tesis presentada para optar al título de Doctor de la Universidad de Buenos Aires
en el área Ciencias de la Computación

Pablo Ariel Heiber

Directora de tesis:	Dra. Verónica Becher
Consejera de estudios:	Dra. Verónica Becher

Buenos Aires, 2014

Una perspectiva computacional sobre números normales

Resumen: La normalidad es una forma débil de azar. Un número real es normal en una base entera dada si su expansión en esa base es balanceada: todos los bloques de la misma cantidad de dígitos tienen igual frecuencia en la expansión. La normalidad absoluta es normalidad en toda base. En esta tesis resolvemos varios problemas sobre normalidad:

- La existencia de números absolutamente normales computables era conocida, pero no se conocía ningún algoritmo que computara uno en tiempo polinomial. Nosotros damos un algoritmo que computa uno en tiempo apenas mayor a cuadrático.
- Mostramos que el conjunto de números absolutamente normales, como subconjunto de los reales, no tiene otras propiedades aritméticas que las impuestas por la definición de normalidad. Técnicamente, demostramos que el conjunto de números absolutamente normales es Π^0_3-completo.
- Extendemos la caracterización conocida de normalidad en términos de incompresibilidad mediante autómatas finitos. Analizamos exhaustivamente todas las maneras de mejorar un simple autómata finito agregando memoria de diferentes formas, permitiendo no-determinismo y permitiendo la lectura de la entrada más de una vez.
- Demostramos que la normalidad se preserva bajo reglas de selección basadas en prefijos finitos o sufijos infinitos reconocidos por autómatas finitos, pero no ambos simultáneamente. Esto extiende un resultado conocido para el caso de prefijos.

palabras clave: números normales, complejidad algorítmica, complejidad descriptiva, autómatas finitos, compresibilidad

A computational perspective on normal numbers

Abstract: Normality is a weak form of randomness. A real number is normal to a given integer base if its expansion in that base is balanced: all blocks of the same number of digits occur with the same frequency in the expansion. Absolute normality is normality to all bases. We solve several problems on normality:

- It was known that computable absolutely normal numbers exist, but no algorithm was known to compute one in polynomial time. We give an algorithm that computes one in just above quadratic time.
- We show that the set of absolutely normal numbers, as a subset of the real numbers, has no other arithmetical properties than those imposed by the definition of normality. Technically, we prove that the set of absolutely normal numbers is Π^0_3-complete.
- We extend the known characterization of normality in terms of incompressibility by deterministic finite automata. We exhaust all ways of enhancing a simple finite state automaton by adding memory in different forms, allowing non-determinism, and allowing to read the input more than once.
- We prove that normality is preserved by selection rules based on finite prefixes or infinite suffixes being recognized by finite automata, but not both simultaneously. This extends a known result about the prefixes case.

keywords: normal numbers, algorithmic complexity, descriptive complexity, finite automata, compressibility

Agradecimientos

En esta sección quiero agradecer a quienes contribuyeron con el desarrollo de la tesis, pero también a los que contribuyeron con otras partes del doctorado, o con mi formación y motivación de entrar en este mundo en general.

El orden no es cronológico ni por importancia. Los agradecimientos están agrupados de forma más o menos temática, lo que provoca que algunas pocas personas aparezcan más de una vez. Es mi intención que cada persona reciba su agradecimiento y no lo compare con los agradecimientos de los demás.

A cada persona o grupo intento agradecerle en su idioma principal.

En primer lugar, no me alcanzan las palabras para agradecerle a Verónica Becher. Por dirigir la tesis, por supuesto, pero eso es sólo una pequeña parte. Le agradezco compartir su brillantez, expresada de una forma distinta a la que yo estoy o estaba acostumbrado. Por escribir las introducciones que yo nunca quise escribir, hasta el punto en que, sorpresivamente, dejó de molestarme escribirlas personalmente. Por siempre valorar mi opinión respecto de la escritura, los temas de investigación, la conveniencia de un viaje o de un lugar de publicación. Por ayudarme consiguiendo fondos para viajar y tener oportunidad de estar en congresos y pasantías, e incluso buscar un departamento para quedarnos ambos en París. Por ayudarme a hacer contactos de primer nivel académico y humano. Por entender y aceptar mis temas y tiempos individuales. Por venir a investigar a casa e invitarme a la suya para evitarme el viaje hasta Ciudad Universitaria. Por aconsejarme en temas incluso ajenos a la tesis o la investigación. Por charlar de docencia, de política universitaria, de política en general, de chismes, de cosas personales. Por generar un ambiente de colaboración y honestidad en el que da gusto trabajar. Hay ciertas cosas que un director o directora siempre hace, otras tantas que deberían hacer y unas cuantas más que yo creo que sería ideal que hagan. Verónica hizo mucho más que eso. Gracias.

Secondly, I would like to thank two people whose contributions and advice in research deeply influenced my work, to the point of acting almost as co-advisors for some parts of it.

A big thank you to Theodore Slaman. His ideas and guidance were key in the development of some big results. And also, for managing the difficult task of giving advice on research career without being condescending and always speaking like an equal. For always taking criticism (which I sometimes overdo) in a good way, despite the difference in experience and accumulated prestige. And for sharing his insights on how the academic world works. At all times working with him I felt intellectually appreciated and personally taken care of.

J'adresse aussi un vif remerciement à Olivier Carton, qui, après ma directrice, c'est le chercheur avec qui j'ai plus travaillé. Grâce à lui, j'ai appris beaucoup d'automates, un thème qui ne se dégage pas immédiatement du titre ou du plan de thèse. Aujourd'hui je me sens capable de travailler ce sujet, et je ne peux que déduire que cela a été par l'action de-et-avec Olivier. Je lui remercie d'avoir profondément partagé avec moi ses bonnes idées, de telle manière que je puisse désormais en avoir d'autres; de finir mes études à Paris -ce qui a été un plaisir-; des déjeuners et de la invitation aux pique-niques, aussi. Lors du travail avec lui, j'ai appris à travailler en équipe longtemps malgré la distance physique. C'est dommage, car j'ai beaucoup joui de travailler auprès du même Olivier.

I would like to thank the three people that took the time to read and judge this thesis.

A Antonio Montalbán, que aceptó que aprovecháramos su presencia en Buenos Aires y usó tiempo de su viaje para leer la tesis y asistir a la defensa.

à Jean-Eric Pin, qui a fourni son expérience, qui est venu de France pour assister à la soutenance de la thèse et l'a soigneusement lue en simplifiant notre travail, tout à temps.

A Diego Vaggione, por aceptar completar el jurado leyendo una tesis alejada de su tema de investigación.

También quiero agradecer a Delia Kesner y a Pablo Jacovkis por aceptar actuar como suplentes del jurado de la tesis.

Additional thanks to Joos Heintz, Elvira Mayordomo, Satyadev Nandakumar, André Nies, Santiago Figueira and Laurent Bienvenu, for taking the time, at different moments in the past years, to listen or read about my work and commenting, warning, providing references, and pointing out existing results I was unaware of. Also to many anonymous referees that assessed my papers.

Gracias también a Melisa, Pablo B. y Alejandro, por compartir el inicio de un grupo de investigación conmigo y con Verónica. Aunque los resultados de esos momentos no hayan entrado en el trabajo compilado en esta tesis, sin duda fueron el puntapié inicial. A Pablo T., que ingresó al grupo y a la oficina mas adelante, cuando yo ya estaba trabajando en otros temas, por su simpatía como compañero.

A Delia Kesner, por su invaluable ayuda en cada una de mis visitas a París-Diderot con una de las partes más difíciles de la investigación: el papeleo y la plata.

A Facundo López Bristot, por su confianza para tenerme como director de tesis de licenciatura, a pesar de ser mi primera experiencia haciéndolo, y por la prolijidad para expresar y presentar los resultados. A Nicolás Rosner, que junto con Facundo ayudó a traducir un artículo sólo disponible en Alemán. El entendimiento de ese artículo fue el puntapié inicial para una parte importante de los resultados que aquí presentamos. A Martín Epsztein, por permitirme codirigir su tesis de licenciatura y continuar así con estas experiencias, además de proveer un derivado concreto de los resultados obtenidos en mi trabajo de investigación.

A Esteban Feuerstein por la confianza que me permitió dar los primeros pasos en investigación. A Jorge Lucángeli Obes y Matías López Rosenfeld por acompañarme en ese camino inicial. Y un agradecimiento especial a Marcelo Mydlarz, por compartir su experiencia y por sus consejos a la hora de escribir y preparar presentaciones, que aprovecho hasta el día de hoy.

Para terminar con los agradecimientos a las contribuciones directas sobre el trabajo de investigación, quisiera agradecerle mucho a Martín Mereb, Martín Szyld y Sergio Yuhjtman, por escuchar innumerables preguntas de diversas áreas de la matemática, deducir qué estaba queriendo decir, y proveer respuestas, ideas o referencias para destrabarme en ese camino.

Agradezco también a Eliana Romano y Daniela Sant Ana por su ayuda traduciendo parte de los agradecimientos.

Agradezco al CONICET por la beca de doctorado que me permitió realizar este trabajo de investigación. A la Facultad de Ciencias Exactas y Naturales y a la Universidad de Buenos Aires por mantener, dentro de toda su imperfección y las presiones externas en contrario, una educación gratuita y de altísima calidad, con acceso irrestricto, siendo todavía una excepción en el mundo actual de educación privada y elitista.

A la Agencia Nacional de Promoción Científica y Técnica, al CONICET, a la Universidad de Buenos Aires, al CNRS de Francia, a la Universidad París-Diderot, a Yahoo! Research, a IBM Argentina, a Biosidus y a Microsoft Research, por distintas becas, subsidios y financiamiento en general que permitieron que pueda realizar pasantías y asistir a congresos que aportaron a mi formación y al desarrollo de esta investigación.

Por último, quiero agradecer a TPSH por el servicio de repositorio SVN que facilitó enormemente el desarrollo de las investigaciones, especialmente de la escritura de artículos y de esta tesis.

Hace casi 12 años comencé a cursar la licenciatura en computación y hace 11 que estoy relacionado con el departamento de computación. No pasa tanto tiempo sin la sensación de que uno forma parte de una comunidad. Tanto tiempo de convivencia, por supuesto, también viene con roces, desacuerdos y peleas, pero igualmente siempre sentí al DC como un lugar agradable para estudiar y trabajar. No voy a nombrar gente en este apartado, porque son demasiados. Agradezco entonces a los miembros del DC, a sus estudiantes, docentes y el resto del personal, por formar, a pesar de desacuerdos académicos, políticos y a veces hasta de forma, una comunidad en la que se puede formarse y crecer.

A mis docentes, ya que siento que tuve una buena formación académica en general, y además, que pude conocer a través de ellos distintas formas de impacto de la disciplina, desde lo muy teórico a lo muy práctico.

También al departamento y a la facultad la posibilidad de hacer docencia desde hace 7 años. A mis varios compañeros de cátedra y a los estudiantes, sin los cuales esa posibilidad no hubiera tenido sentido.

Terminando con el segmento, quiero agradecer a los miembros de La Mella en Exactas, por mostrarme que es un desperdicio venir a la universidad pública solamente a estudiar.

I would like to acknowledge the impact the International Collegiate Programming Contest had in my development as a computer scientist. The skills acquired while participating, writing problems and organizing the competition were significant in succeeding in academia. Skills not restricted to problem solving and programming, but also in communicating effectively, being organized and working in teams.

A Francisco Roslan (Pancho), por estar conmigo en varios equipos participando y alentarme a estudiar y entrenar mucho. A Hernán Bandura, Matías del Hoyo y Alejandro Deymonnaz, por ser parte del equipo en distintos momentos. A Darío Fischbein por entrenarnos en todos los casos, y especialmente por iluminarme, quizás sin darse cuenta, en la importancia de las estructuras de datos.

I would like to thank the ICPC Headquarters for creating and organizing the competition, and specifically to Marsha Poucher, who diligently answered my many questions and requests while putting competitions together.

Un agradecimiento especial a Irene Loiseau, por todos los años de trabajo organizando la competencia en Argentina sin el cual seguramente yo y muchos otros no podríamos haberla aprovechado. A los que ayudaron en la organización mientras yo era participante, cuando la cantidad de gente para trabajar y el nivel organizativo general no eran los mejores. Especialmente a Pablo Coll y Alejandro Strejilevich de Loma que se pusieron mucho de ese trabajo al hombro.

A toda la organización de la regional latinoamericana de la ICPC, por mejorar la llegada y el desarrollo de la competencia a nivel regional, y por incorporarme efectivamente una vez que concluí mi participación. A Ricardo Anido por me dar lugar na equipe de desenvolvimento da prova e me ajudar a fazer meu trabalho nela.

A Rohit Gheyi e Rodolfo Azevedo pelo convite a dar aulas no Brasil e me receber em todos os momentos da melhor maneira possível. A Dovier Antonio Ripoll Méndez por la posibilidad de dar clases en Cuba haciéndome sentir cómodo en la tarea.

A Fidel Schaposnik, Alejandro Santos, Joaquin Rodrigues, Ramiro Lafuente, Eric Destefanis, Javier Godoy, Alejandro Strejilevich de Loma, Irene Loiseau, por sumarse y aportar a la existencia de una competencia Argentina por primera vez. A los muchos que se sumaron luego para hacerla mejorar y crecer.

Antes de entrar a la universidad, durante el colegio secundario, dos instituciones contribuyeron a formar mi perfil de matemático y computador. Agradezco a la orientación en informática de la escuela ORT y a la Olimpiada Matemática Argentina.

En particular, quisiera agradecer a uno de mis primeros profesores de informática, Guillermo Lutzky, por tenerme paciencia con mis adolescencias y apoyarme con la vocación informática que estaba incubando.

También a quienes me prepararon para las competencias de matemática. A Graciela Pasoviola, por la organización. A Liliana Kurzrok, a Matilde Lalín, a Martín Mereb y a Jhonatan Barmak, por el entrenamiento. Un agradecimiento especial en ese sentido va para Alejandro Marcu, quien me llevo sorpresivamente al Torneo de Computación y Matemática, mi primer competencia con contenidos de programación, no sin antes enseñarme la Criba de Eratóstenes, el primer algoritmo “de verdad” que aprendí. También a quienes en mis años me entrenaron para las competencias de matemática. A Graciela Pasoviola, por la organización. A Liliana Kurzrok, a Matilde Lalín, a Martín Mereb y a Jhonatan Barmak, por el entrenamiento. Un agradecimiento especial en ese sentido va para Alejandro Marcu, quien me llevo sorpresivamente al Torneo de Computación y Matemática, mi primer competencia con contenidos de programación, no sin antes enseñarme la Criba de Eratóstenes, el primer algoritmo “de verdad” que aprendí.

A mis muchos compañeros y amigos de las olimpiadas de matemática, y especialmente a Martín Szyld y a Diego Materyn por hacer de la formación y el entrenamiento un momento de disfrute.

Por último, quiero agracederle a Adrián Escandarani por la oportunidad de ejercer en la escuela ORT mi primer trabajo docente. También a Gonzalo Zabala por admitirme luego de egresado en la organización de la Olimpiada Metropolitana de Informática, un rol que sin duda me inspiró.

Para concluir con los agradecimientos, quiero reconocer a aquellos que en este largo camino me alentaron, me sostuvieron, se alegraron conmigo y me soportaron, todo en su debido momento. Esa gente que no necesito nombrar con apellidos. Es importante notar que es mucho más fácil hacer una lista de las ayudas más visibles y concretas que tienen que ver con la tesis, que de las otras. Que la parte siguiente sea menos extensa en palabras de ninguna manera indica que sea menos extensa en gratitud y en cariño que las anteriores.

Agradezco a mis amigos de TPSH por ponerle onda a las horas compartidas de estudio y de hacer trabajos prácticos. A mis otros amigos de la facultad, Esteban, Guido, Daniela, Leandro, y algunos otros que seguro me olvido.

También a mis amigos del alma, que no los tengo gracias a la carrera, pero que igual hicieron el aguante. Es difícil decir algo más, ya que no puedo agruparlos sin cometer injusticias o ponerlos individualmente, así que muchas gracias a todos, cada uno sabe que parte del agradecimiento le corresponde. Gracias a Martín, Diego, Sergio, Meli, Emi, Alberto, Dai, Joy, Dami, Rodrigo, Ivana, Martín, y a los otros que seguro me estoy olvidando.

A mi familia, que con su composición rara y despareja, con sus cosas buenas y malas, con peleas y desacuerdos, siempre me apoya y aporta cosas que ni siquiera puedo nombrar.

Agradezco a mis suegros Diana y Sergio por casi adoptarme y estar de mi lado aunque no los obliguen los lazos sanguíneos.

A mis hermanas Dana y Paloma, por ser tan diferentes y proveer una ventana a la parte del mundo que casi nunca miro.

A mis padres por alentarme a aprender mucho y a crecer a pesar de que la sociedad muchas veces no me entiende. Por enseñarme el valor del conocimiento y del razonamiento.

A Leila por aguantarse mas que nadie todos los efectos colaterales de esta tesis y de gran parte de mi carrera profesional. Por ponerse contenta con mis cosas aún cuando no las entienda. Por hacer el esfuerzo de entenderlas. Por darme horizontes nuevos que resignifican mi trabajo y le dan perspectiva. Por ayudarme a dormir, a llegar a la facultad, a concentrarme para escribir, a pasar los momentos tensos de cada viaje. Por amarme y dejarse amar.

Dedicado al movimiento estudiantil y docente. A los que hicieron la reforma de 1918, a los que apoyaron el Cordobazo, los que aguantaron noches infames de bastones largos y de lápices. A los que construyeron y defendieron una educación pública, laica, de calidad, gratuita y con acceso irrestricto, sin la cual esta tesis no habría existido.

Contents

We, the authors. The present thesis is written in first-person plural. Since different parts of the work were done by different sets of people, different instances of *we* may mean different sets. In all cases it means Pablo Ariel Heiber together with a subset of Verónica Becher, Olivier Carton and Theodore A. Slaman.

CHAPTER 1

Introduction

This thesis is about normality, in particular, normality as a notion of randomness. Normality is a combinatorial property of real numbers. In this chapter we introduce normality informally and motivate its study from a computational perspective. A formal presentation of normality is delayed until the next chapter.

Normality has now more than one hundred years and it is a subject of study in number theory, ergodic theory, and theoretical computer science, specifically in combinatorics of words. The thesis solves selected open problems on normal numbers in relation with computing machines. We start here with some historical references that illustrate our path through the subject, and the relevance of the problems we solved in the context of the previous results. The references we give should not be taken as a thoroughly complete historical account but as a trace of our work towards the problems we treat in this thesis.

1.1 Brief history and motivation

At the end of the 17th century and beginnings of the 18th, Jakob Bernoulli developed the foundations of probability theory, in an attempt to formally support the bases of experimental science. Much of this work was later published in [Ber13]. A pillar of the theory is the very first version of the law of large numbers. In natural language, the law of large numbers prescribes that if the same random experiment is repeated, the frequency of occurrence of each possible outcome converges to its real probability. Then, randomness was regarded as a property of the experiment, but not a property of the objects. This thesis, on the other hand, focuses on randomness as a property of the objects.

In 1959, Karl Popper published "The logic of scientific discovery" [Pop59]. In his book he develops, in his words, a method to construct a random-like object. He deals with a specific kind of object: infinite words of zeroes and ones. He argues that for such an object to be random-like, it ought to contain each possible short word in a minimalistic way. He goes on to conclude that this means that the object adheres to the law of large numbers. That is, the actual frequency of each word in the object converges to the probability of that word, given by a uniform distribution of probability for the all the words of each given length.

Popper's concept aimed at discussing epistemology was mathematically formalized long before. In 1909, Émile Borel defined normal numbers [Bor09]. To define normality, Borel considered conditions on the expansion of each real number. Thus, in some sense, he studied an object similar to Popper's: infinite words of digits. The condition Borel prescribed on the expansion of a real number for it to be normal is adherence to the law of large numbers. More precisely, a real number is normal to a given integer base if its expansion in that base is balanced: all blocks of the same number of digits occur with the same frequency in the expansion. He also defined absolutely normal numbers as those that are normal in every possible integer base. A fundamental theorem, proved by Borel as soon as he gave the definition of normality, is that almost all real numbers are absolutely normal. Thus, it is a regular situation to obtain a normal number when choosing a real number at random. This justifies the chosen name *normal* and it also explains why normality is a necessary condition for randomness.

Borel's theorem ensuring that almost all real numbers are absolutely normal implies, of course, their existence. Borel asked immediately for a concrete example. It has not been easy to exhibit specific instances, even for normality to one base. Borel conjectured that all irrational algebraic numbers are absolutely normal, but to this day, normality could not be proved or disproved even for a single example of an algebraic irrational on a single base. Usual mathematical constants like π, e or $\sqrt{2}$ are conjectured to be absolutely normal, and some empirical evidence seems to point in that direction [BC01, SNS13], but a proof is still missing.

The first example of a number normal to a given base is due to Champernowne in 1933 [Cha33]. It is the concatenation of all positive integers expressed in that base, in order. This construction was generalized by Copeland and Erdös in 1946 [CE46] who considered the concatenation of the terms of a sequence obtained by certain polynomials. In particular, the concatenation of all the primes expressed in some base yields a normal number to that base. Many other examples were developed later based on the concatenation idea. This way of proceeding only provides non-algebraic examples, thus leaving Borel's conjecture still open.

For absolutely normal numbers, the first constructions are independently due to Henri Lebesgue and Wacław Sierpiński [Leb17, Sie17] but they are not finitary based. Thus, these constructions do not stand as algorithms to compute instances, but they provide names to particular instances. The first algorithm to output absolutely normal numbers was given by Alan Turing, presumably in 1937 [Tur92]. But unfortunately his manuscript was not complete and remained unpublished until the edition of his Collected Works. In 1962, Schmidt gave a computable construction [Sch62]. Although he noticed that his construction is effective, there has been no analysis of its computational complexity, and it has not circulated in the computer science community.

The availability of algorithms to compute absolutely normal numbers became better known only recently. Becher, Figueira and Picchi reconstructed Turing's algorithm and made a full computational analysis of it [BFP07]. Becher and Figueira also gave a recursive reformulation of Sierpiński's construction [BF02]. Both algorithms turned out to have double exponential time complexity. These algorithms are too slow to be used in practice, in contrast to the optimal time complexity and simplicity of a construction as Champernowne's for normality to a given base. This opened the question of the existence of feasible algorithms to generate absolutely normal numbers. Such feasible algorithm would be one step closer to exhibiting concrete examples of absolutely normal numbers.

During the middle of the 20th century, a different approach on defining random objects took the scene. In 1948, Shannon developed a general theory for measuring information content [Sha48]. In 1965, Kolmogorov formalized a different approach for measuring information content of finite words based on computability [Kol65], but failed to use it to define a randomness notion for infinite words. In 1966, his student Martin-Löf succeeded in defining randomness for infinite words based on computability [ML66], which turns out to be equivalent to a definition based on Kolmogorov's ideas. A few years later Chaitin showed that Shannon's and Kolmogorov's approach to define information content were compatible [Cha75]. A similar theory was developed by Levin [ZL70, Lev73].

This convergence of approaches to a unique way of defining randomness created a new area called algorithmic information theory [Cha87, Nie09, DH10]. Absolute normality is a necessary but not sufficient condition for Martin-Löf's randomness. Since algorithmic information theory is focused on definitions based on definability, computability and predictability with Turing machines, a combinatorial definition like normality has not been the center of attention. However, many of the aspects that have been studied for randomness as part of algorithmic information theory can be studied for normality. We cover some of those aspects.

In 1968, Agafonov studied a relation between computational power and normality, by showing that a particular class of functions computed by finite automata preserve normality [Aga68]. Normality, a combinatorial property, was thus related to a computing machine. The full set of machine operations that preserve normality is still unknown. Indeed, only a few forms of preservation of normality have been studied. In contrast, the problem of preservation of randomness has been fully understood by algorithmic information theory.

These is another relation between normality and computing machines, which also involves finite automata: it is possible to define normality in terms of incompressibility. This result was obtained for the first time by joining a theorem by Schnorr and Stimm [SS72] with a theorem by Dai, Lathrop, Lutz and Mayordomo [DLLM04]. The result characterizes normality as incompressibility by finite automata, the simplest computing machines. This characterization of normality in terms of incompressibility by finite automata is parallel to the definition of randomness in terms of incompressibility by Turing machines in algorithmic information theory. This gave the study of normality as a notion of randomness a new push. The theorem opened the question of whether machines that are more powerful that finite automata but less than a Turing machine could compress normal sequences.

1.2 Contributions and open problems

The thesis makes three main original contributions. The first one is a solution to the question of existence of feasible algorithms to compute an absolutely normal number. We present one that computes the first n digits in time polynomial in n; in fact, just above quadratic. The algorithm uses combinatorial tools to control divergence from normality. Speed of computation is achieved at the sacrifice of speed of convergence to normality. This is currently the fastest algorithm that produces an absolutely normal number. The question of its overall optimality, on the other hand, remains open. Its efficiency allowed us to program a practical version of the algorithm and compute the first digits of the absolutely normal output [BEHS13]. The first few digits are $0.4031290542003809132371428380827059102765 11\cdots$.

The technique we use to give the algorithm allowed us to show a subsidiary result. We show that the set of absolutely normal numbers, as a subset of the real numbers, has no other arithmetical properties than those expressible with three alternations of quantifiers, starting with a for all. Technically, this is a result in descriptive set theory that says that the set of absolutely normal numbers is $\mathbf{\Pi}^0_3$-complete in the

Borel hierarchy of subsets of real numbers. Similarly, the set of absolutely normal numbers is Π^0_3-complete in the effective Borel hierarchy. These results are compiled in Chapter 3 and published in full in the articles [BHS13b, BHS13a].

The second main contribution is the characterization of normality as incompressibility by several different automata. Compression is achieved through transducers, which are automata augmented with an output tape. We prove that transducers, deterministic or not, even non-real time, with no extra memory or just a single counter, cannot compress normal infinite words. We exhaust all combinations of determinism, real-time, and additional memory and identify the models that can compress normal infinite words. These results answer the question of characterizations of normality by incompressibility for almost all possible combinations including memory models with counters and/or stacks, leaving open only the case of deterministic pushdown automata.

What happens when augmenting transducers with the ability to read the input several times instead of just one? These new transducers are called two-way transducers. We show that generalizing the definition of compressibility to this setting is not straightforward, but we prove the equivalence between the several possibilities, both for deterministic and non-deterministic two-way transducers. Then, we prove that the two-way enhancement does not improve the compressibility power of transducers. That is, two-way transducers, deterministic or non-deterministic, cannot compress normal numbers. We also show that the classical definition of compressibility does not generalize well to two-way transducers with extra unbounded memory, even in its simplest form: a single counter.

Results on compressibility are presented in Chapters 4 and 5. Chapter 4 contains the results on one-way automata, summarized in Table 4.1, which are also contained in our publication [BCH13]. The results on two-way automata appear in Chapter 5 and are published in [CH13].

The third main contribution is about preservation of normality. We consider the case of selection functions done by finite automata. Complementing Agafonov's theorem for prefix selection, which we present with a detailed proof, we show that suffix selection preserves normality too. Suffix selection involves non-deterministic machines, which makes the proof more interesting than a simple generalization. We also show simple two-sided selection rules that do not preserve normality. This is only the beginning of the study on preservation of normality. Functions other than selection rules remain to be studied. Even among selection rules, there are questions to be answered such as the existence of two-sided rules that do preserve normality, or generalization of prefix or suffix selection to more powerful automata. These results are in Chapter 6 of this document and published in [BCH13].

The formal presentation begins by introducing the basic definitions, properties and examples of normality with complete and elementary proofs. This presentation is in Chapter 2.

Presentation and methodology. To understand the proofs in this thesis it is not necessary to have background knowledge on normal numbers. The prerequisites are just basic results of mathematics and computer science and a few non-basic theorems in computability theory and in automata theory. The problems in this thesis are treated from first principles and the proofs are as elementary as possible. This work started by giving direct proofs of known theorems as we show it in Chapters 2, 4 and 6, and also in [BH13]. We believe that these direct proofs let us uncover the new mathematical truths in this dissertation.

CHAPTER 2

Normality and combinatorics on words

In this chapter we introduce the basic definitions and results of normality. Section 2.1 introduces the notation for the entire thesis. In Section 2.2 we prove some basic lemmas about word combinatorics. Section 2.3 gives the formal definition of normality. Section 2.4 introduces the concept of discrepancy, which is used to study normality. In Section 2.5 we prove the equivalency of several well known alternative definitions of normality, and more advanced lemmas to prove normality. Section 2.6 provides a list of interesting examples of normality. Finally, Section 2.7 introduces infinite de Bruijn words, a particular interesting example that requires more development than the previous.

2.1 Notation

2.1.1 Basics

As usual, we write $\mathbb{N}$ for the set of positive integers and $\mathbb{Z}$ for the set of all integers. If A is a finite set, we denote its cardinality by $|A|$. We write log without an explicit subindex as an abbreviation for $\log_2$, the logarithm in base 2. We write mod for the modulus operator, that is, if n and m are integers, $m \bmod n = m - n\lfloor m/n \rfloor$. We use $\mu()$ to denote Lebesgue measure [Leb02]. A function f is t-to-one if the function that maps each y to its number $|\{x : f(x) = y\}|$ of pre-images is bounded by t. A function is bounded-to-one if it is t-to-one for some integer t.

We use the asymptotic notation $O(f)$ and $o(f)$ to denote the family of functions bounded by some constant times f and the family of functions bounded by any constant times f, respectively. Namely, $O(f) = \{g : \exists c, n_0, \forall n > n_0, g(n) \leqslant cf(n)\}$ and $o(f) = \{g : \forall c, \exists n_0, \forall n > n_0, g(n) \leqslant cf(n)\}$. When we say that a function g is, or it is equal to, $O(f)$ or $o(f)$, it means that $g \in O(f)$ or $g \in o(f)$, respectively. For a more detailed presentation of the asymptotic notation, see [CSRL01].

As usual, a function $f : A \to B$ with countable A and B is computable when there is a Turing machine that realizes it, that is, when given some representation of A in the input tape, it finishes execution with a representation of the corresponding element of B in the output tape. A function $f : A \to B$ with uncountable A or B is computable if it is the limit of computable functions with countable domain and range. In particular, a real number is computable if there is a computable function that given n returns a rational within 2^{-n} of the real number. Thus, a real number is computable if it can be written down by a machine [Tur36]. Since irrational numbers require infinitely many digits, the machine writing it down never halts.

2.1.2 Words and reals

In this document, an alphabet is a finite set of at least two symbols. A word over alphabet A is a sequence of symbols from A, which can be finite or infinite. A^ℓ is the set of words of ℓ symbols from A, $A^{<\ell} = \bigcup_{0 \leqslant \ell' < \ell} A^{\ell'}$ is the set of words of less than ℓ symbols from A, $A^* = \bigcup_{\ell \geqslant 0} A^\ell$ is the set of all finite words over A and A^ω is the set of all infinite words over A.

The empty word is denoted by λ. The length of a finite word u is $|u|$. If a is a symbol, we also denote by a the word consisting of only the symbol a. If u is a finite word and x is a finite or infinite word, ux is the concatenation of u and then x. For a finite word u, u^n is equal to $uuu \cdots u$, with u repeated n times, and u^ω is equal to $uuuuu \cdots$ with u repeated infinitely many times. Finally, if u is a finite word, $\tilde{u}$ denotes the reverse of u, that is, the word that results from reading u from right to left.

Concatenation as juxtaposition and powering to integers, $*$ and ω are naturally extended to sets of words: if L is a set of finite words and u is a word, we define $uL = \{uv : v \in L\}$, $Lu = \{vu : v \in L\}$, $L^n = \{u_1u_2 \cdots u_n : u_i \in L\}$, $L^* = \bigcup_{n \geqslant 0} L^n$ and $L^\omega = \{u_1u_2u_3 \cdots : u_i \in L\}$. Juxtaposition can also be extended to pairs of sets of words, if L and L' are sets of words, $LL' = \{uv : u \in L, v \in L'\}$.

We number the symbols of words starting from 1. Given integers $1 \leqslant i \leqslant j$ such that x is either infinite or $|x| \geqslant j$, $x[i]$ is the symbol in position i and $x[i..j]$ is the finite word consisting of the symbols in positions i to j, in order. For an integer n, $x \upharpoonright n = x[1..n]$ is the word consisting of the first n symbols of x and $x \urcorner n$ is the word consisting of all but the first n symbols of x. Notice that $x \upharpoonright n$ is always a finite word, while $x \urcorner n$ is finite if and only if x is finite. We say that u is a subword of x if and only if $u = x[i..j]$ for appropriate i and j. We let

$$\operatorname{occ}(u, v) = |i : \{1 \leqslant i \leqslant |u| - |v| + 1 \text{ and } u[i..|v|] = v\}|$$

be the number of occurrences of v inside u and

$$\operatorname{boc}(u, v) = |i : \{1 \leqslant i \leqslant \lfloor |u|/|v| \rfloor \text{ and } u[(i-1)|v| + 1..i|v|] = v\}|$$

be the number of block occurrences of v inside u, that is, the number of occurrences inside the blocks of length $|v|$ of u.

We mostly use lowercase letters close to the beginning of the alphabet (a, b, c, d) to identify symbols, letters in the second half (s, t, u, v, w) to identify finite words and letters close to the end (x, y) to identify infinite words. Lowercase letters in the middle of the alphabet (i, j, k, ℓ, n, m) we mostly use for integers. Uppercase letters and Greek letters are reserved for more complex objects such as sets or tuples, or significant numerical constants.

A base is an integer greater than or equal to 2. When we say a word is in base b, we mean a word whose symbols are the digits of base b, that is, a word over the alphabet $\{0, 1, \cdots, b-1\}$. We identify a finite or infinite word $x = a_1a_2a_3 \cdots$ in base b with the real number $.x = \sum_i a_i b^{-i}$. Given a real number $r \in [0, 1)$ its base-b expansion $(r)_b$ is the unique infinite word x in base b such that $r = .x$ and infinitely many symbols of x are different from $b - 1$. Since reals are thereby identified with infinite words, we mostly use the same lowercase letters to represent them as we use for infinite words.

We use the phrase b-adic interval to refer to a semi-open interval I of the form $[a/b^m, (a+1)/b^m)$, for integers a, b and m such that $0 \leqslant a < b^m$. We move freely between b-adic intervals and base-b representations. Every finite word u in base b corresponds to a unique b-adic interval $[.u, .u + b^{-|u|})$. In this way, the base-b expansions of reals in a b-adic interval correspond to extensions of the finite word associated to it.

2.2 Some lemmas

The following technical lemmas are helpful to reduce equations concerning occurrences. We use it in the following without an explicit reference.

Lemma 2.2.1. *Let ℓ be a non-negative integer. For each word v over alphabet A,*

$$\sum_{u \in A^{|v|+\ell}} \operatorname{occ}(u, v) = (\ell+1)|A|^{\ell} \quad \text{and} \quad \sum_{u \in A^{|v|\ell}} \operatorname{boc}(u, v) = \ell |A|^{|v|(\ell-1)}.$$

Proof. To count the total number of occurrences of v in the set of words of length $|v| + \ell$, we count, for each possible position i, the number of u that contain v in position i. For occ there are $\ell + 1$ possible positions i at which an occurrence of v can begin, and there are $|A|^{\ell}$ ways of completing a word of length $|v| + \ell$ if we fix v occurring at position i. For boc, there are ℓ positions i at which an occurrence of v can begin, and there are $|A|^{|v|(\ell-1)}$ ways of completing a word of length $|v|\ell$ if we fix v occurring at position i. □

Lemma 2.2.2. *Let u, v be words over A such that $|v| < |u|$. Then,*

$$\operatorname{occ}(u, v) = \sum_{j=1}^{|v|} \operatorname{boc}(u \upharpoonright j-1, v).$$

Proof. Notice that $\operatorname{boc}(u \upharpoonright j-1, v)$ is the number of occurrences of v inside u at a position i such that $i \bmod |v| = j$. If we sum over each possible value of $i \bmod |v|$, we account for all occurrences. □

The following are basic results in Shannon's information theory [Sha48]. The next two lemmas are applied at the beginning of Chapter 4.

Lemma 2.2.3. *For fixed t and b, the unique maximum in $(0, t)$ of the function $p \mapsto -p \log_b p - (t-p) \log_b (t-p)$ is at $p = t/2$.*

Proof. Simple calculations show that the only zero of the derivative of the function is at $p = t/2$, and that the function at $p = t/2$ is greater than at $p = t/4$ or $p = 3t/4$. □

Corollary 2.2.4. *For fixed b, the unique maximum over the set of probability distributions $p_1, p_2, \ldots, p_n$ of the function*

$$p_1, p_2, \ldots, p_n \mapsto -\sum_{i=1}^{n} p_i \log_b p_i$$

is at $p_1 = p_2 = \cdots = p_n = 1/n$.

Proof. Assume a maximum of the function such that not every p_i is equal to $1/n$. So, there is i such that $p_i < 1/n$ and j such that $p_j > 1/n$. By Lemma 2.2.3, changing both those values to $(p_i + p_j)/2$ yields a larger value for the function. □

According to Shannon's source coding theorem, the optimal code length for a symbol is $-\log_b p$, where b is the number of symbols used to make output codes and p is the probability of the input symbol. The most common entropy encoding technique is Huffman coding [Huf52].

Definition. A set of finite words is *prefix-free* if no word in the set is a prefix of another word in the set.

Theorem 2.2.5. *A finite multiset of integers $\{\ell_1, \ell_2, \dots, \ell_n\}$ is the multiset of lengths of a prefix-free set of words over alphabet $|A|$ if and only if $\sum_{i=1}^{n} |A|^{-\ell_i} \leqslant 1$.*

Proof. Let $\{a_1, a_2, \dots, a_{|A|}\} = A$.

Let us begin with the "only if" part. Assume a set $L \subseteq A^*$ with minimum sum of word lengths $\sum_{i=1}^{n} |A|^{-\ell_i}$. such that for its multiset of lengths the claim does not hold. First, L cannot be the empty set, because on the empty set the claim holds. Then, if $\lambda \in L$, then L cannot contain any extension of λ, thus, $L = \{\lambda\}$ and its multiset of lengths is $\{0\}$ for which the claim holds. If $\lambda \notin L$, consider $L_{a_1}, L_{a_2}, \dots, L_{a_{|A|}}$ such that $L_a = \{u : au \in L\}$ are the words of L that start with a, with the starting a removed. By definition, L_a is prefix-free and with smaller sum of word lengths than L, therefore, $\sum_{u \in L_a} |A|^{-|u|} \leqslant 1$. By definition, $L = a_1 L_{a_1} \cup a_2 L_{a_2} \cup \dots \cup a_{|A|} L_{a_{|A|}}$, then

$$\sum_{u \in L} |A|^{-|u|} = \sum_{a \in A} \sum_{u \in L_a} |A|^{-|au|} = \sum_{a \in A} |A|^{-1} \sum_{u \in L_a} |A|^{-|u|} \leqslant \sum_{a \in A} |A|^{-1} = 1.$$

For the "if" part, if $n \leqslant |A|$ then the set $\{a_1{}^{\ell_1}, a_2{}^{\ell_2}, \dots, a_n{}^{\ell_n}\}$ proves the claim. Assume then that n is minimum and there is $\{\ell_1, \ell_2, \dots, \ell_n\}$ that fulfill the hypothesis but it is not the multiset of lengths of any prefix free set. Without loss of generality, assume $\ell_1 \leqslant \ell_2 \leqslant \dots \leqslant \ell_n$. Consider the multiset of lengths $\{\ell_1, \ell_2, \dots, \ell_{n-|A|}, \ell_{n-|A|+1} - 1\}$,

$$|A|^{-(\ell_{n-|A|+1}-1)} + \sum_{i=1}^{n-|A|} |A|^{-\ell_i} \leqslant |A||A|^{-\ell_{n-|A|+1}} + \sum_{i=1}^{n-|A|} |A|^{-\ell_i} =$$

$$\sum_{i=n-|A|+1}^{n} |A|^{-\ell_{i-|A|+1}} + \sum_{i=1}^{n-|A|} |A|^{-\ell_i} \leqslant \sum_{i=n-|A|+1}^{n} |A|^{-\ell_i} + \sum_{i=1}^{n-|A|} |A|^{-\ell_i} \leqslant 1.$$

Since the size of the new set is $n - |A| + 1 < n$, by hypothesis, there is a prefix free set $L' = \{u_1, u_2, \dots, u_{n-|A|}, v\}$ such that $|u_i| = \ell_i$ and $|v| = \ell_{n-|A|+1} - 1$. Consider the set $L = \{u_1, u_2, \dots, u_{n-|A|}, v(a_1^{\ell_{n-|A|+1}-|v|}), v(a_2^{\ell_{n-|A|+2}-|v|}), \dots, v(a_{|A|}^{\ell_n - |v|})\}$. Clearly, its multiset of lengths is $\{\ell_1, \ell_2, \dots, \ell_n\}$. Moreover, L is prefix free because L' is and $|v| < \ell_{n-|A|+j}$ for each $j = 1..|A|$, so there is no prefix within the extensions of v. □

2.3 Definition of normality

We come now to the most important definition of this thesis, the definition of normality. This definition was introduced by Émile Borel in 1909 [Bor09]. Several formulations based on combinatorics on words are equivalent to Borel's original formulation. We give our own presentation of the most important equivalencies in the following sections. There exist also several equivalent characterizations of normality in terms of number theoretic properties, see for instance [Bug12]. Another possible characterization of normality is in terms of incompressibility. This is the main subject of Chapters 4 and 5.

Definition. Let x be an infinite word over alphabet A, $r \in [0,1)$ be a real number and b a base. x is *simply normal* if for each symbol $a \in A$,

$$\lim_{n\to\infty} \mathrm{boc}(x \upharpoonright n, a)/n = 1/|A|.$$

x is *ℓ-simply normal* if for each finite word $u \in A^\ell$,

$$\lim_{n\to\infty} \mathrm{boc}(x \upharpoonright n\ell, u)/n = 1/|A|^\ell.$$

x is *normal* if it is ℓ-simply normal for every $\ell > 0$.
r is *(simply) normal* to base b if the infinite word $(r)_b$ is (simply) normal.
r is *absolutely normal* if it is normal to every base.
r is *absolutely abnormal* if it is normal to no base.

The following are some basic properties about normality.

Theorem 2.3.1. *A real number r is simply normal to base b^ℓ, if and only if $(r)_b$ is ℓ-simply normal.*

Proof. Let r be a real number and $x = (r)_b$. Notice in the definition of simple normality that the subword $u \in A^\ell$ being examined by the definition of boc is exactly the i-th digit in base $b^{|u|}$ of r. It follows immediately that the condition for u in the definition of ℓ-simply normal is equivalent to the condition of simple normality to base b^ℓ. $\square$

Corollary 2.3.2. *A real number is normal to base b if and only if it is simply normal to all bases that are powers of b.*

Proof. Immediate from Theorem 2.3.1 and the definition of normality and simple normality for real numbers. $\square$

Corollary 2.3.3. *A real number is absolutely normal if and only if it is simply normal to all bases.*

Proof. Immediate by Corollary 2.3.2 and the definitions of normal and simply normal. $\square$

Notice that absolute normality is equivalent to being simply normal to every base, but absolute abnormality is not equivalent to being not simply normal to every base. Recall that a real number is normal to either none or infinitely many bases.

Theorem 2.3.4. *If a real number is simply normal to just finitely many bases then it is absolutely abnormal. If a real number is normal to just finitely many bases then it is absolutely abnormal.*

Proof. Fix a base b. By contraposition, if R were normal to base b, then it would be simply normal to all bases b^ℓ, a contradiction to the hypothesis of the first claim. Simple normality to all bases b^ℓ implies by definition normality to all bases b^ℓ, which contradicts the hypothesis of the second claim. $\square$

The following lemma shows that occurrences of a finite subword within a normal infinite word cannot be too far apart.

Lemma 2.3.5. *Let x be a normal infinite word and w a word over the same alphabet A. There is an increasing function $h(n) = o(n)$ depending on x and w, such that for any n there is an occurrence of w in the word $x[n..n + h(n)]$.*

Proof. Let $x = v_1 v_2 v_3 \cdots$ where each $|v_i| = |w|$. Let $i_1, i_2, i_3, \cdots$ be the increasing sequence of occurrences of w in the blocks v_i, i.e., $v_i = w \Leftrightarrow i = i_n$. Then, by definition of normality

$$\lim_{n\to\infty} \frac{n}{i_n} = |A|^{-|w|} \quad \text{and} \quad \lim_{n\to\infty} \frac{n+1}{i_{n+1}} = \lim_{n\to\infty} \frac{n}{i_{n+1}} = |A|^{-|w|}$$

therefore,

$$\lim_{n\to\infty} \frac{i_n}{n} = \lim_{n\to\infty} \frac{i_{n+1}}{n} = |A|^{|w|}$$

and then

$$\lim_{n\to\infty} \frac{i_{n+1} - i_n}{n} = 0.$$

Letting $h(n) = |w| \max\{i_{m+1} - i_m : |w| i_{m+1} \leqslant n\}$ finishes the proof. □

Notice that since h is increasing, for large enough n there is also an occurrence of w in the word $x[n - h(n)..n]$.

The following is a technical lemma. In the sequel we use it several times without making explicit reference to it.

Lemma 2.3.6. *Let $f_1, f_2, f_3, \cdots$ be an increasing sequence of positive integers such that $f_n - f_{n-1}$ goes to zero as n goes to infinity. Then, for any word u and infinite word x,*

$$\lim_{n\to\infty} \frac{\operatorname{boc}(x \upharpoonright n, u)}{n} = \lim_{n\to\infty} \frac{\operatorname{boc}(x \upharpoonright f_n, u)}{f_n}$$

and

$$\lim_{n\to\infty} \frac{\operatorname{occ}(x \upharpoonright n, u)}{n} = \lim_{n\to\infty} \frac{\operatorname{occ}(x \upharpoonright f_n, u)}{f_n}.$$

Proof. Let m_n be such that $f_{m_n} < n \leqslant f_{m_n+1}$. Then,

$$\frac{\operatorname{boc}(x \upharpoonright n, u)}{n} \leqslant \frac{\operatorname{boc}(x \upharpoonright f_{m_n+1}, u)}{f_{m_n}} = \frac{\operatorname{boc}(x \upharpoonright f_{m_n+1}, u)}{f_{m_n+1}} \left(1 - \frac{f_{m_n+1} - f_{m_n}}{f_{m_n+1}}\right),$$

which implies

$$\lim_{n\to\infty} \frac{\operatorname{boc}(x \upharpoonright n, u)}{n} \leqslant \lim_{n\to\infty} \frac{\operatorname{boc}(x \upharpoonright f_{m_n+1}, u)}{f_{m_n+1}}.$$

Analogous arguments apply for the lower bound and for occ in place of boc. □

The following theorem provides a simple characterization of normality.

Theorem 2.3.7. *An infinite word is normal if and only if it is ℓ-simply normal for all but finitely many $\ell > 0$.*

Proof. The "only if" part is immediate from the definition. For the "if" part, assume x is ℓ-simply normal for all but finitely many ℓ and let m be the maximum length such that x is not m-simply normal. Fix an $\ell > 0$. Since $2\ell m > m$, x is $2\ell m$-simply normal, so for each $v \in A^{2\ell m}$,

$$\lim_{n\to\infty} \operatorname{boc}(x \upharpoonright n2\ell m, v)/n = |A|^{-2\ell m}.$$

Fix arbitrary $u \in A^\ell$. Since $2m/n$ goes to zero as n goes to infinity,

$$\begin{aligned}\lim_{n\to\infty} \frac{\operatorname{boc}(x \upharpoonright n\ell, u)}{n} &= \lim_{n\to\infty} \frac{\operatorname{boc}(x \upharpoonright n2m\ell, u)}{2mn} \\ &= \lim_{n\to\infty} \frac{\sum_{v\in A^{2m\ell}} \operatorname{boc}(x \upharpoonright n2m\ell, v) \operatorname{boc}(v,u)}{2mn} \\ &= |A|^{-2m\ell} \sum_{v\in A^{2m\ell}} \frac{\operatorname{boc}(v,u)}{2m} \\ &= |A|^{-2m\ell} \frac{|A|^{2m\ell-\ell}2m}{2m} = |A|^{-\ell}.\end{aligned}$$

Thus, x is ℓ-simply normal for every $\ell > 0$, and therefore, x is normal. □

A given property of infinite words is invariant under suffixes, or shift invariant, when x has the property if and only if any suffix of x has it. It is clear from the definition that simple normality is shift invariant. However, the following example shows that 2-simple normality is not. Consider the infinite word $(01011010)^\omega$ over the binary alphabet $\{0,1\}$. It is clearly 2-simply normal, because each word of length 2 occurs exactly once in the period. However $(00101101)^\omega \upharpoonleft 1 = (01011010)^\omega$ is not 2-simply normal, because the word 00 does not occur in the pattern. Of course 00 does occur as a subword of $(01011010)^\omega$, but since it does not occur properly aligned, $\operatorname{boc}((01011010)^\omega \upharpoonright n, 00) = 0$ for every n.

Even though ℓ-simple normality is not shift invariant for some ℓ, normality is, as shown by the following theorem.

Theorem 2.3.8. *Every suffix of an infinite normal word is normal.*

Proof. Let $x \in A^\omega$ be a normal word and $x \upharpoonleft n$ be a fixed suffix of x. Let us prove the result for $n = 1$. Since $x \upharpoonleft 0 = x$ and $x \upharpoonleft n+1 = (x \upharpoonleft n) \upharpoonleft 1$, the result for all $n \geqslant 0$ follows then by induction.

Assume x is normal. Fix ℓ and let us show that $x \upharpoonleft 1$ is ℓ-simply normal. For that, fix $u \in A^\ell$ and $\varepsilon > 0$ and let us show that $\lim_{n\to\infty} |\operatorname{boc}((x \upharpoonleft 1) \upharpoonright n\ell, u)/n - 1/|A|^\ell| \leqslant \varepsilon$.

Let $m = \lceil 2/\varepsilon \rceil$ and write $x = v_1 v_2 v_3 \cdots$ where each $|v_i| = m\ell$ to get

$$x \upharpoonleft 1 = (v_1 \upharpoonleft 1) v_2 v_3 \cdots = (v_1 \upharpoonleft 1)(v_2 \upharpoonright 1)(v_2 \upharpoonleft 1)(v_3 \upharpoonright 1)(v_3 \upharpoonleft 1)(v_4 \upharpoonright 1) \cdots .$$

Since m/n goes to zero as n goes to infinity,

$$\begin{aligned}&\left|\lim_{n\to\infty} \frac{\operatorname{boc}((x \upharpoonleft 1) \upharpoonright n\ell, u)}{n} - 1/|A|^\ell\right| = \left|\lim_{n\to\infty} \frac{\operatorname{boc}((x \upharpoonleft 1) \upharpoonright mn\ell, u)}{mn} - 1/|A|^\ell\right| \\ &= \left|\lim_{n\to\infty} \frac{\sum_{i=1}^n \operatorname{boc}((v_i \upharpoonleft 1)(v_{i+1} \upharpoonright 1), u)}{mn} - 1/|A|^\ell\right| \\ &\leqslant \left|\lim_{n\to\infty} \frac{\sum_{i=1}^n \operatorname{boc}((v_i \upharpoonleft 1), u)}{mn} - 1/|A|^\ell\right| + \frac{1}{m} \\ &\leqslant \left|\lim_{n\to\infty} \frac{\sum_{v\in A^{m\ell}} |\{i \leqslant n : v_i = v\}| \operatorname{boc}(v \upharpoonleft 1, u)}{mn} - 1/|A|^\ell\right| + \frac{1}{m} \\ &\leqslant \left|\lim_{n\to\infty} \frac{\sum_{v\in A^{m\ell}} \operatorname{boc}(v, x \upharpoonright n) \operatorname{boc}(v \upharpoonleft 1, u)}{mn} - 1/|A|^\ell\right| + \frac{1}{m} \\ &\leqslant \left|\frac{\sum_{v\in A^{m\ell}} \operatorname{boc}(v \upharpoonleft 1, u)}{|A|^{m\ell} m} - 1/|A|^\ell\right| + \frac{1}{m} \\ &\leqslant \left|\frac{\sum_{v\in A^{m\ell}} \operatorname{boc}(v, u)}{|A|^{m\ell} m} - 1/|A|^\ell\right| + \frac{2}{m} \\ &\leqslant \left|\frac{m|A|^m}{|A|^{m\ell} m} - 1/|A|^\ell\right| + \frac{2}{m} = \frac{2}{m} \leqslant \varepsilon.\end{aligned}$$

□

Corollary 2.3.9. *If $x \in A^\omega$ is normal, then for any finite word $u \in A^*$, ux is normal.*

Proof. By contraposition, assume that for some ℓ, ux is not ℓ-simply normal. So for every $m \geqslant 0$, $ux \upharpoonleft \ell m$ is not ℓ-simply normal. In particular $ux \upharpoonleft \ell|u|$ is not ℓ-simply normal, but $ux \upharpoonleft \ell|u|$ is a suffix of x, and by Theorem 2.3.8 it is ℓ-simply normal. This is a contradiction. Then, ux must be ℓ-simply normal. □

2.4 Discrepancy

The *simple discrepancy* of a finite word indicates the difference between the actual number of occurrences of the symbols and their expected average. The definition of normality can be given in terms of discrepancy.

Definition. Let u be a word over A. The *simple discrepancy* of u is

$$D(u,A) = \max\{|\operatorname{occ}(u,a)/|u| \ - \ 1/|A|\,| : a \in A\}.$$

The *ℓ-block discrepancy* of u is

$$D_\ell(u,A) = \max\{|\operatorname{occ}(u,v)/|u| \ - \ 1/|A|^\ell\,| : v \in A^\ell\}.$$

Notice that $D(u,A)$ is a number between 0 and $1 - 1/|A|$, $D_\ell(u,A)$ is a number between 0 and $1 - 1/|A|^\ell$ and $D(u,A^\ell) \neq D_\ell(u,A)$ because the first is by definition $D(u,A^\ell) = \max\{|\operatorname{boc}(u,v)/|u| - 1/|A|\,| : v \in A^\ell\}$ so it counts block occurrences instead of all occurrences. When dealing with real numbers in base b we write $D(u,b)$ as an abbreviation for $D(u,\{0,1,\cdots,b-1\})$.

The definition of discrepancy allows us to rewrite the definition of simply normal more concisely. An infinite word x over alphabet A is simply normal if and only if $\lim_{n\to\infty} D(x \upharpoonright n, A) = 0$. It is also possible to define normality using block discrepancy, but the proof of the equivalence is not a simple rewrite We establish it in Theorem 2.5.1, in the next section.

The next lemma bounds the discrepancy of a concatenation of words. We use it very often in the sequel, without making explicit reference to it.

Lemma 2.4.1. *If $u_1,\ldots,u_n$ are words over A,*

$$D(u_1u_2\cdots u_n, A) \leqslant \sum_{i=1}^{n} D(u_i,A)|u_i| \Bigg/ \sum_{j=1}^{n}|u_j|.$$

Proof. Let $a \in A$ be a symbol maximizing $|\operatorname{occ}(u_1u_2\cdots u_n,a)/|u_1u_2\cdots u_n| - 1/|A|\,|$.

$$\begin{aligned} D(u_1u_2\cdots u_n, A) &= |\operatorname{occ}(u_1u_2\cdots u_n,a)/|u_1u_2\cdots u_n| - 1/|A|\,| \\ &\leqslant \left| \left(\sum_{i=1}^{n} \operatorname{occ}(u_i,a) \Bigg/ \sum_{j=1}^{n}|u_j| \right) - \frac{1}{|A|} \right| \\ &\leqslant \left| \sum_{i=1}^{n} \operatorname{occ}(u_i,a) - \frac{|u_i|}{|A|} \right| \Bigg/ \sum_{j=1}^{n}|u_j| \\ &\leqslant \sum_{j=1}^{n} |u_i| \left| \frac{\operatorname{occ}(u_i,a)}{|u_i|} - \frac{1}{|A|} \right| \Bigg/ \sum_{j=1}^{n}|u_j| \\ &\leqslant \sum_{j=1}^{n} D(u_i,|A|)|u_i| \Bigg/ \sum_{j=1}^{n}|u_j|\,. \end{aligned}$$

□

For any alphabet, almost every sufficiently long word has small discrepancy (simple and block). We need an explicit bound for the number of words of a given length having larger simple discrepancy than a given value. The next lemma gives such a bound. We follow Hardy and Wright's classic text [HW60], but sharpen the value obtained there.

Let the number of words of length k in alphabet A where a given digit occurs exactly i times be $p_A(k,i) = \left(\begin{array}{c} k \\ i \end{array}\right)(|A|-1)^{k-i}$.

Lemma 2.4.2 ([HW60], adapted from Theorem 148)**.** *Let A be an alphabet and $k \geqslant 0$ be a word length. For every real ε such that $6/k \leqslant \varepsilon \leqslant 1/|A|$, $\sum\limits_{0 \leqslant i \leqslant k/|A| - \varepsilon k} p_A(k,i)$ and $\sum\limits_{k/|A|+\varepsilon k \leqslant i \leqslant k} p_A(k,i)$ are at most $|A|^k e^{-|A|\varepsilon^2 k/6}$.*

Proof. Observe that for each i such that $i \leqslant k/|A|$, $p_A(k,i-1) < p_A(k,i)$ holds; and for each i such that $i > k/|A|$, $p_A(k,i) < p_A(k,i-1)$. The strategy to prove the wanted bounds is to "shift" the first sum to the right by $m = \lfloor \varepsilon k/2 \rfloor$ positions, and the second sum to the left by $m+1$ positions. We start with the first sum. Let $a = k/|A| - \varepsilon k$. For each i such that $0 \leqslant i \leqslant a$,

$$p_A(k,i) = \frac{p_A(k,i)}{p_A(k,i+1)} \cdot \frac{p_A(k,i+1)}{p_A(k,i+2)} \cdot \ldots \cdot \frac{p_A(k,i+m-1)}{p_A(k,i+m)} \cdot p_A(k,i+m).$$

The largest quotient in the expression above is $\frac{p_A(k,i+m-1)}{p_A(k,i+m)}$. Using the symbolic expression for $p_A(k,i)$, $\dfrac{p_A(k,i)}{p_A(k,i+1)} = \dfrac{(i+1)(|A|-1)}{k-i}$. Then,

$$\begin{aligned}
\frac{p_A(k,i)}{p_A(k,i+1)} &\leqslant \frac{p_A(k,\lfloor a \rfloor + m - 1)}{p_A(k,\lfloor a \rfloor + m)} = \frac{(\lfloor a \rfloor + m)(|A|-1)}{k - \lfloor a \rfloor - m + 1} \\
&< \frac{(k/|A| - \varepsilon k/2)(|A|-1)}{k - k/|A| + \varepsilon k/2} = 1 - \frac{\varepsilon |A|/2}{1 - 1/|A| + \varepsilon/2} \\
&< 1 - \varepsilon\ |A|/2, \qquad \text{(using } \varepsilon \leqslant 1/|A|) \\
&< e^{-|A|\varepsilon/2}.
\end{aligned}$$

Since $m = \lfloor \varepsilon k/2 \rfloor$ and $\varepsilon k \geqslant 6$, it follows that

$$e^{-|A|\varepsilon m/2} \leqslant e^{-|A|\varepsilon(\varepsilon k/2-1)/2} = e^{-|A|\varepsilon^2 k/4 + |A|\varepsilon/2} \leqslant e^{-|A|\varepsilon^2 k/6}.$$

We obtain $p_A(k,i) < e^{-|A|\varepsilon^2 k/6}\ p_A(k,i+m)$. From $\sum_{0 \leqslant i \leqslant k} p_A(k,i) = |A|^k$ we can conclude

$$\sum_{0 \leqslant i \leqslant a} p_A(k,i) \ < e^{-|A|\varepsilon^2 k/6} \sum_{0 \leqslant i \leqslant a} p_A(k,i+m) \ \leqslant \ |A|^k e^{-|A|\varepsilon^2 k/6}.$$

To bound the second sum we shift the sum to the left by $m+1$ positions. Let $z = \lceil k/|A| + \varepsilon k \rceil$. For any integer i such that $z - m < i \leqslant k$,

$$p_A(k,i) = \frac{p_A(k,i)}{p_A(k,i-1)} \cdot \frac{p_A(k,i-1)}{p_A(k,i-2)} \cdot \ldots \cdot \frac{p_A(k,i-m)}{p_A(k,i-m-1)} \cdot p_A(k,i-m-1)$$

where these quotients increase as the indices decrease. So,

$$\begin{aligned}
\frac{p_A(k,i)}{p_A(k,i-1)} &\leqslant \frac{p_A(k,\lceil z \rceil - m)}{p_A(k,\lceil z \rceil - m - 1)} = \frac{k - \lceil z \rceil + m + 1}{(\lceil z \rceil - m)(|A|-1)} \\
&\leqslant \frac{k - k/|A| - \varepsilon k/2 + 1}{(k/|A| + \varepsilon k/2)(|A|-1)} \\
&< 1 - \varepsilon |A|/3.
\end{aligned}$$

To see the last inequality observe that it is equivalent to $\varepsilon|A| - 2/|A| - \varepsilon < 1 - \frac{6}{\varepsilon|A|k}$, which is implied by $1 - 2/|A| < 1 - \frac{6}{\varepsilon|A|k}$, because $\varepsilon|A| \leqslant 1$ and $\varepsilon > 3/k$. Therefore, $\dfrac{p_A(k,i)}{p_A(k,i-1)} < e^{-|A|\varepsilon/3}$. Then, using $m+1$,

$$p_A(k,i) < e^{-|A|\varepsilon(m+1)/3} p_A(k, i-m-1) \leqslant e^{-\frac{|A|\varepsilon^2 k}{6}} p_A(k, i-m-1).$$

From the last inequality and the fact $\sum\limits_{0 \leqslant i \leqslant k} p_A(k,i) = |A|^k$ we can conclude that $\sum\limits_{z \leqslant i \leqslant k} p_A(k,i) < |A|^k e^{-|A|\varepsilon^2 k/6}$. □

We can now give an upper bound for the number of words with simple discrepancy larger than a given value.

Lemma 2.4.3. *The number of words $u \in A^k$ such that $D(u,A) > \varepsilon$ is bounded by $2|A|^{k+1} e^{-|A|\varepsilon^2 k/6}$.*

Proof. Let $a \in A$ be a symbol. If $D(u,A) > \varepsilon$, then, for some symbol $a \in A$, $|\operatorname{occ}(u,a)/|u| - 1/|A|\,| > \varepsilon$. By Lemma 2.4.2, the number of words u of length k such that $|\operatorname{occ}(u,a)/|u| - 1/|A|\,| > \varepsilon$ is at most $2|A|^k e^{-|A|\varepsilon^2 k/6}$. Thus, counting the $|A|$ possible choices for a we obtain $|A| 2|A|^k e^{-|A|\varepsilon^2 k/6}$. □

Lemma 2.4.4. *Let $t \geqslant 2$ be an integer and let ε and δ be real numbers between 0 and 1, with $\varepsilon \leqslant 1/t$. Let k be the least integer greater than $\lceil 6/\varepsilon \rceil$ and $-\ln(\delta/2t)6/\varepsilon^2$. Then, for any alphabet A such that $|A| \leqslant t$ and for every integer $k' \geqslant k$, the fraction of words u over A of length k' for which $D(u,A) > \varepsilon$ is less than δ.*

Proof. By Lemma 2.4.3, the number of words u of length k such that $D(u,A) > \varepsilon$ is bounded by $2|A|^{k+1} e^{-|A|\varepsilon^2 k/6}$. To have this constitute a fraction of no more than δ of all the $|A|^k$ words, it is sufficient that $\delta > 2|A| e^{-\varepsilon^2 k/6}$. This is implied by $k > -\ln(\delta/(2|A|))6/\varepsilon^2$.

Since δ is less than or equal to 1, if $2 \leqslant |A| \leqslant t$ then $-\ln(\delta/(2t)) \geqslant -\ln(\delta/(2|A|))$, and hence $k \geqslant -\ln(\delta/(2|A|))6/\varepsilon^2$. Then for any $k' > k$, the number of words u' of length k' such that $D(u',A) > \varepsilon$ is a fraction of no more than δ of all the $|A|^{k'}$ sequences, as required. □

2.5 Equivalent definitions of normality

We state here two well known formulations of normality that are equivalent to the definition. In the next sections we will use one to show normality of several examples.

Theorem 2.5.1. *An infinite word x over alphabet A is normal if and only if for every $u \in A^*$, $\lim_{n\to\infty} \operatorname{occ}(x \upharpoonright n, u)/n = |A|^{-|u|}$.*

Theorem 2.5.2. *An infinite word x over alphabet A is normal if and only if there is a constant c such that for every $u \in A^*$, $\limsup_{n\to\infty} \operatorname{occ}(x \upharpoonright n, u)/n \leqslant c|A|^{-|u|}$.*

Proof of Theorems 2.5.1 and 2.5.2. We prove both theorems by showing the following are equivalent,

(1) $x \in A^\omega$ is normal.

(2) for every $u \in A^*$, $\lim_{n\to\infty} \operatorname{occ}(x \upharpoonright n, u)/n = |A|^{-|u|}$.

(3) there is c such that for every $u \in A^*$, $\limsup_{n\to\infty} \operatorname{occ}(x \upharpoonright n, u)/n \leqslant c|A|^{-|u|}$.
(4) for every $u \in A^*$, $\lim_{n\to\infty} \operatorname{boc}(x \upharpoonright n, u)/n \leqslant |A|^{-|u|}$.

To see (1) implies (2), assume x is normal and fix $u \in A^*$. By Theorem 2.3.8, the suffixes of x, $x \upharpoonleft 0, x \upharpoonleft 1, \ldots, x \upharpoonleft |u|-1$ are all normal. Since $|u|/n$ goes to zero when n goes to infinity,

$$\begin{aligned}\lim_{n\to\infty} \frac{\operatorname{occ}(x \upharpoonright n, u)}{n} &= \lim_{n\to\infty} \frac{\operatorname{occ}(x \upharpoonright n|u|, u)}{(n|u|)} \\ &= \lim_{n\to\infty} \sum_{j=1}^{|u|} \frac{\operatorname{boc}((x \upharpoonright n|u|) \upharpoonleft j-1, u)}{(n|u|)} \\ &= \sum_{j=1}^{|u|} \frac{|A|^{-|u|}}{|u|} = |A|^{-|u|}.\end{aligned}$$

(2) implies (3) is immediate setting $c = 1$.

To see (3) implies (4), fix x and c such that (3) holds. Fix a length ℓ and $\varepsilon > 0$ and let us show that for $u \in A^\ell$, $\lim_{n\to\infty} \operatorname{boc}(x \upharpoonright n, u)/n \leqslant |A|^{-|u|} + 2\varepsilon$. Since ℓ/n goes to zero as n goes to infinity, this is equivalent to showing

$$\lim_{n\to\infty} \operatorname{boc}(x \upharpoonright n\ell, u)/(n\ell) \leqslant |A|^{-|u|} + 2\varepsilon.$$

For the rest of the proof of this claim, fix the alphabet A^ℓ and interpret u as a symbol of the alphabet and x as a word over A^ℓ by grouping blocks of length ℓ into a single symbol. Notice that this change affects lengths and indices within words. In this notation, it is enough for us to prove $\lim_{n\to\infty} D(x \upharpoonright n, A^\ell) \leqslant 2\varepsilon$.

Fix k large enough such that $2|A^\ell|e^{-|A^\ell|\varepsilon^2 k/6} ck \leqslant \varepsilon$, which is possible because the expression decreases when k increases and its limit is 0. Let $x = v_1 v_2 v_3 \cdots$ with $|v_i| = k$. Let $V = \{v \in (A^\ell)^k : D(v, A^\ell) > \varepsilon\}$ be the set of words with high discrepancy. By Lemma 2.4.3, $|V| \leqslant 2|A^\ell|^{k+1} e^{-|A^\ell|\varepsilon^2 k/6}$.

$$\begin{aligned}\lim_{n\to\infty} D(x \upharpoonright n, A^\ell) &= \lim_{n\to\infty} D(v_1 v_2 \cdots v_{\lfloor n/k \rfloor}(v_{\lfloor n/k \rfloor+1} \upharpoonright n \bmod k), A^\ell) \\ &\leqslant \lim_{n\to\infty} D(v_{\lfloor n/k \rfloor+1} \upharpoonright n \bmod k, A^\ell)\frac{n \bmod k}{n} + \sum_{i=1}^{\lfloor n/k \rfloor} D(v_i, A^\ell)\frac{k}{n} \\ &\leqslant \lim_{n\to\infty} \sum_{i=1}^{\lfloor n/k \rfloor} D(v_i, A^\ell)\frac{k}{n}.\end{aligned}$$

Since we are assuming (3), among the v_is, in the limit, at most $c|A^\ell|^{-k}n$ can be equal to any particular v. Further, at most $|V|c|A^\ell|^{-k}n$ many is are such that $v_i \in V$,

$$\begin{aligned}\lim_{n\to\infty} D(x \upharpoonright n, A^\ell) &\leqslant \lim_{n\to\infty} \sum_{i=1}^{\lfloor n/k \rfloor} D(v_i, A^\ell)\frac{k}{n} \\ &\leqslant \lim_{n\to\infty} \left(\sum_{1 \leqslant i \leqslant \lfloor n/k \rfloor, v_i \in V} 1\frac{k}{n} \right) + \left(\sum_{1 \leqslant i \leqslant \lfloor n/k \rfloor, v_i \notin V} \varepsilon\frac{k}{n} \right) \\ &\leqslant \lim_{n\to\infty} |V|c|A|^{-k}n\frac{k}{n} + \left\lfloor \frac{n}{k} \right\rfloor \varepsilon\frac{k}{n} \\ &\leqslant |V|c|A^\ell|^{-k}k + \varepsilon \\ &\leqslant 2|A^\ell|^{k+1}e^{-|A^\ell|\varepsilon^2 k/6}c|A^\ell|^{-k}k + \varepsilon \\ &\leqslant 2|A^\ell|e^{-|A^\ell|\varepsilon^2 k/6}ck + \varepsilon \leqslant 2\varepsilon.\end{aligned}$$

Finally, we prove (4) implies (1). Assume (4) and the negation of (1). Then, x is not normal, hence it is not ℓ-simply normal for some ℓ, so there is $u_1 \in A^\ell$ and an increasing subsequence of positions $n_1, n_2, n_3, \cdots$ in which for a given word u_1, $\lim_{i\to\infty} \operatorname{boc}(x \restriction n_i, u_1)/n_i = f_1 \neq |A|^{-|u_1|}$. Let $\{u_1, u_2, \cdots, u_{|A|^\ell}\} = A^\ell$ be an enumeration of A^ℓ. Let $n_{1,i} = n_i$ and for each $j = 2, 3, \cdots, |A|^\ell$ define $n_{j,1}, n_{j,2}, n_{j,3}, \cdots$ a subsequence of $n_{j-1,1}, n_{j-1,2}, n_{j-1,3}, \cdots$ such that $\lim_{i\to\infty} \operatorname{boc}(x \restriction n_{j,i}, u_j)/n_{j,i} = f_j$ exists. Thus, on the sequence of positions $n_{|A|^\ell,1}, n_{|A|^\ell,2}, n_{|A|^\ell,3}, \cdots$ the limit exists for all words of length ℓ. By (4), each $f_j \leqslant |A|^{-\ell}$ and $f_1 < |A|^{-\ell}$. This is a contradiction because if all limits exist, the sum of the limiting frequencies $\sum_{j=1}^{|A|^\ell} f_j$ needs to be 1. $\square$

Lemma 2.5.3. *Let $u_1, u_2, u_3, \ldots$ be a sequence of finite words over A of non-decreasing length such that for every $u \in A^*$, $\lim_{n\to\infty} \operatorname{occ}(u_n, u)/|u_n| = |A|^{-|u|}$ and $\sup\{|u_{n+1}|/|u_1 u_2 \cdots u_n| : n \geqslant 1\} = k < \infty$. Then, $u_1 u_2 u_3 \cdots$ is normal.*

Proof. Let us set $x = u_1 u_2 u_3 \cdots$. Fix a word u over A and let us show that $\limsup_{n\to\infty} \operatorname{occ}(x \restriction n, u)/n \leqslant (k+1)|A|^{-|u|}$. Then, by Theorem 2.5.2, x is normal. To prove the claim, we fix $\varepsilon > 0$ and show that, for sufficiently large n, $\operatorname{occ}(x \restriction n, u)/n \leqslant (k+1)|A|^{-|u|} + \varepsilon$.

Notice that the hypothesis implies that $\lim_{n\to\infty} |u_n| = \infty$. By hypothesis, let i_0, i_1, i_2 be integers such that

for every $i \geqslant i_0$, $\operatorname{occ}(u_i, u)/|u_i| \leqslant |A|^{-|u|} + \varepsilon/(3(k+1))$,

for every $i \geqslant i_1$, $i|u|/|u_1 u_2 \cdots u_i| \leqslant \varepsilon/(3(k+1))$ and

for every $i \geqslant i_2$, $|u_1 u_2 \cdots u_{i_0-1}|/|u_1 u_2 \cdots u_i| \leqslant \varepsilon/(3(k+1))$.

For given n, let i be the index such that $|u_1 u_2 \cdots u_i| < n \leqslant |u_1 u_2 \cdots u_{i+1}|$. Assume n is large enough to make $i \geqslant \max(i_0, i_1, i_2)$.

$$\begin{aligned}
\operatorname{occ}(x \restriction n, u)/n &= \operatorname{occ}((u_1 u_2 \cdots u_i u_{i+1}) \restriction n, u)/n \\
&\leqslant \operatorname{occ}(u_1 u_2 \cdots u_i u_{i+1}, u)/|u_1 u_2 \cdots u_i| \\
&\leqslant (k+1) \operatorname{occ}(u_1 u_2 \cdots u_i u_{i+1}, u)/|u_1 u_2 \cdots u_i u_{i+1}| \\
&\leqslant (k+1) \frac{i|u| + \sum_{j=1}^{i+1} \operatorname{occ}(u_j, u)}{|u_1 u_2 \cdots u_i u_{i+1}|} \\
&\leqslant (k+1) \left(\sum_{j=1}^{i_0-1} \frac{\operatorname{occ}(u_j, u)}{|u_1 u_2 \cdots u_i u_{i+1}|} + \sum_{j=i_0}^{i+1} \frac{\operatorname{occ}(u_j, u)}{|u_1 u_2 \cdots u_i u_{i+1}|} \right) + \frac{\varepsilon}{3} \\
&\leqslant (k+1) \left(\sum_{j=1}^{i_0-1} \frac{|u_j|}{|u_1 u_2 \cdots u_i u_{i+1}|} + \sum_{j=i_0}^{i+1} \frac{\operatorname{occ}(u_j, u)}{|u_1 u_2 \cdots u_i u_{i+1}|} \right) + \frac{\varepsilon}{3} \\
&\leqslant (k+1) \left(\sum_{j=i_0}^{i+1} \frac{\operatorname{occ}(u_j, u)}{|u_1 u_2 \cdots u_i u_{i+1}|} \right) + \frac{2\varepsilon}{3} \\
&\leqslant (k+1) \left(\sum_{j=i_0}^{i+1} \frac{(|A|^{-|u|} + \varepsilon/(3(k+1)))|u_j|}{|u_{i_0} u_{i_0+1} \cdots u_i u_{i+1}|} \right) + \frac{2\varepsilon}{3} \\
&\leqslant (k+1)|A|^{-|u|} + \varepsilon.
\end{aligned}$$

$\square$

2.6 Simple examples

Consider first the infinite words 0^ω, $(001)^\omega$ and $(01)^\omega$ over the binary alphabet $\{0,1\}$. The first two are not simply normal, because the frequency of 0 is higher than the frequency of 1. The last one is simply normal because the frequency of 0 and 1 is clearly 1/2. However, it is not 2-simply normal because the word 00 does not even occur in it, so its frequency is 0 instead of the required 1/4. This shows that an infinite word can be simply normal but not normal.

In this section we prove normality for some simple infinite words over an arbitrary alphabet A. The next section elaborates a more sophisticated case.

We introduce three families of normal infinite words inspired on Champernowne's original example [Cha33].

Definition. Fix an alphabet A. For each length $\ell \geqslant 1$, let w_ℓ be the concatenation, in lexicographical order, of all words of length ℓ, namely, $w_\ell = w_{\ell,1}w_{\ell,2}\cdots w_{\ell,|A|^\ell}$ where $w_{\ell,i}$ is the i-th word in lexicographic order of the set A^ℓ.

The *complete Champernowne* constant for A is $w_1w_2w_3\cdots$.
The *palindromic Champernowne* constant for A is $w_1\tilde{w}_1w_2\tilde{w}_2w_3\tilde{w}_3\cdots$.
The *delayed Champernowne* constant for A is ${w_1}^{1!^2}{w_2}^{2!^2}{w_3}^{3!^2}\cdots$.

Lemma 2.6.1. *Fix an alphabet A. For w_ℓ defined as in the definition of Champernowne constants, for every $u \in A^*$, $\lim_{\ell\to\infty} \mathrm{occ}(w_\ell, u)/|w_\ell| = |A|^{-|u|}$.*

Proof. Let w_ℓ and $w_{\ell,i}$ be as in the definition of Champernowne constants. Within w_ℓ less than $|A|^\ell|u|$ occurrences of u do not lie completely within some $w_{\ell,i}$; therefore,

$$\begin{aligned}\lim_{\ell\to\infty}\frac{\mathrm{occ}(w_\ell,u)}{|w_\ell|} &\leqslant \lim_{\ell\to\infty}\frac{|A|^\ell|u| + \sum_{i=1}^{|A|^\ell}\mathrm{occ}(w_{\ell,i},u)}{|A|^\ell\ell}\\ &\leqslant \lim_{\ell\to\infty}\frac{|A|^\ell|u| + (\ell-|u|+1)|A|^{\ell-|u|}}{|A|^\ell\ell} = |A|^{-|u|}.\end{aligned}$$

Also,

$$\begin{aligned}\lim_{\ell\to\infty}\frac{\mathrm{occ}(w_\ell,u)}{|w_\ell|} &\geqslant \lim_{\ell\to\infty}\frac{\sum_{i=1}^{|A|^\ell}\mathrm{occ}(w_{\ell,i},u)}{|A|^\ell\ell}\\ &\geqslant \lim_{\ell\to\infty}\frac{(\ell-|u|+1)|A|^{\ell-|u|}}{|A|^\ell\ell} = |A|^{-|u|}.\end{aligned}$$

Putting both inequalities together yields the desired result. □

Theorem 2.6.2. *The complete Champernowne, palindromic Champernowne and delayed Champernowne constants for any alphabet are normal.*

Proof. Consider applying Lemma 2.5.3 to all 3 cases. The complete Champernowne constant and the delayed Champernowne constants are normal directly by Lemma 2.6.1, because they can be factorized into parts equal to some w_ℓ, and the indices of those w_ℓ are increasing. For the palindromic Champernowne constant, notice that $\mathrm{occ}(w_\ell,u) = \mathrm{occ}(\tilde{w}_\ell,\tilde{u})$, therefore, Lemma 2.6.1 is also true if we take $\tilde{w}_\ell$ instead of w_ℓ, therefore, the limit holds also for interleaving both sequences $w_1, w_2, w_3, \ldots$ and $\tilde{w}_1, \tilde{w}_2, \tilde{w}_3, \ldots$. □

The complete Champernowne constant is interesting because it is an illustration of the earliest example of a normal infinite word. The palindromic version we use in the following chapters. The delayed version is interesting because it is extremely well approximated by rationals, as the following theorem illustrates.

Theorem 2.6.3. *The delayed Champernowne constant for the alphabet of base b, $\{0, 1, \ldots, b-1\}$, interpreted as a real number in base b, is a Liouville number. That is, for each n there is a rational p/q that is within $1/q^n$ of it.*

Proof. Fix base b and n. Let $x = {w_1}^{1!^2}{w_2}^{2!^2}{w_3}^{3!^2}\cdots$ and the rational approximation $x_n = {w_1}^{1!^2}{w_2}^{2!^2}\cdots{w_{n-1}}^{(n-1)!^2}{w_n}^{\omega}$. The rational $.x_n$ can be written with $\sum_{i=1}^{n-1} i!^2|w_i|$ fixed digits and a period of $|w_n|$ digits. Then, the denominator of one of the fractions representing $.x_n$ has $|w_n| + \sum_{i=1}^{n-1} i!^2|w_i|$ digits, so it is lower than $b^{|w_n|+\sum_{i=1}^{n-1} i!^2|w_i|}$. Also, x and x_n have the first $\sum_{i=1}^{n} i!^2|w_i|$ digits in common; hence, they are within $b^{-\sum_{i=1}^{n} i!^2|w_i|}$ of each other. We just need to prove that $\sum_{i=1}^{n} i!^2|w_i|/(|w_n| + \sum_{i=1}^{n-1} i!^2|w_i|)$ is greater than n,

$$\frac{\sum_{i=1}^{n} i!^2|w_i|}{|w_n| + \sum_{i=1}^{n-1} i!^2|w_i|} > \frac{n!^2|w_n|}{|w_n| + \sum_{i=1}^{n-1}(n-1)!^2|w_n|} > \frac{n!^2}{1+(n-1)(n-1)!^2} > n.$$

□

Notice that delayed Champernowne constant is calibrated to meet Liouville's condition. However, it would be straightforward to delay the usage of each w_i even more to meet even tighter requirements of rational approximations. This is a simplified version of the idea from [NV12]. In [Bug02] the existence of absolutely normal Liouville numbers is proven. We recently submitted an article proving the existence of computable instances [BHS14].

Our last explicit example is a direct consequence of Lemma 2.5.3.

Theorem 2.6.4. *If $x \in A^\omega$ is normal, the infinite word $(x \upharpoonright 1)(x \upharpoonright 2)(x \upharpoonright 3)\cdots$, called the* triangulation *of x, is also normal.*

Proof. By setting $u_n = x \upharpoonright n$ we can apply Lemma 2.5.3 because its hypothesis is exactly what we get from normality of x and Theorem 2.5.1. □

The previous theorem can be easily generalized to a concatenation of prefixes of lengths given by any non-decreasing sequence, and the same proof applies.

Moreover, from the way the triangulation is constructed, each word $x \upharpoonright n$ has all but the last symbol in common with the previous one $x \upharpoonright (n-1)$, making the triangulation of any x extremely compressible with Lempel and Ziv's algorithm [ZL78]. This shows that LZ compression compresses not only every non-normal infinite word, but also, some normal infinite words. We consider the question of incompressibility of normal infinite words in Chapters 4 and 5.

2.7 Infinite de Bruijn words

Infinite de Bruijn words are nice instances of normal infinite words. This has been proved first by Edgardo Ugalde [Uga00]. We start by proving that infinite de Bruijn words exist. An earlier presentation of the first results in this section can also be found in our article [BH11].

Definition ([Bru46, SM94])**.** A (non cyclic) *de Bruijn word* of order n over A is a word of length $|A|^n + n - 1$ such that every word of length n occurs exactly once as a consecutive substring.

See [BP07] for a fine presentation and history of de Bruijn words. We give first the proof of the following theorem.

Theorem 2.7.1. *Every de Bruijn word of order n over A with $|A| \geqslant 3$ can be extended to a de Bruijn word of order $n+1$. Every de Bruijn word of order n over A with $|A| = 2$ can* not *be extended to order $n+1$, but it can be extended to order $n+2$.*

Recall that a *Hamiltonian* cycle of a graph is a cycle in which each vertex of the graph occurs exactly once. An *Eulerian* cycle is a cycle in which each edge of the graph occurs exactly once. A graph that admits an Eulerian cycle is called an *Eulerian* graph. An undirected graph is *connected* if for every pair of vertices, there is a path between them. A directed graph is *strongly connected* if for every pair of vertices there is a directed path between them. A directed graph is *regular* if each vertex has the same number of incoming and outgoing edges as all other vertices. Given a directed graph G, its *line graph* is a directed graph whose vertices are the edges of G, and whose edges correspond to the directed paths of length two of G.

For a fixed alphabet A, a de Bruijn graph of order n, which we denote by G_n, is a graph whose vertices are all words of length n, and the edges link overlapping words w, v such that $w[2..n] = v[1..n-1]$. The edges of G_n can be labeled with words of length $n+1$, such that the edge (w, v) is labeled with $w[1]v = w(v[n])$. Then, each possible word of length $n+1$ in k symbols appears in exactly one edge of G_n. Moreover, the line graph of G_n is exactly G_{n+1}. The label of a path $v_1, ..., v_t$ in G_n is the word that contains as subwords exactly the words $v_1, ..., v_t$, in that order, namely, $v_1[1]v_2[1]...v_{t-1}[1]v_t$. Note that the label of a path of length t is a word of length $t+n-1$. If we take a path of length t in G_n and consider the set of $t-1$ traversed edges, it is easy to see that they form a path that has the same label in G_{n+1}. Consequently, the label of a Hamiltonian cycle in G_n is a de Bruijn word of order n, and the label of an Eulerian cycle in G_n is a de Bruijn word of order $n+1$, because an Eulerian cycle in G_n is a Hamiltonian cycle in G_{n+1}.

We base the proof of Theorem 2.7.1 in the characterization of Eulerian directed graphs by I.J. Good [Goo46], which states that a directed graph is Eulerian if and only if it is strongly connected and the in-degree and out-degree of each vertex coincide.

Proposition 2.7.2 (folklore)**.** *A directed graph G in which each vertex has its in-degree equal to its out-degree is strongly connected if and only if its underlying undirected graph is connected.*

Proof. The "only if" is immediate. Fix G and let $d(v)$ be the in and out-degree of vertex v in G. Let u be an arbitrary vertex of G. Let U be the set of vertices that are accessible from u, that is, the smallest set such that (1) $u \in U$, and (2) for every edge $(v, w) \in G$, if $v \in U$ then $w \in U$. Note that, by definition, there is a directed path from u to each of the vertices in U. Let G' be the subgraph of G induced by U. It is clear that each vertex v in G' has out-degree $d(v)$, because every outgoing edge in G that has its first endpoint in U is in G' by condition (2). Also, the in-degree of each vertex v in G' is less than or equal to $d(v)$ because G' is a subgraph of G. Since the sum of the in-degrees is the same as the sum of the out-degrees in G', the in-degree of vertex v in G' must also be $d(v)$. Therefore, if $w \notin G'$, then there is no edge in G, in any direction, that connects w to any vertex v in G' (because that would make the in- or out-degree of v greater than $d(v)$). Since the underlying graph of G is connected, there is no such w, which implies $G' = G$. Since every vertex in G' is accessible from u, every vertex in G is accessible from u. Since this is valid for any u, G is strongly connected. □

Lemma 2.7.3. *A Hamiltonian cycle in a de Bruijn graph over an alphabet of at least three symbols can be extended to an Eulerian cycle in the same graph.*

Proof. Let H be a Hamiltonian path in G_n. Let I be the graph resulting of removing the edges in H from G_n. We first prove that the underlying undirected graph of I is connected. For an arbitrary pair of vertices u, v, we recursively define below a sequence of pairs u_i,v_i, for $0 \leqslant i \leqslant n$, satisfying the following properties:

(1) $u = u_0$ and $v = v_0$.

(2) For each $i < n$, there is an edge from u_i to u_{i+1} in I. Analogously for v_i.

(3) The last i symbols of u_i and v_i coincide.

Let $u_0 = u$ and $v_0 = v$. For each $i = 0..n-1$, let a_{i+1} be such that setting $u_{i+1} = u_i[2..n]a_{i+1}$ and $v_{i+1} = v_i[2..n]a_{i+1}$ make the edges (u_i, u_{i+1}) and (v_i, v_{i+1}) not belong to H. Such a symbol a_{i+1} exists because each vertex has exactly one of its (at least three) outgoing edges used in H. By definition the edges (u_i, u_{i+1}) and (v_i, v_{i+1}) are in G_n but not in H, hence they are in I. If the last $i < n$ symbols of v_i and u_i coincide, so do the last $i + 1$ symbols of u_{i+1} and v_{i+1}, because they are the last i symbols of u_i plus a_{i+1}. By condition (3), $v_n = u_n$, thus, u and v belong to the same connected component of the underlying undirected graph of I. Since this is valid for any pair u,v, the underlying undirected graph of I is connected.

Since G_n and H are regular, I is also regular. Then, by Proposition 2.7.2, I is strongly connected. These two properties ensure I is Eulerian. Adding an Eulerian cycle of I to H gives the desired extension. □

Lemma 2.7.4. *Lemma 2.7.3 fails if the alphabet has just two symbols.*

Proof. Consider the de Bruijn graph of order 1. It has just one Hamiltonian cycle. The removal of this cycle leaves the two points of the graph disconnected, with a self-loop edge each. As argued by Lempel in [Lem70], the same failure occurs at every order, because removing a Hamiltonian cycle from the graph leaves the self loops (that always exist in the vertices of the form a^n) isolated in the residual graph. □

Lemma 2.7.5. *A Hamiltonian cycle in a de Bruijn graph in two symbols can be extended to an Eulerian cycle in the de Bruijn graph of the next order.*

Proof. Let H be a Hamiltonian cycle in G_n. Since G_{n+1} is the line graph of G_n, H corresponds to a simple cycle in G_{n+1} that goes through half of the points of G_{n+1}. Let I be the graph that results from removing the edges in H from G_{n+1}. We first prove that the underlying undirected graph of I is connected. Let us call the two symbols of the alphabet 0 and 1. Note that for any word s of length n, H contains exactly one of the two vertices $s0$ and $s1$ of G_{n+1}. This is because s corresponds exactly to one vertex in G_n, and $s0$ and $s1$ correspond to its outgoing edges. Since H is a Hamiltonian cycle in G_n, exactly one of these edges is used in H. This in turn implies that any vertex in G_{n+1} has exactly one successor in H and one successor not in H. For an arbitrary pair of vertices u, v in G_{n+1}, we recursively define below a sequence of pairs u_i,v_i, for $0 \leqslant i \leqslant n+1$, satisfying the following properties:

(1) For each i, at least one of u_i or v_i is not in H.

(2) There is an edge from u to u_0, and there is an edge from v to v_0.

(3) For each $i \leqslant n$, there is an edge from u_i to u_{i+1} in I. Analogously for v_i.

(4) The last i symbols of u_i and v_i coincide.

Let u_0 be any successor of u and v_0 be the successor of v not in H. For $0 \leqslant i \leqslant n$, if $v_i \notin H$, let a_{i+1} be such that $u_i[2..n+1]a_{i+1}$ is not in H. Otherwise, let a_{i+1} be such that $v_i[2..n+1]a_{i+1}$ is not in H. Then, set $u_{i+1} = u_i[2..n+1]a_{i+1}$ and $v_{i+1} = u_i[2..n+1]a_{i+1}$. By definition, at least one of the endpoints of both (u_i, u_{i+1}) and (v_i, v_{i+1}) is not in H, so both edges are in I. Moreover, at least one of u_{i+1} and v_{i+1} is not in H. If the last $i < n$ symbols of v_i and u_i coincide, so do the last $i+1$ symbols of u_{i+1} and v_{i+1}, because they are the last i symbols of u_i plus a_{i+1}. By condition (4), $u_{n+1} = v_{n+1}$, thus, u and v belong to the same connected component of the underlying undirected graph of I. Since this is valid for any pair u,v, the underlying undirected graph of I is connected.

Every vertex in I has its in-degree equal to its out-degree, because it is a cycle subtracted from a regular graph. So, by Proposition 2.7.2, I is strongly connected. These two properties ensure I is Eulerian. Adding an Eulerian cycle in I to H gives the desired extension. □

We are now ready to give the proof of the already stated Theorem 2.7.1:

Proof of Theorem 2.7.1. de Bruijn words of order n correspond exactly to the Hamiltonian cycles in de Bruijn graphs G_n. In turn, the Hamiltonian cycles in G_{n+1} are exactly the Eulerian cycles in G_n. Now the first assertion follows from Lemma 2.7.3 and the second from Lemma 2.7.4 and Lemma 2.7.5. □

This property allows to define infinite de Bruijn as the limit of the extensions. This turn out to be normal infinite words, providing the last example of normality of this chapter.

Definition. An infinite word over an alphabet of at least three symbols is an *infinite de Bruijn word* if it is the inductive limit of extending de Bruijn words of order n, for each n. In case of a two-symbol alphabet, an infinite de Bruijn word is the limit of extending de Bruijn words of order $2n$, for each n.

Corollary 2.7.6. *Infinite de Bruijn words exist over any alphabet.*

Proof. Immediate from Theorem 2.7.1. □

Now we prove that infinite de Bruijn words are normal.

Lemma 2.7.7. *If $u \in A^\ell$ is a word of length ℓ, then it occurs in a de Bruijn word of order $n \geqslant \ell$ between $|A|^{n-\ell}$ and $|A|^{n-\ell} + n - \ell$ times.*

Proof. If u occurs at a position $1 \leqslant i \leqslant |A|^n$, then it is the beginning of an occurrence of a word of length n. There are exactly $|A|^{n-\ell}$ words of length n whose first ℓ symbols are u. Then, there are exactly $n - \ell$ other positions in a de Bruijn word of order n at which a subword of length ℓ may start. □

Theorem 2.7.8. *Infinite de Bruijn words are normal.*

Proof. Let x be an infinite de Bruijn word over A. Fix a word u of length ℓ and $n > |A|^\ell + \ell - 1$. By definition, for each n, $x \upharpoonright |A|^{2n} + 2n - 1$ (multiplication by 2 is to account for $|A| = 2$) is a de Bruijn word. Thus, we can bound the number of occurrences of u in any given prefix of x by the number of occurrences of u in the shortest prefix that is a de Bruijn word, yielding the bound

$$\limsup_{k\to\infty} \frac{\mathrm{occ}(u, x \upharpoonright k)}{k} \leqslant \limsup_{n\to\infty} \frac{\mathrm{occ}(u, x \upharpoonright |A|^{2n} + 2n - 1)}{|A|^{2n-2} + 2n - 3}$$

Further, using Lemma 2.7.7,

$$\limsup_{n\to\infty} \frac{\mathrm{occ}(u, x \upharpoonright |A|^{2n} + 2n - 1)}{|A|^{2n-2} + 2n - 3} \leqslant \limsup_{n\to\infty} \frac{|A|^{2n-\ell} + 2n - \ell}{|A|^{2n-2} + 2n - 3} = |A|^{-\ell+2}.$$

Therefore, by Theorem 2.5.2 using $c = |A|^2$, x is normal. □

Theorems 2.7.1 and 2.7.8 raise the question of giving efficient algorithms to construct infinite de Bruijn words. According to the proof of Lemmas 2.7.3 and 2.7.5, the extension problem is just to construct an Eulerian cycle in the remaining graph. Any of the known algorithms for Eulerian cycles is usable, even the ancient algorithm of Fleury [Fle83]. However, this may not be the most efficient way of proceeding, as it requires the output to be computed in chunks that grow exponentially.

If instead of infinite words we concentrate on real numbers, questions about normality to different bases naturally arise. In particular, we can wonder about absolute normality and absolute abnormality. Rational numbers, being periodic, are not normal to any base, and thus, are examples of absolutely abnormal numbers. Borel conjectured that all irrational algebraic numbers are absolutely normal, and this is one of the most important open problems on normal numbers today. Also it has been conjectured that usual mathematical constants such as π and e are absolutely normal. So far we have just some empirical evidence [BC01, SNS13].

The construction of particular instances of absolutely normal real numbers has not been satisfactory nor easy. We concentrate on this problem in Chapter 3.

CHAPTER 3

Complexity

In this chapter we consider constructing absolutely normal numbers computably. We devise a general method that yields two main results. One is an algorithm with low time complexity to compute an absolutely normal number. The other is a completeness result on the descriptive complexity of the set of absolutely normal numbers.

The first algorithm that computes an absolutely normal number was given by Alan Turing [Tur92] but only became known after it was reconstructed and proved to be correct by Becher, Figueira and Picchi [BFP07]. A similar algorithm is the computable reformulation [BF02] of Wacław Sierpiński's construction [Sie17]. Unfortunately, these two algorithms are extremely inefficient in terms of the time it takes to compute the first n digits of the number. They both require time that is double exponential in n. Another construction of an absolutely normal number was given by Wolfgang Schmidt [Sch62]. He remarks his number is "clearly defined" and, in fact, it is clearly computable, but its time complexity has not been analyzed.

Jack Lutz and Elvira Mayordomo were the first to announce the existence of an absolutely normal number computable in polynomial time; they did it in Seventh International Conference on Computability, Complexity and Randomness, Cambridge, United Kingdom, in July 2012. Santiago Figueira and André Nies reported another proof. See [May13] and [FN13] for details. Their arguments analyze polynomial-time martingales, a device from the theory of algorithmic randomness, and the means to diagonalize against them. In contrast, the algorithm we present as the first result here constructs the expansion of an absolutely normal number directly from the definition, bounding how far away from normality it can go. The technique is thus combinatorial. This result is published in [BHS13b].

Alexander Kechris, in the early 1990s, was interested in knowing how difficult it is to describe the set of absolutely normal numbers. Specifically, he asked whether the set of real numbers which are normal to base two is complete for the class $\mathbf{\Pi}^0_3$ in the Borel hierarchy. The Borel hierarchy is a categorization of sets according to how difficult they are to describe with first order logic formulas. A set is $\mathbf{\Pi}^0_n$ if it can be described with a sentence using no more than n alternating quantifiers, starting with a $\forall$. Such a set is complete for the class $\mathbf{\Pi}^0_n$ if it is not possible to describe it in a simpler way. The classes $\mathbf{\Sigma}^0_n$ are analogous, but consider the alternation of quantifiers starting with $\exists$. The effective Borel hierarchy is a similar categorization that requires the formula after the quantifiers to be computable.

Ki and Linton [KL94] proved that, indeed, the set of numbers that are normal to any fixed base is complete in the $\boldsymbol{\Pi}^0_3$ class. However, their proof technique does not extend to the case of absolute normality, that is, normality to all bases simultaneously, nor to the effective Borel hierarchy. We show that the set of absolutely normal numbers is also $\boldsymbol{\Pi}^0_3$-complete. In fact, the set of absolutely normal numbers and the set of normal numbers to any fixed base are complete for the Π^0_3 class in the effective Borel hierarchy. We give, as a second result, an explicit reduction that proves the two completeness results. Our work on this is published in [BHS13a].

In Section 3.1 we develop some tools to construct real numbers computably. In Section 3.2 we present our algorithm that computes absolutely normal numbers and prove its correctness. In Section 3.3 we give the necessary implementation details and we show that its time complexity is just above quadratic. Section 3.4 recalls the Borel hierarchy on subsets of real numbers and discusses the descriptive complexity of the set of absolutely normal numbers. Sections 3.5 and 3.6 give two algorithms that are the base to show in Section 3.7 a reduction that proves the set of absolutely normal numbers is complete in its level of the Borel hierarchy.

3.1 Construction of normality

First, we discuss the ingredients to construct a real number x that is simply normal to a single base b. We employ these means later for several bases simultaneously, which by Corollary 2.3.3 yields an absolutely normal result.

We consider a sequence of b-adic intervals $I_1, I_2, I_3, \ldots$ constructed by recursion on i such that for each i, I_{i+1} is a subset of I_i, and such that $\lim_{i\to\infty} \mu(I_i) = 0$. The real number x determined by this sequence is the unique element of $\bigcap_{i\geqslant 1} I_i$, i.e. the limit of the left endpoints of these intervals. We let v_i be the word in base b such that I_i is the b-adic interval $[.v_i, .v_i + b^{-|v_i|})$. We let u_{i+1} be the word such that $v_i u_{i+1} = v_{i+1}$. Thus, for any $k \leqslant i$, $v_i = v_k u_{k+1} u_{k+1} \cdots u_i$ and x is equal to $.v_k u_{k+1} u_{k+1} u_{k+3} \cdots$.

We work towards ensuring that simple discrepancy decreases as we consider longer initial segments in the base-b expansion of x. We do so by choosing u_{i+1} so that $D(u_{i+1}, b)$ is smaller than a self-imposed threshold ε_{i+1}, where the function $i \mapsto \varepsilon_i$ is monotonically decreasing.

We need to determine the appropriate length of u_{i+1}. By allowing $|u_{i+1}|$ be sufficiently large, existence of a word u_{i+1} such that $D(u_{i+1}, b) < \varepsilon_{i+1}$ is ensured. By allowing $|u_{i+1}|$ be sufficiently small in comparison to $|v_i|$, it is ensured that for each ℓ less than or equal to $|u_{i+1}|$, $D(v_{i+1} \upharpoonright (|v_i| + \ell), b)$ is not much larger than $D(v_i, b)$, i.e. the variations of simple discrepancy within prefixes of u_{i+1} introduce only small variations of simple discrepancy within prefixes of v_{i+1}. Our task is to arrange for $\lim_{i\to\infty} \varepsilon_i = 0$ while maintaining the appropriate proportions in length between v_i and u_{i+1} to comply to the hypothesis of the following corollary of Lemma 2.4.1.

Corollary 3.1.1. *Take as given words v and $u_1, u_2, \ldots, u_m$ in base b. Suppose ε satisfies the following conditions.*

1. *$D(v, b) < \varepsilon$.*
2. *For each i, $D(u_i, b) < \varepsilon$.*
3. *For each i, $|u_i|/|v u_1 u_2 \cdots u_{i-1}| < \varepsilon$.*

Then for every ℓ less than or equal to $|u_1 u_2 \cdots u_m|$,

$$D\left((v u_1 u_2 \cdots u_m) \upharpoonright (|v| + \ell), b\right) < 2\varepsilon.$$

Proof. Let $\ell' = |v| + \ell$. Applying the hypotheses and Lemma 2.4.1, it can be seen that for i such that $|u_1 u_2 \cdots u_i| \leqslant \ell < |u_1 u_2 \cdots u_i u_{i+1}|$,

$$\begin{aligned} D((vu_1u_2\cdots u_m) \restriction \ell', b) &\leqslant D(v,b)\frac{|v|}{\ell'} + \left(\sum_{j=1}^{i} D(u_j,b)\frac{|u_j|}{\ell'}\right) + \frac{|u_{i+1}|}{\ell'} \\ &< \varepsilon\left(\frac{|v|}{\ell'} + \frac{\sum_{j=1}^{i}|u_j|}{\ell'}\right) + \varepsilon < 2\varepsilon. \end{aligned}$$

□

We turn to working simultaneously with bases $b \in \{2, \ldots, t\}$ in the context of stage i of a construction by recursion. Instead of one interval I_i, we work with a nested sequence of intervals, $I_{i,2} \supset I_{i,3} \supset \ldots I_{i,t}$, such that each $I_{i,b}$ is b-adic. The following lemma shows that the lengths of these intervals need not shrink too quickly.

Lemma 3.1.2. *For any non-empty interval I and any base b, there is a b-adic subinterval I_b such that $\mu(I_b) \geqslant \mu(I)/(2b)$. Moreover, such subinterval can be computed uniformly from I and b.*

Proof. Let m be least such that $1/b^m$ is less than $\mu(I)$, namely, $m = \lceil -\log_b(\mu(I))\rceil$. Note that $1/b^m$ is greater than or equal to $\mu(I)/b$, since $1/b^{m-1} \geqslant \mu(I)$. If there is a b-adic interval of length $1/b^m$ strictly contained in I, then let I_b be such an interval, and note that I_b has length greater than or equal to $\mu(I)/b$. Otherwise, there must be an a such that a/b^m is in I and neither $(a-1)/b^m$ nor $(a+1)/b^m$ belongs to I. Thus, $2/b^m$ is greater than $\mu(I)$. However, since $1/b^m < \mu(I)$ and b is greater than or equal to 2, $2/b^{m+1}$ is less than $\mu(I)$. So, at least one of the two intervals

$$\left[\frac{ba-1}{b^{m+1}}, \frac{ba}{b^{m+1}}\right) \quad \text{or} \quad \left[\frac{ba}{b^{m+1}}, \frac{ba+1}{b^{m+1}}\right)$$

must be contained in I. Let I_b be such. Then, the length of I_b is

$$\frac{1}{b^{m+1}} = \frac{1}{2b}\,\frac{2}{b^m} > \mu(I)/(2b).$$

In either case, the length of I_b is greater than $\mu(I)/(2b)$. It is easy to see that the steps described in the proof can be coded into an algorithm. In the upcoming Lemma 3.3.1 we show an efficient option for doing it. □

With the following definition and corollary, we specify the sequences of nested intervals we are interested in.

Definition. A *t-sequence* is a nested sequence of $t-1$ intervals, $\vec{I} = (I_2, \ldots, I_t)$, such that I_2 is dyadic and for each base b, I_{b+1} is a $(b+1)$-adic subinterval of I_b such that $\mu(I_{b+1}) \geqslant \mu(I_b)/2(b+1)$. We let $w_b(\vec{I})$ be the word in base b such that $.w_b(\vec{I})$ is the expansion of the left endpoint of I_b in base b.

Corollary 3.1.3. *For every non-empty dyadic interval I_2 and integer $t \geqslant 2$ there is a t-sequence $\vec{I}$ starting with I_2. Moreover, such t-sequence can be computed uniformly from I and t.*

Proof. Direct by iteratively applying Lemma 3.1.2. □

If $\vec{I} = (I_2, \ldots, I_t)$ is a t-sequence, then for any base $b \leqslant t$ and any real $x \in I_t$, x has $w_b(\vec{I})$ as an initial segment of its expansion in base b. If, further, $\vec{I'} = (I'_2, \ldots I'_{t'})$ is a t'-sequence with $t \leqslant t'$ such that $I'_2 \subset I_t$ and $x \in I'_{t'}$, then for each $b \leqslant t$, $\vec{I'}$ specifies how to extend $w_b(\vec{I})$ to a longer initial segment $w_b(\vec{I'})$ of the base b expansion of x. As opposed to arbitrary nested sequences, for t-sequences there is a function that gives a lower bound for the ratio between the measures of I_b and $I_{b'}$, for any two bases b and b' both less than or equal to t. That is, assuming $b > b'$, we can state the following inequality, which we use repeatedly in the sequel:

$$\mu(I_b) \geqslant \frac{\mu(I_{b'})}{2^{b-b'}\ b!/b'!}.$$

3.2 An algorithm with absolutely normal output

Our construction of the real x is by recursion and written in terms of two given functions, $i \mapsto t_i$ and $i \mapsto \varepsilon_i$. The first determines the number of bases to be considered at stage i and the second determines an upper bound on the allowed discrepancy in each of those bases. At stage $i+1$, we start with a t_i-sequence $\vec{I_i} = (I_{i,2}, I_{i,3}, \ldots, I_{i,t_i})$ given from the previous stage, with associated words $w_b(\vec{I_i})$, for $b \leqslant t_i$.

Definition. For $b \leqslant t_i$, let $v_{i+1,b}$ be $w_b(\vec{I}_{i+1})$, the base b representation of the left-endpoint of $I_{i+1,b}$, and let $u_{i+1,b}$ be $u_b(\vec{I}_{i+1})$, i.e. $v_{i+1,b} = v_{i,b}u_{i+1,b}$.

Algorithm 3.2.1. Assume given computable functions $i \mapsto t_i$ and $i \mapsto \varepsilon_i$ such that t_i and $1/\varepsilon_i$ are non-decreasing in i and unbounded, with $\varepsilon_i \leqslant 1/t_i$. Let δ_{i+1} be the following upper bound of the fraction of words in base b for $b \leqslant t_i$, of the length considered at stage $i+1$, that can be discarded,

$$\delta_{i+1} = \frac{1}{8\ t_i}\ \frac{1}{2^{t_i+t_{i+1}}t_i!t_{i+1}!}.$$

Let k_{i+1} be the length for the word in base t_i to be appended at stage $i+1$,

$$k_{i+1} = \max(\lceil 6/\varepsilon_{i+1}\rceil, \lceil -\ln(\delta_{i+1}/(2t_i))6/\varepsilon_{i+1}^2\rceil) + 1.$$

Initialization. Start with $\vec{I_0} = ((I_{0,2}))$, with $I_{0,2} = [0, 1)$.

Recursion step $i+1$. Determine the t_{i+1}-sequence $\vec{I}_{i+1}$ for stage $i+1$ as follows.

1. Let L be a dyadic subinterval of I_{i,t_i} such that $\mu(L) \geqslant \mu(I_{i,t_i})/4$.
2. For each dyadic subinterval J_2 of L of measure $\mu(J_2) = 2^{-\lceil \log t_i \rceil k_{i+1}}\mu(L)$, let $\vec{J} = (J_2, J_3, \ldots, J_{t_{i+1}})$ be a t_{i+1}-sequence for J_2.
3. Let $\vec{I}_{i+1}$ be the leftmost of the t_{i+1}-sequences $\vec{J}$ considered above such that for each $b \leqslant t_i$, $D(u_b(\vec{J}), b) \leqslant \varepsilon_{i+1}$.

We let x be the unique real in the intersection of the intervals in the sequences $\vec{I_i}$. Expressed in base b, $x = \lim_{i\to\infty} .v_{i,b}$, obtained as the concatenation of the words at each step i, $(x)_b = u_{1,b}\ u_{2,b}\ u_{3,b} \cdots$.

Theorem 3.2.2. *The output x of Algorithm 3.2.1 is well defined.*

Proof. To show that x is well-defined, we just need to verify that at each stage $i+1$ there is t_{i+1}-sequence $\vec{I}_{i+1}$. Computability of the first two steps of the algorithm is justified by Lemma 3.1.2 and Corollary 3.1.3, respectively. To prove that the third step always finds a suitable candidate, we compare the measures of two sets. Let S be the union of the set of intervals $J_{t_{i+1}}$ over the $2^{\lceil \log t_i \rceil k_{i+1}}$-many t_{i+1}-sequences $\vec{J} = (J_2, \ldots, J_{t_{i+1}})$. By Lemma 3.1.2 we have

$$\mu(L) \geqslant \mu(I_{i,t_i})/4.$$

Since $\vec{J}$ is a t_{i+1}-sequence, we also have

$$\mu\big(J_{t_{i+1}}\big) \geqslant \frac{1}{2^{t_{i+1}} t_{i+1}!} \mu(J_2)\,.$$

Observe that the possibilities for J_2 form a partition of L. Hence,

$$\mu(S) \geqslant \frac{1}{2^{t_{i+1}} t_{i+1}!} \mu(L)\,.$$

Combining inequalities,

$$\mu(S) \geqslant \frac{1}{2^{t_i} t_i!}\ \frac{1}{4}\ \frac{1}{2^{t_{i+1}} t_{i+1}!}\ \mu(I_{i,2})\,.$$

Let N be the subset of S defined as the union of the set of intervals $J_{t_{i+1}}$ which occur in t_{i+1}-sequences which are *not suitable.* A t_{i+1}-sequence $\vec{J}$ is *not suitable* if for some $b \leqslant t_i$, $D(u_b(\vec{J}), b) > \varepsilon_{i+1}$. By construction, $u_2(\vec{J})$ has length $\lceil \log t_i \rceil\, k_{i+1}$ and for each $b \leqslant t_i$, $u_b(\vec{J})$ has length greater than or equal to k_{i+1}. Each $\vec{J}$ considered at stage $i+1$ is such that for every $b \leqslant t_i$ each interval J_b is a subinterval of $I_{i,b}$. According to Lemma 2.4.4 and by the choice of k_{i+1}, for each $b \leqslant t_i$, the subset of $I_{i,b}$ consisting of reals with base b representations $.v_{i,b} u_b(\vec{J})$ for which $D(u_b(\vec{J}), b) > \varepsilon_{i+1}$ has measure less than $\delta_{i+1}\ \mu(I_{i,b})$, thus less than $\delta_{i+1}\ \mu(I_{i,2})$. Hence,

$$\mu(N) < t_i\ \delta_{i+1}\ \mu(I_{i,2})\,.$$

By the choice of δ_{i+1},

$$\mu(N) < \frac{1}{4\ 2^{t_i} t_i!\ 2^{t_{i+1}} t_{i+1}!}\ \mu(I_{i,2})\,.$$

Then, $\mu(N) < \mu(S)$. Since S is a superset of N, this proves that there is a suitable choice of $\vec{J}$ at the third step. □

We turn now to giving sufficient conditions on the functions $i \mapsto \varepsilon_i$ and $i \mapsto t_i$ to ensure the output x is absolutely normal. Since $(x)_b = u_{1,b}\, u_{2,b}\, u_{3,b} \cdots$ and $D(u_{i,b}, b)$ goes to zero as i goes to infinity, we are in a similar position as in the hypotheses of Lemma 2.5.3 but for the case of simple normality. Since we have additional hypotheses over the length of the $u_{i,b}$, we can give an explicit upper bound on the way the discrepancy approaches zero.

Theorem 3.2.3. *Suppose that the functions $i \mapsto t_i$ and $i \mapsto \varepsilon_i$ are monotonic and such that $\lim_{i\to\infty} t_i = \infty$ and $\lim_{i\to\infty} \varepsilon_i = 0$. Furthermore, suppose that for each i and for each $b \leqslant t_i$, $|u_{i+1,b}|/|v_{i,b}| < \varepsilon_{i+1}$. Then, the real x constructed in terms of these functions is absolutely normal.*

Proof. Let b be a base and let $\varepsilon > 0$. Choose an integer s so that b is less than t_s and $2\varepsilon_s$ is less than ε. During stages $i+1$ after s, we ensure of the constructed real x that the base b representation of x is obtained by appending to $v_{i,b}$ words $u_{i+1,b}$ for which $D(u_{i+1,b}, b) < \varepsilon_s$. Thus, $D(u_{s+1,b}\ u_{s+2,b} \cdots u_{n,b}, b) < \varepsilon_s$ for any n. Fix s_1 so that $|v_{s,b}|/(s_1 - |v_{s,b}|) < \varepsilon_s$. By noting that we add at least one new symbol in the base b representation of x during every stage after s, we have that $D(v_{s,b}\ u_{s+1,b}\ u_{s+2,b} \cdots u_{n,b}, b)$ is less than ε_s. Then, Corollary 3.1.1 applies to conclude that for every ℓ, $D((x)_b \restriction |v_{s,b}\ u_{s+1,b}\ u_{s+2,b} \cdots u_{s_1,b}| + \ell, b) < 2\varepsilon_s < \varepsilon$, which is sufficient to prove the theorem. □

3.3 Implementation and time complexity

We calculate the time complexity of the algorithm by counting the number of elementary operations required to output the first i symbols, where an elementary operation takes a fixed amount of time. We also count the number of mathematical operations performed by the algorithm, where mathematical operations include addition, subtraction, comparison, multiplication, division and logarithm.

Algorithm 3.2.1 depends on two given monotonic functions $i \mapsto t_i$ and $i \mapsto \varepsilon_i$. By controlling the rates at which t_i and ε_i approach their limits, we can control the number of operations required to run the construction. Thus, the count of the performed operations up to step i is given as a product of two factors, one that depends only on t_i and ε_i which can be made arbitrarily small, and the other that does not, which is the significant factor.

We say that a number is *small* if it can be bounded by a function of t_i and ε_{i+1}. By the virtue of the algorithm all values are polynomial in the inverse of the measure of the smallest interval I being considered, so they can be represented by $O(-\log \mu(I))$ binary digits. *Expensive* mathematical operations are multiplications and divisions having both operands non-small. *Non-expensive* mathematical operations are operations having at least one small operand and also all additions, subtractions and comparisons. Expensive operations require $O((-\log \mu(I))^2)$ elementary operations, while for the non-expensive $O(g(x)(-\log \mu(I)))$ elementary operations suffice, where g is some increasing function and x is small.

We represent b-adic intervals as tuples of four integers $\langle a, b, m, p\rangle$ such that the represented intervals are $[a/b^m, (a+1)/b^m)$ and $p = b^m$. The last terms p are kept just for efficiency of computation. For a b_1-adic interval $I_1 = \langle a_1, b_1, m_1, p_1\rangle$ and a b_2-adic interval $I_2 = \langle a_2, b_2, m_2, p_2\rangle$ we define $\mathrm{left}(I_1, I_2) = a_1\ p_2$ and $\mathrm{right}(I_1, I_2) = a_2\ p_1$.

The next lemma bounds the needed operations to find a b-adic subinterval of a given interval. It is intended that the given values be previously computed data; the proof revisits the existential result given in Lemma 3.1.2.

Lemma 3.3.1. *Suppose we are given two bases b_1 and b_2 and two b_1-adic intervals J_1 and I_1. We are also given a b_2-adic interval I_2 such that $J_1 \subseteq I_2 \subseteq I_1$, and the integers $\ell_I = \mathrm{left}(I_1, I_2)$ and $r_I = \mathrm{right}(I_1, I_2)$. Suppose we want to compute a b_2-adic subinterval J_2 of J_1 such that $\mu(J_2) \geqslant \mu(J_1)/(2b_2)$, and also compute the integers $\ell_J = \mathrm{left}(J_1, J_2)$ and $r_J = \mathrm{right}(J_1, J_2)$. The result can be obtained by two alternative computations, one takes $O((-\log \mu(J_1))^2)$ elementary operations; the other takes $O(g(b_1, b_2, -\log(\mu(J_1)/\mu(I_1)))(-\log \mu(J_1)))$ elementary operations, where g is some increasing function. In either case, $O(g(b_1, b_2, -\log(\mu(J_1)/\mu(I_1))))$ mathematical operations suffice.*

Proof. For $s = 1, 2$, let I_s and J_s be given by $\langle e_s, b_s, n_s, q_s\rangle$ and $\langle a_s, b_s, m_s, p_s\rangle$, respectively. Notice that $\mu(I_s) = 1/q_s = 1/b_s^{n_s}$ and $\mu(J_s) = 1/p_s = 1/b_s^{m_s}$. Within this proof, small values are those that can be bounded by $g(b_1, b_2, -\log(\mu(J_1)/\mu(I_1)))$. In particular, later in the proof it becomes clear that for each s, $m_s - n_s$ and $a_s - e_s$ are small.

First we give a computation that uses $O(g(b_1, b_2, -\log(\mu(J_1)/\mu(I_1))))$ mathematical operations, all of which are non-expensive. We start calculating the small values $b_s^{m_s-n_s}$. Using iterated squaring it takes $O(\log(m_s - n_s))$ multiplications, requiring $O(\log b_s^{2(m_s-n_s)}) = O((-\log(\mu(J_s)/\mu(I_s)))^2)$ elementary operations, in total. Notice that $\mu(J_2)/\mu(I_2) > \mu(J_1)/(2b_2\mu(I_1))$ and so

$$O(-\log(\mu(J_2)/\mu(I_2))) \subseteq O(-\log(\mu(J_1)/\mu(I_1))).$$

We need to find a_2, m_2, p_2, ℓ_J and r_J, such that:

$$\begin{array}{lrcll}
(1) & a_1/b_1^{m_1} & \leqslant & a_2/b_2^{m_2} & \text{and} \quad (a_2+1)/b_2^{m_2} \leqslant (a_1+1)/b_1^{m_1} \\
(2) & 1/b_1^{m_1} & \leqslant & 2b_2/b_2^{m_2} & \\
(3) & p_2 & = & b_2^{m_2} & \\
(4) & \ell_J = \text{left}(J_1,J_2) & = & a_1\ p_2 & \text{and} \quad r_J = \text{right}(J_1,J_2) = a_2\ p_1.
\end{array}$$

Since $J_2 \subseteq I_2$, $n_2 \leqslant m_2$. From $\mu(J_2) \geqslant \mu(J_1)/(2b_2)$ and $\mu(I_2) \leqslant \mu(I_1)$ we can conclude that $\mu(J_2) \geqslant (\mu(I_2)/(2b_2))(\mu(J_1)/\mu(I_1))$. Application of $-\log_{b_2}$ to both sides yields $m_2 \leqslant n_2+(m_1-n_1)\log_{b_2} b_1+2$. So there are at most $(m_1-n_1)\log_{b_2} b_1+2$ possible values for m_2, and we can iterate through each of them. From $J_2 \subseteq I_2$ we also infer that $e_2 b_2^{m_2-n_2} \leqslant a_2 \leqslant (e_2+1)b_2^{m_2-n_2}-1$, which means that there are $b_2^{m_2-n_2}$ possible values for a_2 and we can iterate through each of them. Since the number of iterations required to try the possibilities for both m_2 and a_2 are small numbers, they can be bounded by choosing g appropriately. To compute the starting and ending values in such iterations we only need a small number of non-expensive mathematical operations, and to change between consecutive values we need only addition. We then check for each pair if all requirements are met. Since Lemma 3.1.2 ensures that m_2 and a_2 exist, the described procedure eventually finds a suitable pair meeting the requirements.

For a given pair a_2 and m_2, we can compute p_2 by $p_2 = q_2 b_2^{m_2-n_2}$ with a single non-expensive mathematical operation. To calculate r_J first notice that

$$\begin{aligned}
r_J &= a_2 p_1 \\
&= a_2 b_1^{m_1} \\
&= (a_2-e_2)q_1 b_1^{m_1-n_1} + e_2 q_1 b_1^{m_1-n_1} \\
&= q_1(a_2-e_2)b_1^{m_1-n_1} + r_I b_1^{m_1-n_1}.
\end{aligned}$$

Since $a_s - e_s$ is small, r_J can be obtained from the last expression using only a constant number of non-expensive mathematical operations, because all factors are small except the first one of each term. The calculation of ℓ_J is similar. At this point, ℓ_J, r_J and p_2 meet the requirements by their construction. To check the requirements for a_2 and m_2, notice that requirement (1) is equivalent to $0 \leqslant \text{right}(J_1,J_2) - \text{left}(J_1,J_2) \leqslant p_2 - p_1$ and requirement (2) is equivalent to $p_2 \leqslant 2b_2 p_1$, and both can be checked with a constant number of non-expensive mathematical operations, given that we already calculated $\ell_J = \text{left}(J_1,J_2)$ and $r_J = \text{right}(J_1,J_2)$.

An alternative way of computing can be achieved by replacing the iteration through possible values of a_2 and m_2 by their direct computation using the given bounds and rounding. This entails a constant number of expensive mathematical operations. □

The next lemma counts the steps in one complete stage of our algorithm. As in the previous lemma, it is intended that the given values be previously computed data. We count all operations except the computation of t_{i+1}, ε_{i+1} and k_{i+1}, which is postponed until the subsequent theorem.

Lemma 3.3.2. *Assume we are given i, t_i, ε_{i+1} and t_{i+1}. Then, there is computable function $h(t,\varepsilon)$, increasing in t and $1/\varepsilon$, such that stage $i+1$ of Algorithm 3.2.1 can be completed in $O(h(t_{i+1},\varepsilon_{i+1}))$ mathematical operations. Let n be the minimum number of digits that are sufficient to represent each of the endpoints of the intervals of $\vec{I}_i$. In case $t_{i+1} = t_{i-1}$ stage $i+1$ requires $O(h(t_{i+1},\varepsilon_{i+1})\ n)$ elementary operations; otherwise it requires $O(h(t_{i+1},\varepsilon_{i+1})\ n^2)$ elementary operations.*

Proof. We count the operations needed to run all the steps of stage $i+1$. Assume first that $t_{i+1} = t_{i-1}$. Then, all bases considered in stage $i+1$ were also considered in stages i and $i-1$. Lemma 3.3.1 applies to count the operations needed to find subintervals. In each application of the lemma, the values of I_1, I_2, ℓ_I and r_I in the hypothesis are carried forward from the computation in the previous stage. Then, $-\log(\mu(J_1)\,/\mu(I_1))$ is bounded by $\lceil \log t_i \rceil\, k_{i+1}$ and hence is a small value. Since $O(-\log \mu(J_1)) = O(n)$, finding a subinterval requires at most $O(g(t_{i+1}, \varepsilon_{i+1}))$ mathematical operations or $O(g(t_{i+1}, \varepsilon_i)\; n)$ elementary operations.

We write h with a subindex to indicate a function of t_{i+1} and ε_{i+1}. Let h_o be such that each non-expensive mathematical operation in this procedure uses at most $O(h_o)$ elementary operations and each expensive mathematical operation uses at most $O(h_o\; n)$. The computation can be organized in the following steps.

- Compute $\lceil \log(t_i) \rceil$, δ_{i+1} and k_{i+1}. This takes a constant number of non-expensive mathematical operations or $O(h_o)$ elementary operations.

- Compute a dyadic subinterval L of I_{i,t_i} such that $\mu(L) \geqslant \mu(I_{i,t_i})/4$. This takes $O(g(t_{i+1}, \varepsilon_{i+1}))$ mathematical operations or $O(g(t_{i+1}, \varepsilon_{i+1})\; n)$ elementary operations.

- In increasing order of left endpoint, consider the dyadic subintervals J_2 of L:

1. For each possible J_2, we determine a t_{i+1}-sequence $\vec{J}$ starting with J_2. This takes $O(t_{i+1}\;\, g(t_i, \varepsilon_{i+1}))$ mathematical operations or $O(h_1\;\, g(t_i, \varepsilon_{i+1})\;\, n)$ elementary operations.

2. For each $b \leqslant t_i$, compute the base-b representation of the left endpoint $u_b(\vec{J})$ of $\vec{J}_b$. This requires $O(|u_b(\vec{J})|)$ non-expensive mathematical operations, because each operation has at least one operand is a base and hence depends only on t_i, and $|u_b(\vec{J})| \leqslant \lceil \log t_i \rceil\, k_{i+1}$ also depends only on t_i. Therefore, this takes $O(h_2)$ mathematical operations or $O(h_2\; h_o)$ elementary operations.

3. Calculate thresholds for the number of occurrences of each symbol to check $D(u_b(\vec{J}), b) \leqslant \varepsilon_{i+1}$. Such thresholds are of the form $(1/b + \varepsilon_{i+1})|u_b(\vec{J})|$ and therefore can be calculated with a constant number of non-expensive mathematical operations for each base. Hence, $O(t_i) = O(h_3)$ non-expensive mathematical operations or $O(h_3\; h_o)$ elementary operations in total.

4. The counting of occurrences and the comparison against the threshold operate only on small values (bounded by $\max(t_i, |u_b(\vec{J})|)$). This step takes $O(|u_b(\vec{J})|\;\; \max(t_i, |u_b(\vec{J})|))$ non-expensive mathematical operations for each base. In total, this is
$$O(t_i\; |u_b(\vec{J})|\;\; \max(t_i, |u_b(\vec{J})|)) = O(h_4)$$
mathematical operations or $O(h_4\; h_o)$ elementary operations.

The search stops upon finding a suitable t_{i+1}-sequence, before exhausting $2^{\lceil \log t_i \rceil k_{i+1}}$ many intervals J_2. This requires at most $h_* = 2^{\lceil \log t_i \rceil k_{i+1}}$ iterations. We can complete the proof for the case $t_{i+1} = t_{i-1}$ by setting $h = h_*\; (h_1\; g + h_2 + h_3 + h_4)\; h_o$.

If $t_{i+1} > t_{i-1}$, then it is possible that for some uses of Lemma 3.3.1 we do not have any previously computed data. In this case we set the intervals in the hypothesis of the lemma as $I_1 = I_2 = [0, 1)$ and $\ell_I = r_I = 0$, making
$$-\log(\mu(J_s)\,/\mu(I_s)) = -\log(\mu(J_s)) = O(n)$$
for $s = 1, 2$, and thus requiring $O(g\; n^2)$ elementary operations for each application of the alternative computation in Lemma 3.3.1. This requires $O(h_*\; h_1\; g\; n^2)$ elementary

operations more than in the previous case. Using the same h as before, this case entail at most $O(h\, n^2)$ elementary operations. □

Theorem 3.3.3. *Suppose f is a computable non-decreasing unbounded function. Algorithm 3.2.1 computes an absolutely normal number x such that, for any base b, it outputs the first i symbols in the base b representation of x after performing $O(f(i)\ i)$ mathematical operations or $O(f(i)\ i^2)$ elementary operations.*

Proof. We define functions $i \mapsto t_i$ and $i \mapsto \varepsilon_i$ simultaneously with running an implementation of Algorithm 3.2.1. Let $t_1 = 2$ and $\varepsilon_1 = 1/2$ Assume $k_1 = 1$ and $f(1)$ is known data, having a value greater than $h(2, 1)$, for the h as in Lemma 3.3.2 For the recursion stage $i + 1$, assume that $t_i = z$ and $\varepsilon_i = 1/z$ are given, with $z \geqslant 2$, and that $\vec{I}_i$ is the result of the construction as determined by the first i many values of t and ε with associated words $v_{i,b}$. If the number of stage $i + 1$ is a power of 2, we execute i many elementary operations in the computation of the initial values of f, obtaining the values of f on the numbers less than or equal to some integer m. Notice that $1 \leqslant m \leqslant i$. Define δ by

$$\delta = \frac{1}{8\ t_i}\ \frac{1}{2^{t_i+z+1}\ t_i!(z+1)!},$$

which would be the value of δ_{i+1} if we were to define $t_{i+1} = z+1$. Let $k(\varepsilon, \delta, t)$ be the function defined by the calculation of k as given in Lemma 2.4.4. Finally, we execute i many elementary operations in the computations of the functions $k(1/(z+1), \delta, z+1)$ and $h(z + 1, 1/(z + 1))$. If we obtain values for these functions within the allotted number of operations and they satisfy the inequalities $h(z + 1, 1/(z + 1)) < f(m)$ and, for each $b \leqslant t_i$,

$$\frac{\lceil \log(z+1) \rceil\, k(1/(z+1), \delta, z+1) + \lceil -\log(\delta) \rceil}{|v_{i,b}|} < \frac{1}{z+1},$$

then define $t_{i+1} = z + 1$ and $\varepsilon_{i+1} = 1/(z + 1)$. Otherwise, let $t_{i+1} = t_i = z$ and $\varepsilon_{i+1} = \varepsilon_i = 1/z$. We then complete stage $i + 1$ of the construction and thereby complete the recursion step in the definitions of the functions t and ε. Clearly, $i \mapsto t_i$ and $i \mapsto \varepsilon_i$ are computable, $i \mapsto t_i$ is non-decreasing, and $i \mapsto \varepsilon_i$ is non-increasing. Applying the assumptions on f, $\lim_{i\to\infty} t_{i+1} = \infty$ and $\lim_{i\to\infty} \varepsilon_{i+1} = 0$. Further, in the construction determined by these functions, if during stage $j + 1$ the value of ε_{j+1} is lowered from $1/(z - 1)$ to $1/z$, then for each $b \leqslant t_j$,

$$\frac{\lceil \log(z+1) \rceil\, k(1/z, \delta_{j+1}, z+1) + \lceil -\log(\delta_{j+1}) \rceil}{|v_{j,b}|} < \frac{1}{z}.$$

For every subsequent stage $i + 1$ during which $\varepsilon_{i+1} = 1/z$ and for every $b \leqslant z + 1$,

$$|u_{i+1,b}| \leqslant \lceil \log(z+1) \rceil\, k(1/(z+1), \delta_{j+1}, z+1) + \lceil -\log(\delta_{j+1}) \rceil$$

and $|v_{i+1,b}| \geqslant |v_{j,b}|$, so $|u_{i+1,b}|/|v_{i,b}|$ is less than or equal to $1/(z + 1)$. Thus, the construction satisfies the hypotheses of Theorem 3.2.3 and thereby produces an absolutely normal number.

All mentioned mathematical operations are non-expensive, because the only non-small operands in them are of the form $|v_{i,b}|$ and all those appear on independent calculations. The computations of the values of t and ε during stage $i + 1$ add only $O(i)$ elementary operations to the construction itself. Since that calculation is only done when the stage number is a power of two, in total this adds $O(i)$ extra elementary operations. Since f is non-decreasing, for every i, if $t_{i+1} = z + 1$

then $h(z+1, 1/(z+1)) < f(i+1)$. From the way t_i is defined, $t_{i+1} > t_{i-1}$ in at most $O(\log i)$ stages; therefore, using Lemma 3.3.2 the total number of required elementary operations is $O(i\ f(i+1)\ i) + \sum_{j=1}^{\log i} f(2^j+1)\ 2^{j^2}) = O(f(i+1)\ i^2)$. Since each stage produces at least one extra symbol in every base considered, i stages are enough to produce the first i symbols in any of those bases. □

3.4 The Borel hierarchy

Recall that the Borel hierarchy for subsets of the real numbers is the stratification of the σ-algebra generated by the open sets with the usual interval topology. For references see Kechris' textbook [Kec95] or Marker's lecture notes [Mar02].

A set A is $\boldsymbol{\Sigma}^0_1$ if and only if A is open and A is $\boldsymbol{\Pi}^0_1$ if and only if A is closed. A is $\boldsymbol{\Sigma}^0_{n+1}$ if and only if it is a countable union of $\boldsymbol{\Pi}^0_n$ sets, and A is $\boldsymbol{\Pi}^0_{n+1}$ if and only if it is a countable intersection of $\boldsymbol{\Sigma}^0_n$ sets.

A is hard for a Borel class if and only if every set in the class is reducible to A by a continuous map. A is complete in a class if it is hard for this class and belongs to the class.

When we restrict to intervals with rational endpoints and computable countable unions and intersections, we obtain the effective or lightface Borel hierarchy. One way to present the finite levels of the effective Borel hierarchy is by means of the arithmetical hierarchy of formulas in the language of second-order arithmetic. Atomic formulas in this language assert algebraic identities between integers or membership of real numbers in intervals with rational endpoints. A formula in the arithmetic hierarchy involves only quantification over integers. A formula is Π^0_0 and Σ^0_0 if all its quantifiers are bounded. It is Σ^0_{n+1} if it has the form $\exists x\, \theta$ where θ is Π^0_n, and it is Π^0_{n+1} if it has the form $\forall x\, \theta$ where θ is Σ^0_n.

A set A of real numbers is Σ^0_n (respectively Π^0_n) in the effective Borel hierarchy if and only if membership in that set is definable by a formula which is Σ^0_n (respectively Π^0_n). Notice that every Σ^0_n set is $\boldsymbol{\Sigma}^0_n$ and every Π^0_n set is $\boldsymbol{\Pi}^0_n$. In fact for every set A in $\boldsymbol{\Sigma}^0_n$ there is a Σ^0_n formula and a real parameter such that membership in A is defined by that Σ^0_n formula relative to that real parameter.

A is hard for an effective Borel class if and only if every set in the class is reducible to A by a computable map. As before, A is complete in an effective class if it is hard for this class and belongs to the class. Since computable maps are continuous, proofs of hardness in the effective hierarchy often yield proofs of hardness in general by relativization. This is the case in our work.

3.4.1 The set of absolutely normal numbers

By the form of its definition, normality to a fixed base is explicitly a Π^0_3 property of real numbers. The same holds for absolute normality. Absolute abnormality is, for all bases, the negation of normality, hence a Π^0_4 property.

Lemma 3.4.1 (As in [Mar02]). *The set of real numbers that are normal to a given base is* Π^0_3.
The set of real numbers that are absolutely normal is Π^0_3.
The set of real numbers that are absolutely abnormal is Π^0_4.

Thus, to prove completeness of the set of absolutely normal real numbers for the class Π^0_3 we need only prove hardness. We prove our hardness result for the Borel hierarchy by relativizing a hardness result for Π^0_3 subsets of the natural numbers. Let $\mathcal{L}$ be the language of first order arithmetic. As usual, a sentence is a formula without free variables.

We define an algorithm that maps Π^0_3 sentences in the language of first order arithmetic $\mathcal{L}$ to infinite words of zeros and ones. If the given sentence is true then the corresponding binary sequence is the expansion in base two of an absolutely normal number. Otherwise, the corresponding binary sequence is the expansion in base two of an absolutely abnormal number. We define this algorithm as a composition of two algorithms. We use Baire space $\mathbb{N}^\mathbb{N}$ as an intermediate working space. The first algorithm maps sentences in $\mathcal{L}$ to infinite sequences of positive integers that reflect the truth or falsity of the given sentences. The second algorithm maps these sequences of elements in $\mathbb{N}^\mathbb{N}$ to binary infinite words with the appropriate condition on normality.

In the proof of Theorem 3.7.1, where we prove the actual completeness result, we consider a computable reduction (from positive integers to positive integers) defined from the algorithm just described.

Recall that a Π^0_3 formula in first order arithmetic is equivalent to one starting with a universal quantifier $\forall$, followed by the quantifier "there are only finitely many" $\exists^{<\infty}$ and ended by a computable predicate, see Theorem XVII and Exercise 14-27 in [Rog87]. The computable predicate in this equivalent form comes from the Σ^0_0 subformula of the original.

3.5 An algorithm from sentences to infinite sequences

We define a computable function that maps a sentence in $\mathcal{L}$ to an infinite sequence of positive integers that reflect the truth or falsity of the input sentence.

Algorithm 3.5.1. Let $\forall i \exists^{<\infty} j\ C(i,j)$ be the input Π^0_3-sentence, with C a computable predicate.

For every positive integer n in increasing order, let $i = \max\{k \in \mathbb{N} : 2^k \text{ divides } n\}$ and let $j = n/2^i$. If $j = 1$ or $C(i,j)$ then append $i, i+1, \ldots, i+j-1$ to the output sequence.

Lemma 3.5.2. *If φ is a Π^0_3 sentence in $\mathcal{L}$ then Algorithm 3.5.1 outputs an infinite sequence f of integers such that the subsequence of f's first occurrences is an enumeration of $\mathbb{N}$ in increasing order and the following dichotomy holds:*

If φ is true then no positive integer occurs infinitely often in f.

If φ is false then all but finitely many integers occur infinitely often in f.

Proof. Assume φ of the form $\forall i \exists^{<\infty} j\ C(i,j)$, where C is computable. We say that a tuple $\langle i,j\rangle$ is *appending* if $j = 1$ or $C(i,j)$. It is clear by inspection that all possible pairs $\langle i,j\rangle$ with $i,j \in \mathbb{N}$ are processed, and that $\langle i+1,j\rangle$ and $\langle i,j+1\rangle$ are always processed after $\langle i,j\rangle$.

Thus, the first occurrence of an integer $i+1$ in f is due to the appending tuple $\langle i+1,1\rangle$ or an appending tuple $\langle i',j\rangle$ with $i' < i+1$ and $i'+j > i+1$. In the first case, the appending tuple $\langle i+1,1\rangle$ is processed after $\langle i,1\rangle$, which appends i to the output. In the second case, the processing of an appending tuple $\langle i',j\rangle$ with $i' < i+1$ and $i'+j > i+1$ appends i right before $i+1$ to the output. Thus, $i+1$ occurs for the first time in f after i.

Suppose now φ is true. For any i, there are finitely many appending tuples of the form $\langle i',j\rangle$ with $i' \leqslant i$. After all such appending tuples have been processed, i will not be appended to the output sequence. Thus no positive integer can occur infinitely often in f.

Now suppose φ is false. Let i be such that there are infinitely many j such that $C(i,j)$. Let k be any positive integer. Each time an appending tuple of the form

$\langle i, j\rangle$ with $k < j$ is processed, $i + k$ is appended to the output. Since we assumed there are infinitely many such tuples, $i + k$ is appended to the output an infinite number of times. Thus, all integers greater than i occur infinitely often in the output sequence. $\square$

3.6 An algorithm with absolutely normal or abnormal output

We give an algorithm that maps a sequence of infinite integers to an infinite binary word representing an absolutely normal or absolutely abnormal number.

We use the ingredients of Algorithm 3.2.1, but in a different way. We start defining a function that, for an integer i and a t-sequence $\vec{I}$ (for some t), constructs an $(i + 1)$-sequence inside $\vec{I}$ with good properties. The method is to determine the expansion of the rational endpoints of each b-adic interval in the $(i + 1)$-sequence. Since the respective b-adic intervals are nested, the determination of the expansions is done by adding suffixes.

We introduce three functions of i, $(\delta_i)_{i\geqslant1}$, $(k_i)_{i\geqslant1}$ and $(\ell_i)_{i\geqslant1}$, that act as parameters for the construction of an $(i+1)$-sequence. The integer k_i indicates how many digits in base $(i + 1)$ can be determined in each step; thus, $k_i\lceil\log(i + 1)\rceil$ indicates how many digits in base 2 can be determined in each step (keep in mind that, in general, more digits are needed to ensure the same precision in base 2 than in a larger base). The integer ℓ_i limits how many digits in base 2 can *at most* be determined in each step. And the rational δ_i bounds the relative measure of any two intervals in two consecutive nested $(i+1)$-sequences. We call REF_i the function that given $\vec{I}$ constructs an $(i + 1)$-sequence by recursion. It is a search through nested $(i + 1)$-sequences until one with good properties is reached. The choice we make for $(\delta_i)_{i\geqslant1}$, $(k_i)_{i\geqslant1}$ and $(\ell_i)_{i\geqslant1}$, allow us to prove the correctness of the construction.

Definition. Let $(k)_{i\geqslant1}$ and $(\ell)_{i\geqslant1}$ be the computable sequences of positive integers and let $(\delta)_{i\geqslant1}$ be the computable sequence of positive rational numbers less than 1 such that, for each $i \geqslant 1$,

$$\delta_i = \frac{1}{2^{2i-2}\ (i+1)!^2}$$

$$k_i = \text{least integer greater than} \max\left(\lceil 6(i+2)\rceil, -\ln\left(\frac{\delta_i}{2(i+1)^2}\right)6(i+2)^2\right)$$

$$\ell_i = k_i\lceil\log(i+1)\rceil + \lceil-\log(\delta_i)\rceil.$$

Algorithm 3.6.1. The algorithm REF_i maps a $(p + 1)$-sequence $\vec{I} = (I_2, \ldots, I_{p+1})$ into an $(i + 1)$-sequence $\text{REF}_i(\vec{I})$, that we define recursively.

Initial step 0. Let $\vec{I_0} = (I_{0,2}, \ldots, I_{0,i+1})$ be an $(i + 1)$-sequence where $I_{0,2}$ is the leftmost dyadic subinterval of I_{p+1} such that $\mu(I_{0,2}) \geqslant \mu(I_{p+1})/4$.

Recursive step $j + 1$. Let $\vec{I}_{j+1}$ be the $(i + 1)$-sequence such that

- Let L be the leftmost dyadic subinterval of $I_{j,i+1}$ such that $\mu(L) \geqslant \mu(I_{j,i+1})/4$.
- Partition L into $k_i\lceil\log(i + 1)\rceil$ many dyadic subintervals of equal measure $2^{-k_i\lceil\log(i+1)\rceil}\mu(L)$. For each such subinterval J_2 of L, define the $(i+1)$-sequence $\vec{J} = (J_2, J_3, \ldots, J_{i+1})$.
- Let $\vec{I}_{j+1}$ be the leftmost of the $(i + 1)$-sequences $\vec{J}$ considered above such that for each base $b \leqslant i + 1$, $D(u_b(\vec{J}), b) \leqslant 1/(i + 2)$, where $u_b(\vec{J})$ is such that $w_b(\vec{I_j})u_b(\vec{J}) = w_b(\vec{J})$.

Repeat the recursive step until step n when all the following hold for every base $b \leqslant i+1$:

$$\begin{array}{lll} a. & |w_b(\vec{I_n})| & > & \ell_{i+1}(i+3) \\ b. & D(w_b(\vec{I_n}), b) & \leqslant & 2/(i+2) \\ c. & D_{2\ell_i}(w_b(\vec{I_n}), b) & > & b^{-2\ell_i-1}. \end{array}$$

Finally, let $\text{R}\textsc{ef}_i(\vec{I}) = \vec{I_n}$.

The following lemmas show that for every positive integer i, the function $\text{R}\textsc{ef}_i$ is well defined and it is computable.

Lemma 3.6.2. *There is always a suitable (i+1)-sequence $\vec{J}$ to be selected in the recursive step of Algorithm 3.6.1.*

Proof. Consider the recursive step $j+1$ of the algorithm. Let S be the union of the set of intervals J_{i+1} over the $2^{k_i \lceil \log(i+1) \rceil}$ many $(i+1)$-sequences $\vec{J}$. We have

$$\mu(L) \geqslant \mu(I_{j,i+1})/4$$

and, since $\vec{J}$ and $\vec{I_j}$ are $(i+1)$-sequences, we have

$$\mu(J_{i+1}) \geqslant \frac{\mu(J_2)}{2^{i-2}(i+1)!} \quad \text{and} \quad \mu(I_{j,i+1}) \geqslant \frac{\mu(I_{j,2})}{2^{i-2}(i+1)!}.$$

Since the possibilities for J_2 form a partition of L,

$$\mu(S) \geqslant \frac{\mu(L)}{2^{i-2}(i+1)!} \geqslant \frac{\mu(I_{j,i+1})}{2^{i}(i+1)!} \geqslant \frac{\mu(I_{j,2})}{2^{2i-2}(i+1)!^2} = \delta_i \mu(I_{j,2}).$$

Let us say that an $(i+1)$-sequence $\vec{J}$ is *not suitable* if for some base $b \leqslant i+1$,

$$D(u_b(\vec{J}), b) > 1/(i+2).$$

Let N be the subset of S defined as the union of the set of intervals J_{i+1} which occur in $(i+1)$-sequences which are *not suitable.* Each $\vec{J}$ considered at stage $i+1$ is such that for every base $b \leqslant i+1$ each interval J_b is a subinterval of $I_{j,b}$. By definition, $|u_b(\vec{J})| > k_i$ for each b and $\vec{J}$. By Lemma 2.4.4 with $t = i+1$, $\varepsilon = 1/(i+2)$, $\delta = \delta_i/(i+1)$ and $k = k_i$, for each base $b \leqslant i+1$, the subset of $I_{j,b}$ consisting of reals with base b expansions starting with $w_b(\vec{I_j})u_b(\vec{J})$ for which $D(u_b(\vec{J}), b) > 1/(i+2)$ has measure less than $\delta\mu(I_{j,b})$, and hence, less than $\delta\mu(I_{j,2})$. Therefore,

$$\mu(N) < (i+1)\delta\mu(I_{j,2}) = \delta_i\mu(I_{j,2}) \leqslant \mu(S).$$

This proves that S is a proper superset of N, therefore, there is always a suitable $(i+1)$-sequence. □

Lemma 3.6.3. *The recursion in Algorithm 3.6.1 finishes for every input sequence $\vec{I}$.*

Proof. Using Lemma 3.1.2 or Corollary 3.1.3 the needed b-adic subintervals with the appropriate measure and $(i+1)$-sequences can be found computably. Lemma 3.6.2 ensures that a suitable $\vec{J}$ can always be found in each recursive step. All the other tasks in the recursive step are clearly computable. It remains to check that the ending conditions of the recursion are eventually met.

Let $u_{b,j+1}$ and v_b be such that $w_b(\vec{I_j})u_{b,j+1} = w_b(\vec{I}_{j+1})$ and $w_b(\vec{I_0}) = w_b(\vec{I})v_b$. Then, $1 \leqslant |u_{b,j}|$ and

$$\begin{aligned}
|u_{b,j}| &= |w_b(\vec{I_j})| - |w_b(\vec{I}_{j-1})| \\
&= -\log_b \frac{\mu(I_{j,b})}{\mu(I_{j-1,b})} \\
&= -\log_b \frac{\mu(I_{j,b})}{\mu(I_{j,2})} \frac{\mu(I_{j,2})}{\mu(I_{j-1,i+1})} \frac{\mu(I_{j-1,i+1})}{\mu(I_{j-1,b})} \\
&\leqslant -\log_b \frac{1}{2^{b-3}b!} \frac{1}{4\ 2^{k_i\lceil \log(i+1)\rceil}} \frac{1}{2^{i+1-b}(i+1)!/b!} \\
&\leqslant -\log \frac{1}{2^{i-2}(i+1)!} \frac{1}{4\ 2^{k_i\lceil \log(i+1)\rceil}} \\
&\leqslant k_i \lceil \log(i+1)\rceil - \log \delta_i \\
&\leqslant \ell_i.
\end{aligned}$$

For any k, $w_b(\vec{I_k}) = w_b(\vec{I})v_b\ u_{b,1}\ u_{b,2} \cdots u_{b,k}$. Also, the recursive step establishes $D(u_{b,j}, b) \leqslant 1/(i+2)$. Notice that, in each of the three conditions $(a), (b)$ and (c), the right side of the inequality is fixed. For condition (a), $|w_b(\vec{I_n})|$ is thus equal to $|w_b(\vec{I})v_b\ u_{b,1}\ u_{b,2} \cdots u_{b,n}|$, so it is strictly increasing on n, which means it is greater than the required lower bound for sufficiently large n. For condition (b), observe that

$$\begin{aligned}
D(w_b(\vec{I_n}), b) &= D(w_b(\vec{I})v_b\ u_{b,1}\ u_{b,2} \cdots u_{b,n}, b) \\
&\leqslant |w_b(\vec{I})v_b|/|w_b(\vec{I_n})| + D(u_{b,1}\ u_{b,2} \cdots u_{b,n}, b) \\
&\leqslant |w_b(\vec{I})v_b|/|w_b(\vec{I_n})| + 1/(i+2).
\end{aligned}$$

In the right hand side, the first term approaches 0 for large n, so the entire expression is less than $2/(i+2)$ for sufficiently large n. For condition (c), observe that the recursive step ensures that $u_{b,j}$ is never all zeros. So, a sequence of $2\ell_i$ zeros does not occur in $u_{b,1}\ u_{b,2} \cdots u_{b,n}$. By definition,

$$D_{2\ell_i}(w_b(\vec{I})v_b\ u_{b,1}\ u_{b,2} \cdots u_{b,n}, b) \geqslant \left| \frac{\mathrm{occ}(w_b(\vec{I})v_b\ u_{b,1}\ u_{b,2} \cdots u_{b,n}, 0^{2\ell_i})}{|w_b(\vec{I})v_b\ u_{b,1}\ u_{b,2} \cdots u_{b,n}|} - \frac{1}{b^{2\ell_i}} \right|.$$

Since $\mathrm{occ}(w_b(\vec{I})v_b\ u_{b,1}\ u_{b,2} \cdots u_{b,n}, 0^{2\ell_i})$ is bounded by a constant, for sufficiently large n, the discrepancy $D_{2\ell_i}(w_b(\vec{I})v_b\ u_{b,1}\ u_{b,2} \cdots u_{b,n}, b)$ is arbitrarily close to $b^{-2\ell_i}$. □

Lemma 3.6.4. *Let $\vec{I}$ be an arbitrary $(p+1)$-sequence, $i \geqslant 1$ be an integer and $\vec{R}$ be* REF$_i(\vec{I})$. *For every base $b \leqslant \min(i,p)+1$,*

1. $R_2 \subseteq I_{p+1}$
2. $D(w_b(\vec{R}), b) \leqslant 2/(i+2)$
3. $D_{2\ell_i}(w_b(\vec{R}), b) > b^{-2\ell_i-1}$
4. $|w_b(\vec{R})| > \ell_{i+1}(i+3)$
5. *For each ℓ such that $|w_b(\vec{I})| \leqslant \ell \leqslant |w_b(\vec{R})|$,*

$$D(w_b(\vec{R}) \upharpoonright \ell, b) \leqslant D(w_b(\vec{I}), b) + \lceil -\log(\delta_p)\rceil/|w_b(\vec{I})| + 1/(i+2) + \ell_i/|w_b(\vec{I})|.$$

Proof. Let $u_{b,j+1}$ and v_b be such that $w_b(\vec{I_j})u_{b,j+1} = w_b(\vec{I}_{j+1})$ and $w_b(\vec{I_0}) = w_b(\vec{I})v_b$, as in the proof of Lemma 3.6.3. Then, $1 \leqslant |u_{b,j}| \leqslant \ell_i$, $D(u_{b,j}, b) \leqslant 1/(i+2)$, and for any k it holds that $w_b(\vec{I_k}) = w_b(\vec{I})v_b\ u_{b,1}\ u_{b,2} \cdots u_{b,k}$.

Fix a base b. Point (1) follows by induction in the recursive steps in the definition of $\text{REF}_i(\vec{I})$, since each subsequent interval is contained in the previous one. Points (2), (3) and (4) follow from the termination condition in that recursion. For point (5), use the above definition of v_b and the parameter δ_p.

$$\begin{aligned}
|v_b| &= |w_b(\vec{I_0})| - |w_b(\vec{I})| \\
&= -\log_b \frac{\mu(I_{0,b})}{\mu(I_b)} \\
&= -\log_b \frac{\mu(I_{0,b})}{\mu(I_{0,2})} \frac{\mu(I_{0,2})}{\mu(I_{p+1})} \frac{\mu(I_{p+1})}{\mu(I_b)} \\
&= -\log_b \frac{1}{2^{b-3}b!} \frac{1}{4} \frac{1}{2^{p+1-b}(p+1)!/b!} \\
&= -\log \frac{1}{2^p(p+1)!} \\
&\leqslant -\log \delta_p \\
&\leqslant \lceil -\log(\delta_p) \rceil .
\end{aligned}$$

Then, for each m, $D(u_{b,1}\ u_{b,2} \cdots u_{b,m}, b) \leqslant 1/(i+2)$ and

$$D(w_b(\vec{I})v_b\ u_{b,1}\ u_{b,2} \cdots u_{b,m}, b) \ \leqslant\ D(w_b(\vec{I}), b) + \lceil -\log(\delta_p) \rceil / |w_b(\vec{I})| + 1/(i+2).$$

Finally, for fixed ℓ, let m and ℓ' be such that

$$(w_b(\vec{I})v_b\ u_{b,1}\ u_{b,2} \cdots u_{b,m})(u_{b,m+1} \restriction \ell') = w_b(\vec{R}) \restriction \ell.$$

Then,

$$\begin{aligned}
D(w_b(\vec{R}) \restriction \ell, b) &= D((w_b(\vec{I})v_b\ u_{b,1}\ u_{b,2} \cdots u_{b,m})(u_{b,m+1} \restriction \ell'), b) \\
&\leqslant D(w_b(\vec{I}), b) + \frac{\lceil -\log(\delta_p) \rceil}{|w_b(\vec{I})|} + \frac{1}{i+2} + \frac{|u_{b,m+1}|}{|w_b(\vec{I})|} \\
&\leqslant D(w_b(\vec{I}), b) + \frac{\lceil -\log(\delta_p) \rceil}{|w_b(\vec{I})|} + \frac{1}{i+2} + \frac{\ell_i}{|w_b(\vec{I})|}.
\end{aligned}$$

□

Algorithm 3.6.5. We define the algorithm LIMREF that takes infinite sequences of positive integers f to infinite binary words. As a notation abuse, we write $\text{LIMREF}(f)$ to mean also the real number represented by the infinite binary word.

$$\text{LIMREF}(f) \text{ is the unique element in } \bigcap_{j=1}^{\infty} (\vec{R_j})_2,$$

$$\text{where } \vec{R_0} = ([0,1)) \text{ and } \vec{R}_{j+1} = \text{REF}_{f_{j+1}}(\vec{R_j}).$$

That is, $\text{LIMREF}(f)$ is the real obtained by iterating applications of REF_i where i is determined by the positive integers in f.

Lemma 3.6.6. *Algorithm* LIMREF *is uniformly computable from its input f.*

Proof. By point (1) of Lemma 3.6.4, for each $j \geqslant 1$, $\textsc{LimRef}(f)$ is inside every interval in every (f_j+1)-sequence $\vec{R}_j$, and therefore, $w_2(\vec{R}_j)$ is a prefix of $(\textsc{LimRef}(f))_2$. By Lemma 3.6.3, each application of $\textsc{Ref}$ is computable and the necessary iteration is also clearly computable. □

Lemma 3.6.7. *Let f be a sequence of positive integers such that the subsequence of f's first occurrences is an enumeration of $\mathbb{N}$ in increasing order and no positive integer occurs infinitely often in f. Then,* $\textsc{LimRef}(f)$ *is an absolutely normal number.*

Proof. Fix a base b and $\varepsilon > 0$. Let us prove that $D((\textsc{LimRef}(f))_b \restriction \ell, b) \leqslant \varepsilon$ for each sufficiently large ℓ. Let j_0 be large enough such that $f_j > \max(b, \lceil 8/\varepsilon \rceil)$ for every $j \geqslant j_0$. Consider $\ell > |w_b(\vec{R}_{j_0})|$, and noticing that $(|w_b(\vec{R}_j)|)_{j\in\mathbb{N}}$ is an increasing sequence, let j be such that $|w_b(\vec{R}_j)| \leqslant \ell < |w_b(\vec{R}_{j+1})|$. Observe that $(\textsc{LimRef}(f))_b \restriction \ell = w_b(\vec{R}_{j+1}) \restriction \ell$. Now note that $1/(f_j+2) \leqslant \varepsilon/8$ and apply point (2) of Lemma 3.6.4 to $\vec{R}_j = \textsc{Ref}_{f_j}(\vec{R}_{j-1})$ to conclude that

$$D(w_b(\vec{R}_j), b) \leqslant 2/(f_j+2) \leqslant \varepsilon/4.$$

By hypothesis, $f_j, f_{j+1} > b > 1$, so let $j_1 < j$ be such that $f_{j_1} = f_j - 1$ and $j_2 < j+1$ be such that $f_{j_2} = f_{j+1} - 1$. By point (4) of Lemma 3.6.4,

$$|w_b(\vec{R}_j)| \geqslant |w_b(\vec{R}_{j_1})| > \ell_{f_j}(f_j+2) > \lceil -\log(\delta_{f_j}) \rceil (f_j+2),$$

then,

$$\lceil -\log(\delta_{f_j}) \rceil / |w_b(\vec{R}_j)| < 1/(f_j+2) \leqslant \varepsilon/8.$$

Similarly,

$$|w_b(\vec{R}_j)| \geqslant |w_b(\vec{R}_{j_2})| > \ell_{f_{j+1}}(f_{j+1}+2),$$

then,

$$\ell_{f_{j+1}}/|w_b(\vec{R}_j)| < 1/(f_{j+1}+2) \leqslant \varepsilon/8.$$

Consider Lemma 3.6.4 again, but applied to $\vec{R}_{j+1} = \textsc{Ref}_{f_{j+1}}(\vec{R}_j)$. By point (5),

$$D(w_b(\vec{R}_{j+1}) \restriction \ell, b) \leqslant D(w_b(\vec{R}_j), b) + \frac{\lceil -\log(\delta_{f_j}) \rceil}{|w_b(\vec{R}_j)|} + \frac{1}{f_j+2} + \frac{\ell_{f_{j+1}}}{|w_b(\vec{R}_j)|}.$$

By the bounds established above, each term on the right part of the inequality is at most $\varepsilon/4$. So,

$$D((\textsc{LimRef}(f))_b \restriction \ell, b) = D(w_b(\vec{R}_{j+1}) \restriction \ell, b) \leqslant \varepsilon.$$

Notice that by the choice of j_0, j, j_1 and j_2 the sequences $\vec{R}_{j_0}, \vec{R}_j, \vec{R}_{j_1}, \vec{R}_{j_2}$ all contain a b-adic interval, hence the function w_b is defined on them. □

Lemma 3.6.8. *Let f be a sequence of positive integers such that all but finitely many occur infinitely often in f. Then,* $\textsc{LimRef}(f)$ *is absolutely abnormal.*

Proof. Fix a base b such that b appears infinitely often in f. By the conditions imposed on f, $(\textsc{LimRef}(f))_{b+1}$ has infinitely many prefixes of the form $x_{b+1}(\textsc{Ref}_b(\vec{I}))$ for some $\vec{I}$. By point (3) of Lemma 3.6.4,

$$D_{2\ell_b}(x_{b+1}(\textsc{Ref}_b(\vec{I})), b+1) > (b+1)^{-2\ell_b-1}.$$

Hence, for infinitely many prefixes of $(\textsc{LimRef}(f))_{b+1}$ their discrepancy to blocks of length $2\ell_b$ in base $b+1$ is bounded away from 0. Then, $\textsc{LimRef}(f)$ is not normal to base $b+1$. Since all but finitely many bases can be chosen as $b+1$, $\textsc{LimRef}(f)$ is not normal to all but finitely many bases. By Theorem 2.3.4 it is absolutely abnormal. □

3.7 Completeness results

Theorem 3.7.1. *There is a computable reduction from Π^0_3 sentences φ in $\mathcal{L}$ to indices e such that the following implications hold.*

If φ is true then e is the index of a computable absolutely normal number.

If φ is false then e is the index of a computable absolutely abnormal number.

Proof. Algorithm 3.5.1 outputs infinite sequences of positive integers and Algorithm 3.6.5 takes as input infinite sequences of positive integers. Thus, we can compose them into an algorithm that takes as input a sentence in $\mathcal{L}$ and returns as output a binary infinite word. Lemmas 3.5.2 and 3.6.7 guarantee that if φ is a true Π^0_3 sentence, then the output is an absolutely normal number. By Lemmas 3.5.2 and 3.6.8, if φ is false, then the output is an absolutely abnormal number.

By the $S_{m,n}$-theorem [Rog87], given φ, we can computably obtain the index of an appropriate infinite binary word. This completes the proof. □

Corollary 3.7.2. *The set of absolutely normal numbers is Π^0_3-complete, and hence $\boldsymbol{\Pi^0_3}$-complete.*

Proof. Lemma 3.4.1 states that the corresponding sets are in the Π^0_3 and $\boldsymbol{\Pi^0_3}$ classes. The hardness result in the effective case is immediate from Theorem 3.7.1 by relativization. We have the reduction from a Π^0_3 sentence in first order arithmetic to an appropriate index for a computable real number. By relativization, we obtain a reduction from a Π^0_3 statement about a real number x to an appropriate index of a real number which is computable from x.

For the non-effective case, recall that to prove hardness of subsets of reals at levels in the Borel hierarchy it is sufficient to consider subsets of Baire space $\mathbb{N}^{\mathbb{N}}$, because there is a continuous function from the real numbers to $\mathbb{N}^{\mathbb{N}}$ that preserves $\boldsymbol{\Pi}^0_3$ definability. Baire space admits a syntactic representation of the levels in the Borel hierarchy in arithmetical terms, namely a subset of $\mathbb{N}^{\mathbb{N}}$ can be defined by a Π^0_3 formula with a fixed parameter $P \in \mathbb{N}^{\mathbb{N}}$. The analysis given for the effective case, but now relativized to x and P, applies. □

The reduction in Theorem 3.7.1 gives just two possibilities: absolute normality or absolute abnormality; that is, normality to all bases simultaneously, or to no base at all. Consequently, it also separates normality in base b from non-normality in base b, for any given b. This gives an alternate proof of Ki and Linton's theorem in [KL94] for $\boldsymbol{\Pi^0_3}$-completeness, that also covers the case of Π^0_3-completeness.

Corollary 3.7.3. *For every base b, the set of normal numbers in base b is Π^0_3-complete, and hence $\boldsymbol{\Pi^0_3}$-complete.*

Proof. Observe that normality in all bases implies normality in each base. And absolute abnormality is lack of normality in every base. Thus, the same reductions used in the proof of Corollary 3.7.2 also prove the completeness results for just one fixed base. □

Another consequence of Theorem 3.7.1 is that absolute abnormality, which is a Π^0_4 property, is hard for the classes Σ^0_3 and $\boldsymbol{\Sigma}^0_3$.

CHAPTER 4

Compressibility

This chapter deals with the characterization of normality by compressibility using finite automata. A fundamental theorem relates normality and finite automata: an infinite word is normal to a given alphabet if and only if it cannot be compressed by lossless finite transducers. These are deterministic finite automata with injective input-output behavior. This result was first obtained by joining a theorem by Schnorr and Stimm [SS72] with a theorem by Dai, Lathrop, Lutz and Mayordomo [DLLM04]. A proof of one of the required implications was published in an article by Bourke, Hitchcock and Vinodchandran [BHV05]. The first result in this chapter is a direct proof of that theorem. The version here is a slight improvement of the one we published in [BH13].

What is the true power needed to compress normal words? Of course, transducers augmented with enough computational power are equivalent to a Turing machine, hence they can compress computable normal infinite words. Here we analyze different ways of augmenting a plain deterministic transducer. We consider deterministic versus real-time non-deterministic and non-real-time transition functions; having zero, one, or more counters; with or without a stack.

The power gained by allowing non-deterministic behavior is not obvious and depends on the context. Deterministic and the non-deterministic automata recognize the same sets of finite words, known as the rational sets of finite words. However, deterministic Büchi automata are strictly less expressive than the non-deterministic ones (see [PP04]). For instance, deterministic Büchi automata recognize a proper subclass of sets of infinite words than the non-deterministic ones. Furthermore, functions and relations realized by deterministic transducers are proper subclasses of rational functions and relations realized by non-deterministic ones.

We prove that lossless transducers, even non-deterministic non-real-time ones, but with no extra memory or just a single counter, cannot compress normal infinite words. Adding memory yields compressibility results: non-real-time non-deterministic transducers with more than one counter can compress some normal words. Also real-time non-deterministic transducers with a stack can do it.

Table 4.1 summarizes the results we obtain about compressibility of normal infinite words by different kinds of transducers. The columns represent different levels of restrictions on the transitions. The first column represents determinism, that is, there is exactly one transition that leaves a given state by reading a given symbol. The second column represents the possibility that several transitions leaving

extra memory	deterministic	non-deterministic	non-real-time
none	Not compress (Theorem 4.1.4)	Not compress (Theorem 4.3.1)	Not compress (Corollary 4.3.4)
one counter	Not compress	Not compress	Not compress (Theorem 4.4.2)
multiple counters	Not compress (Theorem 4.2.1)	Not compress (Corollary 4.4.1)	Compress (Turing complete)
one stack	?	Compress (Theorem 4.5.1)	Compress
one stack and one counter	Compress (Theorem 4.5.2)	Compress	Compress (Turing complete)

Table 4.1: Compressibility of normal infinite words by different kinds of transducers.

the same state read the same symbol. The restriction represented in the third column adds the possibility of also having transitions that do not read any symbol (usually called λ-transitions). The rows of the table represent different memory models. In all cases there is a bounded memory represented by states, and each row details possible additions of counters or stacks. The realized relation is assumed to be bounded-to-one. The case of a deterministic transducers with a single stack remains open:

Question. Can deterministic pushdown transducers compress normal words?

The rest of this chapter is organized as follows. Section 4.1 introduces the main concepts in the simplest form of deterministic transducers. In Sections 4.2 and 4.3 we consider non-determinism and counters, independently. Section 4.4 studies the joint occurrence of counters and non-determinism in a transducer. Finally, Section 4.5 considers pushdown transducers. These are the weakest machines we have found that can compress normal words. All results in this chapter are contained in [BCH13].

4.1 Deterministic transducers

We recall the standard definition of a deterministic transducer and fix notation. A transducer is an automaton equipped with an output tape. While it processes an input word, it produces an output word. The execution of each transition consumes at most one symbol from the input and produces a word on the output. Thus, the transducer outputs the concatenation of all the words output in the executed transitions.

We first introduce transducers where the underlying automaton is deterministic. These transducers are also called sequential in the literature [Sak09].

Definition. A deterministic *transducer* is a tuple $T = \langle Q, A, B, \delta, q_0 \rangle$, where

- Q is a finite set of states,
- A and B are the input and output alphabets, respectively,
- $\delta : Q \times A \to B^* \times Q$ is the transition function,

- $q_0 \in Q$ is the starting state.

The transducer T processes infinite words over A: if at state p symbol a is processed, T moves to state q and outputs v where $\langle v, q\rangle = \delta(p, a)$. In this case, we write $p \xrightarrow{a|v} q$.

A *finite run* of the transducer is a finite sequence of consecutive transitions

$$p_0 \xrightarrow{a_1|v_1} p_1 \xrightarrow{a_2|v_2} p_2 \cdots p_{n-1} \xrightarrow{a_n|v_n} p_n$$

and we write $p_0 \xrightarrow{u|v} p_n$ where $u = a_1a_2\cdots a_n$ and $v = v_1v_2\cdots v_n$.

An *infinite run* of the transducer is a sequence of consecutive transitions

$$p_0 \xrightarrow{a_1|v_1} p_1 \xrightarrow{a_2|v_2} p_2 \xrightarrow{a_3|v_3} p_3 \cdots$$

and we write $p_0 \xrightarrow{x|y} \infty$ where $x = a_1a_2a_3\cdots$ and $y = v_1v_2v_3\cdots$. An infinite run is accepting if $p_0 = q_0$. This is the Büchi acceptance condition where all states are accepting. We write $T(x)$ to refer to the word such that $q_0 \xrightarrow{x|T(x)} \infty$.

Hereafter a transducer is a deterministic transducer unless it is explicitly indicated. Notice that $T(x)$ is not required to be an infinite word. However, in the next theorems we impose a condition on the transducers thar ensures their output is infinite.

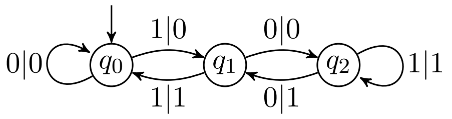

Figure 4.1: A transducer for the division by 3 in base 2.

The transducer pictured in Figure 4.1 realizes the following function from binary words to binary words. If the input x is the binary expansion of some real number $.x$, then output is the binary expansion of $.x/3$. This function is not one-to-one since dyadic numbers have two binary expansions. The two binary expansions $01111\cdots$ and $10000\cdots$ of $1/2$ are mapped to the unique binary expansion $0010101\cdots$ of $1/6$.

We say that a state q is reachable if there is a finite run from the starting state to q.

Definition. Let $T = \langle Q, A, B, \delta, q_0\rangle$ be a transducer.

1. T is *one-to-one* if the function $x \mapsto T(x)$ is one-to-one.
2. T is *lossless* if for every pair of different words u_1 and u_2, it is not true that $q_0 \xrightarrow{u_1|v} p$ and $q_0 \xrightarrow{u_2|v} p$ for some word v and state p.
3. T is *bounded-to-one* if the function $x \mapsto T(x)$ is bounded-to-one.

Being lossless is a property defined on the structure of the transducer whereas being one-to-one or bounded-to-one is defined on the realized function. Observe that the same function can be realized by a lossless or a non-lossless transducer as shown by the following example.

Consider the two transducers pictured in Figure 4.2. They both realize the shift function that maps each infinite word $x = a_1a_2a_3\cdots$ to the infinite word $y = a_2a_3a_4\cdots$ obtained by removing its first symbol. The first one is lossless whereas the second one is not. However they are both 2-to-one, and of course, neither is one-to-one.

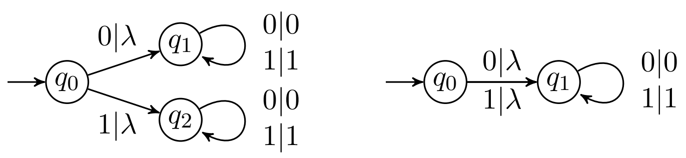

Figure 4.2: A lossless and a 2-to-one transducer.

Proposition 4.1.1. *one-to-one transducer is lossless. A lossless transducer is bounded-to-one.*

Proof. Let $T = \langle Q, A, B, \delta, q_0 \rangle$ be a transducer. For the first implication, let T be not lossless. Then there are different words u_1 and u_2, a word v and a state p such that $q_0 \xrightarrow{u_1|v} p$ and $q_0 \xrightarrow{u_2|v} p$. Let x be a non-periodic infinite word, then it is clear that $u_1x \neq u_2x$ but $q_0 \xrightarrow{u_1|v} p \xrightarrow{x|y} \infty$ and $q_0 \xrightarrow{u_2|v} p \xrightarrow{x|y} \infty$ for some y, thus $T(u_1x) = T(u_2x) = vy$, so T is not one-to-one.

For the second implication, let T be lossless. By the lossless property, a run of T cannot contain a cyclic run $p \xrightarrow{u|\lambda} p$ with p a reachable state, therefore $T(x)$ is infinite for all x. Let $m = \max\{|w| : a \in A, p, q \in Q, p \xrightarrow{a|w} q\}$ be the length of the longest output of a transition of T and $x_1, \ldots, x_n$ be different infinite words such that $y = T(x_i)$ for $i = 1, \ldots, n$. Then, for each such i let u_i be the prefix of x_i where $|u_i| = \ell$, $u_i \neq u_j$ for $i \neq j$. Let $k = \max\{|w| : 1 \leqslant i \leqslant n, p \in Q, q_0 \xrightarrow{u_i|w} p\}$. Let v_i be the shortest prefix of x_i such that $q_0 \xrightarrow{v_i|w_i} p_i$ with $|w_i| > k$. Note that $k+1 \leqslant |w_i| \leqslant k+m$ and by definition, for each i, $|v_i| > \ell$, hence the v_i are pairwise different. Since T is lossless, the tuples $\delta(q_0, v_i)$ are pairwise different and they are included in the set $Q \times \{y \upharpoonright j : k+1 \leqslant j \leqslant k+m\}$. Therefore, there are at most $|Q|m$ such tuples, so $n \leqslant |Q|m$ and T is $(|Q|m)$-to-one. □

Definition. An infinite word $x = a_1a_2a_3\cdots$ is *compressible* by a deterministic transducer if its accepting run $q_0 \xrightarrow{a_1|v_1} q_1 \xrightarrow{a_2|v_2} q_2 \xrightarrow{a_3|v_3} q_3 \cdots$ satisfies

$$\liminf_{n\to\infty} \frac{|v_1v_2\cdots v_n|}{n} \frac{\log|B|}{\log|A|} < 1.$$

Notice that the factor $\log|B|/\log|A|$ in the definition above just accounts for the necessary recoding from one alphabet to another one. Hereafter, to ease the presentation of the proofs we assume that $|A| = |B|$. The generalization is straightforward: if the lengths of words over B are multiplied by the factor $\log|B|/\log|A|$ whenever they are compared to lengths of words over A, all the proofs hold for the general case.

We prove first that there is a deterministic transducer that compresses any non-normal infinite word. Since every transducer we consider here is at least as powerful as a deterministic one, the next result applies to all of them.

Theorem 4.1.2. *Not normal infinite words are compressible by a one-to-one transducer with the same input and output alphabet.*

Proof. Assume x is not normal, then it is not ℓ-simply normal for some ℓ. Then, there is an increasing subsequence of positions $n_1, n_2, n_3, \cdots$ such that for a given word $u_1 \in A^\ell$, $\lim_{i\to\infty} \mathrm{boc}(x \upharpoonright n_i, u_1)/n_i = f_1 \neq |A|^{-|u_1|}$. Let $\{u_1, u_2, \cdots, u_{|A|^\ell}\} = A^\ell$. Let $n_{1,i} = n_i$ and for each $j = 2, 3, \cdots, |A|^\ell$ define $n_{j,1}, n_{j,2}, n_{j,3}, \cdots$ a subsequence of $n_{j-1,1}, n_{j-1,2}, n_{j-1,3}, \cdots$ such that $\lim_{i\to\infty} \mathrm{boc}(x \upharpoonright n_{j,i}, u_j)/n_{j,i} = f_j$ exists. Thus, on the sequence of positions $n_{|A|^\ell,1}, n_{|A|^\ell,2}, n_{|A|^\ell,3}, \cdots$ the limit exists for all words of length ℓ.

Our approach is to apply Huffman coding [Huf52] to the blocks of ℓ symbols of x to take advantage of the unbalanced frequencies. However, there is a rounding cost in doing so that may result in no overall compression. To minimize the rounding cost, we encode m blocks at a time.

Let us call superblocks to the concatenation of m blocks of ℓ symbols each. We associate to each superblock a desired code length. To the superblock $u_{i_1}u_{i_2}\cdots u_{i_m}$ we assign the desired length $\left\lceil -\sum_{j=1}^m \log_{|A|} f_{i_j} \right\rceil$. It is easy to check that the multiset of desired lengths for the $|A|^{m\ell}$ possible superblocks meets the condition of Theorem 2.2.5, and therefore there is a one-to-one coding function $c : A^{m\ell} \to A^*$ such that $|c(u_{i_1}u_{i_2}\cdots u_{i_m})| = \left\lceil -\sum_{j=1}^m \log_{|A|} f_{i_j} \right\rceil$ and the image of c is prefix free.

For each positive integer m define a transducer $T_m = \langle A, A, A^{<m\ell}, \delta, \lambda\rangle$ where

$$\delta(u,a) = \begin{cases} \langle ua, \lambda\rangle & |ua| < m\ell \\ \langle \lambda, c(ua)\rangle & |ua| = m\ell. \end{cases}$$

T_m applies c to each block of $m\ell$ symbols and concatenates the results, in order. Since c is one-to-one and its image is prefix-free, T_m is one-to-one.

Consider the accepting run of T_m over x factorized into subruns of $m\ell$ steps

$$q_0 \xrightarrow{s_1|c(s_1)} q_1 \xrightarrow{s_2|c(s_2)} q_3 \xrightarrow{s_3|c(s_3)} q_4 \cdots$$

where $s_k = u_{i_{k,1}}u_{i_{k,2}}\cdots u_{i_{k,m}}$. Notice that within each subrun all steps but the last output nothing. Consider now the compressibility limit over positions $n_j = n_{|A|^\ell,j}$,

$$\liminf_{j\to\infty} \frac{|c(s_1)c(s_2)\cdots c(s_{\lfloor n_j/(m\ell)\rfloor})| \log|A|}{n_j \log|A|},$$

that is equal to

$$\begin{aligned}
\liminf_{j\to\infty} \sum_{k=1}^{\lfloor n_j/(m\ell)\rfloor} \frac{|c(s_k)|}{n_j} &= \liminf_{j\to\infty} \sum_{k=1}^{\lfloor n_j/(m\ell)\rfloor} \frac{\left\lceil -\sum_{h=1}^m \log_{|A|} f_{i_{k,h}} \right\rceil}{n_j} \\
&\leqslant \liminf_{j\to\infty} \sum_{k=1}^{\lfloor n_j/(m\ell)\rfloor} \frac{1}{n_j} + \sum_{h=1}^{m} \frac{-\log_{|A|} f_{i_{k,h}}}{n_j} \\
&\leqslant \frac{1}{m\ell} - \frac{1}{\ell}\sum_{i=1}^{|A|^\ell} f_i \log_{|A|} f_i.
\end{aligned}$$

Since $f_1 \neq |A|^{-\ell}$, by Corollary 2.2.4,

$$-\sum_{i=1}^{|A|^\ell} f_i \log_{|A|} f_i < \log_{|A|} |A|^\ell = \ell.$$

Therefore,

$$-\sum_{i=1}^{|A|^\ell} f_i \log_{|A|} f_i/\ell < 1,$$

so we can let m be large enough such that

$$-\sum_{i=1}^{|A|^\ell} f_i \log_{|A|} f_i/\ell + 1/(m\ell) < 1.$$

Putting all the inequalities together, T_m compresses x. □

We now turn to incompressibility results. We first show that deterministic transducers do not compress normal infinite words. We use the following lemma. Due to its generality, it also applies to more powerful transducers.

Lemma 4.1.3. *Let ℓ be a positive integer, and let $u_1, u_2, u_3, \ldots$ be words of length ℓ over the alphabet A such that $u_1u_2u_3\cdots$ is simply normal to word length ℓ. Let*

$$C_0 \xrightarrow{u_1|v_1} C_1 \xrightarrow{u_2|v_2} C_2 \xrightarrow{u_3|v_3} C_3 \cdots$$

be a run where each C_i is a configuration of some kind of transducer. Assume there is a real $\varepsilon > 0$ and a set $U \subseteq A^\ell$ of at least $(1-\varepsilon)|A|^\ell$ words such that $u_i \in U$ implies $|v_i| \geqslant \ell(1-\varepsilon)$. Then,

$$\liminf_{n\to\infty} \frac{|v_1v_2\cdots v_n|}{n\ell} \geqslant (1-\varepsilon)^3.$$

Proof. Assume words u_i as in the hypothesis. By definition of normality to word length ℓ, let n_0 be such that for every $u \in A^\ell$ and $n \geqslant n_0$,

$$|\{i : 1 \leqslant i \leqslant n, u_i = u\}| \geqslant n|A|^{-\ell}(1-\varepsilon).$$

Then, for every $n \geqslant n_0$,

$$\begin{aligned}
|v_1v_2\cdots v_n| &= \sum_{i=1}^{n} |v_i| \\
&\geqslant \sum_{1\leqslant i\leqslant n, u_i\in U} |v_i| \\
&\geqslant \sum_{1\leqslant i\leqslant n, u_i\in U} \ell(1-\varepsilon) \\
&\geqslant n|A|^{-\ell}(1-\varepsilon)\sum_{u\in U} \ell(1-\varepsilon) \\
&\geqslant n|A|^{-\ell}(1-\varepsilon)(1-\varepsilon)|A|^{\ell}\ell(1-\varepsilon) \\
&\geqslant (1-\varepsilon)^3 n\ell.
\end{aligned}$$

□

Theorem 4.1.4. *Normal infinite words are not compressible by bounded-to-one transducers.*

Proof. Fix a normal infinite word $x = a_1a_2a_3\cdots$, a bounded-to-one transducer $T = \langle Q, A, B, \delta, q_0\rangle$ with only reachable states, a real $\varepsilon > 0$ and the accepting run $q_0 \xrightarrow{a_1|v_1} q_1 \xrightarrow{a_2|v_2} q_2 \xrightarrow{a_3|v_3} q_3 \cdots$. It suffices to show that there is ℓ and U such that Lemma 4.1.3 applies to this arbitrary choice of T and ε.

Let $h_u = \min\{|v| : p, q \in Q, p \xrightarrow{u|v} q\}$ be the minimum number of symbols that the processing of u can contribute to the output. Let $U_\ell = \{u : |u| = \ell, h_u \geqslant (1-\varepsilon)\ell\}$ be the set of words of length ℓ with relatively large contribution to the output. Let t be such that T is t-to-one. For each pair of states $p, q \in Q$ and word v, consider the set $U' = \{u : p \xrightarrow{u|v} q\}$. Since p is reachable, let u_0, v_0 be such that $q_0 \xrightarrow{u_0|v_0} p$. Thus, for different $u_1, u_2 \in U'$, $q_0 \xrightarrow{u_0u_1|v_0v} q$ and $q_0 \xrightarrow{u_0u_2|v_0v} q$, therefore $T(u_0u_1x) = T(u_0u_2x)$ for any x, and by the definition of t, $|U'| \leqslant t$. By bounding the cardinality of the sets U', we can bound the complement of U_ℓ.

$$\begin{aligned}
|\{u : p \xrightarrow{u|v} q\}| &\leqslant t \\
|\{u : p, q,\ p \xrightarrow{u|v} q\}| &\leqslant |Q|^2 t \\
|\{u : h_u < (1-\varepsilon)\ell\}| &\leqslant |Q|^2 t |B|^{(1-\varepsilon)\ell+2}.
\end{aligned}$$

Thus,

$$|U_\ell| \geqslant |A|^\ell - |Q|^2 t |B|^{(1-\varepsilon)\ell+2}.$$

Fix ℓ such that $|U_\ell| > |A|^\ell(1-\varepsilon)$, which is possible because the second term in the last inequality is $o(|A|^\ell)$ (recall $|A| = |B|$), and take $U = U_\ell$. By construction, the only run of T over x fulfills the hypothesis of Lemma 4.1.3 using states as configurations. The application of the lemma finishes the proof. □

4.2 Counter transducers

We consider deterministic transducers augmented with a fixed number of *counters.* Each counter contains an integer value. The execution of each transition reads exactly one input symbol, checks each counter for being zero or non-zero, and increments or decrements each counter by some amount.

Thus, a counter can only increase or decrease a bounded amount when processing a symbol of the input. Notice we are assuming by default a real-time restriction. We consider the consequences of removing this restriction later. When the range 1 to k is clear from the context, we write $\overline{m}$ for the k-tuple $\langle m_1, \ldots, m_k\rangle$ and $\overline{0}$ for the k-tuple $\langle 0, \ldots, 0\rangle$.

Definition. A (deterministic) *k-counter transducer* is a tuple $T = \langle Q, A, B, \delta, q_0\rangle$, where

- Q is a finite set of states,
- A and B are the input and output alphabets, respectively,
- $\delta : Q \times \{\text{true}, \text{false}\}^k \times A \to B^* \times Q \times \mathbb{Z}^k$ is the transition function,
- $q_0 \in Q$ is the starting state.

The transducer T processes infinite words over A: if at state p with counter i having value m_i symbol a is processed, T moves to state q, stores value $n_i = m_i + d_i$ in counter i and outputs v where $\langle v, q, \overline{d}\rangle = \delta(p, \overline{c}, a)$ and each c_i indicates whether $m_i = 0$ or not. Such a transition of the transducer is denoted by $\langle p, \overline{m}\rangle \xrightarrow{a|v} \langle q, \overline{n}\rangle$.

A *finite run* of the transducer is a finite sequence of consecutive transitions

$$\langle p_0, \overline{m}_0\rangle \xrightarrow{a_1|v_1} \langle p_1, \overline{m}_1\rangle \xrightarrow{a_2|v_2} \langle p_2, \overline{m}_2\rangle \ \cdots \ \langle p_{n-1}, \overline{m}_{n-1}\rangle \xrightarrow{a_n|v_n} \langle p_n, \overline{m}_n\rangle$$

and we write $\langle p_0, \overline{m}_0\rangle \xrightarrow{u|v} \langle p_n, \overline{m}_n\rangle$ where $u = a_1 a_2 \cdots a_n$ and $v = v_1 v_2 \cdots v_n$.

An *infinite run* of the transducer is a sequence of consecutive transitions

$$\langle p_0, \overline{m}_0\rangle \xrightarrow{a_1|v_1} \langle p_1, \overline{m}_1\rangle \xrightarrow{a_2|v_2} \langle p_2, \overline{m}_2\rangle \xrightarrow{a_3|v_3} \langle p_3, \overline{m}_3\rangle \ \cdots$$

and we write $\langle p_0, \overline{m}_0\rangle \xrightarrow{x|y} \infty$ where $x = a_1 a_2 a_3 \cdots$ and $y = v_1 v_2 v_3 \cdots$. An infinite run is accepting if $p_0 = q_0$ and all initial values of the counters are 0, namely, $\overline{m}_0 = \overline{0}$. This is the Büchi acceptance condition where all states are accepting. We write $T(x)$ to refer to the word such that $\langle q_0, \overline{0}\rangle \xrightarrow{x|T(x)} \infty$.

Notice that for given p, $\overline{m}$ and u there is a unique choice of v, q and $\overline{n}$ such that $\langle p, \overline{m}\rangle \xrightarrow{u|v} \langle q, \overline{n}\rangle$. A configuration $\langle q, \overline{m}\rangle$ is reachable if there is a finite run from the starting configuration $\langle q_0, \overline{0}\rangle$ to $\langle q, \overline{m}\rangle$. Extending the definition of bounded-to-one and compressibility to counter transducers is straightforward.

Definition. A counter transducer T is *bounded-to-one* if the function $x \mapsto T(x)$ is bounded-to-one.

Definition. An infinite word $x = a_1a_2a_3\cdots$ is *compressible* by a counter transducer if its accepting run $\langle q_0, \overline{0}\rangle \xrightarrow{a_1|v_1} \langle q_1, \overline{m}_1\rangle \xrightarrow{a_2|v_2} \langle q_2, \overline{m}_2\rangle \xrightarrow{a_3|v_3} \langle q_3, \overline{m}_3\rangle \cdots$ satisfies

$$\liminf_{n\to\infty} \frac{|v_1v_2\cdots v_n| \log|B|}{n \log|A|} < 1.$$

Theorem 4.2.1. *Normal infinite words are not compressible by bounded-to-one counter transducers.*

Before proving Theorem 4.2.1 we need to introduce some technical tools.

Definition. Let $T = \langle Q, A, B, \delta, q_0\rangle$ be a counter transducer. We call D_T the maximum absolute value of a counter increment or decrement in a transition in δ, $D_T = \max\{I\}$, where

$$I = \{|d_i| : p, q \in Q, a \in A, v \in B^*, \overline{c} \in \{\text{true}, \text{false}\}^k, \overline{d} \in \mathbb{Z}^k, \langle v, q, \overline{d}\rangle = \delta(p, \overline{c}, a)\}.$$

All runs with values of the counters far from 0 have a similar behavior, because if a counter does not become zero during the run, then its value has no impact. We formalize this with the concept of *template*.

Definition. For each positive integer L let $\pi_L : \mathbb{Z} \to [-L, L] \cup \{-\infty, +\infty\}$ be the function that identifies each integer in $(-\infty, -L)$ with $-\infty$ and each integer in $(L, +\infty)$ with $+\infty$,

$$\pi_L(n) = \begin{cases} -\infty & \text{if } n < -L \\ n & \text{if } -L \leqslant n \leqslant L \\ +\infty & \text{if } n > -L \end{cases}$$

The function π_L can be extended component-wise to tuples of integers by setting $\pi_L(\overline{m}) = \overline{n}$ where $n_i = \pi_L(m_i)$.

For a positive integer L, an *L-template* for a k-counter transducer is a triple $\langle \overline{M}, \overline{N}, \overline{O}\rangle$ where each $\overline{M}, \overline{N}, \overline{O}$ is a k-tuple in $([-L, L] \cup \{-\infty, +\infty\})^k$. We say that a run $\langle p, \overline{m}\rangle \xrightarrow{u|v} \langle q, \overline{n}\rangle$ of the transducer *complies* only to one L-template; namely, $\langle \pi_L(\overline{m}), \pi_L(\overline{n}), \pi_L(\overline{m} - \overline{n})\rangle$.

Observe that there are exactly $(2L+3)^{3k}$ L-templates of a k-counter transducer.

Lemma 4.2.2. *For two words of the same length ℓ, u and u', let $\langle p, \overline{m}\rangle \xrightarrow{u|v} \langle q, \overline{n}\rangle$ and $\langle p, \overline{m}'\rangle \xrightarrow{u'|v} \langle q, \overline{n}'\rangle$ be two runs of the same k-counter transducer T that comply to the same (ℓD_T)-template $\langle \overline{M}, \overline{N}, \overline{O}\rangle$. Then, there exists a run $\langle p, \overline{m}'\rangle \xrightarrow{u|v} \langle q, \overline{n}'\rangle$ of the same transducer.*

Proof. First notice that $m_i - n_i$ is bounded by ℓD_T because at each step of the run the counter cannot increase or decrease more than D_T. Thus, since both runs comply to the same (ℓD_T)-template and the difference in counters is within the interval $[-\ell D_T, \ell D_T]$, the difference in counters on the other run $m'_i - n'_i$ must coincide. This is true for each counter, so $\overline{m} - \overline{n} = \overline{m}' - \overline{n}'$ which implies $\overline{m} - \overline{m}' = \overline{n} - \overline{n}'$. Let us write the run over $u = a_1a_2\cdots a_\ell$ explicitly:

$$\langle p, \overline{m}\rangle = \langle p_0, \overline{m}_0\rangle \xrightarrow{a_1|v_1} \langle p_1, \overline{m}_1\rangle \xrightarrow{a_2|v_2} \cdots \xrightarrow{a_\ell|v_\ell} \langle p_\ell, \overline{m}_\ell\rangle = \langle q, \overline{n}\rangle$$

where $v = v_1 v_2 \cdots v_\ell$. Consider the tuple $\overline{d} = \overline{m} - \overline{m}' = \overline{n} - \overline{n}'$. We need only to prove that the following run over $u' = a_1' a_2' \cdots a_\ell'$,

$$\langle p, \overline{m}' \rangle = \langle p_0, \overline{m}_0 - \overline{d} \rangle \xrightarrow{a_1'|v_1} \langle p_1, \overline{m}_1 - \overline{d} \rangle \xrightarrow{a_2'|v_2} \cdots \xrightarrow{a_\ell'|v_\ell} \langle p_\ell, \overline{m}_\ell - \overline{d} \rangle = \langle q, \overline{n}' \rangle,$$

is a valid run. Let us show that in configuration $\langle p_j, \overline{m}_j \rangle$ counter i is zero if and only if it is zero in configuration $\langle p_j, \overline{m}_j - \overline{d} \rangle$. That is, $m_{j,i}$ is zero if and only if $m_{j,i} - d_i$ is zero. Then, validity of each step follows from validity of the corresponding step in the original run. If d_i is zero, the requirement follows immediately. If d_i is non-zero, then $m_i \neq m_i'$, and therefore m_i and m_i' are both greater than ℓD_T or both smaller than ℓD_T. Assume $m_i, m_i' > \ell D_T$. Since $m_{j,i} \geqslant m_{0,i} - D_T j = m_i - D_T j$ and $j \leqslant \ell$, $m_{j,i}$ is positive. And $m_{j,i} - d_i = m_{j,i} - m_i + m_i' = m_i' - (m_i - m_{j,i}) \geqslant m_i' - D_T j$ is also positive. If, on the other hand, $m_i, m_i' < \ell D_T$, it follows by symmetry that both $m_{j,i}$ and $m_{j,i} - d_i$ are negative. □

Proof of Theorem 4.2.1. Fix a normal infinite word x, a bounded-to-one k-counter transducer
$T = \langle Q, A, B, \delta, q_0 \rangle$, a real $\varepsilon > 0$ and the accepting run

$$\langle q_0, \overline{0} \rangle \xrightarrow{a_1|v_1} \langle q_1, \overline{m}_1 \rangle \xrightarrow{a_2|v_2} \langle q_2, \overline{m}_2 \rangle \xrightarrow{a_3|v_3} \cdots .$$

It suffices to show that there is ℓ and U such that Lemma 4.1.3 applies to this arbitrary choice of T and ε.

Let $h_u = \min\{|v| : p, q \in Q, \overline{m}, \overline{n} \in \mathbb{Z}^k, \langle p, \overline{m} \rangle \xrightarrow{u|v} \langle q, \overline{n} \rangle\}$ be the minimum number of symbols that the processing of u can contribute to the output. Let $U_\ell = \{u : |u| = \ell, h_u \geqslant (1-\varepsilon)\ell\}$ be the set of words of length ℓ with relatively large contribution to the output. Let t be such that T is t-to-one. For each pair of states $p, q \in Q$, word v and (ℓD_T)-template τ, consider the following set $U' = \{u : \overline{m}, \overline{n}, \langle p, \overline{m} \rangle \xrightarrow{u|v} \langle q, \overline{n} \rangle$ complies to $\tau, \langle p, \overline{m} \rangle$ is reachable$\}$. Let us name the n different words in U' by $u_1, u_2, \ldots, u_n$. Since each u_i is in U', let $\overline{m}_i$ and $\overline{n}_i$ be such that $\langle p, \overline{m}_i \rangle \xrightarrow{u_i|v} \langle q, \overline{n}_i \rangle$ complies to τ and $\langle p, \overline{m}_i \rangle$ is reachable. By Lemma 4.2.2 with $u = u_i$ and $u' = u_1$, for each i there exist runs $\langle p, \overline{m}_1 \rangle \xrightarrow{u_i|v} \langle q, \overline{n}_1 \rangle$. Since $\langle p, \overline{m}_1 \rangle$ is reachable, let u_0, v_0 be such that $\langle q_0, \overline{0} \rangle \xrightarrow{u_0|v_0} \langle p, \overline{m}_1 \rangle$. Therefore, there is an accepting run $\langle q_0, \overline{0} \rangle \xrightarrow{u_0|v_0} \langle p, \overline{m}_1 \rangle \xrightarrow{u_i|v} \langle q, \overline{n}_1 \rangle \xrightarrow{x|y} \infty$ which shows $T(u_0 u_i x) = v_0 v y$ for each i. Therefore, by definition of t, $n \leqslant t$ and $|U'| \leqslant t$. Now we can continue the proof as in the case with no counters, given in Theorem 4.1.4. We showed

$$|\{u : \overline{m}, \overline{n}, \langle p, \overline{m} \rangle \xrightarrow{u|v} \langle q, \overline{n} \rangle \text{ complies to } \tau, \langle p, \overline{m} \rangle \text{ is reachable}\}| \leqslant t,$$

which implies that the union over all possible p, q and τ, is not greater than $|Q|^2 (2\ell D_T + 3)^{3k} t$ and considering the union over all possible lengths not greater than $(1-\varepsilon)\ell$ yields

$$|\{u : |u| = \ell, h_u < (1-\varepsilon)\ell\}| \leqslant |Q|^2 (2\ell D_T + 3)^{3k} t |B|^{(1-\varepsilon)\ell + 2},$$

and

$$|U_\ell| \geqslant |A|^\ell - |Q|^2 (2\ell D_T + 3)^{3k} t |B|^{(1-\varepsilon)\ell + 2}.$$

Fix ℓ such that $|U_\ell| > |A|^\ell (1-\varepsilon)$ and apply Lemma 4.1.3 with $U = U_\ell$ to the considered run. This completes the proof. □

4.3 Non-deterministic transducers

We consider *non-deterministic transducers.* We focus first in those that operate in real-time, that is, they process exactly one input alphabet symbol per transition. Then we give the general case.

Definition. A *non-deterministic transducer* is a tuple $T = \langle Q, A, B, \delta, q_0, F\rangle$, where

- Q is a finite set of states,
- A and B are the input and output alphabets, respectively,
- $\delta \subset Q \times A \times B^* \times Q$ is a finite transition relation,
- $q_0 \in Q$ is the starting state.
- $F \subseteq Q$ is the set of accepting states,

The transducer T processes infinite words over A: if at state p symbol a is processed, T may move to a state q and output v whenever $\delta(p, a, v, q)$. In this case, we write $p \xrightarrow{a|v} q$. Finite and infinite runs are defined as in the case of deterministic transducers, and an infinite run is accepting if it starts at q_0 and visits infinitely often accepting states. This is the classical Büchi acceptance condition. We write $T(x, y)$ whenever there is an accepting run $q_0 \xrightarrow{x|y} \infty$.

Definition. A non-deterministic transducer T is *bounded-to-one* if and only if the function $y \mapsto |\{x : T(x, y)\}|$ is bounded.

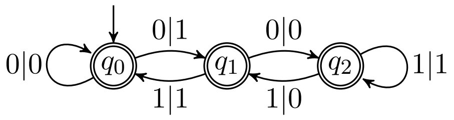

Figure 4.3: A transducer for the multiplication by 3 in base 2.

Exchanging the input and the output of each transition of the transducer of Figure 4.1 yields the transducer of Figure 4.3. This transducer is non-deterministic. If the input x is the infinite binary expansion in base 2 of some real number $.x < 1/3$, then the output is the binary expansion of $3 \times .x$.

Definition. An infinite word $x = a_1a_2a_3\cdots$ is *compressible* by a non-deterministic transducer if it has an accepting run $q_0 \xrightarrow{a_1|v_1} q_1 \xrightarrow{a_2|v_2} q_2 \xrightarrow{a_3|v_3} q_3 \cdots$ satisfying

$$\liminf_{n\to\infty} \frac{|v_1v_2\cdots v_n| \log|B|}{n \log|A|} < 1.$$

Theorem 4.3.1. *Normal infinite words are not compressible by bounded-to-one non-deterministic transducers.*

Proof. Fix a normal infinite word $x = a_1a_2a_3\cdots$, a real $\varepsilon > 0$, a bounded-to-one non-deterministic transducer $T = \langle Q, A, B, \delta, q_0, F\rangle$, and an accepting run, $q_0 \xrightarrow{a_1|v_1} q_1 \xrightarrow{a_2|v_2} q_2 \xrightarrow{a_3|v_3} q_3 \cdots$. It suffices to show that there is ℓ and U such that Lemma 4.1.3 applies to this arbitrary choice of ε, T and accepting run.

Let $h_u = \min\{|v| : 0 \leqslant i, j, q_i \xrightarrow{u|v} q_j\}$ be the minimum number of symbols that the processing of u can contribute to the output in the run we fixed. Let $U_\ell = \{u : |u| = \ell, h_u \geqslant (1-\varepsilon)\ell\}$ be the set of words of length ℓ with relatively large contribution to the output. Let t be such that T is t-to-one. For each pair of states q_i, q_j in the run and for each word v, consider the set $U' = \{u : q_i \xrightarrow{u|v} q_j\}$. Let $u_0 = a_1a_2 \cdots a_i$, $v_0 = v_1v_2 \cdots v_i$, $x_0 = a_{j+1}a_{j+2}a_{j+3} \cdots$ and $y_0 = v_{j+1}v_{j+2}v_{j+3} \cdots$. By definition, there exists a finite run $q_0 \xrightarrow{u_0|v_0} q_i$ and an infinite run $q_j \xrightarrow{x_0|y_0} \infty$ that goes infinitely often through accepting states. Thus, for different $u_1, u_2 \in U'$, there are accepting runs $q_0 \xrightarrow{u_0u_1x_0|v_0vy_0} \infty$ and $q_0 \xrightarrow{u_0u_2x_0|v_0vy_0} \infty$, from which it follows that $T(u_0u_1x_0, v_0vy_0)$ and $T(u_0u_2x_0, v_0vy_0)$. Therefore, by definition of t, $|U'| \leqslant t$. Now we can continue the proof as in the deterministic case, given in Theorem 4.1.4,

$$|\{u : q_i \xrightarrow{u|v} q_j\}| \leqslant t.$$

Thus,

$$|U_\ell| \geqslant |A|^\ell - |Q|^2 t |B|^{(1-\varepsilon)\ell+2}.$$

Fix ℓ such that $|U_\ell| > |A|^\ell(1-\varepsilon)$ and apply Lemma 4.1.3 with $U = U_\ell$ to the considered run. This completes the proof. □

4.3.1 Non-real-time non-deterministic transducers

A *non-real-time* (non-deterministic) transducer is a tuple $T = \langle Q, A, B, \delta, q_0, F\rangle$ which is identical to a real-time one except that that the transition relation δ is a finite subset of $Q \times (A \cup \{\lambda\}) \times B^* \times Q$ instead of $Q \times A \times B^* \times Q$. This allows the transducer to make progress without processing a symbol of the input word. If the machine is at state p and $\delta(p, \lambda, v, q)$ holds, the machine may move to state q and output v without processing the next input symbol. We extend the definition and notation of runs and accepting runs to this case, and write the relation T in the same way as for non-deterministic transducers.

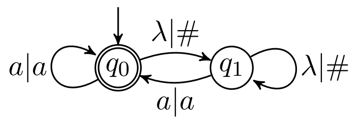

Figure 4.4: A non-real-time transducer that inserts dummy symbols #.

We give here an example of a relation T that can be realized by a non-real-time transducer but that cannot be realized by a real-time one. Let A be any alphabet and let B be the alphabet $A \cup \{\#\}$ obtained by adding to A a new dummy symbol $\# \notin A$. In this machine, $T(x, y)$ if and only if y is obtained by inserting dummy symbols in x. This relation T is clearly one-to-one since x is recovered from y by removing all the symbols #. It is easy to see that this relation cannot be realized by a real-time non-deterministic transducer. On the other hand, Figure 4.4 shows a non-real-time transducer that realizes it. Note that state q_0 is accepting. This forces the transducer to copy x entirely. Otherwise, the word y could end with a tail $\#^\omega$ and the transducer would not be bounded-to-one.

Definition. An infinite word x over A is *compressible* by a non-real-time transducer if it has an accepting run $q_0 \xrightarrow{b_1|v_1} q_1 \xrightarrow{b_2|v_2} q_2 \xrightarrow{b_3|v_3} q_3 \cdots$, where each $b_i \in A \cup \{\lambda\}$ and $x = b_1b_2b_3 \cdots$ satisfying

$$\liminf_{n\to\infty} \frac{|v_1v_2 \cdots v_n| \log |B|}{|b_1b_2 \cdots b_n| \log |A|} < 1.$$

In this definition, if every b_i were different from λ then $|b_1 b_2 \cdots b_n| = n$ and the definition would coincide with compressibility for real-time transducers.

Real-time transducers always read the entire input. However, non-real-time transducers may loop forever using transitions without input. The next lemma shows that this cannot happen for accepting runs of bounded-to-one non-real-time transducers.

Lemma 4.3.2. *Any accepting run of a bounded-to-one non-real-time transducer reads the entire input.*

Proof. Fix the transducer T. If an accepting run over input wx with output y reads only w, then the same run is an accepting run of any input wx', thus, making $T(wx', y)$ true for any x', which contradicts the bounded-to-one condition. □

Theorem 4.3.3. *For any bounded-to-one non-real-time transducer there exists a (real-time) non-deterministic transducer that compresses exactly the same infinite words.*

Proof. Let $T = \langle Q, A, B, \delta, q_0, F\rangle$ be a non-real-time transducer. Let us show that for any run of T consisting of n transitions $p_0 \xrightarrow{\lambda|v_1} p_1 \xrightarrow{\lambda|v_2} p_2 \cdots p_{n-1} \xrightarrow{\lambda|v_n} p_n$ there exists a run $p_0 \xrightarrow{\lambda|v} p_n$ consisting of no more than $2|Q|$ transitions, such that $|v| \leqslant |v_1 v_2 \cdots v_n|$ and it visits an accepting state if and only if the original one does. If $n > 2|Q|$, there is a state q visited three times in the run, so we can write the run as $p_0 \xrightarrow{\lambda|w_1} q \xrightarrow{\lambda|w_2} q \xrightarrow{\lambda|w_3} q \xrightarrow{\lambda|w_4} p_n$ where $v_1 v_2 \cdots v_n = w_1 w_2 w_3 w_4$. Notice that if $q = p_0$ and/or $q = p_n$ the first and/or last subruns may be empty. If the subrun $q \xrightarrow{\lambda|w_2} q$ visits an accepting state, then we take the run $p_0 \xrightarrow{\lambda|w_1} q \xrightarrow{\lambda|w_2} q \xrightarrow{\lambda|w_4} p_n$ that is shorter and outputs no more than the original while still visiting an accepting state. If the subrun does not visit an accepting state, then we take the run $p_0 \xrightarrow{\lambda|w_1} q \xrightarrow{\lambda|w_3} q \xrightarrow{\lambda|w_4} p_n$, which is also shorter and produces no more output. Also, since we removed a subrun that does not visit an accepting state, the new run has the required property. By induction, this proves the claim.

Consider the real-time non-deterministic transducer $T' = \langle Q, A, B, \delta', q_0, F\rangle$ where $\delta'(p, a, vw, q)$ if and only if there is a state r such that $p \xrightarrow{\lambda|v} r$ with at most $2|Q|$ transitions and $r \xrightarrow{a|w} q$. From the initial claim and Lemma 4.3.2, it is easy to see that an infinite word x is compressed by T if and only if it is compressed by a run of T that does not use more than $2|Q|$ consecutive transitions of the form $p \xrightarrow{\lambda|v'} q$. It is also easy to check that any such run induces a run in T' which also compresses x. □

Corollary 4.3.4. *Normal infinite words are not compressible by bounded-to-one non-real-time transducers.*

Proof. Immediate combining Theorem 4.3.1 and Theorem 4.3.3. □

4.4 Non-deterministic transducers with counters

It is straightforward to combine the proofs of incompressibility for (real-time) non-deterministic transducers and counter transducers, respectively Theorems 4.3.1 and 4.2.1, to obtain the following corollary.

Corollary 4.4.1. *Normal infinite words are not compressible by bounded-to-one non-deterministic counter transducers.*

Proof. Combine the proofs of Theorem 4.2.1 and Theorem 4.3.1. The only non-trivial point to notice is that, when defining U' as in the proof of Theorem 4.3.1, in addition to requiring that $\langle p, \overline{m} \rangle$ be reachable, we need that there exists an infinite run starting from $\langle q, \overline{n} \rangle$ that goes infinitely often through accepting states. It is easy to check that these requirements are met and they do not invalidate any step of the proof. □

As it stands, our proof for counter transducers cannot be extended to non-real-time because in the non-real-time case the processing of a fixed word may increase or decrease the counter an unbounded amount. This voids the argument we used to give an upper bound of the set U'. On the other hand, a non-real-time transducer with two or more counters is Turing-complete [Min61], so it can compress computable normal infinite words.

This raises the question of whether non-real-time transducers augmented with a single counter can compress normal infinite words. We answer it in the following theorem.

A *non-real-time* 1*-counter transducer* is a tuple $T = \langle Q, A, B, \delta, q_0, F \rangle$, identical to a non-real-time transducer, but the transition relation δ is a subset of $Q \times \{\text{true}, \text{false}\} \times (A \cup \{\lambda\}) \times B^* \times Q \times \mathbb{Z}$. This means that the transducer can check the zero or non-zero status of the counter and modify it in each transition. Definitions for bounded-to-one and compressible straightforwardly extend to this setting.

Theorem 4.4.2. *Normal infinite words are not compressible by bounded-to-one non-real-time* 1*-counter transducers.*

In the proof of Theorem 4.2.1 we used Lemma 4.2.2 to deal with counters. We bounded their increase or decrease by the length of the input word. For non-real-time transducers this argument fails. Lemma 4.4.7 gives an alternative argument. This is a quite technical result, that we prove using Lemmas 4.4.3 to 4.4.6.

For a non-real-time 1-counter transducer T, as for deterministic counter transducers, let D_T be the maximum absolute value of a counter increment or decrement in a transition of T.

Lemma 4.4.3. *For any non-real-time* 1*-counter transducer* T *there is another non-real-time* 1*-counter transducer* T' *that realizes the same relation, compresses exactly the same infinite words and has* $D_{T'} \leqslant 1$.

Proof. We can simply emulate increasing (or decreasing) by k with k steps that do no input or output, and increase (or decrease) by 1. Since the maximum possible k is bounded by D_T, we can do it with finitely many states and transitions. Formally, if $T = \langle Q, A, B, \delta, q_0, F \rangle$ we let $T' = \langle Q \times [-D_T, D_T], A, B, \delta', \langle q_0, 0 \rangle, F \times \{0\} \rangle$ where

$$\begin{aligned}\delta' = {} & \{\langle \langle q, k+1 \rangle, c, \lambda, \lambda, +1, \langle q, k \rangle \rangle : 0 \leqslant k < D_T, c \in \{\text{true}, \text{false}\}\} \ \cup \\ & \{\langle \langle q, k-1 \rangle, c, \lambda, \lambda, -1, \langle q, k \rangle \rangle : D_T < k \leqslant 0, c \in \{\text{true}, \text{false}\}\} \ \cup \\ & \{\langle \langle p, 0 \rangle, c, a, v, 0, \langle q, k \rangle \rangle : \delta(p, c, a, v, k, q)\}.\end{aligned}$$

It is immediate to check that any step $\langle p, n \rangle \xrightarrow{a|v} \langle q, m \rangle$ of T induces a finite run $\langle \langle p, 0 \rangle, n \rangle \xrightarrow{a|v} \langle \langle q, 0 \rangle, m \rangle$ of at most $D_T + 1$ steps of T' and viceversa, which in turn implies they realize the same relation and compress the same infinite words. By inspection of δ' it is clear that $D_{T'} \leqslant 1$. □

Lemma 4.4.4. *Let* $\langle q_0, 0 \rangle \xrightarrow{a_1|v_1} \langle q_1, m_1 \rangle \xrightarrow{a_2|v_2} \langle q_2, m_2 \rangle \xrightarrow{a_3|v_3} \langle q_3, m_3 \rangle \cdots$ *be an accepting run of a bounded-to-one non-real-time* 1*-counter. If there is a prefix*

$\langle q_0, 0\rangle \xrightarrow{a_1a_2\cdots a_n|v_1v_2\cdots v_n} \langle q_n, m_n\rangle$ of the run such that each accepting run that starts with it does not contain more configurations with a counter value 0*, then each such accepting run does not compress $a_1a_2a_3\cdots$.*

Proof. Let $T = \langle Q, A, B, \delta, q_0, F\rangle$ be the transducer, and let n be as in the hypothesis. We can emulate the process using a transducer without a counter. Let $T' = \langle Q \cup \{r_0, r_1, \ldots, r_n\}, A, B, \delta', r_0, F\rangle$ be a non-real-time transducer, where

$$\begin{aligned}\delta' = \{p &\xrightarrow{a|v} q : d \in \mathbb{Z}, \delta(p, a, \text{false}, v, q, d)\}\cup\\ \{r_i &\xrightarrow{a_{i+1}|v_{i+1}} r_{i+1} : 0 \leqslant i < n\}\cup\\ \{r_n &\xrightarrow{a_{n+1}|v_{n+1}} q_{n+1}\}\end{aligned}$$

and $p \xrightarrow{a|v} q$ stands for the tuple $\langle p, a, v, q\rangle$. Clearly, for each accepting run of T',

$$r_0 \xrightarrow{a_1|v_1} r_1 \xrightarrow{a_2|v_2} \cdots \xrightarrow{a_n|v_n} r_n \xrightarrow{b_{n+1}|w_{n+1}} p_{n+1} \xrightarrow{b_{n+2}|w_{n+2}} \cdots$$

there is an accepting run of T,

$$\langle q_0, 0\rangle \xrightarrow{a_1|v_1} \langle q_1, m_1\rangle \xrightarrow{a_2|v_2} \cdots \xrightarrow{a_n|v_n} \langle q_n, m_n\rangle \xrightarrow{b_{n+1}|w_{n+1}}$$

$$\langle p_{n+1}, m'_{n+1}\rangle \xrightarrow{b_{n+2}|w_{n+2}} \langle p_{n+2}, m'_{n+2}\rangle \xrightarrow{b_{n+3}|w_{n+3}} \cdots$$

because by hypothesis the accepting run of T does not visit a configuration with a counter value 0 after step n, and then every transition is valid. Therefore, since T is bounded-to-one, T' is bounded-to-one. By Corollary 4.3.4, T' does not compress $a_1a_2a_3\cdots$, so the given run of T does not compress it either. □

Lemma 4.4.5. *Let T be a non-real-time* 1*-counter transducer with $D_T \leqslant 1$ and $k \in \mathbb{N}$. A finite run of T, $\langle p, m\rangle \xrightarrow{u|v} \langle q, n\rangle$ such that $|m - n| \geqslant (k + 1)(|u| + |v| + 1)$, contains a run $\langle p', m'\rangle \xrightarrow{\lambda|\lambda} \langle q', n'\rangle$ with $|m' - n'| \geqslant k$ and $(m - n)(m' - n') > 0$.*

Proof. Assume $m-n \geqslant (k+1)(|u|+|v|+1)$. The case $m-n \leqslant -(k+1)(|u|+|v|+1)$ follows by symmetry. The counter decreases by at least $(k + 1)(|u| + |v| + 1)$ during the run. At most $|u| + |v|$ of that decrease happens in transitions reading input, or writing output or both. Then, the counter decreases by at least $k(|u| + |v| + 1)$ in transitions with no input nor output. Those are divided into at most $|u| + |v| + 1$ consecutive groups by the $|u| + |v|$ transitions that do input or output or both. By pigeonhole principle, at least one of those groups decreases the counter by at least k, yielding the desired run. □

Lemma 4.4.6. *Let $T = \langle Q, A, B, \delta, q_0, F\rangle$ be a non-real-time* 1*-counter transducer with $D_T \leqslant 1$. Assume $\langle p, m\rangle \xrightarrow{u_1|v_1} \langle q_1, h_1\rangle \xrightarrow{u_2|v_2} \langle q_2, h_2\rangle \xrightarrow{u_3|v_3} \langle r, m\rangle$ is a finite run of T such that all the values of the counter within it are greater than $|Q|^2 + 1$ and $\min(h_1, h_2) - m \geqslant (|Q|^2 + 1)(|u_1| + |v_1| + |u_2| + |v_2| + 1)$. Then, there is a run of T of the form $\langle p, m\rangle \xrightarrow{u_1|v_1} \langle q_1, h'_1\rangle \xrightarrow{u_2|v_2} \langle q_2, h'_2\rangle \xrightarrow{u_3|v_3} \langle r, m\rangle$ such that $0 < h_1 - h'_1 = h_2 - h'_2 \leqslant |Q|^2$ and the subrun $\langle q_1, h'_1\rangle \xrightarrow{u_2|v_2} \langle q_2, h'_2\rangle$ uses the same sequence of transitions as the original subrun $\langle q_1, h_1\rangle \xrightarrow{u_2|v_2} \langle q_2, h_2\rangle$ but with the counter values shifted by $h_1 - h'_1 = h_2 - h'_2$.*

Proof. By Lemma 4.4.5 there are runs

$$\langle p_1, m_1\rangle \xrightarrow{\lambda|\lambda} \langle p_2, m_2\rangle \xrightarrow{\lambda|\lambda} \cdots \xrightarrow{\lambda|\lambda} \langle p_k, m_k\rangle \text{ inside } \langle p, m\rangle \xrightarrow{u_1|v_1} \langle q_1, h_1\rangle$$

with $m_k - m_1 \geqslant |Q|^2$, and

$$\langle r_1, n_1\rangle \xrightarrow{\lambda|\lambda} \langle r_2, n_2\rangle \xrightarrow{\lambda|\lambda} \cdots \xrightarrow{\lambda|\lambda} \langle r_\ell, n_\ell\rangle \text{ inside } \langle q_2, h_2\rangle \xrightarrow{u_3|v_3} \langle r, m\rangle$$

with $n_1 - n_\ell \geqslant |Q|^2$. Letting

$$\rho = \langle p, m\rangle \xrightarrow{u_{1,1}|v_{1,1}} \langle p_1, m_1\rangle \xrightarrow{\lambda|\lambda} \cdots \xrightarrow{\lambda|\lambda} \langle p_k, m_k\rangle \xrightarrow{u_{1,2}|v_{1,2}} \langle q_1, h_1\rangle, \text{ and}$$
$$\sigma = \langle q_2, h_2\rangle \xrightarrow{u_{3,1}|v_{3,1}} \langle r_1, n_1\rangle \xrightarrow{\lambda|\lambda} \cdots \xrightarrow{\lambda|\lambda} \langle r_\ell, n_\ell\rangle \xrightarrow{u_{3,2}|v_{3,2}} \langle r, m\rangle,$$

the run can be written as

$$\rho \xrightarrow{u_2|v_2} \sigma.$$

Since $D_T \leqslant 1$, the value of the counter is set in every integer in the range $[m_1, m_k]$ in a configuration of the first run and $[n_\ell, n_1]$ in a configuration of the second run. Let us pair the value m_1 with n_ℓ, $m_1 + 1$ with $n_\ell + 1$ and so on. We have at least $|Q|^2 + 1$ such pairs, so consider the first $|Q|^2 + 1$ pairs. By pigeonhole principle, there are indices i_1, i_2, j_1, j_2 such that $1 \leqslant i_1 < i_2 \leqslant k, 1 \leqslant j_2 < j_1 \leqslant \ell$ and m_{i_1} is paired with n_{j_1}, m_{i_2} is paired with n_{j_2}, $p_{i_1} = p_{i_2}$ and $r_{j_1} = r_{j_2}$. By definition of the pairing $m_{i_1} - n_{j_1} = m_{i_2} - n_{j_2} = m_1 - n_\ell$; therefore, $0 < m_{i_2} - m_{i_1} = n_{j_2} - n_{j_1} = \Delta \leqslant |Q|^2$. We construct a run as required in the statement of the lemma.

- subtract Δ from the value of the counter to all configurations between $\langle p_{i_2}, m_{i_2}\rangle$ and $\langle r_{j_2}, n_{j_2}\rangle$ inclusive, in the original run;
- identify $\langle p_{i_1}, m_{i_1}\rangle = \langle p_{i_2}, m_{i_2} - \Delta\rangle$ and $\langle r_{j_2}, n_{j_2} - \Delta\rangle = \langle r_{j_1}, n_{j_1}\rangle$; and
- remove $\langle p_{i_1}, m_{i_1}\rangle \xrightarrow{\lambda|\lambda} \cdots \xrightarrow{\lambda|\lambda} \langle p_{i_2}, m_{i_2}\rangle$ and $\langle r_{j_2}, n_{j_2}\rangle \xrightarrow{\lambda|\lambda} \cdots \xrightarrow{\lambda|\lambda} \langle r_{j_1}, n_{j_1}\rangle$.

These modifications do not invalidate the run because $\Delta \leqslant |Q|^2$, so, the values of the counter everywhere in the run are positive, as in the original. □

The following lemma has the role that Lemma 4.2.2 had for real-time transducers, and it is the key piece in the proof of Theorem 4.4.2.

Lemma 4.4.7. *Let $T = \langle Q, A, B, \delta, q_0, F\rangle$ be a non-real-time 1-counter transducer with $D_T \leqslant 1$. Let $\langle q_0, 0\rangle \xrightarrow{u_1|v_1} \langle q_1, m_1\rangle \xrightarrow{u_2|v_2} \langle q_2, m_2\rangle \xrightarrow{u_3|v_3} \langle q_3, 0\rangle$ be a prefix of an accepting run of T. Then, there is a finite run $\langle q_1, n_1\rangle \xrightarrow{u_2|v_2} \langle q_2, n_2\rangle$ contained in an accepting run of T and with $|n_1|, |n_2| \leqslant 2(|Q|^2 + 1)(|u_2| + |v_2| + 1)$.*

Proof. Let $K = (|Q|^2 + 1)(|u_2| + |v_2| + 1)$. Let us prove the following claim: For a run

$$\rho = \langle q_0, 0\rangle \xrightarrow{u_1|v_1} \langle q_1, m_1\rangle \xrightarrow{u_2|v_2} \langle q_2, m_2\rangle \xrightarrow{u_3|v_3} \langle q_3, 0\rangle$$

with the requirements of the hypothesis, if $|m_1| > 2K$ or $|m_2| > 2K$, there is another run

$$\rho' = \langle q_0, 0\rangle \xrightarrow{u_1|v_1} \langle q_1, m_1'\rangle \xrightarrow{u_2|v_2} \langle q_2, m_2'\rangle \xrightarrow{u_3|v_3} \langle q_3, 0\rangle$$

where the same requirements hold and $m_1' \leqslant m_1, m_2' \leqslant m_2$ and $m_1' + m_2' < m_1 + m_2$. Then, by iterating this until m_1 and m_2 are not greater than $2K$, we obtain a run whose middle part is the run required in the statement.

Let us consider first $m_1 > 2K$. Since $D_T \leqslant 1$ inside $\langle q_0, 0\rangle \xrightarrow{u_1|v_1} \langle q_1, m_1\rangle$ there is a value of the counter exactly at K. We can thus get a shortest suffix

$\langle p, K\rangle \xrightarrow{u_{1,2},v_{1,2}} \langle q_1, m_1\rangle$ so that within it the counter does not take values below K. There are two cases.

If within $\langle q_1, m_1\rangle \xrightarrow{u_2|v_2} \langle q_2, m_2\rangle$ the counter takes a value below K (possibly $m_2 = K$), let $\langle q_1, m_1\rangle \xrightarrow{u_{2,1}|v_{2,1}} \langle r, K\rangle$ be the shortest prefix so that within it the counter does not take values below K. By applying Lemma 4.4.6 to the subrun

$$\rho_1 = \langle p, K\rangle \xrightarrow{u_{1,2},v_{1,2}} \langle q_1, m_1\rangle \xrightarrow{\lambda|\lambda} \langle q_1, m_1\rangle \xrightarrow{u_{2,1}|v_{2,1}} \langle r, K\rangle$$

we get another run

$$\rho'_1 = \langle p, K\rangle \xrightarrow{u_{1,2},v_{1,2}} \langle q_1, m'_1\rangle \xrightarrow{\lambda|\lambda} \langle q_1, m'_1\rangle \xrightarrow{u_{2,1}|v_{2,1}} \langle r, K\rangle$$

and replacing ρ_1 by ρ'_1 in ρ we get a ρ' as required.

If, on the other hand, within $\langle q_1, m_1\rangle \xrightarrow{u_2|v_2} \langle q_2, m_2\rangle$ the counter does not take a value below K, we consider $\langle q_2, m_2\rangle \xrightarrow{u_3|v_3} \langle q_3, 0\rangle$ and take $\langle q_2, m_2\rangle \xrightarrow{u_{3,1}|v_{3,1}} \langle r, K\rangle$ to be a shortest prefix such that within it the counter does not take a value below K. By applying Lemma 4.4.6 to the subrun

$$\rho_2 = \langle p, K\rangle \xrightarrow{u_{1,2},v_{1,2}} \langle q_1, m_1\rangle \xrightarrow{u_2|v_2} \langle q_2, m_2\rangle \xrightarrow{u_{3,1}|v_{3,1}} \langle r, K\rangle$$

we get another run

$$\rho'_2 = \langle p, K\rangle \xrightarrow{u_{1,2},v_{1,2}} \langle q_1, m'_1\rangle \xrightarrow{u_2|v_2} \langle q_2, m'_2\rangle \xrightarrow{u_{3,1}|v_{3,1}} \langle r, K\rangle$$

and replacing ρ_2 by ρ'_2 in ρ we get a ρ' as required.

The cases $m_1 < -2K$, $m_2 > 2K$ and $m_2 < -2K$ follow by symmetric arguments.

□

Proof of Theorem 4.4.2. Fix a normal infinite word x, a bounded-to-one non-real-time 1-counter transducer $T = \langle Q, A, B, \delta, q_0, F\rangle$, a real $\varepsilon > 0$ and the accepting run

$$\langle q_0, 0\rangle \xrightarrow{a_1|v_1} \langle q_1, m_1\rangle \xrightarrow{a_2|v_2} \langle q_2, m_2\rangle \xrightarrow{a_3|v_3} \langle q_3, m_3\rangle \cdots .$$

By Lemma 4.3.2, T reads the entire input. It suffices to show that there is ℓ and U such that Lemma 4.1.3 applies to this arbitrary choice of T and ε. By Lemma 4.4.3, let us assume without loss of generality that $D_T \leqslant 1$.

By Lemma 4.4.4, if there is a prefix of the run that cannot be extended to an accepting run visiting a configuration with a counter value 0, the run does not compress the input. Assume there is no such prefix. Let $h_u = \min\{|v| : \langle q_i, m_i\rangle \xrightarrow{u|v} \langle q_j, m_j\rangle\}$ be the minimum number of symbols that the processing of u can contribute to the output in the run we fixed. Let $U_\ell = \{u : |u| = \ell, h_u \geqslant (1-\varepsilon)\ell\}$ be the set of words of length ℓ with relatively large contribution to the output.

Let t be such that T is t-to-one. For states p, q, integers m, n word v and length ℓ, consider $U' = \{u \in A^\ell :$ an accepting run of T contains $\langle p, m\rangle \xrightarrow{u|v} \langle q, n\rangle\}$. Let $u_1, u_2, \ldots, u_n \in U'$ be pairwise different. Let

$$\langle q_0, 0\rangle \xrightarrow{u_0|v_0} \langle p, m\rangle \xrightarrow{u_1|v} \langle q, n\rangle \xrightarrow{x|y} \infty$$

be an accepting run that justifies $u_1 \in U'$. By construction, for each i there are accepting runs

$$\langle q_0, 0\rangle \xrightarrow{u_0|v_0} \langle p, m\rangle \xrightarrow{u_i|v} \langle q, n\rangle \xrightarrow{x|y} \infty.$$

Thus, $T(u_0u_ix, v_0vy)$ and $|U'| \leqslant n \leqslant t$.

Assume u produces an output v somewhere along the run we fixed; that is, there is a prefix of the run $\langle q_0, 0\rangle \xrightarrow{u_0|v_0} \langle q_i, m_i\rangle \xrightarrow{u|v} \langle q_j, m_j\rangle$. By our assumption this can be extended to a run of the form $\langle q_0, 0\rangle \xrightarrow{u_0|v_0} \langle q_i, m_i\rangle \xrightarrow{u|v} \langle q_j, m_j\rangle \xrightarrow{u_1|v_1} \langle p, 0\rangle$ that is the prefix of an accepting run. Apply Lemma 4.4.7 to this prefix. If $u \in U'$ for $p = q_i$, $q = q_j$, $\ell = |u|$ and arbitrary values of m and n, then u is also in U' for m and n having absolute value not greater than $2(|Q|^2+1)(\ell+|v|+1)$. So, if $h_u < (1-\varepsilon)\ell$, $u \in U'$ for some values of p, q, ℓ, v, m and n with the restriction $|v| < (1-\varepsilon)\ell$ and $|m|, |n| \leqslant 2(|Q|^2+1)(\ell+|v|+1)$. There are at most

$$|Q|^2(4(|Q|^2+1)(\ell+(1-\varepsilon)\ell+1)+1)^2|B|^{(1-\varepsilon)\ell}$$

combinations of those values with the mentioned restrictions. Since each $|U'| \leqslant t$ there are at most

$$t|Q|^2(4(|Q|^2+1)(\ell+(1-\varepsilon)\ell+1)+1)^2|B|^{(1-\varepsilon)\ell+2}$$

values of u among those. This magnitude is $o(|A|^\ell)$, so we can fix ℓ such that $|U_\ell| > |A|^\ell(1-\varepsilon)$ and take $U = U_\ell$. By construction, the fixed run of T over x fulfills the hypothesis of Lemma 4.1.3 using tuples of states and a counter value as configurations. □

4.5 Pushdown transducers

A *pushdown transducer* is a transducer equipped with a single stack as memory that can hold elements of a finite given alphabet. The machine makes decisions based on the input and the top symbol of the stack. When moving, it can pop symbols from or push symbols to the stack. A special bottom symbol $\perp$ as top represents the empty stack.

If the alphabet of the stack (symbols that can be pushed into the stack, not including the bottom symbol) is unary, the only relevant information for the current configuration is the number of symbols contained in the stack. Also, the automaton can only test for the number to be zero (empty stack) or non-zero (non-empty stack). The stack is then equivalent to a counter with only non-negative values, although counters with positive and negative values can also be emulated with the help of the automaton states. This shows that a stack gives at least as much power as a counter.

In this section we show that either non-deterministic pushdown transducers or deterministic pushdown transducers with a single additional counter can compress normal infinite words. The latter implies that deterministic pushdown transducers with at least two stacks can compress normal infinite words. The question remains open for deterministic pushdown transducers with a single stack.

In both cases we show that a particular transducer can compress the same infinite word. Let x_0 be a normal infinite word and let $u_i = x_0 \upharpoonright 2^{i-1}$ be its prefix of length 2^{i-1}. We work with the infinite word $x_1 = u_1\tilde{u}_1u_2\tilde{u}_2u_3\tilde{u}_3\cdots$, where $\tilde{u}_i$ is the reverse word of u_i. It is easy to see that this infinite word is also normal, for instance with an argument similar to Champernowne's original argument [Cha33].

4.5.1 Non-deterministic pushdown transducer

In a non-deterministic pushdown transducer with a stack each transition depends on the current state, the top symbol of the stack and the input symbol. It produces an output word, a word of stack symbols that replaces the top symbol, and the new state. The transition relation δ is a finite subset of $Q \times C \times A \times B^* \times Q \times C^*$ where Q is the state set, A and B are the input and output alphabets and C is the stack alphabet. Note that the top symbol of the stack is always replaced by a word w over the stack alphabet. This means that the transducer pops the top symbol if $|w| = 0$, replaces the top symbol by another if $|w| = 1$ and pushes several symbols if $|w| > 1$.

Theorem 4.5.1. *There is a one-to-one non-deterministic pushdown transducer that compresses a normal infinite word.*

Proof. We give a transducer that realizes the following relation. For each input word x, the output words y satisfy $T(x, y)$ where

$$\begin{array}{rlll} T(x,y) & \Leftrightarrow & (x = w_1\tilde{w}_1w_2\tilde{w}_2\cdots w_n\tilde{w}_n\cdots & , \; y = w_1\#w_2\#\cdots\#w_n\#\cdots) \quad \vee \\ & & (x = w_1\tilde{w}_1w_2\tilde{w}_2\cdots w_n\tilde{w}_nx' & , \; y = w_1\#w_2\#\cdots\#w_n\#x'). \end{array}$$

It is easy to uniquely recover x from either form of output $y = w_1\#w_2\#\cdots\#w_n\#\cdots$ or $y = w_1\#w_2\#\cdots\#w_n\#x'$, thus the relation is one-to-one. In the case where the input has the form $w_1\tilde{w}_1\cdots w_n\tilde{w}_n\cdots$, one possible output is $w_1\#\cdots\#w_n\#\cdots$. Moreover, each factor of the input $w_i\tilde{w}_i$ produces exactly the corresponding factor of the output $w_i\#$, thus leading to a compression of the input.

The transducer proceeds as follows. It guesses non-deterministically either a factorization $x = w_1\tilde{w}_1\cdots w_n\tilde{w}_n\cdots$ or a factorization $x = w_1\tilde{w}_1\cdots w_n\tilde{w}_nx'$ of the input word x. Note that the second case is just a degenerate case of the first one where w_{n+1} becomes infinite. The transducer uses two states q_0 and q_1. It is in state q_0 when it reads a factor w_i of the remaining tail and it is in state q_1 when it reads a factor $\tilde{w}_i$. In state q_0, each read symbol is output and pushed on the stack. The transducer non-deterministically either stays in state q_0 or moves to states q_1 to decide if the factor ended. An extra symbol $\#$ is output when it moves to state q_1. In state q_1, each read symbol is compared with the top symbol popped from the stack. If these two symbols do not match, the run fails. If they do match, nothing is output. The transducer moves back to state q_0 when the stack is empty. Note that the transducer is indeed real-time: an input symbol is read at each step of the run.

The complete transition table of the transducer is given below. The tuple $\langle p, s, a, v, q, \lambda\rangle$ is in δ if and only if the cell at row p and column $\langle s, a\rangle$ contains $\langle v, q, \text{pop}\rangle$. The tuple $\langle p, s, a, v, q, w\rangle$ is in δ for non-empty w if and only if that cell contains $\langle v, q, \text{push}\, w\rangle$. To ease the read of the table, a variable a is used in the columns, which can take the value 0 or 1, and $\bar{a}$ is $1 - a$.

	$\langle a, a\rangle$	$\langle a, \bar{a}\rangle$	$\langle a, \perp\rangle$
q_0	$\langle a, q_0, \text{push}\, a\rangle$	$\langle a, q_0, \text{push}\, a\rangle$	$\langle a, q_0, \text{push}\, a\rangle$
	$\langle a\#, q_1, \text{push}\, a\rangle$	$\langle a\#, q_1, \text{push}\, a\rangle$	$\langle a\#, q_1, \text{push}\, a\rangle$
q_1	$\langle \lambda, q_1, \text{pop}\rangle$		$\langle a, q_0, \text{push}\, a\rangle$
			$\langle a\#, q_1, \text{push}\, a\rangle$

□

4.5.2 Deterministic multi-pushdown

In Theorem 4.2.1 we proved that real-time transducers with any number of counters cannot compress normal infinite words. Here we show that adding a single counter to a deterministic pushdown machine gives enough power to compress a normal infinite word.

A deterministic pushdown transducer with one counter uses a stack and a counter as extra memory. Each of its transitions depends on the current state, the top symbol of the stack, the zero or non-zero status of the counter, and the input symbol. It produces an output word, a word of stack symbols that replaces the top symbol, an integer to increase or decrease the counter and the new state. Thus, transitions are given by a function $\delta : Q \times C \times \{\text{true}, \text{false}\} \times A \to B^* \times Q \times C^* \times \mathbb{Z}$.

Theorem 4.5.2. *There is a one-to-one deterministic transducer with one stack and one counter that compresses a normal infinite word.*

Proof. We give a transducer that realizes the following function. An infinite input word factorized as $w_1 w'_1 w_2 w'_2 w_3 w'_3 \cdots$ where $|w_i| = |w'_i| = 2^{i-1}$ is mapped to the output word $w_1 v_1 \# w_2 v_2 \# w_3 v_3 \# \cdots$ where $v_i = w'_i \upharpoonright \ell_i$ with ℓ_i being the length of the longest common prefix between w'_i and $\tilde{w}_i$. Since from w_i and v_i we can easily recover w_i and w'_i, the function is one-to-one. In the case where $w'_i = \tilde{w}_i$ for all i, the output for a prefix of the form $w_1 \tilde{w}_1 w_2 \tilde{w}_2 \cdots w_n \tilde{w}_n$ is $w_1 \# w_2 \# \cdots w_n$ and therefore such an input is compressible.

The required transducer uses the counter and the stack to recognize the positions where a w_i or w'_i starts. When starting to read w_i, the counter contains $|w_i| = 2^{i-1}$ and the stack is empty. While reading each symbol of w_i we decrease the counter by 1 and push it to the stack. Therefore, the counter at 0 indicates the first symbol of w'_i and $\tilde{w}_i$ is in the stack. While reading each symbol of w'_i, we pop from the stack and increase the counter by 2, so the empty stack indicates the beginning of w_{i+1} and the counter is left at 2^i. By default, we output λ while reading from w'_i. We compare the symbols read from w'_i and the top of the stack. When they mismatch for the first time, we output the current symbol and move to a state that does the same process, but outputs the current symbol instead of λ. When moving between states, we control the counter and the stack to avoid off-by-1 errors.

Notice that since the starting value of the counter is 0, we need a special starting state for processing w_1. The transducer has 4 states, a state q_0 to start, a state q_1 to read w_i and push, a state q_2 to read w'_i and pop while it coincides with the stack and a state q_3 to read the rest of w'_i, pop the rest of the stack and write, to use after a mismatch.

The complete transition table for the transducer is given below. The cell at row q and column $\langle a, s, c\rangle$ contains $\delta(q, s, c, a)$, where q is a state, a is a symbol from the input, s is the top symbol from the stack (or $\perp$ for empty stack) and c is represented as 0 for true and $\varnothing$ for false, to indicate zero and non-zero counter, respectively. The cells contain a tuple $\langle v, q, s, i\rangle$ where v is the word to be output, q is the new state, s is an instruction to perform in the stack as in the previous proof, and i is an increment to the counter. As before, to ease the read of the table, a variable a is used in the columns, which can take the value 0 or 1, and $\bar{a}$ is $1 - a$. Underscores (_) are also used to represent that any value is acceptable. Different instances of an underscore are not necessarily equal in value.

	$\langle a, \_, \_\rangle$		
q_0	$\langle a, q_1, \text{push}\, a, 0\rangle$		
	$\langle a, \_, \varnothing\rangle$	$\langle a, a, 0\rangle$	$\langle a, \bar{a}, 0\rangle$
q_1	$\langle a, q_1, \text{push}\, a, -1\rangle$	$\langle \lambda, q_2, \text{pop}, +2\rangle$	$\langle a, q_3, \text{pop}, +2\rangle$
	$\langle a, a, \_\rangle$	$\langle a, \bar{a}, \_\rangle$	$\langle a, \perp, \_\rangle$
q_2	$\langle \lambda, q_2, \text{pop}, +2\rangle$	$\langle a, q_3, \text{pop}, +2\rangle$	$\langle \# a, q_1, \text{push}\, a, -1\rangle$
	$\langle a, a, \_\rangle$	$\langle a, \bar{a}, \_\rangle$	$\langle a, \perp, \_\rangle$
q_3	$\langle a, q_3, \text{pop}, +2\rangle$	$\langle a, q_3, \text{pop}, +2\rangle$	$\langle \# a, q_1, \text{push}\, a, -1\rangle$

□

CHAPTER 5

Compressibility with two-way machines

In this chapter we study the power of compression of two-way transducers on normal infinite words. As opposed to the one-way machines considered in the previous chapter, two-way transducers have an input reading head that may be moved backwards to re-read parts of the input several times.

The introduction of two-way transducers can be traced back to the very beginning of the study of transducers [AHU69]. Two-way transducers have a nice logical characterization [EH01] and the equivalence of two of them is a decidable problem [Gur82, IK87].

The interest in these transducers have been recently renewed by the introduction of an equivalent model called streaming string transducers [AC10].

The definition of compressibility used in the previous chapter for one-way machines does not generalize in a unique way to two-way machines. A significant part of the work we do here is considering several ways in which the definition may be generalized and prove that they are all equivalent. The extent of this equivalency is stronger for deterministic two-way transducer than for non-deterministic ones. This implies that results for non-deterministic are not strict generalizations of the results for the deterministic case, which motivates us to present them separately. However, a significant part of the work done for the deterministic case is reused for the non-deterministic transducers.

The main result of this chapter is that normal infinite words are not compressible by deterministic nor by non-deterministic two-way transducers. The result for the deterministic case is presented in Section 5.1 and the result for the non-deterministic case in Section 5.2. In Section 5.3 we show with an example that the traditional way of defining compressibility collapses for two-way transducers with unbounded memory, even in its simplest form of a single counter. To deal with that case, new notions of compressibility would have to be developed. The results in this chapter are included in [CH13].

5.1 Deterministic Two-way transducers

Deterministic two-way transducers have a two-way input tape and a one-way output tape. This means that they have two independent heads: one over the input tape and another one over the output tape. The input head is two-way: it can move back and forth. On the contrary, the output head is one-way: it can only move forward. Each cell of the input tape contains a symbol of the input word. Its first cell contains the left end-marker $\vdash$ that allows the transducer to recognize that its input head is at the first cell of the tape. When the transducer reads the left end-marker, it can only move its input head to the right.

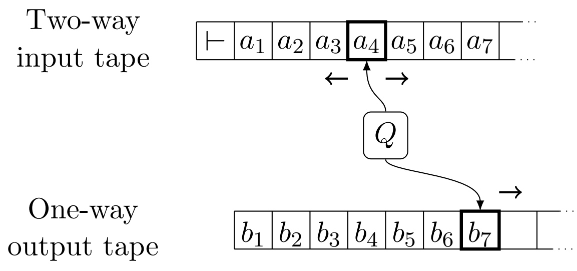
Two-way input tape
$\vdash$ a_1 a_2 a_3 a_4 a_5 a_6 a_7
$\leftarrow$ $\rightarrow$
Q
$\rightarrow$
One-way output tape
b_1 b_2 b_3 b_4 b_5 b_6 b_7

Figure 5.1: Working principle of a two-way transducer.

In order to simplify the presentation, we assume that the input and output alphabets of all transducers are the same alphabet A. All the results of the chapter can easily be extended to the setting where the output alphabet B is different from the input alphabet A. To do it, when comparing lengths of input and output words, we would just need to multiply lengths of words over A and B by the factors $\log|A|$ and $\log|B|$, respectively.

Definition. A (deterministic) *two-way transducer* is a tuple $T = \langle Q, A, \delta, q_0\rangle$, where

- Q is a finite set of states,
- A is the input and output alphabet,
- $\delta : Q \times (A \cup \{\vdash\}) \to \{\lhd, \rhd\} \times A^* \times Q$ is the transition function,
- $q_0 \in Q$ is the starting state.

such that if $\delta(p, \vdash) = \langle d, v, q\rangle$ then $d = \rhd$. The transducer T processes infinite words over A: if at state p symbol a is processed, T moves to state q, moves the reading head to the left or right depending on d and outputs v where $\langle d, v, q\rangle = \delta(p, a)$.

Let $x = a_1a_2a_3\cdots$ be a fixed infinite word over A and $a_0 = \vdash$. Whenever $\langle d, v, q\rangle = \delta(p, a_m)$, we write $\langle p, m\rangle \xrightarrow{|v} \langle q, n\rangle$ where $n = m - 1$ if $d = \lhd$ and $n = m + 1$ if $d = \rhd$. We do not write the input over the arrow because it is always the symbol below the reading head, namely, a_m. In this notation, the tuples represent the current configuration of a machine with the current state and the current position of the input head.

A *finite run* of the transducer over x is a finite sequence of consecutive transitions

$$\langle p_0, m_0\rangle \xrightarrow{|v_1} \langle p_1, m_1\rangle \xrightarrow{|v_2} \langle p_2, m_2\rangle \cdots \langle p_{n-1}, m_{n-1}\rangle \xrightarrow{|v_n} \langle p_n, m_n\rangle$$

and we write $\langle p_0, m_0\rangle \xrightarrow{|v} \langle p_n, m_n\rangle$ where $v = v_1v_2\cdots v_n$. We also refer to finite runs over words w when all positions m_i in the run but the last are between 1 and $|w|$. The last position m_n is allowed to be between 0 and $|w| + 1$. It is 0 if the run has left w on the left and it is $|w| + 1$ if it has left $|w|$ on the right.

An *infinite run* of the transducer over x is an infinite sequence of consecutive transitions

$$\langle p_0, m_0\rangle \xrightarrow{|v_1} \langle p_1, m_1\rangle \xrightarrow{|v_2} \langle p_2, m_2\rangle \xrightarrow{|v_3} \langle p_3, m_3\rangle \cdots$$

and we write $\langle p_0, m_0 \rangle \xrightarrow{|y} \infty$ where $y = v_1 v_2 v_3 \cdots$. An infinite run is accepting if $p_0 = q_0$ and $m_0 = 1$. This is the Büchi acceptance condition where all states are accepting.

The transducers that we consider in this section are deterministic: for a state p and an input symbol a, there is exactly one possible move of the transducer. It follows that for a given infinite word x, there is exactly one run over x starting at the starting state q_0. We write $T(x)$ to refer to the word output by the accepting run $\langle q_0, 1 \rangle \xrightarrow{|T(x)} \infty$ over x. The function which maps any input word x to $T(x)$ is said to be *realized* by the transducer.

We call deterministic two-way transducer simply two-way transducers and only add the modifier for the non-deterministic case.

Definition. A two-way transducer T is *bounded-to-one* if the function $x \mapsto T(x)$ is bounded-to-one.

In a run of a two-way transducer a given position in the input can be visited more than once. However, we can prove that in a bounded-to-one two-way transducer, any given position is visited a bounded number of times.

Proposition 5.1.1. *Any accepting run of a bounded-to-one two-way transducer $T = \langle Q, A, \delta, q_0 \rangle$ visits all positions of the input. Moreover, each position in the input is visited at most $|Q|$ times.*

Proof. By contradiction, let ρ be an accepting run over x that does not visit a given position n. Since the input head is moved one position at a time, ρ also does not visit any position $m \geqslant n$. Therefore, if the first n symbols of x and x' coincide, ρ is an accepting run over x' as well. Since there are infinitely many such x', T is not bounded-to-one, which contradicts the hypothesis. For the second point, if the run contains a configuration $\langle q, n \rangle$ twice, since the transducer is deterministic, the run is periodic. Therefore, the run does not visit the entire input, which contradicts the first point. □

The fact that a position can be visited multiple times introduces a difficulty for defining compressibility. We could consider the length of the output done at the first visit, or the last, or anything in between. However, we show that for normal inputs, all alternatives are equivalent. Figure 5.2 shows the positions of the input head along a possible run of a two-way transducer. In each case, the part of the run whose output is considered for each ratio is highlighted.

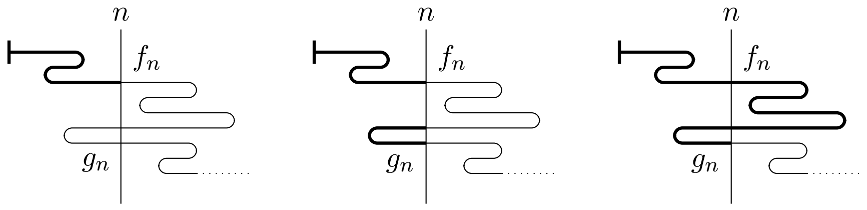

Figure 5.2: Parts of the run used for first-hit, middle and last-hit ratios.

Definition. Let $f_n = \min\{i : m_i = n\}$ and $g_n = \max\{i : m_i = n\}$ be the first and last visit of position n of the input in the accepting run ρ of a bounded-to-one two-way transducer

$$\rho = \langle q_0, 1 \rangle \xrightarrow{|v_1} \langle q_1, m_1 \rangle \xrightarrow{|v_2} \langle q_2, m_2 \rangle \xrightarrow{|v_3} \langle q_3, m_3 \rangle \cdots$$

The *first-hit ratio*, *middle ratio* and *last-hit ratio* at n are, respectively,

$$\frac{1}{n}\sum_{i\leqslant f_n}|v_{i+1}|,\quad \frac{1}{n}\sum_{m_i\leqslant n}|v_{i+1}|\quad \text{and}\quad \frac{1}{n}\sum_{i\leqslant g_n}|v_{i+1}|.$$

In each case, the ratio of ρ is the inferior limit when n goes to infinity.

Notice that the first-hit ratio is less than or equal to the middle ratio and the middle ratio is less than or equal to the last-hit ratio at every position, and also in the limit.

Theorem 5.1.2. *The first-hit, middle and last-hit ratios of the accepting run of a bounded-to-one two-way transducer over a normal infinite word coincide.*

Before proving the theorem, let us give an example showing that for non-normal inputs, the ratios might be different. Figure 5.3 shows a deterministic two-way transducer working over the binary alphabet $\{0,1\}$. Each circle with a number represents a state and an arrow from p to q labeled $a\ d\ v$ represents a transition leaving from p when reading a, moving the input head in direction d, making output v and moving to state q, that is, if δ is the transition function of the transducer, $\delta(p,a)=\langle d,v,q\rangle$. We use an arrow with multiple labels as a concise representation of multiple arrows between the same pair of states, one with each label. When for some pair p,a there is no arrow leaving from p with label starting with a, it means that $\delta(p,a)$ can be defined as any value, because it is not used.

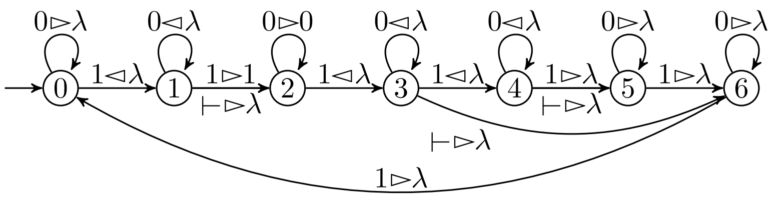

Figure 5.3: Deterministic two-way transducer with different ratios.

The depicted transducer processes the entire input, making some turns. For each 1 encountered in the input, it moves to the next 1 and back without making any output, then it does it again but copying the input while moving forward, and finally goes back to the previous 1 and forth once more without output. Since the realized function is the identity, the transducer is clearly bounded-to-one. The first of the described trips makes the first-hit ratio of a position only account for the output done before the previous 1. The trip backwards delays the last-hit, effectively making the last-hit ratio account for the output done up to the following 1. Since each position of the input is copied while reading it, the middle ratio is always 1. In an input where the distance between consecutive 1s grows fast enough, the previous observation shows that the first-hit ratio converges to 0, while the last-hit ratio diverges to $+\infty$. The positions of the input head along a run on a given input with small frequency of 1s is given in Figure 5.4, highlighting the parts of the run when the input is copied to the output.

We develop some formal tools that aid the proof of Theorem 5.1.2. First we introduce a measure of how far a run is from a one-way run, namely, how much further than position n it can go before its last visit to that position.

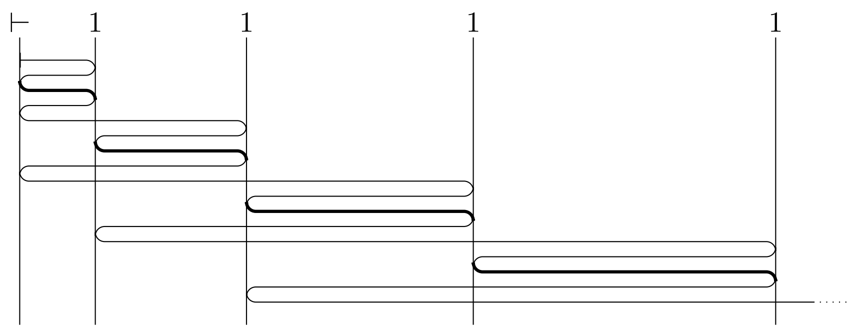

Figure 5.4: Shape of a run with different ratios.

Definition. Let $g_n = \max\{i : m_i = n\}$ be the last visit of position n in a run ρ with finitely many visits to each position

$$\rho = \langle p_0, 1\rangle \xrightarrow{|v_1} \langle p_1, m_1\rangle \xrightarrow{|v_2} \langle p_2, m_2\rangle \xrightarrow{|v_3} \langle p_3, m_3\rangle \cdots$$

The *lookahead* of ρ is the function $\Lambda(n) = \max\{m_k - n : k \leqslant g_n\}$.

Notice that $\Lambda(n) \geqslant 0$ for every n, and $\Lambda(n) = 0$ for every n if and only if the run is one-way. Also notice that the last visit of position n is always before the first visit of position $n + \Lambda(n) + 1$.

Proposition 5.1.3. *The first-hit, middle and last-hit ratios of a run with finitely many visits to each position and lookahead satisfying $\Lambda(n) = o(n)$ coincide.*

Proof. Fix the run $\langle p_0, 1\rangle \xrightarrow{|v_1} \langle p_1, m_1\rangle \xrightarrow{|v_2} \langle p_2, m_2\rangle \xrightarrow{|v_3} \langle p_3, m_3\rangle \cdots$ and let $f_n = \min\{i : m_i = n\}$ and $g_n = \max\{i : m_i = n\}$ be the first and last visits of position n. Let $\overline{\Lambda}(n) = \max_{n' \leqslant n} \Lambda(n')$ be a non-decreasing upper bound on Λ. It is clear that $\overline{\Lambda}(n) = o(n)$ and also $g_n \leqslant f_{n+\Lambda(n)+1}$ which together with the monotonicity of f and $\overline{\Lambda}$ implies $g_n \leqslant f_{n+\overline{\Lambda}(n)+1}$, and thus, $g_{n-\overline{\Lambda}(n)-1} \leqslant f_n$.

Then, for every n,

$$\frac{1}{n} \sum_{i \leqslant f_n} |v_{i+1}| \geqslant \frac{n - \overline{\Lambda}(n) - 1}{n} \left(\frac{1}{n - \overline{\Lambda}(n) - 1} \sum_{i \leqslant g_{n-\overline{\Lambda}(n)-1}} |v_{i+1}| \right).$$

Since $(n - \overline{\Lambda}(n) - 1)/n$ converges to 1, this shows that first-hit ratio is greater than or equal to the last-hit ratio, which finishes the proof. □

We want to show that the lookahead of a run of a bounded-to-one two-way transducer over a normal infinite word is $o(n)$. For that purpose, we introduce the concept of a valve. A valve of a bounded-to-one two-way transducer is a word such that a run cannot cross it from right to left.

Definition. Let $T = \langle Q, A, \delta, q_0\rangle$ be a two-way transducer. A state p is a *return state* for an input word w if and only if there is a run over w which starts at the first position of w and exits w on the left, namely,

$$\langle p, 1\rangle \xrightarrow{|v_1} \langle q_1, m_1\rangle \xrightarrow{|v_2} \langle q_2, m_2\rangle \cdots \langle q_{n-1}, m_{n-1}\rangle \xrightarrow{|v_n} \langle q_n, 0\rangle$$

where $1 \leqslant m_i \leqslant |w|$. An input word w is a *valve* for T if and only if it has a maximum number of return states.

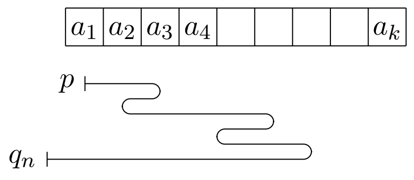

Figure 5.5: Shape of the run showing a return state p of a word $w = a_1 \cdots a_k$.

Notice that any return state of a word w is also a return state of a right extension wu of w.

Lemma 5.1.4. *Every two-way transducer has a valve.*

Proof. Immediate from the fact that the number of return states of any given word is bounded by $|Q|$. Note that as a consequence of Proposition 5.1.1, for a bounded-to-one two-way transducer this bound is actually $|Q| - 1$. □

The next lemma formalizes the main property about valves, namely, that a run cannot cross it from right to left.

Lemma 5.1.5. *Let $T = \langle Q, A, \delta, q_0 \rangle$ be a bounded-to-one two-way transducer and*

$$\langle p_0, 1 \rangle \xrightarrow{|v_1} \langle p_1, m_1 \rangle \xrightarrow{|v_2} \langle p_2, m_2 \rangle \xrightarrow{|v_3} \langle p_3, m_3 \rangle \cdots$$

be a run over an infinite word x. Let $w = x[n..n + |w| - 1]$ be a valve. Then, if $m_{j_0} \geqslant n + |w|$ for some j_0, then, for all $j \geqslant j_0$, $m_j \geqslant n$.

Proof. By contradiction, assume there is an integer j_0 such that $m_{j_0} \geqslant n + |w|$ and there is some integer $j > j_0$ for which $m_j < n$. Then, let $m = \max\{m_k : j_0 \leqslant k \leqslant j\} \geqslant n + |w|$ and consider the word $x[n..m]$. This is an extension to the right of w, so it has all the return states w has and at least one extra, namely, the one used to enter w from the left before leaving on the right at step j_0. Since by definition w has a maximum number of return states, this contradicts the hypothesis. □

Proposition 5.1.6. *The lookahead Λ of a run of a bounded-to-one two-way transducer over a normal infinite word satisfies $\Lambda(n) = o(n)$.*

Proof. By Lemma 5.1.4, let w be a valve for the transducer and x be the normal input. Let h be the function given by Lemma 2.3.5 for x and w and let n be an arbitrary position. By Lemma 2.3.5 there is an occurrence of w in the word $x[n..n + h(n)]$, and applying Lemma 5.1.5 shows that $\Lambda(n) \leqslant h(n) = o(n)$. □

We are now in conditions to give the proof for Theorem 5.1.2.

Proof of Theorem 5.1.2. Apply Proposition 5.1.1, Proposition 5.1.6 and then Proposition 5.1.3 to the accepting run. □

Also recall that non-normal infinite words are already compressible by deterministic one-way transducers [BH13]. These equivalencies together enable us to define compressible with no arbitrary choices.

Definition. An infinite word is *compressible* by a bounded-to-one two-way transducer if and only if any of its first-hit, middle or last-hit ratios is less than 1.

Theorem 5.1.7. *Normal infinite words are not compressible by bounded-to-one two-way transducers.*

To prove Theorem 5.1.7, we first introduce the technical definition of a template that we use to prove the main result of this section. A template captures the behavior of the run in a given interval of positions. Informally, the template keeps track, for each traversal of the word, of the entry state, the direction of the traversal, the output and the exit state. We start with the definition of the $\langle m, n\rangle$-factorization of a run for two positions $m \leqslant n$ of the input.

Definition. Given two positions m and n such that $m \leqslant n$ and given an accepting run, its *$\langle m, n\rangle$-factorization* is a writing of the run as

$$\langle q_0, 1\rangle \xrightarrow{|v_1} \langle q_1, m_1\rangle \xrightarrow{|v_2} \langle q_2, m_2\rangle \xrightarrow{|v_3} \langle q_3, m_3\rangle \xrightarrow{|v_4} \cdots \langle q_{2k}, m_{2k}\rangle \xrightarrow{|y} \infty,$$

where each part $\langle q_{2i+1}, m_{2i+1}\rangle \xrightarrow{|v_{2i+2}} \langle q_{2i+2}, m_{2i+2}\rangle$ reads only positions in the range $[m, n]$ and each part $\langle q_{2i}, m_{2i}\rangle \xrightarrow{|v_{2i+1}} \langle q_{2i+1}, m_{2i+1}\rangle$ reads only positions outside $[m, n]$.

Note that in the previous definition $m_{2i+1} \in \{m, n\}$ because the run enters the interval $[m, n]$, moreover, m_{2i+1} is m if the entry is from the left and n if it is from the right. Also $m_{2i+2} \in \{m-1, n+1\}$ because the run has just left the interval $[m, n]$. Moreover, m_{2i+1} is $m-1$ if the exit is to the left and $n+1$ if it is to the right.

Definition. Given the $\langle m, n\rangle$-factorization of a run,

$$\langle q_0, 1\rangle \xrightarrow{|v_1} \langle q_1, m_1\rangle \xrightarrow{|v_2} \langle q_2, m_2\rangle \xrightarrow{|v_3} \langle q_3, m_3\rangle \xrightarrow{|v_4} \cdots \langle q_{2k}, m_{2k}\rangle \xrightarrow{|y} \infty,$$

the *$\langle m, n\rangle$-template* τ of the run is a list of elements of the set $Q \times \{\lhd, \rhd\} \times A^* \times Q$ where item i in the list, starting from 0, is $\langle q_{2i+1}, d_i, v_{2i+2}, q_{2i+2}\rangle$ where $d_i = \lhd$ if and only if $m_{2i+2} = m - 1$ and $d_i = \rhd$ if and only if $m_{2i+2} = n + 1$. Its *weight* is $|\tau| = \sum_{i=1}^{n} |v_{2i}|$.

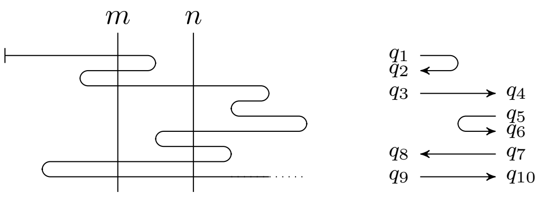

Figure 5.6: A run and its $\langle m, n\rangle$-template.

Proof of Theorem 5.1.7. Fix a normal infinite word x, a bounded-to-one two-way transducer $T = \langle Q, A, \delta, q_0\rangle$, a real $\varepsilon > 0$ and the accepting run over x

$$\langle q_0, 1\rangle \xrightarrow{|v_1} \langle q_1, m_1\rangle \xrightarrow{|v_2} \langle q_2, m_2\rangle \xrightarrow{|v_3} \langle q_3, m_3\rangle \cdots$$

Let us prove that the middle ratio is at least $(1-\varepsilon)^3$.

For each u, let $S_u = \{\tau : \tau$ is the $\langle m, n\rangle$-template of the run and $x[m..n] = u\}$. Also let $h_u = \min\{|\tau| : \tau \in S_u\}$ be the minimum number of symbols that the processing of u can contribute to the output and let $U_\ell = \{u : |u| = \ell, h_u \geqslant (1-\varepsilon)\ell\}$ be the set of words of length ℓ with relatively large contribution to the output.

Let t be an integer such that T is t-to-one. For each length ℓ and each τ that is an $\langle m, n\rangle$-template of the run for some m and n, consider the set $U' = \{u \in$

$A^\ell : \tau \in S_u\}$. Let $u_1, u_2 \in U'$ be different, and let m and n be positions such that $x[m..n] = u_1$ and the $\langle m, n\rangle$-template of the run is τ. Consider the input word $x' = x[1..m-1]u_2x[n+1..]$. Since τ is in S_{u_2} there is another accepting run over x' with the same output as the run over x. Therefore, $|U'| \leqslant t$.

Let us bound the number of different templates $\tau = (\langle p_i, d_i, v_i, q_i\rangle)_{1\leqslant i\leqslant n}$ with a given fixed list of v_is. By Proposition 5.1.1 we know that $n \leqslant |Q|$. For each item of the list, there are at most $2|Q|^2$ ways of choosing the remaining elements p_i, q_i and d_i. Thus, there are at most $(2|Q|^2)^{|Q|}$ templates for any given fixed list of v_is.

Let us bound the number of lists of at most $|Q|$ words v_i with the sum of their lengths bounded by a given integer k. Since we are interested in the asymptotic behavior of the bound, assume $k > 2|Q|$. We can consider $v = v_1v_2\cdots v_n$ the concatenation of all the words in the sequence. There are at most $|A|^{k+1}$ possible choices of v. For fixed v, there are at most $\binom{k+m-1}{m-1}$ lists of m words that produce it, so at most $|Q|\binom{k+|Q|-1}{|Q|-1}$ possible lists. Thus, there are at most $|Q|\binom{k+|Q|-1}{|Q|-1}|A|^{k+1}$ lists of at most $|Q|$ words with the sum of their lengths bounded by k.

Putting both bounds together, we can see that the number of templates of weight not greater than k is at most

$$K_k = (2|Q|^2)^{|Q|}|Q|\binom{k+|Q|-1}{|Q|-1}|A|^{k+1}.$$

By bounding the mentioned sets U', we can bound the complement of U_ℓ.

$$\begin{aligned}|\{u \in A^\ell : \tau \in S_u\}| &\leqslant t\\ |\{u \in A^\ell : \tau \in S_u, |\tau| \leqslant (1-\varepsilon)\ell\}| &\leqslant K_{\lceil(1-\varepsilon)\ell\rceil}t\\ |\{u \in A^\ell : h_u \leqslant (1-\varepsilon)\ell\} &\leqslant K_{\lceil(1-\varepsilon)\ell\rceil}t\\ |U_\ell| &\geqslant |A|^\ell - K_{\lceil(1-\varepsilon)\ell\rceil}t\end{aligned}$$

Fix ℓ such that $|U_\ell| > |A|^\ell(1-\varepsilon)$, which is possible because $K_{\lceil(1-\varepsilon)\ell\rceil}t$ is $o(|A|^\ell)$.

Let $x = u_1u_2u_3\cdots$, where each $|u_i| = \ell$. Since x is ℓ-simply normal, let n_0 be such that for all $n \geqslant n_0$, $|\{i \leqslant n : u_i = u\}| \geqslant n|A|^{-\ell}(1-\varepsilon)$. Let τ_i be the $\langle (i-1)\ell+1, i\ell\rangle$-template. Then, for all $n \geqslant n_0$,

$$\begin{aligned}\sum_{0<m_i\leqslant n\ell} |v_{i+1}| &= \sum_{i=1}^{n} |\tau_i|\\ &\geqslant \sum_{u\in A^\ell} h_u|\{i \leqslant n : u_i = u\}|\\ &\geqslant \sum_{u\in A^\ell} h_u n|A|^{-\ell}(1-\varepsilon)\\ &\geqslant \sum_{u\in U_\ell} h_u n|A|^{-\ell}(1-\varepsilon)\\ &\geqslant \sum_{u\in U_\ell} (1-\varepsilon)\ell n|A|^{-\ell}(1-\varepsilon)\\ &\geqslant |U_\ell|(1-\varepsilon)\ell n|A|^{-\ell}(1-\varepsilon)\\ &\geqslant |A|^\ell(1-\varepsilon)(1-\varepsilon)\ell n|A|^{-\ell}(1-\varepsilon) = (1-\varepsilon)^3\ell n.\end{aligned}$$

Thus,

$$\liminf_{n\to\infty} \frac{1}{n\ell} \sum_{0<m_i\leqslant n\ell} |v_{i+1}| \geqslant (1-\varepsilon)^3.$$

Since $\sum_{m_i=0} |v_{i+1}|$ is bounded and does not depend on n,

$$\liminf_{n\to\infty} \frac{1}{n\ell} \sum_{m_i \leqslant n\ell} |v_{i+1}| \geqslant (1-\varepsilon)^3,$$

and since $\sum_{n\ell<m_i<(n+1)\ell} |v_{i+1}|$ is uniformly bounded by $\ell|Q|D_T$ where D_T is the bound on the length of the outputs of all transitions of T, the middle ratio is also not less than $(1-\varepsilon)^3$. □

5.2 Non-deterministic two-way transducers

In this section we introduce non-determinism into two-way transducers. We define them with a real-time restriction, that is, disallowing transitions that do not read the input (usually called λ-transitions). However, this restriction does not affect the power level of these transducers, because a λ-transition can be simulated by moving the input head forward and changing to a new special state and then moving the input head backwards. The definition of a non-deterministic two-way transducer is very similar to the definition of a deterministic one but the transition function δ is replaced by a relation.

Definition. A *non-deterministic two-way transducer* is a tuple $T = \langle Q, A, \delta, q_0, F\rangle$, where

- Q is a finite set of states,
- A is the input and output alphabet,
- $\delta \subset Q \times (A \cup \{\vdash\}) \times \{\lhd, \rhd\} \times A^* \times Q$ is the transition relation,
- $q_0 \in Q$ is the starting state,
- $F \subseteq Q$ is the set of accepting states.

such that if $\delta(p, \vdash, d, v, q)$ then $d = \rhd$. The transducer T processes infinite words over A: if at state p symbol a is processed, T may move to state q, move the reading head to the left or right depending on d and output v where $\delta(p, a, d, v, q)$.

Let $x = a_1a_2a_3\cdots$ be a fixed infinite word over A and $a_0 = \vdash$. Whenever $\delta(p, a_m, d, v, q)$, we write $\langle p, m\rangle \xrightarrow{|v} \langle q, n\rangle$ where $n = m-1$ if $d = \lhd$ and $n = m+1$ if $d = \rhd$. Finite and infinite runs over x are defined as in the case of deterministic two-way transducers, and an infinite run is accepting if it starts at $\langle q_0, 1\rangle$ and visits infinitely often accepting states. This is the classical Büchi acceptance condition.

We write $T(x, y)$ whenever there is an accepting run $\langle q_0, 1\rangle \xrightarrow{|y} \infty$ over x. This relation is said to be *realized* by the transducer.

Definition. A non-deterministic two-way transducer T is *bounded-to-one* if and only if the function $y \mapsto |\{x : T(x, y)\}|$ is bounded.

Note that if a run visits infinitely often a given position, the last visit of a position is not defined, and middle and last-hit ratios cannot be defined as before. The following example shows that for non-deterministic two-way transducers, the analog of Proposition 5.1.1 fails. However, we prove afterwards a weaker version of it that is enough to properly define the ratios.

Figure 5.7 shows a non-deterministic two-way transducer. When the transducer is at non-accepting state 0, it may move the input head anywhere. However, in order to be an accepting run, it needs to eventually move to state 1. The only way to do that is to place the input head at the beginning of the input. While at state 1, the transducer just copies the input and moves forward. This makes the realized relation exactly the identity, so the transducer is bounded-to-one. It is clear that there is no bound on the number of visits to a given position, because while at state 0, the transducer may loop between any two or more positions. However, after fixing a run, the number of visits to each position is finite.

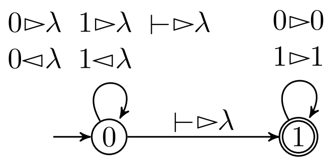
$0\rhd\lambda$ $1\rhd\lambda$ $\vdash\rhd\lambda$
$0\lhd\lambda$ $1\lhd\lambda$
$0\rhd 0$
$1\rhd 1$
0
$\vdash\rhd\lambda$
1

Figure 5.7: Non-deterministic two-way transducer with unbounded visits to some positions.

Proposition 5.2.1. *Any accepting run of a bounded-to-one non-deterministic two-way transducer visits all positions of the input. Moreover, each position in the input is visited finitely many times.*

Proof. The first point can be proved in the same way as for deterministic two-way transducers. For the second point, assume there is an accepting run over an input word x that visits a given position n infinitely often. Then, it visits some configuration $\langle q, n\rangle$ infinitely often, for some state q. Therefore, there is a run $\langle q, n\rangle \xrightarrow{|v} \langle q, n\rangle$ that visits a final state and a run $\langle q_0, 1\rangle \xrightarrow{|w} \langle q, n\rangle$. Then, there is an accepting run that visits only finitely many positions by using infinitely many times the cycle $\langle q, n\rangle \xrightarrow{|v} \langle q, n\rangle$ after starting with $\langle q_0, 1\rangle \xrightarrow{|w} \langle q, n\rangle$, which contradicts the first point. □

Proposition 5.2.1 allows us to define *first-hit*, *middle* and *last-hit* ratios exactly as in the deterministic case.

We have not been able to extend Theorem 5.1.2 to the non-deterministic case, even though we conjecture that it holds. However, we can prove a weaker equivalence between the ratios, which enables us to define compressibility using either, as before.

Theorem 5.2.2. *For any accepting run ρ of a bounded-to-one non-deterministic two-way transducer over some normal infinite word, there is another accepting run ρ' over the same input such that the first-hit, middle and last-hit ratios of ρ are not greater than the corresponding ratio of ρ', and all ratios of ρ' coincide.*

As before, we develop some tools to aid in the proof of Theorem 5.2.2.

We use the same definition for *lookahead* of a run as in the deterministic case. Recall Proposition 5.1.3 and notice that in the proof the run is fixed and the determinism of the transducer is not used, so it can easily be checked that it is also valid in the non-deterministic case.

The approach we use to prove Proposition 5.1.6 fails for the non-deterministic case. In that proof, we bound the lookahead using a function depending only on the transducer. The example already shown in Figure 5.7 illustrates that this cannot be done for non-deterministic two-way transducers. We prove a weaker version of Proposition 5.1.6, which in turn is the element that induces the need for a weaker version of Theorem 5.1.2 in the non-deterministic case.

Next we show that from a given run of a bounded-to-one non-deterministic two-way transducer we can get another one with small lookahead and without losing compression, namely, the ratios of the new run are not greater than the ratios of the original. For that purpose, we introduce the concept of a *shortcutter*. A shortcutter is a word that contains shortcuts, which are runs between a given pair of states that enter and leave the word on its right end and have small output.

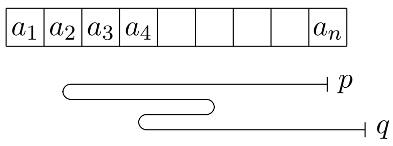

Figure 5.8: Shape of a run showing a shortcut of $w = a_1 \cdots a_n$.

Definition. Fix a non-deterministic two-way transducer $T = \langle Q, A, \delta, q_0, F\rangle$. A *shortcut* is an element of $Q \times Q \times \{\text{true}, \text{false}\} \times A^*$. A word w contains a shortcut $\langle p, q, c, v\rangle$ if and only if there is a run over w $\langle p, |w|\rangle \xrightarrow{|v} \langle q, |w| + 1\rangle$ where c is true if and only if the run visits a final state. A *shortcutter* is a word w such that any left extension vw has no better shortcuts, namely, if vw has a shortcut $\langle p, q, c, v\rangle$, then w has a shortcut $\langle p, q, c, v'\rangle$ with $|v'| \leqslant |v|$.

Lemma 5.2.3. *Every non-deterministic two-way transducer has a shortcutter.*

Proof. There are a bounded number of choices for p, q and c in the shortcut. For each such choice either there is no v such that $\langle p, q, c, v\rangle$ is a shortcut or there is a minimum one. Notice that any shortcut of w is also a shortcut of a left extension uw of w. This monotonicity proves that all minimums are among the shortcuts of some word. □

Lemma 5.2.4. *Let $T = \langle Q, A, \delta, q_0\rangle$ be a bounded-to-one non-deterministic two-way transducer and*

$$\langle q_0, 1\rangle \xrightarrow{|v_1} \langle p, n\rangle \xrightarrow{|v} \langle q, n+1\rangle \xrightarrow{|y} \infty$$

be a run over an infinite word x such that $\rho = \langle p, n\rangle \xrightarrow{|v} \langle q, n+1\rangle$ only visits positions less than or equal to n. Let $w = x[n - |w| + 1..n]$ be a shortcutter. Then, there is another run

$$\langle q_0, 1\rangle \xrightarrow{|v_1} \langle p, n\rangle \xrightarrow{|v'} \langle q, n+1\rangle \xrightarrow{|y} \infty$$

where $\rho' = \langle p, n\rangle \xrightarrow{|v'} \langle q, n+1\rangle$ visits only positions in the range $[n - |w| + 1, n]$ and $|v'| \leqslant |v|$. Furthermore, ρ visits a final state if and only if ρ' does.

Proof. Let m be the least position visited by ρ. If $m \geqslant n - |w| + 1$, then setting $\rho' = \rho$ finishes the proof. Otherwise, consider the word uw occurring from position m to position n in x. Since this is a left extension of w, by definition of shortcutter, there exists a ρ' with the desired properties. □

A consequence of $|v'| \leqslant |v|$ in the statement of the lemma, is that the first-hit ratio of the run using ρ at any position m is not greater than the first-hit ratio of the run using ρ' at m. For $m \leqslant n$, both ratios are the same, because the runs up to the first visit to position m coincide. For $m \geqslant n$, ρ' outputs an extra v', but does not output v, so the total length of the output up to m does not increase.

Proposition 5.2.5. *For any accepting run ρ of a bounded-to-one non-deterministic two-way transducer over a normal infinite word, there is another accepting run ρ' over the same input such that the first-hit ratio of ρ' at any position n is not greater than the first-hit ratio of ρ at n, and the lookahead Λ of ρ' satisfies $\Lambda(n) = o(n)$.*

Notice that we do not state any relation between ρ and ρ' with respect to the middle and last-hit ratios. However, in the proof of Theorem 5.2.2 it is clear that since all ratios become equal and the first-hit ratio is always the least of them, it is also true that the other ratios do not increase after the entire process.

Proof. Let x be the normal infinite word used as input and T be the bounded-to-one non-deterministic two-way transducer used for ρ. By Lemma 5.2.3, let w be a shortcutter for T. Applying Lemma 5.2.4 for w to ρ iteratively, we can get a run ρ' that never crosses an occurrence of w from right to left. Notice that ρ' is also accepting because each application of Lemma 5.2.4 replaces a part of the run visiting a final state with another run that also visits a final state, so for any position n if there is a visit to a final state in ρ after the first visit to position n, then the same is true for ρ'. As is remarked after Lemma 5.2.4, its application does not increase the first-hit ratio at any position. Finally, as we stated, w acts like a valve for ρ', so by applying Lemma 2.3.5 on x and w we can conclude that the Λ, the lookahead of ρ', satisfies $\Lambda(n) = o(n)$. □

Proof of Theorem 5.2.2. By hypothesis, ρ is an accepting run of a bounded-to-one non-deterministic two-way transducer. Apply Proposition 5.2.5 to get ρ'. The first-hit ratio of ρ' is not greater than the first-hit ratio of ρ, which is in turn not greater than the middle and last-hit ratios of ρ, and by Proposition 5.1.3 all ratios of ρ' are equal, then each ratio of ρ' is not greater than the corresponding ratio of ρ. □

Definition. An infinite word is *compressible* by a non-deterministic two-way transducer if and only if there is a run of the transducer over that infinite word such that any of its first-hit, middle or last-hit ratios is less than 1.

Theorem 5.2.6. *Normal infinite words are not compressible by bounded-to-one non-deterministic two-way transducers.*

In the deterministic case, we only use the fact that a run over a normal infinite word has a bounded number of visits (Proposition 5.1.1) to prove that its middle ratio is not less than 1. In the non-deterministic case, as we remarked above, this property does not hold on every run. However, in the same way we did for other properties, we can modify the run in a way that does not increase the ratios and has a bounded number of visits to each position. This enables the proof of Theorem 5.1.7 for deterministic two-way transducers to apply also to the non-deterministic case.

Lemma 5.2.7. *Given an accepting run ρ of a non-deterministic two-way transducer $T = \langle Q, A, \delta, q_0, F\rangle$ with lookahead $\Lambda(n) = o(n)$ there is another accepting run ρ' of the same transducer over the same input such that its lookahead $\Lambda'(n) = o(n)$, the last-hit ratio of ρ' at any position n is no greater than the last-hit ratio of ρ at n and ρ' visits each position at most $2|Q|$ times.*

Proof. If ρ contains a run $\langle q, n\rangle \xrightarrow{|v_1} \langle q, n\rangle \xrightarrow{|v_2} \langle q, n\rangle$ with three visits to the same state at the same position, we can replace it by just one of the two runs $\langle q, n\rangle \xrightarrow{|v_1} \langle q, n\rangle$ or $\langle q, n\rangle \xrightarrow{|v_2} \langle q, n\rangle$, keeping always one with a visit to a final state, if there is one. It is straightforward to see that this change does not increase last-hit ratio or the lookahead of any position. Applying this procedure for each position iteratively, the process converges to an accepting run ρ' that does not visit more than twice the same position at the same state. □

Proof of Theorem 5.2.6. Fix a run ρ of a bounded-to-one non-deterministic two-way transducer over a normal infinite word. Apply Proposition 5.2.5 to get a new run ρ_1 with lookahead $\Lambda_1(n) = o(n)$ and first-hit ratio no greater. By Proposition 5.1.3, ρ_1 has all ratios equal, and thus, each is no greater than the corresponding ratio of ρ. Applying Lemma 5.2.7 to ρ_1, we get a new run ρ_2 with lookahead $\Lambda_2(n) = o(n)$, last-hit ratio no greater than ρ_1 and a bounded number of visits to each position. By Proposition 5.1.3 again, ρ_2 has all ratios equal, and thus, each is not greater than the corresponding ratio of ρ_1, and therefore also of ρ. The proof of Theorem 5.1.7 applies to ρ_2, finishing the proof. □

5.3 Two-way transducers with counters

In this section we show that two-way transducers with unbounded memory, even in the limited sense of a single counter, do not admit a definition of compressibility based on the ratios we use in the previous sections.

We illustrate this by giving an example of a bounded-to-one deterministic 1-counter two-way transducer such that for any input infinite word, the middle and last-hit ratio are not even well defined, while the first-hit ratio is always strictly less than 1.

Definition. A (deterministic) 1-*counter two-way transducer* is a tuple $T = \langle Q, A, \delta, q_0\rangle$, where

- Q is a finite set of states,
- A is the input and output alphabet,
- $\delta : Q \times (A \cup \{\vdash\}) \times \{\text{true}, \text{false}\} \to \{\lhd, \rhd\} \times \mathbb{Z} \times A^* \times Q$ is the transition function,
- $q_0 \in Q$ is the starting state.

such that if $\delta(p, \vdash, z) = \langle d, i, v, q\rangle$ then $d = \rhd$. The transducer T processes infinite words over A: if at state p symbol a is processed and z represents the counter being equal to 0, T moves to state q, moves the reading head to the left or right depending on d, outputs v and adds i to the counter, where $\langle d, i, v, q\rangle = \delta(p, a, z)$.

Let $x = a_1a_2a_3\cdots$ be a fixed infinite word over A and $a_0 = \vdash$. Whenever $\langle d, i, v, q\rangle = \delta(p, a_m, z)$, we write $\langle p, m, k\rangle \xrightarrow{|v} \langle q, n, k+i\rangle$ where $n = m-1$ if $d = \lhd$, $n = m+1$ if $d = \rhd$ and $z = \text{true} \Leftrightarrow k = 0$.

Finite and infinite runs are defined the same as in two-way transducers without counters with configurations including a third variable to represent the value contained in the counter, as explained above. An infinite run is accepting if it starts with $\langle q_0, 1, 0\rangle$.

We write $T(x)$ to refer to the word output by the accepting run $\langle q_0, 1, 0\rangle \xrightarrow{|T(x)} \infty$ over x. The function which maps each input x to $T(x)$ is said to be *realized* by the transducer.

Definition. A 1-counter two-way transducer T is *bounded-to-one* if and only if the function $x \mapsto T(x)$ is bounded-to-one.

Figure 5.9 shows a deterministic 1-counter two-way transducer working over the binary alphabet $\{0, 1\}$. In this case, an arrow from p to q labeled $z, a\ d\ v, i$ represents a transition leaving from p when reading a and $z \in \{0, \varnothing\}$ is 0 if and only if the counter is 0, moving the input head in direction d, adding i to the counter, making output v and moving to state q, that is, if δ is the transition function of the transducer, $\delta(p, a, z) = \langle d, v, i, q\rangle$ where 0 represents true and $\varnothing$ represents false. The rest of the representation is as in previous examples.

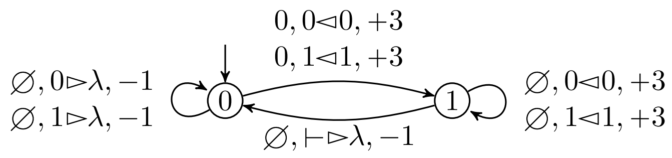
$0, 0{\lhd}0, +3$
$0, 1{\lhd}1, +3$
$\varnothing, 0{\rhd}\lambda, -1$
$\varnothing, 1{\rhd}\lambda, -1$
0
1
$\varnothing, 0{\lhd}0, +3$
$\varnothing, 1{\lhd}1, +3$
$\varnothing, \vdash{\rhd}\lambda, -1$

Figure 5.9: Deterministic 1-counter two-way transducer with bad ratios.

The depicted transducer moves the input head independently of the input. The input head starts at position 1, moves to the left-end marker and then iterates moving forward to position 3^n and back to the left-end marker, for $n = 1, 2, 3, \ldots$. State 0 represents going forward and state 1 going backwards. Change from 0 to 1 is detected by the counter being 0. Change from 1 to 0 is detected by reading the left-end marker. While going backwards, we increment the counter by 3, ensuring next time it goes forward, the transducer goes 3 times further. This movement makes infinitely many visits to all positions, which implies that there is no last visit to any position, thus making it impossible to define middle or last-hit ratios.

The transducer makes no output going forward, and copies the input when going backwards. This implies that the first visit to position 3^n the total output done is $\sum_{n'=1}^{n-1} 3^{n'} = (3^n - 1)/2$ making the first-hit ratio at any position n slightly less than 1/2, and converge in the limit to 1/2.

Figure 5.10 shows the position of the input head along the beginning of a run, highlighting the parts of the run where it copies the input.

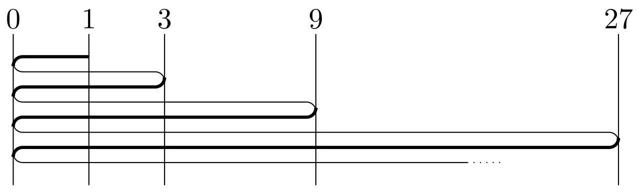

Figure 5.10: Shape of the run.

From input x the output is $\tilde{u}_1\tilde{u}_2\tilde{u}_3\cdots$ where $u_n = x[1..3^n]$ and $\tilde{u}_n$ is the word obtained by reversing u_n. It is easy to check that the realized function is thus invertible, making the transducer one-to-one.

Notice that changing the powers of 3 to powers of $b > 3$ we obtain a family of examples each with first-hit ratio $1/(b - 1)$. This shows that the first-hit ratio can be arbitrarily close to 0 while the middle and last-hit ratio are undefined.

CHAPTER 6

Selection

In this chapter we consider the preservation of normality under a family of functions called *selection* functions.

A celebrated theorem by V. N. Agafonov in 1968 describes an operation by a finite automaton that selects symbols from an infinite word by looking at its prefixes and by recognizing those that belong to a rational set. In case the word is normal, the word obtained by the selected symbols is normal as well. Agafonov published it in [Aga68], but unfortunately the proof there depends on work only available in the Russian literature. M.O'Connor [O'C88] gave another proof of Agafonov's theorem using automata predictors, and Broglio and Liardet [BL92] generalized it to arbitrary alphabets.

The first result in this chapter, Theorem 6.1.1, is an alternative proof of Agafonov's Theorem using the characterization of normality in terms of incompressibility. The presentation here improves the one we published in [BH13]. The second result, Theorem 6.1.2, complements Agafanov's theorem. We show that selection based on suffixes, as opposed to prefixes, also preserves normality. However, we show that there are simple two-sided selection rules that do not preserve normality and we exhibit one. These results are included in [BCH13].

It is known that Agafonov's theorem fails for slightly more powerful selection functions. Merkle and Reimann [MR06] showed that normality is preserved neither by selection using deterministic one-counter sets (recognized by deterministic one-counter automata) nor by selection using linear sets (recognized by one-turn pushdown automata). A characterization of the functions that preserve normality is still unknown. This is an open problem in that deserves further investigation.

The remainder of the chapter is organized as follows. Section 6.1 introduces the main definitions of the selection rules we study and formally states the mentioned results. In Sections 6.2, 6.3 and 6.4 we present the proofs for the result on prefix selection, suffix selection and two-sided selection, respectively.

6.1 Selection by rational languages

We consider the selection of symbols from an infinite word, deciding one after the other, and define a word with the selected symbols. A selection rule is a function f on finite or infinite words such that $f(x)$ is a subsequence of symbols of x. The question we tackle is which selection rules preserve normality, that is, which families of functions guarantee that $f(x)$ is normal when x is normal. Notice that if it is allowed to read the symbol being decided, it would be possible to "select only zeroes", or yield similar rules that do not preserve normality.

We call prefix selection to those functions that look only at symbols before the position being decided. We call suffix selection to those functions looking only at symbols after the position being selected. And we call two-sided selection to those looking at both sides. Our presentation of prefix selection subsumes the selection defined by Agafonov in [Aga68]. Suffix selection and two-sided selection are new.

Definition. Let $x = a_1 a_2 a_3 \cdots$ be an infinite word over alphabet A. Let $L \subseteq A^*$ be a set of finite words over A and $X \subseteq A^\omega$ a set of infinite words over A.

The word obtained by *prefix-selection* of x by L is $x \upharpoonleft L = a_{p(1)} a_{p(2)} a_{p(3)} \cdots$, where $p(j)$ is the j-th smallest integer in the set $\{i : a_1 a_2 \cdots a_{i-1} \in L\}$.

The word obtained by *suffix-selection* of x by X is $x \upharpoonright X = a_{p(1)} a_{p(2)} a_{p(3)} \cdots$, where $p(j)$ is the j-th smallest integer in the set $\{i : a_{i+1} a_{i+2} a_{i+3} \cdots \in X\}$.

To fix notation let us recall the definition of a finite automaton, and the definition of a rational set of finite or infinite words.

Definition. A *finite automaton* is a tuple $S = \langle Q, A, \delta, q_0, F \rangle$ where

- Q is the set of states,
- A is the input alphabet,
- $\delta \subseteq Q \times A \times Q$ is a finite transition relation,
- $q_0 \in Q$ is the starting state and
- $F \subseteq Q$ is the set of accepting states

The automaton is deterministic if δ is a function $Q \times A \to Q$. The automaton processes symbols as the corresponding transducer, disregarding the output. The runs and accepting runs are defined as in the case of transducers.

Definition. A set of finite words L is *rational* if there is a deterministic finite automaton $S = \langle Q, A, \delta, q_0, F \rangle$ such that $L = \{u : q \in F, q_0 \xrightarrow{u} q\}$. A set of infinite words X is rational if there is a (possibly non-deterministic) finite automaton $S = \langle Q, A, \delta, q_0, F \rangle$ such that $X = \{x : \text{there is an accepting run } q_0 \xrightarrow{x} \infty \text{ of } T\}$.

We prove Agafonov's theorem and the counterpart theorem for suffix selection. However, there are simple two-sided selection rules that do not preserve normality.

Theorem 6.1.1 (Agafonov [Aga68])**.** *Prefix selection by a rational set preserves normality.*

Theorem 6.1.2. *Suffix selection by a rational set preserves normality.*

Theorem 6.1.3. *There are extremely simple two-sided selection rules that do not preserve normality.*

To prove each of these three theorems we use the characterization of normality given by Theorem 2.5.1, which considers the limit frequency of words in a given infinite word, disregarding the positions where these words occur.

6.2 Prefix selection

We give a proof of Theorem 6.1.1. The presentation here improves the one we published in [BH13].

Lemma 6.2.1. *If there is an accepting run of a deterministic finite automaton over a normal infinite word that visits infinitely often exactly the states $\{q_1, \dots, q_n\}$, then, for each of those states q_i and word u, there is another one of those states q_j such that $q_i \xrightarrow{u} q_j$.*

Proof. Let $Q_\infty = \{q_1, \dots, q_n\}$ be the set of states visited infinitely often in the accepting run ρ. Let ρ' be a suffix of ρ such that only states in Q_∞ are visited in ρ'. Since all are visited infinitely often in ρ', it is clear that for any pair $q_i, q_j \in Q_\infty$ there is a subrun from q_i to q_j. By way of contradiction, assume the statement does not hold. Without loss of generality, assume there is u such that $q_1 \xrightarrow{u} p$ and $p \notin Q_\infty$. We build a word $u_1u_2\cdots u_n$ such that being the input to any state in Q_∞, it goes outside Q_∞. Let $u_1 = u$, so $q_1 \xrightarrow{u} p$. Inductively, consider the state p' such that $q_{i+1} \xrightarrow{u_1u_2\cdots u_i} p'$. If $p' \in Q_\infty$ then set u'_{i+1} such that $p' \xrightarrow{u'_{i+1}} q_1$ and $u_{i+1} = u'_{i+1}u$ such that $q_{i+1} \xrightarrow{u_1u_2\cdots u_iu_{i+1}} p$. If $p' \notin Q_\infty$, set $u_{i+1} = \lambda$. In both cases, we obtain $q_{i+1} \xrightarrow{u_1u_2\cdots u_iu_{i+1}} r$ with $r \notin Q_\infty$. Then, for each subrun of the form $r_1 \xrightarrow{u_1u_2\cdots u_n} r_2$, either r_1 is not in Q_∞, or $r_1 = q_i$ and $q_i \xrightarrow{u_1u_2\cdots u_i} r$ with $r \notin Q_\infty$. By normality of the input word, there are infinitely many subruns of the form $r_1 \xrightarrow{u_1u_2\cdots u_n} r_2$ in ρ'. Hence, some state not in Q_∞ is visited infinitely often in ρ', hence in ρ, contradicting the assumption. □

Lemma 6.2.2. *If $a_1a_2a_3\cdots$ is a normal infinite word and $q_0 \xrightarrow{a_1} p_1 \xrightarrow{a_2} p_2 \xrightarrow{a_3} \cdots$ is the accepting run of a deterministic finite automaton that visits infinitely often state q_1, then,*

$$\liminf_{n\to\infty} \frac{|\{i : i \leqslant n, p_i = q_1\}|}{n} > 0.$$

Proof. Let $Q_\infty = \{q_1, \dots, q_n\}$ be the set of states visited infinitely often in the accepting run ρ. Let ρ' be a suffix of ρ such that only states in Q_∞ are visited in ρ'. Since all the states in Q_∞ are visited infinitely often in ρ', it is clear that for any pair $q_i, q_j \in Q_\infty$ there is a subrun from q_i to q_j. We build a word $u_1u_2\cdots u_n$ such that when it is the input to any state in Q_∞, it visits q_1. Let $u_1 = \lambda$, so $q_1 \xrightarrow{u_1} q_1$. By Lemma 6.2.1, $q_{i+1} \xrightarrow{u_1u_2\cdots u_i} q_j$ for some j, so let u_{i+1} be such that $q_{i+1} \xrightarrow{u_1u_2\cdots u_i} q_j \xrightarrow{u_{i+1}} q_1$. Since the automaton is deterministic, each time a subrun of the form $r_1 \xrightarrow{u_1u_2\cdots u_n} r_2$ occurs in ρ', state q_1 is visited, because if $r_1 = q_i$, the prefix $q_i \xrightarrow{u_1u_2\cdots u_i} q_1$ visits q_1 by definition. By normality of the input, $u_1u_2\cdots u_n$ occurs with a fixed positive frequency ε, so q_1 is visited at least with that same minimum frequency in ρ', and therefore in ρ. □

Lemma 6.2.3. *For any set of finite words L, the function $x \mapsto \langle x \upharpoonright L, x \upharpoonright A^*\backslash L\rangle$ is one-to-one.*

Proof. Let $y_1 = x \upharpoonright L$ and $y_2 = x \upharpoonright A^*\backslash L$. By definition, y_1 contains some symbols of x, in the same relative order, and y_2 contains the complement, also in the same relative order. It is possible to reconstruct x by interleaving appropriately the symbols in y_1 and y_2. For each $i \geqslant 1$, the i-th symbol of x comes from y_1 if and only if $x \upharpoonright (i-1) \in L$. Thus, there is a unique x such that $y_1 = x \upharpoonright L$ and $y_2 = x \upharpoonright A^*\backslash L$. □

We now introduce *two-output transducers*. These are regular transducers with two output tapes instead of one. In terms of compressibility they are equivalent to regular transducers. We give the proof for the deterministic case. It is straightforward to extend it to other cases.

Definition. A (deterministic) *two-output transducer* is a tuple $T = \langle Q, A, B, \delta, q_0 \rangle$, where each element is as in deterministic transducers except the transition function $\delta : Q \times A \to B^* \times B^* \times Q$, which gives two output words.

T processes infinite words over A: if at state p symbol a is processed, T moves to state q and outputs v on tape 1 and w on tape 2, where $\langle v, w, q \rangle = \delta(p, a)$. In this case, we write $p \xrightarrow{a|v,w} q$. Finite runs, infinite runs and accepting runs are defined as for deterministic transducers. When putting together several steps, concatenation of each output is done component-wise; thus, the concatenation of the two runs

$$p_0 \xrightarrow{u_1|v_1,w_1} p_1 \quad \text{and} \quad p_1 \xrightarrow{u_2|v_2,w_2} p_2 \qquad \text{is denoted by} \qquad p_0 \xrightarrow{u_1u_2|v_1v_2,w_1w_2} p_2.$$

We write $T(x)$ to refer to the 2-tuple of infinite words such that $q_0 \xrightarrow{x|T(x)} \infty$

Definition. A two-output transducer T is *bounded-to-one* if the function $x \mapsto T(x)$ is bounded-to-one.

Definition. An infinite word $x = a_1a_2a_3 \cdots$ is *compressible* by a two-output transducer if its accepting run $q_0 \xrightarrow{a_1|v_1,w_1} q_1 \xrightarrow{a_2|v_2,w_2} q_2 \xrightarrow{a_3|v_3,w_2} q_3 \cdots$ satisfies

$$\liminf_{n\to\infty} \frac{(|v_1v_2\cdots v_n| + |w_1w_2\cdots w_n|)\log|B|}{n\log|A|} < 1.$$

Theorem 6.2.4. *An infinite word is compressible by a bounded-to-one two-output transducer if and only if it is compressible by a bounded-to-one deterministic transducer.*

Proof. The "if" part is immediate by not using one of the output tapes. For the "only if" part, let $x = a_1a_2a_3\cdots$ be an infinite word over A compressible by a bounded-to-one two-output transducer $T = \langle Q, A, B, \delta, q_0 \rangle$. The idea is to interleave both outputs in blocks of m symbols with one extra symbol before each block that identifies which output it came from. We maintain in the finite memory (the states) a queue buffer for each output tape of up to m symbols. Each time we have at least m symbols from the same tape, we output as many blocks as possible and drop the corresponding symbols from the buffer.

Let $b_1, b_2 \in B$ be different symbols. We use b_i to mark that a given output block comes from tape i. Let $O_m : B^* \times B \to B^*$ and $L_m : B^* \to B^*$, be such that, if the current buffer contains u, $O_m(u, b)$ is what is to be output, adding the extra symbol b before each block of m symbols, and $L_m(u)$ is what is left in the buffer. Namely, for $|u_i| = m$ and $|v| < m$,

$$O_m(u_1u_2\cdots u_kv, b) = bu_1bu_2\cdots bu_k \quad \text{and} \quad L_m(u_1u_2\cdots u_kv) = v.$$

For each positive integer m, let $T_m = \langle Q \times B^{<m} \times B^{<m}, A, B, \delta_m, \langle q_0, \lambda, \lambda \rangle \rangle$ be a deterministic transducer with

$$\delta_m(\langle p, v, w \rangle, a) = \langle \langle q, L_m(vv'), L_m(ww') \rangle, O_m(vv', b_1)O_m(ww', b_2) \rangle$$

where $\delta(p, a) = \langle q, v', w' \rangle$.

From an output y of T_m we can get a unique tuple of outputs y_1, y_2 of T, where y_i is the concatenation, in order, of the last m symbols of each block of $m + 1$ symbols in y that starts with b_i. Therefore, from T being bounded-to-one we can conclude each T_m is bounded-to-one. Fix an input word x and consider the accepting run of T over x,

$$q_0 \xrightarrow{a_1|v_1,w_1} q_1 \xrightarrow{a_2|v_2,w_2} q_2 \xrightarrow{a_3|v_3,w_3} q_3 \cdots$$

and the accepting run of T_m over x,

$$\langle q_0, \lambda, \lambda\rangle \xrightarrow{a_1|v_1'} C_1 \xrightarrow{a_2|v_2'} C_2 \xrightarrow{a_3|v_3'} C_3 \cdots$$

where each $C_i \in \{q_i\} \times B^{<m} \times B^{<m}$. By construction, the output of T_m has at most one extra symbol per m symbols of some output of T,

$$|v_1'v_2'\cdots v_n'| \leqslant \frac{m+1}{m}\left(|v_1v_2\cdots v_n| + |w_1w_2\cdots w_n|\right).$$

By compressibility of x,

$$\liminf_{n\to\infty} \frac{(|v_1v_2\cdots v_n| + |w_1w_2\cdots w_n|)\log|B|}{n\log|A|} < 1.$$

Then, let m be large enough such that the following inequality holds:

$$\liminf_{n\to\infty} \frac{(|v_1v_2\cdots v_n| + |w_1w_2\cdots w_n|)\log|B|}{n\log|A|}\frac{m+1}{m} < 1.$$

Together with the previous claim, this implies T_m compresses x. □

Proof of Theorem 6.1.1. Assume $x \in A^\omega$ is normal and $L \subseteq A^*$ is rational such that $x \upharpoonright L$ is infinite and not normal. By Lemma 6.2.3, $x \mapsto \langle x \upharpoonright L, x \upharpoonright A^*\backslash L\rangle$ is one-to-one. Since $x \upharpoonright L$ is not normal, by Theorem 4.1.2 there is a bounded-to-one deterministic transducer T with the same input and output alphabets that compresses it. Therefore, the function $x \mapsto \langle T(x \upharpoonright L), x \upharpoonright A^*\backslash L\rangle$ is bounded-to-one. We can compose the automaton S that accepts L with T to get a two-output transducer T' that realizes that function. It carries out S and T in parallel, sending each symbol from the input to S. If the symbol is not selected, it is output in tape 2. Else, the symbol feeds T which produces an output in tape 1.

Let $S = \langle Q_S, A, \delta_S, q_{0,S}, F\rangle$ be a deterministic automaton recognizing the rational set L and let $T = \langle Q_T, A, A, \delta_T, q_{0,T}\rangle$ be a transducer compressing $x \upharpoonright L$. The transducer T' is given by $T' = \langle Q_S \times Q_T, A, A, \delta, \langle q_{0,S}, q_{0,T}\rangle\rangle$ where

$$\begin{aligned}\delta = \{&\langle p_s, p_t\rangle \xrightarrow{a|\lambda,a} \langle q_s, p_t\rangle : p_s \notin F, p_s \xrightarrow{a} q_s\} \cup \\ \{&\langle p_s, p_t\rangle \xrightarrow{a|v,\lambda} \langle q_s, q_t\rangle : p_s \in F, p_s \xrightarrow{a} q_s, p_t \xrightarrow{a|v} q_t\}\end{aligned}$$

and $p \xrightarrow{a|v,w} q$ stands for the tuple $\langle p, a, v, w, q\rangle$.

Now, if $\langle q_{0,S}, q_{0,T}\rangle \xrightarrow{a_1|v_1,w_1} \langle q_{1,S}, q_{1,T}\rangle \xrightarrow{a_2|v_2,w_2} \langle q_{2,S}, q_{2,T}\rangle \xrightarrow{a_3|v_3,w_3} \cdots$ is the accepting run of $x = a_1a_2a_3\cdots$ on T', then $q_{0,S} \xrightarrow{a_1} q_{1,S} \xrightarrow{a_2} q_{2,S} \xrightarrow{a_3} \cdots$ is a run over S by construction. Also by construction $q_{i,S} \notin F$ implies $w_i = a_i$ and $v_i = \lambda$ and $q_{i,S} \in F$ implies $w_i = \lambda$. Then,

$$\liminf_{n\to\infty} \frac{(|v_1v_2\cdots v_n| + |w_1w_2\cdots w_n|)\log|A|}{n\log|A|} = \liminf_{n\to\infty} \frac{\sum_{i\leqslant n: q_{i,S}\in F}|v_i| + \sum_{i\leqslant n: q_{i,S}\notin F} 1}{n}$$

By $x \upharpoonright L$ being infinite and Lemma 6.2.2, there is $\varepsilon > 0$ such that

$$\liminf_{n\to\infty} |\{i \leqslant n : q_{i,S} \in F\}|/n = \varepsilon.$$

Let $n_1, n_2, \ldots$ be an increasing sequence such that

$$\lim_{j\to\infty} |\{i \leqslant n_j : q_{i,S} \in F\}|/n_j = \varepsilon$$

and apply it to the inferior limit above, getting

$$\liminf_{n\to\infty} \frac{\sum_{i\leqslant n: q_{i,S}\in F} |v_i| + \sum_{i\leqslant n: q_{i,S}\notin F} 1}{n} \leqslant \liminf_{j\to\infty} \frac{\sum_{i\leqslant n_j: q_{i,S}\in F} |v_i| + \sum_{i\leqslant n_j: q_{i,S}\notin F} 1}{n_j}$$

Since $\lim_{j\to\infty} \sum_{i\leqslant n_j: q_{i,S}\notin F} 1/n_j = (1-\varepsilon)$ and $z = \lim_{j\to\infty} \frac{\sum_{i\leqslant n_j: q_{i,S}\in F} |v_i|}{\varepsilon n_j} < 1$ because T compresses $x \upharpoonright L$, we get

$$\begin{aligned}\liminf_{j\to\infty} \frac{\sum_{i\leqslant n_j: q_{i,S}\in F} |v_i| + \sum_{i\leqslant n_j: q_{i,S}\notin F} 1}{n_j} &\leqslant \liminf_{j\to\infty} \frac{\varepsilon n_j z + (1-\varepsilon) n_j}{n_j} \\ &= \varepsilon z + (1-\varepsilon) \\ &< 1.\end{aligned}$$

Putting all the inequalities above together yields,

$$\liminf_{n\to\infty} \frac{(|v_1 v_2 \cdots v_n| + |w_1 w_2 \cdots w_n|)\log |A|}{n \log |A|} < 1,$$

which means T' compresses x. But by Theorem 6.2.4 and Theorem 4.1.4, this is impossible. Therefore, the assumption that $x \upharpoonright L$ is not normal must be false. □

6.3 Suffix selection

The proof of Theorem 6.1.2 is similar to the one for prefix selection, but it has additional subtleties. Since the automaton that recognizes rational sets of infinite words is not necessarily deterministic, we need variants of Lemmas 6.2.1 and 6.2.2. To do so we use the characterization of rational sets of infinite words that provides co-determinism instead of determinism given by Carton and Michel [CM03].

Definition. A *Büchi automaton* is a tuple $S = \langle Q, A, \delta, Q_0, F\rangle$ where

- Q is the set of states,
- A is the input alphabet,
- $\delta \subseteq Q \times A \times Q$ is a finite transition relation,
- $Q_0 \subseteq Q$ is the set of starting states
- $F \subseteq Q$ is the set of accepting states

The processing of the input symbols and the definition of the run coincides with those for a non-deterministic automaton. A run is accepting if it starts at *any* of the starting states and visits an accepting state infinitely often.

Note that we allow Büchi automata to have several starting states. This needed is by Theorem 6.3.1.

Theorem 6.3.1 (Carton and Michel [CM03]). *Any rational set of infinite words is accepted by a prophetic automaton.*

Proposition 6.3.2 ([CM03]). *A prophetic automaton is co-deterministic. That is, if q is a state in at least one infinite run and a is a symbol, there is exactly one state p such that $p \xrightarrow{a} q$.*

Proof. Since q is a state in at least one infinite run assume $q \xrightarrow{x} \infty$. Let $p \xrightarrow{ax} \infty$ be the only run over ax. Since it contains as a suffix a run over x, and there is only one such run, the second state in the run must be q, and thus $p \xrightarrow{a} q$. By way of contradiction, assume $p \xrightarrow{a} q$ and $p' \xrightarrow{a} q$. Then, $p \xrightarrow{ax} \infty$ and $p' \xrightarrow{ax} \infty$, contradicting the condition of the prophetic automaton. □

The next lemmas do the job of Lemmas 6.2.1 and 6.2.2 in the opposite direction. The proofs are almost the same as the ones of Lemmas 6.2.1 and 6.2.2 but using the fact that prophetic automata are co-deterministic instead of deterministic.

Lemma 6.3.3. *If there is an accepting run of a prophetic automaton over a normal infinite word that visits infinitely often exactly the states $\{q_1, \dots, q_n\}$, then, for each of those states q_j and each word u, there is another one of those states q_i such that $q_i \xrightarrow{u} q_j$.*

Proof. Let $Q_\infty = \{q_1, \dots, q_n\}$ be the set of states visited infinitely often in the accepting run ρ. Let ρ' be a suffix of ρ such that only states in Q_∞ are visited in ρ'. Since all are visited infinitely often in ρ', it is clear that for any pair $q_i, q_j \in Q_\infty$ there is a subrun from q_i to q_j. By way of contradiction, assume the statement does not hold, and without loss of generality, assume there is u such that $p \xrightarrow{u} q_1$ and $p \notin Q_\infty$. We build a word $u_n u_{n-1} \cdots u_1$ such that finishing its process in any state in Q_∞, it ensures a visit to a state outside Q_∞. Let $u_1 = u$, so $p \xrightarrow{u_1} q_1$. Inductively, consider the only state p' such that $p' \xrightarrow{u_i u_{i-1} \cdots u_2 u_1} q_{i+1}$. If $p' \in Q_\infty$ then set u'_{i+1} such that $q_1 \xrightarrow{u'_{i+1}} p'$ and $u_{i+1} = uu'_{i+1}$ such that $p \xrightarrow{u_{i+1} u_i \cdots u_2 u_1} q_{i+1}$. If $p' \notin Q_\infty$, set $u_{i+1} = \lambda$. In both cases, we obtain $r \xrightarrow{u_{i+1} u_i \cdots u_2 u_1} q_{i+1}$ with $r \notin Q_\infty$. Thus, each time there is a subrun $r_1 \xrightarrow{u_n u_{n-1} \cdots u_2 u_1} r_2$, either r_2 is not in Q_∞, or $r_2 = q_i$ and there is a prefix of the subrun $r \xrightarrow{u_i u_{i-1} \cdots u_2 u_1} r_2$ with $r \notin Q_\infty$. By normality of the input word, there are infinitely many subruns of the form $r_1 \xrightarrow{u_n u_{n-1} \cdots u_2 u_1} r_2$ in ρ'. Then, some state not in Q_∞ is visited infinitely often in ρ', and therefore in ρ, contradicting the assumption. □

Lemma 6.3.4. *If $a_1 a_2 a_3 \cdots$ is a normal infinite word and $p_0 \xrightarrow{a_1} p_1 \xrightarrow{a_2} p_2 \xrightarrow{a_3} \cdots$ is the accepting run of a prophetic automaton that visits infinitely often state q_1, then,*

$$\liminf_{n\to\infty} \frac{|\{i : 1 \leqslant i \leqslant n, p_i = q_1\}|}{n} > 0.$$

Proof. Let $Q_\infty = \{q_1, \dots, q_n\}$ be the set of states visited infinitely often in the mentioned run ρ. Let ρ' be a suffix of ρ such that only states in Q_∞ are visited in ρ'. Since all are visited infinitely often in ρ', it is clear that for any pair $q_i, q_j \in Q_\infty$ there is a subrun from q_i to q_j. We build a word $u_n u_{n-1} \cdots u_1$ such that finishing its process in any state in Q_∞, it ensures a visit to q_1. Let $u_1 = \lambda$, so $q_1 \xrightarrow{u_1} q_1$. By Lemma 6.3.3, $q_j \xrightarrow{u_i u_{i-1} \cdots u_2 u_1} q_{i+1}$ for some j, so let u_{i+1} be such that $q_1 \xrightarrow{u_{i+1}} q_j \xrightarrow{u_i u_{i-1} \cdots u_2 u_1} q_{i+1}$. Then, since the automaton is co-deterministic, each time $r_1 \xrightarrow{u_n u_{n-1} \cdots u_2 u_1} r_2$ occurs in ρ', state q_1 is visited, because if $r_2 = q_i$, the suffix $q_1 \xrightarrow{u_i u_{i-1} \cdots u_2 u_1} r_2$ visits q_1 by definition. By normality of the input, $u_n u_{n-1} \cdots u_2 u_1$ occurs with a fixed positive frequency ε, so q_1 is visited at least with that minimum frequency on ρ', and therefore on ρ. □

When selecting with a rational set of infinite words, we need to check a prophetic automaton. Notice that the unique run for the input x contains as suffixes the unique runs for all the suffixes of x. Moreover, the run over x visits accepting states infinitely often if and only if so does each of the runs over the suffixes of x. Thus, a symbol is selected according to the state at which prophetic automaton arrives after processing it. This mirrors of prefix selection by a finite automata, where a symbol is selected according to the state that the automaton leaves before processing it.

It is straightforward to generalize two-output transducers to make them non-deterministic and get an analogous of Theorem 6.2.4. However Lemma 6.2.3 cannot be used because it is based on the determinism of the automaton recognizing the rational set. A mirror Lemma 6.2.3 would need a finishing state, which does not exist for infinite runs. Moreover, functions of the form $x \mapsto \langle x \upharpoonleft X, x \upharpoonleft A^\omega\backslash X\rangle$ are not necessarily bounded-to-one, as the following example illustrates. Consider $A = \{0, 1\}$ and $X = 01A^\omega$ the set of binary infinite words that start with 01. The selection rule induced by X is then the following: select symbol at position i if and only if, the two symbols at positions $i + 1$ and $i + 2$ are 0 and 1, respectively. Consider words $x = 000 \cdots 000010101 \cdots$ that start with a positive number of zeroes, and then alternate zeroes and ones. It is clear that all of them get mapped by $x \mapsto \langle x \upharpoonleft X, x \upharpoonleft A^\omega\backslash X\rangle$ to $\langle 0111 \cdots , 000 \cdots\rangle = \langle 01^\omega, 0^\omega\rangle$.

To solve the last problem we insert the current state once in a while in the output in a predictable way. We also add a way to synchronize the two outputs, so that for each state we can calculate the two prefixes that were output up to that point. Then, the whole splitting process is one-to-one, because from a finishing state and two finite words we can uniquely recover the originating word by doing the same as in the proof of Lemma 6.2.3, but from right to left.

To add the identification for the state and the synchronization, we need to get inside the two-output merging done in the proof of Theorem 6.2.4, so each time we insert a block from one of the tapes, we also write the current state and the number of symbols left in the buffer of the other tape. This increases the overhead of the interleaving from 1 symbol per m symbols of input to $k + \lceil\log m\rceil$ symbols per m symbols of input, for some constant k required to encode the states. However, $(k + \lceil\log m\rceil)/m$ is also arbitrarily close to 0, so compressibility is maintained in the same way as in the proof of Theorem 6.2.4.

Proof of Theorem 6.1.2. Assume $x \in A^\omega$ is normal and X is a rational set of infinite words. By Theorem 6.3.1, assume X is recognized by the prophetic automaton $S = \langle Q_S, A, \delta_S, Q_{0,S}, F\rangle$. Let $q_0 \xrightarrow{a_1} q_1 \xrightarrow{a_2} q_2 \xrightarrow{a_3} \cdots$ be the unique run of S over $x = a_1a_2a_3 \cdots$. Note that the suffix of the run $q_i \xrightarrow{a_{i+1}} q_{i+1} \xrightarrow{a_{i+2}} q_{i+2} \xrightarrow{a_{i+3}} \cdots$ is the unique run of S over the suffix of the input $a_{i+1}a_{i+2}a_{i+3} \cdots$. If finitely many of the q_i are in F, then none of these runs is accepting and $x \upharpoonleft X$ is empty, and thus, finite. From now on, assume infinitely many of the q_i are in F. Symbol a_i is selected if and only if the run $q_i \xrightarrow{a_{i+1}} q_{i+1} \xrightarrow{a_{i+2}} q_{i+2} \xrightarrow{a_{i+3}} \cdots$ is accepting. Since we know it visits an accepting state infinitely often, the only additional condition is that $q_i \in Q_{0,S}$.

Assume $x \upharpoonleft X$ is not normal and by Theorem 4.1.2, let $T = \langle Q_T, A, A, \delta_T, q_{0,T}\rangle$ be a bounded-to-one deterministic transducer that compresses it. We build a family of bounded-to-one non-deterministic transducers T_m where each T_m simulates S and splits the output into the selected and non-selected parts x_1 and x_2, passes x_1 through T and then merges $T(x_1)$ and x_2 in blocks of m digits as in the proof of Theorem 6.2.4. While merging, it adds to blocks from $T(x_1)$ an indicator of the state of S where the automaton is standing, and the number of digits left in the buffer of x_2. This allows to recover, for infinitely many prefixes of each of the bounded possibilities of x_1, a corresponding prefix of x_2 and a finishing state. These, in turn, imply that the originating prefix of x is unique, making the construction bounded-to-one overall.

As in the proof of Theorem 6.2.4, let b_1 and b_2 be two different symbols of the output alphabet A. Let $(q_S)_{b_1,b_2}$ for a state $q_S \in Q_S$ be an injective codification of the states as integers in the range $[0, |Q_S|-1]$, written with exactly $\lceil \log |Q_S| \rceil$ binary digits. Let $(k)_{b_1,b_2}$, for k an integer between 0 and $m-1$, be k written in binary with exactly $\lceil \log m \rceil$ binary digits. To write binary digits over alphabet A, we use b_{i+1} to represent digit i. Let $O_m : A^* \times A^* \to A^*$ and $L_m : A^* \to A^*$ be as follows, for $|u_i| = m$ and $|v| < m$,

$$O_m(u_1u_2\cdots u_kv, w) = wu_1wu_2\cdots wu_k \quad \text{and} \quad L_m(u_1u_2\cdots u_kv) = v.$$

Observe that in this case O_m can place any word before each block and not just a single symbol. Let

$$T_m = \langle Q_S \times Q_T \times A^{<m} \times A^{<m}, \delta_m, \langle q_0, q_{0,T}, \lambda, \lambda\rangle, F \times Q_T \times A^{<m} \times A^{<m}\rangle,$$

where δ_m is the set of all transitions

$$\langle p_S, p_T, v, w\rangle \xrightarrow{a|O_m(vv', b_1(q_S)_{b_1,b_2}(|w|)_{b_1,b_2})} \langle q_S, q_T, L_m(vv'), w\rangle$$

such that $p_S \xrightarrow{a} q_S$, $q_S \in Q_{0,S}$ and $p_T \xrightarrow{a|v'} q_T$, together with the set of all transitions

$$\langle p_S, p_T, v, w\rangle \xrightarrow{a|O_m(wa,b_2)} \langle q_S, p_T, v, L_m(wa)\rangle$$

such that $p_S \xrightarrow{a} q_S$ and $q_S \notin Q_{0,S}$.

Thus, δ_m is defined as the union of two disjoint cases, the case where the current symbol is selected and the case when it is not. Since T is deterministic and the way the buffers behave encoded in L_m and O_m is also deterministic, each transition of T_m is associated in a bijective way with a transition of S. Consequently, each run of T_m is associated with a unique run of S, and thus, there is exactly one run over each input. Moreover, the unique run of T_m over x is associated to the unique run of S over x, and it has the form

$$\langle q_0, p_0, \lambda, \lambda\rangle \xrightarrow{a_1|v_1} \langle q_1, p_1, \ldots\rangle \xrightarrow{a_2|v_2} \langle q_2, p_2, \ldots\rangle \xrightarrow{a_3|v_3} \cdots$$

where the ... in the configurations represent the content of the buffers, which we do not record. Since infinitely many of the q_i are in F, by definition of the accepting states of T_m, infinitely many of the $\langle q_i, p_i, \ldots\rangle$ in the given run are accepting. Since $\langle q_0, p_0, \lambda, \lambda\rangle$ is the starting state, the given run is an accepting run of T_m over x.

The output of T_m contains an overhead of at most $k_m = 1 + \lceil \log |Q_S| \rceil + \lceil \log m \rceil$ extra symbols per m symbols of the input, while outputting exactly the input on one case (the non-selecting) and a word asymptotically shorter than the input on the other case (selecting). By Lemma 6.3.4 and the fact that $(m + k_m)/m$ is arbitrarily close to 1 for m large enough, it is straightforward to combine the reasoning in the proof of Theorem 6.2.4 and in the last part of the proof of Theorem 6.1.1 to show that the given run compresses the input for m large enough. Since it is an accepting run, T_m for m large enough compresses x.

To see that each T_m is bounded-to-one, assume we are given the output y and let us show that there are bounded number of possible inputs that produce it. First, parse y into blocks of $m + 1$ or $m + 1 + \lceil \log |Q_S| \rceil + \lceil \log m \rceil$ symbols, where the length of each block is simply determined by its first symbol being b_1 or b_2. This uniquely reconstructs $T(x_1)$ and x_2 as mentioned above. There are a bounded number of possible x_1 because T is bounded-to-one. Fix one. Each block of $T(x_1)$ gives enough information (a pair of a state q and an integer k_1) to associate a given prefix u_1 of x_1 (the shortest prefix that produces a prefix of $T(x_1)$ to fill that many blocks of output) to a unique prefix u_2 of x_2 (if k_2 is the number of finished blocks of x_2 before u_1 is processed and k_1 is the informed integer, u_2 is exactly the first $mk_2 + k_1$ symbols of x_2) and a state of S. Since S is co-deterministic, u_1 and u_2 can be merged into a unique u, which is a prefix of the original input. This can be done for arbitrarily large prefixes, because we have the extra information infinitely often, which uniquely determines an input. In conclusion, if T is t-to-one, each T_m is also t-to-one.

We showed the existence of a bounded-to-one non-deterministic transducer that compresses a normal input x. This contradicts Theorem 4.3.1. Therefore, the assumption that $x \upharpoonright X$ is not normal must be false. □

6.4 Two-sided selection

We show that a selection rule that looks at both sides of the symbol being decided, in general, does not preserve normality.

Proof of Theorem 6.1.3. Let $x = a_1a_2a_3\cdots$ be a normal infinite word over $\{0,1\}$ and let y be the result of selecting all symbols between two zeroes, formally, let $y = a_{p(1)}a_{p(2)}a_{p(3)}\cdots$ where $p(j)$ is the j-th smallest integer in $\{i : a_{i-1} = a_{i+1} = 0\}$. We show that y is not normal.

Let m_n be the length of the shortest prefix of x that contains n instances of either 000 or 010,

$$m_n = \min\{m : |\{i : 2 \leqslant i \leqslant m-1, a_{i-1} = a_{i+1} = 0\}| = n\}.$$

By normality of x, infinitely many symbols are selected. So, each m_n is well defined. Let $y = b_1b_2b_3\cdots$ and $k_n = |\{i : 1 \leqslant i \leqslant n-1, b_ib_{i+1} = 00\}|$. By definition,

$$\begin{aligned} k_n \geqslant &|\{i : 1 \leqslant i \leqslant m_n - 3, a_ia_{i+1}a_{i+2}a_{i+3} = 0000\}| + \\ &|\{i : 1 \leqslant i \leqslant m_n - 7, a_ia_{i+1}a_{i+2}a_{i+3}a_{i+4}a_{i+5}a_{i+6} = 0001000\}|. \end{aligned}$$

Therefore,

$$\begin{aligned} \lim_{n\to\infty} \frac{k_n}{n} &\geqslant \lim_{n\to\infty} \frac{|\{i : 1 \leqslant i \leqslant m_n - 3, a_ia_{i+1}a_{i+2}a_{i+3} = 0000\}|}{n} + \\ &\qquad \frac{|\{i : 1 \leqslant i \leqslant m_n - 7, a_ia_{i+1}a_{i+2}a_{i+3}a_{i+4}a_{i+5}a_{i+6} = 0001000\}|}{n} \\ &> \lim_{n\to\infty} \frac{|\{i : 1 \leqslant i \leqslant m_n - 3, a_ia_{i+1}a_{i+2}a_{i+3} = 0000\}|}{n} \\ &> \lim_{n\to\infty} \frac{|\{i : 1 \leqslant i \leqslant m_n - 3, a_ia_{i+1}a_{i+2}a_{i+3} = 0000\}|}{m_n} \frac{m_n}{n}. \end{aligned}$$

By definition of normality and the properties of limit,

$$\lim_{n\to\infty} \frac{|\{i : 1 \leqslant i \leqslant m_n - 3, a_ia_{i+1}a_{i+2}a_{i+3} = 0000\}|}{m_n} = \frac{1}{2^4} \quad \text{and} \quad \lim_{n\to\infty} \frac{m_n}{n} = 2^2.$$

This proves $\lim_{n\to\infty} k_n/n > 2^{-4}\ 2^2 = 1/4$. which implies y is not normal. □

The proof above shows the frequency of the word 00 inside y is not the expected 1/4. This suffices to prove that normality is not preserved by this selection rule. A longer and more precise calculation can show that the frequency is exactly 3/10.

Bibliography

[AC10] R. Alur and P. Cerný. Expressiveness of streaming string transducers. In *FSTTCS*, volume 8 of *LIPIcs*, pages 1–12. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2010.

[Aga68] V. N. Agafonov. Normal sequences and finite automata. *Soviet Mathematics Doklady*, 9:324–325, 1968.

[AHU69] A. V. Aho, J. E. Hopcroft, and J. D. Ullman. A general theory of translation. *Mathematical Systems Theory*, 3:193–221, 1969.

[BC01] David H. Bailey and Richard E. Crandall. On the random character of fundamental constant expansions. *Experimental Mathematics*, 10(2):175–190, 2001.

[BCH13] Verónica Becher, Olivier Carton, and Pablo Ariel Heiber. Normality and automata. Preprint, 2013.

[BEHS13] Verónica Becher, Martin Epszteyn, Pablo Ariel Heiber, and Theodore A. Slaman. Efficiently computing an absolutely normal number. Preprint, 2013.

[Ber13] J. Bernoulli. *Ars conjectandi.* Landmarks of science. Impensis Thurnisiorum, fratrum, 1713.

[BF02] Verónica Becher and Santiago Figueira. An example of a computable absolutely normal number. *Theoretical Computer Science*, 270(1-2):947–958, 2002.

[BFP07] Verónica Becher, Santiago Figueira, and Rafael Picchi. Turing's unpublished algorithm for normal numbers. *Theoretical Computer Science*, 377(1-3):126–138, 2007.

[BH11] Verónica Becher and Pablo Ariel Heiber. On extending de bruijn sequences. *Information Processing Letters*, 111(18):930–932, 2011.

[BH13] Verónica Becher and Pablo Ariel Heiber. Normal numbers and finite automata. *Theoretical Computer Science*, 477:109–116, 2013.

[BHS13a] Verónica Becher, Pablo Ariel Heiber, and Theodore A. Slaman. Normal numbers and the borel hierarchy. *Fundamenta Matemathicae*, 2013. In press.

[BHS13b] Verónica Becher, Pablo Ariel Heiber, and Theodore A. Slaman. A polynomial-time algorithm for computing absolutely normal numbers. *Information and computation*, 232:1–9, 2013.

[BHS14] Verónica Becher, Pablo Ariel Heiber, and Theodore A. Slaman. A computable absolutely normal Liouville number. Preprint, 2014.

[BHV05] C. Bourke, J. Hitchcock, and N. V. Vinodchandran. Entropy rates and finite-state dimension. *Theoretical Computer Science*, 349:392–406, 2005.

[BL92] A. Broglio and P. Liardet. Predictions with automata. symbolic dynamics and its applications. *Contemporary Mathematics*, 135:111–124, 1992. Also in Proceedings AMS Conference in honor of R. L. Adler. New Haven CT - USA 1991.

[Bor09] Émile Borel. Les probabilités dénombrables et leurs applications arithmétiques. *Rendiconti del Circolo Matematico di Palermo*, 27:247–271, 1909.

[BP07] Jean Berstel and Dominique Perrin. The origins of combinatorics on words. *European Journal of Combinatorics*, 28(3):996–1022, 2007.

[Bru46] N. G. Bruijn. A combinatorial problem. *Koninklijke Nederlandse Akademie v. Wetenschappen*, 49:758–764, 1946.

[Bug02] Yann Bugeaud. Nombres de liouville et nombres normaux. *Comptes Rendus Mathematique*, 335(2):117–120, 2002.

[Bug12] Yann Bugeaud. *Distribution Modulo One and Diophantine Approximation.* Number 193 in Cambridge Tracts in Mathematics. Cambridge University Press, Cambridge, UK, 2012.

[CE46] Arthur H. Copeland and Paul Erdös. Note on normal numbers. *Bulletin American Mathematical Society*, 52:857–860, 1946.

[CH13] Olivier Carton and Pablo Ariel Heiber. Normality and two-way automata. Preprint, 2013.

[Cha33] D. G. Champernowne. The construction of decimals normal in the scale of ten. *Journal of the London Mathematical Society*, 8:254–260, 1933.

[Cha75] Gregory J. Chaitin. A theory of program size formally identical to information theory. *Journal of the ACM*, 22:329–340, 1975.

[Cha87] Gregory Chaitin. *Algorithmic Information Theory.* Cambridge Tracts in Theoretical Computer Science, Cambridge University Press, 1987.

[CM03] O. Carton and M. Michel. Unambiguous Büchi automata. *Theoretical Computer Science*, 297:37–81, 2003.

[CSRL01] Thomas H. Cormen, Clifford Stein, Ronald L. Rivest, and Charles E. Leiserson. *Introduction to Algorithms.* McGraw-Hill Higher Education, 2nd edition, 2001.

[DH10] Rodney G. Downey and Denis R. Hirschfeldt. *Algorithmic Randomness and Complexity.* Springer, 2010.

[DLLM04] J. Dai, J. Lathrop, J. Lutz, and E. Mayordomo. Finite-state dimension. *Theoretical Computer Science*, 310:1–33, 2004.

[EH01] J. Engelfriet and H. J. Hoogeboom. MSO definable string transductions and two-way finite-state transducers. *ACM Transactions on Computational Logic*, 2(2):216–254, 2001.

[Fle83] M. Fleury. Deux problemes de geometrie de situation. *Journal de mathematiques elementaires*, pages 257–261, 1883.

[FN13] Santiago Figueira and André Nies. Feasible analysis, randomness, and base invariance. *Theory of Computing Systems*, 2013. In press.

[Goo46] I. J. Good. Normal Recurring Decimals. *J. London Math. Soc.*, s1-21(3):167–169, July 1946.

[Gur82] E. M. Gurari. The equivalence problem for deterministic two-way sequential transducers is decidable. *SIAM Journal on Computing*, 11(3):448–452, 1982.

[Huf52] D. Huffman. A method for the construction of minimum-redundancy codes. In *Institute of Radio Engineers*, pages 1098–1102, 1952.

[HW60] G. H. Hardy and Edward Maitland Wright. *An introduction to the theory of numbers / by G.H. Hardy and E.M. Wright.* Clarendon Press Oxford, 4th ed., 2nd (corr.) impression. edition, 1960.

[IK87] K. Culik II and J. Karhumäki. The equivalence problem for single-valued two-way transducers (on NPDT0L languages) is decidable. *SIAM Journal on Computing*, 16(2):221–230, 1987.

[Kec95] Alexander S. Kechris. *Classical descriptive set theory*, volume 156 of *Graduate Texts in Mathematics.* Springer-Verlag, New York, 1995.

[KL94] Haseo Ki and Tom Linton. Normal numbers and subsets of $\mathbb{N}$ with given densities. *Fundamenta Mathematica*, 144(2):163–179, 1994.

[Kol65] A. N. Kolmogorov. Three approaches to the quantitative definition of information. *Problems of Information Transmission*, 1(1):1–7, 1965.

[Leb02] H. Lebesgue. *Intégrale, longueur, aire.* PhD thesis, Université de Paris, 1902.

[Leb17] Henri Lebesgue. Sur certaines démonstrations d'existence. *Bulletin de la Société Mathématique de France*, 45:132–144, 1917.

[Lem70] Abraham Lempel. On a homomorphism of the de Bruijn Graph and its applications to the design of feedback shift registers. *IEEE Transactions on Computers*, C-19:204–1209, 1970.

[Lev73] L. A. Levin. On the notion of a random sequence. *Soviet Math. Dokl.*, 14(5):1413–1416, 1973.

[Mar02] David Marker. Descriptive set theory. Course Notes, 2002.

[May13] Elvira Mayordomo. Construction of an absolutely normal real number in polynomial time. Preprint, 2013.

[Min61] M. L. Minsky. Recursive unsolvability of posts problem of ”tag” and other topics in the theory of turing machines. *Annals of Math*, 74:437–454, 1961.

[ML66] P. Martin-Löf. The definition of random sequences. *Information and Control*, 9(6):602–619, 1966.

[MR06] W. Merkle and J. Reimann. Selection functions that do not preserve normality. *Theory of Computing Systems*, 39(5):685–697, 2006.

[Nie09] André Nies. *Computability and Randomness.* Oxford Logic Guides. Oxford Science Publications, 2009.

[NV12] Satyadev Nandakumar and Santhosh Kumar Vangapelli. Normality and finite-state dimension of liouville numbers. *CoRR*, 2012.

[O'C88] M. G. O'Connor. An unpredictability approach to finite-state randomness. *Journal of Computer and System Sciences*, 37(3):324–336, 1988.

[Pop59] Karl Popper. *The logic of scientific discovery.* Hutchinson, 1959.

[PP04] Dominique Perrin and Jean-Éric Pin. *Infinite Words.* Elsevier, 2004.

[Rog87] Hartley Rogers, Jr. *Theory of recursive functions and effective computability.* MIT Press, Cambridge, MA, second edition, 1987.

[Sak09] J. Sakarovitch. *Elements of automata theory.* Cambridge University Press, 2009.

[Sch62] Wolfgang M. Schmidt. Über die normalität von zahlen zu verschiedenen basen. *Acta Arithmetica*, 7(3):299–309, 1962.

[Sha48] C. E. Shannon. A mathematical theory of communication. *Bell System Technical Journal*, 1948.

[Sie17] Wacław Sierpiński. Démonstration élémentaire du théorème de M. Borel sur les nombres absolument normaux et détermination effective d'un tel nombre. *Bulletin de la Société Mathématique de France*, 45:127–132, 1917.

[SM94] Camille Flye Sainte-Marie. Question 48. *L'intermédiaire des mathématiciens*, 1:107–110, 1894.

[SNS13] Johan Sejr, Brinch Nielsen, and Jakob Grue Simonsen. An experimental investigation of the normality of irrational algebraic numbers. *Mathematis of Computation*, 82:1837–1858, 2013.

[SS72] C. P. Schnorr and H. Stimm. Endliche automaten und zufallsfolgen. *Acta Informatica*, 1:345–359, 1972.

[Tur36] Alan M. Turing. On computable numbers, with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42:230–265, 1936.

[Tur92] A. M. Turing. A note on normal numbers. *Collected Works of A.M. Turing: Pure Mathematics*, pages 117–119, 1992.

[Uga00] Edgardo Ugalde. An alternative construction of normal numbers. *Journal de Théorie des Nombres de Bordeaux*, 12:165–177, 2000.

[ZL70] A. K. Zvonkin and L. A. Levin. The complexity of finite objects and the development of the concepts of information and randomness by means of the theory of algorithms. *Russian Math. Surveys*, 25:83–124, 1970.

[ZL78] Jacob Ziv and Abraham Lempel. Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, 24(5):530–536, 1978.