TECHNISCHE UNIVERSITÄT WIEN
Vienna | Austria

DIPLOMARBEIT

Abzählung von Automaten, formalen Sprachen und verwandten Strukturen

Ausgeführt am Institut für
Diskrete Mathematik und Geometrie
der Technischen Universität Wien

unter der Anleitung von
Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Bernhard Gittenberger

durch

Georg Sedlitz
Vogelsanggasse 39-41/8-9
1050 Wien

Datum

Unterschrift

TECHNISCHE
UNIVERSITÄT
WIEN
Vienna|Austria

DIPLOMA THESIS

Enumeration of Automata, Formal Languages and Related Structures

---

Author
Georg Sedlitz

Supervisor
Ao.Univ.Prof. Dipl.-Ing. Dr.techn. Bernhard Gittenberger

Abstract

One way of characterizing regular languages is through finite deterministic or nondeterministic automata. A counting sequence can be obtained by considering automata with n states over an input alphabet of size k. We study the asymptotic behaviour of this sequence for different classes of automata and their relations to other structures. Upper and lower bounds are obtained for the number of automata and the number of accepted languages using a variety of methods ranging from number theory and graph theory to complex analysis. Furthermore, we will introduce a method of random sampling for initially connected automata.

Acknowledgement

I want to thank my supervisor, Dr. Bernhard Gittenberger, for introducing me to this topic and for his encouragement during the writing process of this thesis. Furthermore I want to thank both Dr. Bernhard Gittenberger and Dr. Alois Panholzer for their excellent lectures on enumerative combinatorics and analysis of algorithms, which were a joy to attend and provided the basis for this thesis.

I also want to thank my family for supporting me throughout my studies and always being there for me. Finally I want to thank my girlfriend Astrid for thoroughly proofreading this thesis and for her much valued advices.

Contents

Chapter 1

Introduction

Regular expressions and finite automata were originally developed with neuron nets and switching circuits in mind [12, p.9]. Since then, a wide range of different applications has arisen. To name a few: the lexical analyzer of a compiler, pattern matching and text processing in text editors [12, p.46], natural language processing and speech recognition [18].
Probably the clearest and most accessible example of finite automata is the use in vending machines [2].

Example: A vending machine Let us consider a vending machine that only accepts €1 and 50 ct coins. Chocolate bars can be bought for a price of €1.50. There are 3 possible inputs for this machine: Inserting €1, inserting €0.50 and selecting a chocolate bar. The vending machine will keep track of the balance. If there is enough money to buy a chocolate bar, the machine will dispense the chocolate bar and, if necessary, the change. The machine will not react to invalid inputs such as selecting a chocolate bar when the balance is only €0.50 or inserting more money when the balance is already €1.50. In this case the additional money is dispensed immediately. The behaviour of this machine can be visualized with the diagram in Figure 1.1.

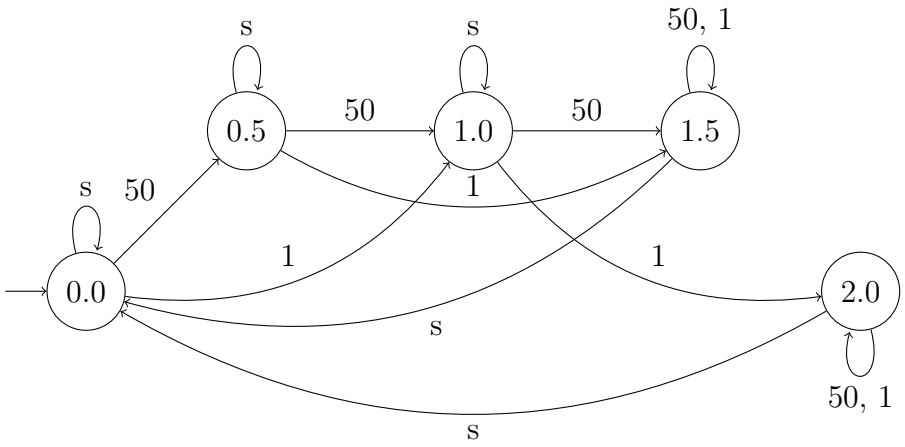


Figure 1.1: A vending machine with inputs "s" for selecting, "50" for inserting 50 ct and "1" for inserting €1.

Aside from the applications in software engineering and computer science, finite automata and regular languages are also of interest from a strictly mathematical point of view, due to the naturalness of the concept of regular languages. The class of regular languages can be defined not only using regular expressions, regular grammars or finite automata, but also by purely algebraic approaches such as recognizing monoids or right-congruence classes.

In this thesis we will focus on enumerating automata and their languages. The study of automata and their counting sequences according to various criteria goes back to the year 1959, when *Victor A. Vyssotsky* [26] wrote a technical report on this subject at the Bell Telephone Laboratories. Since then a lot of work has gone into counting automata, but often ignoring the languages. More recently, not only automata with different properties have been enumerated, but also the accepted languages of automata with special properties [5]. We try to cover the most important results on this subject and will introduce some works of Jeffrey Shallit, Robert W. Robinson, Michael Domaratzky, Valery A. Liskovets et al.

1.1 Basic definitions and notions

1.1.1 Automata and regular expressions

We can think of a deterministic finite automaton (*DFA*) as a machine that reads words letter by letter. Depending on the letter it reads, the automaton changes its state. After the word is read, the automaton does or does not accept the word, depending on the current state.

Definition 1.1. More precisely, a DFA is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$ where

- Q is a finite set of states,
- Σ is the alphabet, a finite set of letters,
- δ is the transition function, a map $Q \times \Sigma \to Q$, which gives the next state depending on the current state and the letter that is being read,
- q_0 is the initial state,
- and $F \subseteq Q$ is the set of accepting states.

The transition function can be naturally extended to all pairs of states q and finite words ω in Σ^* as follows: Let $\omega \in \Sigma^*$ be a word, $a \in \Sigma$ a letter and $q \in Q$ a state, then we can recursively define $\delta(q, \omega a) := \delta(\delta(q, \omega), a)$. Sometimes we want to write $q \cdot \omega$ instead of $\delta(q, \omega)$ in favour of readability if the context is clear.

Definition 1.2. Let $M = (Q, \Sigma, \delta, q_0, F)$ be a DFA, then we say that M accepts the word ω if and only if $\delta(q_0, \omega) \in F$ and the language accepted by M is the set of all words accepted by M.

Definition 1.3. We say that a DFA is initially connected if for every state p there exists a word ω such that $\delta(q_0, \omega) = p$.

A DFA is minimal if there exists no other DFA with fewer states accepting the same language.

It is well known that minimal automata are unique up to isomorphism, meaning that for two minimal automata $M = (Q, \Sigma, \delta, q_0, F)$ and $M' = (Q', \Sigma, \delta', q_0', F')$ accepting the same language, there exists a bijection $\phi : Q \to Q'$ such that $\phi(F) = F'$ and $\phi(\delta(q, \omega)) = \delta'(\phi(q), \omega)$ for all states q, q' and all words ω (see [12], Theorem 3.10 and Subsection 1.1.2).

Definition 1.4. Two states p, q of a DFA are said to be equivalent if for all words ω we have $\delta(p, \omega) \in F \Leftrightarrow \delta(q, \omega) \in F$.

A DFA with two equivalent states cannot be minimal, since one could replace the two states with a single state.

A nondeterministic finite automaton (*NFA*) is defined as above as a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, except for the transition function $\delta : Q \times \Sigma \to 2^Q$ which now maps a pair (q, a) to a subset of states. We say that an NFA accepts some word ω if there exists a path in the transition diagram starting at q_0, labelled with the letters of ω and ending at an accepting state. Initial connectedness of an NFA is defined in the same way as for a DFA: Every state is reachable from the initial state q_0.

There are different ways of defining regular expressions, for example as the class of languages accepted by a DFA. Another approach to the class of regular languages is through *regular expressions*. Naturally, every regular expression describes a regular language.

Definition 1.5. A regular expression over a finite alphabet Σ can be defined using a small set of rules:

- $\emptyset$ and ϵ, the empty string, are regular expressions.
- Every letter $\alpha \in \Sigma$ is a regular expression.
- For regular expressions E_1 and E_2 we have: $E_1 \cup E_2$, $E_1 \cdot E_2$ and E_1^* are again regular expressions, thus regular languages are closed under union, concatenation and the *Kleene star*.

Note that $\Sigma = \{a\} \cup \{b\} \cup \dots$ is also a regular expression as a finite union of singletons.

Another special type of automaton is an *NFA with ϵ-transitions*. The definition of an NFA can be extended such that changes of states are possible without reading a letter. We will not go into the details, but nonetheless want to state the following important result.

Theorem 1.6. The following classes of languages are one and the same, namely the *regular languages*:

- languages accepted by a DFA
- languages accepted by an NFA
- languages accepted by an NFA with ϵ-transitions
- languages defined by a regular expression.

For the proof and additional information we refer to *John E. Hopcroft* [12, pp.13-35].

1.1.2 The Myhill-Nerode theorem

The previously mentioned property of uniqueness up to isomorphism of DFAs does not hold for minimal NFAs.

To get a small glimpse of automata theory and a better understanding of minimal DFAs we want to briefly present the *Myhill-Nerode theorem* and its proof. Not only will we get a proof for the uniqueness of minimal DFAs but also see a purely algebraic approach to regular languages.

Definition 1.7. Let D be a DFA with alphabet A. The relation $\sim_D$ defined by

$$v, w \in A^* : v \sim_D w \quad :\Leftrightarrow \delta(q_0, v) = \delta(q_0, w)$$

is called the *right congruence of D* and is an equivalence relation on A^*.

We identify two words if they end in the same state when read by the automaton. This means that the automaton is not able to distinguish between those words.

Definition 1.8. Let L be a language. The relation $\sim_L$ defined by

$$w_1, w_2 \in A^* : w_1 \sim_L w_2 \quad :\Leftrightarrow \forall v \in A^* : w_1 v \in L \Leftrightarrow w_2 v \in L$$

is called the *right congruence of L* and is an equivalence relation on A^*.

We observe that $v \sim_D w$ implies $v \sim_{L(D)} w$, therefore $\sim_D$ is a refinement of $\sim_{L(D)}$.

We define $w^{-1}L := \{v | wv \in L\}$ as the *left quotient of the language L*. Note that the left quotients are exactly the equivalence classes of the right congruence of L, since

$$\forall w_1, w_2 \in A^* : \quad w_1 \sim_L w_2 \Leftrightarrow w_1^{-1}L = w_2^{-1}L.$$

Theorem 1.9 (Myhill-Nerode)**.** Let A be a finite alphabet and $L \subset A^*$. We call the number of equivalence classes of $\sim_L$ the *index of* $\sim_L$. The following two statements are equivalent.

1. L is a regular language.
2. The index of $\sim_L$ is finite.

Sketch of the proof: (1. $\Rightarrow$ 2.) If L is regular then there is a DFA $M = (Q, \Sigma, \delta, q_0, F)$ that accepts L. W.l.o.g. the automaton M is initially connected. The set of states Q is finite and the equivalence classes of $\sim_M$ can be identified with the states $q \in Q$. Because $\sim_M$ is a refinement of $\sim_{L(M)}$ we get

$$\text{index}(\sim_L) \leq \text{index}(\sim_M) < \infty.$$

(2. $\Rightarrow$ 1.) Given the fact that L is of finite index, a canonical automaton that accepts L can be constructed using the left quotients as states. We define:

- $Q := \{w^{-1}L | w \in A^*\}$
- $q_0 := \epsilon^{-1}L = L$
- $F := \{w^{-1}L | w \in L\}$
- $\delta(w^{-1}L, v) := v^{-1}(w^{-1}L) = (wv)^{-1}L.$

This automaton not only accepts L, but is also minimal since for any other (initially connected) DFA M' with $L(M') = L$ and states Q' we have

$$|Q| = \text{index}(\sim_L) \leq \text{index}(\sim_{M'}) = |Q'|.$$

Furthermore it can be shown that every other minimal DFA that accepts L is isomorphic to the constructed automaton M by a natural bijection [12, p.68].

1.1.3 Equivalence of DFAs and NFAs

Since DFAs and NFAs are of the same expressive power, the set of all languages over a fixed alphabet accepted by NFAs equals the set of languages accepted by DFAs. We find:

Theorem 1.10. Let Σ be an alphabet.

1. For every DFA there exists an NFA accepting the same language.
2. For every NFA there exists a DFA accepting the same language.

Proof. 1. A given deterministic finite automaton $D = (Q, \Sigma, \delta, q_0, F)$ is basically already an NFA. More precisely: The transition diagram does not change, but formally one has to modify the transition function from $\delta(q, a) = q_a$ to $\delta(q, a) = \{q_a\}$.

2. For a given NFA $N = (Q, \Sigma, \delta, q_0, F)$ we construct a DFA D over the same alphabet, in the following manner:

- $D := (Q', \Sigma, \delta', q_0', F')$
- $Q' := 2^Q$, the set of all subsets of Q.
- The transition function δ' is chosen in such a way that it can keep track of all possibles states in the NFA for some input word,

$$q_0' := \{q_0\} \quad \text{and} \quad \delta'(q', a) := \bigcup_{q \in q'} \delta(q, a).$$

  With this choice the set $\delta'(q_0', \omega)$ contains all the states that can be reached reading the word ω in the original given NFA N.

- Therefore every state in Q' that contains at least one of $N's$ final state should be final and

$$F' := \left\{ q' \in 2^Q | q' \cap F \neq \emptyset \right\}.$$

□

The construction above gives $g_k(n) \leq G_k(n) \leq g_k(2^n)$ with $G_k(n)$ denoting the number of distinct languages accepted by NFAs with n states and $g_k(n)$ denoting the number of distinct languages accepted by DFAs with n states, which will be examined later in this thesis. The constructed DFA has 2^n states but is not necessarily minimal, thus the minimal DFA of an NFA language has $\leq 2^n$ states. For now it remains unclear if this bound is tight. In other words: Are there NFAs with n states such that the minimal DFA accepting the same language has exactly 2^n states? We will give an answer to this question in Theorem 4.16 as observed by *Domaratzki, Kisman and Shallit* [5].

1.1.4 A small example

Let us recap this section by studying a simple language over the alphabet $\Sigma = \{a, b\}$,

$$L = \{a, b\}^* aa\{a, b\}^*.$$

This is the language of all words containing at least two consecutive a's. L is regular. According to the Myhill-Nerode theorem the relation $\sim_L$ is of finite index and can be used to construct the minimal DFA accepting L. To calculate all the left-quotients of L, we use the identities

- $w^{-1}(v^{-1}L) = (vw)^{-1}L$,
- $w^{-1}(L_1 \cup L_2) = w^{-1}L_1 \cup w^{-1}L_2$ and
- $(w^{-1}L)^c = w^{-1}(L^c)$.

We start with $q_0 := \epsilon^{-1}L = L$ and calculate left quotients until this process does not generate new ones,

$$a^{-1}L = a^{-1}\left(\Sigma^+ aa\Sigma^* \cup aa\Sigma^*\right) = \Sigma^* aa\Sigma^* \cup a\Sigma^* = L \cup a\Sigma^*.$$

This will be the state $\delta(q_0, a)$. Now we calculate $\delta(q_0, b)$ by

$$b^{-1}L = b^{-1}\left(\Sigma^* aa\Sigma^*\right) = L.$$

This is again q_0. Next we try $\delta(q_0, aa)$. This approach can be thought of as a breadth-first search in the transition diagram. We calculate $(aa)^{-1}L$ by

$$a^{-1}\left(a^{-1}L\right) = a^{-1}\left(L \cup a\Sigma^*\right) = \Sigma^*.$$

This is a dead end since for all words ω we have $\omega^{-1}\Sigma^* = \Sigma^*$. So our next transition to be calculated is $\delta(q_0, ab)$,

$$b^{-1}\left(a^{-1}L\right) = b^{-1}\left(L \cup a\Sigma^*\right) = L \cup \emptyset = L.$$

Since this set of left-quotients is closed under forming left-quotients we have found our states and already know the transition function. The states are

$$q_0 := L, \quad q_1 := a^{-1}L = L \cup a\Sigma^* \quad \text{and} \quad q_2 := (aa)^{-1}L = \Sigma^*.$$

The final states are the states $\omega^{-1}L$ with $\omega \in L$. Since $\epsilon \notin L$ and $a \notin L$ but $aa \in L$, the single final state is q_2. The resulting DFA in Figure 1.2 is the unique (up to isomorphism) minimal DFA accepting L.

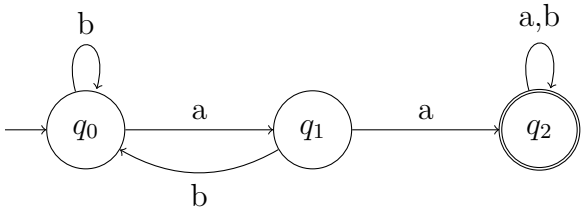


Figure 1.2: The DFA obtained from the left-quotients of L

The language L can also be described using an NFA.

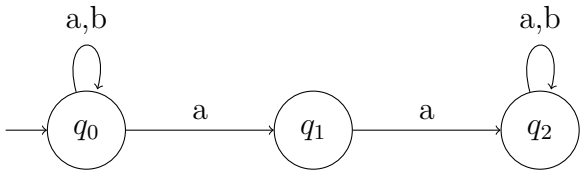


Figure 1.3: An NFA accepting L

To get a better picture of the equivalence of DFA and NFA languages, let us take a look at the algorithm for finding a DFA accepting the same language. The states of the new DFA are all the subsets of the NFA,

$$\emptyset, \{q_0\}, \{q_1\}, \{q_2\}, \{q_0, q_1\}, \{q_0, q_2\}, \{q_1, q_2\} \quad \text{and} \quad \{q_0, q_1, q_2\}.$$

We will name these states $\emptyset, 0, 1, 2, 01, 02, 12$ and 012. The initial state is $\{q_0\} = 0$ and the final states are all sets containing a final state of the original automaton: 2, 02, 12 and 012. Now we just have to detect which set of states can be reached from a set of states using a single letter. The resulting DFA is visualized in Figure 1.4.

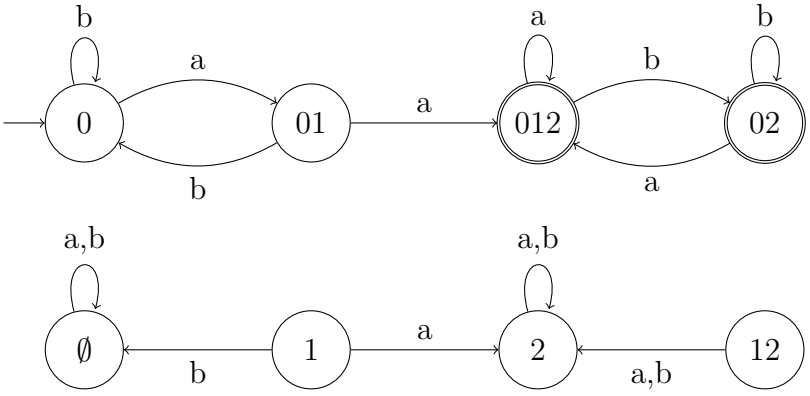


Figure 1.4: A DFA accepting L

Notice that the states $\emptyset$, 1, 2 and 12 are unreachable. This procedure simply gives one DFA accepting L, not necessarily the minimal DFA.

1.2 Some words on asymptotics

Upper and lower bounds give us an idea of the growth of a sequence or function. More precise estimates are given by other (simpler) terms with the same asymptotic behaviour.

Definition 1.11. The functions f and g are said to be *asymptotically equivalent*, written

$$f \sim g$$

if and only if

$$\lim_{x\to\infty} \frac{f(x)}{g(x)} = 1.$$

This is an equivalence relation.

In terms of the Landau notation we find that $f \sim g \Leftrightarrow f - g = o(g)$.
As an example for asymptotic equivalence we give Stirling's well known approximation formula. It states that

$$n! \sim \left(\frac{n}{e}\right)^n \sqrt{2\pi n}. \tag{1.1}$$

This asymptotic estimate is often of great use, since factorials arise in many counting problems.
Another example is the following observation: Two polynomials $p(x)$ and $q(x)$ are asymptotically equivalent if and only if they are of the same order and their leading coefficients are the same. Note that one can multiply and divide when working with asymptotic functions, given that the functions are nonzero. For example let $a_n/b_n \sim c_n$ then we get $a_n \sim b_n \cdot c_n$.

Another example for handling asymptotic functions is

$$a_n \sim b_n \Rightarrow \log a_n \sim \log b_n.$$

This is true since

$$\begin{aligned}\lim_{n\to\infty} \frac{\log a_n}{\log b_n} &= \lim_{n\to\infty} \frac{\log a_n - \log b_n}{\log b_n} + 1 = \lim_{n\to\infty} \frac{\log \frac{a_n}{b_n}}{\log b_n} + 1 \\ &= \lim_{n\to\infty} \frac{\log 1}{\log b_n} + 1 = 1.\end{aligned}$$

On the other hand, we have $n \sim n+1$ but $\exp(n) \nsim \exp(n+1)$ since $\exp(n)/\exp(n+1) \to 1/e \neq 1$. This raises the question:

For which functions h does $f \sim g$ imply $h(f) \sim h(g)$?

Furthermore: Under which conditions does $f \sim g$ imply $f^{-1} \sim g^{-1}$? Answers to both of these questions are provided by *R. C. Entringer* [7]. Without proof we state the following theorem.

Theorem 1.12.

1. Let $f(x) \to \infty$ and $f(x) \sim g(x)$ as $x \to \infty$. If $h(x)$ is monotonic and $h'(x)/h(x) = \mathcal{O}(1/x)$ then $h(f(x)) \sim h(g(x))$.
2. If $h'(x)$ is strictly monotonic for x sufficiently large and $h(x)/h'(x) = o(1)$, then there are functions $f(x)$ and $g(x)$ such that $f(x) \sim g(x)$ but $h(f(x)) \nsim h(g(x))$.

For example the natural logarithm $h(x) = \log(x)$ satisfies these conditions since

$$\frac{h'(x)}{h(x)} = \frac{1}{x \log(x)} = \mathcal{O}(1/x) \quad \text{for} \quad x \to \infty.$$

For the proof of Theorem 4.2 we will need the following conditions that preserve asymptotic equivalence for inverse functions.

Theorem 1.13.

1. Let $f(x) \to \infty$ and $f(x) \sim g(x)$ as $x \to \infty$. If $h(x) \sim f^{-1}(x)$, $g^{-1}(x)$ exists, h is monotonic and $h'(x)/h(x) = \mathcal{O}(1/x)$, then $h(x) \sim g^{-1}(x)$. Therefore $f^{-1}(x) \sim g^{-1}(x)$.
2. Let $f(x) \to \infty$ and $h(x)/h'(x) = o(1)$. If $h(x) = f^{-1}(x)$ and both $h'(x)$ and $h(x)/h'(x)$ are strictly increasing for x sufficiently large then there is a function $g(x)$ with inverse $g^{-1}(x)$ such that we have $f(x) \sim g(x)$ but $f^{-1}(x) \nsim g^{-1}(x)$.

Chapter 2

Important Methods

In this thesis we will not explain enumerative combinatorics from scratch. We will roughly outline the most important methods we use and refer to other works like [25], that provide an introduction to this topic.

2.1 Powerseries

Definition 2.1. For a real or complex sequence $(a_n)_{n\in\mathbb{N}}$ we call the power series

$$A(z) = \sum_{n\geq 0} a_n z^n$$

its *generating function* (GF), or ordinary generating function (OGF). The power series

$$\hat{A}(z) = \sum_{n\geq 0} \frac{a_n z^n}{n!}$$

is called the *exponential generating function* (EGF) of a_n.

Generating functions are an important tool when studying counting sequences. They can be approached as formal power series or as analytic functions. Formal power series can be observed without worrying about convergence (this would be a problem for fast-growing coefficients). The big advantage of analytic functions is that many results from complex analysis are applicable. We will

distinguish between these two approaches only if necessary.

A few examples: The generating function of the sequence

$$\langle 12, 8, 4, 0, 0, 0, 0, ... \rangle \quad \text{is} \quad 12 + 8z + 4z^2.$$

The GF of the constant sequence $a_n = 1$ is

$$\sum_{n \geq 0} z^n = \frac{1}{1-z}.$$

The GF of $f_n = n$ for $n \in \mathbb{N}$ can be obtained by observing

$$\left(\frac{1}{1-z}\right)' = \left(1 + z + z^2 + z^3 + ...\right)' = 1 + 2z + 3z^2 + 4z^3 + \dots.$$

and multiplying with z

$$\sum_{n \geq 0} n z^n = z \cdot \left(\frac{1}{1-z}\right)' = \frac{z}{(1-z)^2}.$$

We often want to obtain the coefficients for a given GF. We write $[z^n]F(z)$ for the n-th coefficient of the function $F(z)$,

$$[z^k]F(z) = [z^k] \sum_{n \geq 0} f_n z^n = f_k.$$

The n-th coefficient can be obtained by deriving and then evaluating at 0, i.e. $f_n = \frac{F^{(n)}(0)}{n!}$. Sometimes a closed expression can be found, for example

$$[z^n](1+z)^\alpha = \binom{\alpha}{n}.$$

Using Newton's generalized binomial coefficients,

$$\binom{r}{k} = \frac{r \cdot (r-1) \cdot ... \cdot (r-k+1)}{k!},$$

we get an even stronger result. For $r \in \mathbb{C}$ and $k \in \mathbb{N}$ we have the following lemma.

Lemma 2.2. If $\alpha, \beta \in \mathbb{C}$ then it holds that

$$[z^n]\frac{1}{(1-\beta z)^\alpha} = \binom{\alpha+n-1}{n}\beta^n.$$

Proof. We apply the binomial theorem.

$$\begin{aligned}
[z^n]\frac{1}{(1-\beta z)^\alpha} &= [z^n]\sum_{k\geq 0}\binom{-\alpha}{k}(-\beta)^k z^k = \binom{-\alpha}{n}(-1)^n\beta^n \\
&= \frac{-\alpha\cdot(-\alpha-1)\cdot\ldots\cdot(-\alpha-n+1)}{n!}(-1)^n\beta^n \\
&= \frac{\alpha\cdot(\alpha+1)\cdot\ldots\cdot(\alpha+n-1)}{n!}\beta^n = \binom{\alpha+n-1}{n}\beta^n
\end{aligned}$$

□

We observe some important properties for coefficients of GFs.

Lemma 2.3. For power series $A(z)$, $B(z)$ and $\lambda \in \mathbb{C}$ we have

$$\begin{aligned}
[z^n]\left(A(z)+B(z)\right) &= [z^n]A(z) + [z^n]B(z), \\
[z^n]\left(\lambda A(z)\right) &= \lambda\cdot[z^n]A(z) \quad \text{and} \\
[z^n]A(z) &= [z^{n+1}]\left(z\cdot A(z)\right).
\end{aligned}$$

2.2 Lagrange's inversion formula

For a given formal power series $g(z) = \sum_{n\geq 0} g_n z^n$ an inverse power series g^{-1} exists if and only if $g_0 = 0$ and $g_1 \neq 0$. It follows that $g(z)$ can be written as

$$g(z) = z\left(g_1 + g_2 z + g_3 z^2 + \ldots\right) = z\frac{1}{\phi(z)},$$

with some power series ϕ with $\phi(0) \neq 0$. We substitute z with $g^{-1}(z)$ to get the identity

$$z = \frac{g^{-1}(z)}{\phi(g^{-1}(z))}$$

which can be used to calculate the coefficients of $g^{-1}(z)$. This explains the name of the following theorem, *Lagrange's inversion formula*. Not only can it be applied to inverse functions, but also to functions which are not given explicitly, but instead as a specific solution of an equation.

Theorem 2.4 (Lagrange's inversion formula)**.** Let $\phi(x) = \sum_{n\geq 0} \phi_n x^n$ with $\phi_0 \neq 0$ and $f(u)$ be another power series. If further $z = \frac{u}{\phi(u)}$ then

$$[z^n]f(u) = \frac{1}{n}[u^{n-1}]f'(u)\left(\phi(u)\right)^n.$$

Let us for example consider the inverse function of $x \mapsto xe^x$, the Lambert-W function, which we will need later. Let us call this function W_0. We would like to give a formula for

$$[z^n]W_0(z) = [z^n]\sum_{n\geq 0} w_n z^n,$$

although there is no closed expression for $W_0(z)$. Nevertheless it is possible using Theorem 2.4. We apply it to the function W_0. As W_0 is the inverse of $x \mapsto xe^x$ we know that $z = W_0(z)e^{W_0(z)}$. Thus, we have

$$z = \frac{W_0}{e^{-W_0}}.$$

Theorem 2.4 is applicable with $f(u) = u$, $u = W_0(z)$ and $\phi(u) = e^{-u}$. We get

$$[z^n]W_0(z) = \frac{1}{n}[u^{n-1}]e^{-nu} = \frac{1}{n}[u^{n-1}]\sum_{k\geq 0}\frac{(-nu)^k}{k!} = \frac{(-n)^{n-1}}{n!}.$$

2.3 Combinatorial structures

For a comprehensive introduction to enumerative combinatorics see [25], [24] and [8].

Definition 2.5. A countable set $\mathcal{A}$ with a real valued weight function w is called a *combinatorial structure*, if and only if:

- $\forall a \in \mathcal{A} : w(a) \in \mathbb{N}$
- $\forall n \in \mathbb{N} : |w^{-1}(n)| < \infty$.

Given such a structure, we can write

$$\mathcal{A}_n := \{x \in \mathcal{A} | w(x) = n\}$$

and study the counting sequence of $\mathcal{A}$, which we denote by $a_n := |\mathcal{A}_n|$. The GF of this sequence is

$$A(z) = \sum_{\alpha \in \mathcal{A}} z^{w(\alpha)} = \sum_{n \geq 0} a_n z^n.$$

We will refer to this sequence simply as the GF of $\mathcal{A}$ if the weight function is unambiguous. There is a very useful correspondence between the construction of a combinatorial structure and its GF.

Theorem 2.6. Let $\mathcal{A}$ and $\mathcal{B}$ be combinatorial structures with generating functions $A(z) = \sum_{n \geq 0} a_n z^n$ and $B(z) = \sum_{n \geq 0} b_n z^n$. Then we have:

- If $\mathcal{A} \cap \mathcal{B} = \emptyset$ then the GF of $\mathcal{C} = \mathcal{A} \dot{\cup} \mathcal{B}$ is $C(z) = A(z) + B(z)$.
- The Cartesian product $\mathcal{D} = \mathcal{A} \times \mathcal{B}$ with the weight function $(a, b) \mapsto w(a) + w(b)$ has the generating function $D(z) = A(z) \cdot B(z)$.

In addition to

$$\begin{aligned} \text{Disjoint union:} \quad & \mathcal{A} \dot{\cup} \mathcal{B} \to A(z) + B(z), \\ \text{Cartesian product:} \quad & \mathcal{A} \times \mathcal{B} \to A(z) \cdot B(z), \end{aligned}$$

we also observe finite sequences and substitution of combinatorial structures:

$$\begin{aligned} \text{Sequences:} \quad & \mathcal{A}^* \to \frac{1}{1 - A(z)}, \\ \text{Substitution:} \quad & \mathcal{A}(\mathcal{B}) \to A(B(z)). \end{aligned}$$

Finite sequences, also written as $\textsc{Seq}(\mathcal{A})$ instead of $\mathcal{A}^*$, are defined by

$$\mathcal{A}^* := \{\epsilon\} \,\dot\cup\, \mathcal{A} \,\dot\cup\, (\mathcal{A}\times\mathcal{A}) \,\dot\cup\, (\mathcal{A}\times\mathcal{A}\times\mathcal{A}) \,\dot\cup \ldots$$

For the substitution, objects in $\mathcal{A}$ are made up of atoms and every atom is replaced with an object of $\mathcal{B}$. Note that the structure $\mathcal{B}$ must not contain an object of size 0.

Labelled structures: The exponential generating functions will be of use when handling labelled structures. A labelled combinatorial structure of size n has n distinct labels, one for each atom, for example the integers 1 to n. Without any claim to comprehensiveness we give some important constructions for labelled structures and their exponential generating functions:

$$\begin{aligned}
\text{Disjoint union:} \quad & \mathcal{A} \,\dot\cup\, \mathcal{B} && \to \hat{A}(z) + \hat{B}(z), \\
\text{Labelled product:} \quad & \mathcal{A} * \mathcal{B} && \to \hat{A}(z) \cdot \hat{B}(z), \\
\text{Sequences:} \quad & \mathcal{A}^* && \to \frac{1}{1-\hat{A}(z)}, \\
\text{Set:} \quad & \textsc{Set}(\mathcal{A}) && \to \exp(\hat{A}(z)), \\
\text{Cycles:} \quad & \textsc{Cyc}(\mathcal{A}) && \to \log\left(\frac{1}{1-\hat{A}(z)}\right).
\end{aligned}$$

The labelled product of two objects with n_1 and n_2 labels is the Cartesian product of the unlabelled objects with $(n_1 + n_2)$ labels assigned to the atoms.

Let us demonstrate these ideas by finding the EGF of permutations of n elements. We write $\mathcal{Z}$ for an atom and $\mathcal{P}$ for the class of all permutations. A permutation is a set of labelled cycles, therefore

$$\mathcal{P} = \textsc{Set}(\textsc{Cyc}(\mathcal{Z})).$$

It follows that the EGF of $\mathcal{P}$ is

$$\hat{P}(z) = \exp\left(\log\frac{1}{1-z}\right) = \frac{1}{1-z} = \sum_{n\geq 0} \frac{n! z^n}{n!},$$

which is the EGF of the sequence $n!$.

2.4 Multivariate generating functions

A *multivariate generating function* is a power series with two or more arguments. For example, the bivariate case is

$$F(x,y) = \sum_{n,k\geq 0} f_{n,k} x^n y^k.$$

The sequence of coefficients has a multi-index. It can be used as a "refinement" of an ordinary generating function. The coefficient $f_{n,k}$ could for example equal the number of objects of size n and parameter k.

Example: Binary trees They can be specified by the equation

$$\mathcal{B} = \{\circ\} \,\dot\cup\, \mathcal{B}\times\{\bullet\}\times\mathcal{B}$$

where $\circ$ is the tree consisting only of a root and $\bullet$ is an internal node. When counting binary trees with n internal nodes the above equation translates to

$$B(z) = 1 + zB^2(z)$$

for the OGF $B(z)$. This gives

$$B(z) = \frac{1-\sqrt{1-4z}}{2z}.$$

Now we are interested in the number of binary trees with n internal nodes and k of them being right side descendants of the root (see Figure 2.1). Let $d_{n,k}$ be this number and

$$D(z,y) = \sum_{n,k\geq 0} d_{n,k} z^n y^k$$

the bivariate GF. We can modify the construction of the binary trees and mark every right side internal node with the variable y. We get

$$D(z,y) = 1 + zB(z)B(yz) = 1 + \frac{(1-\sqrt{1-4z})(1-\sqrt{1-4yz})}{4yz}.$$

Knowing that

$$[z^n]B(z) = \frac{1}{n+1}\binom{2n}{n},$$

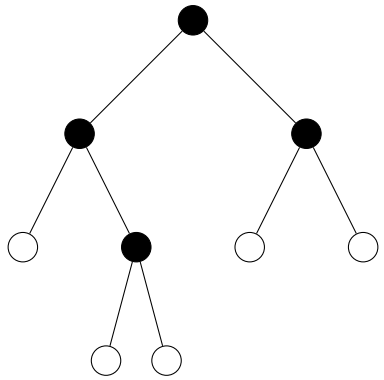


Figure 2.1: A binary tree with $n = 4$ and $k = 1$

we can deduce

$$d_{n,k} = [z^n y^k] D(z, y) = \frac{1}{k+1}\binom{2k}{k}\frac{1}{n-k}\binom{2n-2k-2}{n-k-1}.$$

This is not surprising, since a binary tree of this kind can be interpreted as a pair of two binary trees with sizes $n-k-1$ and k.

Example: Set partitions In Chapter 6 we will come across the Stirling numbers of the second kind, which count the number of partitions of a set of size n into k parts. Throughout this thesis, when speaking of Stirling numbers, we will always mean the Stirling numbers of the second kind. Let $\left\{ {n \atop k} \right\}$ be the number of said partitions and let $\mathcal{P}$ be the class of all partitions. A partition can be constructed as a set of labelled sets of size > 0,

$$\mathcal{P} = \textsc{Set}(\textsc{Set}_{\geq 1}(\mathcal{Z})).$$

This translates directly to the EGF

$$P(z) = e^{e^z - 1}.$$

We can now mark each part making up the partition with the variable y. We get the bivariate EGF

$$e^{y(e^z-1)} = \sum_{n,k \geq 0} \frac{\left\{ {n \atop k} \right\} z^n y^k}{n!}.$$

2.5 Introduction to the saddle point method

A comprehensive guide to this method is found in the book *Analytic Combinatorics* by *Flajolet and Sedgewick* [8]. A saddle point of an analytic function f is a solution of the *saddle point equation*

$$f'(z) = 0$$

with $f(z) \neq 0$. Let ζ be a saddle point. Imagining the function $|f|$ as a landscape, we see that ζ is really a saddle point in terms of real analysis. The reason for this is the maximum modulus principle which states that for any z_0 in the domain of some analytic function, z_0 cannot be a local maximum of $|f(z)|$. It can also not be a local minimum if z_0 is a nonzero of $f(z)$.

The general idea of the saddle point method is to estimate some complex integral. This can be done by choosing a path of integration that goes through a saddle point of the integrand in such a way, that in an asymptotic sense, only the parts of the curve close to the saddle point are of interest.
We want to focus on Cauchy coefficient integrals, which appear when studying enumerative combinatorics.

Let us demonstrate the saddle point method by finding an asymptotic estimate for the inverse factorial (which we already know due to Stirling's formula). We define

$$K_n := \frac{1}{n!} = [z^n]e^z = \frac{1}{2\pi i}\oint \frac{e^z}{z^{n+1}}dz.$$

The integrand $F(z) = e^z \cdot z^{-n-1}$ has a saddle point at $\zeta = n+1$. Now we fix a path of integration encircling the origin. Usually we would choose $r := \zeta$, but often it is sufficient to settle for an integration path that passes near the saddle point, but does not go right through it. For convenience we fix $r := n$ and get

$$\begin{aligned} K_n &= \frac{1}{2\pi i}\oint \frac{e^z}{z^{n+1}}dz = \frac{1}{2\pi}\int_{-\pi}^{\pi} \frac{e^{ne^{i\theta}}}{(ne^{i\theta})^{n+1}} \cdot ne^{i\theta}d\theta \\ &= \frac{1}{2\pi}\frac{e^n}{n^n}\int_{-\pi}^{\pi} e^{n(e^{i\theta}-i\theta-1)}d\theta. \end{aligned}$$

We will denote the new exponent by f and $e^{i\theta} - i\theta - 1$ by $h(\theta)$, thus $f(\theta) = n \cdot h(\theta)$. We see that the integrand peaks at $\theta = 0$ since

$$|e^{nh(\theta)}| = |e^{n(e^{i\theta}-i\theta-1)}| = e^{\mathrm{Re}\left(n(e^{i\theta}-i\theta-1)\right)} = e^{n(\cos\theta-1)}.$$

Our next step is to split the integral into two parts,

$$I_n := K_n^{(0)} + K_n^{(1)} = \int_{-\theta_0}^{\theta_0} e^{nh(\theta)}d\theta + \int_{\theta_0}^{2\pi-\theta_0} e^{nh(\theta)}d\theta,$$

with integration paths $\mathcal{C}_0$ and $\mathcal{C}_1$ such that:

1. The second part does not contribute in the sense of asymptotics, meaning $K_n^{(1)} = o\left(I_n\right)$.

2. Along the path of integration of the relevant part $K_n^{(0)}$ it holds that

   $$f(z) = f(\zeta) + \frac{1}{2}f''(\zeta)(z-\zeta)^2 + \mathcal{O}(\eta_n)$$

   with $\eta_n \to 0$ for $n \to \infty$ uniformly on $\mathcal{C}_0$. This guarantees an approximation with a Gaussian Integral.

Looking at the second condition for our function $nh(\theta) = n\sum_{k=2}^{\infty}(i\theta)^k/k!$, we find that $n\theta^3 \to \infty$ must hold.
This perfectly fits the rule of thumb for the choice of θ_0 for Cauchy coefficient integrals from *Analytic Combinatorics* [8]:

$$f''(\zeta)\theta_0^2 \to \infty, \qquad f'''(\zeta)\theta_0^3 \to 0.$$

Flajolet and Sedgewick refer to this choice as the *saddle-point dimensioning heuristic*. Following this heuristic, we want to satisfy

$$n\theta_0^2 \to \infty \quad \text{and} \quad n\theta_0^3 \to 0$$

with $\theta_0 = n^\alpha$, which leads us to $-\frac{1}{2} < \alpha < -\frac{1}{3}$. Let $\theta_0 := n^{-\frac{2}{5}}$. We show that $K_n^{(1)}$ does not contribute to I_n by verifying that it is exponentially small for $n \to \infty$:

$$\begin{aligned}|K_n^{(1)}| &= \left|\int_{\theta_0}^{2\pi-\theta_0} e^{nh(\theta)}d\theta\right| \leq \int_{\theta_0}^{2\pi-\theta_0} |e^{nh(\theta)}|d\theta \leq 2\pi e^{n(\cos\theta_0-1)} \\ &= 2\pi e^{n(\cos n^{-0.4}-1)} = \mathcal{O}\left(e^{-\frac{1}{2}n^{1/5}}\right).\end{aligned}$$

The last identity can be verified by

$$n(\cos n^{-2/5} - 1) = n\sum_{k=1}^{\infty} \frac{n^{-4k/5}}{(2k)!}(-1)^k = -\frac{n^{1/5}}{2} + \sum_{k=2}^{\infty} \frac{n^{1-4k/5}}{(2k)!}(-1)^k$$

and

$$\left|\sum_{k=2}^{\infty} \frac{n^{1-4k/5}}{(2k)!}(-1)^k\right| \leq \sum_{k=0}^{\infty} \frac{1}{(2k)!} = \cosh(1).$$

Now let us move to the central part $K_n^{(0)}$. We have $h(\theta) = -\theta^2/2 + \mathcal{O}(\theta^3)$ for $|\theta| \leq \theta_0$, thus

$$K_n^{(0)} = \int_{-\theta_0}^{\theta_0} e^{nh(\theta)} d\theta = \int_{-n^{-2/5}}^{n^{-2/5}} e^{-n\theta^2/2} d\theta \left(1 + \mathcal{O}(n^{-1/5})\right).$$

We substitute $t := \sqrt{n}\theta$ and get an incomplete Gaussian integral,

$$K_n^{(0)} = \frac{1}{\sqrt{n}} \int_{-n^{1/10}}^{n^{1/10}} e^{-t^2/2} dt \left(1 + \mathcal{O}(n^{-1/5})\right).$$

Lemma 2.7. For $c > 0$ it holds that

$$\int_c^{\infty} e^{-t^2/2} dt = \mathcal{O}\left(e^{-c^2/2}\right).$$

Proof.

$$\frac{\int_c^{\infty} e^{-t^2/2} dt}{e^{-c^2/2}} = \int_c^{\infty} e^{-(t^2-c^2)/2} dt \leq \int_c^{\infty} e^{-(t-c)^2/2} dt = \int_0^{\infty} e^{-u^2/2} du = \sqrt{\frac{\pi}{2}}$$

□

Therefore we have

$$K_n^{(0)} \sim \frac{1}{\sqrt{n}} \int_{-\infty}^{\infty} e^{-t^2/2} dt = \sqrt{\frac{2\pi}{n}}.$$

Finally we estimate

$$\frac{1}{n!} = \frac{1}{2\pi}\frac{e^n}{n^n} \int_{-\pi}^{\pi} e^{n(e^{i\theta} - i\theta - 1)} d\theta = \frac{1}{2\pi}\frac{e^n}{n^n} I_n \sim \frac{1}{2\pi}\frac{e^n}{n^n}\sqrt{\frac{2\pi}{n}} = \frac{1}{e^{-n} n^n \sqrt{2\pi n}},$$

which is exactly the asymptotic behaviour we expected.

Chapter 3

Minimal Deterministic Finite Automata

Throughout this thesis we will use the same notation as *Domaratzki, Kisman and Shallit* used in [5].

Theorem 3.1. Let $f_k(n)$ be the number of different minimal DFAs up to isomorphism with n states and a k letter input alphabet and let $g_k(n)$ be the number of distinct languages accepted by DFAs with n states over a k letter alphabet. Then we have

$$g_k(n) = f_k(1) + f_k(2) + ... + f_k(n).$$

Proof. The right-hand side of the equation is the number of different minimal DFAs with at most n states. This is also the number of distinct languages accepted by DFAs with at most n states. For any language accepted by an automaton M with $m < n$ states, we can simply add some isolated states, such that the language is also accepted by an automaton with n states. Thus, the right-hand side equals $g_k(n)$. □

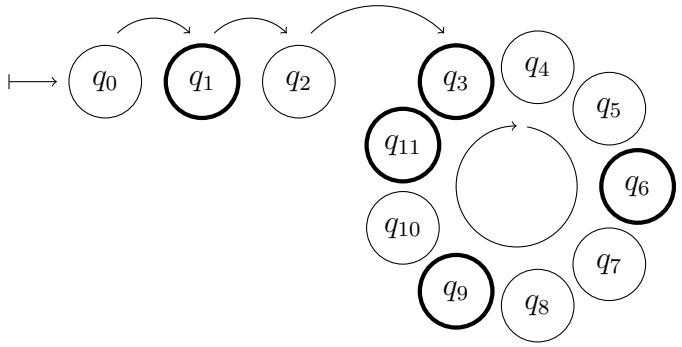


Figure 3.1: A minimal DFA with 12 states and $\Sigma = \{a\}$

3.1 The unary case

The results in this section are based on the work of *Domaratzki, Kisman and Shallit* [5]. For a one letter alphabet we can characterize the minimal DFAs with n states by three conditions.

Theorem 3.2. A DFA M with states $Q = \{q_0, q_1, ..., q_{n-1}\}$ and $\Sigma = \{a\}$ is minimal if and only if all of the following conditions hold.

a) M is initially connected. Thus, it consists of a "tail" and a "loop", i.e. $\delta(q_i, a) = q_{i+1}$ for $0 \leq i \leq n-2$ and $\delta(q_{n-1}, a) = q_j$ for some j.

b) The loop (containing the states $q_j, q_{j+1}, ..., q_{n-1}$) is minimal. It cannot be replaced by an equivalent smaller loop.

c) If $j \neq 0$, the states q_{j-1} and q_{n-1} have to be of opposite "finality". Therefore exactly one of them is an element of F.

To count these automata, one has to find a formula for the number of primitive words. A word ω is said to be primitive if and only if it cannot be written as $\omega = \alpha^j$ for some $\alpha \in \Sigma^*$ and some integer $j \geq 2$. The loop in the transition diagram is minimal, if and only if the 0-1-sequence obtained by the sequence of finalities of states is a primitive word. For example the cycle $[q_3, q_4, \ldots, q_{11}]$ in Figure 3.1 would translate to the sequence $[1, 0, 0, 1, 0, 0, 1, 0, 1]$. The word

100100101 is primitive, thus the given loop is minimal.

To count primitive words we use the Möbius function: $\mu(n) = (-1)^j$ if n is the product of j distinct prime numbers and $\mu(n) = 0$ otherwise.

Theorem 3.3. The number of primitive words of length n over a k-letter alphabet denoted by $T(n, k)$ is

$$T(n,k) = \sum_{d|n} \mu(d) k^{n/d}.$$

Proof. A nonprimitive word is the power of some unique primitive word. Since the number of primitive words is the number of all words minus the number of nonprimitive words we get

$$T(n,k) = k^n - \sum_{\substack{d|n \\ d<n}} T(d,k).$$

Now we show that the explicit formula satisfies this recursion. We get

$$\begin{aligned} \sum_{d|n} \mu(d) k^{n/d} &= k^n - \sum_{\substack{d|n \\ d<n}} \sum_{j|d} \mu(j) k^{d/j} \\ k^n &= \sum_{d|n} \mu\left(\frac{n}{d}\right) k^d + \sum_{\substack{j\cdot i|n \\ j\cdot i<n}} \mu(j) k^i \\ 0 &= \sum_{\substack{d|n \\ d<n}} \mu\left(\frac{n}{d}\right) k^d + \sum_{\substack{j\cdot i|n \\ j\cdot i<n}} \mu(j) k^i. \end{aligned}$$

We compare the coefficients of k^i for every $i|n$ and $i < n$ which gives

$$0 = \mu\left(\frac{n}{i}\right) + \sum_{\substack{j|n/i \\ j<n/i}} \mu(j) = \sum_{j|n/i} \mu(j).$$

The last equation is a well known property of the Möbius function and can be proven by induction over the number of prime factors. □

If an automaton with n states has a tail of length j for some $1 \leq j < n$, then there exists one state with fixed finality. This is the state right before entering the loop. There are no restrictions regarding the other states in the tail. Therefore we can give the following formula.

Theorem 3.4. As before, let $T(n,k)$ be the number of primitive words of length n over a k-letter alphabet. Then we have

$$f_1(n) = T(n,2) + \sum_{1\le j\le n-1} T(n-j,2)2^{j-1}.$$

To get an asymptotic estimate for $g_1(n)$ (and also for $f_1(n)$) we use the identity

$$g_1(n) = \sum_{1\le t\le n} f_1(t)$$

and another formula for $g_1(n)$.

Theorem 3.5. The number $g_1(n)$ of languages over a 1-letter alphabet satisfies

$$g_1(n) = \sum_{1\le t\le n} T(t,2)2^{n-t}.$$

Proof.

$$\begin{aligned}
g_1(n) &= \sum_{1\le t\le n} f_1(t) = \sum_{1\le t\le n}\left(T(t,2) + \sum_{1\le j\le t-1} T(t-j,2)2^{j-1}\right)\\
&= \sum_{1\le t\le n}\left(T(t,2) + \sum_{1\le i\le t-1} T(i,2)2^{t-i-1}\right)\\
&= \sum_{1\le t\le n} T(t,2) + \sum_{1\le i\le t-1} T(i,2)\cdot\left(\sum_{t=i+1}^{n} 2^{t-i-1}\right)\\
&= \sum_{1\le t\le n} T(t,2) + \sum_{1\le i\le t-1} T(i,2)\cdot\left(2^{n-i}-1\right)\\
&= \sum_{1\le t\le n} T(t,2)2^{n-t}
\end{aligned}$$

□

Now we get a good asymptotic estimate for $g_1(n)$.

Theorem 3.6. We have

$$g_1(n) = 2^n \left(n - \alpha + \mathcal{O}\left(n2^{-n/2}\right)\right)$$

where α is a constant with $\alpha \in [1.38, 1.39]$.

Proof. Using Theorem 3.3 and Theorem 3.5 we get

$$\begin{aligned} g_1(n) &= \sum_{1\le t\le n} T(t,2)2^{n-t} = \sum_{1\le t\le n} \left(\sum_{d|t} \mu(d)2^{t/d}\right) 2^{n-t} \\ &= 2^n \left(\sum_{1\le t\le n} \sum_{d|t} \mu(d)2^{t/d-t}\right) = 2^n \left(n + \sum_{1\le t\le n} \sum_{\substack{d|t \\ d\neq 1}} \mu(d)2^{t/d-t}\right). \end{aligned}$$

Thus, we only need to further investigate the expression

$$\sum_{1\le t\le n} \sum_{\substack{d|t \\ d\neq 1}} \mu(d)2^{t/d-t}.$$

We use the substitution $t = kd$ and get

$$\begin{aligned} \sum_{1\le t\le n} \sum_{\substack{d|t \\ d\neq 1}} \mu(d)2^{t/d-t} &= \sum_{2\le d\le n} \mu(d) \sum_{1\le k\le \frac{n}{d}} 2^{k-kd} \\ &= \sum_{2\le d\le n} \mu(d) \left(\sum_{k\ge 1} 2^{k(1-d)} - \sum_{k>\frac{n}{d}} 2^{k(1-d)}\right) \\ &= \sum_{2\le d\le n} \mu(d) \left(\frac{2^{1-d}}{1-2^{1-d}} + \mathcal{O}\left(2^{n/d-n}\right)\right) \\ &= \left(\sum_{2\le d\le n} \mu(d) \cdot \frac{1}{2^{d-1}-1}\right) + \mathcal{O}\left(n2^{n/2-n}\right) \\ &= \left(-\sum_{2\le d\le n} \frac{\mu(d)}{1-2^{d-1}}\right) + \mathcal{O}\left(n2^{-n/2}\right). \end{aligned}$$

We define α by

$$\alpha := \sum_{d=2}^{\infty} \frac{\mu(d)}{1-2^{d-1}}.$$

This concludes the proof. □

Since $f_1(n) = g_1(n) - g_1(n-1)$ we also obtain a good asymptotic estimate for the number of unary minimal DFAs.

Corollary 3.7. The number of distinct unary minimal DFAs with n states is

$$f_1(n) = 2^{n-1}\left(n + 1 - \alpha + \mathcal{O}\left(n2^{-n/2}\right)\right).$$

3.2 Alphabets with more than one letter

As it becomes more and more difficult to count automata with larger input alphabets, we will now focus on lower and upper bounds for $f_k(n)$ and $g_k(n)$.

3.2.1 Lower bounds

We start with a simple constructive lower bound for $f_k(n)$ as given in [5].

Theorem 3.8. The numbers $f_k(n)$ satisfy for all n the inequality

$$f_k(n) \geq f_1(n)n^{(k-1)n}.$$

Proof. Every minimal DFA with n states and alphabet $\Sigma = \{0\}$ can be extended to the alphabet $\{0, 1, ..., k-1\}$ by extending its transition function δ. This can be done in $n^{(k-1)n}$ ways, since for every n states there are $(k-1)$ remaining letters and there are n possible images (states) to choose from. □

Domaratzki, Kisman and Shallit [5] give an even tighter bound, by varying the letter for the unary DFA and taking care of double-counted automata.

Theorem 3.9. The numbers $f_k(n)$ also satisfy for all n the inequality

$$f_k(n) \geq f_1(n)\left(kn^{(k-1)n} - \binom{k}{2}n!n^{(k-2)n}\right).$$

Proof. Like in the previous theorem we construct a minimal DFA, but now we observe reductions to every letter in $\Sigma = \{0, 1, ..., k-1\}$. This gives us $f_1(n)kn^{(k-1)n}$ automata, but we double-count the automata whose restriction is minimal for more than one letter. The number of those automata is at most $f_1(n)\binom{k}{2}n!n^{(k-2)n}$, that is: $\binom{k}{2}$ choices for two letters $i < j$, $f_1(n)$ ways of constructing a minimal DFA with the letter i, $n!$ possibilities for the j-transitions, such that the restriction to j is initially connected and $n^{(k-2)n}$ ways of defining the transition function for the remaining letters. □

Note that not all of these constructed automata, which are enumerated by the expression $f_1(n)\binom{k}{2}n!n^{(k-2)n}$, have minimal restrictions for the two chosen letters. With a little more effort we can further improve this bound given by *Domaratzki, Kisman and Shallit* [5] using the well known principle of inclusion and exclusion and describing a class of minimal automata.

Corollary 3.10. A lower bound for $f_k(n)$ is given by

$$f_k(n) \geq f_1(n) \cdot \frac{1}{n!}\left(n^{nk} - (n^n - n!)^k\right) =: r_k(n),$$

where $r_k(n)$ denotes the number of DFAs for which the restriction to at least one letter is initially connected and the restriction to the smallest letter according to $\Sigma = \{0, 1, 2, ..., k-1\}$ is minimal.

Proof. Interpreting the expression $f_1(n)\binom{k}{2}n!n^{(k-2)n}$ as the number of DFAs where the restriction to at least two letters is initially connected and the restriction of the smallest of those letters is minimal, we see that the given bound in the previous theorem contains just the first terms of the principle of inclusion and exclusion. Thus, the automata with said properties are enumerated

as follows:

$$
\begin{aligned}
&r_k(n) = \\
&= f_1(n)\left(\binom{k}{1}n^{n(k-1)} - \binom{k}{2}(n!)^1 n^{n(k-2)} + \ldots - (-1)^k\binom{k}{k}(n!)^{k-1}n^{n\cdot 0}\right) \\
&= \frac{f_1(n)}{n!}\left(\binom{k}{1}(n!)^1 n^{n(k-1)} - \binom{k}{2}(n!)^2 n^{n(k-2)} + \ldots - (-1)^k\binom{k}{k}(n!)^k n^{n\cdot 0}\right) \\
&= \frac{f_1(n)}{n!}\cdot n^{nk} - \frac{f_1(n)}{n!}\cdot (n^n - n!)^k \\
&= f_1(n)\cdot\frac{1}{n!}\left(n^{nk} - (n^n - n!)^k\right).
\end{aligned}
$$

□

This result is a tighter bound, though it does not improve the asymptotic estimate, as $n! = o(n^n)$ according to Stirling's formula (1.1). Thus, the relevant term in the expression

$$f_1(n)\left(\binom{k}{1}(n!)^0 n^{n(k-1)} - \binom{k}{2}(n!)^1 n^{n(k-2)} + \cdots - (-1)^k\binom{k}{k}(n!)^{k-1}n^{n\cdot 0}\right)$$

is still $f_1(n)kn^{n(k-1)}$.
There is also a direct interpretation of the obtained closed formula. As n^n counts all possible transition functions for one specific letter and $n!$ counts the ones that are initially connected, it follows that there are $(n^n - n!)^k$ transition functions, where the restriction to no letter is initially connected. So there are $(n^{nk} - (n^n - n!)^k)$ transition functions with at least one letter with initially connected restriction. Now, we remove the choices made for the smallest letter with the factor $1/n!$ and replace it with an unary minimal DFA with the factor $f_1(n)$. This includes choosing the final states.

3.2.2 Upper bounds

An upper bound for the number of minimal DFAs can be found by a simple argument given by *Robinson* [21].

Theorem 3.11. Let $f_k(n)$ be the number of distinct minimal DFAs with n states over a k letter alphabet then

$$f_k(n) \leq \frac{2^n n^{kn}}{(n-1)!}.$$

Proof. There are 2^n possible ways of choosing the set of final states and n^{kn} ways of choosing the transition function. We can remove the automata obtained this way that are not initially connected and still have an upper bound, since every minimal DFA is initially connected. For the remaining automata the names of the states do not matter with the exception of the initial state q_0. Thus, we can divide by $(n-1)!$ to not longer distinguish between automata which are isomorphic. □

This bound is easily improved by considering only initially connected transition structures. We write $C_k(n)$ for the number of initially connected deterministic finite automata with n states over a k letter alphabet without final states,

$$f_k(n) \leq \frac{2^n C_k(n)}{(n-1)!}.$$

We see that $\frac{C_k(n)}{(n-1)!}$ is the number of unlabelled initially connected deterministic finite automata without final states. In Chapter 6 we will take a closer look at initially connected deterministic finite automata and obtain formulas for $C_k(n)$ and $\frac{C_k(n)}{(n-1)!}$.

We claim that there is an even stronger result: This upper bound also holds for the number of DFA languages $g_k(n)$. Our claim is, that initially connected DFAs with n states accept all languages with minimal DFAs of size n or smaller. This statement is reduced to the following lemma.

Lemma 3.12. For every DFA with $m < n$ states there is an initially connected DFA with n states which accepts the same language.

Proof. To see this, we take the initially connected component of a DFA and show that we can add one state without violating the initial connectedness.

The underlying graph of every DFA over $k \geq 2$ letters has at least one state with at least two input edges. More precisely, for every DFA $M = (Q, \Sigma, \delta, q_0, F)$ we have

$$\exists q \in Q : \exists (q_i, a_i) \neq (q_j, a_j) : \delta(q_i, a_i) = \delta(q_j, a_j).$$

This holds since the in-degrees d^- and out-degrees d^+ satisfy

$$\sum_{q \in Q} d^-(q) = \sum_{q \in Q} d^+(q) = \sum_{q \in Q} k = nk \geq 2n.$$

Let $\delta(q_i, a_i) = \delta(q_j, a_j)$ like mentioned above. We can now redirect (q_j, a_j) to a new state $\tilde{q}_t$ which is a copy of $q_t = \delta(q_j, a_j)$. Not only do we copy the finality of q_t but also the behaviour of the transition function.

We get an automaton $M' = (Q', \Sigma, \delta', q_0, F')$ with the following construction.

- $Q' := Q \dot{\cup} \tilde{q}_t$, i.e. we add one new state to Q.
- $\delta' := \delta$ with the exception of $\delta'(q_j, a_j) = \tilde{q}_t$. Furthermore we define $\delta'(\tilde{q}_t, a) := \delta(q_t, a)$ for all $a \in \Sigma$.
- If $q_t \in F$ then $F' := F \cup \tilde{q}_t$. Otherwise $F' := F$.

See Figures 3.2 and 3.3 for an example.

For an input alphabet of size $k = 1$ this construction is not always possible. It fails if and only if the unary automaton is just a cycle. W.l.o.g. we have $q_i \cdot a = q_{i+1}$ for $i \in [0, n-2]$ and $q_{n-1} \cdot a = q_0$. In this case we have

$$\delta(q_0, \epsilon) = \delta(q_{n-1}, a) = q_0$$

and we are able to add a copy of q_0 to separate these two transitions like above. □

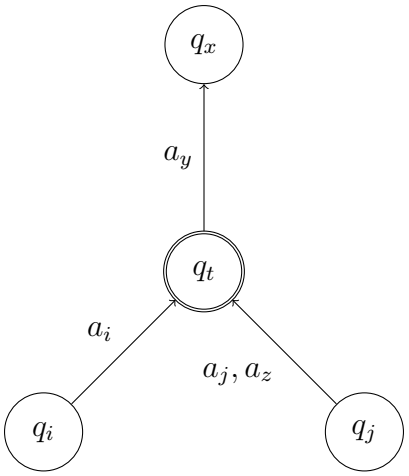


Figure 3.2: Two edges are targeting the same node of the underlying graph of a DFA

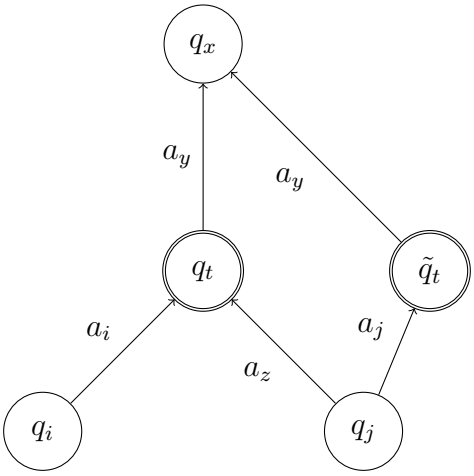


Figure 3.3: One state is added to the DFA, still preserving the initial connectedness

Using this idea we can argue that instead of all the possible transition functions it suffices to observe only initially connected automata without final states, as we do not loose any languages.

Theorem 3.13. Let $C_k(n)$ be the number of initially connected deterministic finite automata with n states over a k letter alphabet without final states, then we have

$$f_k(n) \leq g_k(n) \leq \frac{2^n C_k(n)}{(n-1)!}.$$

Proof. We have $f_k(n) \leq g_k(n)$ as explained in Theorem 3.1 and $g_k(n) \leq \frac{2^n C_k(n)}{(n-1)!}$ as a result of Lemma 3.12. □

Chapter 4

Nondeterministic Finite Automata

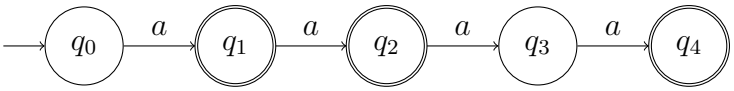
In this chapter we want to focus on nondeterministic automata and their languages. We already know from Subsection 1.1.3 that the regular languages over a fixed alphabet are accepted by DFAs and NFAs respectively. The difference here is the state complexity of a given language.

We will focus on NFA languages and write $G_k(n)$ for the number of different languages accepted by NFAs with n states and alphabet size k. Remember that we used $g_k(n)$ to denote the number of different languages accepted by DFAs with n states and alphabet size k. In this chapter we will present results by *Pomerance, Robson and Shallit* [20] and *Domaratzki, Kisman and Shallit* [5].

4.1 The unary case

4.1.1 Lower bounds

Let $\Sigma = \{a\}$. A simple lower bound for $G_1(n)$ is 2^n as every subset of the powers $\{\epsilon, a, a^2, ..., a^{n-1}\}$ can be accepted by an NFA. The following automaton, for example, accepts the language $L = \{a, a^2, a^4\}$.



We want to improve this lower bound for $G_1(n)$ by constructing specific NFAs. To do so, we need some basic results from number theory.

Theorem 4.1 (Prime number theorem)**.** Let $\pi(x)$ be the number of primes smaller or equal to x. Then

$$\pi(x) \sim \frac{x}{\log x}.$$

A short proof of Theorem 4.1 was found by *D. J. Newman* [27].

Theorem 4.2. Let p_n be the n-th prime number, then

$$p_n \sim n \log n.$$

Proof. As in [11, p.9], we look at the function $y = x/(\log x)$. We get

$$\log y = \log x - \log \log x.$$

Since $\log \log x = o(\log x)$ for $x \to \infty$, we get

$$\log y \sim \log x \quad \text{and} \quad x = y \log x \sim y \log y.$$

Thus, the inverse function to $x/\log x$ is asymptotically equivalent to $x \log x$. Since $\pi(p_n) = n$, the result follows. □

There are tighter bounds for the growth of primes for larger values of n. We refer to *Rosser and Schoenfeld* [22]. We even have

$$n(\log n + \log \log n - 3/2) < p_n < n(\log n + \log \log n - 1/2)$$

for $n \geq 20$. We only need the weaker result

$$p_n < n(\log n + \log \log n) \quad \text{for} \quad n \geq 6$$

for the estimate in the proof of the following theorem, which was observed by *Domaratzki, Kisman and Shallit* [5].

Theorem 4.3. For $G_1(n)$, the number of one letter NFA languages, we get

$$G_1(n) > 2^{n+(2.295-o(1))\sqrt{\frac{n}{\log n}}}.$$

Here the expression $o(1)$ will always represent a positive zero sequence.

Proof. Let $C \geq 2$ be a constant to be determined later. For a given $n \geq 2$ there is a unique whole number b that satisfies

$$bp_{\lfloor Cb \rfloor} \leq n < (b+1)p_{\lfloor C(b+1) \rfloor},$$

since the intervals $[bp_{\lfloor Cb \rfloor}, (b+1)p_{\lfloor C(b+1) \rfloor}[$ are a partition of $\mathbb{N}$. We get

$$\begin{aligned} b+1 &> \frac{n}{p_{\lfloor C(b+1) \rfloor}} \\ &> \frac{n}{C(b+1)\left(\log C(b+1) + \log\log C(b+1)\right)}. \end{aligned}$$

Note that $\log\log x = o(\log x)$. Also since $n \leq p_n$ and $C \geq 2$ we have

$$\begin{aligned} (b+1)p_{\lfloor C(b+1) \rfloor} &\leq n \\ (b+1)C(b+1) &\leq n \\ \sqrt{C}(b+1) &\leq \sqrt{n} \\ \log\sqrt{C} + \log(b+1) &\leq \frac{1}{2}\log n \\ \log(b+1) &\leq \frac{1}{2}\log n. \end{aligned}$$

Using this we can estimate

$$\begin{aligned} b+1 &> \frac{n}{C(b+1)\log C(b+1)\,(1+o(1))} \\ (b+1)^2 &> \frac{n}{C\log C(b+1)\,(1+o(1))} \\ &> \frac{n}{C\left(\log C + \frac{1}{2}\log n\right)(1+o(1))} \\ &= \frac{n}{C\frac{1}{2}\log n\,(1+o(1))}. \end{aligned}$$

We use $(1+o(1))^{-1} = (1-o(1))$ and calculate the square root:

$$b+1 > \sqrt{\frac{2n}{C\log n}}\,(1-o(1))\,.$$

Finally, we get

$$b > \sqrt{\frac{2n}{C\log n}}\,(1-o(1)) - 1 \tag{4.1}$$

$$= \sqrt{\frac{2n}{C\log n}}\left(1-o(1)-\sqrt{\frac{C\log n}{2n}}\right) \tag{4.2}$$

$$= \left(\sqrt{\frac{2}{C}}-o(1)\right)\sqrt{\frac{n}{\log n}}. \tag{4.3}$$

From the first $\lfloor Cb \rfloor$ prime numbers we choose b distinct primes r_i such that $r_1 < r_2 < ... < r_b$. There are $\binom{\lfloor Cb \rfloor}{b}$ ways of doing so. We define an NFA with a tail of length $s := n - (r_1 + r_2 + ... + r_b)$. Note that by construction s is a positive integer. The last state of the tail splits nondeterministically into b cycles with length r_i for the i-th cycle. In Figure 4.1 we present an example for $s = 3$ and primes $r_1 = 2$, $r_2 = 3$ and $r_3 = 5$. The final states for this automaton are yet to be chosen. A word $\omega = a^t$ with $t \in \mathbb{N}$ is accepted either in the tail or in one of the cycles (or not at all). The states of the cycles (for some prime r_i) are labelled with $q_{i,0}, q_{i,1}, ..., q_{i,r_i-1}$ with respect to the order of the cycle. We distinguish the following cases:

- Case 1: $t < s$

  ω is accepted $\Leftrightarrow$ the state q_t is final.

- Case 2: $t \geq s$

  ω is accepted $\Leftrightarrow$ there is one of the chosen primes r_i with $t - s \equiv e \pmod{r_i}$ and $q_{i,e}$ is final.

Now let us consider all possible choices of final states for the tail and the cycles such that for every cycle it holds that

- there is at least one final state and

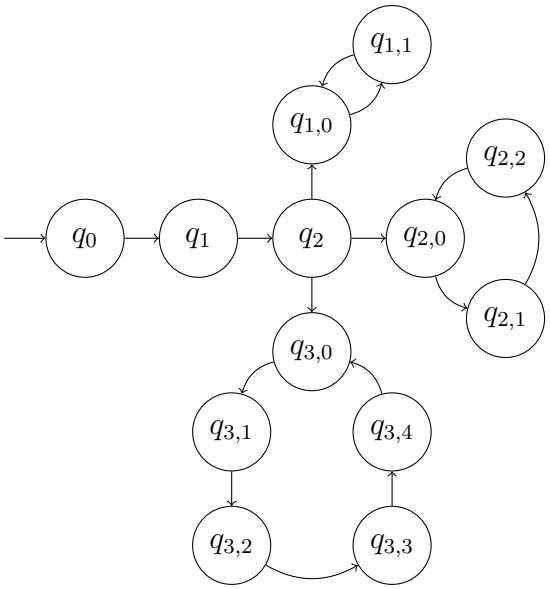


Figure 4.1: An unary NFA with cycles of prime length

- there is at least one nonfinal state,

otherwise the cycle could be reduced. We claim that all of these described n-state automata accept distinct languages.

Suppose that there are two different automata $M = (Q, \Sigma, \delta, q_0, F)$ and $M' = (Q', \Sigma, \delta', q_0', F')$ with cycle lengths $R = \{r_1, ..., r_b\}$ and $R' = \{r_1', ..., r_b'\}$ accepting the same language i.e. $L(M) = L(M')$. We look at the case $R = R'$: We get w.l.o.g. $Q = Q'$ and $\forall i \in \{1, ..., b\} : r_i = r_i'$. Since the two automata are not the same but accept the same language, they have to differ in their finalities of the cycles. Assume that this happens in the cycle with length r_l and w.l.o.g. $q_{l,e} \in F$ but $q_{l,e} \notin F'$. By construction, in every other cycle there is at least one nonfinal state. For $j \neq l$ we have $q_{j,c_j} \notin F'$. By the Chinese remainder theorem there exists a $t \geq s$ (actually there is an infinite number of them) such that

- $t - s \equiv e \pmod{r_l}$ and

- $t - s \equiv c_j \pmod{r_j}$ for $j \neq l$.

Therefore a^t is accepted by M but not by M'. This is a contradiction to $L(M) = L(M')$.
Now we look at the case $R \neq R'$:
W.l.o.g. let $r_l \in R$ and $r_l \notin R'$. Then there is a state $q_{l,e} \in F$ and for all other cycles in M' with length $r'_j \in R'$ there is a state $q'_{j,c_j} \notin F'$. Again we can use the Chinese remainder theorem to get a t with $t > s$ and $t > s'$ such that

- $t - s \equiv e \pmod{r_l}$ and
- $t - s' \equiv c_j \pmod{r'_j}$ for all $r'_j \in R'$.

This concludes the proof of our claim that all these automata accept different languages. Now we will count them to receive the desired lower bound.

Let N be the number of automata in this construction. We had $\binom{\lfloor Cb \rfloor}{b}$ ways of choosing the prime numbers. In the tail every choice of final states is allowed, in a cycle only two choices are forbidden (all final and all nonfinal). This gives

$$2^s(2^{r_1} - 2)(2^{r_2} - 2) \dots (2^{r_b} - 2) = 2^n(1 - 2^{1-r_1})(1 - 2^{1-r_2}) \dots (1 - 2^{1-r_b})$$

possible ways of choosing the final states. We estimate

$$(1 - 2^{1-r_1})(1 - 2^{1-r_2}) \dots (1 - 2^{1-r_b}) \geq \prod_{i \geq 1}(1 - 2^{1-p_i}).$$

Since this infinite product converges to $\beta \approx 0.34564$, we get $N \geq \beta 2^n \binom{\lfloor Cb \rfloor}{b}$. We apply Stirling's approximation formula (1.1) to the binomial coefficient. Of course Cb is not necessarily an integer, so for example by $(Cb)!$ we actually mean the gamma function. We get

$$\begin{aligned}
\binom{\lfloor Cb \rfloor}{b} &= \frac{\lfloor Cb \rfloor!}{b!(\lfloor Cb \rfloor - b)!} > \frac{(Cb-1)!}{b!(Cb-b)!} = \frac{(Cb)!}{Cb \cdot b!(b(C-1))!} \\
&\sim \frac{\sqrt{2\pi Cb} \cdot \left(\frac{Cb}{e}\right)^{Cb}}{Cb \cdot \sqrt{2\pi b}\left(\frac{b}{e}\right)^b \sqrt{2\pi b(C-1)}\left(\frac{b(C-1)}{e}\right)^{b(C-1)}} \\
&= C^{Cb}(C-1)^{b(1-C)}(2\pi C(C-1))^{-1/2} b^{-3/2} \\
&= 2^{b\left[C\log_2 C + (1-C)\log_2(C-1) + 1/b \log_2\left((2\pi C(C-1))^{-1/2} b^{-3/2}\right)\right]}.
\end{aligned}$$

Since $\lim_{b\to\infty} b^{1/b} = 1$, we have

$$N \geq \beta 2^{n+b(\gamma+o(1))},$$

where $\gamma = C\log_2 C + (1-C)\log_2(C-1)$. We can now apply the previously obtained estimate 4.3. The constant β vanishes since $\log_2 \beta\sqrt{\frac{\log n}{n}}$ also converges to zero. We have

$$N \geq 2^{n+(\tilde{\gamma}-o(1))\sqrt{\frac{n}{\log n}}}$$

with $\tilde{\gamma} = \sqrt{2/C}(C\log_2 C + (1-C)\log_2(C-1))$. We choose C such that $\tilde{\gamma}$ is maximized. This gives $C \approx 4.141$ and $\tilde{\gamma} > 2.295$. □

4.1.2 Upper bounds

Before we move on to larger alphabets, we take a look at upper bounds for $G_1(n)$. In this subsection, we will present the bound given by *Pomerance, Robson and Shallit* [20] (with a minor modification), which seems to be the best upper bound known at this time. This is done by finding a decomposition for the NFA languages, which can be described by a few parameters. To get there, we have to introduce some notation first.

Definition 4.4. Let $L \subset \{a\}^*$ be a unary language and $c \in \mathbb{N}$.

- L is c-monotonic :$\Leftrightarrow$

$$\forall n \geq 0 : a^n \in L \Rightarrow a^{c+n} \in L.$$

- L is c-periodic after N :$\Leftrightarrow$

$$\forall n \geq N : a^n \in L \Leftrightarrow a^{c+n} \in L.$$

If a language is c-monotonic, then for every word $\omega \in L$ the words $\omega a^{kc} \in L$ for $k \in \mathbb{N}$. So at some point, for sufficiently big words, the question whether a word $\omega = a^x$ is part of the language L depends only on the value of $x \pmod c$. Given a unary NFA $M = (Q, \Sigma, \delta, q_0, F)$, let us write $G(M)$ for the underlying digraph of M. The transition diagram of M can already be interpreted as

a graph, more precisely, $G(M)$ is a graph with vertices $V := Q$ and edges $E := \{(q_1, q_2) \in Q \times Q : q_2 \in \delta(q_1, a)\}$. Furthermore, we will write $L(M, s)$ for the set of strings having an accepting path that contains the state s. An accepting path is a "witness" for the fact that $\omega \in L$. This can be defined recursively or in the unary case as a path in $G(M)$ consisting of $|\omega| + 1$ nodes starting in q_0 and ending in a final state.

Lemma 4.5. Some properties needed for the desired decomposition:

a) For unary languages L_1 and L_2 we have: If L_1 and L_2 are c-monotonic, so is $L_1 \cup L_2$. If L_1 and L_2 are c-periodic after N, so is $L_1 \cup L_2$.

b) Let M be an NFA with q states, s a state of M and let there exist a directed cycle of length c containing s. Then $L(M, s)$ is c-monotonic. Also, $L(M, s)$ is c-periodic after $(c + 1)(q - 1)$.

Proof. We will only give a proof for the c-periodicity of $L(M, s)$.
Let $a^l \in L = L(M)$ with $l \geq (c+1)q - 1 = (c+1)(q-1) + c$, let s be part of a cycle of length c and let there be an accepting path of a^l which contains s. Let this path be $\mathcal{P} = (p_0, p_1, \ldots, p_l)$. We will show that there is an accepting path through s for the word a^{l-kc} for some positive integer k. This suffices because of the mentioned monotonicity.
Let $p_i = s$ be the first occurrence of s in the path $\mathcal{P}$. We split the path in prefix $P = (p_0, \ldots, p_i = s)$ and suffix $S = (p_i = s, \ldots, p_l)$. Together they consist of $l + 2 > (c + 1)q$ states. Now let p' and s' be two states which occur most frequently in P respectively S. Note that the sum of the occurrences of p' and s' is at least $c + 2$. If two of these nodes in the path P are separated by a subpath containing exactly $k \equiv 0 \pmod{c}$ edges, we can cut out this cycle and receive an accepting path of a^{l-kc} through s.

If we do not find such a subpath, consider the following. Cut out the cycle from any occurrence of p' to the last occurrence of p' in P. These cycles all have different length (number of edges) modulo c. If the lengths were in the same residue class, their difference would be a cycle with length $k \equiv 0 \pmod{c}$, so we would have found that cycle in the last step. We cut out the cycle from the

first occurrence of s' in S to any other occurrence. Again, the lengths are all in different residue classes mod c. Let the length of the cycle we removed from P be d and the length of the cycle we removed from S be e. None of these lengths are congruent 0 (mod c). Let v_k be the number possible values for k (mod c) and v_e the number of possible values for $-e$ (mod c). By construction it holds that $v_k + v_e \geq c$. By the pigeon hole principle there are two cycles in P and S with $d + e \equiv 0 \pmod{c}$. Thus, removing those finally results in an accepting path of a^{l-kc} through s. □

For our next step towards the upper bound we can make use of a result observed by *S. Rao Kosaraju* [15] that states the following.

Lemma 4.6. In an undirected graph the following statement is true: If every 3 circuits have a vertex in common, then there exists a vertex that is part of every circuit.

We say that a directed graph is *of girth c* if and only if every directed cycle is at least of length c and c is the maximal value with this property. If the graph is acyclic, we define its girth as ∞. In combination with Lemma 4.6 this gives:

Lemma 4.7. Let G be a directed graph with q vertices. If the girth of G is larger than $2q/3$, then there exists at least one vertex that is part of every cycle.

Proof. If every cycle has more than $2q/3$ vertices, every two cycles have more than $2q/3-(q-2q/3) = q/3$ vertices in common. Thus, every three cycles have more than $q/3 - (q - 2q/3) = 0$ vertices in common. The result follows. □

Now we are ready to formulate the decomposition.

Theorem 4.8. Let M be a unary nondeterministic finite automaton with q states. Then there exists an integer $r \geq 0$, a strictly increasing sequence $c_1 < \cdots < c_r$, languages $L_1, \ldots, L_r$ and an NFA M_{r+1} such that

$$L(M) = \left(\bigcup_{1 \leq i \leq r} L_i \right) \cup L(M_{r+1})$$

and the languages L_i are c_i-monotonic and c_i-periodic after $(c_r+1)(q-1)$. In addition, if M_{r+1} has q' states, then the girth of $G(M_{r+1})$ is larger than $2q'/3$. If $r \geq 1$ then $q' \geq c_r/2$. It holds that $q' = q - (c_1 + \cdots + c_r)$.

Proof. We give a recursive algorithm to decompose $L(M_i)$. Let n_i be the number of states in M_i and c_i be its girth. If the girth of M_i is larger than $2n_i/3$, our algorithm terminates with $r = 0$ and $M_i = M$. Note that this is also the case when $c_i = \infty$. In every other case we further decompose

$$L(M_i) = L_i \cup L(M_{i+1}),$$

where L_i is the language of all words ω for which there exists an accepting path containing a cycle of length c_i. The automaton M_{i+1} is obtained by removing all states in all cycles of length c_i. This reduction results in the disjoint union $L(M_i) = L_i \dot{\cup} L(M_{i+1})$. The language L_i is c_i-monotonic and c_i-periodic. This can be shown using the previous lemma. We may need the union of more than one c_i-monotonic (and c_i-periodic) language, as we may have to remove more than one cycle. Lemma 4.5 also gives us the c_i-periodicity after $(c_r+1)(q-1)$. If necessary, we can construct M_{i+1} in a way so that it has exactly $n_i - c_i$ states as we remove at least c_i states and can leave some states disconnected as "dummy states". Therefore $q' = q - (c_1 + \cdots + c_r)$ holds. If we reach the termination condition $c_{r+1} > 2q'/3$ we have $c_r \leq 2n_r/3$ and $q' = n_r - c_r$, thus $q' \geq 3c_r/2 - c_r = c_r/2$, which concludes the proof. □

We have not used Lemma 4.6 yet, but we will need it soon.

First, we claim that with the described decomposition, a unary NFA language is completely specified given

1) the integers $c_1 < \cdots < c_r$,

2) for $1 \leq i \leq r$ and $0 \leq j \leq c_i$ ($0 \leq j < c_i$ already covers all the congruence classes) the information whether or not there exists n such that $n \equiv j \pmod{c_i}$ and $a^n \in L_i$,

3) for $1 \leq i \leq r$ and $0 \leq j \leq c_i$ the cardinality of

$$L_{ij} := \left\{ a^n \in L_i \setminus \bigcup_{1 \leq t < i} L_t : n < (c_r + 1)(q-1) \wedge n \equiv j \pmod{c_i} \right\}$$

4) and the residual language $L(M_{r+1})$.

We know from Lemma 4.5 that all languages L_i are c_i-monotonic and c_i-periodic after $(c_r + 1)(q - 1)$, so it suffices to know the words of L_i with length less than $(c_r + 1)(q - 1)$ and the congruence classes modulo c_i that are eventually covered by L_i. Actually it is enough to know the first occurrence of a word from some congruence class, as the later occurrences are "produced" by the c_i-monotonicity. Thus, we only have to specify for each $j < c_i$ the shortest word with length congruent to $j \pmod{c_i}$. This has to be done only if the word was not mentioned for some other language L_t with $t < i$. It follows that L_i is determined by the cardinality given in 4).

Our goal is find upper bounds for the possible choices of the parameters 1), 2), 3) and 4). This will allow us to give an upper bound for $G_1(n)$. We start with the following observation:

Lemma 4.9. 1) The integers $c_1 < \cdots < c_r \leq q$ are a subset of $\{1, 2, \ldots, q\}$, thus we have 2^q possibilities to choose them.
2) For each c_i we have to decide c_i times if there exists such a word, therefore we have

$$2^{\sum_{i=1}^{r} c_i} = 2^{q-q'}$$

possibilities.

We move on to 3), the cardinalities of the sets L_{ij}. In order to get an upper bound, we have to prove the following lemma.

Lemma 4.10. Let $q > 0$ and let q' be the local extremum of the function

$$f : [0, q] \to \mathbb{R} : x \mapsto x^{q-x}.$$

Then it holds that

$$q' = \mathcal{O}\left(\frac{q}{\log q}\right)$$

for $q \to \infty$.

Proof. We differentiate f and want to find the zero of the derivation

$$f'(x) = x^{q-x} \cdot \left(-\log(x) + \frac{q-x}{x}\right) = x^{q-x} \cdot \left(\frac{q}{x} - 1 - \log(x)\right).$$

We proceed to solve for x,

$$\begin{aligned} 0 &= \frac{q}{x} - 1 - \log x \\ \log x &= \frac{q}{x} - 1 \\ x &= e^{\frac{q}{x}-1} \\ qe &= \frac{q}{x} e^{\frac{q}{x}} \\ W_0(qe) &= \frac{q}{x} \\ x &= \frac{q}{W_0(qe)}. \end{aligned}$$

The function W_0 is the Lambert-W function mentioned in Section 2.2. It remains to be shown that

$$\frac{q}{W_0(qe)} = \mathcal{O}\left(\frac{q}{\log q}\right).$$

Let $\alpha > 1$. Then we have

$$\frac{\log q}{\alpha} \cdot q^{\frac{1}{\alpha}-1} \leq e$$

for q sufficiently large. Furthermore, this gives:

$$\begin{aligned}\frac{\log q}{\alpha} \cdot q^{\frac{1}{\alpha}-1} \leq e &\Leftrightarrow \frac{\log q}{\alpha} \cdot e^{\frac{\log q}{\alpha}} \leq qe \\ &\Leftrightarrow \frac{\log q}{\alpha} \leq W_0(qe) \\ &\Leftrightarrow \frac{q}{W_0(qe)} \leq \alpha \cdot \frac{q}{\log q} \\ &\Leftrightarrow \frac{q}{W_0(qe)} = \mathcal{O}\left(\frac{q}{\log q}\right).\end{aligned}$$

This concludes the proof. □

Lemma 4.11. There are less than $(Cq/\log q)^q$ possibilities for the parameters in 3).

Proof. We know that there are

$$\binom{m+n-1}{n-1}$$

ways of choosing n (ordered) integers which sum to the value m. Therefore, there are

$$\binom{m+n}{n}$$

ways of choosing $n+1$ integers which sum to m. We can also reinterpret it by dropping the last integer to get the number of choices for n integers which sum to a value $\leq m$. We estimate

$$\binom{m+n}{n} < \frac{(m+n)^n}{n!}.$$

For every c_i and for every congruence class we have to choose one integer, i.e. $q - q'$ integers that sum up to at most $(c_r + 1)(q-1)$. Using the estimate we have at most

$$\frac{((c_r+1)(q-1)+q-q')^{q-q'}}{(q-q')!}$$

possibilities. We have shown already that $c_r \leq 2q'$. With this inequality and Stirling's approximation formula we have

$$\begin{aligned}\frac{((c_r+1)(q-1)+q-q')^{q-q'}}{(q-q')!} &= \frac{(2qq'+2q-3q'-1)^{q-q'}}{(q-q')!} \\ &\leq \frac{(4qq')^{q-q'}}{(q-q')!} \\ &\sim \frac{(4eqq')^{q-q'}}{(q-q')^{q-q'}\sqrt{2\pi(q-q')}} \\ &\leq \left(\frac{cqq'}{q-q'}\right)^{q-q'} =: B\end{aligned}$$

as an upper bound with $c = 4e$. If $q' > 2q/3$ then

$$B \leq (cq^2)^{q/3} = (c^{1/3}q^{2/3})^q = \mathcal{O}\left(\frac{c^{1/3}q}{\log q}\right)^q.$$

Notice that we can use the notation $\mathcal{O}$ once the term only depends on q. The other case is $q' < 2q/3$, where

$$B \leq \left(\frac{cqq'}{\frac{1}{3}q}\right)^{q-q'} = (3cq')^{q-q'} \leq (3c)^q \cdot (q')^{q-q'}.$$

We try to maximize the term $(q')^{q-q'}$. Through differentiation one can see that the optimal choice for q' satisfies $q' = \mathcal{O}(q/\log q)$ for $q \to \infty$. We showed this in Lemma 4.10. We have

$$B \leq (3cq')^q \leq \left(\frac{3c\alpha q}{\log q}\right)^q.$$

In every case our bound satisfies $B \leq (Cq/\log q)^q$ for some constant C. The result follows. □

This leaves us with the different choices for the residual language 4).

Lemma 4.12. There are less than 2^{10q} choices for the residual language $L(M_{r+1})$ in 4).

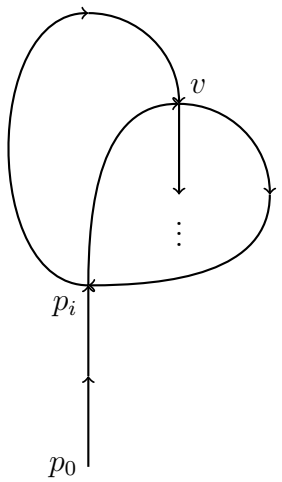


Figure 4.2: Every cycle contains the state v.

Proof. If the graph $G(M_{r+1})$ is acyclic, then the language $L(M_{r+1})$ is finite and there are $2^{q'}$ possibilities to choose it. Otherwise the girth of $G(M_{r+1})$ is larger than $2q'/3$. Thus, with Lemma 4.7 there is one vertex that is part of every cycle. We split the language $L(M_{r+1}) = A \dot{\cup} B$ with

$$A := \{a^n | n < q' \wedge a^n \in L(M_{r+1})\}, \quad B := \{a^n | n \geq q' \wedge a^n \in L(M_{r+1})\}.$$

There are $2^{q'}$ possibilities for A. Let ω be a word in B and $(p_0, p_1, \ldots, p_f)$ an accepting path of ω. Let p_i be the first state to be repeated. Because of Lemma 4.7 there is a vertex v in this cycle, that is also part of every other cycle. Starting from this vertex v we take the remaining accepting path $(v, \ldots, p_f)$. If its length is still larger than or equal to q' we can again apply the pigeon hole principle to find the first repeating state in this path. It has to be v, since v lies in every cycle. See Figure 4.2. Therefore every accepting path for a word $\omega \in B$ can be split into

a) an initial part of length less than q';

b) a concatenation of cycles; note that two of these cycles can have more than one state in common and the set of these cycles can also be empty;

c) a tail of length less than q'.

These accepting paths are specified when given

i) the different lengths of acyclic paths from the state p_0 to v, which form a subset of $\{0, 1, \ldots, q' - 1\}$;

ii) the different cycle lengths as a subset of $]2q'/3, q']$;

iii) the different lengths of acyclic paths from v to any final state as a subset of $\{0, 1, \ldots, q' - 1\}$.

This gives at most $2^{q'} \cdot 2^{q'/3} \cdot 2^{q'} = 2^{7q'/3}$ possibilities for B. We multiply with the $2^{q'}$ choices for A and get

$$2^{10q'/3} + 2^{q'} \leq 2^{10q}$$

possibilities for $L(M_{r+1})$, which include the acyclic graphs too. □

Finally we can put things together to get an upper bound for $G_1(n)$.

Theorem 4.13. [20] The number of distinct unary languages accepted by NFAs with n states $G_1(n)$ satisfies

$$G_1(n) \leq \left(\frac{cn}{\log n}\right)^n$$

for some constant c.

Proof. To sum up: For the four parts of the decomposition we have the upper bounds

$$2^q, \quad 2^{q-q'}, \quad (Cq/\log q)^q, \quad 2^{10q}$$

as observed in Lemma 4.9, Lemma 4.11 and Lemma 4.12. Multiplying them gives the desired result. With $q' \leq q$, the product is

$$G_1(n) \leq 2^n \cdot 2^n \cdot \left(\frac{Cn}{\log n}\right)^n \cdot 2^{10n} \leq \left(\frac{cn}{\log n}\right)^n$$

for some constant $c > 0$. □

4.2 Alphabets with more than one letter

Let $G_k(n)$ be the number of distinct languages accepted by automata with n states and alphabet size $k \geq 2$.

4.2.1 Lower bounds

For a lower bound, we can use a similar approach as in Theorem 3.8. Again we follow the work of *Domaratzki, Kisman and Shallit* [5].

Theorem 4.14. For $k \geq 2$ we have

$$G_k(n) \geq n2^{(k-1)n^2}.$$

Proof. We build an automaton with n states and alphabet $\Sigma = \{0, 1, \ldots, k-1\}$ in such a way, that the reduction to the letter 0 is one single loop. Exactly one state is chosen to be final. The transition function for the letters $\{1, \ldots, k-1\}$ are chosen arbitrarily. We refer to Figure 4.3 for the transition diagram of a specific automaton with 12 states constructed in this way.

More precisely we define for $Q = \{q_0, q_1, \ldots, q_{n-1}\}$ the transition function

$$\delta(q_l, 0) := q_{(l+1 \mod n)} \quad \text{and for } j \neq 0 \quad \delta(q_l, j) := H_{l,j}$$

with $H_{l,j} \subset Q$ any subset of states and the set of final states $F := \{q_i\}$ for some i. We claim that two distinct automata built this way accept different languages. We see that the final states cannot differ because if they would, the restricted languages to $\{0\}$ would differ, therefore the accepted languages could not be the same.

Now let $M = (Q, \Sigma, \delta, q_0, F)$ and $M' = (Q', \Sigma, \delta', q_0', F')$ be two different automata (obtained by the mentioned construction) with both state sets $Q = Q' = \{q_0, q_1, \ldots, q_{n-1}\}$ and $F = F' = \{q_i\}$. Speaking in terms of the transition diagram: The two diagrams differ, therefore w.l.o.g. there is at least one edge in M that is not in M'. Let this edge connect the states q_a and q_b labelled with the letter $\alpha \neq 0$. We have $q_b \in \delta(q_a, \alpha)$ but $q_b \notin \delta'(q_a, \alpha)$. We can now

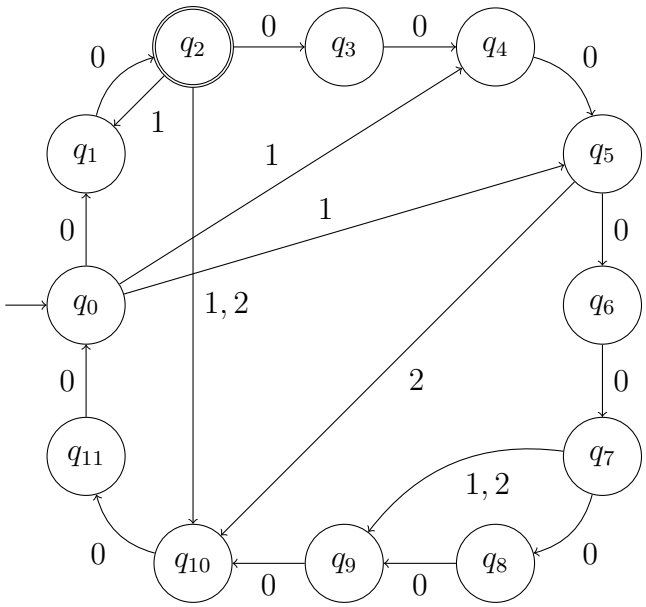


Figure 4.3: A constructed NFA with 12 states and alphabet size $k = 3$

give a word

$$\omega := 0^a \alpha 0^{(i-b \mod n)}$$

with

$$\omega \in L(M) \quad \text{but} \quad \omega \notin L(M').$$

Thus, two distinct automata accept different languages.

We have n possibilities to choose the final state. For the transition functions we have to choose $(k-1)n$ times one of the 2^n possible subsets of states. This leaves $n2^{(k-1)n^2}$ choices. □

4.2.2 Upper bounds

Theorem 4.15. The number $G_k(n)$ of distinct languages accepted by an NFA with n states over a k-letter alphabet satisfies

$$G_k(n) \leq (2n-1)2^{kn^2} + 1.$$

Proof. We count the possibilities for the transition function $\delta : Q \times \Sigma \to 2^Q$. There are $2^{|Q|\cdot|Q|\cdot|\Sigma|} = 2^{kn^2}$ such functions. Now let us choose the final states. If there is at least one final state, then q_0 is either final or nonfinal and the other final states are $\{q_1, q_2, \ldots, q_t\}$ with $t \leq n-1$. This holds w.l.o.g., as the names of all states except q_0 are interchangeable. If q_0 is final, t can be any number from 0 to $n-1$. If q_0 is not final, t can range from 1 to $n-1$. That gives $n+(n-1) = 2n-1$ possibilities for the final states, if there is at least one. The only case left is the NFA without any final states. This gives one additional language, the empty language. Therefore we have $(2n-1)2^{kn^2}+1$. □

4.3 NFA vs. DFA

As every DFA can be interpreted as an NFA, more languages are recognized by NFAs with n states than by DFAs with n states. We want to put the state complexity of DFA and NFA languages further into relation [5, p.11].

Theorem 4.16. Let Σ be an alphabet of size ≥ 2 and let $n \geq 2$. Then there are at least 2^{n-2} distinct languages L over the alphabet Σ such that

- L has an NFA acceptor with n states and
- the minimal DFA of L has 2^n states.

Proof. W.l.o.g. we have $\{a, b\} \subset \Sigma$. We construct 2^{n-2} NFAs with n states that accept different languages which have minimal DFA acceptors of size 2^n. We will only use the letters a and b since every language $L \subset \{a, b\}^*$ is also a language over the alphabet Σ. For every subset $S \subset \{q_1, \ldots, q_{n-1}\}$ we define an NFA $M(S) = (Q, \Sigma, \delta_S, q_0, F)$ with

$$Q = \{q_0, q_1, \ldots, q_{n-1}\}$$
$$\Sigma = \{a, b\}$$
$$F = \{q_{n-1}\}$$

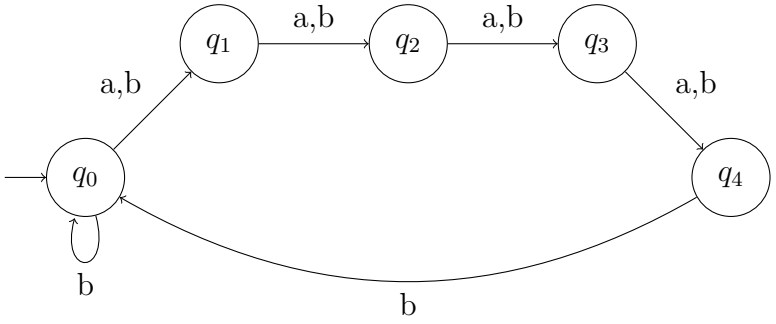


Figure 4.4: The NFA $M(S)$ for $n = 5$ and $S = \emptyset$

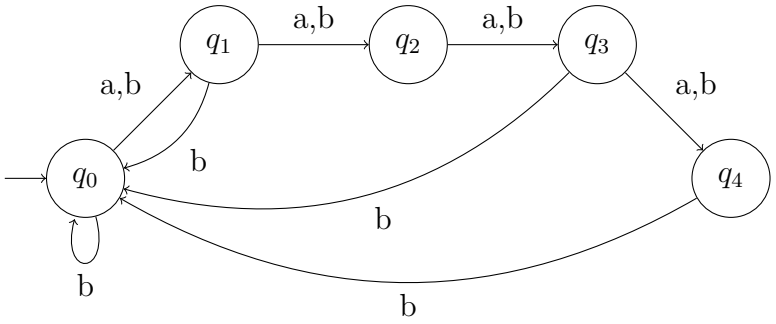


Figure 4.5: The NFA $M(S)$ for $n = 5$ and $S = \{q_1, q_3\}$

and the transition function

$$\delta_S(q_i, a) = \begin{cases} \{q_{i+1}\} & \text{for} \quad 0 \leq i < n-1 \\ \emptyset & \text{for} \quad i = n-1 \end{cases}$$

$$\delta_S(q_i, b) = \begin{cases} \{q_0, q_1\} & \text{for} \quad i = 0 \\ \{q_0, q_{i+1}\} & \text{for} \quad 1 \leq i \leq n-2 \wedge q_i \in S \\ \{q_{i+1}\} & \text{for} \quad 1 \leq i \leq n-2 \wedge q_i \notin S \\ \{q_0\} & \text{for} \quad i = n-1. \end{cases}$$

See Figure 4.4 and Figure 4.5 for an example. We remember the corresponding DFA explained in Subsection 1.1.3 and claim the following.
For every word of the form $b^{n-1}w$ with $|w| = n$, we reach a different subset of Q. Thus, every state of the corresponding DFA is reachable.
When reading b^{n-1}, every state of $M(S)$ is reached. After this, reading the

letter a results in a right shift of the reachable states, making q_0 unreachable. Reading the letter b results in a cyclic permutation in the same direction whereupon q_0 is reachable. Thus, the 2^n words of the form $b^{n-1}w$ correspond to the 2^n states in the DFA. Note that this holds for every set S. To prove the minimality of the DFA we have to show that no two sets of states are equivalent. Let $T, U \subset Q$ be two states of the DFA with $T \neq U$. W.l.o.g. we have $q_i \in T$ and $q_i \notin U$ for some i. Then $\delta_S(T, a^{n-1-i})$ contains the final state q_i and is therefore a final state of the DFA, but $\delta_S(U, a^{n-1-i})$ does not contain the final state.

Now we have to show that for two different subsets $S \neq S'$ the automata $M(S)$ and $M(S')$ accept different languages. Again, w.l.o.g., let $q_i \in S$ but $q_i \notin S'$ for some $1 \leq i \leq n-2$. Then the word $w = a^i b a^{n-1}$ is accepted by $M(S)$ but not by $M(S')$. □

Chapter 5

Finite Languages and Their Acceptors

The acceptor of a finite language cannot contain a cycle with a final state. This fact keeps the structure of these automata simple to a certain degree. For the unary case explicit formulas can be obtained. We write $f'_k(n)$ for the number of minimal DFAs accepting a finite language, $g'_k(n)$ for the number of distinct finite languages accepted by some DFA and $G'_k(n)$ for the number of distinct finite languages accepted by some NFA, always observing automata with n states and input alphabet size k. The bounds we give were observed by *Domaratzki, Kisman and Shallit* [5].

5.1 Finite language DFAs

For the unary case we make use of Theorem 3.2.

Theorem 5.1. For a one letter alphabet we get $f'_1(1) = 1$, $f'_1(n) = 2^{n-2}$ for $n \geq 2$ and $g'_1(n) = 2^{n-1}$.

Proof. With only one state, a unary DFA can only accept $\emptyset$ and $\{a\}^*$. Only one of these languages is finite.

Since a unary DFA always consists of a tail and a cycle, the cycle has to be of size one, otherwise the DFA would accept an infinite language. This last

state must be nonfinal and the state before this sink has to be final (because of the minimality). For the remaining $n-2$ states the final states can be chosen at will and produce 2^{n-2} automata.

We use $g'_1(n) = f'_1(1)+f'_1(2)+\cdots+f'_1(n)$ and the geometric series to obtain $g'_1(n) = 2^{n-1}$. □

Before we move on to larger alphabets we have to better understand the structure of finite language DFAs.

Lemma 5.2. If M is a DFA with $n \geq 2$ states accepting a finite language, then it is isomorphic to a DFA $M' = (Q, \Sigma, \delta, q_0, F)$ with the following properties:

a) $\delta(q_{n-1}, a) = q_{n-1}$ for all letters $a \in \Sigma$,

b) $\delta(q_{n-2}, a) = q_{n-1}$ for all letters $a \in \Sigma$,

c) $q_{n-1} \notin F$,

d) $q_{n-2} \in F$,

e) $\delta(q_i, a) = q_j$ with $i < n-1$ implies $i < j$.

Note that for $n = 1$ only a), c) and e) hold. The properties a) to d) specify that there is one *dead* state which reaches only itself and is not final. Furthermore there is a *pre-dead* state (see also [17]) which is final and reaches only the dead state. Property e) is possible because the DFA has no cycle including a final state.

Proof. Since M is minimal, it has to be initially connected. We discard the states from which a final state cannot be reached. There is at least one such state, otherwise there would be a cycle containing a final state. Actually, there is exactly one such state, since two of these states would be equivalent, contrary to the assumption that M is minimal. The remaining graph is acyclic. Therefore we can rename the states in such a way that e) holds. Now we add the sink again with the name q_{n-1}, which then satisfies a) and c) and we get a

minimal automaton M'. The conditions b) and d) remain to be shown. Since transitions from q_{n-2} only reach higher numbered states, they have to point to q_{n-1}. If q_{n-2} was not final, it would be equivalent to the state q_{n-1}. Thus, M' would not be minimal. □

Theorem 5.3. Let $k \geq 2$ and $n \geq 2$, then we have

$$f'_k(n) \geq 2^{n-2} \left((n-1)!\right)^{k-1}.$$

Proof. We construct different automata with states $Q = \{q_0, q_1, \ldots, q_{n-1}\}$ such that the restriction to the letter 0 is of the form

$$\delta(q_i, 0) = q_{i+1} \quad \text{for} \quad 0 \leq i \leq n-2 \quad \text{and} \quad \delta(q_{n-1}, 0) = q_{n-1}.$$

Furthermore we choose the final states to be any subset $F \subset Q$ with $q_{n-1} \notin F$ and $q_{n-2} \in F$. There are 2^{n-2} ways of doing so. To satisfy e) from Lemma 5.2 we have $(n-1)!$ possible ways of choosing the transition function for a single letter, since the index of the state that is pointed to has to always be larger. Two such automata accept distinct languages and are minimal because of the minimality of the restriction to 0. □

The characterization of minimal finite language DFAs in Lemma 5.2 would give us an upper bound of $f'_k(n) \leq 2^{n-2}((n-1)!)^k$ but the bound obtained in Theorem 3.11 is tighter in terms of asymptotics. The bound

$$f'_k(n) \leq f_k(n) \leq \frac{2^n n^{kn}}{(n-1)!} = o\left(2^{n-2}((n-1)!)^k\right)$$

is indeed the best bound known so far.

5.2 Finite language NFAs

Moving on to NFAs accepting finite languages, we get an exact formula for the unary case and bounds for larger alphabets.

Theorem 5.4. We have $G'_1 = 2^n$ and for $k \geq 2$ we get

$$2^{(k-1)n(n-1)/2} \leq G'_k(n) \leq 2^{n-1+kn(n-1)/2}.$$

Proof. As we observed in Subsection 4.1.1, every subset of $\{\epsilon, a, a^2, \ldots, a^{n-1}\}$ can be accepted by an NFA with n states, so $G'_1(n) \geq 2^n$. If such an NFA accepted a word w with $|w| > n$ then the accepting path would contain a cycle thus the accepted language would not be finite. Thus $G'_1 = 2^n$.

For the bounds on $G'_k(n)$, we assume that transitions go only from lower labelled states to higher labelled states. This is possible by a similar argument as in Lemma 5.2. In addition, the state q_{n-1} is final, otherwise all edges pointing towards q_{n-1} could be removed. The state q_i has $n - i - 1$ possible transitions to higher labelled states for each letter. This gives us

$$\prod_{i=0}^{n-1} 2^{k(n-1-i)}$$

ways of choosing the transition function and 2^{n-1} ways of choosing the remaining final states. As an upper bound we get

$$2^{n-1} \prod_{i=0}^{n-1} 2^{k(n-1-i)} = 2^{n-1+\sum_{i=0}^{n-1} k(n-1-i)} = 2^{n-1+k\sum_{j=0}^{n-1} j} = 2^{n-1+kn(n-1)/2}.$$

For the lower bound we use a similar technique as in the proof of Theorem 4.14. We construct different automata accepting distinct languages. Let $M = (Q, \Sigma, \delta, q_0, F)$ be an automaton with the following properties:

- $Q = \{q_0, q_1, \ldots, q_{n-1}\}$
- $\Sigma = \{0, 1, \ldots, k-1\}$
- Transitions only occur towards higher labelled states.
- $\delta(q_i, 0) = \{q_{i+1}\}$ for $i \neq n-1$
- $F = \{q_{n-1}\}$.

The transitions for the letters 1 to $k-1$ are chosen in all possible (valid) ways. Again, we see that the accepted languages are distinct for two of these automata. If we have $q_{a+m} \in \delta(q_a, \alpha)$ but $q_{a+m} \notin \delta'(q_a, \alpha)$ for transition functions δ and δ' of two different automata and $\alpha \neq 0$, then we can give a word

$$\omega := 0^a \alpha 0^{n-1-a-m}$$

that is accepted by one of these automata but not by the other. Now the number of all the possible transition functions can be obtained. There are $\binom{n}{2}$ edges pointing to a higher labelled state, thus for each of the $k-1$ remaining letters we decide whether or not to include this edge. Thus,

$$G'_k(n) \geq 2^{(k-1)n(n-1)/2}.$$

□

Chapter 6

Initially Connected DFAs

An important class of deterministic finite automata is the class of initially connected DFAs. In these automata every state is reachable by some word starting from q_0. This condition is quite reasonable, since we can always cut away nonreachable parts of an automaton without altering its behaviour. Initial connectedness is not as strong as the condition to be minimal, since there are still initially connected DFAs that are not minimal.
As an example we have three automata accepting the language

$$\mathcal{L} = \{0,1\}^* \setminus \{\epsilon, 0, 1\}\,.$$

The automata shown in Figure 6.1, Figure 6.2 and Figure 6.3 all accept the language $\mathcal{L}$ but have different properties.

Throughout this chapter we will study initially connected DFAs with and without final states, in short ICDFAs and ICDFA$_\emptyset$s. We will write

- $B_k(n)$ for the number of nonisomorphic ICDFA$_\emptyset$s, or for the number of ICDFAs with unlabelled states and

- $A_k(n) = 2^n B_k(n)$ for the number of nonisomorphic ICDFAs

with n and k representing the number of states and the alphabet size.

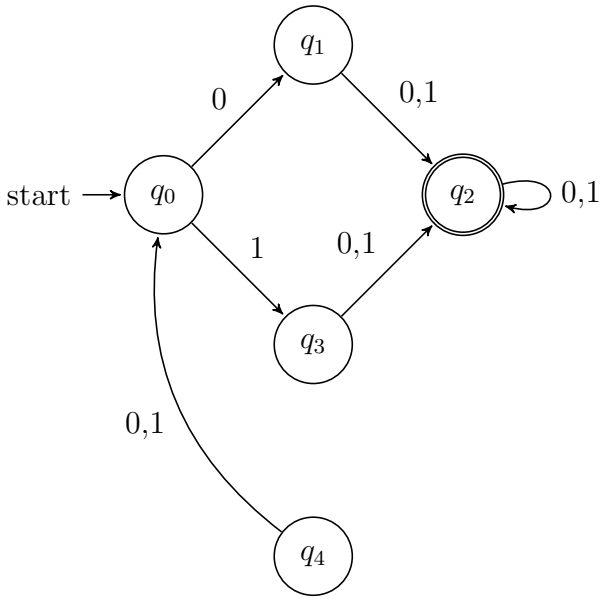


Figure 6.1: A not initially connected DFA

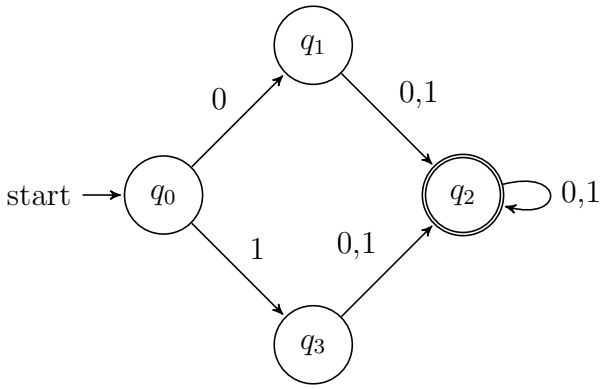


Figure 6.2: A not minimal but initially connected DFA

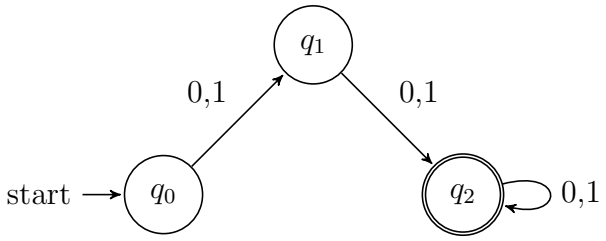


Figure 6.3: A minimal DFA

6.1 Representation and enumeration

Both *Robinson* [21] and *Liskovets* [16] independently gave the following formula.

Theorem 6.1. For the number of nonismorphic ICDFA$_\emptyset$s $B_k(n)$ it holds that

$$B_k(n) = \frac{C_k(n)}{(n-1)!}$$

with

$$C_k(n) = n^{kn} - \sum_{1 \le j < n} \binom{n-1}{j-1} C_k(j) n^{k(n-j)},$$

where $C_k(n)$ is the number of ICDFA$_\emptyset$ transition structures, i.e. ICDFAs without final states but not necessarily nonisomorphic.

Proof. There are n^{kn} different transition structures. We subtract those that are not initially connected, to be more precise, all automata with an initially connected component of size $< n$. If the initally connected component is of size j, the state q_0 has to be part of the initially connected component. The remaining $j-1$ states are chosen from the $n-1$ other states. Then there are $C_k(j)$ initially connected transition structures to connect the chosen j states. The transition function for the remaining $n-j$ states is chosen in every possible way, giving $n^{k(n-j)}$ possibilities. Finally $C_k(n)$ can be divided by $(n-1)!$ since the names of the states other than q_0 are irrelevant. This concludes the proof. □

When generating automata randomly to get experimental results, the computing time depends strongly on the internal representation of the automaton. *Almeida, Moreira and Reis* [1] used a string representation for ICDFA$_\emptyset$s. We will also see a different formula for $B_k(n)$ as a direct positive summation.

6.1.1 String representation

The results in this subsection were observed by *Almeida, Moreira and Reis* [1]. As noted before, the names of the states of an ICDFA$_\emptyset$ are irrelevant. A representation using those names would therefore not be unique. We give a way of obtaining a canonical order on the states, making their names obsolete. Let $M = (Q, \Sigma, \delta, q_0)$ be an ICDFA$_\emptyset$ with $\Sigma = \{0, 1, \ldots, k-1\}$ and $|Q| = n$. We construct a numbering $\phi : Q \to \{0, \ldots, n-1\}$ by traversing the transition function in a breadth-first way. We will use brackets to denote intervals of integers, for example $[a, b[= \{x \in \mathbb{Z} | a \leq x < b\}$. The number $\phi(q_i)$ will be the canonical state number of q_i. The function ϕ is defined by the Algorithm 1.

Algorithm 1 Construct canonical state numbers

```
1: φ(q_0) ← 0
2: i ← 0
3: s ← 0
4: repeat
5:     for j ∈ [0, k − 1] do
6:         if δ(φ^{-1}(s), j) ∉ φ^{-1}([0, i]) then
7:             φ(δ(φ^{-1}(s), j)) ← i + 1
8:             i ← i + 1
9:     i ← i + 1
10: until s ≥ i
```

In words: The initial state q_0 gets the number 0. Every adjacent state that is not numbered yet gets a new number according to the order of the letter it is reached with. If every neighbour has a number, repeat this procedure for the state with the next number until every state has a number. The variable

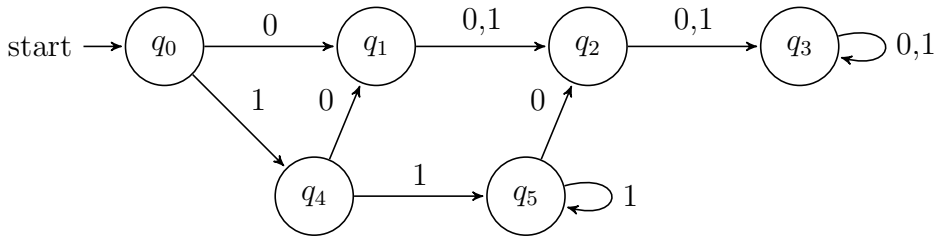


Figure 6.4: An $\text{ICDFA}_\emptyset$

i tracks the highest used number and s is the number of the state that is currently checking all of its adjacent states.

The function ϕ is a bijection. The new ordering of the states can be thought of like this: $\phi(q_a) < \phi(q_b)$ if one of the following two conditions hold:

- The state q_a is reached by a shorter word than q_b.
- q_a and q_b are reached by equally short words, but the state q_a is reached by a smaller word in a lexicographical sense.

We can now replace Q with $[0, n-1]$ according to the given algorithm. Applying the numbering algorithm to the $\text{ICDFA}_\emptyset$ given in Figure 6.4 results in the numbering $\phi(q_0) = 0$, $\phi(q_1) = 1$, $\phi(q_4) = 2$, $\phi(q_2) = 3$, $\phi(q_5) = 4$ and $\phi(q_3) = 5$. A string representation is now possible by simply giving all the transitions for all states and all letters. For the automaton in Figure 6.4 the representing string is

$$\begin{aligned} s &= [\delta(0,0), \delta(0,1), \delta(1,0), \delta(1,1), \delta(2,0), \delta(2,1), \delta(3,0), \ldots, \delta(5,1)] \\ &= [1,2,3,3,1,4,5,5,3,4,5,5]. \end{aligned}$$

More generally the i-th entry of the representing string is

$$s_i = \delta\left(\lfloor i/k \rfloor, i \mod k\right) \quad \text{for} \quad i \in [0, kn[.$$

We are now able to translate an $\text{ICDFA}_\emptyset$ to a string, but not every string $s \in [0, n[^*$ of length kn corresponds to an $\text{ICDFA}_\emptyset$. We give a characterization of these strings as observed by *Almeida, Moreira and Reis* [1].

Theorem 6.2. There is a bijection between the nonisomorphic ICDFA$_\emptyset$s with n states over the alphabet $\Sigma = [0, k[$ and the strings $(s_i)_{i\in[0,kn[}$ with $s_i \in [0, n[$ satisfying the conditions

$$\forall m \in [2, n[\colon\ \forall i \in [0, kn[\colon\ (s_i = m \Rightarrow \exists j \in [0, i[\colon\ (s_j = m-1))\,, \quad \text{(R1)}$$

$$\forall m \in [1, n[\colon\ \exists j \in [0, km[\colon\ (s_j = m)\,. \quad \text{(R2)}$$

From now on, we will call the representing string of a given ICDFA$_\emptyset$ its *canonical string*.

Proof. The condition (R1) guarantees that a state m can only occur in the string, if its predecessor already occurred earlier in the string. This corresponds to the canonical numbering of states ϕ. A violation of (R2) would result in the state m being unreachable from the initial state 0. In addition (R2) guarantees that every state occurs at least once in the string. We see that the automaton obtained from a string satisfying (R2) is initially connected since we can backtrack every state to the initial state. □

Instead of (R1) and (R2) we use a different set of rules to be able to enumerate ICDFA$_\emptyset$s. For a canonical string of an ICDFA$_\emptyset$ we define f_j as the index (starting at 0) of the first occurrence of the label j for $j \in [1, n[$ and call them the sequence of *flags*. Corresponding to (R1) and (R2) we get

$$\forall m \in [2, n[\colon\ (f_j > f_{j-1})\,, \quad \text{(G1)}$$

$$\forall m \in [1, n[\colon\ (f_m < km)\,. \quad \text{(G2)}$$

Let $F_{k,n}$ be the number of allowed flags for ICDFA$_\emptyset$s with n states over k letters. Since the flags represent newly discovered states in a breadth-first approach, the flags can be seen as the internal nodes of the k-ary search tree. The rules (G1) and (G2) together describe the outline of a generalized *Dyck path*. Dyck paths and numerous other Catalan objects are enumerated by the Fuss-Catalan numbers $C_n^{(k)}$ [10, p.361]. Other structures like k-ary rooted trees and bijections between these structures are given by *Schmidthammer* [23]. We will use k-ary trees to prove the following theorem.

Theorem 6.3. We have

$$F_{k,n} = C_n^{(k)} = \frac{1}{(k-1)n+1}\binom{kn}{n}.$$

Proof. The flags correspond to the internal nodes of the search tree of an $\mathrm{ICDFA}_\emptyset$ in a natural way. Internal nodes are newly discovered states and leaves are transitions to already known states. The rules (G1) and (G2) guarantee the connectedness. The inequality $f_1 < k$ for example says that the state number 1 has to be discovered by the initial state.

Thus, a formula for $F_{k,n}$ is obtained by counting k-ary trees with n internal nodes. Let $\mathcal{T}$ be the class of k-ary trees. It holds that

$$\mathcal{T} = \{\bullet\} \times \left(\{\epsilon\} \dot\cup \mathcal{T}^k\right).$$

The GF of $\mathcal{T}$ (counting all the nodes, not just internal ones) satisfies

$$T(z) = z\left(1 + T^k(z)\right).$$

Theorem 2.4 can be applied. Note that a tree with n internal nodes has $kn+1$ nodes. We have

$$\begin{aligned}
F_{k,n} &= [z^{kn+1}]T(z) \\
&= \frac{1}{kn+1}[u^{kn}](1+u^k)^{kn+1} \\
&= \frac{1}{kn+1}[u^{kn}]\sum_{j=0}^{kn+1}\binom{kn+1}{j}u^{kj} \\
&= \frac{1}{kn+1}\binom{kn+1}{n} \\
&= \frac{1}{(k-1)n+1}\binom{kn}{n}.
\end{aligned}$$

□

Let us recall the $\mathrm{ICDFA}_\emptyset$ in Figure 6.4. The flags of its canonical string are $[f_1, \ldots, f_5] = [1, 2, 3, 6, 7]$ and the corresponding binary tree is given in

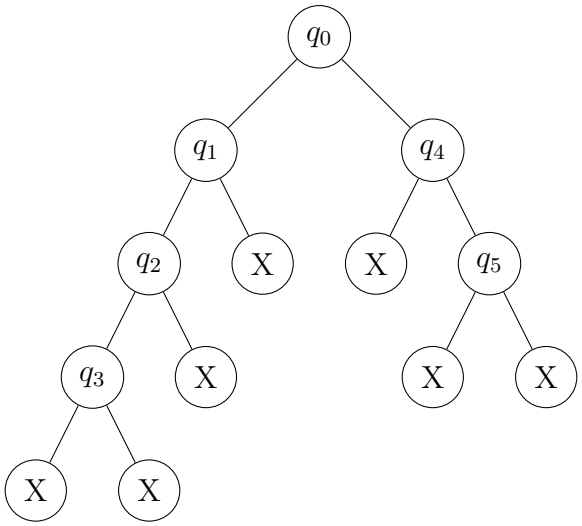


Figure 6.5: The search tree corresponding to the flags of a canonical string

Figure 6.5, where a left child corresponds to the letter 0, a right child to the letter 1 and a node labelled with X to a state that was already discovered. For each allowed sequence of flags we can choose the remaining string entries as follows:

$$i < f_1 \Rightarrow s_i = 0, \tag{G3}$$

$$\forall j \in [1, n-2] : (f_j < i < f_{j+1} \Rightarrow s_i \in [0, j]), \tag{G4}$$

$$i > f_{n-1} \Rightarrow s_i \in [0, n[. \tag{G5}$$

This set of rules, (G1)-(G5), also characterizes the canonical strings of all possible ICDFA$_\emptyset$s. It can be used for the generation of all ICDFA$_\emptyset$s shown by *Almeida, Moreira and Reis* [1]. We note the relation to Catalan objects and will focus on the random generation of ICDFA$_\emptyset$s in Chapter 7.

If we consider only strings $s \in [0, n[^{kn}$ that satisfy (G1), we get an upper bound for $B_k(n)$. The last k symbols in the canonical string s can be chosen from $[0, n[$ without any restriction. This gives n^k possibilities. The other part of the string is an element of the language

$$\{a_1 a_2 \dots a_k \in [0, n[^* \mid \forall i : a_i \leq \max\{a_1, \dots, a_{i-1}\} + 1\},$$

where every letter in $[1, n[$ occurs at least once. The set of these strings can also be defined by a regular expression

$$L_n := L\left(0^* \prod_{j\in[1,n[} j(0+\cdots+j)^*\right).$$

Every word of length m in L_n corresponds to a partition of $[1, m]$ into at most n parts, as observed by *Moreira and Reis* [19]. Since we study initially connected automata, all the states $\{1, \ldots, n-1\}$ occur in the canonical string. The state 0 does not necessarily occur, thus the resulting partition has either n parts or $n-1$ parts. We will obtain a partition from a string $s = s_1, s_2, \ldots, s_m$ by constructing classes A_i with $j \in A_{s_j}$. The string $s = [0, 0, 1, 0, 2]$ for example translates to the partition

$$\{\{1, 2, 4\}, \{3\}, \{5\}\}.$$

We recall the string we obtained from Figure 6.4,

$$s = [1, 2, 3, 3, 1, 4, 5, 5, 3, 4, 5, 5].$$

This would result in a partition with only 5 classes, even though the automaton has 6 states. We can fix this by adding the letter 0 to the beginning of every string, guaranteeing the occurrence of every letter. This confirms the result given by *Almeida, Moreira and Reis* [1] that for $c \in \mathbb{N}$ the words in L_c with length m are enumerated by the Stirling numbers $\left\{{m+1 \atop c}\right\}$. In addition this insight gives a neat formula for the ordinary generating function of the Stirling numbers, since the regular expression of

$$0 \cdot L_c = L\left(\prod_{j\in[0,c[} j(0+\cdots+j)^*\right)$$

translates directly to

$$\sum_{n\geq 0} \left\{{n \atop c}\right\} z^n = \frac{z^c}{(1-z)(1-2z)\ldots(1-cz)}.$$

Note that the additional zero is the reason for the shift to $\left\{ {m+1 \atop c} \right\}$ instead of $\left\{ {m \atop c} \right\}$. We can also write the Stirling numbers of the second kind as

$$\left\{ {n \atop k} \right\} = \frac{1}{k!} \sum_{j=0}^{k} \binom{k}{j} (-1)^j (k - j^n)$$

as described in [8, pp.735–737]. For the asymptotic analysis of the Stirling numbers we will only work with the exponential generating function.

Theorem 6.4. By counting the possibilities for the first $kn - k$ entries in the canonical string and the last k separately and multiplying the number of possibilities, we have

$$B_k(n) \leq \left\{ {k(n-1)+1 \atop n} \right\} n^k.$$

This bound given by *Almeida, Moreira and Reis* [1] is tighter than the bound given by *Bassino and Nicaud* [3], which we get as a consequence of this estimate.

Corollary 6.5. We have

$$B_k(n) \leq n \left\{ {kn \atop n} \right\}.$$

Proof. We show this weaker result by proving

$$\left\{ {n-i \atop m} \right\} \leq \frac{1}{m^i} \left\{ {n \atop m} \right\}$$

by induction over i for $0 \leq i \leq n - m$, which can be applied to the bound given in Theorem 6.4.
Let $i = 0$. Obviously $\left\{ {n-i \atop m} \right\} \leq \frac{1}{m^i} \left\{ {n \atop m} \right\}$. Using the well known identity

$$\left\{ {n \atop m} \right\} = m \left\{ {n-1 \atop m} \right\} + \left\{ {n-1 \atop m-1} \right\},$$

we estimate for $i \in [1, n-m[$ that

$$\left\{ {n-i-1 \atop m} \right\} \leq \frac{1}{m} \left\{ {n-i \atop m} \right\} \leq \frac{1}{m^{i+1}} \left\{ {n \atop m} \right\}.$$

□

Our goal now is to obtain an exact formula for $B_k(n)$ as a direct positive summation by counting all possible canonical strings. For a string to satisfy both (R1) and (R2) we have to consider the sequence of flags f_j with $j \in [1, n[$. We define $f_n := kn$. The set of possible canonical strings for a given sequence of flags is given by the expression

$$0^{f_1} \prod_{j \in [1,n[} j(0 + \cdots + j)^{f_{j+1} - f_j - 1}. \tag{6.1}$$

We take all possible flag sequences into consideration to get a regular expression for all canonical strings:

$$\sum_{f_1=0}^{k-1} \sum_{f_2=f_1+1}^{2k-1} \cdots \sum_{f_{n-1}=f_{n-2}+1}^{(n-1)k-1} \left(0^{f_1} \prod_{j \in [1,n[} j(0 + \cdots + j)^{f_{j+1} - f_j - 1} \right).$$

Using this regular expression, the canonical strings can be easily enumerated. Note that for a given flag sequence the language defined by (6.1) is of size $\prod_{j=1}^{n} j^{f_j - f_{j-1} - 1}$. As observed by *Almeida, Moreira and Reis* [1], this gives the following formula.

Theorem 6.6. The number of nonisomorphic ICDFA$_\emptyset$s with n states over k letters satisfies

$$B_k(n) = \sum_{f_1=0}^{k-1} \sum_{f_2=f_1+1}^{2k-1} \cdots \sum_{f_{n-1}=f_{n-2}+1}^{(n-1)k-1} \left(\prod_{j=1}^{n} j^{f_j - f_{j-1} - 1} \right).$$

6.1.2 Boxed Dyck diagrams

A similar approach to the representation of ICDFA$_\emptyset$s is shown by *Bassino and Nicaud* [3]. This subsection is based on their work. Again we will come across

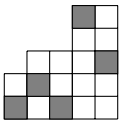


Figure 6.6: The boxed diagram corresponding to $(2, 3, 3, 5, 5)$ and $(1, 2, 1, 5, 3)$

Catalan objects and set partitions. This time, we will use the representation as the basis for the random sampler explained in Chapter 7.

For a given ICDFA$_\emptyset$ $M = (Q, \Sigma, \delta, q_0)$ with n states over k letters, we find a natural order of the states by the lexicographical order of words. More precisely, we give an order of the states Q according to the words

$$w(q) := \min_{lex}\{w \in \Sigma^* | \delta(q_0, w) = q\}.$$

We choose the numbers $\{1, 2, \ldots, n\}$ as the canonical names of the states, such that $w(1) < w(2) < \cdots < w(n)$. Note that this order of words leads to a depth-first traversal of the ICDFA$_\emptyset$ contrary to the last approach, which led to a breadth-first traversal. An ICDFA$_\emptyset$ can be represented as a diagram with special properties. A diagram is a sequence of integers $(x_1, x_2, \ldots, x_m)$ that can be drawn as a nondecreasing sequence of columns. A boxed diagram has exactly one marked box for each column, see Figure 6.6. We can describe such a diagram with two sequences, $(x_i)_{i\in[1,m]}$ and $(y_i)_{i\in[1,m]}$ with $x_i \geq y_i$ for all $i \in [1, m]$. The diagram is of width n and height m if it consists of n columns with $x_m = n$. The diagrams that represent an ICDFA$_\emptyset$ are the k-Dyck boxed diagrams.

Definition 6.7. A k-Dyck boxed diagram of size n is a boxed diagram of width $(k-1)n+1$ and height n, with column heights $(x_1, \ldots, x_{(k-1)n+1})$ satisfying the Dyck condition $x_i \geq \lceil i/(k-1) \rceil$ for $i \leq (k-1)n$.

We see the similarity to the string representation of ICDFA$_\emptyset$s, as Dyck diagrams are also enumerated by the Catalan numbers.

We give an algorithm to obtain the strings $(x_i)_{i\in I}$ and $(y_i)_{i\in I}$ with $I = [1, (k-1)n+1]$ as a pseudo code taken from [3]. These strings will be denoted

by Max and Boxed, which are initialized as empty lists. The variable "nbr" is the number of labelled (or discovered) states. The algorithm (Algorithm 2)

Algorithm 2 From DFA to boxed Dyck

```
1:  procedure FromDfaToBoxedDyck(𝒟)
2:      Max = (); Boxed = ();
3:      i ← 0
4:      s ← 0
5:      for q in Q do
6:          Visited[q] = false
7:          Number[q] = 0
8:          nbr = 0
9:      DepthFirst(𝒟, q_0, Max, Boxed, nbr)
10:     return(Max, Boxed)
11:
12: procedure DepthFirst(𝒟, q, Max, Boxed, nbr)
13:     Visited[q] = true
14:     nbr = nbr + 1
15:     Number[q] = nbr
16:     for a in Σ, in the lexicographical order do
17:         if Visited[q · a] then
18:             Append(Max, nbr)
19:             Append(Boxed, Number[q · a])
20:         else
21:             DepthFirst(𝒟, q · a, Max, Boxed, nbr)
```

can be described as follows:

- Traverse the $\mathrm{ICDFA}_\emptyset$ in a depth-first manner according to the lexicographical order of words.
- The initial state gets the number 1.
- If a new state is encountered during this process, assign the next unused state number.

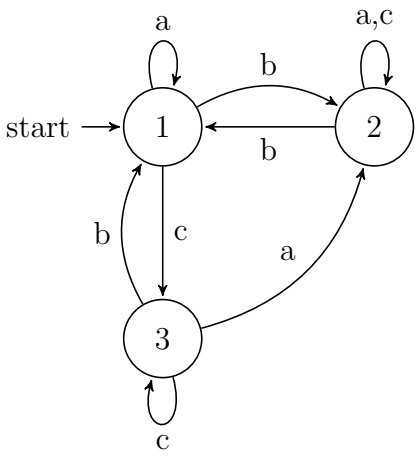


Figure 6.7: An ICDFA$_\emptyset$ with canonical state names

- If the traversal of the ICDFA$_\emptyset$ leads to an already discovered state, create a new column in the diagram of height equal to the number of currently labelled states and mark the i-th box of this new column, where i is the already known state we reached in this step.

Let us execute this algorithm for the automaton given in Figure 6.7, an automaton with $n = 3$ states over $k = 3$ letters. We list all the words ω that are considered during the depth-first traversal together with the number of currently known states x_i and the number of the state $q_0 \cdot \omega$, which is y_i.

ω	x_i	y_i	new state
ϵ			1
a	1	1	
b			2
ba	2	2	
bb	2	1	
bc	2	2	
c			3
ca	3	2	
cb	3	1	
cc	3	3	

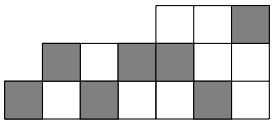


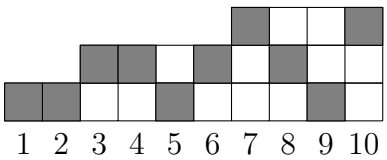
Figure 6.8: The boxed diagram corresponding to Max = $(1, 2, 2, 2, 3, 3, 3)$ and Boxed = $(1, 2, 1, 2, 2, 1, 3)$

The resulting sequences are $x = (1, 2, 2, 2, 3, 3, 3)$ and $y = (1, 2, 1, 2, 2, 1, 3)$. Since $3 = n = k$, the sequences are of length $(k-1)n + 1 = 7$. Figure 6.8 shows the corresponding diagram. This process can be easily reversed to obtain an $\text{ICDFA}_\emptyset$ from a given k-Dyck boxed diagram. The representation of an $\text{ICDFA}_\emptyset$ as a diagram allows us to establish a connection to set partitions.

Lemma 6.8. There is a bijection from the set of boxed diagrams of width m and height n to the set of partitions of a set with $n + m$ elements into n parts.

Proof. Starting with a boxed diagram of width m and height n we add n columns with heights $1, 2, \ldots, n$. The topmost box in these new columns is marked. Each new column is inserted at the leftmost position that does not violate the definition of a diagram. This gives $n + m$ columns. We can now construct a partition according to the marked boxes: The numbers i and j are in the same part if and only if the i-th and the j-th column have marks at the same height. □

We take the diagram from Figure 6.8 as an example. It is of width 7 and of height 3. We insert the columns ,  and  at the leftmost possible positions which results in the following diagram.



The partition of the set $\{1, 2, \ldots, 10\}$ according to the marked boxes is

$$\{\{1, 2, 5, 9\}, \{3, 4, 6, 8\}, \{7, 10\}\}.$$

This approach proves again, that every $\text{ICDFA}_\emptyset$ with n states over k letters corresponds to exactly one set partition of $kn + 1$ elements into n parts. Note that this does not hold the other way around, since we disregarded the Dyck condition for the diagrams. Therefore we only get the upper bound

$$B_k(n) \leq \left\{ {kn + 1 \atop n} \right\}.$$

To slightly improve this bound and to prove again the bound obtained in Subsection 6.1.1 we split a diagram of width $(k-1)n+1$ and height n into its last column, which is of height n, and the shorter diagram of width $(k-1)n$. There are n possibilities to mark a box in the last column and each residual diagram corresponds to one of $\left\{ {kn \atop n} \right\}$ set partitions. This gives

$$B_k(n) \leq n \left\{ {kn \atop n} \right\},$$

which we have already shown using the string representation.

6.1.3 Asymptotics of ICDFAs

We note that since $A_k(n) = 2^n B_k(n)$, we have $A_k(n) \leq n2^n \left\{ {kn \atop n} \right\}$. In [3], *Bassino and Nicaud* give a tight lower bound for $A_k(n)$. For a fixed alphabet size of k they write $\mathcal{F}_{m,n}$ and $\mathcal{S}_{m,n}$ for the set of k-Dyck boxed diagrams and boxed diagrams of width m and height n. We have $\mathcal{F}_{m,n} \subset \mathcal{S}_{m,n}$ and $B_k(n) = n|\mathcal{F}_{(k-1)n,n}|$. To obtain a lower bound for $B_k(n)$ the equation

$$\begin{aligned} |\mathcal{F}_{(k-1)n,n}| &= |\mathcal{S}_{(k-1)n,n}| - |\mathcal{S}_{(k-1)n,n} \setminus \mathcal{F}_{(k-1)n,n}| \\ &= \left\{ {kn \atop n} \right\} - |\mathcal{S}_{(k-1)n,n} \setminus \mathcal{F}_{(k-1)n,n}| \end{aligned}$$

is used and the term $|\mathcal{S}_{(k-1)n,n} \setminus \mathcal{F}_{(k-1)n,n}|$ is overestimated. Together with the upper bound, Bassino and Nicaud find that $A_k(n) = \Theta(n2^n \left\{ {kn \atop n} \right\})$. There is an even stronger relation to the Stirling numbers as observed by *Korshunov* in [13] and [14]. The original result does not use the notion of Stirling numbers or set partitions. Korshunov stated the following:

Theorem 6.9. The number of nonisomorphic initially connected deterministic finite automata $A_k(n)$ satisfies

$$A_k(n) \sim \left(1 - \frac{ka_k}{1+a_k}\right)^{-1/2} \hat{E}_k \frac{2^n \nu^n(k) n^{kn}}{(n-1)!}$$

with

$$\hat{E}_k = \frac{1 + \sum_{r=1}^{\infty} \frac{1}{r}\binom{kr}{r-1} \left(e^k \nu(k)\right)^{-r}}{1 + \sum_{r=1}^{\infty} \binom{kr}{r} \left(e^k \nu(k)\right)^{-r}}.$$

We will use this without proof and are interested in establishing a connection to set partitions. *Bassino and Nicaud* [3] were able to reformulate this estimate in terms of Stirling numbers (see Theorem 6.11). Thus, we have to further investigate the numbers of distinct set partitions. The search of a better estimate of our bounds boils down to the question: What is the asymptotic growth of the Stirling numbers $\left\{ {kn \atop n} \right\}$? This question has already been answered by *Good* [9]. He gave an asymptotic expansion for $\left\{ {n \atop m} \right\}$ with $n/m = \Theta(1)$.

6.2 A closer look at the Stirling numbers

We base our calculations on the work of *Flajolet and Sedgewick* [8]. To estimate the Stirling numbers we are interested in, we have to take a look at the bivariate generating function of set partitions as mentioned in Section 2.4. The combinatorial construction of set partitions

$$\mathcal{P} = \textsc{Set}(\textsc{Set}_{\geq 1}(\mathcal{Z}))$$

gives the EGF

$$\sum_{n,k \geq 0} \left\{ {n \atop k} \right\} \frac{z^n y^k}{n!} = e^{y(e^z - 1)}.$$

Our goal is to estimate

$$\left\{ {kn \atop n} \right\} = (kn)![z^{kn} y^n] e^{y(e^z-1)} = (kn)![z^{kn}] \frac{(e^z-1)^n}{n!}.$$

For the estimate of $[z^{kn}](e^z-1)^n$ we use the saddle point method of *Flajolet and Sedgewick* [8, p.587]. Our problem can be regarded as a special case of this method, since the power of the generating function increases with the coefficient we want to extract. The method is applicable as follows.

We observe functions of the form $A(z)\cdot B(z)^n$ which satisfy the conditions

L_1 : A and B are analytic at 0 with nonnegative coefficients and $B(0)\neq 0$.

L_2 : B is aperiodic in a sense that there is no analytic function β and integer $p\geq 2$ such that $B(z)=\beta(z^p)$.

L_3 : The radius of convergence of A is at least as large as the radius of B.

Theorem 6.10. Under the conditions L_1, L_2 and L_3, we get

$$[z^{\lambda n}]A(z)\cdot B(z)^n \sim A(\zeta)\frac{B(\zeta)^n}{\zeta^{\lambda n+1}\sqrt{2\pi n\eta}},$$

where ζ is the unique positive solution to

$$\zeta\frac{B'(\zeta)}{B(\zeta)}=\lambda$$

and

$$\eta=\frac{d^2}{dz^2}\left(\log(B(z))-\lambda\log(z)\right)\Big|_{z=\zeta}.$$

We apply this to our initial problem and get

$$[z^{kn}](e^z-1)^n=[z^{kn}]z^n\left(\frac{e^z-1}{z}\right)^n=[z^{(k-1)n}]\left(\frac{e^z-1}{z}\right)^n.$$

Note that (e^z-1) cannot be assigned to $B(z)$ since we need $B(0)\neq 0$. Thus, we have $A(z)=1$, $B(z)=\frac{e^z-1}{z}$ and $\lambda=k-1$. We get

$$\begin{aligned}
&\zeta\frac{B'(\zeta)}{B(\zeta)}=k-1\\
\Leftrightarrow\ &\frac{\zeta e^\zeta}{e^\zeta-1}-1=k-1\\
\Leftrightarrow\ &(\zeta-k)e^{\zeta-k}=-ke^{-k}\\
\Leftrightarrow\ &\zeta=W_0(-ke^{-k})+k,
\end{aligned}$$

where we abbreviate ζ_k by ζ and where W_0 is the *Lambert-W function* we mentioned in Section 2.2.

Note that $W_0(-ke^{-k}) \neq -k$ even though W_0 is the inverse function of $f : x \mapsto xe^x$. This is due to the fact that f is bijective on $[-1, \infty[$, therefore $W_0(x)$ is well defined for $x \in [-1/e, \infty[$. In the previous calculation $-ke^{-k}$ is a valid argument for W_0 but as $k > 1$ the value of $-k$ leaves the range $[-1, \infty[$ thus $W_0(-ke^{-k}) \neq -k$.

Now we have to determine the value of η by

$$\begin{aligned}
\eta &= \frac{d^2}{dz^2} \left(\log(B(z)) - \lambda \log(z) \right) \Big|_{z=\zeta} \\
&= \frac{d^2}{dz^2} \log\left(\frac{e^z - 1}{z^k} \right) \Big|_{z=\zeta} \\
&= \frac{d}{dz} \left(\frac{e^z}{e^z - 1} - \frac{k}{z} \right) \Big|_{z=\zeta} \\
&= \frac{-e^\zeta}{(e^\zeta - 1)^2} + \frac{k}{\zeta^2}.
\end{aligned}$$

Since $\zeta = W_0(-ke^{-k}) + k$, we also have $e^\zeta = \frac{k}{k-\zeta}$. This gives

$$\eta = \frac{\frac{-k}{k-\zeta}}{\left(\frac{k}{k-\zeta} - 1 \right)^2} + \frac{k}{\zeta^2} = \frac{k}{\zeta^2} \left(\zeta - (k-1) \right).$$

Using the same notation as in Theorem 6.10, we have

$$\left\{ {kn \atop n} \right\} = \frac{(kn)!}{n!} [z^{kn}] \left(e^z - 1 \right)^n.$$

Together with Stirling's approximation formula, which gives

$$\frac{(kn)!}{n!} \sim \frac{(kn/e)^{kn} \sqrt{2\pi kn}}{(n/e)^n \sqrt{2\pi n}} = k^{kn} \left(\frac{n}{e} \right)^{(k-1)n} \sqrt{k},$$

we get

$$\begin{aligned}
\left\{ {kn \atop n} \right\} &\sim k^{kn} \left(\frac{n}{e}\right)^{(k-1)n} \sqrt{k} \frac{B(\zeta)^n}{\zeta^{(k-1)n+1}\sqrt{2\pi n\eta}} \\
&= k^{kn} \left(\frac{n}{e}\right)^{(k-1)n} \sqrt{k} \frac{B(\zeta)^n}{\zeta^{(k-1)n+1}\sqrt{2\pi n \frac{k}{\zeta^2}\left(\zeta-(k-1)\right)}} \\
&= \sqrt{\frac{1}{2\pi\left(\zeta-(k-1)\right)}} \frac{k^{kn}}{(e\zeta)^{(k-1)n}} B(\zeta)^n n^{(k-1)n-1/2} \\
&= \alpha \frac{k^{kn}}{(e\zeta)^{(k-1)n}} B(\zeta)^n n^{(k-1)n-1/2} \\
&= \alpha\beta^n n^{(k-1)n-1/2}
\end{aligned}$$

with

$$\alpha := \sqrt{\frac{1}{2\pi(\zeta-(k-1))}} \quad \text{and} \quad \beta := \frac{k^k(e^\zeta-1)}{e^{k-1}\zeta^k}.$$

We can now give the asymptotic estimate of $A_k(n)$ using the Stirling numbers of the second kind by reformulating Korshunov's result [13, 14].

Theorem 6.11. It holds that

$$A_k(n) \sim E_k n 2^n \left\{ {kn \atop n} \right\}$$

with

$$E_k = \frac{1+\sum_{r=1}^{\infty} \frac{1}{r}\binom{kr}{r-1}\left(e^k\nu(k)\right)^{-r}}{1+\sum_{r=1}^{\infty}\binom{kr}{r}\left(e^k\nu(k)\right)^{-r}}.$$

Proof. We explore Korshunov's result given in Theorem 6.9 and establish a connection to the Stirling numbers. *Korshunov* [13, 14] writes a_k for the unique root of $1+x = xe^{k/(1+x)}$ in the interval $[0,1]$. Note that $k \geq 2$. In addition he defines

$$\nu(k) := a_k^{a_k}(1+a_k)^{k-1-a_k}.$$

We claim that $\zeta = \frac{k}{1+a_k}$. We already know the identity $k/(k-\zeta) = e^\zeta$. It

follows that

$$\begin{aligned}\frac{k}{k-\zeta} = e^{\zeta} &\Leftrightarrow k = (k-\zeta)e^{\zeta} \\ &\Leftrightarrow \frac{k}{\zeta} = \left(\frac{k}{\zeta} - 1\right) e^{\zeta} \\ &\Leftrightarrow 1 + \left(\frac{k}{\zeta} - 1\right) = \left(\frac{k}{\zeta} - 1\right) e^{\frac{k}{1+\left(\frac{k}{\zeta}-1\right)}} \\ &\Leftrightarrow \left(\frac{k}{\zeta} - 1\right) \text{ is the root of } 1 + x = xe^{k/(1+x)}.\end{aligned}$$

Thus, $a_k = \frac{k}{\zeta} - 1$ and $a_k = \frac{k}{\zeta}e^{-\zeta}$. For the first term in Korshunov's estimate we get

$$\left(1 - \frac{ka_k}{1+a_k}\right)^{-1/2} = (1 - \zeta(k7\zeta - 1))^{-1/2} = \sqrt{\frac{1}{\zeta - (k-1)}} = \alpha\sqrt{2\pi}.$$

Recall that

$$\beta = \frac{k^k(e^{\zeta} - 1)}{e^{k-1}\zeta^k}.$$

We find that

$$\begin{aligned}\nu(k) &= a_k^{a_k}(1 + a_k)^{k-1-a_k} \\ &= \left(\frac{k}{\zeta}e^{-\zeta}\right)^{\frac{k}{\zeta}-1} \left(\frac{k}{\zeta}\right)^{k-\frac{k}{\zeta}} \\ &= \left(\frac{k}{\zeta}\right)^{k-1} e^{\zeta-k}.\end{aligned}$$

Using the identity

$$e^{\zeta} = \frac{k}{\zeta}(e^{\zeta} - 1),$$

we get

$$e\nu(k) = \frac{k^{k-1}}{\zeta^{k-1}e^{k-1}}e^{\zeta} = \frac{k^k}{e^{k-1}}\frac{(e^{\zeta} - 1)}{\zeta^k} = \beta.$$

We have $e^k\nu(k) = \beta e^{k-1}$, thus the quotients $\hat{E}_k$ and E_k are equal. We are now

able to transform Korshunov's estimate [13, 14]

$$\begin{aligned} A_k(n) &\sim \left(1 - \frac{ka_k}{1+a_k}\right)^{-1/2} \hat{E}_k \frac{2^n \nu^n(k) n^{kn}}{(n-1)!} \\ &= \alpha\sqrt{2\pi} E_k \frac{2^n \beta^n n^{kn+1}}{e^n (n!)} \\ &\sim \alpha\sqrt{2\pi} E_k \frac{2^n \beta^n n^{kn-n+1}}{\sqrt{2\pi n}} \\ &= E_k n 2^n \alpha \beta^n n^{(k-1)n-1/2} \\ &\sim E_k n 2^n \left\{ {kn \atop n} \right\}. \end{aligned}$$

This concludes the proof. □

Bassino and Nicaud [3] also provide numerical results for

$$\frac{A_n(k)}{2^n n \left\{ {kn \atop n} \right\}}.$$

This is of interest since these quotients will give the probability of successfully generating an ICDFA from a random set partition. Due to the asymptotic equivalence we have

$$E_k = \lim_{n\to\infty} \frac{A_n(k)}{2^n n \left\{ {kn \atop n} \right\}}.$$

It holds that $\lim_{k\to\infty} E_k = 1$. The obtained numerical results include for example $E_2 \approx 0.74490782$, $E_3 \approx 0.87341820$, $E_4 \approx 0.93931196$ and $E_{26} \approx 0.99999999987$. The calculations suggest that even for small alphabets we have a reliable percentage of set partitions that can be transformed to ICDFAs.

Chapter 7

Random Sampling

In this chapter our goal is to randomly generate automata using the previous results, primarily the representations of ICDFAs as boxed Dyck diagrams. Not only do we want to generate automata of a fixed size, but to do this uniformly. Let us demonstrate this with a small example.

Example: Set partitions We want to randomly generate partitions of the set $\{1, 2, 3, 4\}$ consisting of 2 parts. The $\left\{ {4 \atop 2} \right\} = 7$ partitions we are interested in are given in Figure 7.1. Naturally, when sampling combinatorial objects, we want to avoid listing all possible objects at all cost. To sample one of these 7 partition we could proceed as follows.

- Draw 2 elements at random from $\{1, 2, 3, 4\}$.
- Declare them to be elements of different parts.
- Assign the remaining 2 elements to a randomly chosen part with probabilities $1/2$ each.

$$\{\{1,2,3\},\{4\}\} \quad \{\{1,2,4\},\{3\}\} \quad \{\{1,3,4\},\{2\}\} \quad \{\{1,2\},\{3,4\}\}$$

$$\{\{1,3\},\{2,4\}\} \quad \{\{1,4\},\{2,3\}\} \quad \{\{1\},\{2,3,4\}\}$$

Figure 7.1: All partitions of 4 elements consisting of 2 parts

Let γ be the partition obtained by this procedure. We see that

$$\mathbb{P}(\gamma = \{\{1,2,3\},\{4\}\}) = \frac{3}{\binom{4}{2}}\left(\frac{1}{2}\right)^2 = \frac{1}{8}, \quad \text{but}$$
$$\mathbb{P}(\gamma = \{\{1,2\},\{3,4\}\}) = \frac{4}{\binom{4}{2}}\left(\frac{1}{2}\right)^2 = \frac{1}{6}.$$

Thus, this method does not uniformly create partitions. This brings us to the Boltzmann model, which provides a framework for random generation of objects where the probability for a certain object only depends on its size.

7.1 The Boltzmann Model

The idea behind the Boltzmann model is to relax the condition of generating objects of a fixed size. Instead we will generate objects with fixed average size, which we can tune to fit our requirements. This approach is easy to implement, uses little amount of pre-calculation and generates objects of the same size with the same probability. We use a method as explained by *Duchon, Flajolet, Louchard and Schaeffer* [6], the notion of a *Boltzmann sampler*.

Definition 7.1. Let $\mathcal{C}$ be a combinatorial structure with a weight function $|\cdot|$ with OGF $C(z) = \sum_{n\geq 0} C_n z^n$ and $x > 0$ a real number smaller than the radius of convergence of C. The Boltzmann model with parameter x is the probability distribution

$$\mathbb{P}_x(\gamma) = \frac{1}{C(x)} \cdot x^{|\gamma|}.$$

Analogously for the exponential generating function $\hat{C}(z) = \sum_{n\geq 0} C_n \frac{z^n}{n!}$ the exponential Boltzmann model is

$$\mathbb{P}_x(\gamma) = \frac{1}{\hat{C}(x)} \cdot \frac{x^{|\gamma|}}{|\gamma|!}.$$

Note that these are probability distributions for $\Omega = \mathcal{C}$ since

$$\sum_{\gamma\in\mathcal{C}} \mathbb{P}_x(\gamma) = \frac{1}{C(x)} \sum_{\gamma\in\mathcal{C}} x^{|\gamma|} = \frac{1}{C(x)} C(x) = 1.$$

The parameter x can be used to change the average size of the generated objects. Let the random variable N be the size of the generated object. The first and second moments of N are

$$\mathbb{E}_x(N) = \frac{|\gamma| x^{|\gamma|}}{C(x)} = x\frac{C'(x)}{C(x)} \quad \text{and}$$

$$\mathbb{E}_x(N^2) = \frac{x^2 C''(x) + xC'(x)}{C(x)}.$$

Thus, the variance can be obtained by

$$\begin{aligned}\mathbb{V}_x(N) &= \mathbb{E}_x(N^2) - \mathbb{E}_x(N)^2 \\ &= \frac{x^2 C''(x) + xC'(x)}{C(x)} - x^2\frac{C'(x)^2}{C(x)^2} \\ &= x \cdot \frac{d}{dx}\mathbb{E}_x(N).\end{aligned}$$

Note that choosing the parameter x such that $\mathbb{E}_x(N) = n$ yields the best results for sampling objects of size n. There is a simple explanation for this: We want to maximize the probability

$$\mathbb{P}_x(|\gamma| = n) = C_n\frac{x^n}{C(x)}.$$

To get

$$\frac{d}{dx}\mathbb{P}_x(|\gamma| = n) = C_n\frac{nx^{n-1}C(x) - x^n C'(x)}{C(x)^2} = 0$$

we have to satisfy

$$n = x\frac{C'(x)}{C(x)}.$$

The term $x\frac{C'(x)}{C(x)}$ is exactly the expected value of N.

In the following we will write $\Gamma A(x)$ or simply ΓA for the Boltzmann sampler with parameter x which uniformly generates objects of the combinatorial structure $\mathcal{A}$.

7.1.1 Building a Boltzmann sampler

To build a working sampler we make use of the fact that many combinatorial structures can be defined by basic set operations. We can reduce the problem of generating complex objects to generating simpler ones.

We will write $\texttt{Bern}(x)$ for a random sample drawn from a Bernoulli distribution with parameter x. Analogously we will write $\texttt{Geom}(x)$, $\texttt{Pois}(x)$, $\texttt{NonzeroPoisson}(x)$, $\texttt{Loga}(x)$ and $\texttt{Uniform}(x)$ for a geometric distribution, Poisson distribution, nonzero Poisson distribution, logarithmic distribution and uniform distribution.

Disjoint union: Suppose we have combinatorial structures $\mathcal{A}$ and $\mathcal{B}$ with $\mathcal{C} = \mathcal{A} \dot{\cup} \mathcal{B}$. If we already have Boltzmann samplers ΓA and ΓB we can proceed as follows. The Boltzmann model for $\mathcal{C}$ is

$$\mathbb{P}_x(\gamma) = \frac{x^{|\gamma|}}{A(x) + B(x)}.$$

The probability to generate an object of the class $\mathcal{A}$ is

$$\mathbb{P}_x(\gamma \in \mathcal{A}) = \sum_{\gamma \in \mathcal{A}} \frac{x^{|\gamma|}}{A(x) + B(x)} = \frac{A(x)}{A(x) + B(x)}.$$

Thus, we obtain the sampler ΓC, which uses ΓA with probability $A(x)/C(x)$ and ΓB with probability $B(x)/C(x)$. We write

$$\Gamma C(x) = \left(\texttt{Bern}\left(\frac{A(x)}{C(x)}\right) \to \Gamma A(x) \middle| \Gamma B(x) \right),$$

which reads as follows: If a Bernoulli random variable is 1 then execute $\Gamma A(x)$. Else, execute $\Gamma B(x)$.

Cartesian product: Let $\mathcal{C} = \mathcal{A} \times \mathcal{B}$ and let $\gamma = (\alpha, \beta) \in \mathcal{C}$ be some fixed element in $\mathcal{C}$. Then we have

$$\mathbb{P}_x(\gamma) = \frac{x^{|\gamma|}}{C(x)} = \frac{x^{|\alpha|+|\beta|}}{A(x)B(x)} = \frac{x^{|\alpha|}}{A(x)} \cdot \frac{x^{|\beta|}}{B(x)},$$

which is exactly the product of the probabilities for obtaining α and β with ΓA and ΓB independently. Thus, the sampler ΓC returns just the ordered pair results from the samplers ΓA and ΓB,

$$\Gamma C(x) = (\Gamma A(x), \Gamma B(x)).$$

Sequences: Let $\mathcal{C} = \textsc{Seq}(\mathcal{A})$ be the class of finite sequences of elements from $\mathcal{A}$ and $\gamma \in \mathcal{C}$. Then we have

$$\mathbb{P}_x(\gamma \in \mathcal{A}^k) = \frac{A(x)^k}{C(x)} = A(x)^k \left(1 - A(x)\right),$$

which is a geometric distribution with parameter $\lambda = 1 - A(x)$. The geometric distribution $\mathbb{P}(X = k) = (1-\lambda)^k\lambda$ can also be implemented as an iterated Bernoulli experiment until the first successful outcome. We get the sampler

$$\Gamma C(x) = \Big((\Gamma A(x))^k \text{ with } k = \texttt{Geom}(1 - A(x))\Big),$$

or since $C(z) = 1 + A(z)C(z)$ we can also define

$$\Gamma C(x) = \Big(\texttt{Bern}(1 - A(x)) \to \Gamma 1 \Big| (\Gamma A(x), \Gamma C(x))\Big).$$

Note that $\Gamma 1$ is the Boltzmann sampler of the class $\{\epsilon\}$. This sampler always returns the object ϵ (the sequence of length 0), since the Boltzmann sampler of a singleton set always returns the only available object. This is the reason why the recursive approach terminates.

The three presented constructions hold for the ordinary Boltzmann model as well as the exponential model. Before moving on to exponential Boltzmann samplers, we want to give an example.

Example: Products with parentheses Given a product of $n+1$ numbers separated by n multiplication operations, for example $((a * (a * a)) * a)$, there are C_n ways of defining the order of multiplication by inserting parentheses. It is well known that

$$C_n = \frac{1}{n+1}\binom{2n}{n} \quad \text{and} \quad C(z) = \sum_{n\geq 0} C_n z^n = \frac{1 - \sqrt{1-4z}}{2z}.$$

The numbers C_n with $n \in \mathbb{N}$ are the Catalan numbers, or the Fuss-Catalan numbers $C_n^{(2)}$ we already observed in Theorem 6.3. Binary trees, Dyck paths and noncrossing partitions are only a few of the structures also enumerated by these numbers. The class of products with parentheses $\mathcal{P}$ can be defined by the equation

$$\mathcal{P} = \{a\} \,\dot\cup\, \mathcal{P} \times \{*\} \times \mathcal{P},$$

thus the generating function $C(z)$ satisfies

$$C(z) = 1 + zC(z)^2.$$

The corresponding Boltzmann sampler is

$$\Gamma C(x) = \left(\texttt{Bern}\left(\frac{1}{C(x)}\right) \to \Gamma 1 \,\middle|\, (\Gamma C(x), *, \Gamma C(x)) \right).$$

Here the sampler $\Gamma 1$ always returns the product "a" with only one number and $n = 0$ multiplication signs and $(\Gamma C(x), *, \Gamma C(x))$ denotes a triple consisting of a a randomly sampled product, a multiplication sign and another product. To generate a product of size n we have to find a suitable value for the parameter x. We find that

$$\mathbb{E}_x(N) = x\frac{C'(x)}{C(x)} = \frac{1}{2}\left(\frac{1}{\sqrt{1-4x}} - 1\right).$$

For $n \in \mathbb{N}$ and $n = \mathbb{E}_x(N)$ we find that

$$x = \frac{1}{4}\left(1 - \frac{1}{(2n+1)^2}\right).$$

If, for example, we want to generate a product with $n = 5$ multiplication signs, we set the parameter x to $30/121$.

The probability that the Boltzmann sampler with parameter x that satisfies $n = \mathbb{E}_x(N)$ returns a product with exactly $n + 1$ numbers is therefore

$$\begin{aligned}
\mathbb{P}_x(|\gamma| = n) &= C_n \frac{x^n}{C(x)} \\
&= \frac{1}{n+1}\binom{2n}{n} \frac{\frac{1}{4^{n+1}}\left(1 - \frac{1}{(2n+1)^2}\right)^{n+1}}{\frac{1}{2}\left(1 - \frac{1}{2n+1}\right)} \\
&\sim \frac{1}{2n^{3/2}\sqrt{\pi}}\left(1 - \frac{1}{2n+1}\right)^n \left(1 + \frac{1}{2n+1}\right)^{n+1} \\
&\sim \frac{1}{2n^{3/2}\sqrt{\pi}}.
\end{aligned}$$

The value for $\mathbb{P}_x(|\gamma| = n)$ with $n = 5$ is $\mathbb{P}_x(|\gamma| = 5) \approx 0.0214628444286$. Thus, on average we have to sample ≈ 47 products to get one with size 5. This

Listing 7.1: A Boltzmann sampler for products with $n+1$ factors and n multiplication operations, written in Python.

```
from math import sqrt
from scipy.special import binom
from scipy.stats import bernoulli

def C(x):
    return (1-sqrt(1-4*x))/(2*x)

def construct(p):
    if bernoulli.rvs(p):
        return "a"
    else:
        return "(" + construct(p) + "*" +construct(p) + ")"

def sample_product(n):
    x=1/4*(1-1/(2*n+1)**2)
    p=1/C(x)
    product=""
    while product.count("*")!=n:
        product=construct(p)
    return product
```

approach is known as the *rejection method*: Generate random samples until a sample satisfies certain conditions. In our case, this results in a uniform distribution on all objects of size n.

The sampler for products with parentheses including the rejection method is easily implemented for example in Python as shown in Listing 7.1. A language like C++ is of course more suitable for serious experiments. We only want to show the ease of implementation and choose Python in favour of readability and a more compact code.

7.1.2 Exponential Boltzmann samplers

When dealing with labelled objects we use exponential generating functions to describe counting sequences. The disjoint union construction for Boltzmann samplers works exactly like in the unlabelled case with ordinary generating functions. For EGFs $\hat{A}(z)$ and $\hat{B}(z)$ the product $\hat{C}(z) = \hat{A}(z) \cdot \hat{B}(z)$ corresponds to the labelled product $\mathcal{C} = \mathcal{A} \star \mathcal{B}$. A sampler ΓC for the labelled product is obtained in the following manner:

- Generate objects with ΓA and ΓB independently. Let us call these objects α and β.
- Randomly assign the labels $\{1, \ldots, |\alpha| + |\beta|\}$ to the atoms in (α, β).

Sampling finite sequences of labelled objects is again a combination of disjoint union and the labelled product.

Sets: Let $\mathcal{A}$ be a labelled combinatorial structure and $\mathcal{C} = \textsc{Set}(\mathcal{A})$. The EGFs of $\mathcal{A}$ and $\mathcal{C}$ satisfy $\hat{C}(z) = \exp(\hat{A}(z))$. The Boltzmann sampler generates an object $\gamma \in \mathcal{C}$ with the probability

$$\mathbb{P}_x(\gamma) = \frac{1}{\hat{C}(x)} \frac{x^{|\gamma|}}{|\gamma|!}.$$

Furthermore the probability for γ to be a set of k components from $\mathcal{A}$ is

$$\frac{1}{\hat{C}(x)} \frac{\hat{A}(x)^k}{k!} = e^{-\hat{A}(x)} \frac{\hat{A}(x)^k}{k!},$$

which is a Poisson distribution with parameter $\lambda = \hat{A}(x)$. It is defined by the law $\mathbb{P}(X = k) = e^{-\lambda}\lambda^k/k!$. This gives the Boltzmann sampler

$$\Gamma C(x) = \Big((\Gamma A(x))^k \text{ with } k = \texttt{Pois}(\hat{A}(x)) \Big),$$

where $(\Gamma A(x))^k$ denotes the labelled product of k independently sampled objects of $\mathcal{A}$.

Cycles: For the sake of completeness we also give a Boltzmann sampler for the cycle construction mentioned in Section 2.3. Let $\mathcal{C} = \textsc{Cyc}(\mathcal{A})$, thus

$$\hat{C}(z) = \log\left(\frac{1}{1-\hat{A}(z)}\right) = \sum_{k\geq 1}\frac{\hat{A}(z)^k}{k}.$$

The probability for a sampled object γ to be a cycle of k components from $\mathcal{A}$ is

$$\frac{1}{\hat{C}(x)}\frac{\hat{A}(x)^k}{k} = \frac{1}{-\log(1-\hat{A}(x))}\frac{\hat{A}(x)^k}{k},$$

which is a logarithmic distribution with parameter $\lambda = \hat{A}(x)$. The logarithmic distribution is defined by the law

$$\mathbb{P}(X=k) = \frac{-1}{\log(1-\lambda)}\frac{\lambda^k}{k}.$$

The Boltzmann sampler is essentially the same as for the set construction, where the Poisson distribution is simply replaced by a logarithmic distribution,

$$\Gamma C(x) = \Big((\Gamma A(x))^k \text{ with } k = \texttt{Loga}(\hat{A}(x))\Big).$$

7.2 Sampling automata

In this section we want to efficiently sample ICDFAs with n states over an alphabet of size k. Thus, we have to build a sampler for set partitions of kn elements with n parts. We recall the set partitions we counted in Section 6.2 and their EGF. Let $\mathcal{P}_n$ be the class of set partitions with n parts, then the EGF of $\mathcal{P}_n$ is

$$P_n(z) = \frac{(e^z-1)^n}{n!}.$$

We will now observe the exponential Boltzmann model of $\mathcal{P}_n$. Let M be the size of the partition generated by the Boltzmann sampler with parameter x. The expected value of M is

$$\mathbb{E}_x(M) = x\frac{\mathcal{P}_n'(x)}{\mathcal{P}_n(x)} = nx\frac{e^x}{e^x-1}.$$

Our goal is to generate an object of size kn, thus $nx\frac{e^x}{e^x-1} = kn$. We see that the parameter x only depends on k and has the same value as ζ which we calculated during the application of Theorem 6.10. Therefore $x = W_0(-ke^{-k}) + k$. The exponential generating function of the nonempty sets is $N(z) = e^z - 1$, thus the probabilities of the Boltzmann model are those of a nonzero Poisson distribution,

$$\mathbb{P}(X = k) = \frac{\lambda^k}{(e^\lambda - 1)k!} \quad \text{for} \quad k \geq 1$$

with parameter $\lambda = x$. The Boltzmann sampler is therefore

$$\Gamma P_n(x) = \Big(\text{return a relabelled set of } n \text{ independent calls of } \Gamma N(x)\Big) \quad \text{and}$$

$$\Gamma N(x) = \Big(\text{return the set } \{1, \ldots, k\} \text{ with } k = \texttt{NonzeroPois}(x)\Big).$$

There is a simple algorithm that generates random values according to the nonzero Poisson law using only a uniform random number generator. See Algorithm 3 for the pseudo code.

Algorithm 3 Nonzero Poisson

```
1: procedure NonzeroPois(λ)
2:     k ← 1
3:     p ← λ(e^λ − 1)^{−1}
4:     u ← Uniform([0, 1[)
5:     while u ≥ p do
6:         u ← u − p
7:         k ← k + 1
8:         p ← λp/k
9:     return k
```

The probability that the Boltzmann sampler $\Gamma P_n(x)$ succeeds in generating a partition of size kn is

$$\mathbb{P}_x(N = kn) = \frac{x^{kn}}{P_n(x)}[z^{kn}]P_n(z) = \frac{n!x^{kn}}{(e^x - 1)^n}\frac{\left\{{kn \atop n}\right\}}{(kn)!}.$$

Using the estimate for the Stirling numbers from Section 6.2 we deduce that the average number of rejections is $\mathcal{O}(\sqrt{n})$. The code in Listing 7.2 shows a possible implementation of a sampler including the rejection method.

Listing 7.2: A Boltzmann sampler for partitions of kn elements into n parts, written in Python.

```
from math import e
from scipy.special import lambertw
import random

def nonzero_pois(x):
    k=1
    p=x/(e**x-1)
    u=random.random()
    while u >=p:
        u-=p
        k+=1
        p*=x/k
    return k

def sample_partition(n,k):
    zeta=lambertw(-k*e**(-k))+k
    sizes=[]
    while sum(sizes)!=n*k:
        sizes=[]
        for _ in range(n):
            sizes.append(nonzero_pois(zeta))
    labels=list(range(1,k*n+1))
    random.shuffle(labels)
    partition=[]
    for i in sizes:
        l = []
        for _ in range(i):
            l.append(labels.pop())
        partition.append(l)
    return partition
```

We bring together the previous results and give the random sampling algorithm for ICDFAs with n states and alphabet size k:

S1 Calculate $x = W_0(-ke^{-k}) + k$.

S2 Sample a set partition with $\Gamma P_n(x)$ until it produces a partition of size kn.

S3 Transform the partition to a boxed diagram of width $(k-1)n$ and height n as explained in Lemma 6.8.

S4 If the resulting diagram does not satisfy the Dyck condition mentioned in Definition 6.7, go to S2.

S5 Append a column of height n with one randomly chosen marked box.

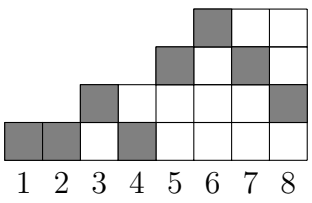
S6 Transform the k-Dyck boxed diagram of size n obtained this way to an ICDFA$_\emptyset$ $M = (Q, \Sigma, \delta, q_0)$ according to the inverse algorithm to Algorithm 2, which gave a bijection from k-Dyck boxed diagrams to ICDFA$_\emptyset$s.

S7 Randomly add final states, i.e. for every state $q \in Q$ decide whether q is final with probability $1/2$.

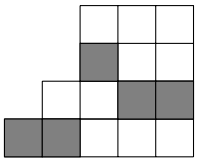
Let us demonstrate the whole procedure by completely showing one possible scenario. Let $n = 4$ and $k = 2$. Suppose that the sampler outputs the partition

$$\{\{1, 2, 4\}, \{3, 8\}, \{5, 7\}, \{6\}\}\,.$$

We get the following diagram:



After removing the leftmost column of each height and appending one column of height 4 we have the following diagram:



Luckily the Dyck condition holds for this boxed diagram. With the sequences Max = $(1, 2, 4, 4, 4)$ and Boxed = $(1, 1, 3, 2, 2)$ the ICDFA$_\emptyset$ shown in Figure 7.2 is obtained.

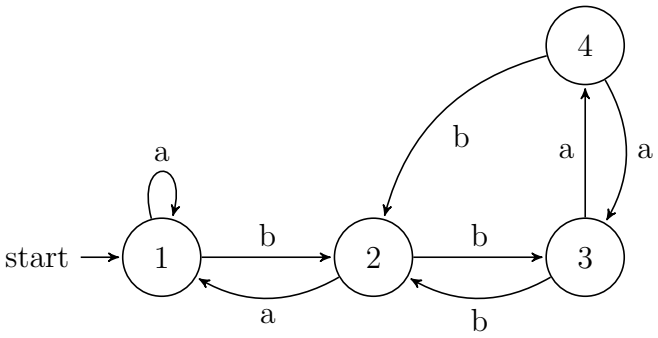


Figure 7.2: An ICDFA$_\emptyset$ generated with a Boltzmann sampler

There are now 2^4 possibilities to randomly choose final states to get an ICDFA.

7.2.1 An open conjecture

This sampler can be extended easily to generate minimal DFAs by adding a last rejection step:

S8 If the resulting automaton is not minimal go to S2.

Using this this method, *Bassino and Nicaud* [3] observed through experiments that on average, less than two draws are enough to sample a minimal DFA over an alphabet of two or more letters. This suggests that a constant percentage of ICDFAs is minimal. The results obtained by *Champarnaud and Paranthoën* [4] show that the empirical probability of a two letter DFA to be minimal is approximately 80%. Furthermore through tests with larger alphabets and 100

states it was observed that almost all DFAs are minimal with the exception of automata with trivial finalities, i.e. automata with no final states or all final states. A proof for this claim has yet to be found. It would guarantee the efficiency of the random sampler of minimal DFAs and provide an estimate of the form $f_k(n) = \Theta(n2^n \left\{ {kn \atop n} \right\})$ for the number of minimal DFAs.

Bibliography

[1] Marco Almeida, Nelma Moreira, and Rogério Reis. Enumeration and generation with a string automata representation. *Theoretical Computer Science*, 387(2):93–102, 2007.

[2] Ashwag Alrehily, Ruqiah Fallatah, and Vijey Thayananthan. Design of Vending Machine using Finite State Machine and Visual Automata Simulator. *International Journal of Computer Applications*, 115(18), 2015.

[3] Frédérique Bassino and Cyril Nicaud. Enumeration and random generation of accessible automata. *Theoretical Computer Science*, 381(1-3):86–104, 2007.

[4] Jean-Marc Champarnaud and Thomas Paranthoën. Random generation of DFAs. *Theoretical Computer Science*, 330(2):221–235, 2005.

[5] Michael Domaratzki, Derek Kisman, and Jeffrey Shallit. On the Number of Distinct Languages Accepted by Finite Automata with n States. *Journal of Automata, Languages and Combinatorics*, 7(4):469–486, 2002.

[6] Philippe Duchon, Philippe Flajolet, Guy Louchard, and Gilles Schaeffer. Boltzmann Samplers for the Random Generation of Combinatorial Structures. *Combinatorics, Probability and Computing*, 13(4-5):577–625, 2004.

[7] R. C. Entringer. Functions and Inverses of Asymptotic Functions. *The American Mathematical Monthly*, 74(9):1095–1097, 1967.

[8] Philippe Flajolet and Robert Sedgewick. *Analytic Combinatorics*. Cambridge University Press, 2009. accessed 17.04.2017. URL: `http://algo.inria.fr/flajolet/Publications/book.pdf`.

[9] Irving J. Good. An asymptotic formula for the differences of the powers at zero. *The Annals of Mathematical Statistics*, 32(1):249–256, 1961.

[10] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik. *Concrete Mathematics*. Addison-Wesley, second edition, 1994.

[11] Godfrey H. Hardy and Edward M. Wright. *An Introduction to the Theory of Numbers*. Oxford at the Clarendon Press, 4th edition, 1960.

[12] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.

[13] Aleksey D. Korshunov. Enumeration of finite automata. *Problemy Kibernetiki*, 34:5–82, 1978. (in Russian).

[14] Aleksey D. Korshunov. On the number of non-isomorphic strongly connected finite automata. *Journal of Information Processing and Cybernetics*, 22(9):459–462, 1986.

[15] S. Rao Kosaraju. On independent circuits of a digraph. *Journal of Graph Theory*, 1(4):379–382, 1977.

[16] Valery A. Liskovets. The number of initially connected automata. *Kibernetika*, 3:16–19, 1969.

[17] Valery A. Liskovets. Exact enumeration of acyclic deterministic automata. *Discrete Applied Mathematics*, 154(3):537–551, 2006.

[18] Mehryar Mohri, Fernando Pereira, and Michael Riley. Weighted Finite-State Transducers in Speech Recognition. *Computer Speech & Language*, 16(1):69–88, 2002.

[19] Nelma Moreira and Rogério Reis. On the density of languages representing finite set partitions. *Journal of Integer Sequences*, 8(05.2):8, 2005.

[20] Carl Pomerance, John M. Robson, and Jeffrey Shallit. Automaticity II: Descriptional complexity in the unary case. *Theoretical computer science*, 180(1):181–201, 1997.

[21] Robert W. Robinson. Counting strongly connected finite automata. In *Graph theory with applications to algorithms and computer science*, pages 671–685. Wiley, 1985.

[22] J. Barkley Rosser and Lowell Schoenfeld. Approximate formulas for some functions of prime numbers. *Illinois Journal of Mathematics*, 6:64–94, 1962.

[23] Jürgen Schmidthammer. Catalan-Zahlen. Zulassungsarbeit zum Staatsexamen, February 1996. accessed 17.04.2017. URL: `http://www.bnv-bamberg.de/home/ba2636/catalanz.pdf`.

[24] Richard P. Stanley. *Enumerative Combinatorics Vol. II.* Cambridge University Press, 2001.

[25] Richard P. Stanley. *Enumerative Combinatorics: Volume 1.* Cambridge University Press, 2nd edition, 2011.

[26] Victor A. Vyssotsky. A counting problem for finite automata. Technical report, Bell Telephone Laboratories, May 1959.

[27] Don Zagier. Newman's Short Proof of the Prime Number Theorem. *The American mathematical monthly*, 104(8):705–708, 1997.