Stefan Hammer, BSc

# The Wiener Index and Automata

**MASTER'S THESIS**

to achieve the university degree of

Diplom-Ingenieur

Master's degree programme: Mathematics

submitted to

**Graz University of Technology**

Supervisor

Priv.-Doz. Dr. Daniele D'Angeli

Institute of Discrete Mathematics

Graz, July 2019

## AFFIDAVIT

I declare that I have authored this thesis independently, that I have not used other than the declared sources/resources, and that I have explicitly indicated all material which has been quoted either literally or by content from the sources used. The text document uploaded to TUGRAZonline is identical to the present master‘s thesis.

| | |
|---|---|
| Date | Signature |

# Acknowledgements

First I would like to thank my thesis advisor Priv.-Doz. Dr. Daniele D'Angeli of the Institute of Discrete Mathematics at TUGraz for introducing me to the interesting topics I was able to research for my master thesis. He provided me with very helpful references and guidance, but still allowed this thesis to be my own work.

Further sincere thanks go to my very good friend and fellow student Christian Lindorfer. Through his valuable suggestions and constructive criticism during the writing process the quality of this document was greatly improved. Moreover I am very grateful to my close friend Daniel Hischenhuber for providing his assistance and pointing out some mistakes.

Lastly, I would like to thank my parents for all their encouragement and support.

# Abstract

The Wiener index is the sum of all distances in a given graph and thus a measure of connectedness. We examine its basic properties and exhibit some known lower and upper bounds. Furthermore we use it to show that the average distance of graphs in a growing sequence of graphs with vertices of bounded degree tends to infinity. Then we discuss methods to calculate the Wiener index and present two algorithms. The first algorithm is well known and can be used for every graph, whereas the second is newly developed and works very efficiently but can only be applied to trees.

A (Mealy) automaton is a machine that is located in a state, it receives an input symbol and depending on the current state and the input symbol it moves to the next state and produces an output symbol. We provide the basic concepts of automata and indicate the requirements necessary to generate associated groups called automata groups. To visualize these groups we introduce Schreier graphs and give some examples. Moreover we exhibit the connection between the automaton group and the free group generated by the states of the automaton.

One very interesting automaton is the so called Basilica automaton. After its formal introduction we take a closer look on the Schreier graphs of its automaton group and examine their structures. Using these insights we prove a new result concerning the exact growth rate of the Wiener index of the Schreier graphs of the Basilica automaton.

# Contents

# List of Figures

# Preface

In 1947 the chemist Harry Wiener published that the boiling points of paraffins can be approximated by $t = aw + bp + c$, where $a, b, c$ are constants that depend on the given isomeric group, $p$ is the polarity number and $w$ is the path number. This path number is defined as the sum of all carbon-carbon bonds between all pairs of carbon atoms (see [40]). In other words, it is the sum of all distances in the appropriate graphical representation of the chemical structure. Later the path number became known as the Wiener index, sometimes also called Wiener number, and its significance in the research of correlations between chemical properties and the structure of molecules was acknowledged (see [33]).

There are various other important applications (see e.g. [10]) and connections to interesting graph properties. For example, the Wiener index is directly related to the average distance of a graph, which is a natural measure for the compactness of a graph. The average distance is especially useful in the fields of communication and architecture (e.g. see [27]).

Thus it is not surprising that a lot of mathematical work has been dedicated to the research of the Wiener index. One major subject is the calculation of the Wiener index on trees. An extensive summary of various results and applications of the Wiener index on trees by Dobrynin et al. can be found in [11]. A different approach was pursued by Gutman et al. in [22]. They examined the inverse problem for the Wiener index, i.e. which natural numbers can occur as the Wiener index of a graph. They were able to prove that every positive integer, except for 2 and 5, is the Wiener index of some connected graph. This research was extended by restricting the type of graphs to a certain class. Wagner proved a similar result for trees in [39] and the analogue for bipartite graphs is shown in [21] by Gutman and Yeh. The aim in this master thesis is to examine the Wiener index in a very different setting, namely combined with the theory of finite automata.

The first finite automata-like structure was defined in 1943 by McCulloch and Pitts in [28], although a mathematical model of a computing machine was already introduced in 1936 by Turing (see [38]). A little more than a decade later Mealy and Moore generalized this theory and published independent papers (see [29] and [31], respectively), that became the foundation of the finite automata theory.

Finite automata, sometimes also called finite-state machines, can be divided into the two basic types acceptor and transducer. Both types are essentially machines that move from a current state to a new state when receiving an input symbol, where the movement depends on the current state and the input symbol. This can be represented by a tuple consisting of a finite set of states, a finite set of input symbols and a transition function that maps a state and an input symbol to a new state. The transition function is then naturally extended to the set of finite words over the input symbols, by simply applying the transition on the symbols successively. The acceptor has additionally given an initial state and a set of final states. A finite word over the input symbols is accepted, if the automaton moves with the given word as input from the initial state to a final state, otherwise it is rejected.

Acceptors are closely related to the concept of formal languages. In fact, Kleene's theorem states, that the language accepted by a finite automaton is a regular language and every regular language is accepted by some finite automaton (see [24] or [35]). Transducers are also related to languages, but in a different way.

In addition to the two sets and the transition function the transducer has a set of output symbols which may coincide with the set of input symbols, and an output function. The output function always maps into the set of output symbols, but the domain depends on a further classification into Mealy and Moore machines. For the Mealy machine the output is determined by the current state and the input symbol, so the domain is the Cartesian product of the set of states and the set of input symbols. The output of the Moore machine depends only on the new state, hence the domain of the output function is the set of states.

Similar to the transition function the output function can be extended, such that a finite word over the input symbols can be processed and an output word of the same length is generated. This is the main usage of the transducer. Common examples are vending machines, traffic lights and bar code scanners. In the sense of producing output Mealy and Moore machines are equivalent, that means for every Mealy machine there is a Moore machine that produces the same output and vice versa, albeit Moore machines tend to have more states. A proof of this fact and more details can be found in Section 3.1 of [36]. Note that in [36] both Mealy and Moore machines have an initial state like acceptors. This is often called an initial automaton.

Here we are interested in Mealy machines having a common set of input and output symbols and simply call them (finite) automata. An automaton of this type is called invertible if the output function restricted to any state is a bijection on the set of input and output symbols. The set of finite words over the input and output symbols can be endowed with the structure of a regular rooted tree and under these considerations the output function restricted to a state is a graph-automorphism on the tree. Hence we can use the restrictions of the output function to states in order to generate a subgroup of the automorphisms on the set of finite words over the input and output symbols. Such groups arising from invertible automata are called automata groups. They were introduced in the beginning of the 1960s (see [14] and [23]), but it took several years until the significance and utility of automata groups were realized. The first major results were contributions to the General Burnside Problem, one of the most famous problems in algebra. The General Burnside Problem asks, whether there are finitely generated infinite torsion groups (more information can be found in [1] and [19]). Various mathematicians used automata in the 1970s and 1980s to construct such groups (see [2], [37], [15] and [20]). Among them was GRIGORCHUK, who also utilized automata groups to show the existance of infinite groups having intermediate growth (more than polynomial and less than exponential, see [16] and [17]), solving another famous problem stated by MILNOR (see [30]).

A structure of great interest in the study of automata groups is the Schreier graph. For a finitely generated group acting on a set, we take the members of the set as vertices and connect them in accordance to the action of a finite set of generators. Considering an automaton group we have an action on the set of finite words over the input and output symbols. Since the length of words is not changed by the action, we get an infinite disconnected graph. Hence we restrict the action to words of a certain length. This leads to the main topic of this thesis, the examination of the growth of the Wiener index of the increasing sequence of Schreier graphs given by an automaton.

In the first chapter we define the class of graphs we are working with and introduce the Wiener index. Then we show some known basic bounds and talk about the algorithmic calculation of the Wiener index. Furthermore we present a new and very efficient algorithm for calculating the Wiener index of a tree.

The second chapter discusses the automaton group. We start by introducing group actions and Schreier graphs. Then we establish the basic automata theory and take a closer look on the automaton group.

The central topic of the third chapter is a new result concerning the Basilica automaton. After defining the Basilica automaton we show how its Schreier graphs can be generated recursively. This is the main ingredient required to determine the order of growth of the Wiener index on the Schreier graphs of the Basilica automaton.

# Chapter 1

# The Wiener index and the average distance

The aim of the first section of this chapter is to define the basic terminology necessary to work with the Wiener index. Usually the Wiener index is studied on simple graphs. However, we will have graphs with parallel edges and loops later on, so we start with a more involved introduction to graph theory. We conclude the section by showing simple known lower and upper bounds of the Wiener index and identify graphs achieving one of the bounds. These bounds can also be found in [12] with some slight alterations and alternative proofs.

In the second section we construct a lower bound of the Wiener index depending on the maximum degree of the given graph. This leads to an interesting result about graph sequences with increasing number of vertices.

The third section is dedicated to the algorithmic calculation of the Wiener index. First we show some basic approaches using algorithms solving the All Pairs Shortest Path Problem and compare their running times. Then we talk about the difference between the All Pairs Shortest Path Problem and the calculation of the Wiener index and mention some classes of graphs for which the calculation of the Wiener index is indeed easier than solving the All Pairs Shortest Path Problem. Lastly we present a newly developed algorithm that calculates the Wiener index of a tree and has linear running time.

## 1.1 Basic notation and simple bounds

**Definition 1.1.** An (undirected) graph $G = (V, E)$ consists of a *set of vertices* $V$, a *set of edges* $E$ and an *incidence relation*. The incidence relation pairs each edge with a set of one or two vertices. Thus if $e$ is incident to $\{u, v\}$ we usually write $e = \{u, v\}$ or $e = \{v, u\}$, although there may be more edges with this relation. If there is another edge $f = \{u, v\}$, we say $e$ and $f$ are *parallel*. Note that $u$ and $v$ may be equal, in this case we call $e$ a *loop*. If $G$ has no loops and no parallel edges it is called *simple*.

For $\{u, v\} \in E$ with $u \neq v$, we say that $u$ and $v$ are *adjacent*, or they are connected by an edge. Furthermore we call $u$ a *neighbour* of $v$ and vice versa. $\mathcal{N}_G(v) := \{w \in V \mid \{v, w\} \in E\}$ is the *set of neighbours* of $v$. The *degree* of a vertex $v$ is denoted by $\deg_G(v)$. It is the number of edges incident to $v$ where loops are counted twice. If there is an $r \in \mathbb{N}_0$ such that $\deg_G(v) = r$ for all vertices $v \in V$, then the graph is called $r$-regular.

Let $G' = (V', E')$ be a second graph. $G \cup G' := (V \cup V', E \cup E')$ is the union of the graphs $G$ and $G'$, where its incidence relation is obtained by the union of the incidence relations of $G$ and $G'$. The vertex and edge sets, respectively, may not be disjoint, so in case an edge $e$ is contained in both graphs and is incident to different vertices in $G$ and $G'$, we add a new edge $e'$ to the union and use for $e$ the incidence of $G$ and for $e'$ the incidence of $G'$. We say $G'$ is a *subgraph* of $G$, written $G' \subset G$, if $V' \subset V$, $E' \subset E$ and the incidence relation of $G'$ is the incidence relation of $G$ restricted to the edges and vertices of $G'$. If there exists a bijective map $\varphi : V \to V'$, where for all $v, w \in V$ the sets $\{v, w\}$ and $\{\varphi(v), \varphi(w)\}$ are incident to the same number of edges, then we say $G$ and $G'$ are *isomorphic*, written $G \cong G'$. Such a map $\varphi$ is called *(graph) isomorphism.* If additionally $G = G'$ then $\varphi$ is also said to be an *automorphism* on $G$.

**Remark 1.2.** If $G = (V, E)$ is a simple graph we can identify each edge with the two incident vertices, i.e.

$$E \subset \binom{V}{2} := \{\{u, v\} \mid u, v \in V, u \neq v\}. \tag{1.1}$$

Furthermore the inequality $|\mathcal{N}_G(v)| \leq \deg_G(v)$, holding for any $v \in V$, simplifies to an equality.

**Definition 1.3.** Let $G = (V, E)$ be a graph. The *simplification* of $G$ is the simple graph $H = (V, E')$ where $E'$ consists of the edges $\{u, v\}$ for all vertices $u, v \in V$, $u \neq v$, that are adjacent.

**Remark 1.4.** The simplification of a graph can also be defined by successively deleting parallel edges and loops from the edge set until the graph is simple. Using this process the simplification is not unique, but all graphs constructed in this manner are isomorphic.

The graph properties used in this chapter are invariant under simplification, so we can always use the simplified graph and get the same result. Thus we will not explicitly state it but treat graphs as if they were simple.

Sometimes it is useful to characterize edges by more than their incident vertices. Hence we introduce labels and a labelling function.

**Definition 1.5.** A graph $G = (V, E)$ is called *(edge-)labelled*, if it has a label function $l_G : E \to L$ attached, where $L$ is a finite non-empty set of labels. We say $e \in E$ is labelled (by) $a \in L$ if $l_G(e) = a$.

Let $G = (V, E)$ and $G' = (V', E')$ be isomorphic graphs with the label functions $l_G : E \to L$ and $l_{G'} : E' \to L$, respectively, and the same set of labels $L$. An isomorphism $\varphi : V \to V'$ is called label preserving if for all $v, w \in V$ and $a \in L$, $\{v, w\}$ and $\{\varphi(v), \varphi(w)\}$ are incident to the same number of edges labelled by $a$. $G$ and $G'$ are called *label-isomorphic* if a label preserving isomorphism from $V$ to $V'$ exists.

Walks and paths are a basic concept of graphs, but the specific definitions vary from author to author. Since we are interested in distances, we only need sequences of non-repeating vertices, where all consecutive vertices are connected via edges, and call them paths.

**Definition 1.6.** Let $G = (V, E)$ be a graph. A *path* in $G$ is a sequence of pairwise distinct vertices $\pi = (v_0, v_1, \dots, v_m)$, where $m \in \mathbb{N}_0$ and $\{v_i, v_{i+1}\} \in E$ for $i = 0, \dots, m-1$. We say $\pi$ *connects* the startpoint $v_0$ and the endpoint $v_m$ and the edges $\{v_i, v_{i+1}\}$ are (contained) in the path. The *length* of $\pi$ is the number of edges in the path, where only one edge is considered per pair of vertices $\{v_i, v_{i+1}\}$. It is equal to the number of vertices in the path minus one and denoted by $|\pi| = m$.

A path $\pi = (v_0, v_1, \dots, v_m)$ with $m \geq 2$ and $\{v_m, v_0\} \in E$ can be extended to a *cycle* $C = (v_0, v_1, \dots, v_m, v_0)$ of length $|C| = m + 1$. A graph that does not contain a cycle is called *acyclic.*

If any two vertices in the graph are connected via some path, we say $G$ is *connected*, else it is *disconnected.* A simple, acyclic, connected graph is called *tree.* The *distance* of two vertices $u$ and $v$ in the graph, denoted by $d_G(u, v)$, is defined as the length of the shortest path connecting $u$ and $v$, if they are connected, and as $\infty$, if they are not. The maximum distance $\operatorname{diam}(G) := \max\{d_G(u, v) \mid u, v \in V\}$ is called the *diameter* of $G$. For a vertex $v \in V$ we set $d_G(v)$ to the *sum of distances* from $v$ to all other vertices, i.e.

$$d_G(v) := \sum_{u \in V \setminus \{v\}} d_G(v, u). \tag{1.2}$$

**Remark 1.7.** For every vertex $v$ the path $(v)$ of length 0 connects $v$ to $v$, therefore $d_G(v, v) = 0$ and $d_G(v)$ can be simplified to

$$d_G(v) = \sum_{u \in V} d_G(v, u). \tag{1.3}$$

Paths and cycles can also be considered as (sub-)graphs or sequences of edges, hence we may use them in this manner.

**Definition 1.8.** Let $G = (V, E)$ be a graph on 2 or more vertices, i.e. $|V| \geq 2$. The *Wiener index* $W(G)$ is defined as the sum of all distances in $G$, written

$$W(G) := \sum_{\{u,v\} \subset V} d_G(u, v)\,. \tag{1.4}$$

The *average distance* of two vertices in $G$ is the sum of all distances divided by the number of distances greater then 0, i.e.

$$\mu(G) := \frac{W(G)}{\binom{|V|}{2}}\,. \tag{1.5}$$

**Remark 1.9.** The Wiener index can also be defined as 0 for a graph with only one vertex, but the average distance is not useful in this instance and the formula cannot be extended. So we excluded this case.

In disconnected graphs the Wiener index and the average distance are infinite. In many situations this fact does not alter the results we are looking at, but we will state the connectedness explicitly if needed.

**Remark 1.10.** We can reformulate the Wiener index using $d_G(\cdot)$:

$$W(G) = \sum_{\{u,v\} \subset V} d_G(u, v) = \frac{1}{2} \sum_{u \in V} \sum_{v \in V} d_G(u, v) = \frac{1}{2} \sum_{u \in V} d_G(u)\,. \tag{1.6}$$

This gives a different formula for the average distance as well:

$$\mu(G) = \frac{W(G)}{\binom{|V|}{2}} = \frac{\sum_{u \in V} d_G(u)}{|V|(|V| - 1)}\,. \tag{1.7}$$

In order to get a deeper understanding on the Wiener index and the average distance, we want to look at some bounds for $d_G(\cdot)$. We start by introducing some more convenient graph properties.

**Definition 1.11.** Let $G = (V, E)$ be a graph, $v \in V$ a vertex and $r \in \mathbb{N}_0$. The *ball* with radius $r$ and center $v$ is the set of vertices with distance less than $r$ to $v$, i.e.

$$\mathcal{B}_v(r) := \{u \in V \mid d_G(v, u) < r\}. \tag{1.8}$$

Its boundary, the set of vertices with distance exactly $r$ to $v$, is named $\mathcal{S}_v(r)$, the *sphere* with radius $r$ and center $v$.

**Proposition 1.12.** *Let $G = (V, E)$ be a connected graph on $n$ vertices. For every vertex $v \in V$,*

$$n - 1 \leq d_G(v) \leq \frac{n(n-1)}{2} \tag{1.9}$$

*and these bounds are tight, i.e. there are graphs containing vertices achieving the lower or the upper bound.*

*Proof.* Take a vertex $v \in V$ and let $n_i$ be the number of vertices of distance $i$ to $v$, i.e. $n_i := |\mathcal{S}_v(i)|$, then

$$d_G(v) = \sum_{i=1}^{n-1} n_i i \quad \text{and} \quad n = \sum_{i=0}^{n-1} n_i \,. \tag{1.10}$$

$n_0 = 1$ and if $n_1 = n - 1$, then $n_i = 0$ for all $i \geq 2$, thus the lower bound is attained. This means that all other vertices are neighbours of $v$. In case $n_1 < n - 1$, the sum increases since $n_2 > 0$ and hence the lower bound holds.

For the upper bound take note that $n_i = 0$ implies $n_j = 0$ for all $j \geq i$. Thus the sum maximizes if $n_i = 1$ for $i = 0, \ldots, n-1$. This gives the desired upper bound:

$$d_G(v) \leq \sum_{i=1}^{n-1} i = \frac{n(n-1)}{2} \,. \tag{1.11}$$

To achieve this bound, $G$ must be a path and $v$ has to be the startpoint or the endpoint of the path. □

Let $K_n := \left(V, \binom{V}{2}\right)$ be the complete graph on $n$ vertices. This graph can be considered the 'most connected graph'. We examine it to obtain some simple lower bounds for the Wiener index and the average distance.

**Remark 1.13.** Any pair of distinct vertices $u, v$ of the complete graph satisfies $d_{K_n}(u, v) = 1$. Thus for every graph $G = (V, E)$ on $n$ vertices

$$W(G) \geq W(K_n) = \sum_{\{u,v\} \subset V} d_{K_n}(u, v) = \left| \binom{V}{2} \right| = \frac{n(n-1)}{2} \tag{1.12}$$

and $\mu(G) \geq \mu(K_n) = 1$.

Conversely a path can be seen as the 'least connected graph'. We will see, that this intuition is right and then calculate the Wiener index of the path, but before addressing this task, we need some more preparatory work.

**Definition 1.14.** Let $G = (V, E)$ be a graph and $G'$ a subgraph of $G$ with vertex set $V' \subset V$. The graph $G[V'] := (V', E')$ is the subgraph of $G$ induced by the vertex set $V'$, where $E'$ is the set of all edges in $E$ that are only incident to vertices of $V'$.

$$G - G' = G - V' := G[V \setminus V'] \tag{1.13}$$

is the subgraph of $G$ induced by $V \setminus V'$. If $G$ is connected and $G - V'$ is not, we call $V'$ a *separator*. In case the separating set $V'$ consists of a single vertex $v \in V$, we call $v$ a *cut-vertex*. Furthermore we simplify the notation to $G - v$. A maximal connected subgraph without a cut-vertex is called a *block*.

**Proposition 1.15.** *Let $G = (V, E)$ be a graph on more than 2 vertices. For every vertex $v \in V$,*

$$W(G) \leq W(G - v) + d_G(v). \tag{1.14}$$

*Proof.* Removing a vertex $v \in V$ from $G$ can only increase distances, i.e. $d_{G-v}(u,w) \geq d_G(u,w)$ for all $u, w \in V \setminus \{v\}$. This fact is enough to complete the proof:

$$W(G) = \sum_{\{u,w\} \subset V \setminus \{v\}} d_G(u,w) + \sum_{u \in V} d_G(v,u) \leq W(G-v) + d_G(v). \tag{1.15}$$

$\square$

**Remark 1.16.** In case the removed vertex $v$ is a cut-vertex, the Wiener index of $G - v$ is infinite and the bound trivially holds.

If we choose a vertex $v$ such that for all $u, w \in V \setminus \{v\}$ there is a shortest path connecting $u$ and $w$ and not containing $v$, then equality holds in (1.14).

Whenever $v$ has only one neighbour it can only be the end point of a shortest path, thus the condition in the second part of the remark is fulfilled. We use this fact in the next result.

**Proposition 1.17.** *Every connected graph $G = (V, E)$ on $n$ vertices satisfies*

$$W(G) \leq \frac{n(n^2-1)}{6}. \tag{1.16}$$

*Equality holds if $G$ is a path.*

*Proof.* We use induction on $n \geq 2$.

For $n = 2$ we have $W(G) = 1$ since $G$ is connected. Thus we have equality in (1.16).

Now suppose (1.16) holds for $n$ and let $G = (V, E)$ be a connected graph on $n + 1$ vertices. Clearly not every vertex in $V$ is a cut-vertex, so we can take $v \in V$ such that $G - v$ is a connected graph on $n$ vertices. Thus, using Proposition 1.12 and 1.15 we are able to verify the inequality:

$$\begin{aligned} W(G) \leq W(G-v) + d_G(v) &\leq \frac{n(n^2-1)}{6} + \frac{(n+1)n}{2} \\ = \quad \frac{n^3 + 3n^2 + 2n}{6} \quad &= \frac{(n+1)((n+1)^2-1)}{6}. \end{aligned} \tag{1.17}$$

Additionally we suppose that $G$ is a path and equality holds in (1.16) for paths of length $n$. Let $v$ be the endpoint of $G$. Then $G - v$ is a path on $n$ vertices and $v$ has only one neighbour. Hence equality holds in the first step of (1.17). In the proof of Proposition 1.12 we have seen that $d_G(v) = (n+1)n/2$ and thus using the induction hypothesis equality in the second step holds as well. $\square$

**Remark 1.18.** The previous proposition leads to the following upper bound for the average distance: Every connected graph $G$ on $n$ vertices satisfies

$$\mu(G) \leq \frac{n+1}{3}. \tag{1.18}$$

## 1.2 A lower bound and its application to the average distance

**Proposition 1.19.** *Let $G = (V, E)$ be a graph on $n$ vertices of bounded degree $k \geq 3$, i.e. $\deg_G(v) \leq k$ for all $v \in V$. Then for every vertex $v \in V$ and for all $m \in \mathbb{N}$,*

$$|\mathcal{B}_v(m)| \leq 1 + k\frac{(k-1)^{m-1} - 1}{k-2}. \tag{1.19}$$

*Proof.* We fix a vertex $v \in V$ and prove by induction over $i$ that

$$n_i := |\mathcal{S}_v(i)| \leq k(k-1)^{i-1}. \tag{1.20}$$

$n_1 = \deg_G(v) \leq k$, so the statement holds for $i = 1$.

Now suppose (1.20) holds for $i \geq 1$. Every vertex $u \in \mathcal{S}_v(i)$ has a neighbour in $\mathcal{S}_v(i-1)$, thus the number of neighbours of $u$ with distance $i+1$ has to be at least one less than the degree of $u$. Since the degree is bounded by $k$ we obtain $|\{w \in \mathcal{N}_G(u) \mid d_G(v, w) = i+1\}| \leq k-1$. This yields (1.20) for $i+1$:

$$n_{i+1} \leq n_i(k-1) \leq k(k-1)^i. \tag{1.21}$$

Using (1.20) we complete the proof:

$$|\mathcal{B}_v(m)| = \sum_{i=0}^{m-1} n_i \leq 1 + \sum_{i=1}^{m-1} k(k-1)^{i-1} = 1 + k\frac{(k-1)^{m-1} - 1}{k-2}. \tag{1.22}$$

□

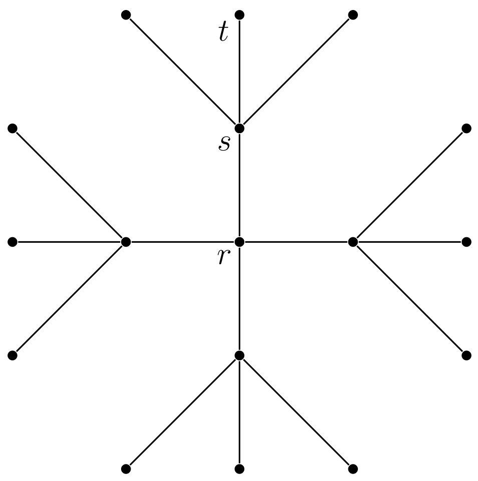


Figure 1.1: Tree for Example 1.20 and 1.22.

**Example 1.20.** We want to compare some bounds given by (1.19) for the tree in Figure 1.1. It is of bounded degree 4, so we can simplify (1.19) to

$$|\mathcal{B}_v(m)| \leq 2 \cdot 3^{m-1} - 1 \tag{1.23}$$

where $v \in V$ and $m \in \mathbb{N}_0$. Hence for $m = 1, 2, 3$ we get

$$|\mathcal{B}_v(m)| \leq \begin{cases} 1 & \text{if } m = 1, \\ 5 & \text{if } m = 2, \\ 17 & \text{if } m = 3. \end{cases} \tag{1.24}$$

By counting vertices we see that for $v = r$ equality holds in all cases, whereas for $v = s$ equality is only true for $m = 1$ and $m = 2$. However, if we take $v = t$ the inequality is strict whenever $m > 1$. So for the most choices of vertices and $m$ the inequality is strict, but there are cases of equality.

Using Proposition 1.19 we can calculate an upper bound for the number of vertices that are 'near' a given vertex $v$, but we also obtain a lower bound for the number of vertices that are 'not near'. We use the second approach in the following proposition.

**Proposition 1.21.** *Let $G = (V, E)$ be a graph on $n$ vertices of bounded degree $k \geq 3$ and*

$$\nu_k(m) := 1 + k\frac{(k-1)^{m-1} - 1}{k-2} \quad \textit{for} \quad m \in \mathbb{N}. \tag{1.25}$$

*Then for all $m \in \mathbb{N}$,*

$$W(G) \geq \frac{nm(n - \nu_k(m))}{2}. \tag{1.26}$$

*Proof.* Take a vertex $v \in V$ and $m \in \mathbb{N}$. By Proposition 1.19 at most $\nu_k(m)$ vertices have a distance of less than $m$ to $v$, so at least $n - \nu_k(m)$ vertices have a distance of at least $m$ to $v$. Thus

$$\begin{aligned} d_G(v) &= \sum_{u \in V} d_G(v, u) \geq \sum_{\substack{u \in V \\ d_G(v,u) \geq m}} d_G(v, u) \\ &\geq |\{u \in V \mid d_G(v, u) \geq m\}|\, m \geq (n - \nu_k(m))m. \end{aligned} \tag{1.27}$$

Applying this estimate on the Wiener index completes the proof:

$$W(G) = \frac{1}{2}\sum_{v \in V} d_G(v) \geq \frac{1}{2}\sum_{v \in V} (n - \nu_k(m))m = \frac{nm(n - \nu_k(m))}{2}. \tag{1.28}$$

□

**Example 1.22.** Let $T$ be the tree given in Figure 1.1. The lower bound of the Wiener index determined in the proposition above is only useful if it is positive and hence the number of vertices should be greater than $\nu_k(m) = 2 \cdot 3^{m-1} - 1$. Since $T$ has 17 vertices $m \geq 3$ does not provide reasonable bounds. Thus we calculate $nm(n - \nu_k(m))/2$ for $m = 1, 2$. This leads to $W(T) \geq 136$ for $m = 1$ and $W(T) \geq 204$ for $m = 2$.

We determine the Wiener index via calculating $d_G(\cdot)$ for $r, s$ and $t$ and using the symmetries of $T$. By counting the number of vertices of a given distance we obtain:

$$\begin{aligned} d_G(r) &= 4\cdot 1 + 12\cdot 2 &&= 28,\\ d_G(s) &= 4\cdot 1 + 3\cdot 2 + 9\cdot 3 &&= 37,\\ d_G(t) &= 1\cdot 1 + 3\cdot 2 + 3\cdot 3 + 9\cdot 4 &&= 52. \end{aligned} \tag{1.29}$$

By symmetry there are exactly 3 other vertices in $T$ that have the same $d_G(\cdot)$ as $s$ and 11 other vertices that are similar to $t$. Thus

$$W(T) = \frac{d_G(r) + 4d_G(s) + 12d_G(t)}{2} = 400. \tag{1.30}$$

This example shows that lower bounds calculated using Proposition 1.21 can be very rough. In fact most bounds determined in this way are kind of rough, but they are still tight enough to provide the following interesting result.

**Theorem 1.23.** *Let $(G_i = (V_i, E_i))_{i\in\mathbb{N}}$ be a sequence of graphs s.t. $n_i := |V_i| \to \infty$ for $i \to \infty$ and all $G_i$ are of bounded degree $k \geq 3$. Then*

$$\mu(G_i) \to \infty \quad \textit{for} \quad i \to \infty.$$

*Proof.* We fix $i$ and set $m_i := 1 + \left\lfloor \log_{k-1} \frac{n_i}{2k} \right\rfloor$. Then

$$\nu_k(m_i) \leq 1 + \frac{\frac{n_i}{2} - k}{k-2} = \frac{\frac{n_i}{2} - 2}{k-2} \leq \frac{n_i}{2}. \tag{1.31}$$

Using this inequality with Proposition 1.21 yields a useful estimate for the Wiener index:

$$W(G_i) \geq \frac{n_i m_i (n_i - \nu_k(m_i))}{2} \geq \frac{n_i^2 (1 + \lfloor \log_{k-1} \frac{n_i}{2k} \rfloor)}{4}. \tag{1.32}$$

Inserting this estimate in the definition of the average distance gives

$$\mu(G_i) = \frac{W(G_i)}{\binom{n_i}{2}} \geq \frac{n_i^2 (1 + \lfloor \log_{k-1} \frac{n_i}{2k} \rfloor)}{2n_i(n_i - 1)} \geq \frac{1}{2} \left\lfloor \log_{k-1} \frac{n_i}{2k} \right\rfloor. \tag{1.33}$$

The last term goes to infinity for $i \to \infty$ as $k$ is fixed. □

## 1.3 Algorithmic calculation of the Wiener index

Throughout this section we suppose to have a graph with $n$ vertices and $m$ edges and use the Landau-notation to compare the running times of algorithms. See section 1.2.11 of [25] for details on the Landau-notation.

A naive approach to determine the Wiener index is to use algorithms solving the All Pairs Shortest Path Problem, for example the Floyd-Warshall Algorithm (see chapter 7 in [26] for details). This algorithm is easy to comprehend and implement but needs $\mathcal{O}(n^3)$ time. Pettie used a more elaborated approach in [34] and showed that the running time can be improved to $\mathcal{O}(mn + n^2 \log\log n)$. In the worst case, if the graph is dense, the running time is still $\mathcal{O}(n^3)$, but for sparse graphs, i.e. $m = \mathcal{O}(n)$, this is a good improvement. All these algorithms are for edge-weighted graphs, so it is not surprising that there are more suitable approaches.

In a graph without weighted edges, the distance from one vertex to all others can be determined with a slightly modified breadth-first search. Therefore, by using the breadth-first search on every vertex we can solve the All Pairs Shortest Path Problem in undirected graphs. In the following we describe an algorithm that puts this idea to calculate the Wiener index into practice and prove its correctness.

---

Breadth-First Search Wiener Index Calculation (BFS-WIC)

*Input:* A connected graph $G = (V, E)$.

*Output:* The Wiener index $w = W(G)$ of the graph $G$.

---

```
(1) Set $w := 0$.
(2) For each vertex $s$ in $V$ do:
    (a) Set $R := \{s\}$, $d := 0$, $t := s$ and add the pair $(s, 0)$ to the empty queue $Q$.
    (b) For each neighbour $u$ of $t$ do:
            If $u \notin R$, then insert $(u, d+1)$ in $Q$ and set $R := R \cup \{u\}$.
    (c) Set $w := w + d$ and remove $(t, d)$ from $Q$.
    (d) If $Q$ is not empty, then take the next pair $(v, c)$ of $Q$, set $t := v$, $d := c$ and go to (b).
(3) Set $w := w/2$.
```

---

A complete and fully functional C++-implementation of the Breadth-First Search Wiener Index Calculation can be found in Appendix A.1.

**Proposition 1.24.** *The* Breadth-First Search Wiener Index Calculation *works correctly and needs $\mathcal{O}(mn)$ time and $\mathcal{O}(m)$ space.*

*Proof.* We prove that $d_G(s)$ is added to $w$ in (2) for each vertex $s$. Then the output is correct by the reformulation of the Wiener index stated in Remark 1.10.

During step (2), $w$ is only changed in (c), so we have to show that each vertex $t$ is stored in the queue $Q$ precisely once, paired with the correct distance

$d = d_G(s,t)$. Each vertex is added at least once to $Q$, since the graph is connected. Furthermore after a vertex is queued it is also added to the set $R$ and hence cannot be added a second time to $Q$. For the distance we use an inductive argument. Clearly $d_G(s,s) = 0$, thus $s$ is correctly added to the queue. Now suppose that all vertices $u$ with distance $d_G(s,u) < d$ were already stored in the queue with the correct distance $d_G(s,u)$ and all vertices $v$ with distance $d_G(s,v) < d-1$ are in $R$. Take a vertex $t$ that is added to the queue with $d$. Then it has to be the neighbour of some vertex $x$ with distance $d_G(s,x) = d-1$, hence $d_G(s,t) \leq d$. Since no vertex $v$ with distance $d_G(s,v) < d-1$ can be connected to $t$ (otherwise $t$ would have been added to $Q$ earlier, as all $v$ are already in $R$), we conclude $d_G(s,t) = d$. This completes the proof of the correctness.

Next we examine the running time. Task ① and task ③ have constant running time, thus it suffices to focus on ②. For a given vertex $s$ every vertex is queued exactly once and the time to process one queued vertex depends only on the number of its neighbours. So for one vertex the executions in ② need $\mathcal{O}(m)$ time. There are $n$ vertices, thus the complete running time is $\mathcal{O}(mn)$.

For the examination of the space that is necessary during the computation only the set $R$ and the queue $Q$ are relevant. Both contain at most $n$ elements, hence the most space is required by the graph itself. Note that $n = \mathcal{O}(m)$ since $G$ is connected, thus $\mathcal{O}(m)$ space is necessary. □

In the case of sparse graphs, the running time of the Breadth-First Search Wiener Index Calculation is quadratic in the number of vertices, this is another good improvement compared to the algorithms above. Under these conditions it is also best possible for an algorithm solving the All Pairs Shortest Path Problem since $\mathcal{O}(n^2)$ distances have to be calculated. For dense graphs the running time is still cubic in the number of vertices. This situation has been intensely studied and some of the latest results for undirected graphs without weighted edges can be found in [5] and [6]. In the first cited paper Chan presented an algorithm with running time approaching $\mathcal{O}(n^3/\log^2 n)$.

Aside from the discussion above, it is not known if it is necessary to calculate all distances in a graph in order to obtain its Wiener index. For some classes of graphs it was shown that calculating the Wiener index needs less time than solving the All Pairs Shortest Path Problem. Dankelmann proved in [9] that the average distance, and thus also the Wiener index of an interval graph can be computed in $\mathcal{O}(m)$ time. He mentioned in the same paper that his algorithm can be modified to calculate the average distance of a tree in $\mathcal{O}(n)$ time. Below we show a different and completely new way to calculate the Wiener index of a tree in $\mathcal{O}(n)$ time with a more concrete explanation.

A *benzenoid graph* is a subgraph of the hexagonal lattice induced by a closed walk and all interior vertices of the walk. It was shown in [7] that the Wiener index can also be calculated in $\mathcal{O}(n)$ time for benzenoid graphs. These are all best possible running times.

Let $T = (V, E)$ be a tree and $u \in V$ a leaf, i.e. a vertex with degree 1 in a tree. Then equality holds in (1.14), i.e.

$$W(T) = W(T - u) + d_T(u). \tag{1.34}$$

Since $u$ is a leaf it is connected to exactly one vertex $t$. As a consequence of the following proposition from [12], it turns out that $d_T(u)$ can be determined from $d_{T-u}(t)$. Thus $W(T)$ can be calculated in terms of $W(T-u)$ and $d_{T-u}(t)$.

**Proposition 1.25.** *Let $G = (V, E)$ be a graph and $u, v \in V$ neighbours. Then*

$$\begin{aligned} &d_G(v) + |\{w \in V \mid d_G(v, w) < d_G(u, w)\}| \\ = {}&d_G(u) + |\{w \in V \mid d_G(u, w) < d_G(v, w)\}|. \end{aligned} \tag{1.35}$$

*Proof.* Take a vertex $v \in V$, let $u \in \mathcal{N}_G(v)$, $w \in V$ and $\pi$ a shortest path from $v$ to $w$. Either $u$ is contained in the path $\pi$ or we can extend $\pi$ by $u$. So there is a path from $u$ to $w$ with length $\leq |\pi| + 1$. Thus $d_G(u, w) \leq d_G(v, w) + 1$. Since $v$ is also a neighbour of $u$ we can switch their roles and get analogously $d_G(v, w) \leq d_G(u, w) + 1$. Therefore $d_G(u, w)$ and $d_G(v, w)$ cannot differ by more than 1.

Next we split $V$ into 3 disjoint sets, i.e.

$$\begin{aligned} V &:= V_1 \cup V_2 \cup V_3 \text{ where} \\ V_1 &:= \{w \in V \mid d_G(v, w) = d_G(u, w)\}, \\ V_2 &:= \{w \in V \mid d_G(v, w) > d_G(u, w)\}, \\ V_3 &:= \{w \in V \mid d_G(v, w) < d_G(u, w)\}. \end{aligned} \tag{1.36}$$

Using the fact from above and this decomposition we can finish the proof:

$$\begin{aligned} d_G(v) &= \sum_{w \in V_1} d_G(v, w) + \sum_{w \in V_2} d_G(v, w) + \sum_{w \in V_3} d_G(v, w) \\ &= \sum_{w \in V_1} d_G(u, w) + \sum_{w \in V_2} (d_G(u, w) + 1) + \sum_{w \in V_3} (d_G(u, w) - 1) \\ &= \sum_{w \in V} d_G(u, w) + \sum_{w \in V_2} 1 - \sum_{w \in V_3} 1 \\ &= d_G(u) + |\{w \in V \mid d_G(v, w) > d_G(u, w)\}| \\ &\qquad - |\{w \in V \mid d_G(v, w) < d_G(u, w)\}|. \end{aligned} \tag{1.37}$$

□

This shows that the difference of $d_G(\cdot)$ for two neighboured vertices can be determined by counting the vertices that are closer to one of the two neighboured vertices. This is very convenient when comparing $d_G(\cdot)$ for neighboured vertices, especially if one of the vertices is a cut-vertex. This is the case for the setting described above.

**Corollary 1.26.** *Let $T = (V, E)$ be a tree and $u \in V$ a leaf connected to the vertex $t \in V$. Then*

$$d_T(u) = d_{T-u}(t) + |V| - 1. \tag{1.38}$$

*Proof.* All vertices except $u$ are closer to $t$ than to $u$ and

$$d_T(t) = d_{T-u}(t) + 1. \tag{1.39}$$

□

Let $T = (V, E)$ be a tree and $R$ a subset of the vertex set $V$ such that $T[R]$ is a subtree of $T$. Suppose we add a vertex $u$ to the set $R$ which is a neighbour of some vertex $t \in R$. Then by (1.34) and (1.38),

$$W(T[R \cup \{u\}]) = W(T[R]) + d_{T[R]}(t) + |R|. \tag{1.40}$$

The main idea of the Tree Wiener Index Calculation described below is the following:

Increase a set of vertices $R$ by walking through the graph like in a depth-first search and calculate $W(T[R])$ every time the set $R$ is changed. Start with a vertex $s$ and set $R := \{s\}$. Let $t$ be the currently processed vertex, $w$ the Wiener index of the current subtree induced by the set $R$ and $d := d_{T[R]}(t)$. If $t$ has a neighbour $u$ not contained in $R$, add $u$ to $R$, change the Wiener index accordingly to 1.40 and update $d$. If all neighbours of $t$ are already in $R$ and not all vertices are processed, go back to the vertex $x_t$ that was active when the vertex $t$ was added to $R$ and update $d$ to represent the sum of all distances to $x_t$ in $T[R]$. Stop the process when every vertex is contained in $R$.

---

Tree Wiener Index Calculation

*Input:* A tree $T = (V, E)$ and a starting vertex $s \in V$.

*Output:* The Wiener index $w = W(T)$ of the tree $T$.

---

```
(1) Set
        $w := 0, \quad b_s := 1,$
        $d := 0, \quad t := s,$
    $R := \{s\}$ and $a_v := 1$ for all vertices $v$ in $V$.
(2) While $R \neq V$ do:
    (a) If $\mathcal{N}_T(t) \subset R$ then set
            $d := d - b_{x_t} + a_t, \quad b_{x_t} := |R|,$
            $a_{x_t} := a_{x_t} + a_t, \quad t := x_t\,.$
```

ⓑ Else take $u \in \mathcal{N}_T(t) \setminus R$, set

$$\begin{aligned} w &:= w + d + |R|, & x_u &:= t,\\ d &:= d + |R|, & b_u &:= |R| + 1,\\ & & t &:= u \end{aligned}$$

and $R := R \cup \{u\}$.

---

**Proposition 1.27.** *The* Tree Wiener Index Calculation *works correctly and needs $\mathcal{O}(n)$ time and space.*

*Proof.* Throughout this proof $t$ denotes the active vertex at the beginning of the task or iteration and $x_t$ the active vertex when $t$ is added to $R$. We call $x_t$ the predecessor of $t$.

We show that the following assertions are true at the beginning of each loop in task ②:

(i) $w$ is the Wiener index of the tree induced by the set $R$, i.e. $w = W(T[R])$.

(ii) $d$ is the sum of all distances to the active vertex $t$ in $W(T[R])$, i.e. $d = d_{T[R]}(t)$.

(iii) If the active vertex $t$ is not equal to the start vertex $s$, then $b_{x_t}$ is the number of vertices of $T[R]$ closer to $x_t$ than to $t$, and $a_t$ is the number of vertices of $T[R]$ closer to $t$ than to $x_t$.

Subsequent to task ①, $R$ has one vertex, thus correctly $w = 0 = W(T[R])$ and $d = 0 = d_{T[R]}(t)$. So we can assume, that at the start of each iteration in ② assertions (i), (ii) and (iii) are correct.

In each loop of task ② either task ⓐ or task ⓑ is performed. First let us look at the situation after an execution of ⓑ. Assertion (i) remains true since the update of $w$ is correct by the assumptions and (1.40). Due to the assumptions and Corollary 1.26 the new value of $d$ is equal to the sum of distances to the next active vertex $u$ in $T[R \cup \{u\}]$, thus (ii) is satisfied as well. Note that $t$ must have been introduced as active vertex in the last iteration or in task ①, hence $b_t = |R|$. Furthermore $t$ is the only vertex in $T[R \cup \{u\}]$ connected to $u$, thus $b_t = |R|$ and $a_u = 1$ satisfy the claims in (iii).

In ⓐ the value of $w$ and the set $R$ are not changed, thus (i) remains true. Assertion (ii) is correct by the assumptions and Proposition 1.25. If $x_t = s$, then there is nothing to prove. In case $x_t$ is not equal to $s$, let $v$ be the predecessor of $x_t$. The most recent time $v$ was active was in the iteration where $x_t$ was added to $R$, hence $b_v$ is set to the number of vertices added to $R$ before $x_t$. This is exactly the number of vertices closer to $v$ than to $x_t$ in $T[R]$, thus the first claim of (iii) is correct. At the time the vertex $t$ was added to

$R$, $a_{x_t}$ was equal to the number of vertices that were closer to $x_t$ than to $v$ in the subtree induced by all vertices that had been added before $t$. All vertices closer to $t$ than to $x_t$ are also closer to $x_t$ than to $v$. Thus the calculation of $a_{x_t}$ in ⓐ is correct and (iii) holds.

Since for every vertex in $R$ every neighbour is added to $R$ and the tree $T$ is connected, the algorithm has to terminate with the correct Wiener index $w = W(T)$.

Clearly task ① is executed in $\mathcal{O}(n)$ time. For task ② we only need to count the number of times either task ⓐ or ⓑ is performed since both this tasks are performed in constant time. In every run of ⓑ a vertex is added to $R$, hence it is executed exactly $n-1$ times. If a vertex is active in task ⓐ it can never be active again, thus ⓐ is executed at most $n-1$ times. So in total we have a running time of $\mathcal{O}(n)$ and thus the algorithm needs $\mathcal{O}(n)$ space. □

# Chapter 2

# Automata and Schreier graphs

We start the first section of the second chapter with an introduction to free monoids and generating sets of groups. Then we define group actions and show how they are used to construct Schreier graphs.

In the second section we present the basic concepts of automata theory and illustrate them by giving some simple examples. We show how to invert special types of automata and extend the transition and output function.

The third section is concentrated on the automaton group. After its formal introduction we look at a simple example of an automaton group. Then we exhibit the connection between the automaton group and the free group generated by the states of the automaton.

## 2.1 Group actions and their visualization via Schreier graphs

In this chapter we will encounter situations where we need edges with a direction. Thus we briefly introduce directed graphs.

**Definition 2.1.** A directed graph $G = (V, E)$ consists of a *set of vertices* $V$, a *set of edges* $E$ and an *incidence relation.* The only difference to undirected graphs is that for directed graphs the incidence relation pairs each edge with an *ordered pair of vertices.* As for undirected graphs, if $e$ is incident to $(u, v) \in V \times V$ we usually write $e = (u, v)$, although there may be more edges with this relation. Other basic concepts of undirected graphs are carried over as well.

**Definition 2.2.** An alphabet is a finite non-empty set $\Sigma$ consisting of elements called symbols or letters.

$$\Sigma^* := \{(w_1, \dots, w_n) \mid n \in \mathbb{N}, w_1, \dots, w_n \in \Sigma\} \cup \{\epsilon\} \tag{2.1}$$

is the set of finite words over the alphabet $\Sigma$, where $\epsilon := ()$ is the empty word. For $w = (w_1, \dots, w_n) \in \Sigma^*$ we simply write $w = w_1 \dots w_n$. A word consisting of $n$ times the letter $s$ is abbreviated by $s^n$. The length of $w$ is the number of symbols used for the word and denoted by $|w| = n$. For $n \in \mathbb{N}_0$ the subset $\Sigma^n := \{w \in \Sigma^* \mid |w| = n\}$ is the set of words of length $n$.

Let $\cdot$ be the concatenation of words, i.e. $v \cdot w = vw := v_1 \dots v_n w_1 \dots w_m$ for $v = v_1 \dots v_n \,, w = w_1 \dots w_m \in \Sigma^*$. Then $(\Sigma^*, \cdot)$ is a monoid with $\epsilon$ as identity element. We call it the free monoid on $\Sigma$.

In order to create a group based on the free monoid we need to introduce some inverses and conceptualize cancelling. This is done in the next definition.

**Definition 2.3.** Let $S$ be a non-empty set and $M := \left(S \cup S^{-1}\right)^*$, where $S^{-1} := \{s^{-1} \mid s \in S\}$ is the *set of inverse symbols* of $S$ and $S \cap S^{-1} = \varnothing$. We call a word $w \in M$, with $w = vv^{-1}$ for some $v \in M$ *skew-symmetric*. The word $w$ is called *non-reduced* if it contains a non-empty skew-symmetric subword, i.e. there are $v, x, y \in M$ with $v \neq \epsilon$ and $w = xvv^{-1}y$, otherwise it is called *reduced*.

For every word in $M$ we end up with exactly one reduced word, when we successively remove any non-empty skew-symmetric subword. Thus we can describe this process with a function $\rho : M \to M$ and call it *reduction*. Using this preparatory work it is possible to define the *free group* generated by $S$, namely

$$\langle S \rangle := (\rho(M), \rho \circ \cdot), \tag{2.2}$$

where $\cdot$ is the concatenation of words and $\circ$ the composition of functions.

**Remark 2.4.** $\langle S \rangle$ can also be defined via the equivalence relation on $M$ given by the pre-images of the reduction $\rho$. Then $\langle S \rangle$ consists of the equivalence classes and the operation is the concatenation of classes.

$|\cdot|$ may be restricted to $\rho(M)$, so every element of $\langle S \rangle$ has a unique length. All words $v, w \in M$ satisfy $|v \cdot w| = |v| + |w|$ and $|\rho(v \cdot w)| \leq |v| + |w|$, thus $|\cdot|$ is additive on $M$ but only sub-additive on $\langle S \rangle$.

**Example 2.5.** To illustrate the definition above we take as an example $S := \{a, b\}$. Then $S^{-1} := \{a^{-1}, b^{-1}\}$ and $M$ is the set of finite words over the alphabet $\{a, b, a^{-1}, b^{-1}\}$. The word $w := a^{-1}a\,b\,b^{-1}$ is not skew-symmetric, but the subwords $w_1 = a^{-1}a$ and $w_2 = b\,b^{-1}$ are skew-symmetric so $w$ is non-reduced. The reduction $\rho$ applied to $w$ may remove $w_1$ first and then $w_2$ or vice versa, but the result is always the empty word $\epsilon$. Thus, with respect to the equivalence relation given by the reduction, the words $w$, $w_1$ and $w_2$ are all equivalent.

As a second word we take $x := a\,a^{-1}b\,b^{-1}a$. It has as non-empty skew-symmetric subwords $x_1 = a\,a^{-1}$, $x_2 = b\,b^{-1}$ and $x_3 = a^{-1}b\,b^{-1}a$. In contrast to before the subword $x_3$ intersects both other subwords. So after removing $x_3$ from $x$ the reduced word $a$ remains, whereas if we remove first $x_1$ or $x_2$ , then we have to remove $x_2$ or $x_1$ afterwards to get the reduced word $a$.

The group operation $\rho \circ \cdot$ applied to the reduced words $a$ and $a^{-1}b$ has the form $\rho(a \cdot a^{-1}b) = b$. This shows that $|\cdot|$ is not additive on $\rho(M)$. Furthermore the group is clearly non-commutative as $\rho(a^{-1}b \cdot a) = a^{-1}b\,a$.

If $(G, *)$ is a group and $S$ a subset of $G$ we would like $\langle S\rangle$ to be a subgroup of $(G, *)$, i.e. $*$ is the group operation of $\langle S\rangle$ and $\rho(M)$ a subset of $G$. This is easy to achieve, but in order to prevent overloading the $\langle\rangle$ notation, we add the group as index.

**Definition 2.6.** Let $(G, *)$ be a group and $S$ a subset of $G$. We set $M = (S \cup S^{-1})^*$ and define the *group reduction* to $G$ as

$$\begin{array}{rccc} \rho_G : & M & \to & G \\ & m_1 \dots m_n & \mapsto & m_1 * \dots * m_n \ . \end{array} \tag{2.3}$$

Then $\langle S\rangle_G := (\rho_G(M), *)$ is the subgroup of $G$ generated by $S$.

We say that $S$ is a *generating set* of $G$ if $\langle S\rangle_G = G$ and the group is *finitely generated* if there exists a finite generating set of $G$.

If the group $G$ is isomorphic to a free group generated by some $B \subset G$ with $B \cap B^{-1} = \varnothing$, written $G \cong \langle B\rangle$, then $G$ is called *free* and $B$ is a *basis* of $G$. In this setting every basis has the same cardinality, called the *rank* of the group.

**Remark 2.7.** Let $S \subset G$ be a generating set of the group $(G, *)$. Every group element can be represented in terms of elements of $S$ and their inverses. If additionally $S$ is a basis, then such a representation is unique up to skew-symmetric subwords.

**Definition 2.8.** Let $(G, *)$ be a group and $M$ a set. A *(left) group action* of $G$ on $M$ is a map $G \times M \to M$, $(g, m) \mapsto g \cdot m$ such that for all $g_1, g_2 \in G$ and all $m \in M$ the following holds:

(i) $\mathbb{1}_G \cdot m = m$,

(ii) $g_1 \cdot (g_2 \cdot m) = (g_1 * g_2) \cdot m$

where $\mathbb{1}_G$ is the identity element of the group $G$.

$G \cdot m := \{g \cdot m \mid g \in G\}$ is the *orbit* and $\mathrm{Stab}_G(m) := \{g \in G \mid g \cdot m = m\}$ the *stabilizer* of $m \in M$. If $G \cdot m = M$ for some $m \in M$, the action is called *transitive.* The action is denoted as *free*, if $\mathrm{Stab}_G(m) = \{\mathbb{1}_G\}$ for all $m \in M$.

**Remark 2.9.** Since the orbits are closed under the action, $G$ also acts on any orbit. If the action is transitive, then $G \cdot m = M$ holds for all $m \in M$.

Suppose that $S \subset G$ is a basis of $G$. Then any function from $S \times M$ to $M$ satisfying (i) and (ii) from Definition 2.8, i.e. $(s_1 * s_1^{-1}) \cdot m = m$ and $s_1 \cdot (s_2 \cdot m) = (s_1 * s_2) \cdot m$ for all $s_1, s_2 \in S$ and $m \in M$, has a unique extension to a group action of $G$ on $M$.

**Definition 2.10.** Let $(G, *)$ be a group generated by $S$ and suppose $G$ acts on a set $M$. We define the labelled graph $\Gamma(G, S, M)$, called *Schreier graph*, to visualize the action of $G$ on $M$ with respect to the generating set $S$. The vertex set of $\Gamma(G, S, M)$ is $M$, the set of labels is $S$ and for any $m \in M$ and $s \in S$ there is an edge $\{m, s \cdot m\}$ labelled by $s$. If the action is transitive, the Schreier graph is connected. In case the action is not transitive, we get one connected component for each orbit. We call these components *orbital Schreier graphs* and they are of the form $\Gamma(G, S, G \cdot m)$ for $m \in M$.

Now suppose $M = \Sigma^*$ for a non-empty set $\Sigma$ and each orbit has only words of the same length, i.e. for all $m \in M$ there is an $n \in \mathbb{N}_0$ such that $G \cdot m \subset \Sigma^n$. Then we can define the *n-th Schreier graph* $\Gamma(G, S, \Sigma^n)$. Note that all $n$-th Schreier graphs are connected if and only if $G$ acts *spherically transitively*, that means the action on words of a fixed length is transitive. In this case every $n$-th Schreier graph is also an orbital Schreier graph.

**Remark 2.11.** For every generator $s \in S$ and every vertex $m \in M$ there are either the two edges $\{m, s \cdot m\}$ and $\{s^{-1} \cdot m, m\}$ or there is a loop $\{m, m\}$. Hence a Schreier graph $\Gamma(G, S, M)$ is a $2|S|$-regular graph with $|S| \cdot |M|$ edges that may be parallel edges or loops.

Usually it is meaningless to include the identity element $\mathbb{1}_G$ in $S$ since it just produces a loop on every vertex. Other elements of the group acting like $\mathbb{1}_G$ on $M$ should be excluded from the generating set as well if possible.

Sometimes Schreier graphs are defined as directed graphs, where the edges $\{m, s \cdot m\}$ are replaced by edges $(m, s \cdot m)$. Then $S$ is additionally required to be symmetric, i.e. $S = S^{-1}$. This concept is, with some minor exceptions, analogous to the above definition.

Schreier graphs are classical objects of study in group theory. This fact becomes more evident under the considerations below.

**Remark 2.12.** Let the action of $G$ on $M$ be transitive and $S$ a generating set of $G$. For $m \in M$ the stabilizer $\operatorname{Stab}_G(m)$ is a subgroup of $G$. Thus we can build the left cosets $G/\operatorname{Stab}_G(m) := \{g * \operatorname{Stab}_G(m) \mid g \in G\}$ and observe that $G$ acts on these with $h \cdot (g * \operatorname{Stab}_G(m)) := (h * g) * \operatorname{Stab}_G(m)$ for $h \in G$ and $g * \operatorname{Stab}_G(m) \in G/\operatorname{Stab}_G(m)$. Furthermore the map

$$\begin{aligned} G/\operatorname{Stab}_G(m) &\to M \\ g * \operatorname{Stab}_G(m) &\mapsto g \cdot m \end{aligned} \tag{2.4}$$

is a bijection that is compatible with the action of $G$ on $G/\operatorname{Stab}_G(m)$ and $M$. Therefore the Schreier graphs $\Gamma(G, S, M)$ and $\Gamma(G, S, G/\operatorname{Stab}_G(m))$ are label-isomorphic.

## 2.2 Automata basics

**Definition 2.13.** A *(finite) automaton* is a quadruple $\mathcal{A} = (S, \Sigma, \alpha, \beta)$ where

(i) $S$ is a non-empty finite set, called *the set of states*,

(ii) $\Sigma$ is a non-empty finite set, called *the set of input and output symbols* or *input/output alphabet*,

(iii) $\alpha : S \times \Sigma \to S$ is a map, called *the transition function*,

(iv) $\beta : S \times \Sigma \to \Sigma$ is a map, called *the output function*.

The automaton can be represented by a directed labelled graph $SD(\mathcal{A}) := (S, E)$, named *(Mealy) state diagram*. The vertices are given by the states $S$ and for every state $p \in S$ and symbol $s \in \Sigma$ there is an edge $(p, \alpha(p, s))$ labelled by $s|\beta(p, s)$. Such an edge corresponds to the transition starting at the state $p$ given the input symbol $s$.

Sometimes it is of interest to get to a state via an application of $\alpha$. We say that a state $p$ is *reachable* if it is included in the image of $\alpha$, i.e. $p \in \alpha(S, \Sigma)$, otherwise $p$ is called *unreachable*.

**Example 2.14.** The first easy example we look at is the incrementation automaton $\mathcal{A}_I = (\{a, id\}, \{0, 1\}, \alpha, \beta)$, also called adding machine, where

$$\begin{array}{ll} \alpha(a,0) := id, & \alpha(id,s) := id, \\ \alpha(a,1) := a, & \beta(a,s) := 1-s, \\ \text{and for } s \in \{0,1\} & \beta(id,s) := s. \end{array} \tag{2.5}$$

We can see the state diagram $SD(\mathcal{A}_I)$ of the incrementation automaton in Figure 2.1.

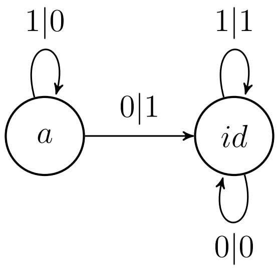


Figure 2.1: State diagram of the incrementation automaton.

For a single input this automaton is not very interesting. Thus, we think about a way to extend $\alpha$ and $\beta$ to $S \times \Sigma^*$. Suppose we have the input sequence 101 and start in the state $a$. The first output symbol is 0 and we stay in state $a$. Then the next output symbol is 1 and we move to state $id$. Finally we get the output 1 and the whole input sequence is processed. The complete output is 011, so, when treating input and output as binary numbers written in least significant bit order, the transition starting in state $a$ increases the input by 1. A more thorough inspection yields that in case $k \neq 2^n - 1$, the transition starting at state $a$ with the input number $k$ represented by $n$ bits gives the output number $k+1$ represented by $n$ bits as well. If the input is $k = 2^n - 1$, then the result is 0.

This example delivers an idea of how to extend $\alpha$ and $\beta$. We do this in a rigorous way and show, that everything works as intended.

**Definition 2.15.** Let $\mathcal{A} = (S, \Sigma, \alpha, \beta)$ be an automaton. We define the extensions $\alpha : S \times \Sigma^* \to S$ and $\beta : S \times \Sigma^* \to \Sigma^*$ recursively. Let $p \in S$, $s \in \Sigma$ and $w \in \Sigma^*$, then

$$\begin{aligned} \alpha(p, sw) &:= \alpha(\alpha(p,s), w) \quad \text{and} \\ \beta(p, sw) &:= \beta(p,s)\beta(\alpha(p,s), w). \end{aligned} \tag{2.6}$$

Furthermore, applications on the empty word are treated in the following way:

$$\begin{aligned} \alpha(p, \epsilon) &:= p, \\ \beta(p, \epsilon) &:= \epsilon. \end{aligned} \tag{2.7}$$

**Remark 2.16.** By using (2.6) recursively every function application of $\alpha$ and $\beta$ can be written in terms of applications of $\alpha$ and $\beta$ on $S \times \Sigma \cup \{\epsilon\}$. Thus, it is easy to see that the extensions are well defined by (2.6) and (2.7).

Interestingly enough it can be shown that $\beta(p, \epsilon) = \epsilon$ follows from (2.6) if every state is reachable. Moreover it is not difficult to prove that everything is well defined if we take a word for $s$ in (2.6) instead of single symbol. Therefore we can split the word in the second argument in any position and it is not mandatory to do it after the first symbol. Then $\beta(p, \epsilon) = \epsilon$ follows also from (2.6), but still not the equation $\alpha(p, \epsilon) = p$, even if every state is reachable.

Since we extended $\alpha$ and $\beta$ in the second argument it seems natural to try the same in the first argument. It will turn out that this works almost as before, but we can extend $\alpha$ and $\beta$ to even more than $S^* \times \Sigma^*$ under certain conditions.

**Definition 2.17.** An automaton $\mathcal{A} = (S, \Sigma, \alpha, \beta)$ is called *invertible* if for all $p \in S$ the output function of the transition starting at the state $p$, namely $\beta_p := \beta(p, \cdot) : \Sigma \to \Sigma$ , is a bijection. In this case we can extend $\beta$ to $\left(S \cup S^{-1}\right) \times \Sigma^*$ via the inversion, i.e.

$$\beta(p^{-1}, \cdot) := (\beta_p)^{-1}. \tag{2.8}$$

Furthermore we define the extension of $\alpha$ to $\left(S \cup S^{-1}\right) \times \Sigma^*$ by

$$\alpha(p^{-1}, \cdot) := \left(\alpha(p, \beta(p^{-1}, \cdot))\right)^{-1} \tag{2.9}$$

while expanding the range to $S \cup S^{-1}$.

With these tools the *inverse automaton* of $\mathcal{A}$ is defined by simply changing the set of states to the set of inverted states, so $\mathcal{A}^{-1} := (S^{-1}, \Sigma, \alpha, \beta)$.

**Remark 2.18.** Let $\mathcal{A} = (S, \Sigma, \alpha, \beta)$ be an invertible automaton. Suppose $\alpha(p, s) = q$ and $\beta(p, s) = t$, then

$$\begin{aligned} \beta(p^{-1}, t) &= s \quad \text{and} \\ \alpha(p^{-1}, t) &= (\alpha(p, \beta(p^{-1}, t)))^{-1} = (\alpha(p, s))^{-1} = q^{-1}. \end{aligned} \tag{2.10}$$

Hence we obtain the state diagram $SD(\mathcal{A}^{-1})$ from the state diagram $SD(\mathcal{A})$ by replacing each vertex $p$ with $p^{-1}$ and changing each label $s|t$ to $t|s$. Consequently $SD(\mathcal{A}^{-1})$ and $SD(\mathcal{A})$ are isomorphic, but usually not label-isomorphic.

**Example 2.19.** Again we take a look at $\mathcal{A}_I = (\{a, id\}, \{0, 1\}, \alpha, \beta)$, the incrementation automaton, which we have already seen in Example 2.14. Clearly $\mathcal{A}_I$ is invertible.

Applying the insights of the remark above, it is very easy to derive the state diagram $SD(\mathcal{A}_I^{-1})$ from $SD(\mathcal{A}_I)$. It can be seen in Figure 2.2 and is obviously rather similar to $SD(\mathcal{A}_I)$ (compare with Figure 2.1).

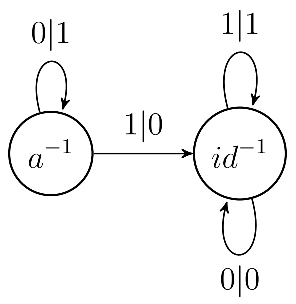


Figure 2.2: State diagram of the inverse incrementation automaton.

In Example 2.14 we started a transition in state $a$ with input 101 and got the output 011. If we take 011 as input and start a transition in state $a^{-1}$ the output is 101. It is not immediately evident by the definition, but it could be expected that starting in the inverse state gives the initial input. In fact for the input number $k$ represented by $n$ bits the transition starting in state $a^{-1}$ produces the output number $(k-1) \mod 2^n$ represented by $n$ bits. Consequently it seems natural to call $\mathcal{A}_I^{-1}$ the decrementation automaton.

Initially we treated $\beta_p$ as a function from $\Sigma$ to $\Sigma$. Since we extended $\beta$ to work with words, it makes sense to use this extension for $\beta_p$ as well.

**Proposition 2.20.** *Let $\mathcal{A} = (S, \Sigma, \alpha, \beta)$ be an invertible automaton. For every state $p \in S \cup S^{-1}$ the extended output function starting at $p$, namely*

$\beta_p = \beta(p, \cdot) : \Sigma^* \to \Sigma^*$, *is bijective and*

$$\beta_p \circ \beta_{p^{-1}} = \beta_{p^{-1}} \circ \beta_p = id_{\Sigma^*}, \tag{2.11}$$

*where* $id_{\Sigma^*}$ *is the identity map on* $\Sigma^*$*.*

*Proof.* Let $p$ be a state in $S \cup S^{-1}$. If (2.11) holds, then $\beta_p$ is a bijection, hence all we need to show is that $\beta_p(\beta_{p^{-1}}(w)) = \beta_{p^{-1}}(\beta_p(w)) = w$ applies for all words $w \in \Sigma^*$. This is done by induction over the length of $w$. Since $\mathcal{A}$ is invertible the claim is clearly true if $|w| = 1$.

Now let $w = w_0 \dots w_n$ and suppose (2.11) holds for all words of length $n$ or less. We set $w' = w_1 \dots w_n$ and use (2.6) to obtain

$$\beta_p(w) = \beta_p(w_0)\beta_{\alpha(p,w_0)}(w'). \tag{2.12}$$

Applying (2.6) again we get

$$\beta_{p^{-1}}(\beta_p(w)) = \beta_{p^{-1}}(\beta_p(w_0))\beta_{\alpha(p^{-1},\beta_p(w_0))}(\beta_{\alpha(p,w_0)}(w')). \tag{2.13}$$

Definition 2.17 and the induction hypothesis show that

$$\alpha(p^{-1}, \beta_p(w_0)) = (\alpha(p, \beta_{p^{-1}}(\beta_p(w_0))))^{-1} = (\alpha(p, w_0))^{-1}. \tag{2.14}$$

Thus, by setting $q := \alpha(p, w_0)$ the right hand side of (2.13) can be simplified to

$$\beta_{p^{-1}}(\beta_p(w_0))\beta_{q^{-1}}(\beta_q(w')). \tag{2.15}$$

Using the induction hypothesis twice we derive

$$\beta_{p^{-1}}(\beta_p(w_0))\beta_{q^{-1}}(\beta_q(w')) = w_0 w'. \tag{2.16}$$

When we exchange the order of $\beta_p$ and $\beta_{p^{-1}}$ we just need to exchange $p$ and $p^{-1}$ in (2.13). Since

$$\alpha(p^{-1}, w_0) = (\alpha(p, \beta_{p^{-1}}(w_0))^{-1} \tag{2.17}$$

we set $q := \alpha(p^{-1}, w_0)$ and all other steps work as before. This finishes the induction and hence (2.11) holds. □

Now we have gathered all tools necessary to define the full extensions of the transition function $\alpha$ and the output function $\beta$.

**Definition 2.21.** Let $\mathcal{A} = (S, \Sigma, \alpha, \beta)$ be an invertible automaton. We define the extensions $\alpha : \langle S\rangle \times \Sigma^* \to \langle S\rangle$ and $\beta : \langle S\rangle \times \Sigma^* \to \Sigma^*$ recursively in the first component. Let $p \in S$, $\bar{q} \in \langle S\rangle$ and $w \in \Sigma^*$, then

$$\begin{aligned} \alpha(\bar{q}p, w) &:= \alpha(\bar{q}, \beta(p, w))\alpha(p, w) \text{ and} \\ \beta(\bar{q}p, w) &:= \beta(\bar{q}, \beta(p, w)). \end{aligned} \tag{2.18}$$

**Remark 2.22.** For $p \in \langle S \rangle$ and $w \in \Sigma^*$ the function executions $\alpha(p, w)$ and $\beta(p, w)$ are determined by first splitting the expressions in the first argument until $\alpha$ and $\beta$ are only executed on states in $S \cup S^{-1}$ and then splitting the words in the second argument.

Let $\mathbb{1}_{\langle S \rangle}$ be the identity element of $\langle S \rangle$ and $w \in \Sigma^*$, then

$$\beta(\mathbb{1}_{\langle S \rangle}, w) = \beta(pp^{-1}, w) = \beta_p(\beta_{p^{-1}}(w)) = w. \tag{2.19}$$

Furthermore

$$\alpha(\mathbb{1}_{\langle S \rangle}, w) = \alpha(\mathbb{1}_{\langle S \rangle}\mathbb{1}_{\langle S \rangle}, w) = \alpha(\mathbb{1}_{\langle S \rangle}, \beta(\mathbb{1}_{\langle S \rangle}, w))\alpha(\mathbb{1}_{\langle S \rangle}, w) \tag{2.20}$$

implies

$$\alpha(\mathbb{1}_{\langle S \rangle}, w) = \mathbb{1}_{\langle S \rangle}. \tag{2.21}$$

At this point we have everything we need to handle $\alpha$ and $\beta$. But for the sake of completeness we show that the extension rules (2.6) and (2.18) can be used more generally.

**Proposition 2.23.** *Let $\mathcal{A} = (S, \Sigma, \alpha, \beta)$ be an invertible automaton. For $p, q \in \langle S \rangle$ and $v, w \in \Sigma^*$ the following equalities hold:*

$$\alpha(p, vw) = \alpha(\alpha(p, v), w), \tag{2.22}$$
$$\alpha(qp, w) = \alpha(q, \beta(p, w))\alpha(p, w), \tag{2.23}$$
$$\beta(p, vw) = \beta(p, v)\beta(\alpha(p, v), w), \tag{2.24}$$
$$\beta(qp, w) = \beta(q, \beta(p, w)). \tag{2.25}$$

*Proof.* We show the equalities for $\beta$ only. The proofs of (2.22) and (2.23) are analogous to the following proofs of (2.25) and (2.24), respectively.

First we show (2.25) by induction over the length of $p \in \langle S \rangle$. If $|p| = 0$, then $p = \mathbb{1}_{\langle S \rangle}$ and thus (2.25) holds since $\beta(\mathbb{1}_{\langle S \rangle}, w) = w$.

Now let $p = p_0 \dots p_n$ and suppose (2.25) holds for all elements of length $n$ or less. By multiple applications of the induction hypothesis we get for $p' := p_1 \dots p_n$ :

$$\begin{aligned} \beta(qp, w) &= \beta(qp_0, \beta(p', w)) \\ &= \beta(q, \beta(p_0, \beta(p', w))) \\ &= \beta(q, \beta(p_0 p', w)). \end{aligned} \tag{2.26}$$

Thus, the induction is complete and (2.25) holds.

Next we use induction over the length of $v$ to show (2.24) for $p \in S \cup S^{-1}$. $|v| = 0$ is equivalent to $v = \epsilon$. We know that $\alpha(p, \epsilon) = p$ and $\beta(p, \epsilon) = \epsilon$ if $p \in S \cup S^{-1}$, so (2.24) holds in this case.

As before we take $v = v_0 \dots v_n$, suppose (2.24) holds for words of length $n$ or less if $p \in S \cup S^{-1}$ and set $v' := v_1 \dots v_n$. Then making use of the induction hypothesis and the definitions of the extensions of $\alpha$ and $\beta$ we obtain:

$$\begin{aligned} \beta(p, vw) &= \beta(p, v_0)\beta(\alpha(p, v_0), v'w) \\ &= \beta(p, v_0)\beta(\alpha(p, v_0), v')\beta(\alpha(\alpha(p, v_0), v')), w) \\ &= \beta(p, v_0v')\beta(\alpha(p, v_0v'), w). \end{aligned} \tag{2.27}$$

Hence for $p \in S \cup S^{-1}$ (2.24) is true.

To finish the proof we show that the result stays the same if we exchange the order of application of (2.24) and (2.25). We consider $\beta(qp, vw)$, where $p, q \in \langle S\rangle$, $v, w \in \Sigma^*$, and begin with (2.24) first:

$$\begin{aligned} \beta(qp, vw) &\overset{(2.24)}{=} \beta(qp, v)\beta(\alpha(qp, v), w) \\ &\overset{(2.25)}{=} \beta(q, \beta(p, v))\beta(\alpha(qp, v), w) \\ &\overset{(2.23)}{=} \beta(q, \beta(p, v))\beta(\alpha(q, \beta(p, v))\alpha(p, v), w) \\ &\overset{(2.25)}{=} \beta(q, \beta(p, v))\beta(\alpha(q, \beta(p, v)), \beta(\alpha(p, v), w)). \end{aligned} \tag{2.28}$$

This time we use (2.25) first:

$$\begin{aligned} \beta(qp, vw) &\overset{(2.25)}{=} \beta(q, \beta(p, vw)) \\ &\overset{(2.24)}{=} \beta(q, \beta(p, v)\beta(\alpha(p, v), w)) \\ &\overset{(2.24)}{=} \beta(q, \beta(p, v))\beta(\alpha(q, \beta(p, v)), \beta(\alpha(p, v), w)). \end{aligned} \tag{2.29}$$

In the first reformulation process the order of step two and three can be exchanged not altering the result. All other steps following the first are deterministic, so the result does not depend on different orders of application of (2.24) and (2.25). Thus (2.24) can also be used if $p \in \langle S\rangle$. □

The main purpose of this lengthy examination of $\alpha$ and $\beta$ was to thoroughly comprehend the action of an automaton. It is easy to see, that already Definition 2.21 and Remark 2.22 provide everything required to guarantee that $\beta$ is a group action of $\langle S\rangle$ on $\Sigma^*$. Unfortunately, the size of $\langle S\rangle$ can be fairly large in comparison to the possible outcomes of the action applied on a single element. This is shown in the following example.

**Example 2.24.** Let $\mathcal{A}_{FBS} = (\{a, id\}, \{0, 1\}, \alpha, \beta)$ be the first-bit-switch automaton, where

$$\begin{aligned} &\alpha(\cdot, \cdot) := id, && \beta(a, s) := 1 - s, \\ &\text{and for } s \in \{0, 1\} && \beta(id, s) := s. \end{aligned} \tag{2.30}$$

The state diagram $SD(\mathcal{A}_{FBS})$ of the first-bit-switch automaton is drawn in Figure 2.3.

Clearly $\mathcal{A}_{FBS}$ is invertible and the state diagram of the inverse first-bit-switch automaton is label-isomorphic to $SD(\mathcal{A}_{FBS})$. Thus for all $w \in \{0, 1\}^*$,

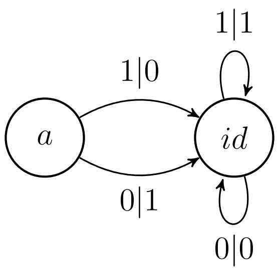


Figure 2.3: State diagram of the first-bit-switch automaton.

$\beta(a, w) = \beta(a^{-1}, w)$ and furthermore $\beta(aa, w) = w$ hold. Using these facts it is not difficult to show that if $w \neq \epsilon$, the orbit $\langle S\rangle \cdot w$ consists of $w$ and another word of the same length where only the first letter differs to $w$. So the action applied to a single element has only two possible outcomes, whereas $\langle \{a, id\}\rangle$ is a free group of rank 2 and thus infinite.

## 2.3 The automaton group

To visualize the action of $\langle S\rangle$ on $\Sigma^*$ via some Schreier graph, we need a suitable generating set. Obviously the smallest generating set of $\langle S\rangle$ is $S$. But as we have seen in Example 2.24 there may be states, e.g. $id$, that act on $\Sigma^*$ like the identity element $\mathbb{1}_{\langle S\rangle}$. This is something we want to avoid in a generating set, as well as different states providing the same action on $\Sigma^*$. For example if we had a state $b$ in Example 2.24 with $\alpha(b, \cdot) := \alpha(a, \cdot)$ and $\beta(b, \cdot) := \beta(a, \cdot)$, then nothing would change for the action, but we would end up with many redundant parallel edges in the resulting Schreier graph. Thus, we construct another group related to the automaton. We start by studying $\Sigma^*$ some more.

**Definition 2.25.** Let $T = (V, E)$ be a tree. We say $T$ is *rooted* if we have a given *root vertex* $r \in V$. Then we classify all other vertices by *levels*. A vertex $v \in V$ is in the *$n$-th level* if the distance between $v$ and $r$ is $n \in \mathbb{N}_0$.

A rooted tree is called *$d$-regular* if the degree of the root is $d \in \mathbb{N}$ and the degree of all other vertices is $d + 1$.

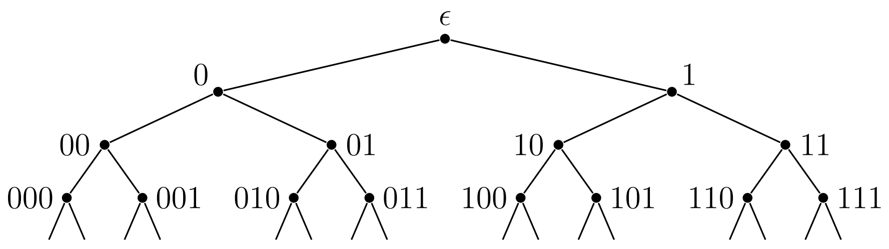


Figure 2.4: 2-regular rooted tree associated to $\Sigma^*$ for $\Sigma = \{0, 1\}$.

**Remark 2.26.** Let $\Sigma$ be a non-empty finite set of symbols. We can view $\Sigma^*$ as rooted tree, where $\Sigma^*$ is the vertex set, $\epsilon$ the root and two vertices are connected by an edge if and only if they are of the form $w$ and $ws$ for some $w \in \Sigma^*$ and $s \in \Sigma$. In particular this is a $|\Sigma|$-regular tree, where the $n$-th level consists of $|\Sigma|^n$ words of length $n$. An exemplary illustration for $\Sigma = \{0, 1\}$ can be seen in Figure 2.4.

**Definition 2.27.** Let $\Sigma$ be a non-empty finite set of symbols. A map from $\Sigma^*$ to $\Sigma^*$ is called *automorphism* if it is an automorphism on the rooted tree given by $\Sigma^*$. We define $\mathrm{Aut}(\Sigma^*)$ to be the *set of all automorphisms* on the rooted tree given by $\Sigma^*$.

**Remark 2.28.** It is easy to see that $\mathrm{Aut}(\Sigma^*)$ paired with the composition of functions forms a group that acts on $\Sigma^*$. Furthermore, any automorphism $\varphi \in \mathrm{Aut}(\Sigma^*)$ has to map the empty word to the empty word since it is the only vertex with degree $|\Sigma|$. Thus, for every $n \in \mathbb{N}_0$ , $\varphi$ is also a bijection from the $n$-th level to itself.

**Remark 2.29.** In Proposition 2.20 we proved for an invertible automaton $\mathcal{A} = (S, \Sigma, \alpha, \beta)$ that for every $p \in S$ the map $\beta_p$ is a bijection from $\Sigma^*$ to $\Sigma^*$. Let $w \in \Sigma^*$, $a \in \Sigma$ and $p \in S$, then $\beta_p(wa) = \beta_p(w)\beta_{\alpha(p,w)}(a)$. So the words $\beta_p(w)$ and $\beta_p(wa)$ are connected with an edge. Hence $\beta_p$ is an element of $\mathrm{Aut}(\Sigma^*)$.

**Definition 2.30.** Let $\mathcal{A} = (S, \Sigma, \alpha, \beta)$ be an invertible automaton. The *automaton group* of $\mathcal{A}$ is defined as the subgroup of $\mathrm{Aut}(\Sigma^*)$ generated by all automorphisms $\beta_p$ , namely

$$G(\mathcal{A}) := \langle \{\beta_p \mid p \in S\} \rangle_{\mathrm{Aut}(\Sigma^*)} \,. \tag{2.31}$$

**Example 2.31.** We want to determine the automaton group of the first-bit-switch automaton $\mathcal{A}_{FBS} = (\{a, id\}, \{0, 1\}, \alpha, \beta)$ introduced in Example 2.24. We showed for $w \in \Sigma^*$ that $\beta_a(\beta_a(w)) = w$, and $\beta_{id}(w) = w$ holds by definition. Thus $G(\mathcal{A}_{FBS}) = \{\beta_{id} , \beta_a\}$, i.e. the automaton group of $\mathcal{A}_{FBS}$ has only 2 elements and is therefore isomorphic to $\mathbb{Z}/2\mathbb{Z}$.

In comparison to $\langle S \rangle$ the automaton group can be much smaller but it still produces the same action on $\Sigma^*$. Therefore it makes sense to utilize it to visualize the action of the automaton. Since the action cannot change the length of words, the complete Schreier graph is always infinite and disconnected. Considering this fact we only look at words of a certain length.

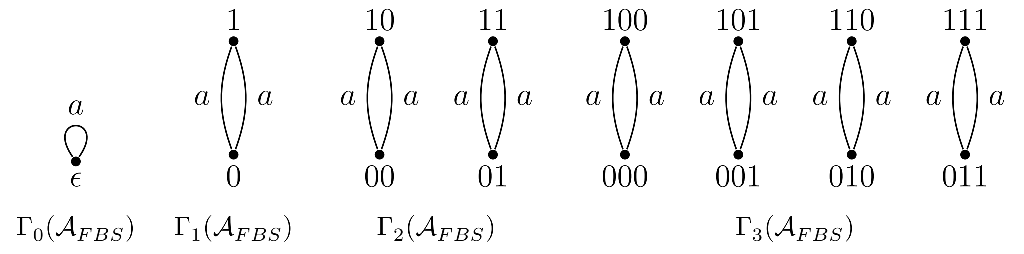
**Definition 2.32.** Let $\mathcal{A} = (S, \Sigma, \alpha, \beta)$ be an invertible automaton. For $n \in \mathbb{N}_0$ we define the *n-th Schreier graph* of $\mathcal{A}$ as the $n$-th Schreier graph of the action of $G(\mathcal{A})$ on $\Sigma^*$, where the corresponding generating set of $G$ is $\{\beta_p \mid p \in S\}$

without the identity element of $\mathrm{Aut}(\Sigma^*)$, written

$$\Gamma_n(\mathcal{A}) := \Gamma(G(\mathcal{A}), \{\beta_p \mid p \in S\} \setminus \{\mathbb{1}_{\mathrm{Aut}(\Sigma^*)}\}, \Sigma^n). \tag{2.32}$$

To simplify the notation we replace each label $\beta_p$ by the state $p$ itself.

**Example 2.33.** We take a look at the first four Schreier graphs of the first-bit-switch automaton $\mathcal{A}_{FBS} = (\{a, id\}, \{0,1\}, \alpha, \beta)$ drawn in Figure 2.5.



$a$
$\epsilon$
$\Gamma_0(\mathcal{A}_{FBS})$
1
0
$\Gamma_1(\mathcal{A}_{FBS})$
10
00
11
01
$\Gamma_2(\mathcal{A}_{FBS})$
100
000
101
001
110
010
111
011
$\Gamma_3(\mathcal{A}_{FBS})$

Figure 2.5: Schreier graphs of the first-bit-switch automaton.

As for any automaton, the zeroth Schreier graph consists of the vertex $\epsilon$ and a loop for every element in the generating set. Like the first Schreier graph it is connected, whereas all other Schreier graphs of $\mathcal{A}_{FBS}$ are disconnected. This stems from the fact that the first-bit-switch automaton can only change the first letter of a word.

We have already seen that every orbit of the action of $\langle\{a, id\}\rangle$ on $\{0,1\}^*$ has exactly two elements, except for the one consisting of only the empty word. The same holds true for the action of $G(\mathcal{A}_{\mathcal{FBS}})$ on $\{0,1\}^*$. Thus, the $n-th$ Schreier graph of $\mathcal{A}_{FBS}$ consists of $2^{n-1}$ pairs of vertices that are connected by double edges labelled $a$.

The actions of the automaton group and $\langle S\rangle$ on $\Sigma^*$ are alike, so it seems natural that the groups are related. We examine their connection in the remainder of this chapter.

**Definition 2.34.** Let $G$ be a group acting on a set $M$. The *pointwise stabilizer* of $M$ in $G$ is defined as the intersection of all stabilizers of elements in $M$, i.e.

$$\mathrm{Stab}_G(M) := \bigcap_{m \in M} \mathrm{Stab}_G(m). \tag{2.33}$$

**Remark 2.35.** Let $a, b \in \mathrm{Stab}_G(M)$ and $m \in M$. Then we get

$$(a * b^{-1}) \cdot m = (a * b^{-1}) \cdot (b \cdot m) = (a * b^{-1} * b) \cdot m = a \cdot m = m. \tag{2.34}$$

Thus, $a * b^{-1} \in \mathrm{Stab}_G(M)$ and consequently $\mathrm{Stab}_G(M)$ is a subgroup of $G$.

For $a \in \mathrm{Stab}_G(M)$, $g \in G$ and $m \in M$ it holds that

$$(g * a * g^{-1}) \cdot m = g \cdot (a \cdot (g^{-1} \cdot m)) = g \cdot (g^{-1} \cdot m) = m. \tag{2.35}$$

This shows that $\mathrm{Stab}_G(M)$ is closed under conjugation and as a result a normal subgroup of $G$.

**Theorem 2.36.** *Let $\mathcal{A} = (S, \Sigma, \alpha, \beta)$ be an invertible automaton. $\langle S\rangle / \mathrm{Stab}_{\langle S\rangle}(\Sigma^*)$ is a group able to inherit the group action of $\langle S\rangle$ on $\Sigma^*$, i.e. $p\,\mathrm{Stab}_{\langle S\rangle}(\Sigma^*) \cdot w = \beta(p, w)$ for $p \in \langle S\rangle$ and $w \in \Sigma^*$. Furthermore there is an isomorphism*

$$\varphi : \langle S\rangle / Stab_{\langle S\rangle}(\Sigma^*) \to G(\mathcal{A}) \tag{2.36}$$

*compatible with the action on $\Sigma^*$.*

*Proof.* We have seen that the pointwise stabilizer of a group on a set is a normal subgroup, therefore $\langle S\rangle / Stab_{\langle S\rangle}(\Sigma^*)$ is a group.

All $p \in \langle S\rangle$, $r \in \mathrm{Stab}_{\langle S\rangle}(\Sigma^*)$ and words $w \in \Sigma^*$ satisfy

$$\beta(pr, w) = \beta(p, \beta(r, w)) = \beta(p, w), \tag{2.37}$$

so clearly $\langle S\rangle / Stab_{\langle S\rangle}(\Sigma^*)$ can inherit the action of $\langle S\rangle$ on $\Sigma^*$.

We show that $\varphi$ given for $p \in \langle S\rangle$ by

$$\varphi(p\,\mathrm{Stab}_{\langle S\rangle}(\Sigma^*)) := \beta_p\,. \tag{2.38}$$

is well defined.

Let $p, q \in \langle S\rangle$ with $p\,\mathrm{Stab}_{\langle S\rangle}(\Sigma^*) = q\,\mathrm{Stab}_{\langle S\rangle}(\Sigma^*)$. This is equivalent to $pq^{-1} \in Stab_{\langle S\rangle}(\Sigma^*)$ and hence $\beta(pq^{-1}, w) = w$ for all $w \in \Sigma^*$, which can be stated as $\beta_{pq^{-1}} = \mathbb{1}_{\mathrm{Aut}(\Sigma^*)}$. We conclude

$$\beta_q = \mathbb{1}_{\mathrm{Aut}(\Sigma^*)} \circ \beta_q = \beta_{pq^{-1}} \circ \beta_q = \beta_p\,. \tag{2.39}$$

Finally for $p \in \langle S\rangle$ and $w \in \Sigma^*$ we infer

$$p\,\mathrm{Stab}_{\langle S\rangle}(\Sigma^*) \cdot w = \beta(p, w) = \beta_p(w). \tag{2.40}$$

□

This was the last theoretic result in this chapter about automata and their visualization via state diagrams and Schreier graphs. To conclude the section we look at an example, where we calculate the Wiener index.

**Example 2.37.** Once more our object of study is the incrementation automaton $\mathcal{A}_I = (\{a, id\}, \{0, 1\}, \alpha, \beta)$ of Example 2.14. First we determine the automaton group $G(\mathcal{A}_I)$. Clearly $\beta_{id} = \mathbb{1}_{\mathrm{Aut}(\{0,1\}^*)}$, hence we focus on $\beta_a$. In Example 2.14 we have seen that $\beta_a$ increases a binary word $w$ of length $n$ by $1 \mod 2^n$, so for $m \in \mathbb{N}$, $(\beta_a)^m(w)$ increases the binary number given by the word $w$ by $m \mod 2^n$. Thus certainly $(\beta_a)^m(w) \neq w$ if $m < 2^n$. In other words, no power of $\beta_a$ is the identity element of $\mathrm{Aut}(\{0, 1\}^*)$ and hence $\beta_a$ has infinite order. This yields $G(\mathcal{A}_I) \cong \langle\beta_a\rangle \cong \mathbb{Z}$, i.e. the automaton group of the incrementation automaton is a free group of rank 1 generated by $\beta_a$.

Now we look at some Schreier graphs of the incrementation automaton in Figure 2.6. Note that the zeroth and the first Schreier graph of $\mathcal{A}_I$ are not included in Figure 2.6 since they are label-isomorphic to the respective Schreier graph of $\mathcal{A}_{FBS}$ (see Figure 2.5).

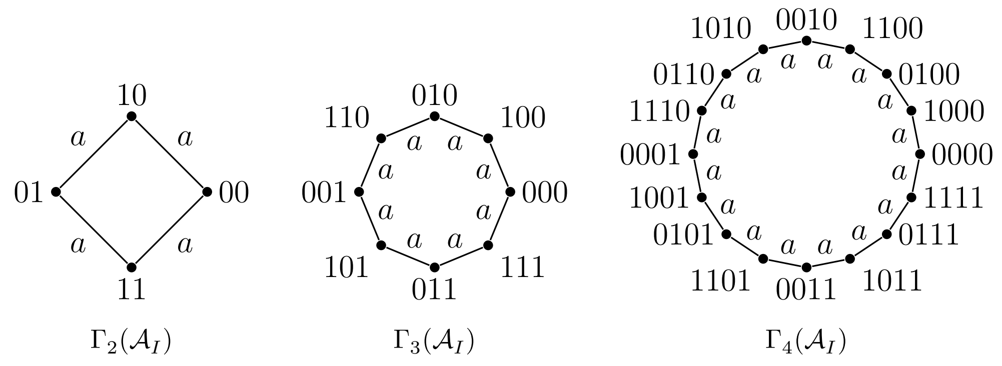


Figure 2.6: Second, third and fourth Schreier graph of the incrementation automaton.

As the Schreier graphs drawn in Figure 2.6 suggest, $\Gamma_n(\mathcal{A}_I)$ is a cycle of length $2^n$. Hence $\Gamma_n(\mathcal{A}_I)$ is connected and it is reasonable to examine the behaviour of the Wiener index and the average distance of these graphs. Clearly for fixed $n \in \mathbb{N}_0$, $d_{\Gamma_n(\mathcal{A}_I)}(\cdot)$ is the same for every vertex, so we calculate it just once via counting. Let $v$ be a vertex of $\Gamma_n(\mathcal{A}_I)$, then

$$\begin{aligned} d_{\Gamma_n(\mathcal{A}_I)}(v) &= 2 \cdot 1 + 2 \cdot 2 + \cdots + 2 \cdot \left( \frac{2^n}{2} - 1 \right) + 1 \cdot \frac{2^n}{2} \\ &= 2 \sum_{k=1}^{2^{n-1}} k - 2^{n-1} = 2^{n-1}(2^{n-1} + 1) - 2^{n-1} = 2^{2(n-1)}. \end{aligned} \tag{2.41}$$

Since $\Gamma_n(\mathcal{A}_I)$ has $2^n$ vertices, we obtain

$$W(\Gamma_n(\mathcal{A}_I)) = \frac{2^n \cdot 2^{2(n-1)}}{2} = 2^{3(n-1)} \tag{2.42}$$

and

$$\mu(\Gamma_n(\mathcal{A}_I)) = \frac{2^{2(n-1)}}{2^n - 1} = \frac{2^n + 1}{4} + \frac{1}{4 \cdot (2^n - 1)}\,. \tag{2.43}$$

This implies that the average distance of the $n$-th Schreier graph of the incrementation automaton tends towards one forth of the length of the cycle, i.e.

$$\lim_{n \to \infty} \frac{\mu(\Gamma_n(\mathcal{A}_I))}{2^n} = \frac{1}{4}\,. \tag{2.44}$$

# Chapter 3

# The Basilica automaton

The main objective of this chapter is the proof of the following new theorem:

**Theorem 3.1.** *The Wiener index of the $n$-th Schreier graph of the Basilica automaton $\mathcal{B}$ is of order $2^{\frac{5n}{2}}$. More precisely, there are constants $c_1, c_2 > 0$ s.t. for all $n \geq 2$,*

$$c_1 2^{\frac{5n}{2}} \leq W(\Gamma_n(\mathcal{B})) \leq c_2 2^{\frac{5n}{2}}. \tag{3.1}$$

The most important ingredient to prove this result are the substitution rules for the Schreier graphs of the Basilica automaton showed by D'Angeli et al. in chapter 3 of [8]. Note that $2^{\frac{5n}{2}}$ is exactly the order of the geometric mean of the simple bounds presented in the first section of chapter one (see Remark 1.13 and Proposition 1.17).

We start the first section of chapter three by introducing the Basilica automaton and formalizing the concept of graph substitution rules. Then we give two examples showing how they are utilized. Thereafter we formulate and prove Proposition 3.1 of [8] using our notations for graph substitution rules. This proposition is subsequently applied to examine the structure of Schreier graphs of the Basilica automaton.

Section two discusses the upper bound of Theorem 3.1. We initiate the section with another known upper bound of the Wiener index depending on the diameter of the graph. Thus, after determining the diameter of the $n$-th Schreier graph of the Basilica, the claim follows easily.

In the third section we prove the lower bound of Theorem 3.1. This requires some more research on the structure of the Schreier graphs of the Basilica automaton.

The fourth and final section of chapter three starts with a comparison of the bounds given by Theorem 3.1 to some calculated values of the Wiener index. Then some conclusions are drawn and two questions leading to further research topics are stated.

## 3.1 The Basilica automaton and graph substitutions

**Definition 3.2.** The *Basilica automaton* $\mathcal{B} = (\{a, b, id\}, \{0, 1\}, \alpha, \beta)$ is defined by the following transitions and outputs for $p \in \{a, b, id\}$ and $s \in \{0, 1\}$:

$$\begin{aligned} \alpha(p,1) &:= id, & \beta(a,s) &:= s, \\ \alpha(a,0) &:= b, & \beta(b,s) &:= 1-s, \\ \alpha(b,0) &:= a, & \beta(id,s) &:= s. \\ \alpha(id,0) &:= id, \end{aligned} \tag{3.2}$$

The corresponding state diagram $SD(\mathcal{B})$ of the Basilica automaton can be seen in Figure 3.1.

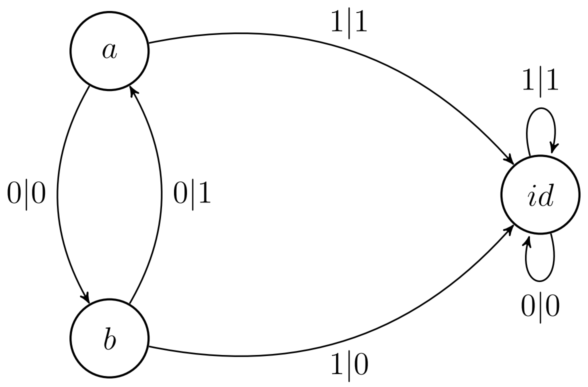


Figure 3.1: State diagram of the Basilica automaton.

The Basilica automaton is invertible, hence its automaton group, called the *Basilica group* $G(\mathcal{B})$, exists.

The Basilica automaton is named after its automaton group introduced by Grigorchuk and Żuk in [18]. The Basilica group is of major interest since it was the first example of an amenable group having exponential growth (see [3]). It has various other interesting properties and was described by Nekrashevych in [32] as the iterated monodromy group of the complex polynomial $z^2 - 1$. Using this description of the Basilica group it can be associated to a compact limit space isomorphic to the Basilica fractal, that is the Julia set of $z^2 - 1$ (see Figure 3.2 and for more details [32]). The Basilica fractal resembles the Basilica di San Marco in Venice together with its reflection in the water, so this is presumably the origin of the name 'Basilica'.

To get a feeling for the action of the Basilica group, we take a look at the first few Schreier graphs drawn in Figure 3.3.

Like the Schreier graphs of the incrementation and the first-bit-switch automaton the Schreier graphs of the Basilica automaton are similar to each other. To formalize the similarities we introduce a new concept playing a major role in the remaining chapter, since it is very convenient to describe the common structures of the Schreier graphs.

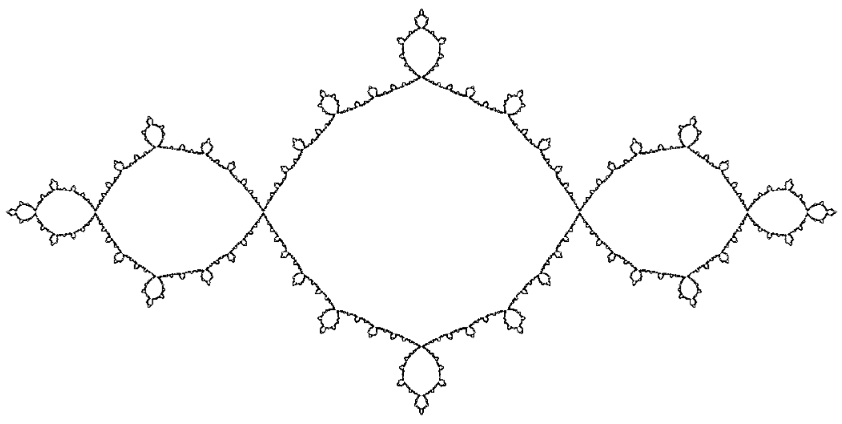


Figure 3.2: Julia set of $z^2 - 1$, drawing by Prokofiev, CC BY-SA 3.0.

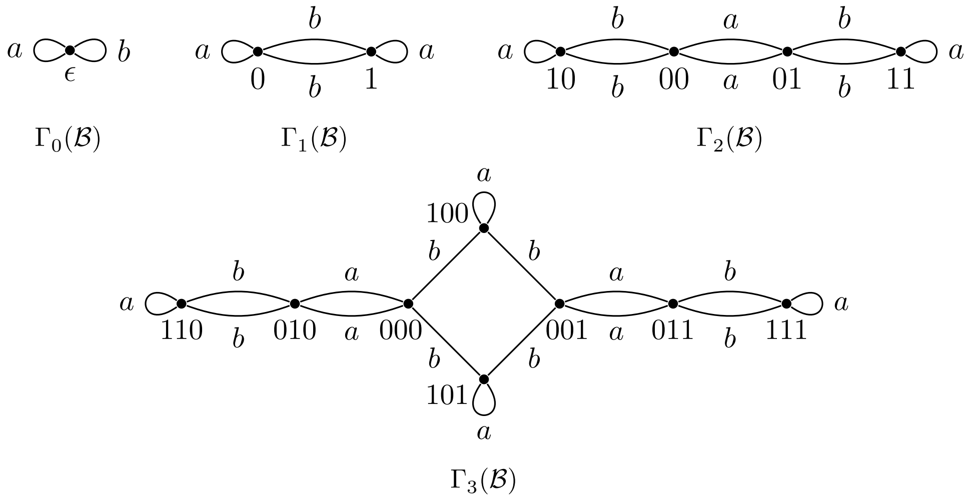


Figure 3.3: Zeroth, first, second and third Schreier graph of the Basilica automaton.

**Definition 3.3.** Let $G = (V, E)$ be a graph. A pair $(G_S\,, G'_S)$ of graphs is called a *graph substitution* applicable on $G$ if $G_S$ is a subgraph of $G$. The result of the application has the form

$$(G_S\,, G'_S)G := (G - G_S) \cup G'_S\,. \tag{3.3}$$

Let $S = \{(G_1\,, G'_1), \ldots, (G_n\,, G'_n)\}$ be a set of graph substitutions applicable on $G$. Then we define the application of $S$ on $G$ by

$$SG := (G - (G_1 \cup \cdots \cup G_n)) \cup G'_1 \cup \cdots \cup G'_n\,. \tag{3.4}$$

We call $S$ a *substitution rule* if $G_i \cong G_j$ and $G'_i \cong G'_j$ for all $i, j = 1, \ldots, n$.

**Remark 3.4.** For two graphs $G$ and $G'$, the pair $(G, G')$ is a substitution applicable on $G$. So there is always a substitution rule that applied on $G$ yields $G'$. But instead of substituting one big graph, the idea behind defining substitution rules is to find a small type of graph with a structure appearing repeatedly in $G$ and substitute it by another type of small graph.

For a set of graph substitutions $S = \{(G_1, G_1'), \dots, (G_n, G_n')\}$ it is possible that graphs $G_i$, $G_j$, for $i \neq j$, have non-empty intersection. The same is true for $G_i'$ and $G_j'$ with $i \neq j$. This is necessary to produce connected graphs. In all uses of graph substitutions here, these intersections consist only of vertices and no edges. This makes the process of 'gluing' the graphs together easier.

In order to better apprehend the definition above and get a deeper understanding of how to utilize the substitution rules we look at two examples concerning Schreier graphs of the first-bit-switch automaton and the incrementation automaton.

**Example 3.5.** We construct a substitution rule $S_n$ consisting of pairs of small graphs such that the $n+1$-th Schreier graph of the first-bit-switch automaton $\mathcal{A}_{FBS}$ of Example 2.24 can be obtained from the $n$-th Schreier graph by applying $S_n$. In Figure 2.5 we can see, that a common structure is a subgraph possessing two vertices connected by two edges. Therefore we take for every $w \in \{0,1\}^{n-1}$ the subgraph $G_w$ consisting of the two vertices $0w$ and $1w$ connected by two edges labelled $a$. A closer examination yields that $G_w$ should be replaced by $G_{0w}$ and $G_{1w}$. For a given $w \in \{0,1\}^{n-1}$ a single substitution can be seen in Figure 3.4. Hence

$$S_n := \{(G_w, G_{0w} \cup G_{1w}) \mid w \in \Sigma^{n-1}\} \tag{3.5}$$

is the substitution rule we need for $S_n\Gamma_n(\mathcal{A}_{FBS}) = \Gamma_{n+1}(\mathcal{A}_{FBS})$.

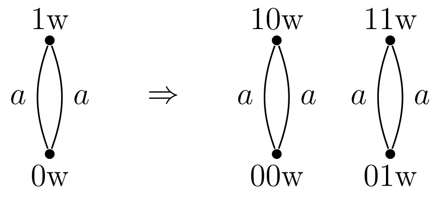


Figure 3.4: Substitution of $G_w$ by $G_{0w} \cup G_{1w}$ for $\Gamma_{n+1}(\mathcal{A}_{FBS})$.

**Example 3.6.** To construct substitution rules for the Schreier graphs of the incrementation automaton $\mathcal{A}_I = (\{a, id\}, \{0,1\}, \alpha, \beta)$ introduced in Example 2.14, we denote two different types of subgraphs. The first type is called $G_w^1$ and consists of vertices $0w, 1w \in \{0,1\}^n$ differing only in the first letter. The second type, called $G_w^2$, has two vertices $1w, 0v \in \{0,1\}^n$, where $v = \beta_a(w)$, i.e. the vertices differ in more than one letter. We substitute the first type $G_w^1$ by $G_{0w}^1 \cup G_{0w}^2$ and the second type $G_w^2$ by $G_{1w}^1 \cup G_{1w}^2$. Figure 3.5 shows the substitutions for given vertices $w, v \in \{0,1\}^{n-1}$, where $v = \beta_a(w)$.

In conclusion we have the two substitution rules

$$\begin{aligned} S_n^1 &= \{(G_w^1, G_{0w}^1 \cup G_{0w}^2) \mid w \in \{0,1\}^{n-1}\}, \\ S_n^2 &= \{(G_w^2, G_{1w}^1 \cup G_{1w}^2) \mid w \in \{0,1\}^{n-1}\}. \end{aligned} \tag{3.6}$$

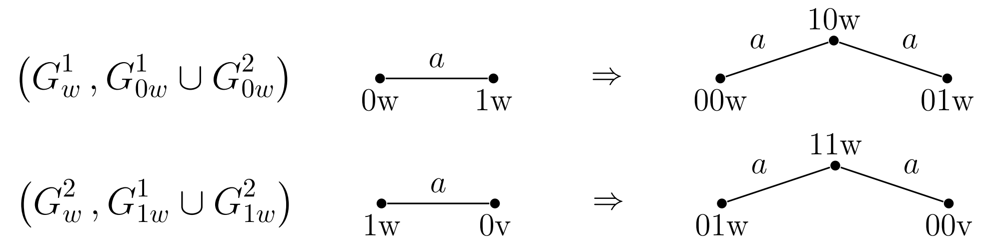

$\left(G^1_w\,, G^1_{0w} \cup G^2_{0w}\right)$
a
0w
1w
⇒
10w
a
a
00w
01w
$\left(G^2_w\,, G^1_{1w} \cup G^2_{1w}\right)$
a
1w
0v
⇒
11w
a
a
01w
00v

Figure 3.5: Substitutions for the Schreier graphs of $\mathcal{A}_I$.

To obtain $\Gamma_{n+1}(\mathcal{A}_I)$ from $\Gamma_n(\mathcal{A}_I)$ by the substitution rules we need to apply both rules at the same time. Thus we take their union, i.e.

$$\left(S^1_n \cup S^2_n\right)\,\Gamma_n(\mathcal{A}_I) = \Gamma_{n+1}(\mathcal{A}_I). \tag{3.7}$$

The same scheme works for the Basilica automaton as well. We state the substitution rules in the following proposition and prove that the application of these rules produces the $n+1$-th Schreier graph of the Basilica from its $n$-th Schreier graph.

**Proposition 3.7.** *For $w, v \in \{0,1\}^*$ with $v = \beta_b(w)$ we define the following graphs:*

- *$G^1_w$ consists of the vertex $1w$ with a loop labelled $a$ attached.*
- *$G^2_w$ consists of the two vertices $0w, 0v$ connected by an edge labelled $a$.*
- *$G^3_w$ consists of the two vertices $w, v$ connected by an edge labelled $b$.*
- *$H^1_w$ consists of the two vertices $11w, 01w$ connected by two edges labelled $b$ and a loop attached to $1w$ labelled $a$.*
- *$H^2_w$ consists of the three vertices $00w, 10v, 00v$, where $10v$ has a loop labelled $a$ attached and is connected to each of the other two vertices with one edge labelled $b$.*
- *$H^3_w := G^2_w$.*

*Furthermore we define the three substitution rules*

- *$S^1_n := \{(G^1_w\,, H^1_w) \mid w \in \{0,1\}^{n-1}\}$,*
- *$S^2_n := \{(G^2_w\,, H^2_w) \mid w \in \{0,1\}^{n-1}\}$,*
- *$S^3_n := \{(G^3_w\,, H^3_w) \mid w \in \{0,1\}^{n}\}$.*

*A visualization of the graphs and substitution rules can be found in Figure 3.6. Then*

$$\Gamma_{n+1}(\mathcal{B}) = \left(S^1_n \cup S^2_n \cup S^3_n\right)\Gamma_n(\mathcal{B}). \tag{3.8}$$

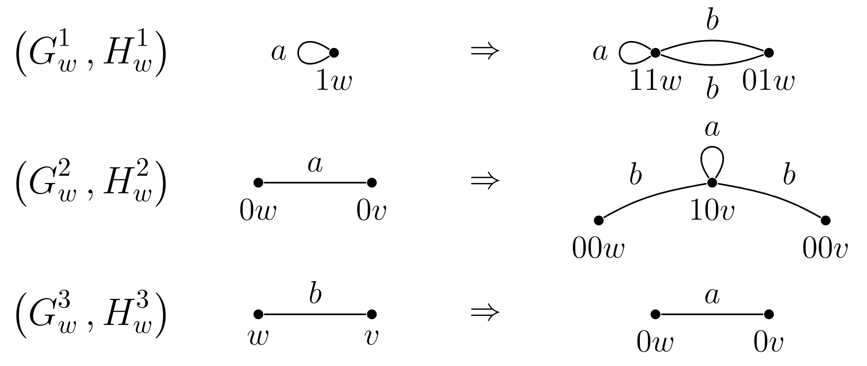


Figure 3.6: Substitutions for the Schreier graphs of $\mathcal{B}$.

*Proof.* First we look at the applications of $\beta_a$ and $\beta_b$ on $\{0,1\}^*$. Let $w \in \{0,1\}^*$ and $v := \beta_b(w)$, then

(i) $\beta_a(1w) = 1w$,

(ii) $\beta_a(0w) = 0v$,

(iii) $\beta_b(11w) = 01w$,

(iv) $\beta_b(01w) = 11w$,

(v) $\beta_b(10w) = 00w$,

(vi) $\beta_b(00w) = 10v$.

By (i), (iii) and (iv), $H^1_w$ is a subgraph of $\Gamma_{n+1}(\mathcal{B})$. $H^2_w$ is a subgraph of $\Gamma_{n+1}(\mathcal{B})$ by (i), (v) and (vi), due to $G^2_w$ being a subgraph of $\Gamma_n(\mathcal{B})$. To see that also $H^3_w$ is a subgraph of $\Gamma_{n+1}(\mathcal{B})$, we only need (ii) and the fact that $G^3_w$ is a subgraph of $\Gamma_n(\mathcal{B})$.

For different words $w, v$ all graphs $H^i_w$ and $H^j_v$ for $i,j \in \{1,2,3\}$ cannot have any edges in their intersection. Furthermore all vertices of $\Gamma_n(\mathcal{B})$ are substituted by applying the three substitution rules. Thus $\left(S^1_n \cup S^2_n \cup S^3_n\right)\Gamma_n(\mathcal{B})$ is a subgraph of $\Gamma_{n+1}(\mathcal{B})$.

To prove equality we show that $\left(S^1_n \cup S^2_n \cup S^3_n\right)\Gamma_n(\mathcal{B})$ has $2^{n+2}$ edges, as many as $\Gamma_{n+1}(\mathcal{B})$. $\Gamma_n(\mathcal{B})$ has $2^{n+1}$ edges, half of which are labelled $a$ and the other half are labelled $b$. All edges labelled $a$ are substituted by three edges and all edges labelled $b$ by only one edge. Hence $\left(S^1_n \cup S^2_n \cup S^3_n\right)\Gamma_n(\mathcal{B})$ has $3 \cdot 2^n + 2^n$ edges. $\square$

Using these handy substitution rules we can construct some more Schreier graphs. In Figure 3.8 we see the fourth and in Figure 3.7 the fifth Schreier graph of the Basilica automaton.

Further on we exhibit some common features of the Schreier graphs of the Basilica automaton, with notations taken from [8].

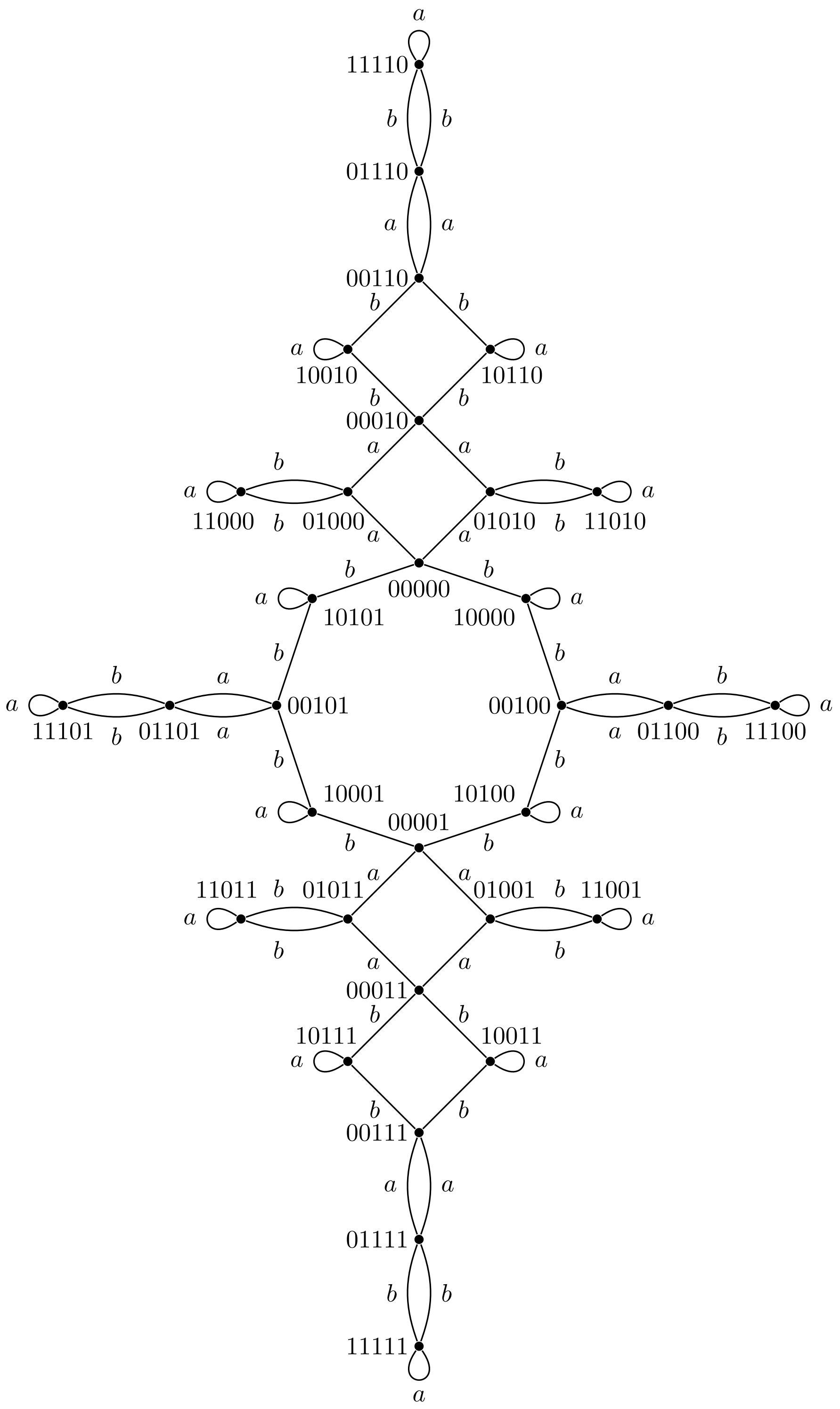


Figure 3.7: Fifth Schreier graph of the Basilica automaton.

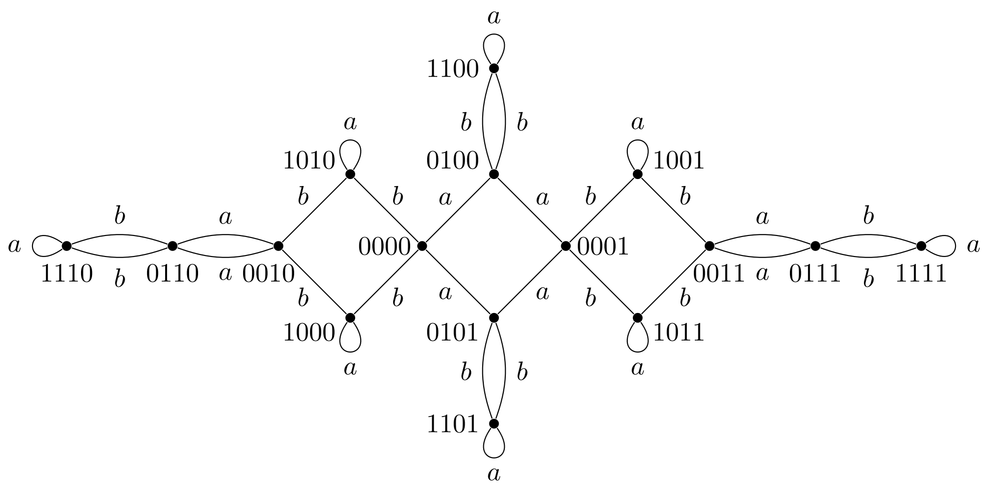


Figure 3.8: Fourth Schreier graph of the Basilica automaton.

**Remark 3.8.** Every vertex $v$ of $\Gamma_n(\mathcal{B})$ is incident to two edges labelled $b$. Either both of these edges connect $v$ with a single vertex or $v$ is part of a cycle with all edges labelled $b$. For edges labelled $a$ in addition to the two cases appearing for edges labelled $b$ a third one is possible. A vertex $v$ is incident to a single loop labelled $a$ if and only if it starts with the letter 1. This is the only case where deleting $v$ from $\Gamma_n(\mathcal{B})$ does not disconnect the graph. Hence $v$ is a cut-vertex if and only if it has no loop attached, which is equivalent to $v$'s first letter being a 0.

This fact, in combination with the substitution rules of Proposition 3.7, yields that blocks of Schreier graphs of the Basilica automaton are either cycles with some loops attached or two vertices connected by two edges where one vertex may be incident to a loop.

**Definition 3.9.** For $n \geq 3$ we call the cycle containing the vertices $0^n$ and $0^{n-1}1$ the *central cycle* of the $n$-th Schreier of the Basilica automaton.

Obviously there cannot be more than one central cycle, but it is not immediately evident, that there always is a cycle containing the vertices $0^n$ and $0^{n-1}1$. The following result elucidates this matter of fact.

**Proposition 3.10.** *For $n \geq 3$ the central cycle of the $n$-th Schreier graph of the Basilica automaton is a longest cycle of length $2^{\lceil \frac{n}{2} \rceil}$ and the vertices $0^n$ and $0^{n-1}1$ are always directly opposite to each other on the central cycle, i.e. at distance $2^{\lceil \frac{n}{2} \rceil - 1}$.*

*Proof.* All edges of a given cycle in $\Gamma_n(\mathcal{B})$ are either labelled $a$ or $b$. If they are labelled $b$, then by the substitution rules of Proposition 3.7 all edges are simply replaced by edges labelled $a$ and thus the length of the cycle does not change in the substitution process. If all edges of the cycle are labelled $a$ then they are replaced by two edges labelled $b$ and a loop, hence the length of the cycle is doubled.

In Figure 3.3 we see that $\Gamma_3(\mathcal{B})$ has a central cycle being the only longest cycle of length $4 = 2^{\lceil \frac{3}{2} \rceil}$. Thus, thanks to the observation before, all claims about the length of the central cycle are true. Furthermore the vertices 000 and 001 are directly opposite to each other. By using an inductive argument in combination with the substitution rules the same holds for $0^n$ and $0^{n-1}$. □

## 3.2 An upper bound on the Wiener index of Schreier graphs of the Basilica automaton

We start this section with a practical upper bound of the Wiener index that depends solely on the diameter of the graph.

**Proposition 3.11.** *Every graph $G = (V, E)$ on $n$ vertices satisfies*

$$W(G) \leq \frac{n(n-1)\operatorname{diam}(G)}{2}. \tag{3.9}$$

*Proof.* For every vertex $v \in V$,

$$d_G(v) = \sum_{u \in V \setminus \{v\}} d_G(v, u) \leq \sum_{u \in V \setminus \{v\}} \operatorname{diam}(G) = (n-1)\operatorname{diam}(G). \tag{3.10}$$

This yields the desired upper bound for the Wiener index:

$$W(G) = \frac{1}{2}\sum_{v \in V} d_G(v) \leq \frac{1}{2}\sum_{v \in V}(n-1)\operatorname{diam}(G) = \frac{n(n-1)\operatorname{diam}(G)}{2}. \tag{3.11}$$

□

**Example 3.12.** All vertices $u, v$ of $K_n$, the complete graph on $n$ vertices, satisfy $d_{K_n}(u, v) = 1$. Hence the diameter of the complete graph is 1 and by Proposition 3.11

$$W(K_n) \leq \frac{n(n-1)}{2}. \tag{3.12}$$

In fact, we have shown in Remark 1.13 that $n(n-1)/2$ is the Wiener index of the complete graph on $n$ vertices, so this bound is tight.

Let us have a look at $\Gamma_n(\mathcal{A}_I)$, the $n$-th Schreier graph of the incrementation automaton introduced in Example 2.14. It is a cycle of length $2^n$, so the diameter is half of its length, i.e. $\operatorname{diam}(\Gamma_n(\mathcal{A}_I)) = 2^{n-1}$. Hence

$$W(\Gamma_n(\mathcal{A}_I)) \leq 2^{2(n-1)}(2^n - 1). \tag{3.13}$$

This upper bound is significantly higher than the actual Wiener index $2^{3(n-1)}$, but still of the same order. It will turn out, that the same holds true for the Schreier graphs of the Basilica automaton.

In order to be able to apply Proposition 3.11 to the Schreier graphs of the Basilica automaton we first need their diameters.

The proof of the following result utilizes generating functions. We only use common basics like the identity

$$\sum_{n=0}^{\infty}(\alpha x)^n = \frac{1}{1-\alpha x}\,. \tag{3.14}$$

More informations about generating function can be found in [13]. Note that when working with formal power series, the convergence can be ignored.

**Proposition 3.13.** *Let $n \in \mathbb{N}$, then*

$$\operatorname{diam}(\Gamma_n(\mathcal{B})) = \begin{cases} 7\cdot 2^{\frac{n-2}{2}} - 4 & \text{if } n \text{ is even,} \\ 5\cdot 2^{\frac{n-1}{2}} - 4 & \text{if } n \text{ is odd.} \end{cases} \tag{3.15}$$

*Proof.* Let $s,t \in \{0,1\}$ and $v,w \in \{0,1\}^n$, such that the vertices $sv$ and $tw$ achieve the maximum distance in the $n+1$-th Schreier graph. Then, due to the substitution rules of Proposition 3.7, $v$ and $w$ have the maximum distance in the $n$-th Schreier graph. $1v$'s only neighbour is $0v$, thus $1v$ has a greater distance to $tw$ than $0v$. We can exchange $v$ and $w$ in this argument, hence $s=t=1$. By applying this scheme recursively we get that the only two words obtaining the maximum distance in the $n$-th Schreier graph are $1^{n-1}0$ and $1^n$.

Thus we need to calculate the length of a shortest path from $1^{n-1}0$ to $1^n$. Clearly there is no unique shortest path from $1^{n-1}0$ to $1^n$, so we take just one shortest path and call it $\pi_n$. We denote with $a_n$ and $b_n$ the number of edges in $\pi_n$ labelled by $a$ and $b$, respectively.

By looking at the zeroth and first Schreier graph of the Basilica automaton we get $a_0 = b_0 = a_1 = 0$ and $b_1 = 1$ (see Figure 3.3). We use the substitution rules of Proposition 3.7 to construct recurrences for $a_n$ and $b_n$ revealing that $a_n$ and $b_n$ do not depend on the choice of $\pi_n$.

$S_n^3$ tells us that every edge labelled $b$ in $\pi_n$ corresponds to an edge labelled $a$ in $\pi_{n+1}$. $S_n^2$ produces two edges labelled $b$ and a loop labelled $a$ that is irrelevant, since loops cannot be part of any path. But $1^{n-1}0$ and $1^n$ always have a loop attached each corresponding to one edge labelled $b$ in $\pi_{n+1}$ due to $S_n^1$. Hence we get

$$\begin{aligned} a_{n+1} &= b_n\,, \\ b_{n+1} &= 2a_n + 2. \end{aligned} \tag{3.16}$$

This leads to

$$b_{n+1} = 2b_{n-1} + 2. \tag{3.17}$$

To make the calculation easier we define $c_n$ to be the even elements and $d_n$ to be the odd elements of the sequence $b_n$, i.e. $c_n = b_{2n}$ and $d_n = b_{2n+1}$. Then

$$\begin{aligned} c_{n+1} &= 2c_n + 2, \text{ where } c_0 = 0, \\ d_{n+1} &= 2d_n + 2, \text{ where } d_0 = 1. \end{aligned} \tag{3.18}$$

We solve the recursion for $c_n$ using generating functions. Multiplication of the recursion for $c_n$ by $x^n$ and summation gives

$$\sum_{n=1}^{\infty} c_n x^n = \sum_{n=1}^{\infty} 2c_{n-1}x^n + \sum_{n=1}^{\infty} 2x^n. \tag{3.19}$$

Let $C(x) := \sum_{n=0}^{\infty} c_n x^n$, then (3.19) can be reformulated to

$$C(x) = \frac{2x}{(1-x)(1-2x)} = \frac{2}{1-2x} - \frac{2}{1-x} \tag{3.20}$$

by using partial fraction expansion. Another reformulation using the identity (3.14) yields

$$C(x) = \sum_{n=0}^{\infty} 2(2^n - 1)x^n. \tag{3.21}$$

Thus $c_n = 2(2^n - 1)$. The same approach for $d_n$ yields $d_n = 3 \cdot 2^n - 2$. Hence

$$b_n = \begin{cases} 2(2^{\frac{n}{2}} - 1) & \text{if } n \text{ is even,} \\ 3 \cdot 2^{\frac{n-1}{2}} - 2 & \text{if } n \text{ is odd.} \end{cases} \tag{3.22}$$

Finally $\operatorname{diam}(\Gamma_n(\mathcal{B})) = a_n + b_n = b_{n-1} + b_n =$

$$\begin{cases} 3 \cdot 2^{\frac{n-2}{2}} - 2 + 2(2^{\frac{n}{2}} - 1) & = 7 \cdot 2^{\frac{n-2}{2}} - 4 \quad \text{if } n \text{ is even,} \\ 2(2^{\frac{n-1}{2}} - 1) + 3 \cdot 2^{\frac{n-1}{2}} - 2 & = 5 \cdot 2^{\frac{n-1}{2}} - 4 \quad \text{if } n \text{ is odd.} \end{cases} \tag{3.23}$$

□

Now by combining this result with Proposition 3.11 we get an upper bound for the Wiener index of the $n$-th Schreier graph of the Basilica automaton.

**Proposition 3.14.** *The Wiener index of the $n$-th Schreier graph of the Basilica automaton $W(\Gamma_n(\mathcal{B}))$ is bounded from above by $\frac{7}{2} \cdot 2^{\frac{5n}{2}}$.*

*Proof.* The $n$-th Schreier graph of the Basilica automaton has $2^n$ vertices and its diameter is bounded from above by $7 \cdot 2^{\frac{n}{2}}$, thus by Proposition 3.11

$$W(\Gamma_n(\mathcal{B})) \leq \frac{2^n(2^n - 1)\operatorname{diam}(\Gamma_n(\mathcal{B}))}{2} \leq \frac{7}{2} \cdot 2^{\frac{5n}{2}}. \tag{3.24}$$

□

This proves the upper bound of Theorem 3.1 with $c_2 = \frac{7}{2}$. Ignoring the negative terms the constant can be reduced by nearly the factor $\frac{1}{2}$, but then the exact value depends on the parity of $n$.

## 3.3 A lower bound on the Wiener index of Schreier graphs of the Basilica automaton

In order to determine a lower bound of the $n$-th Schreier graph of the Basilica automaton Proposition 1.21 can be used. This leads to a lower bound of order $n \cdot 2^{2n}$. Unfortunately there is still a gap to the order $2^{\frac{5n}{2}}$ we want to reach. Thus we think of a different approach based on the structure of the Schreier graphs of the Basilica automaton. We start by introducing the well known handshaking lemma and then expand our research of the structure on the basis of the results shown in the first section of this chapter. The substitution rules shown in Proposition 3.7 prove to be especially valuable.

**Proposition 3.15.** *Every graph $G = (V, E)$ satisfies*

$$\sum_{v \in V} \deg_G(v) = 2 \cdot |E|. \tag{3.25}$$

*Proof.* If we sum up the degrees of all vertices, we count each edge twice. □

**Definition 3.16.** After removing a vertex from $\Gamma_n(\mathcal{B})$ the resulting graph has either one or two connected components. Thus for some vertex $v$, $\Gamma_n(\mathcal{B}) - v$ is the union of two connected subgraphs $G_1 = (V_1, E_1)$ and $G_2 = (V_2, E_2)$, where one subgraph may have no vertices. The graph $\Gamma_n(\mathcal{B}) - v$ has an odd number of vertices, hence we can assume $|V_1| < |V_2|$ without loss of generality. We call the subgraph of $\Gamma_n(\mathcal{B})$ induced by the smaller vertex set $V_1$ plus the vertex $v$ the *decoration* of $v$, written

$$\mathcal{D}(v) := \Gamma_n(\mathcal{B})[V_1 \cup \{v\}]. \tag{3.26}$$

A decoration $\mathcal{D}(v)$ is called a *$k$-decoration* if it is label-isomorphic to the decoration of the vertex $0^k$ in the $k$-th Schreier graph of the Basilica automaton for some $k \in \{1, \dots, n\}$. A few $k$-decorations are drawn in Figure 3.9.

Every part of a Schreier graph of the Basilica except the central cycle can be described by a decoration. This is the essence of the following result.

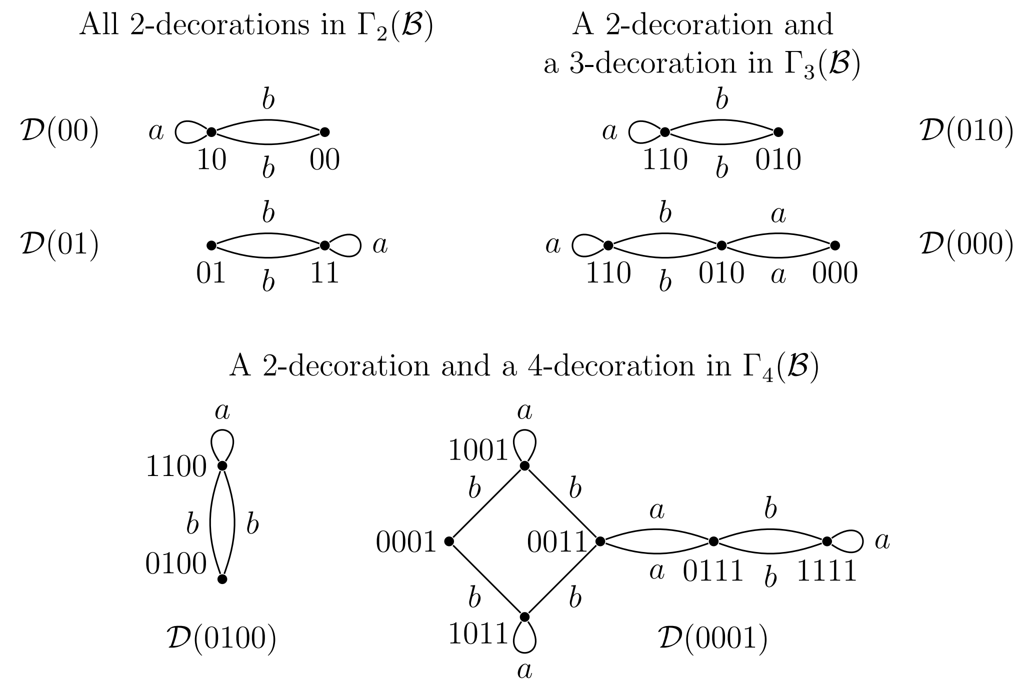
**Proposition 3.17.** *For every $v \in \{0,1\}^n \setminus \{0^n\}$, $\mathcal{D}(v)$ is a $k$-decoration if and only if $v$ is of the form $0^{k-1}1w$ for some $w \in \{0,1\}^{n-k}$.*

*Proof.* For a given vertex $v \in \{0,1\}^n$, let $S_{\mathcal{D}(v)}$ be the restriction of the substitution rules of Proposition 3.7 to the decoration of $v$, i.e.

$$S_{\mathcal{D}(v)} := \{(G_n^i, H_n^i) \in S_n^i \mid G_n^i \subset \mathcal{D}(v), i \in \{1,2,3\}\}. \tag{3.27}$$

Then

$$S_{\mathcal{D}(v)} \mathcal{D}(v) = \mathcal{D}(0v). \tag{3.28}$$



All 2-decorations in Γ2(B)
A 2-decoration and a 3-decoration in Γ3(B)
D(00)
D(01)
D(010)
D(000)
10
00
01
11
110
010
000
A 2-decoration and a 4-decoration in Γ4(B)
1100
0100
1001
0001
0011
1011
0111
1111
D(0100)
D(0001)
a
b

Figure 3.9: Some $k$-decorations (compare with Figure 3.3 and 3.8).

Suppose $v = 0^{k-1}1w \in \{0,1\}^n$ for some $k \in \{1,\dots,n\}$, then $\mathcal{D}(v)$ can be created by applying $k-1$ restricted substitution rules starting with $\mathcal{D}(1w)$. The decoration of $1w$ is just a loop incident to one edge labelled $a$, and thus it is label-isomorphic to $\mathcal{D}(0)$. $k-1$ restricted substitutions applied to $\mathcal{D}(0)$ yield $\mathcal{D}(0^k)$. These restricted substitutions are label-isomorphic to the ones used to obtain $\mathcal{D}(0^{k-1}1w)$ from $\mathcal{D}(1w)$, hence $\mathcal{D}(0^{k-1}1w)$ is label-isomorphic to $\mathcal{D}(0^k)$.

Now let $v \in \{0,1\}^n \backslash \{0^n\}$, then there exist $k \in \{1,\dots,n\}$ and $w \in \{0,1\}^{n-k}$ such that $v = 0^{k-1}1w$. Hence the decoration $\mathcal{D}(v)$ is a $k$-decoration for some $k \in \{1,\dots,n\}$. Furthermore a decoration clearly cannot be a $k$- and an $l$-decoration if $k \neq l$. □

Another ingredient necessary to prove the lower bound is the number of vertices in a $k$-decoration.

**Proposition 3.18.** *A $k$-decoration is a graph on*

$$\frac{2^{k+1} + (-1)^k + 3}{6} \tag{3.29}$$

*vertices.*

*Proof.* To start off we take a similar approach as in the second half of the proof of Proposition 3.13. Let $a_k$ be the number of edges labelled $a$ and $b_k$ the number of edges labelled $b$ in the decoration of $0^k$ in $\Gamma_k(\mathcal{B})$.

By looking at the first and second Schreier graph of the Basilica automaton we obtain $a_1 = a_2 = 1$, $b_1 = 0$ and $b_2 = 2$ (see Figure 3.3).

The decoration of $0^{k+1}$ in $\Gamma_{k+1}(\mathcal{B})$ can be directly obtained by applying the substitution rules of Proposition 3.7 on the decoration of $0^k$ in $\Gamma_k(\mathcal{B})$. An edge labelled $a$ in $\mathcal{D}(0^k)$ corresponds to one edge labelled $a$ and two edges labelled $b$ in $\mathcal{D}(0^{k+1})$ and an edge labelled $b$ corresponds to just one edge labelled $a$. Thus we get

$$\begin{aligned} a_{k+1} &= a_k + b_k\,, \\ b_{k+1} &= 2a_k\,. \end{aligned} \tag{3.30}$$

This yields

$$a_{k+1} = a_k + 2a_{k-1} \tag{3.31}$$

and we set $a_0 := 0$ to complete the recursion of $a_k$. Multiplication of $a_k$ by $x^k$ and summation gives

$$\sum_{k=2}^{\infty} a_k x^k = \sum_{k=2}^{\infty} a_{k-1} x^k + \sum_{k=2}^{\infty} 2a_{k-2} x^k. \tag{3.32}$$

Let $A(x) := \sum_{n=0}^{\infty} a_k x^k$, then (3.32) can be reformulated to

$$A(x) = \frac{x}{(1+x)(1-2x)} = \frac{1}{3}\left(\frac{1}{1-2x} - \frac{1}{1+x}\right) \tag{3.33}$$

by using partial fraction expansion. Another reformulation using the identity (3.14) leads to

$$A(x) = \sum_{k=0}^{\infty} \frac{2^k - (-1)^k}{3} x^k. \tag{3.34}$$

Thus

$$a_k = \frac{2^k - (-1)^k}{3} \quad \text{and} \quad b_k = \frac{2^k + 2(-1)^k}{3}\,. \tag{3.35}$$

Let $n_k$ be the number of vertices in $\mathcal{D}(0^k)$. The vertex $0^k$ has degree 2 in $\mathcal{D}(0^k)$ and all other vertices have degree 4. Hence by the handshaking lemma (Proposition 3.15),

$$4n_k - 2 = 2(a_k + b_k). \tag{3.36}$$

We isolate $n_k$ and insert $a_k$ and $b_k$ to get the desired result

$$n_k = \frac{a_k + b_k + 1}{2} = \frac{2^{k+1} + (-1)^k + 3}{6}\,. \tag{3.37}$$

□

With this proposition we have finished the preparations and gained all tools required to prove the lower bound of Theorem 3.1.

**Proposition 3.19.** *The Wiener index of the $n$-th Schreier graph of the Basilica automaton $W(\Gamma_n(\mathcal{B}))$ is bounded from below by $\frac{1}{12} \cdot 2^{\frac{5n}{2}}$.*

*Proof.* In the first step we show that $d_{\Gamma_n(\mathcal{B})}(0^n) \leq d_{\Gamma_n(\mathcal{B})}(v)$ for all vertices $v \in \{0,1\}^n$. In the second step we find a lower bound for $d_{\Gamma_n(\mathcal{B})}(0^n)$ and insert it into the reformulation of the Wiener index.

Let $v$ be a neighbour of $0^n$ on the central cycle as shown in Figure 3.10. There we can see that all vertices above the dashed line are closer to $v$ and all below are closer to $0^n$. Because of the symmetries of the Schreier graphs, for every vertex above there is one vertex below and vice versa. Thus by Proposition 1.25, $d_{\Gamma_n(\mathcal{B})}(v) = d_{\Gamma_n(\mathcal{B})}(0^n)$. Inductively this equality is true for all vertices $v$ of the central cycle.

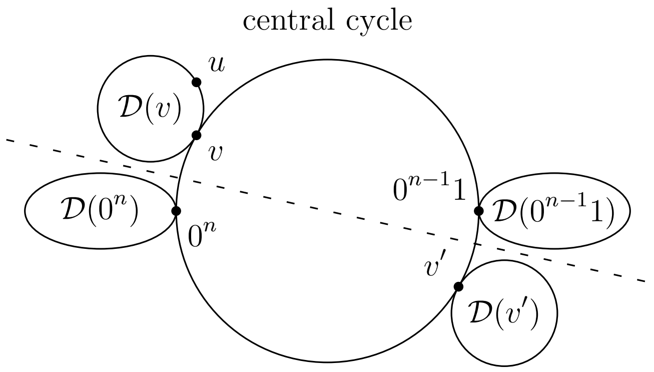


Figure 3.10: Simplified representation of a Schreier graph of the Basilica automaton.

Now for $v$ being a vertex of the central cycle, let $u$ be a neighbour of $v$ not contained in the central cycle. Then $v$ is a cut-vertex and $\mathcal{D}(v)$ is a $k$-decoration for some $k$ in $\{1,\ldots,n\}$, where all vertices outside of this $k$-decoration are closer to $v$ than to $u$ (see Figure 3.10). Combined with Proposition 1.25 we obtain the following inequality:

$$d_{\Gamma_n(\mathcal{B})}(u) + \frac{2^{k+1} + (-1)^k + 3}{6} \geq d_{\Gamma_n(\mathcal{B})}(v) + 2^n - \frac{2^{k+1} + (-1)^k + 3}{6}. \tag{3.38}$$

By shifting the terms and making some estimates we get a lower bound on the difference $d_{\Gamma_n(\mathcal{B})}(u) - d_{\Gamma_n(\mathcal{B})}(v)$. For $n \geq 2$,

$$\begin{aligned} d_{\Gamma_n(\mathcal{B})}(u) - d_{\Gamma_n(\mathcal{B})}(v) &\geq 2^n - 2 \cdot \frac{2^{k+1} + (-1)^k + 3}{6} \\ &\geq 2^n - \frac{2^{n+1} + 4}{3} = \frac{2^n - 4}{3} \geq 0. \end{aligned} \tag{3.39}$$

To summarize, we have finished the first step and showed that for any vertex $u \in \{0,1\}^n$,

$$d_{\Gamma_n(\mathcal{B})}(0^n) \leq d_{\Gamma_n(\mathcal{B})}(u). \tag{3.40}$$

By Proposition 3.10, $0^n$ is directly opposite to $0^{n-1}1$ on the longest cycle of length $2^{\lceil \frac{n}{2} \rceil}$. Every vertex in the decoration of $0^{n-1}1$ is farther away from

$0^n$ than from $0^{n-1}1$, i.e

$$d_{\Gamma_n(\mathcal{B})}(0^n, w) \geq \frac{2^{\frac{n}{2}}}{2} \tag{3.41}$$

for all $w \in \mathcal{D}(0^{n-1}1)$. By Proposition 3.18 the decoration of $0^{n-1}1$ has

$$\frac{2^{n+1} + (-1)^n + 3}{6} > \frac{2^n}{3} \tag{3.42}$$

vertices. We put all these estimates together to get a lower bound on the sum of all distances to $0^n$:

$$d_{\Gamma_n(\mathcal{B})}(0^n) = \sum_{v \in \{0,1\}^n} d_{\Gamma_n(\mathcal{B})}(0^n, v) > \sum_{v \in \mathcal{D}(0^{n-1}1)} d_{\Gamma_n(\mathcal{B})}(0^n, v) > \frac{2^n}{3}\frac{2^{\frac{n}{2}}}{2} = \frac{2^{\frac{3n}{2}}}{6}. \tag{3.43}$$

Finally

$$W(\Gamma_n(\mathcal{B})) = \frac{1}{2}\sum_{v \in \{0,1\}^n} d_{\Gamma_n(\mathcal{B})}(v) \geq \frac{1}{2}\sum_{v \in \{0,1\}^n} d_{\Gamma_n(\mathcal{B})}(0^n) > \frac{2^n}{2} \cdot \frac{2^{\frac{3n}{2}}}{6} = \frac{2^{\frac{5n}{2}}}{12}. \tag{3.44}$$

□

This proves the lower bound of Theorem 3.1 with $c_1 = \frac{1}{12}$. Using the methods from above the constant $c_1$ cannot easily be increased, although there are many strict inequalities in the proof.

## 3.4 Further discussion on the Wiener index of Schreier graphs of the Basilica automaton and general automata

As a part of this thesis a C++-algorithm was implemented that determines a simplified adjacency list of the $n$-th Schreier graph of the Basilica automaton for some given $n \geq 3$. It can be found in Appendix A.2. Using this algorithm and the Breadth-First Search Wiener Index Calculation introduced in Section 1.3 we calculated the Wiener index of $\Gamma_n(\mathcal{B})$ for $n = 3, \ldots, 20$. Due to the exponential growth of the graphs it is hardly feasible to continue this calculation with increasing $n$. In Table 3.1 a comparison of the Wiener index and the bounds of Theorem 3.1 with $c_1 = \frac{1}{12}$ and $c_2 = \frac{7}{2}$ for $n = 2, \ldots, 10$ can be found.

A thorough analysis of the first 20 values with a computer algebra system suggests that

$$W(\Gamma_n(\mathcal{B})) \sim \begin{cases} \frac{123}{224} \cdot 2^{\frac{5n}{2}} + \mathcal{O}(2^{2n}) & \text{if } n \text{ even,} \\ \frac{1}{\sqrt{2}} \cdot \frac{177}{224} \cdot 2^{\frac{5n}{2}} + \mathcal{O}(2^{2n}) & \text{if } n \text{ odd.} \end{cases} \tag{3.45}$$

| $n$ | $\left\lceil \frac{1}{12} \cdot 2^{\frac{5n}{2}} \right\rceil$ | $W(\Gamma_n(\mathcal{B}))$ | $\left\lfloor \frac{7}{2} \cdot 2^{\frac{5n}{2}} \right\rfloor$ |
|---|---|---|---|
| 2 | 3 | 10 | 112 |
| 3 | 16 | 70 | 633 |
| 4 | 86 | 436 | 3584 |
| 5 | 483 | 2728 | 20274 |
| 6 | 2731 | 15952 | 114688 |
| 7 | 15447 | 95392 | 648773 |
| 8 | 87382 | 543040 | 3670016 |
| 9 | 494304 | 3183232 | 20760745 |
| 10 | 2796203 | 17900800 | 117440512 |

Table 3.1: The Wiener index in comparison to its lower and upper bound.

Note that the constants in front of $2^{\frac{5n}{2}}$ are very close to each other since $\frac{177}{\sqrt{2}} \approx 125.16$. Unfortunately it seems rather difficult to prove (3.45), or even directly calculate the Wiener indices of the Schreier graphs of the Basilica automaton. Proposition 3.7, the main ingredient to prove the lower and upper bound, appears hardly suitable for this task.

Aside from determining the exact behaviour of the Wiener index many other interesting questions arise. What can be said about the automaton group if we know the order of growth of the Wiener index? In Example 2.31 it was shown that the first-bit-switch automaton $\mathcal{A}_{FBS}$ generates a finite group and in Example 2.33 we have seen that all except the first two of its Schreier graphs are disconnected. Hence, the Wiener indices of all except the first two Schreier graphs of the first-bit-switch automaton are infinite. For the incrementation automaton $\mathcal{A}_I$ the situation is rather different. Example 2.37 showed that the automaton group of the incrementation automaton is free abelian, thus infinite, and

$$W(\Gamma_n(\mathcal{A}_I)) = \frac{1}{8} \cdot 2^{3n}. \tag{3.46}$$

These facts suggest a connection between the automaton group and the behaviour of the Wiener index on the Schreier graphs. Note that also free non-abelian groups can be realized as automata groups (e.g. automaton 2240 in [4]).

Another interesting aspect is the order of growth of the Wiener index being directly related to the order of growth of the diameter, for all three automatons studied in this master thesis, i.e. for $\mathcal{A} \in \{\mathcal{A}_I, \mathcal{A}_{FBS}, \mathcal{B}\}$,

$$W(\Gamma_n(\mathcal{A})) = \Theta(2^{2n} \operatorname{diam}(\Gamma_n(\mathcal{A}))). \tag{3.47}$$

By Proposition 3.11

$$W(\Gamma_n(\mathcal{A})) = \mathcal{O}(m^{2n} \operatorname{diam}(\Gamma_n(\mathcal{A}))) \tag{3.48}$$

holds for all automata $\mathcal{A}$ with a set of input and output symbols of cardinality $m$, but there is nothing known about the other direction.

# Appendix A

# C++-implementations

## A.1 The Breadth-First Search Wiener Index Calculation

The code of the function is included as complete entity, i.e. it is not split into header and source file. The standard template libraries for vector and queue are necessary to compile the function, thus they are included in the code.

Implementation of the BFS-WIC

```
#include <vector>
#include <queue>

//! \brief Calculates the Wiener index of the graph
//!         given by @p adjacency_list.
//!
//! \param[in]  adjacency_list    a connected graph
//! \return     the calculated Wiener index
uint64_t bfs_wic(const std::vector<std::vector<uint64_t>>&
                      adjacency_list)
{
    uint64_t wiener_index = 0;

    for (size_t i = 0; i < adjacency_list.size(); ++i)
    {
        std::vector<char> vertex_queued(adjacency_list.size());
        vertex_queued[i] = 1;

        std::queue<std::pair<uint64_t, uint64_t>> bfs_queue;
        bfs_queue.emplace(i, 0);

        while (bfs_queue.size() > 0)
        {
            const auto& current_pair = bfs_queue.front();

            for (const auto& vertex :
                 adjacency_list[current_pair.first])
            {
                if (vertex_queued[vertex] == 0)
                {
                    bfs_queue.emplace(vertex,
                                      current_pair.second + 1);
                    vertex_queued[vertex] = 1;
                }
            }

            wiener_index = wiener_index + current_pair.second;
            bfs_queue.pop();
        }
    }

    wiener_index = wiener_index / 2;

    return wiener_index;
}
```

Note that this function does not check the connectedness of the graph. In case a disconnected graph is inserted, the Wiener index of every connected component is calculated and all these indices are summed up. Furthermore there is no mechanism preventing or detecting overflow errors.

## A.2 Creation of adjacency lists of simplified Schreier graphs of the Basilica automaton

As in Appendix A.1 the code is not split into header and source file and we include the necessary standard template libraries in the beginning. The main idea of the following implementation is interpreting each word over the alphabet consisting of the symbols 0 and 1 as a binary number. Thus, every given word of length $n$ can be associated with a certain number between 0 and $2^n - 1$. To switch from a word to a number or vice versa the functions *calcAssociatedVal* and *calcAssociatedWord* are used.

Functions to switch from word/value to value/word

```
#include <vector>

//! \brief Calculates the value associated to the word
//!        given by @p word.
//!
//! \param[in]  word    a word written with 0s and 1s
//! \return     the associated value
size_t calcAssociatedVal(const std::vector<bool>& word)
{
    size_t val = 0;

    for (auto i = word.size(); i > 0; --i)
    {
        val = val * 2 + word[i - 1];
    }

    return val;
}

//! \brief Calculates the word with @p length letters
//!        that is associated to the value given by @p val.
//!
//! \param[in]  val     a value
//! \param[in]  length  the length of the resulting word
//! \return     the associated word
std::vector<bool> calcAssociatedWord(size_t val,
                                     const size_t length)
{
    std::vector<bool> word(length, 0);

    size_t i = 0;
    while (val > 0)
    {
        word[i] = val % 2;
        val = val / 2;
        ++i;
    }

    return word;
}
```

The next six functions correspond to $\beta(s, w)$ where $\beta$ is the output function of the Basilica automaton, $s$ a state and $w$ a word. The state $s$ inserted into the output function is determined by the variable *state_b* and the function itself. For the non-inverted functions we have $s = b$ if *state_b* is true and $s = a$ otherwise. The inverted functions take $s = b^{-1}$ if *state_b* is true and $s = a^{-1}$ if *state_b* is false.

Functions corresponding to the output function of the Basilica automaton

```
//! \brief Applies the output function to the state given
//!        by @p state_b and the word @p word.
//!
//! \param[in]  word        the word inserted into the
//!                         output function
//! \param[in]  state_b     implies state b if state_b is true,
//!                         otherwise implies state a
//! \return     result of the applied output function
std::vector<bool> outputFunc(std::vector<bool> word,
                             bool state_b)
{
    size_t i = 0;
    bool last_val = 0;
    while (i < word.size() && last_val == 0)
    {
        last_val = word[i];
        if (state_b)
        {
            word[i] = !word[i];
        }
        ++i;
        state_b = !state_b;
    }
    return word;
}

//! \brief Applies the output function to the state a
//!         and the word @p word.
//!
//! \param[in]  word        the word inserted into the
//!                         output function
//! \return     result of the applied output function
std::vector<bool> edgeA(std::vector<bool> word)
{
    return outputFunc(word, false);
}

//! \brief Applies the output function to the state b
//!         and the word @p word.
//!
//! \param[in]  word        the word inserted into the
//!                         output function
//! \return     result of the applied output function
std::vector<bool> edgeB(std::vector<bool> word)
{
    return outputFunc(word, true);
}

//! \brief Applies the output function to the inverse state
//!         given by @p state_b and the word @p word.
//!
//! \param[in]  word        the word inserted into the
//!                         output function
//! \param[in]  state_b     implies the inverse state of b if
//!                         state_b is true, otherwise
//!                         implies the inverse state of a
//! \return     result of the applied output function
std::vector<bool> inverseOutputFunc(std::vector<bool> word,
                                    bool state_b)
{
    size_t i = 0;
    bool last_set_val = 0;
    while (i < word.size() && last_set_val == 0)
    {
        if (state_b)
        {
            word[i] = !word[i];
        }
        last_set_val = word[i];
        ++i;
        state_b = !state_b;
    }

    return word;
}

//! \brief Applies the output function to the inverse
//!         state of a and the word @p word.
```

```
//!
//! \param[in]  word        the word inserted into the
//!                         output function
//! \return     result of the applied output function
std::vector<bool> edgeAInverse(std::vector<bool> word)
{
    return inverseOutputFunc(word, false);
}

//! \brief Applies the output function to the inverse
//!        state of b and the word @p word.
//!
//! \param[in]  word    the word inserted into the
//!                         output function
//! \return     result of the applied output function
std::vector<bool> edgeBInverse(std::vector<bool> word)
{
    return inverseOutputFunc(word, true);
}
```

The last C++-function is the main control unit utilizing the above functions to create the adjacency list. In order to avoid parallel edges and loops the output function is not applied to every combination of state and word. This is guaranteed by the if statements.

Implementation of the main control unit

```
//! \brief Creates an adjacency list of the simplified
//!        n-th Schreier graph of the Basilica automaton
//!
//! \param[in]  n   determines the (n-th) Schreier graph
//! \return     the created adjacency list
std::vector<std::vector<uint64_t>>
    createBasilicaAdjacencyList(const unsigned int n)
{
    size_t nr_vertices = 1 << n;
    std::vector<std::vector<uint64_t>>
                    adjacency_list(nr_vertices);

    for (size_t i = 0; i < nr_vertices; ++i)
    {
        adjacency_list[i].push_back(calcAssociatedVal(
                    edgeB(calcAssociatedWord(i, n))));
        if (i % 4 < 2)
        {
            adjacency_list[i].push_back(calcAssociatedVal(
                    edgeBInverse(calcAssociatedWord(i, n))));
        }
        if (i % 2 == 0)
        {
            adjacency_list[i].push_back(calcAssociatedVal(
                    edgeA(calcAssociatedWord(i, n))));

            if (i % 8 < 4)
            {
                adjacency_list[i].push_back(calcAssociatedVal(
                    edgeAInverse(calcAssociatedWord(i, n))));
            }
        }
    }

    return adjacency_list;
}
```

# Bibliography

[1] Adian, S. I.: *The Burnside problem and identities in groups*, volume 95 of *Ergebnisse der Mathematik und ihrer Grenzgebiete [Results in Mathematics and Related Areas]*. Springer-Verlag, Berlin-New York, 1979, ISBN 3-540-08728-1. Translated from the Russian by John Lennox and James Wiegold.

[2] Alešin, S. V.: *Finite automata and the Burnside problem for periodic groups.* Mat. Zametki, 11:319–328, 1972, ISSN 0025-567X.

[3] Bartholdi, L. and Virág, B.: *Amenability via random walks.* Duke Math. J., 130(1):39–56, 2005, ISSN 0012-7094. `https://doi.org/10.1215/S0012-7094-05-13012-5`.

[4] Bondarenko, I., Grigorchuk, R. I., Kravchenko, R., Muntyan, Y., Nekrashevych, V., Savchuk, D., and Šunić, Z.: *On classification of groups generated by 3-state automata over a 2-letter alphabet.* Algebra Discrete Math., (1):1–163, 2008, ISSN 1726-3255.

[5] Chan, T. M.: *More algorithms for all-pairs shortest paths in weighted graphs.* SIAM J. Comput., 39(5):2075–2089, 2010, ISSN 0097-5397. `https://doi.org/10.1137/08071990X`.

[6] Chan, T. M.: *All-pairs shortest paths for unweighted undirected graphs in $o(mn)$ time.* ACM Trans. Algorithms, 8(4):Art. 34, 17, 2012, ISSN 1549-6325. `https://doi.org/10.1145/2344422.2344424`.

[7] Chepoi, V. and Klavžar, S.: *The wiener index and the szeged index of benzenoid systems in linear time.* Journal of Chemical Information and Computer Sciences, 37(4):752–755, 1997. `https://doi.org/10.1021/ci9700079`.

[8] D'Angeli, D., Donno, A., Matter, M., and Nagnibeda, T.: *Schreier graphs of the Basilica group.* J. Mod. Dyn., 4(1):167–205, 2010, ISSN 1930-5311. `https://doi.org/10.3934/jmd.2010.4.167`.

[9] Dankelmann, P.: *Computing the average distance of an interval graph.* Inform. Process. Lett., 48(6):311–314, 1993, ISSN 0020-0190. `https://doi.org/10.1016/0020-0190(93)90174-8`.

[10] Darafsheh, M., Jolany, H., and Khalifeh, M.: *Computing the wiener index of a phenylenic pattern.* Fullerenes Nanotubes and Carbon Nanostructures - FULLER NANOTUB CARBON NANOSTR, 19:749–752, November 2011.

[11] Dobrynin, A. A., Entringer, R., and Gutman, I.: *Wiener index of trees: theory and applications.* Acta Appl. Math., 66(3):211–249, 2001, ISSN 0167-8019. `https://doi.org/10.1023/A:1010767517079`.

[12] Entringer, R. C., Jackson, D. E., and Snyder, D. A.: *Distance in graphs.* Czechoslovak Math. J., 26(101)(2):283–296, 1976, ISSN 0011-4642.

[13] Flajolet, P. and Sedgewick, R.: *Analytic combinatorics.* Cambridge University Press, Cambridge, 2009, ISBN 978-0-521-89806-5. `https://doi.org/10.1017/CBO9780511801655`.

[14] Gluškov, V. M.: *The abstract theory of automata. I.* Magyar Tud. Akad. Mat. Fiz. Oszt. Közl., 13:287–309, 1963.

[15] Grigorchuk, R. I.: *On Burnside's problem on periodic groups.* Funktsional. Anal. i Prilozhen., 14(1):53–54, 1980, ISSN 0374-1990.

[16] Grigorchuk, R. I.: *On the Milnor problem of group growth.* Dokl. Akad. Nauk SSSR, 271(1):30–33, 1983, ISSN 0002-3264.

[17] Grigorchuk, R. I.: *Degrees of growth of finitely generated groups and the theory of invariant means.* Izv. Akad. Nauk SSSR Ser. Mat., 48(5):939–985, 1984, ISSN 0373-2436.

[18] Grigorchuk, R. I. and Żuk, A.: *On a torsion-free weakly branch group defined by a three state automaton.* Internat. J. Algebra Comput., 12(1-2):223–246, 2002, ISSN 0218-1967. `https://doi.org/10.1142/S0218196702001000`, International Conference on Geometric and Combinatorial Methods in Group Theory and Semigroup Theory (Lincoln, NE, 2000).

[19] Gupta, N.: *On groups in which every element has finite order.* Amer. Math. Monthly, 96(4):297–308, 1989, ISSN 0002-9890. `https://doi.org/10.2307/2324085`.

[20] Gupta, N. and Sidki, S.: *On the Burnside problem for periodic groups.* Math. Z., 182(3):385–388, 1983, ISSN 0025-5874. `https://doi.org/10.1007/BF01179757`.

[21] Gutman, I. and Yeh, Y. N.: *The sum of all distances in bipartite graphs.* Math. Slovaca, 45(4):327–334, 1995, ISSN 0139-9918.

[22] Gutman, I., Yeh, Y. N., and Chen, J. C.: *On the sum of all distances in graphs.* Tamkang J. Math., 25(1):83–86, 1994, ISSN 0049-2930.

[23] Hořejš, J.: *Transformations defined by finite automata.* Problemy Kibernet., 9:23–26, 1963.

[24] Kleene, S. C.: *Representation of events in nerve nets and finite automata.* In *Automata studies*, Annals of mathematics studies, no. 34, pages 3–41. Princeton University Press, Princeton, N. J., 1956.

[25] Knuth, D. E.: *The art of computer programming. Vol. 1: Fundamental algorithms.* Second printing. Addison-Wesley Publishing Co., Reading, Mass.-London-Don Mills, Ont, 1969.

[26] Korte, B. and Vygen, J.: *Combinatorial optimization*, volume 21 of *Algorithms and Combinatorics.* Springer, Berlin, 2018, ISBN 978-3-662-56038-9; 978-3-662-56039-6. `https://doi.org/10.1007/978-3-662-56039-6`, Theory and algorithms, sixth edition.

[27] March, L. and Steadman, P.: *The Geometry of Environment*, volume 6. The MIT Press, January 1971.

[28] McCulloch, W. S. and Pitts, W.: *A logical calculus of the ideas immanent in nervous activity.* Bull. Math. Biophys., 5:115–133, 1943. `https://doi.org/10.1007/bf02478259`.

[29] Mealy, G. H.: *A method for synthesizing sequential circuits.* Bell System Tech. J., 34:1045–1079, 1955, ISSN 0005-8580. `https://doi.org/10.1002/j.1538-7305.1955.tb03788.x`.

[30] Milnor, J.: *Problem 5603.* The American Mathematical Monthly, 75(6):685–686, 1968, ISSN 00029890, 19300972. `http://www.jstor.org/stable/2313822`.

[31] Moore, E. F.: *Gedanken-experiments on sequential machines.* In *Automata studies*, Annals of mathematics studies, no. 34, pages 129–153. Princeton University Press, Princeton, N. J., 1956.

[32] Nekrashevych, V.: *Self-similar groups*, volume 117 of *Mathematical Surveys and Monographs.* American Mathematical Society, Providence, RI, 2005, ISBN 0-8218-3831-8. `https://doi.org/10.1090/surv/117`.

[33] Nikolić, S., Trinajstić, N., and Mihalić, Z.: *The wiener index: Development and applications.* Croatica Chemica Acta, 68:105–129, January 1995.

[34] Pettie, S.: *A new approach to all-pairs shortest paths on real-weighted graphs.* Theoret. Comput. Sci., 312(1):47–74, 2004, ISSN 0304-3975. `https://doi.org/10.1016/S0304-3975(03)00402-X`, Automata, languages and programming.

[35] Sakarovitch, J.: *Kleene's theorem revisited.* In *Trends, techniques, and problems in theoretical computer science (Smolenice, 1986)*, volume 281

of *Lecture Notes in Comput. Sci.*, pages 39–50. Springer, Berlin, 1987. `https://doi.org/10.1007/3540185356_29`.

[36] Shallit, J.: *A Second Course in Formal Languages and Automata Theory.* Cambridge University Press, 2008.

[37] Suščans'kiĭ, V. Ī.: *Periodic p-groups of permutations and the unrestricted Burnside problem.* Dokl. Akad. Nauk SSSR, 247(3):557–561, 1979, ISSN 0002-3264.

[38] Turing, A. M.: *On Computable Numbers, with an Application to the Entscheidungsproblem.* Proc. London Math. Soc. (2), 42(3):230–265, 1936, ISSN 0024-6115. `https://doi.org/10.1112/plms/s2-42.1.230`.

[39] Wagner, S. G.: *A class of trees and its Wiener index.* Acta Appl. Math., 91(2):119–132, 2006, ISSN 0167-8019. `https://doi.org/10.1007/s10440-006-9026-5`.

[40] Wiener, H.: *Structural determination of paraffin boiling points.* Journal of the American Chemical Society, 69(1):17–20, 1947. `https://doi.org/10.1021/ja01193a005`, PMID: 20291038.