Formal Languages
An automata-theoretic introduction

Prof. Dr. Günter Hotz
Universität des Saarlandes

Dr. Klaus Estenfeld
Universität des Saarlandes

English translation by
Dipl.-Inform. Armin Reichert
Alumnus Universität des Saarlandes

Preface

This book is meant to be the second edition of the book Hotz/Walter: "Automatentheorie und Formale Sprachen II, Endliche Automaten".

(Translator's remark: English title would be "Automata theory and formal languages II, finite automata".)

But as the book has been completely reworked, it may really be taken as a completely new one.

While in the first edition only the theory of finite automata has been treated, in this edition also an introduction into the theory of context-free languages is given. This was only possible in the available space frame because of an automata-theoretic treatment of the theory.

Such a foundation of the theory has already been proposed by Goldstine in 1977 and has been sketched in various talks. The motivation for developing my lecture (from which this book originates) in that way is however not based on his proposal. It almost automatically arose from dealing with the works of the French school.

(Translator's remark: With the "French school" the works of Schützenberger, Nivat, Perrot, Sakarovitch, Berstel et al. are referred to.)

I must emphasize here the book by Jean Berstel on transductions. Mr. Berstel finally pointed my to the work of Goldstine. I fully support Goldstine's opinion that it would be worth rethinking the whole theory of formal languages along this automata-theoretic lines.

This book is only an introduction into the theory of formal languages. The interested reader who wants to gets a deeper understanding of the theory or who wants to get a different look into it is pointed to the books by Ginsburg, Harrison or Salomaa. Relations to applications can be found in books on compiler design.

Dr. Klaus Estenfeld worked out my lecture "Formal Languages I" which I held on this topic in the winter semester of 1980/81 to become the foundation of this book and he made a number of additions at some places.

Dipl.-Math. Bernd Becker carefully read the manuscript and contributed with his proposals to the success of this book.

The publisher as well as the editors of the series earn our thanks for their patience of waiting for the second edition.

Saarbrücken, August 1981

Günter Hotz

Introduction

There are several reasons for the interest in the theory of formal languages in computer science. Practical problems as they arise in the context of the specification and translation of programming languages find an exact description in the theory of formal languages and so get accessible to an exact treatment. Generation processes definable by formal languages can be interpreted as non-deterministic automata which is as a generalized notion of a computer.

These kinds of generalizations in general are easier to understand than deterministic algorithms which contain more details that do not reflect the original problem but the necessity to unambiguosly define the algorithm. This is part of the reason of the difficulty to prove the correctness of programs in an understandable way. On the other side, the proof of correctness for grammars or other mechanisms for generating languages offers the possibility to study correctness proofs on simpler objects.

The theory of formal languages in this respect contains the theory of algorithms but most often only the theory of **context-free** languages is treated because of her extraordinary simplicity and beauty.

In the light of the theory stand different methods for defining formal language classes, studying their word and equivalence problems and putting them into different hierarchical classifications.

The generation processes themselves become objects of interest in the theory because the generation process of a language in case of programming languages is related to the semantics of the programs.

Of course, in the context of such a pocket book we have to make a strong selection of topics concerning language classes, generation processes as well as basic questions. In doing so, we let us guide by the intention to keep the formal machinery rather small.

Because the theory of finite automata is the foundation for the whole theory of formal languages, we start our book with this topic. In developing this theory we do not consider the technical realization of finite automata by logical circuits and binary storage devices but rather focus on the basic algorithm however it is realized.

Our intuitive notion of finite automaton consists of a finite, oriented graph whose edges are labeled with the symbols from the input alphabet of the automaton. Depending on the input word we look for a path in the graph labelled with that word. If the end point of such a path, originating from the dedicated "start point" of the automaton, is a member of the set of "end points", our automaton "accepts" the word and doesn't do otherwise.

We prove the equivalence of this concept with the other known methods of defining finite automata. We prove the usual closure properties of languages defined by finite automata. Additionally we investigate the relation between deterministic and non-deterministic automata and also 2-way automata.

It is possible to generalize this theory in the direction of considering not only the free monoid of strings (words) over a finite alphabet but also arbitrary monoids.

By considering finite automata with output, which means to attach a second label at the graph edges, we get the theory of **rational transducers**. An extensive treatment of the theory of general transductions can be found in the book by Berstel.

Here we restrict ourselves to some special generalizations of the free monoid (of words), namely the **free group**, the **H-group** (where the relation $xx^{-1} = 1$ holds for x from the generating system, but not $x^{-1}x = 1$) and the **polycyclic monoid** (in addition to $xx^{-1} = 1$ it holds $xy^{-1} = 0$ for $x \neq y$ and $0x = x0 = 0$ for x, y from the generating system).

By investigating the transductions from free monoids into the polycyclic monoids one gets a smooth transition from the theory of finite automata into the theory of context-free languages.

The corresponding construction of the theory of context-free languages leads to a simple path to the most important representation theorems. This includes the theorems of Chomsky-Schützenberger, Shamir and Greibach. Also for the transformation into Greibach normal form we get a simple and efficient algorithm.

In the same easy way as for finite automata one can prove the known closure properties for context-free languages.

In the end we also prove the equivalence of this representation with the usual representation of context-free languages using context-free grammars.

Our development of the theory is very close to the one repeatedly recommended by Goldstine since 1977, but its origins are independent from him. The difference is that we prove Greibach's representation theorem by making our automaton deterministic, namely by switching from output monoids to **monoid rings**. Doing that you get the theorem of Shamir in a natural way and from this the theorem of Greibach.

From the theorem of Shamir you can get quite easily the algorithm of Valiant for deciding the word problem of context-free languages. Because of lack of space this could not be included into this book, the same holds for the treatment of the deterministic languages.

We want to emphasize another advantage of this development of the theory: As known, the exact formalization of the notion of "derivation" when using grammars brings some difficulties. In our theory, the "derivation tree" corresponds to a path in our graph.

Maybe the use of non-free monoids is initially a problem for readers not used to it. But it seems to be the case that defining context-free languages in that way supports the intuition. For example, the usage of "syntax diagrams" for the definition of programming languages gives some evidence for this.

Because we judge the former as rather important, we want to explain it on a specific example, namely the so-called **Dyck language**.

The **Dyck language** $D(X_k)$ contains the correctly nested bracket sequences over k different pairs of brackets, where $k \in \mathbb{N}$.

A formal definition of $D(X_k)$ is as follows:

Let $X_k = \{x_1, \dots, x_k\}$ be an alphabet of k elements. Define $\bar{X}_k = \{\bar{x}_1, \dots, \bar{x}_k\}$ such that $\bar{x}_i$ is regarded as the closing bracket for x_i.

Then it holds:

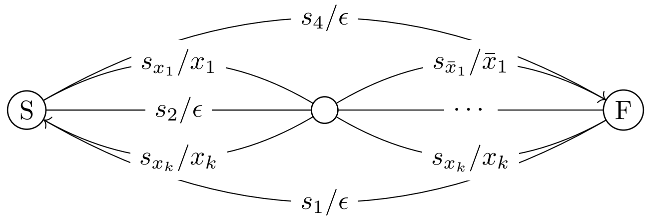
(1) $\epsilon \in D(X_k)$

(2) $u, v \in D(X_k) \Rightarrow u \cdot v \in D(X_k)$

(3) $u \in D(X_k) \Rightarrow x_i \cdot u \cdot \bar{x}_i \in D(X_k), \quad \forall i = 1, \dots, k$

(4) $D(X_k)$ is minimal with (1), (2) and (3).

For $D(X_k)$ we get the following syntax diagram:



If we consider all labellings of paths from S to F we get of course also words not contained in $D(X_k)$, for example $x_1 x_2 \bar{x}_k$ or $x_1 \bar{x}_1 \bar{x}_2$ etc.

We have to guarantee that we get Dyck words only. To do that, we define a homomorphism from the path category of the graph into the polycyclic monoid over $X_k \cup \bar{X}_k$ such that the homomorphic images of the paths from S to F have a special form, for example they have to be equal to the unit of the polycyclic monoid.

Let us consider the word

$$x_1 x_2 \bar{x}_2 \bar{x}_1 x_2 \bar{x}_2 \in D(X_2),$$

then we have different paths

$$s_{x_1} s_2 s_{x_2} s_{\bar{x}_2} s_3 s_{\bar{x}_1} s_1 s_{x_2} s_{\bar{x}_2}$$

and

$$s_{x_1} s_2 s_{x_2} s_{\bar{x}_2} s_3 s_{\bar{x}_1} s_3 s_2 s_{x_2} s_{\bar{x}_2}$$

which both have this word as their label and we can easily define a homomorphism in the sense above.

We get different paths in our graph leading to the acceptance of the same word.

The problem to construct a graph such that for each word in the accepted language exactly one path exists leads to the existence of the **deterministic** finite automaton with storage.

(Translator's remark: This type of automaton is also known as "monoid-automaton" and seems to get more attention nowadays. Some results from this book have also been redetected, see for example [**?**], [**?**].)

CHAPTER 1

Mathematical Foundations

1. Notations, basic terminology

In this first section we want to define the elementary terminology that is used throughout the whole book. We use the usual notions

$$\mathbb{N} = \{0, 1, 2, \ldots\} \text{ for the natural numbers}$$

$$\mathbb{Z} = \{\ldots, -2, -1, 0, 1, 2, \ldots\} \text{ for the integer numbers}$$

$$\mathbb{Q} = \{\frac{a}{b} \mid a, b \in \mathbb{Z}, b \neq 0\} \text{ for the rational numbers}$$

For the set operations we use $\cup$ for the union and $\cap$ for the intersection. Also $A \subset B$, $a \in A$, $a \notin A$, $\bar{A}$, $A - B$, $A \times B$ and $\emptyset$ have their usual meaning. For the power set of a set A we write 2^A or $Pot(A)$. $Card(A)$ denotes the cardinality of A. Logical implication is denoted by $\Rightarrow$.

Mappings are denoted as $f : A \to B$, in which case f is a total mapping. We write $Q(f) = A, Z(f) = B$.

(Translator remark: Q stands for "Quelle" (source) and Z for "Ziel" (target).)

If $f : A \to B$, $g : B \to C$ are mappings, then $f \circ g : A \to C$ is the composed mapping which one gets by applying ***f* first and then *g*** , i.e.

$$(f \circ g)(a) = g(f(a))$$

If $f : A \to B$ and $C \subset A$, then $f(C) = \{f(c) \mid c \in C\}$.

(Translator remark: Note that this is the other way round as the usual definition of function composition but makes perfect sense here in the category theoretic treatment.)

A subset $R \subset A \times B$ is called a **relation** between A and B. $R_f \subset A \times B$ with $R_f = \{(a, b) \mid f(a) = b\}$ is the relation **induced by the mapping** f or the **graph** of f.

Let $f : A \to B$ be a mapping, $A_1 \subset A$ and $g : A_1 \to B$ a mapping. f is called the **continuation** of g if $f(a_1) = g(a_1), a_1 \in A_1$. In this case we also write $f\mid_{A_1} = g$ (in words: f **restricted to** A_1).

A **semi-group** consists of a set M and an associative operation on that set, usually denoted as a multiplication. If a semi-group is commutative, we also use "+" instead of "·".

A semi-group is a if M contains a neutral element. We often denote it with 1_M or shortly 1. In the commutative case we often write 0 instead of 1.

For $A, B \subset M$ we denote by $A \cdot B = \{ab \mid a \in A, b \in B\}$ the **complex product** of A and B.

$A \subset M$ is a **submonoid** of M if the following holds: $1_M \in A$ and A is closed under the operation of M.

For a set A, the set A^* defined as follows, is the smallest submonoid of M which contains A.

More specific,

$$A^* = \bigcap_{M' \in M(A)} M'$$

where

$$M(A) = \{M' \subset M \mid M' \text{ is a submonoid of } M, A \subset M'\}$$

It is easy to see that

$$A^* = \bigcup_{n\geq 0} A^n \text{ with } A^0 = \{1\} \text{ and } A^{n+1} = A^n \cdot A$$

In the same sense the notion $A^+ = A^* - \{1\}$ is defined for semi-groups. A is called the **generation system** of A^* and A^+ resp.

A special meaning for us is assigned to the set of "words" (string) over a fixed alphabet A. We understand as words the finite sequences of elements from the alphabet A as for example (a, b, d, a, c) for alphabet $A = \{a, b, c, d\}$.

We define

$$WORD(A) := \{\epsilon\} \cup A \cup (A \times A) \cup (A \times A \times A) \cup \dots$$

as the **set of words (strings)** over A. The symbol ϵ denotes the **empty word** over A, that is $A^0 = \{\epsilon\}$.

If $v, w \in WORD(A)$ then $v \cdot w$ is the word which you get by concatenating v and w, formally:

$$v = (a_1, \dots, a_k), w = (a_{k+1}, \dots, a_n) \Rightarrow v \cdot w = (a_1, \dots, a_n)$$

With this operation, $WORD(A)$ becomes a monoid which is usually also denoted with A^*.

This is slightly inconsistent because for the first definition of the $*$-operator it holds $(A^*)^* = A^*$ but for the second usage of the $*$-operator it holds $(A*)^* \neq A^*$.

The following example should clarify that:

Let $A = \{a, b, c\}$ and let (a, b, a) and $(b, a) \in A^*$.

$$(a, b, a) \cdot (b, a) = (a, b, a, b, a) \in A^*, \text{ but}$$
$$((a, b, a), (b, a)) \in (A^*)^* \text{ but } \notin A^*$$

Instead of (a) we write just a. In this sense it holds $A \subset A^*$. This also holds in the sense of the first definition of A^*.

If $w = (w_1, \dots, w_n)$ we call $|w| := n$ the **length** of w. Obviously it holds:

$$|w \cdot v| = |w| + |v| \quad \text{and} \quad |\epsilon| = 0$$

The **mirror word** w^R of a word $w = (w_1, \dots, w_n)$ is the word $(w_n, \dots, w_1)$. It holds: $(w \cdot v)^R = v^R \cdot w^R$ and $\epsilon^R = \epsilon$.

In A^* the **reduction rules** hold, i.e.

(1) $w \cdot x = w \cdot y \Rightarrow x = y$

(2) $x \cdot w = y \cdot w \Rightarrow x = y$

We define **left** and **right quotient** for sets of words X, Y:

$$X^{-1} \cdot Y = \{w \mid \exists x \in X,\ y \in Y \text{ with } x \cdot w = y\}$$

and

$$X \cdot Y^{-1} = \{w \mid \exists x \in X,\ y \in Y \text{ with } w \cdot y = x\}$$

Because of the reduction rules it holds:

$\{w\}^{-1} \cdot \{v\}$ and $\{w\}^{-1} \cdot \{v\}$ are either empty or contain a single element.

If $\{w\}^{-1} \cdot \{v\}$ is not empty, we call w a **prefix** of v. If $\{w\} \cdot \{v\}^{-1} \neq \emptyset$, we call v a **suffix** of w.

In the future we will always write just w instead of $\{w\}$ and also w **is prefix of** v, if $w^{-1} \cdot v \neq \emptyset$.

2. Monoid homomorphisms and congruence relations

Definition 2.1 (monoid homomorphism). *A* **monoid homomorphism** *(short: homomorphism) from a monoid M to a monoid S is a mapping $\Phi : M \to S$ with the following properties:*

(1) $\Phi(m_1 \cdot m_2) = \Phi(m_1) \cdot \Phi(m_2), \quad m_1, m_2 \in M$

(2) $\Phi(1_M) = 1_S$

It can be easily shown: if $M_1 \subset M$ is a submonoid of M, then $\Phi(M_1)$ is a submonoid of S. If S_1 is a submonoid of S, then $\Phi^{-1}(S_1)$ is a submonoid of M.

A monoid homomorphism $\Phi : M \to S$ is called

monomorphism: if Φ is injective

epimorphism: if Φ is surjective

isomorphism: if Φ is bijective

Homomorphisms $\Phi : M \to M$ are called **endomorphisms**, isomorphisms $\Phi : M \to M$ are called **automorphisms**.

Monoids M and S are called **isomorphic**, if there exists an isomorphism between M and S.

Of course, a homomorphism cannot be defined arbitrarily on a monoid M. Thus the following two questions arise:

(1) If $M_1 \subset M$ is a submonoid and $\Phi_1 : M_1 \to S$ is an arbitrary mapping. When is it possible to extend Φ_1 to a homomorphism $\Phi : M \to S$?

(2) If Φ_1, Φ_2 both are homomorphisms from M to S which coincide on $M_1 \subset M$. In which way can Φ_1 and Φ_2 be different?

The answer to this question of course depends on the structure of M_1. If $M_1 = \{1_M\}$ then Φ is determined uniquely on M_1 but there is little information on the relation between Φ_1 and $\Phi 2$.

The following two simple theorems which can be found in introductory algebra books are holding:

(1) If M_1 is a generating system of M and $\Phi_1, \Phi_2 : M \to S$ both are monoid homomorphisms which coincide on M_1, then $\Phi 1 = \Phi_2$.

(2) If A is a set and $M = A^*$, and $\Phi_1 : A \to S$ is an arbitrary mapping, then there exists exactly one continuation Φ from Φ_1 which is a monoid homomorphism from A^* to S.

Definition 2.2 (free generating system, free monoid). *A subset $A \subset M$ is called a* **free generating system** *of M, if each mapping $\Phi_1 : A \to S$, where S is an arbitrary monoid, can be continued to a monoid homomorphims in a unique way. A monoid with a free generating system is called a* **free monoid**.

A^* therefore is a free monoid and A is a free generating system of A^*.

It holds also: If A is a free generating system of M and A^* is the monoid of words (string) over A, then A^* and M are isomorphic.

A free monoid has at most one free generating system. From that we can see that the length $|w|$ of a word $w \in A^*$ can be defined in a unique way for any free monoid.

The length mapping L is an example for a monoid homomorphism $L : A^* \to \mathbb{N}$.

If $\Phi : M \to S$ is a monoid homomorphism, then the sets

$$\{\Phi^{-1}(s) \mid s \in S\} \subset Pot(M)$$

form a monoid isomorphic to $\Phi(M)$.

We want to handle now the following question:

Let M be a monoid, $L \subset M$ be any subset of M. Does there exist a monoid S and a homomorphism $\Phi : M \to S$ with the following property: There exists an $s \in S$ with $L \subset \Phi^{-1}(s)$?

Of course, there always exists such an S: Choose $S = \{1\}$ and $\Phi(M) = \{1\}$.

Therefore we strengthen our task: Find S and Φ such that $L \subset \Phi^{-1}(S)$ and for each other homomorphism Ψ with that property holds: $L \subset \Psi^{-1}(S') \Rightarrow \Phi^{-1}(S) \subset \Psi^{-1}(S')$.

We want to describe L as close as possible by a monoid homomorphism.

Such an S and Φ exists for each $L \subset M$ (see Algebra text), it is named $synt_M(L)$ an is constructed as follows:

Definition 2.3 (syntactic congruence). *Let M be a monoid and $L \subset M$. For $a, b \in M$ we define*

$$a \equiv b\ (L) \Leftrightarrow \textit{for all } u, v \in M : u \cdot a \cdot v \in L \Leftrightarrow u \cdot b \cdot v \in L$$

$\equiv (L)$ is a congruence relation, it holds:

(1) Let $[a]_L = \{b \in M \mid a \equiv b\ (L)\}$ then $b \in [a]_L \Rightarrow [a]_L = [b]_L$

(2) If we define $[a]_L \cdot [b]_L := [ab]_L$ (complex product), then

$$synt_M(L) = \{[a]_L \mid a \in M\}$$

becomes a monoid and the mapping

$$\Psi_L : M \to synt_M(L),\ \Psi_L(a) = [a]_L$$

is a monoid epimorphism.

We call $\equiv (L)$ the **syntactic congruence** of L and $synt_M(L)$ the **syntactic monoid** of L wrt. M.

To motivate the name "syntactic monoid" we give an example from German language. Let A be the alphabet of German and L the set of sentences in German. One can denote two words w_1 and w_2 as congruent if they can always be exchanged in each german sentence. There exist words that cannot always be exchanged. In the sentence "Apfel ist eine Kernfrucht" the word "Apfel" can be exchanged by "Birne" but this is not possible in the sentence "Apfel schreibt sich A p f e l".

The difficulty is of semantic nature. If you don't consider semantic correctness of sentences you get a classification of words wrt. their syntactic meaning.

The important notion of "syntactic congruence" has been introduced by M. P. Schützenberger in the context of coding problems.

3. Special monoids and the free group

We have just learned about the syntactic monoid as an example for a monoid. Further information on the theory of syntactic monoids can be found in [**?**] and [**?**].

Let's have a look at more special monoids which we will need again later. To do so, we introduce the notion of **generated congruence relation**.

Let A be an alphabet and $R = \{u_i = v_i \mid i = 1, \dots, n,\ u_i, v_i \in A^*\}$ a set of equations.

Then by the following conditions an congruence relation $\bar{R}$ is uniquely determined:

(1) $\{(u_i, v_i) \mid u_i = v_i \in R\} \subset \bar{R}$

(2) $\bar{R}$ is a congruence relation

(3) $\bar{R} \subset R'$ for all R' fulfilling conditions 1) and 2).

$\bar{R}$ is called the **congruence relation generated by** R over A^*.

The factor monoid $A^*/\bar{R}$ is named also simply A^*/R.

It holds: Words $u, v \in A^*$ are congruent wrt. $\bar{R}$ (Notation: $u \equiv v(\bar{R})$) iff there exists $n \in \mathbb{N}, u_i \in A^*$ with $u_i = u_{i,1} \cdot u_{i,2} \cdot u_{i,3}$ such that for $i = 1, \dots, n$ it holds:

(1) $u = u_1, v = u_n$

(2) $u_{i,1} = u_{i+1,1}, u_{i,3} = u_{i+1,3}, (u_{i,2} = u_{i+1,2}) \in R$ for all $i = 1, \dots, n-1$.

We say: v is constructed from u by applying the equations from R.

The congruence classes of $u \in A^*$ in A^*/R are denoted by $[u]_{A^*/R}$ or just $[u]$.

Definition 3.1. *Let X be an alphabet. Define $X^{-1} := \{x^{-1} \mid x \in X\}$ as the set of formal inverses.*

We can think of x and x^{-1} as corresponding pairs of brackets as we did in the definition of the Dyck languages in the introduction.

We will now consider different partitionings of $(X \cup X^{-1})^*$ wrt. to different congruence relations and investigate the corresponding factor monoids.

Definition 3.2.

$$X^{[*]} := (X \cup X^{-1})^* / \{xx^{-1} = 1 \mid x \in X\}$$

is called the **H-group**.

Now we introduce a special absorbing element 0 by defining:

Definition 3.3.

$$\begin{aligned} X^{(*)} \quad := \quad & (X \cup X^{-1} \cup \{0\})^* / \\ & \{xx^{-1} = 1,\ xy^{-1} = 0,\ 0z = z0 = 0 \mid x, y \in X,\ x \neq y,\ z \in X \cup X^{-1}\ \{0\}\} \end{aligned}$$

is called the **polycyclic monoid**.

Using the naming of the previous section we get:

$$X^{(*)} = synt_{X^*}(D(X))$$

which means: the polycyclic monoid is the syntactic monoid of the Dyck language.

Definition 3.4.

$$F(X) := (X \cup X^{-1})^*/\{xx^{-1} = x^{-1}x = 1 \mid x \in X\}$$

is the **free group** over X.

Remark: It holds $D(X) = [1]_{X^{(*)}}$ and $D(X) = [1]_{X^{[*]}}$, which means the Dyck language is the set of words from $(X \cup X^{-1})^*$ which can be reduced to the empty word.

In the following we will mainly consider the H-group over X.

For $w \in (X \cup X^{-1})^*$ we define the reduced word $|w|$ as follows: If w does not contain a subword of the form xx^{-1} then $|w| = w$. Otherwise, replace the leftmost occurence of xx^{-1} by the empty word 1.

This process is called **reduction** and the result is denoted by $\rho(w)$. One can easily prove:

Lemma 3.1. *There exists a minmal number $k \in \mathbb{N}$ with $\rho^k(w) = |w|$. The number k is called the* **reduction length** *of w. It holds: $\rho(|w|) = |w|$.*

Lemma 3.2.

$$[w] = [w'] \in X^{[*]} \Leftrightarrow |w| = |w'|.$$

Proof:

"$\Leftarrow$":

It holds $w \equiv |w| = |w'| \equiv w' \Rightarrow [w] = [w']$.

"$\Rightarrow''$:

Let $[w] = [w']$. We may assume that w' is created from w by application of an equation $xx^{-1} = 1$. Let $w = w_1xx^{-1}w_2$ and $w' = w_1w_2$.

We show: If k is the reduction length of w_1 then $\rho^{k+1}(w) = \rho^k(w')$ (thus the reduced words are equal).

Proof by induction over k:

$k = 0$: w_1 is already reduced, so $\rho(w) = w_1w_2 = w'$.

$k > 0$: It holds $\rho(w) = \rho(w_1xx^{-1}w_2)$, $\quad \rho(w') = \rho(w_1w_2)$. The reduction length of $\rho(w)$ by induction proposition is $k-1$ and $\rho^k\rho(w) = \rho^{k-1}\rho(w') \Rightarrow$ the reduced word of w and w' is the same so $|w| = |w'|$.

Remark: Using the same argument one can show that the creation of the reduced word doen not depend on the order of the reductions.

Therefore the reduced word for a representant of an element of $X^{[}*]$ is unique, so we can just speak of "the" reduced word in the following.

Remark: These results have been used in [**?**] to obtain a space-optimal algorithm for the analysis of the Dyck language.

Similar results also hold for the free group $F(X)$, see [**?**].

4. Graphs, categories and functors

Before defining graphs formally, we want to describe what we mean by a graph. A graph consists of points and edges. Each edge connects two points which are not necessarily different. You can imagine a graph as streetmap, the cities are the points and the streets are the edges of the graph. The edges may be oriented such that they have a one-way direction. Paths in graphs are sequences of edges that you could drive for example with a car without violating the traffic rules.

One can show that every graph as we will formally define has, with a certain restriction, a faithful(?) image in $\mathbb{R}^3$, see [**?**]. The points of the graph are here the points in $\mathbb{R}^3$, the edges are lines in $\mathbb{R}^3$ which do not intersect pair-wise.

The mentioned restriction is that the graph must not have more points than the cardinality of $\mathbb{R}^3$. The restriction concerning the edges is more severe: It say that there is at most one edge between two points and that the graph has no loops. Loops are edges with just a single point.

From what has been said we see that we may use a concrete geometric picture of a graph without getting our intuition mistaken. The following definition of a graph nevertheless doe not contain any geometry.

Definition 4.1 (graph). *A* **graph** $G = (V, E)$ *consists of a non-empty set* V *of points (also called vertices) and a set* E *of edges and a mapping* $\rho : E- > Pot(V)$ *with* $card(\rho(e)) <= 2$ *for* $e \in E$*.* $\rho(e)$ *is the set of* border points *of* e*.*

Border points of an edge do not need to be different. If $card(\rho(e)) = 2$ we call e a **line**, if $card(\rho(e)) = 1$ we call it a **loop**.

Definition 4.2 (loop-free). *A graph is called* **loop-free** *if it does not contain a loop.*

We introduce an orientation for the edges.

Definition 4.3 (oriented graph). *A graph* $G = (V, E)$ *is called an* **oriented graph** *if there are two mappings* $Q : E \to V$ *and* $Z : E \to V$ *with* $\rho(e) = Q(e), Z(e)$ *for all* $e \in E$*.*

$Q(e)$ is called the **source** and $Z(e)$ the target point of e. The notions of loop and line are naturally transferred to oriented graphs.

For each graph one can assign the corresponding oriented graph $\hat{G}$ by defining two edges (P_1, e, P_2) and (P_2, e, P_1) for every edge e with border points P_1 and P_2 and defining $Q((P_1, e, P_2)) = P_1 = Z((P_2, e, P_1))$ and $Q((P_2, e, P_1)) = P_2 = Z((P_1, e, P_2))$.

Definition 4.4 (connected graph). *A graph* $G = (V, E)$ *is called* **connected** *if in the corresponding oriented graph* $\hat{G}$ *for each points* P *and* P' *there exist edge sequences* $e_1, \ldots, e_k$ *with* $Q(e_1) = P, Z(e_k) = P'$ *and* $Z(e_i) = Q(e_{i+1})$ *for all* $i = 1, \ldots, k-1$*.*

Definition 4.5 (ordered graph). *A loop-free graph* $G = (V, E)$ *is called* **ordered** *if for each point* $P \in V$ *holds: There exists a unique (up-to cyclic permutation) ordering on the set* $\{e \in E \mid P \in \rho(e)\}$*.*

Notation: $\{e \in E \mid P \in \rho(e)\}$ is called the **cycle** belonging to P ($cycle(P)$).

Explanation: Image each point and its adjacent edges to be stuck on a litte circle as in the following figure:

FIGURE

Definition 4.6 (ordered graph). *A loop-free, graph G is called* **ordered** *if for all points $P \in V$ it holds: There exists an ordering*

$$e_1, \ldots, e_k, e'_m, \ldots, e'_1$$

such that

$$\{e_1, \ldots, e_k\} = \{e \in E \mid Z(e) = P\}$$

and

$$\{e'_1, \ldots, e'_m\} = \{e \in E \mid Q(e) = P\}$$

$e_1, \ldots, e_k$ is called the **ordering** of the incoming edges of P and $e'_1, \ldots, e'_m$ the ordering of the outgoing edges of P.

Example:

FIGURE

Definition 4.7 (path). *A* **path** *in an oriented graph G is a sequence*

$$\pi = (Q_1, e_1, \ldots, e_k, Z_k)$$

with $k \geq 1$, $e_1, \ldots, e_k \in E$, $Q(e_1) = Q_1$, $Z(e_k) = Z_k$ and $Q(e_{i+1}) = Z(e_i)$ for $i = 1, \ldots, k-1$.

We extends the mappings Q and Z onto paths by defining $Q(w) := Q_1$ and $Z(w) := Z_k$. Q_1 is called the **start point** and Z_k the **end point** of path w.

k is the **length** of w, written as $L(w) = k$. For $k = 0$ we declare for all points $P \in V$ that $w = (P, P)$ is the path of length 0 from P to P.

Paths in arbitrary graphs are defined by switching to the oriented graph $\hat{G}$.

Definition 4.8 (subpath). *Let $w = (Q, e_1, \ldots, e_k, Z_k)$ be a path. A path $w' = (Q'_1, e'_1, \ldots, e'_m, Z'_m)$ is called a* **subpath** *of w, if it holds: $\exists i, i \leq i \leq k$ such that $e'_j = e'_{i+j-1}$, $j = 1, \ldots, m$ and $i + m - 1 \leq k$.*

A path is called **closed** if $Q(w) = Z(w)$, it is called a **circle** if it is closed and does not contain any closed subpath w' with $L(w') > 0$.

Definition 4.9 (circle-free graph). *A graph $G = (V, E)$ is called* **circle-free** *if there are no circles in G.*

For our purposes we will only consider oriented graphs. For these graphs the following definition reflects a special connectivity property.

Definition 4.10 (star, center). *Let $G = (V, E)$ be an oriented graph and $P \in V$. G is called a* **star around P** *if for each $P' \in V$ there exists a path $w_{P'}$ with $Q(w_{P'}) = P$ and $Z(w_{P'}) = P'$. P is called the* **center** *of G.*

We want to introduce now a special kind of graph that plays a central role in the theory of formal languages.

Definition 4.11 (tree). *A* **tree** *is a circle-free star where for all $P \in V$ it holds $card(\{e \in E \mid Z(e) = P\}) \leq 1$.*

The following lemma holds:

LEMMA 4.1. *A tree has exactly one center which is called the* **root**.

Historical remark: Leonard Euler (1735) at a walk in Königsberg asked himself if he could traverse each of the seven bridges over the Memel in such a way that he would traverse each bridge exactly once. In the figure below you can see a graph describing the situation. Euler gave a simple criterion for the existance of paths that traverse each edge of a graph exactly once (the so called Euler paths).

FIGURE

Now we want to concatenate paths or mathematically, define a product operation on paths. We define:

$$(Q_1, e_1, \ldots, e_k, Z_k) \cdot (Q_{k+1}, e_{k+1}, \ldots, e_n, Z_n)$$

$$:= (Q1, e_1, \ldots, e_n, Z_n) \text{ if } Q_{k+1} = Z_k$$

That means you can concatenate two paths if the end point of the first is the start point of the second path. Obviously it holds:

(1) The product of paths is associative (if defined)
(2) For each point P of a graph G there exists exactly one path $1_P := (P, P)$ such that for each path w it holds:
(3)

$$\begin{aligned} w \cdot 1_P = w, & \text{ if } Z(w) = P \\ 1_P \cdot w = w, & \text{ if } Q(w) = P \end{aligned}$$

We denote the set of paths of a graph G with $\mathcal{W}(G)$.

$\mathcal{W}(G)$ is called the **path category** of G and G in this context is also called **schema**. $\mathcal{W}(G)$ is an important special case of a category.

Notation:

$$\mathcal{W}(G)(P, P') := \{w \in \mathcal{W}(G) \mid Q(w) = p,\ Z(w) = P'\}$$

Categories are algebraic structures with a *partial* operation.

DEFINITION 4.12 (category). *$C = (O, M, Q, Z, \circ)$ is called a* **category** *if the axioms (K1) to (K4) are fulfilled:*

(K1) *O and M are sets and $Q : M \to O$ and $Z : M \to O$ are mappings. $Q(f)$ is the source of f and $Z(f)$ is the target of f, O is the set of* **objects** *and M the set of* **morphisms** *of the category C.*

(K2) *For $f, g \in M$ the operation $\circ$ is defined if $Q(g) = Z(f)$. In this case it holds $f \circ g \in M, Q(f \circ g) = Q(f), Z(f \circ g) = Z(g)$.*

(K3) *The associative law $(f \circ (g \circ h)) = (f \circ g) \circ h$ holds in the sense that each of both sides is defined if one of both is.*

(K4) *For each object $w \in O$ there exists a unit morphism $1_w \in M$ with $Q(1_w) = Z(1_w) = w$ and for all morphisms $f, g \in M$ with $Q(f) = Z(g) = w$: $1_w \circ f = f$ and $g \circ 1_w = g$. It can be easily shown that there exists exactly one unit morphism for each object w.*

Notations: $Obj(C) := O$ is the **set of objects** and $Mor(C) := M$ the **set of morphisms** of the category C.

Historical remark: Euler was already interested in graphs and paths in graphs. The path category has already been used before the notion of category even existed. The axiomatic formulation of categories and its importance for many areas of mathematics has been elaborated by S. Eilenberg and S. MacLane in 1945 [**?**]. Their work has stimulated a broad, very abstract theory of categories. We will only use the notations for structures which are categories and some elementary concepts which also in the theory of formal languages lead to fruitful questions.

We explain the notion of category on a number of examples:

(1) **The category of relations**
Define $REL(O) = (O, M, Q, Z, \circ)$ by:

- Let O be a set of sets ($O \notin O$).
- $M = \{(A, B, R) \mid A \in O, B \in O, R \subset A \times B\}$
- $Q(A, B, R) = A, \quad Z(A, B, R) = B$
- $(A, B, R_1) \circ (B, C, R_1) = (A, C, R')$
  where $R' = \{(a, c) \mid \exists b \in B : (a, b) \in R_1 \text{ and } (b, c) \in R_2\}$

With these definitions $REL(O)$ becomes a category.

(2) **The category of matrices**
Let $MAT(\mathbb{Q}) = (O.M, Q, Z, \circ)$ with

- $O = \mathbb{N}$
- $M =$ the set of $k \times n$ matrices, $k, n \in \mathbb{N}$, with entries from .
- For a $k \times n$ matrix $A_{k,n}$ define source and target mappings by

  $Q(A_{k,n} = k)$ the number of rows

  $Z(A_{k,n} = n)$ the number of columns

With the matrix multiplikation as category operation $\circ$ the set $MAT(\mathbb{Q})$ becomes a category. Units in this category are the $n \times n$ unit matrices.

Analogously to the monoid homomorphisms we introduce structure-preserving mappings between categories, named **functors**.

Definition 4.13 (functor). *Let $C_i = (O_i, M_i, Q_i, Z_i, \circ_i), i = 1, 2$ be two categories and $\phi_1 : O_1 \to O_2$ and $\phi_2 : M_1 \to M_2$ be mappings.*

$\phi = (C_1, C_2, \phi_1, \phi_2)$ is called a **functor** *from C_1 to C_2 if the axioms (F1) to (F3) hold:*

(F1) *The diagram*
$$\begin{tikzcd} O_1 \arrow[d, "\phi_1"] & M_1 \arrow[l, "Q_1"'] \arrow[r, "Z_1"] \arrow[d, "\phi_2"] & O_1 \arrow[d, "\phi_1"] \\ O_2 & M_2 \arrow[l, "Q_2"'] \arrow[r, "Z_2"] & O_2 \end{tikzcd}$$
is commutative.

(F2) $\phi_2(f \circ_1 g) = \phi_2(f) \circ_2 \phi_2(g)$ *for all $f, g \in M_1$ with $Z(f) = Q(g)$.*

(F3) $\phi_2(1_w) = 1_{\phi_1(w)}$ *for all $w \in O_1$.*

A functor ϕ is called injective (surjective, bijective) if ϕ_1 and ϕ_2 are injective (surjective, bijective).

Let's look at some examples:

Example 1: Consider the following oriented graphs G_1 and G_2:

FIGURE

$G1 = (V_1, E_1)$ represents an infinite binary tree. From each point of the tree two edges go out which are labeled with f and g.

$G_2 = (V_2, E_2)$ consists of a single point P_0 and two loops labeled with f and g respectively.

Consider the path categories $\mathcal{W}(G_1)$ and $\mathcal{W}(G_2)$. For $P \in V_1$ define $\phi_1(P) := P_0$ and $\phi_2(1_P) := 1_{P_0}$.

For an edge $e \in E_1$ we define:

$$\phi'(e) = \begin{cases} f & \text{if } e \text{ is marked with } f \\ g & \text{if } e \text{ is marked with } g \end{cases}$$

Now we define for $(P, e_1, \ldots, e_n, P') \in \mathcal{W}(G_1)$:

$\phi_2((P, e_1, \ldots, e_n, P')) = (P_0, \phi_2'(e_1), \ldots, \phi_2'(e_n), P_0)$.

Obviously $\phi = (\mathcal{W}(G_1), \mathcal{W}(G_2), \phi 1, \phi_2)$ is a functor.

It is a special functor because

(1) ϕ is surjective

(2) If P_1 is a point in G_1 and $\bar{w}$ is a path in G_2, then there exists exactly one path w in G_1 with $Q(w) = P_1$ such that $\phi_2(w) = \bar{w}$.

Example 2: Let graphs $G1, G_2$ be given as follows:

FIGURE

Then there exists a surjective functor from $\mathcal{W}(G_1)$ to $\mathcal{W}(G_2)$.

It is possible to construct surjective functors which fulfill (2) from example 1 and other surjective functors which don't.

Example 3: Let G_1 and G_2 be given as:

FIGURE

We define:

$$\begin{aligned} \phi_1(P_1) &= Q_1 \\ \phi_1(P_2) &= Q_2 \\ \phi_1(P_3) &= Q_2 \\ \phi_1(P_4) &= Q_3 \\ \phi_2((P_1, s, P_2)) &= (Q_1, f, Q_2) \\ \phi_2((P_3, r, P_4)) &= (Q_2, g, Q_3) \end{aligned}$$

For the units, the defintion of ϕ_2 is clear.

One can see that $\phi = (\mathcal{W}(G_1), \mathcal{W}(G_2), \phi_1, \phi_2)$ is a functor.

It is remarkable that $\phi_2(\mathcal{W}(G_1))$ is not a category because this set is not closed under the $\circ$ operation.

Example 4: Let G_1 and G_2 be given as follows:

FIGURE

We define:

$$\begin{aligned}
\phi_1(1) &= 1' \\
\phi_1(2) &= 2' \\
\phi_1(3) &= 3' \\
\phi_1(4) &= 3' \\
\phi_1(5) &= 3' \\
\phi_2(a) &= a' \\
\phi_2(b) &= b' \\
\phi_2(c) &= c' \\
\phi_2(d) &= d' \\
\phi_2(e) &= 1_3 \\
\phi_2(f) &= 1_3 \\
\phi_2(g) &= 1_3
\end{aligned}$$

$\phi = (\mathcal{W}(G_1), \mathcal{W}(G_2), \phi_1, \phi_2)$ is a functor.

Example 5: The graph G shall be defined by

FIGURE

Additionally, the following matrices are given:

$$a' = \begin{pmatrix} 1 & 0 & 2 & 1 \\ 0 & 1 & 2 & 5 \\ 1 & 1 & 2 & 1 \end{pmatrix} \qquad b' = \begin{pmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 2 & 2 & 2 \\ 1 & 5 & 1 \end{pmatrix} \qquad c' = \begin{pmatrix} 4 & 5 & 6 \\ 1 & 2 & 3 \end{pmatrix}$$

$$d' = \begin{pmatrix} 7 & 4 \\ 5 & 3 \\ 3 & 5 \\ 4 & 7 \end{pmatrix} \qquad e' = \begin{pmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \\ 3 & 4 & 1 & 2 \\ 1 & 0 & 0 & 0 \end{pmatrix} \qquad f' = \begin{pmatrix} 1 & 2 \\ 2 & 0 \end{pmatrix}$$

We consider $\mathcal{W}(G)$ and $MAT(\mathbb{N})$, the category of matrices over $\mathbb{N}$.

We define $\phi_1(i) = i$ for $i = 2, 3, 4$ and $\phi_2'(x) = x'$ for $x \in \{a, b, c, d, e, f\}$.

ϕ_2' can be extended in a unique way to a mapping $\phi_2 : \mathcal{W}(G) \to MAT(\mathbb{N})$ such that $\phi = (\mathcal{W}(G), MAT(\mathbb{N}), \phi_1, \phi_2)$ is a functor.

We want to define now some special properties of functors.

Definition 4.14. *Let G_1, G_2 be ordered graphs, $\phi = (\mathcal{W}(G_1), \mathcal{W}(G_2), \phi_1, \phi_2)$ a functor.*

ϕ is called **ordered** *or* **order preserving** *if it holds:*

Let $\phi_1(P) = P' \in V_2$ for any $P \in V_1$, then for the ordering $e_1, \ldots, e_k, e_m', \ldots, e_1'$ which belongs to P it holds:

$\phi_2(e_1), \ldots, \phi_2(e_k), \phi_2(e_m'), \ldots, \phi_2(e_1')$ is contained in the ordering that belongs to P' in the given order.

It is possible that lines coincide which are counted only once in that case.

Let's give an example for this definition:

Let $P \in V_1$ be a point with ordering $e_1, e_2, e_3, e_4', e_3', e_2', e_1'$ and $P' \in V_2$ be a point with ordering $r_1, r_2, r_5', r_4', r_3', r_2', r_1'$ as shown in the following figure:

FIGURE

Define ϕ by $\phi_1(P) = P'$ and

$$\phi_2(e_1) = r_1, \phi_2(e_2) = r_2, \phi_2(e_3) = r_2$$
$$\phi_2(e_1') = r_1', \phi_2(e_2') = r_3', \phi_2(e_3') = r_4', \phi_2(e_4') = r_5'$$

Then ϕ respects the ordering in point P.

DEFINITION 4.15. *Let G_1, G_2 be oriented graphs and $\phi = (\mathcal{W}(G_1), \mathcal{W}(G_2), \phi_1, \phi_2)$ be a functor.*

ϕ is called **regular** *$\Leftrightarrow$ the restriction of ϕ_2 to the set $\{e \in E_1 \mid Q(e) = P\}$ and $\{e' \in E_2 \mid Q(e') = \phi_1(P)\}$ and to $\{e \in E_1 \mid Z(e) = P\}$ and $\{e' \in E_2 \mid Z(e') = \phi_1(P)\}$ for $P \in V_1$ is bijective.*

To each incoming / outgoing edge of a point $P \in V_1$ corresponds exactly one incoming / outgoing edge of $\phi_1(P) \in V_2$.

In our example, ϕ was not regular.

We slightly weaken the definition of a regular functor by only postulating regularity on the outgoing edges.

DEFINITION 4.16. *Let G_1, G_2 be oriented graphs and $\phi = (\mathcal{W}(G_1), \mathcal{W}(G_2), \phi_1, \phi_2)$ be a functor.*

ϕ is called **out-regular** *$\Leftrightarrow$ the restriction of ϕ_2 to the set $\{e \in E_1 \mid Q(e) = P\}$ and $\{e' \in E_2 \mid Q(e') = \phi_1(P)\}$ for $P \in V_1$ is bijective.*

The following lemma holds:

LEMMA 4.2. *If $\phi = (\mathcal{W}(G_1), \mathcal{W}(G_2), \phi_1, \phi_2)$ is an out-regular functor, then $\phi(\mathcal{W}(G_1))$ is a category.*

Our next lemma describes a well-known fact from graph theory that has found many applications.

LEMMA 4.3. *To each circle-free star $G = (V, E)$ relative to a point P there exists a tree B and an out-regular functor $(\mathcal{W}(B), \mathcal{W}(G), \phi_1, \phi_2)$ mapping the root of the tree B to the point P. B is determined up to isomorphisms.*

5. Subcategory, generating system

Definition 5.1. *Let*

$$U = (Obj(U), Mor(U), Q_U, Z_U, \circ_U)$$

and

$$C = (Obj(C), Mor(C), Q_C, Z_C, \circ_C)$$

be categories.

U *is called a* **subcategory** *of* $C \Leftrightarrow$

(1) $Obj(U) \subset Obj(C)$ *and* $Mor(U) \subset Mor(C)$
(2) $Q_U = Q_C|_{Mor(U)}$ *and* $Z_U = Z_C|_{Mor(U)}$
(3) $\circ_U = \circ_C|_{Mor(U)\times Mor(U)}$
(4) *For* $w \in Obj(U) \Rightarrow 1_w \in Mor(U)$

U is called **full subcategory** of $C \Leftrightarrow$

$$\forall w_1, w_2 \in Obj(U), f : w_1 \to w_2 \in Mor(C) \Rightarrow f \in Mor(U)$$

This means, all morphisms in C between objects in U are also morphisms in U. $f : w_1 \to w_2$ stands for $Q(f) = w_1 \wedge Z(f) = w_2$.

We want to explain this fact at some examples:

Example 1:

Let $A = \{x, y, z, a, b, c\}$ and $f, g, h : A^* \to A^*$ be mappings defined as follows:

$$f(u_1 \cdot x \cdot u_2 \cdot x \cdots x \cdot u_k \cdot x) = u_1 \cdot ax \cdot u_2 \cdot ax \cdots ax \cdot u_k \cdot ax)$$

where $u_i \in (A - \{x\})^*$,

$$g(u_1 \cdot y \cdot u_2 \cdot y \cdots y \cdot u_k \cdot y) = u_1 \cdot by \cdot u_2 \cdot by \cdots by \cdot u_k \cdot by)$$

where $u_i \in (A - y\})^*$,

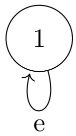
$$h(u_1 \cdot z \cdot u_2 \cdot z \cdots z \cdot u_k \cdot z) = u_1 \cdot cz \cdot u_2 \cdot cz \cdots cz \cdot u_k \cdot cz)$$

where $u_i \in (A - y\})^*$.

Let M be the monoid of mappings generated by $f, g, h : A^* \to A^*$.

Then $C = (A^*, M, Q, Z, \circ)$ is a category, if $\circ$ denotes the monoid operation in M.

Let G be the graph defined by the following figure:



We define a functor $\phi = (\mathcal{W}(G), C, \phi_1, \phi_2)$ by $\phi_1(1) := A^*$ and $\phi_2(e) := f \circ g \circ h$.

$\phi_2(\mathcal{W}(G))$ is a subcategory of C and it holds:

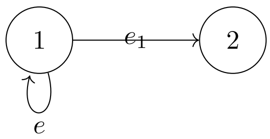
$$\phi_2(\mathcal{W}(G))(xyz) = \{(a^n x b^n y c^n z) \mid n \in \mathbb{N}\}$$

Example 2:

We expand on example 1. In addition to f, g, h we have three monoid homomorphisms f_1, g_1, h_1 defined by:

$$f_1(x) = \epsilon, f_1(u) = u \quad \forall u \in A - \{x\}$$
$$g_1(y) = \epsilon, g_1(u) = u \quad \forall u \in A - y\}$$
$$h_1(z) = \epsilon, h_1(u) = u \quad \forall u \in A - z\}$$

We extend the graph G as follows to a graph G_1:



Consider $\mathcal{W}(G_1)(1,2)$. Then $\mathcal{W}(G_1)(1,2) \cup \{1_1, 1_2\}$ is a subcategory of $\mathcal{W}(G_1)$. In addition, $\mathcal{W}(G)$ is a subcategory of $\mathcal{W}(G_1)$.

We extend the functor ϕ from example 1 onto $\mathcal{W}(G_1)$ by defining:

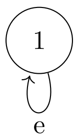
$$\phi_2(e_1) := f_1 \circ g_1 \circ h_1$$

We get:

$$\phi_2(\mathcal{W}(G_1)(1,2))(xyz) = \{(a^n b^n c^n \mid n \in \mathbb{N}\}.$$

Example 3:

Let G be defined as follows:



The full subcategory of $\mathcal{W}(G)$ generated by $\{1', 2', 3'\}$ is the path category $\mathcal{W}(G')$ with the following graph G':

FIGURE

As an exercise, one can show: The mapping ϕ_1 defined by

$$\begin{array}{rcl}
\phi_1(1) & = & 1' \\
\phi_1(4) & = & 1' \\
\phi_1(2) & = & 2' \\
\phi_1(5) & = & 2' \\
\phi_1(3) & = & 3' \\
\phi_1(6) & = & 3'
\end{array}$$

can be extended to a functor $\phi = (\mathcal{W}(G), \mathcal{W}'(G'), \phi_1, \phi_2)$ where ϕ_2 has to be chosen in a suitable way.

Remark: The preimage of a closed path does not have to be closed.

We prove now the following

LEMMA 5.1. *Let*

$$C_i = (O_i, M_i, Q, Z, \circ),\ i = 1,2,3$$

be categories and C_1 and C_2 be subcategories of C_3. Then $C_1 \cap C_2$ is a category.

Proof: It holds

(1) $w \in O_1 \cap O_2 \Rightarrow 1_w \in M_1 \cap M_2$

(2) $f, g \in M_1 \cap M_2 \Rightarrow f \circ g \in M_1 \cap M_2$, if $Z(f) = Q(g)$

It follows that $C_1 \cap C_2$ is a category.

LEMMA 5.2. *Let $C_i = (O_i, M_i, Q, Z, \circ),\ i \in I$, where I is an arbitrary index set, be categories. If $C_I, i \in I$, are subcategories of a category C, then*

$$\tilde{C} := \bigcap_{i \in I} C_i$$

is a category.

The proof is similar as the one of the previous lemma.

DEFINITION 5.2 (generated subcategory). *Let $C = (O, M, Q, Z, \circ)$ be a category and $O_1 \subset O, M_1 \subset M$ and*

$$\mathcal{U}_C(O_1, M_1) := \{C' \mid C' \textit{ is subcategory of } C, O_1 \subset O', M_1 \subset M'\}.$$

Then

$$<O_1, M_1> := \bigcap_{C' \in \mathcal{U}_C(O_1, M_1)} C'$$

is called the subcategory of C generated by (O_1, M_1) and (O_1, M_1) is called the generating system of $<(O_1, M_1)>$.

Obviously for each category $C = (O, M, Q, Z, \circ)$ it holds: $C = <O, M>$.

We say M_1 "generates" $<O_1, M_1>$, if

$$O_1 = \{Q(m) \mid m \in M_1\} \cup \{Z(M) \mid m \in M_1\}.$$

We have already seen an example for a nontrivial generating system.

Let $G = (V, E)$ be a graph, then E is a generating system of $\mathcal{W}(G)$ which means $\mathcal{W}(G) = <E(G)>$. The path category of a graph has a special property namely that $E(G)$ is a **free** generating system of $\mathcal{W}(G)$.

DEFINITION 5.3 (free generating system). *Let $C = (O, M, Q, Z, \circ)$ be a category and $E \subset M$. E is called a* **free generating system** *of C, if the following holds:*

If $C' = (O', M', Q, Z, \circ)$ is an arbitrary category and $\phi_1 : O \to O'$ and $\phi_s : E \to M'$ are mappings which fulfill the following diagram:

$$\begin{tikzcd}
O \arrow[d, "\phi_1"] & E \arrow[l, "Q"'] \arrow[r, "Z"] \arrow[d, "\phi_2'"] & O \arrow[d, "\phi_1"] \\
O' & M' \arrow[l, "Q"'] \arrow[r, "Z"] & O'
\end{tikzcd}$$

Then there exists a unique continuation of ϕ_2' to $\phi_2 : M \to M'$ such that $\phi = (C, C', \phi_1, \phi_2)$ is a functor.

DEFINITION 5.4 (free category). *A category C is called* **free** *if there exists a free generating system E of C.*

We formulate now our observation above as a theorem:

THEOREM 5.1. *Let $G = (V, E)$ be a graph. Then E is a free generating system of $\mathcal{W}(G)$.*

Proof: Let $G = (V, E)$ be a graph and C an arbitrary category. Let $\phi_1 : E \to O$, $\phi_2' : E \to M$ be mappings and the following diagram commute:

$$\begin{tikzcd} V \arrow[d, "\phi_1"] & E \arrow[l, "Q"'] \arrow[r, "Z"] \arrow[d, "\phi_2'"] & V \arrow[d, "\phi_1"] \\ O & M \arrow[l, "Q"'] \arrow[r, "Z"] & O \end{tikzcd}$$

We define

$$\phi_2(P, P) = 1_{\phi_1(P)}, P \in V$$

and

$$\phi_2(e) = \phi_2'(e), e \in E$$

Let $\phi_2(w)$ be defined for all $w \in \mathcal{W}(G)$ with $|w| \leq n, n \geq 1$ and ϕ_2 be compatible with Q and Z for all these paths w.

Further let ϕ_2 be uniquely determined for these paths w and it holds:

$$\phi_2(w \cdot v) = \phi_2(w) \cdot \phi_2(u)$$

for all w, v with $|w \cdot v| \leq n$.

Now let $v = (P, s_1, \ldots, s_{n+1}, P') \in \mathcal{W}(G)$. We split v into

$$v = (P, \underbrace{s_1, \ldots, s_n}_{v_1}, P'') \cdot (P'', \underbrace{s_{n+1}}_{v_2}, P')$$

.

By induction hypothesis, $\phi_2(v_1)$ and $\phi_2(v_2)$ are defined.

For ϕ_2 to become a functor, necessarily $\phi_2(v) = \phi_2(v1) \cdot \phi_2(v_2)$ must hold.

By assumption, ϕ_2 is compatible with source and target mappings Q and Z for v_1 and v_2. Therefore $Z(\phi_2(v_1)) = Q(\phi_2(v_2))$ and $\phi_2(v)$ is defined.

Let $v = u_1 \cdot u_2$ be any partition of v, so

$$v = (P, \underbrace{s_1, \ldots, s_j}_{u_1}, \bar{P}) \cdot (\bar{P}, \underbrace{s_{j+1}, \ldots, s_{n+1}}_{u_2}, P').$$

By induction hypothesis it holds:

(1) $Z(\phi_2(u_1)) = Q(\phi_2(u_2))$, so $\phi_2(u1) \cdot \phi_2(u_2)$ is defined.

(2) With $u_2' = (\bar{P}, s_{j+1}, \ldots, s_n, P'')$ we have

$$\begin{aligned} \phi_2(u_1) \cdot \phi_2(u_2) &= \phi_2(u_1) \cdot (\phi_2(u_2') \cdot \phi_2(v_2)) \\ &= (\phi_2(u_1) \cdot \phi_2(u_2')) \cdot \phi_2(v_2) \\ &= \phi_2(v_1) \cdot \phi_2(v_2) \\ &= \phi_2(v) \end{aligned}$$

It remains to show

$$\phi_1(Q(v)) = Q(\phi_2(v)), \quad \phi_1(Z(v)) = Z(\phi_2(v))$$

This follows directly from $Q(v) = Q(v_1)$ and $Z(v) = Z(v_2)$ by induction hypothesis.

THEOREM 5.2. *To each category C there exists a free category F and a surjective functor $\phi = (F, C, \phi_1, \phi_2)$.*

Proof: For the category C, create the oriented graph $G_C = (V, E)$ with $V = Obj(C)$ and $E = \{f \mid Q(f) = O_1, Z(f) = O_2, f : O_1 \to O_2 \in M\}$.

That means: The objects of the category become the points of the graph and each morphism becomes an edge between the corresponding source and target objects. Because $\mathcal{W}(G_C)$ is a free category by theorem 1 we can choose F to be exactly this category.

Let $\phi_1 : V \to O$ with $\phi_1(w) = w$ and $\phi_2' : E \to M$ with $\phi_2'(f) = f$ be mappings and $\phi_2 : \mathcal{W}(G_C) \to M$ be the continuation of ϕ_2' such that $\phi = (\mathcal{W}(G_C), C, \phi_2, \phi_2)$ becomes a functor.

By construction, ϕ is surjective.

The following theorem tells about the uniqueness of free generating systems.

THEOREM 5.3. *If E and E' are free generating systems of a category F, then $E = E'$.*

The proof is similar to the one of the corresponding theorem for free monoids.

6. Grammars and derivations

In the introduction of this book we already learned about formal languages. The wish to describe these in general infinite sets of words by a finite generating system leads to the notion of a **grammar**.

DEFINITION 6.1 (Chomsky grammar). *$G = (N, T, P, s)$ is a* **Chomsky grammar***, if*

(1) *N is a finite, nonempty set of* **nonterminal symbols**

(2) *T is a finite, nonempty set of* **terminal symbols** *with $N \cap T = \emptyset$*

(3) *$P \subset N^+ \times (N \cup T)^*$ is a finite set of* **productions**

(4) *$S \in N$ is the* **axiom** *or* **start symbol**

Notation: For $p = (u, v) \in P$ we also write $u \xrightarrow{p} v$ and $Q(p) = u, Z(p) = v$ denote the source and target of a production.

Examples:

(1) $G_1 = (N, T, P, S)$ with $N = \{S\}, T = \{x, x'\}$,
$P = \{S \to SS, S \to xSx', S \to \epsilon\}$

(2) $G_2 = (N, T, P, S)$ with $N = \{S, X\}, T = \{x, x'\}$
$P = \{S \to xSX, S \to xX, X \to x'S, X \to x'\}$

To use a grammar for generating the words of a language, starting with the axiom there are intermediate words generated by application of the productions, until the produced word will contain terminal symbols only. This leads to the notation of a "derivation" which we will now define formally.

DEFINITION 6.2 (directly derivable, derivable). *Let $G = (N, T, P, S)$ be a grammar and let $w, w' \in (N \cup T)^*$.*

w' is **directly derivable** *from w in G, notation: $w \underset{G}{\Rightarrow} w'$, if there are segmentations $w = w_1 \cdot u \cdot w_2$ and $w' = w_1 \cdot v \cdot w_1$ and a production $(u, v) \in P$.*

w' is **derivable** *from w, notation: $w \overset{*}{\underset{G}{\Rightarrow}} w'$, if there exists a sequence of words*

$$w = w_0, \ldots, w_n = w', \quad n \in \mathbb{N}, w_i \in (N \cup T)^*$$

such that for each $0 \leq i \leq n : w_i \underset{G}{\Rightarrow} w_{i+1}$.

Such a sequence is called a **derivation** of length n. Q and Z can be extended in a natural way to derivations.

A derivation is called **canonic** or **leftmost**, if in each step $w \underset{G}{\Rightarrow} w'$ it holds:

If $w = w_1 \cdot u \cdot w_2$ and $w' = w_1 \cdot v \cdot w_2$ are the segmentations and $(u, v) \in P$ the applied production, then $w_1 \in T^*$ which means that always the leftmost nonterminal is replaced.

If the grammar is known we omit the index G from the symbols $\underset{G}{\Rightarrow}$ and $\overset{*}{\underset{G}{\Rightarrow}}$.

Let us consider some properties of the relation $\overset{*}{\underset{G}{\Rightarrow}}$:

LEMMA 6.1. *Let G be a grammar. Then the following holds:*

(1) $(u, v) \in P \Rightarrow u \overset{*}{\underset{G}{\Rightarrow}} v$

(2) $w \overset{*}{\underset{G}{\Rightarrow}} w$ *(reflexivity)*

(3) $w \overset{*}{\underset{G}{\Rightarrow}} w' \wedge w' \overset{*}{\underset{G}{\Rightarrow}} w'' \Rightarrow w \overset{*}{\underset{G}{\Rightarrow}} w''$ *(transitivity)*

(4) $w_1 \overset{*}{\underset{G}{\Rightarrow}} w_1' \wedge w_2 \overset{*}{\underset{G}{\Rightarrow}} w_2' \Rightarrow w_1 \cdot w_2 \overset{*}{\underset{G}{\Rightarrow}} w_1' \cdot w_2'$ *(compatibility with monoid operation)*

Here $w, w', w'', w_1, w_2, w_1', w_2' \in (N \cup T)^*$.

Proof:

(1) follows from the definition of $\overset{*}{\underset{G}{\Rightarrow}}$.

(2) clear with $n = 0$ in the definition of $\overset{*}{\underset{G}{\Rightarrow}}$.

(3) There exist sequences $w = w_0, \ldots, w_n = w',\ w' = w_0', \ldots, w_m' = w''$ with $w_i \overset{*}{\underset{G}{\Rightarrow}} w_{i+1}$ and $w_j' \overset{*}{\underset{G}{\Rightarrow}} w_{j+1}'$. Because $w_n = w' = w_0'$, the composed sequence $w = w_0, \cdots, w_n, w_1', \cdots, w_n' = w''$ is a derivation from w to w''.

(4) Exercise for the reader

Notation: An intermediate word that is generated by a derivation starting with the axiom is called a **sentence form** of G. We define:

Definition 6.3.

$$SF(G) := \{w \in (N \cup T)^* \mid S \overset{*}{\underset{G}{\Rightarrow}} w\}$$

is the set of **sentence forms** *of* G.

Now we are able to define the formal language generated by a grammar.

Definition 6.4. *Let* $G = (N, T, P, S)$ *be a grammar.*

$$L(G) := \{w \in T^* \mid S \overset{*}{\underset{G}{\Rightarrow}} w\}$$

is the **language generated by** G.

Note: $L(G) = SF(G) \cap T^*$.

Examples:

(1) One can see that for the grammar G_1 given in the previous example 1 it holds: $L(G_1)$ is the Dyck language over the alphabet $\{x, x'\}$.

(2) Let $G = (N, T, P, S)$ with $N = \{S\}, T = \{a, b\}, P = \{S \to aSb, S \to \epsilon\}$. Then $L(G) = \{a^n b^n \mid n \in \mathbb{N}$.

The simple proof is left to the reader.

Grammars are compared with relation to the languages they generate. We define:

Definition 6.5 (weak grammar equivalence). G *is* **weakly equivalent** *to* $G' \Leftrightarrow L(G) = L(G')$.

Remark: The reader should convince himself that the grammars G_1 and G_2 from the first example generate the same language.

Of course you can define infinitely many different grammars for each language.

We now can define different classes of grammars (and languages generated by these) depending on certain restrictions of their production system. In the next chapter we will meet the so called *right-linear* grammars and their languages.

Special importance, also from a practical point of view, have the so called *contextfree* grammars.

DEFINITION 6.6 (context-free grammar). *A grammar* $G = (N, T, P, S)$ *is called* **context-free** *if* $P \subset N \times (N \cup T)^*$.

The term "context-free" describes the fact that in a sentence-form a nonterminal may be replaced by the right-hand side of a production without need to respect the "context" to the left and right around that nonterminal symbol.

In chapter 4 we will treat context-free grammars in depth.

CHAPTER 2

Finite Automata

1. The finite automaton, regular sets in X^*, $REG(X^*)$

Let $G = (V, E)$ be a finite, oriented graph, X a finite set and $\alpha = (\mathcal{W}(G), X^*, \alpha_1, \alpha_2)$ a functor with $\alpha_1 : V \to \{X^*\},\ \alpha_2 : \mathcal{W}(G) \to X^*$.

(Remark by the translator: Here the free monoid X^* is regarded as a category $X^* = (\{X^*\}, X^*, Q, Z, \cdot\})$ where $\cdot$ is the monoid operation (word concatenation). Words are treated as morphisms with source and target X^*. Klingt komisch, is aber so.)

Definition 1.1 (nondeterministic finite automaton). *$\mathcal{A} = (G, X^*, \alpha)$ is called a* **nondeterministic finite automaton**.

If $S, F \in V$ are points of the graph G, we call $\mathcal{A} = (G, X^*, S, F, \alpha)$ a finite automaton with start and final states or shortly a **finite acceptor**.

In the following we will use the terms acceptor and automaton as synonyms.

If the finite automaton works over the free monoid X^* we write shorter just X instead of X^*, otherwise we specify the monoid explicitly.

Definition 1.2 (accepted set). *If we define*

$$\mathcal{W}(G)(S, F) := \{w \in \mathcal{W}(G) \mid Q(w) \in S \wedge Z(w) \in F\},$$

then

$$L_{\mathcal{A}} := \alpha_2(\mathcal{W}(G)(S, F))$$

is called the set **accepted by** *the automaton.*

We will also write shortly α instead of α_2 and $\mathcal{W}()(S, F)$ instead of $\mathcal{W}(G)(S, F)$.

Definition 1.3 (regular language over free monoid). *Let X be an alphabet.*

$$REG(X^*) := \{L \subset X^* \mid \textit{there exists a finite automaton } \mathcal{A} \textit{ with } L = L_{\mathcal{A}}\}$$

$REG(X^*)$ is the set of **regular languages** *over the free monoid X^*.*

Remark: We defined here the finite automaton via its "state graph". Most often, the definition is given using the "next state relation" as follows:

$$\delta = \{(a, P_1, P_2) \in X \times V \times V) \mid \\ \text{there exists an edge } e \text{ with } Q(e) = P_1, Z(e) = P_2 \text{ and } \alpha(e) = a \in X\}.$$

δ my be regarded as a relation between $X \times V$ and V where X is the input alphabet and V the state set of the automaton

$$\mathcal{B} = (X, V, \delta, S, F)$$

The elements of V denote the current state of the automaton $\mathcal{B}$.

If the automaton $\mathcal{B}$ is in state $z \in V$ and reads the symbol $x \in X$ then it changes into state $z' \in V$ where $(x, z, z') \in \delta$. If there doesn't exist such a z' the automaton halts.

This interpretation can be visualized as follows:

FIGURE

Let's return to out definition of the finite automaton. We explain its working based on our definition:

The automaton $\mathcal{A} = (G, X, S, F, \alpha)$ may be interpreted as a nondeterministic algorithm. The points of graph G define the possible states of the algorithm, the elements of X are the input alphabet.

The nondeterministic automaton $\mathcal{A}$ which reads a symbol $x \in X$ while in state $P \in V$ changes into state P' if there exists an edge e from P to P' with label $\alpha(e) = x$. If the graph has no such edge originating in P the automaton is set "out of service".

A finite acceptor accepts a word $w \in X^*$ if there exists a path from a point in S to a point in F which is labeld with w.

Let's consider some examples for finite automata:

Example 1: Let $X = \{a, b\}$ and $L = \{(ab)^{2n} \mid n \in \mathbb{N}\}$. It holds: $L \in REG(\{a, b\}^*)$.

The following acceptor accepts L (exercise):

$\mathcal{A} = (G, \{a, b\}, 1, 1, \alpha)$.

FIGURE

Example 2: Lexical analysis, check for special characters.

In every programming language there exist special character combinations (reserved words) that mark certain program actions. These have to be identified during the lexical analysis. We give a finite acceptor which realizes such a check for a selection of reserved words:

Let the set of reserved words be { 'BEGIN', 'END', 'ELSE', 'IF', FI', 'FOR', 'INTEGER', 'THEN', 'LOOP', 'POOL', 'PROCEDURE' }.

The following acceptor accepts this set:

FIGURE

The images of the edges under the mapping α are shown as edge labels. The points of the graph are the ovals with their labels. Start and final states are given by $S = \{$ START $\}$ and $F = \{$ STOP $\}$.

The labels of the points are chosen such that one can see the information stored by the automaton.

Now we want to prove some properties of $REG(X^*)$. To do that, we need some basic properties for finite automata.

LEMMA 1.1. *Let $\mathcal{A} = (G, X, S, F, \alpha)$ be a finite automaton. Then there exists a finite automaton $\mathcal{A}' = (G', X, S', F', \alpha')$ such that $card(S') = card(F') = 1$ and $L_{\mathcal{A}} = L_{\mathcal{A}'}$.*

An automaton with a single start state is called **initial**.

Proof: If $card(S) = card(F) = 1$ we are done.

(Comment by translator: The proof in the book is completely unreadable because of all these tildes, primes, indices etc. Therefore it is reformulated here.)

Let $card(S) > 1$ or $card(F) > 1$.

1. Add new edges leaving the new start state S':

Define the set of all edges leaving an old start state by

$$OUT := \{e \in E \mid Q(e) \in S\}$$

Add the following new edges to the graph:

$$OUT' := \{e' = (S', Z(e)) \mid e \in OUT,\ \alpha'(e') := \alpha(e)\}$$

2. Add new edges reaching the new final state F':

Define the set of all edges reaching an old final state by

$$IN := \{e \in E \mid Z(e) \in F\}$$

Add the following new edges to the graph:

$$IN' := \{e' = (Q(e), F') \mid e \in IN,\ \alpha'(e') := \alpha(e)\}$$

To each new edge we assign the same label as the edge from which it has been derived.

The new automaton $\mathcal{A}' = (G', X, \{S'\}, \{F'\}, \alpha')$ is defined by the graph $G' = (V \cup \{S', F'\}, E \cup OUT' \cup IN')$ and the new labeling α' which is identical to α for all existing edges and is defined as shown above for the new edges.

It is easily shown that $L_{\mathcal{A}'} = L_{\mathcal{A}}$.

Lemma 1.2. *Let $\mathcal{A} = (G, X, S, F, \alpha)$ be a finite automaton. Then there exists an automaton $\mathcal{A}' = (G', X, S', F', \alpha')$ with $\alpha'(e) \in X\ \forall e \in E(G)$ and $L_{\mathcal{A}} = L_{\mathcal{A}'}$.*

Proof: We "split" all edges according to their labels.

(1) Let $e \in E$ with $\alpha_2(e) = x_1 \cdots x_k,\ k > 1, x_i \in X$.
Remove edge e and add new edges $e'_1, \ldots, e'_k$ and new points $P'_1, \ldots, P'_{k-1}$ such that $(Q(e), e'_1, \ldots, e'_k, Z(e)) \in \mathcal{W}(G')$ and define a new graph $G' = (V', E')$. The labeling of the new edges is defined by $\alpha'(e'_i) := x_i$ for $i = 1, \ldots, k$.

Then $\alpha_2(e) = \alpha'_2(e'_1) \cdots \alpha'_2(e'_k)$.

(2) Let $e \in E$ be an edge labeled with ϵ.

(a) (Remove ϵ-loops)
If $Q(e) = Z(e) : E' := E - e$

(b) (Remove ϵ-edges which cannot be continued to a longer path)
If there is not $e' \in E$ with $Q(e') = Z(e') : E' := E - e$

(c) (Skip ϵ-edges that can be continued and remove the ϵ-edge)
If there exists an edge $e' \in E$ with $Q(e') = Z(e) : E'' := E - e$. Add new edges: $E' := E'' \cup \{\tilde{e} \mid Q(\tilde{e}) = Q(e), Z(\tilde{e}) = Z(e'),\ alpha'(\tilde{e}) := \alpha(e')\}$

If in step (b) or (c) the target of the edge is a final state, then add the source of the edge to the set of final states.

Continue this algorithm inductively until no more ϵ-edges remain in the graph. The algorithm terminates because the point and edge sets are finite. For the new automaton $\mathcal{A}'$ that results from this algorithm holds: $L_{\mathcal{A}'} = L_{\mathcal{A}}$.

If we apply this algorithm to our automaton from example 1, we obtain:

FIGURE

Now we want to prove some closure properties of $REG(X^*)$.

THEOREM 1.1 (Regular languages are closed under union and intersection).

$$L, L' \in REG(X^*) \Rightarrow L \cup L' \in REG(X^*) \wedge L \cap L' \in REG(X^*)$$

Proof: Let $L = L_{\mathcal{A}}$ and $L' = L_{\mathcal{B}}$ with automata

$$\mathcal{A} = (G_A, X, S_A, F_A, \alpha)$$

and

$$\mathcal{B} = (G_B, X, S_B, F_B, \beta).$$

We may assume that the edge and point sets of both automata graphs are disjoint.

(1) Closure under union: Define

$$\gamma_2(e) := \begin{cases} \alpha_2(e), & e \in E(G_A) \\ \beta_2(e), & e \in E(G_B) \end{cases}$$

Then the automaton $\mathcal{C} = (G_A \cup G_B, X, S_A \cup S_B, F_A \cup F_B, \gamma)$ accepts the language $L_{\mathcal{A}} \cup L_{\mathcal{B}}$.

(2) Closure under intersection: Define $G' = (V', E')$ where

$$V' = V_A \times V_B$$

$$E' = \{(e_A, e_B) \in E_A \times E_B \mid \alpha_2(e_A) = \beta_2(e_B)\}.$$

By lemma 2 we may assume that the edge labels are all single symbols from X.

We define the new labeling δ_2 by

$$\delta_2 : E' \to X,\ \delta_2((e_A, e_B) = \alpha_2(e_A)).$$

For the automaton $\mathcal{A}' = (G', X, S_A \times S_B, F_A \times F_B, \delta)$ then holds: $L_{\mathcal{G}'} = L_{\mathcal{A}} \cap L_{\mathcal{B}}$ and this automaton is called the **cartesian product** of $\mathcal{A}$ and $\mathcal{B}$.

THEOREM 1.2 (Regular languages are closed under mirror operation).

$$L \in REG(X^*) \Rightarrow L^R \in REG(X^*)$$

Proof: Let $\mathcal{A} = (G, X, S_{\mathcal{A}}, F_{\mathcal{A}}, \alpha)$ be a finite acceptor for L.

Create the graph $G' = (V(G), E')$ where E' is a set with the same cardinality as E. There exists a bijection from E to E' mapping edges as follows:

$$e : P \to P' \in E \Leftrightarrow e' : P' \to P \in E'$$

which means we reverse the orientation of the edges.

For the resulting automaton $\mathcal{A}' = (G', X, S_{\mathcal{A}}, F_{\mathcal{A}}, \alpha')$ with $\alpha'(e') = \alpha(e)$ for all edges of G' it holds: $L_{\mathcal{A}'} = {L_{\mathcal{A}}}^R = L^R \in REG(X^*)$.

DEFINITION 1.4 (deterministic and complete automaton). *Let $\mathcal{A} = (G, X, S, F, \alpha)$ be a finite automaton with $\alpha(E(G)) \subset X$ (each edge is labeled with a single symbol).*

$\mathcal{A}$ is called **deterministic** *$\Leftrightarrow$ for all $e, e' \in E(G)$ with $Q(e) = Q(e')$ and $\alpha(e) = \alpha(e')$ it holds: $e = e'$.*

$\mathcal{A}$ is called **complete** *$\Leftrightarrow$ for each $P \in V(G)$ and $x \in X$ there exists an edge $e \in E(G)$ with $Q(e) = P$ and $\alpha(e) = x$.*

THEOREM 1.3 (Existence of complete, deterministic acceptor). *If $\mathcal{A}$ is a finite automaton, then there exists a complete, deterministic automaton $\mathcal{A}'$ which accepts the same language.*

Proof: From our lemmata we may assume that $card(S_{\mathcal{A}}) = 1$ and $\alpha_2(e) \in X$ for all edges e in the graph of $\mathcal{A}$.

We construct an automaton $\mathcal{A}'$ as follows ("subset construction"):

Instead of the points of the graph G our new graph has the power set $Pot(V(G))$ of $V(G)$ as its point set.

For $\tilde{P} \in Pot(V(G))$ we define

$$\begin{aligned} N(x, \tilde{P}) := \quad & \{P \in V(G) \mid \text{there exists an edge } e \in E(G) \text{ with} \\ & Q(e) \in \tilde{P}, Z(e) = \tilde{P} \text{ and } \alpha(e) = x\} \end{aligned}$$

The empty set $\emptyset$ is also an element of the power set such that it will also become a point of the new graph.

Let $\tilde{P}, \tilde{R} \in Pot(V(G))$. These points are connected by an edge e with label $\alpha_s'(e) = x \Leftrightarrow \tilde{R} = N(x, \tilde{P})$.

This defines our new graph $G' = (V', E')$.

The start and final states are defined as follows:

$$S_{\mathcal{A}'} = \{S_{\mathcal{A}}\}$$
$$F_{\mathcal{A}'} = \{\tilde{R} \in Pot(V(G)) \mid \tilde{R} \cap F_{\mathcal{A}} \neq \emptyset\}$$

This completes the definition of automaton $\mathcal{A}' = (G', X, S_{\mathcal{A}'}, F_{\mathcal{A}'}, \alpha')$.

We first prove that $\mathcal{A}'$ is complete and deterministic:

$\mathcal{A}'$ has a single start state $S_{\mathcal{A}'} = \{S_{\mathcal{A}}\}$, the edge labels are all single symbols and

$$card(\{e \in E' \mid Q(e) = \tilde{P}, \alpha_2'(e) = x\}) = 1$$

for all $\tilde{P} \in V(G')$.

Now we prove that the accepted languages are equal:

"$\subset$": We use diagrams of the following form:

$$\begin{array}{c} P \xrightarrow{x} R \\ \tilde{P} \xrightarrow{x} \tilde{R} \end{array}$$

which are to be understood as follows: For $P \xrightarrow{x} R \in E(G)$ there exists by construction $\tilde{P} \xrightarrow{x} \tilde{R} \in E(G')$ with $P \in \tilde{P}$ and $R \in \tilde{R}$.

Starting with $S_{\mathcal{A}'} = \{S_{\mathcal{A}}\} = \{\{P_0\}\}$ and concatenating these diagrams, we get for $x_1 \cdots x_k \in L_{\mathcal{A}}$:

$$\begin{array}{l} P_0 \xrightarrow{x_1} P_1 \xrightarrow{x_2} P_2 \longrightarrow \ldots \longrightarrow P_{k-1} \xrightarrow{x_k} P_k \in F_{\mathcal{A}} \\ \tilde{P}_0 \xrightarrow{x_1} \tilde{P}_1 \xrightarrow{x_2} \tilde{P}_2 \longrightarrow \ldots \longrightarrow \tilde{P}_{k-1} \xrightarrow{x_k} \tilde{P}_k \end{array}$$

That means $\tilde{P}_k \cap F_{\mathcal{A}} \neq \emptyset \Rightarrow \tilde{P}_k \in F_{\mathcal{A}'} \Rightarrow w \in L_{\mathcal{A}'}$.

$$\Rightarrow L_{\mathcal{A}} \subset L_{\mathcal{A}'}$$

"$\supset$": Here we use diagrams

$$\begin{array}{c} \tilde{P} \xrightarrow{x} \tilde{R} \\ P \xrightarrow{x} R \end{array}$$

which are to be understood as follows: For $\tilde{P} \xrightarrow{x} \tilde{R} \in E(G')$ and $R \in \tilde{R}$ there exists $P \in \tilde{P}$ with $P \xrightarrow{x} R \in E(G)$.

For $x_1 \cdots x_K \in L_{\mathcal{A}'}$ we start with $F_{\mathcal{A}'}$ and continue the diagram from right to left. For $P_k \in \tilde{P}_k \cap F_{\mathcal{A}}$ we get

$$\begin{array}{rl} S_{\mathcal{A}'} \ni & \tilde{P}_0 \xrightarrow{x_1} \tilde{P}_1 \xrightarrow{x_2} \tilde{P}_2 \longrightarrow \ldots \longrightarrow \tilde{P}_{k-1} \xrightarrow{x_k} \tilde{P}_k \in F_{\mathcal{A}'} \\ & P_0 \xrightarrow{x_1} P_1 \xrightarrow{x_2} P_2 \longrightarrow \ldots \longrightarrow P_{k-1} \xrightarrow{x_k} P_k \in F_{\mathcal{A}} \end{array}$$

Because of $S_{\mathcal{A}'} = \{S_{\mathcal{A}}\} = \{\{P_0\}\}$ it holds: $x_1 \cdots x_k \in L_{\mathcal{A}} \Rightarrow L_{\mathcal{A}'} \subset L_{\mathcal{A}}$.

$$\Rightarrow L_{\mathcal{A}'} \subset L_{\mathcal{A}}$$

From both inclusions we get $L_{\mathcal{A}} = L_{\mathcal{A}'}$.

Remark: The state set grows exponentially when the automaton is made deterministic. The following example will clarify this fact [**?**].

Example: Let $X = \{a, b\}$. Define for arbitrary, but fixed $n \in \mathbb{N}$

$$L_n := \{w \mid w = w_1 \cdot w_2 \text{ with } w1 \neq w_2, |w_1| = |w_2| = n, w_1, w_2 \in X^*\}$$

$L_n \in REG(X^*)$ because there exists a nondeterministic finite acceptor $\mathcal{A}_n$ with $L_n = L_{\mathcal{A}_n}$.

We want to give the automaton for $n = 3$:

$\mathcal{A}_3 = (G_3, \{a, b\}, \{1\}, \{22\}, \alpha)$ with the following graph G_3:

FIGURE

The labeling α is shown at the edges.

Exercise: Show that this automaton accepts L_3.

If we construct for L_3 from the nondeterministic acceptor $\mathcal{A}_3$ the deterministic acceptor $\mathcal{A}'_3$, the graph G'_3 looks as follows:

FIGURE

The edges pointing upwards are labeled with a and the edges pointing downwards with b.

G'_3 is the graph of the complete (up to loops in the final state and state $\emptyset$), deterministic acceptor $\mathcal{A}'_3$ and accepts the language L_3 (exercise).

In [**?**] it is proved that this automaton is minimal in the number of states.

Making a state graph deterministic brings advantages as well as disadvantages. A program simulating finite automata based on the given state graph is fast for a deterministic graph. However it needs more space because of the possibly very large representation of the deterministic graph.

If the program uses the nondeterministic graph, it can follow all alternatives in parallel similar to our construction of the deterministic automaton. It needs less memory but the computing time can grow linear for each simulation step.

We want to state an important consequence from our last theorem.

THEOREM 1.4 (Regular languages are closed under complement).

$$L \in REG(X^*) \Rightarrow \bar{L} = X^* - L \in REG(X^*)$$

Proof:

$L \in REG(X^*) \Rightarrow$ there exists a complete, deterministic finite acceptor $\mathcal{A}$ with $L_{\mathcal{A}} = L$.

Define $\mathcal{A}' = (G, X, S_{\mathcal{A}}, F_{\mathcal{A}}, \alpha)$ where $F_{\mathcal{A}} := V(G) - F_{\mathcal{A}}$.

Because $\mathcal{A}$ is complete and deterministic, every word $w \in X^*$ determines a unique path starting in $S_{\mathcal{A}}$.

This $w \in X^*$ uniquely determines a point $P \in V(G)$ with $w \in \alpha(\mathcal{W}(W)(S_{\mathcal{A}}, P))$.

It is either $P \in F_{\mathcal{A}}$ or $P \in F_{\mathcal{A}'}$ and from $F_{\mathcal{A}} \cap F_{\mathcal{A}'} = \emptyset$ follows $w \in L_{\mathcal{A}} \Leftrightarrow w \notin L_{\mathcal{A}'}$ and therefore $L_{\mathcal{A}'} = \bar{L_{\mathcal{A}}}$.

Theorem 1.5 (Regular languages are closed under concatenation).

$$L_1, L_2 \in REG(X^*) \Rightarrow L_1 \cdot L_2 \in REG(X^*)$$

Proof: $L_1, L_2 \in REG(X^*) \Rightarrow$ there exist finite acceptors $\mathcal{A}_i = (G_i, X, S_i, F_i, \alpha_i)$ with $L_i = L_{\mathcal{A}i},\ i = 1, 2$.

We define a new acceptor

$$\mathcal{A} = (G, X, S, F, \alpha)$$

as follows:

Vertices:

$$V(G) := V(G_{\mathcal{A}_1}) \cup V(G_{\mathcal{A}_2}) \text{ where } V(G_{\mathcal{A}_1}) \cap V(G_{\mathcal{A}_2}) = \emptyset$$

Edges:

$$E(G) = E(G_{\mathcal{A}_1}) \cup E(G_{\mathcal{A}_2}) \cup \text{ 'bridge'' edges defined as follows:}$$

FIGURE (bridge edge)

Here $R \in F1$ is a final state in $\mathcal{A}_1$, $x = \alpha_1(e_1)$ is the label of edge e_1, P' is a start state in $\mathcal{A}_2$.

For each such configuration, a new "bridge" edge e is added to $E(G)$ where

$$Q(e) := P,\ Z(e) := P' \text{ and } \alpha(e) := x.$$

Start and final states: $S := S_{\mathcal{A}_1}$ and $F := F_{\mathcal{A}_2}$.

We now prove that $L_{\mathcal{A}} = L_{\mathcal{A}1} \cdot L_{\mathcal{A}2}$.

(1) $L_{\mathcal{A}} \supset L_{\mathcal{A}1} \cdot L_{\mathcal{A}2}$

Let $u = u_1 \cdots u_n \in L_{\mathcal{A}1}$ and $v = v_1 \cdots v_m \in L_{\mathcal{A}2}$. Then there exist accepting paths

$$S_1 \ni P_0 \xrightarrow{u_1} P_1 \xrightarrow{u_2} \ldots \xrightarrow{u_n} P_n \in F_1$$

and

$$S_2 \ni Q_0 \xrightarrow{v_1} Q_1 \xrightarrow{v_2} \ldots \xrightarrow{v_m} Q_m \in F_2$$

By construction of graph G there exists a path in $\mathcal{W}(G)(S1, F2)$ of the form

$$S_1 \ni P_0 \xrightarrow{u_1} P_1 \xrightarrow{u_2} \ldots P_{n-1} \xrightarrow{u_n} Q_0 \xrightarrow{v_1} Q_1 \xrightarrow{v_2} \ldots \xrightarrow{v_m} Q_m \in F_2$$

It follows $u = u_1 \cdots u_n \cdot v_1 \cdots v_m \in L_{\mathcal{A}}$ thus $L_{\mathcal{A}1} \cdot L_{\mathcal{A}2} \subset L_{\mathcal{A}}$.

(2) $L_{\mathcal{A}} \subset L_{\mathcal{A}1} \cdot L_{\mathcal{A}2}$

Let
$$S \ni P_0 \xrightarrow{w_1} P_1 \xrightarrow{w_2} \dots \xrightarrow{w_n} P_n \in F$$
be an accepting path in $\mathcal{W}(G)(S, F)$. Then there exist states $P_i \in S_2$ and $P_j \in V(G_1)$ with $j < i$ because $V(G_1) \cap V(G_2) = \emptyset$.

FIGURE

Therefore the subpath
$$S_2 \ni P_i \xrightarrow{w_{i+1}} P_{i+1} \longrightarrow \dots \xrightarrow{w_n} P_n \in F_2$$
contains only edges and vertices from $\mathcal{A}_2$ and is an accepting path for the word $w_{i+1} \cdots w_n \in L_{\mathcal{A}2}$.

It holds $P_{i-1} \in V_1$ and by construction of the graph G there exists an edge labeled with w_i which ends in a final state of $\mathcal{A}_1$. So $w_1 \cdots w_i \in L_{\mathcal{A}1}$.

Together we get
$$w_1 \cdots w_i \cdots w_n \in L_{\mathcal{A}1} \cdot L_{\mathcal{A}2} \Rightarrow L_{\mathcal{A}} \subset L_{\mathcal{A}1} \cdot L_{\mathcal{A}2}.$$

From (1) and (2) it follows $L_{\mathcal{A}} = L_{\mathcal{A}1} \cdot L_{\mathcal{A}2}$.

Theorem 1.6 (Regular languages are closed under Kleene-star).
$$L \in REG(X^*) \Rightarrow L^* \in REG(X^*)$$

FIGURE (graph construction)

Let $\mathcal{A} = (G, X, S, F, \alpha)$ be a finite acceptor for L, create the finite acceptor $\mathcal{A}' = (G', X, S, F, \alpha')$ as follows:

Graph: Take the graph G and add the following edges:

(1) For each edge
$$e : P \xrightarrow{x} R \in E(G),\ R \in F$$
leading to a final state and each start state P_0 add an edge
$$e' : P \xrightarrow{x} P_0$$
labeled by $\alpha'(e') := x$.

(2) For each state $P_0 \in S$ and each final state $R \in F$ add an edge (if not already existing)
$$e' : P_0 \xrightarrow{\epsilon} R$$
labeled by $\alpha'(e') := \epsilon$.

For the finite acceptor $\mathcal{A}'$ it holds (exercise):
$$L_{\mathcal{A}'} = L^*$$

We have seen (lemma 1) that for nondeterministic automata a single start and a single final state are sufficient, but in the deterministic case, a single final state in insufficient in general.

Theorem 1.7 (Regular languages are closed under homomorphism).
$$L \in REG(X^*), \phi : X^* \to Y^* \textit{ monoid homomorphism } \Rightarrow$$
$$\phi(L) := \{\phi(w) \mid w \in L\} \in REG(X^*)$$

Proof: Let $\mathcal{A} = (G, X, S, F, \alpha)$ be a finite acceptor for L. Then

$$\mathcal{B} = (G, Y, S, F, \beta)$$

with $\beta_2(e) := \phi(\alpha_2(e)),\ e \in E(G)$, is an acceptor for $\phi(L)$ because

$$\phi(L) = \phi(\alpha_2(\mathcal{W}(G)(S, F))) = \beta_2(\mathcal{W}(G)(S, F))$$

Therefore

$$\phi(L) \in REG(Y^*)$$

After having proved closure properties, we prove some other important properties of regular sets.

LEMMA 1.3. *For $x \in X$ it holds $\{x\} \in REG(X^*)$ and $\emptyset \in REG(X^*)$.*

Proof: Set $G = (\{S, F\}, \{e\})$ be a graph, then the finite acceptor

$$\mathcal{A}_x = (G, \{x\}, \{S\}, \{F\}, \alpha)$$

with $\alpha_2(e) = x$ fulfills $L_{\mathcal{A}} = \{x\}$.

Set $G = (\{S, F\}, \emptyset)$, then $\mathcal{A} = (G, \emptyset, \{S\}, \{F\}, \emptyset)$ fulfills $L_{\mathcal{A}} = \emptyset$.

LEMMA 1.4. *Set $G = (V, E)$. Then $\mathcal{W}(G)(S, F) \in REG(E^*)$.*

This means the set of paths of a graph which start in a start state and end in a final state is a regular set over the set of edges of the graph. The proof is trivial, just label the edges with their name.

DEFINITION 1.5 (local set). *Let $S, F \subset X$ and $R \subset X \times X$ be a relation on X with*

$$L = \{x_1 \cdots x_k \mid x_1 \in S, x_k \in F, (x_i, x_{i+1}) \in R,\ i = 1, \ldots, k-1\}$$

LEMMA 1.5 (The accepting path set of a FA is local). *Let $G = (V, E)$ be a graph. Then $\mathcal{W}(G)(S, F)$ is local over E.*

Proof: Let

$$\begin{aligned} S &= \{e \in E \mid Q(e) \in S\} \\ F &= \{e \in E \mid Z(e) \in F\} \\ R &= \{(e, e') \in E \times E \mid Z(e) = Q(e')\} \end{aligned}$$

Then the claim is immediately proved.

LEMMA 1.6 (local sets are regular).

$$L \textit{ local over } X \Rightarrow L \in REG(X^*)$$

Proof: Let S and F be the subsets of X from the definition of a local set. We define a graph $G = (V, E)$ as follows:

$$\begin{aligned} V &= X \cup \{\bar{S}\} \text{ with } X \cap \{\bar{S}\} = \emptyset \\ E &= \{e : \{\bar{S}\} \to x \mid x \in S\} \cup \{e : x \to y \mid (x, y) \in R\} \end{aligned}$$

Consider the finite acceptor $\mathcal{A} = (G, X, \{\bar{X}\}, F, \alpha)$ with labeling α defined by $\alpha_2(e) = Z(e)$ for each edge $e \in E$.

Then it is clear that $L_{\mathcal{A}} = L$.

LEMMA 1.7 (Each regular set is homomorphic image of a local set). *$L \in REG(X^*)$, then there exists a local set R over X and a monoid homomorphism ϕ such that*

$$\phi(R) = L$$

Proof: Exercise

We want to summarize our results to some main theorems where we organize these results differently.

Let $X_\infty = \{x_1, x_2, \ldots\}$ be an infinite alphabet. Define

$$REG(X_\infty^*) = \bigcup_{X \subset X_\infty} REG(X^*), X \text{ is finite}$$

Main Theorem 1.1. (1) *$REG(X_\infty^*)$ is closed under union, complex product and Kleene-star *.*

(2) *If $\phi : X_\infty^* \to X_\infty^*$ is a monoid homomorphism, then $REG(X_\infty^*)$ is closed under ϕ.*

(3) *Every $L \in REG(X_\infty^*)$ is the homomorphic image of a local set over X_∞^*.*

(4) *$REG(X_\infty^*)$ contains the sets $\{x\}, x \in X_\infty^*$ and the empty set $\emptyset$.*

Main Theorem 1.2. (1) *$REG(X_\infty^*)$ is a Boolean Algebra with operations union, intersection and complement.*

(2) *Every language $L \in REG(X_\infty^*)$ is accepted by some complete, deterministic automaton.*

To prove the first main theorem we don't need the fact that X^* is a **free** monoid. In the proof of main theorem 2 we used that fact.

There exist examples of monoids M for which the second main theorem does not hold for $REG(M)$, in both sentences (see exercises).

Of special interest are the monoids $M = F(X)$, where $F(X)$ is the free group generated by X (see chapter 1.3) and $M = X^\oplus$ where $X^\oplus$ is the free commutative group generated by X.

Languages $L \in REG(X^\oplus)$ are also called **semi-linear**.

Exercices:

(1) Let $\phi : (\{x, y\}^*, \cdot) \to (\mathbb{Z}/3\mathbb{Z} \times \mathbb{Z}/4\mathbb{Z}, +)$ be given by $\phi(x) = (0, 1), \phi(y) = (1, 0)$.

Construct a finite automaton $\mathcal{A}$ with $L_\mathcal{A} = \phi^{-1}((0, 0))$ and present $\phi^{-1}((0, 0))$ as a rational set.

(2) Let $\mathcal{A} = (G, X, S, F, \alpha)$ be a finite automaton with n states. Prove:

   (a) If $x \in L_\mathcal{A}$ with $|x| >= n$, then there exists words $u, vw \in X^*$ with $x = u \cdot v \cdot w, v \neq \epsilon$ and $u \cdot v^m \cdot w \in falangA\ \forall m \geq 1$ ("pumping lemma").

   (b) $L_\mathcal{A}$ infinite $\Leftrightarrow \exists x \in X^*$ with $n \leq |x| \leq 2n$ and $x \in L_\mathcal{A}$.

(3) Show: The set $L = \{a^n b^n\} \mid n \in \mathbb{N}$ is not regular.

(4) Let $L \in REG(X^*), M \subset X^*$ finite. Show: $M^{-1}L \in REG(X^*)$. What if $M \subset REG(X^*)$ is an arbitrary regular set?

2. Rational sets in $X^*, RAT(X^*)$

In this section we want to introduce a second characterization of regular sets over free monoids. We need the following definition.

Definition 2.1 (rationally closed). *If M is a monoid, then $K \in Pot(M)$ is called is called* **rationally closed** $\Leftrightarrow$ *K is closed unter union $\cup$, complex product $\cdot$ and Kleene-star *.*

Definition 2.2. *$RAT(M)$ is the smallest, rationally closed subset of $Pot(M)$ that contains the single element sets $\{m\}, m \in M$ and the empty set $\emptyset$.*

From main theorem 1 immediately follows:

Lemma 2.1. *$RAT(X^*) \subset REG(X^*)$.*

From the remark following the main theorems it follow that this also holds for arbitray monoids M because $\{m\} \in REG(M)$ for $m \in M$.

This lemma is the first part of the following theorem by Kleene:

Theorem 2.1 (Kleene).

$$REG(X^*) = RAT(X^*)$$

Proof: We still have to proof the inclusion $REG(X^* \subset RAT(X^*))$.

Let $\mathcal{A}$ be a finite automaton. We show that $L_{\mathcal{A}}$ can be generated from $\emptyset$ and $\{x\}, x \in X$ using $\cup$, $\cdot$ and *-operations.

We prove this by induction over the number k of edges in the graph of the automaton when the number of vertices is fixed.

We may assume that the automaton has a single start state, $card(S) = 1$.

$k = 0$: Without loss of generality, we may assume $S \cap F = \emptyset$. It follows $L_{\mathcal{A}} = \emptyset$ which is a rational set over X^*.

Induction step: Let the claim be true for $i \leq k, k > 1$ and let G_k be a graph with k edges.

Add a new edge $e : P \xrightarrow{x} R$ labeled with $x \in X$ to the graph G_k. The resulting graph shall be G_{k+1}. Then the set of accepting paths in G_{k+1} can be written as

$$\begin{aligned} \mathcal{W}(G_{k+1})(S,F) &= \mathcal{W}(G_k)(S,F) \\ &\cup \mathcal{W}(G_k)(S,P) \cdot e \cdot \Big(\mathcal{W}(G_k)(R,P) \cdot e\Big)^* \cdot \mathcal{W}(G_k)(R,F) \end{aligned}$$

FIGURE

By induction hypothesis it holds:

$$\alpha(\mathcal{W}(G_k)(S,F)) \in RAT(X^*)$$
$$\alpha(\mathcal{W}(G_k)(S,P)) \in RAT(X^*)$$
$$\alpha(\mathcal{W}(G_k)(R,F)) \in RAT(X^*)$$
$$\alpha(\mathcal{W}(G_k)(R,P)) \in RAT(X^*)$$

From this we get $\alpha(\mathcal{W}(G_{k+1})(S,F)) \in RAT(X^*)$.

This concludes the proof of Kleene's theorem.

3. Sets recognizable by homomorphisms, $REC(X^*)$

In this section we investigate a third possibility for characterizing subsets of X^*.

DEFINITION 3.1.

$$\begin{aligned} REC(X^*) \quad := \quad & \{L \subset X^* \mid \textit{ there exists a finite monoid } H \\ & \textit{and a monoid homomorphism } \mu : X^* \to H \\ & \textit{with } L = \mu^{-1}(T), T \subset H\} \end{aligned}$$

is the class of **recognizable sets** *over* X.

LEMMA 3.1.

$$REG(X^*) \subset REC(^*)$$

Proof: Let $L \in REG(X^*)$, then there exists a complete, deterministic finite automaton $\mathcal{A} = (G, X, S, F, \alpha)$ with graph $G = (V, E)$ accepting L.

We define

$$H := Map(V, V) := \{f : V \to V \mid f \text{ is a mapping}\}$$

and define a homomorphism $\mu : X^* \to H$ as follows:

Let μ be the homomorphic continuation of mapping μ' where

$$\mu'(x)(P) = R$$
$$\Leftrightarrow$$
$$\text{there exists an edge } e \in E \text{ with } Q(e) = P, Z(e) = R \text{ and } \alpha(e) = x \in X$$

We define further

$$T := \{f \in H \mid f(S) \in F\}$$

Then it holds $L_{\mathcal{A}} = \mu^{-1}(T)$ (exercise).

LEMMA 3.2.

$$REC(X^*) \subset REG(X^*)$$

Proof: Sei $L \in REC(X^*)$, thus let H be a finite monoid and $\mu : X^* \to H$ be a monoid homomorphism and $L = \mu^{-1}(T), T \subset H$.

We construct a graph $G = (V, E)$:

Vertex set: $V := H$.

Edges: $E := \{(h, a) \mid h \in H, a \in X\}$ with $Q((h, a)) = h, Z((h, a)) = h \cdot \mu(a)$.

FIGURE

Let $\mathcal{A} = (G, X, S, F, \alpha)$ with $S = \{1_H\},\ F = T,\ \alpha((h, a)) = a$.

We show: $L = L_{\mathcal{A}}$.

"$\subset$": Let $w \in L$, so $w \in X^*$ with $\mu(w) \in T$.

If $w = x_1 \cdots w_n$ with $x_i \in X$, then $\mu(x_1) \cdots \mu(x_n) = t \in T$.

We consider

$$v = (1_H, (1_H, x_1), (\mu(x_1), x_2), (\mu(x_1 \cdot x_2), x_3), \ldots, (\mu(x_1 \cdots x_{n-1}), x_n), \mu(w))$$

$v \in \mathcal{W}(G)$ with $\alpha(v) = x_1 \cdots x_n = w$. It even holds $v \in \mathcal{W}(G(S, F))$ because $\mu(w) \in T \Rightarrow w \in L_{\mathcal{A}}$.

"$\supset$": Let $w \in L_{\mathcal{A}}$, then there exists a path $v \in \mathcal{W}(G)(\{1_h\}, T)$ with $\alpha(v) = w$. For $w = x_1 \cdots x_n$ then obviously holds $\mu(x_1 \cdots x_n) \in T$ thus $w \in \mu^{-1}(T) = L$ from which the claim follows.

From both lemmata immediately follows

THEOREM 3.1 (Regular sets of free monoid are exactly the recognizable sets).

$$REG(X^*) = REC(X^*)$$

In the first section of this chapter we showed closure properties for regular sets. We show now closure under inverse homomorphism which is rather easy using our last theorem.

THEOREM 3.2 (Recognizable sets are closed under inverse homomorphism). *Let $\phi : X^* \to Y^*$ a monoid homomorphism, $L \in REC(Y^*) \Rightarrow \phi^{-1}(L) \in REC(X^*)$.*

Let $\phi : X^* \to Y^*$ and L should be defined by monoid homomorphism $\mu' : Y^* \to H$ as $L = \mu'^{-1}(T)$ for a subset $T \subset H$.

We set $\mu = \phi \circ \mu'$ and have $\mu : X^* \to H$ is a monoid homomorphism and $\phi^{-1}(L) = \mu^{-1}(T)$ from which follows $\phi^{-1}(L) \in REC(X^*)$.

Exercise: For the integer numbers $\mathbb{Z}$ the set of recognizable sets $REC(\mathbb{Z})$ shall be defined in analogy to $REC(X^*)$.

Characterize $REC(\mathbb{Z})$! Are there nonempty, infinite sets in $REC(\mathbb{Z})$?

4. Right-linear languages, $r{-}LIN(X^*)$

In this section we investigate an additional method for defining special subsets of X^*. We use the mechanism introduced in chapter 1.6, the Chomsky grammars.

We consider only a very restricted type of productions and obtain the **right-linear** grammars.

DEFINITION 4.1 (right-linear Chomsky grammar). *A Chomsky grammar* $G = (N, X, P, S)$ *is called*

$$\begin{array}{rl} \textbf{right-linear} & \textit{if } P \subset N \times (X \cdot N \cup X) \\ \textbf{left-linear} & \textit{if } P \subset N \times (N \cdot X \cup X) \end{array}$$

X is the **terminal alphabet**.

As in chapter 1.6 we have the notions of *derivation* and *generated language*.

DEFINITION 4.2 (right-linear languages).

$r{-}LIN(X^*) = \{L \subset X^* \mid$ *there exists a right-linear grammar* G *with* $L = L(G)\}$

is the class of **right-linear languages** *in* X^*.

In the same way one defines the class of **left-linear languages**.

The following theorem holds:

THEOREM 4.1.

$$REG(X^*) = r{-}LIN(X^*)$$

Proof:

"$\supset$": Let $L \in r{-}LIN(X^*)$ with $L = L(G),\ G = (N, X, P, S)$.

We construct a graph $G' = (V, E)$ with vertices $V = N \cup \{F\}, F \notin N$ and we use the productions of the grammar as edge set E:

$$\begin{array}{rcl} p : v \to xv' \in P & \Rightarrow & e : v \xrightarrow{x} v' \in E \\ p : v \to x \in P & \Rightarrow & e : v \xrightarrow{x} F \in E \end{array}$$

(Translator remark: $e : v \xrightarrow{x} v'$ is just a shorter notation for $Q(e) = v,\ Z(e) = v',\ \alpha(e) = x$)

This defines a finite acceptor $\mathcal{A} = (G', X, S, F, \alpha)$.

It is easily seen that

$$w \in \mathcal{W}(G') \Rightarrow Q(w) \underset{G}{\overset{*}{\Rightarrow}} \alpha(w) \cdot Z(w) \text{ , if } Z(w) \in N$$

For paths $w \in \mathcal{W}(G')(S, F)$ we get $\alpha(w) \in L$.

From this it follows $L_{\mathcal{A}} \subset L(G)$.

Let $w \in L(G)$ be a word generated by the right-linear grammar G, then there exists a sequence of derivation steps

$$S \underset{G}{\Rightarrow} x_1 v_1 \underset{G}{\Rightarrow} x_1 x_2 v_2 \underset{G}{\Rightarrow} \ldots \underset{G}{\Rightarrow} x_1 \cdots x_{n-1} v_{n-1} \underset{G}{\Rightarrow} w$$

Then there exists a path

$$w' = \big(S, (S, x_1 v_1), (v_1, x_2 v_2), \ldots, (v_{n-1}, x_{n-1} v_{n-1}), (v_{n-1}, x_n), F\big) \in \mathcal{W}(G')(S, F)$$

and the label of this path is

$$\alpha(w') = x_1 \cdots x_n = w$$

This means, the word w is accepted by the finite acceptor, $w \in L_{\mathcal{A}}$.

Together, we get $L_{\mathcal{A}} = L(G)$.

"$\subset$":

Let $L \in REG(X^*),\ L = L_{\mathcal{A}}$ with a finite acceptor with a single start state $\mathcal{A} = (G, X, S_{\mathcal{A}}, F_{\mathcal{A}}, \alpha),\ G = (V, E)$ and $|\alpha(e)| = 1$ for each edge $e \in E$.

We define a right-linear grammar $RLG = (N, X, P, S)$ with $N = E, S = S_{\mathcal{A}}$ and production set

$$\begin{aligned} P &= \{(q, \alpha(e) \cdot r) \mid \exists \text{ edge } e : q \to r \in E\} \\ &\cup \ \{(q, \alpha(e)) \mid \exists \text{ edge } e : q \to r \in E,\ r \in F_{\mathcal{A}}\} \end{aligned}$$

It holds $L(RLG) = L_{\mathcal{A}}$ (exercise) which completes the proof of the theorem.

Remark: It also holds $l\!-\!LIN(X^*) = REG(X^*)$. This can be seen by theorem 2 in chapter 2.1 which states that $REG(X^*)$ is closed under the mirror-operation.

Given a right-linear grammar for L one can directly define a left-linear grammar for the mirror language L^R: Replace each production $X \to t \cdot Y$ by a left-linear production $X \to Y \cdot t$. Because of $L^{R^R} = L$ it then holds $L \in l\!-\!LIN(X^*)$.

We can therefore in all of the following results replace $r\!-\!LIN(X^*)$ by $l\!-\!LIN(X^*)$.

We extend now our main theorems from section 1. Let X_∞ again be the countably infinite alphabet and let $RAT(X_\infty^*)$ and $REC(X_\infty^*)$ be defined as before.

Main Theorem 4.1.

(1) $REG(X_\infty^*) = RAT(X_\infty^*) = REC(X_\infty^*) = r\!-\!LIN(X_\infty^*) = l\!-\!LIN(X_\infty^*)$

(2) $REG(X_\infty^*)$ *is closed under union, intersection, complement, Kleene-star, complex product, monoid homomorphisms and inverse homomorphisms*

Each of these language classes motivates a different generalization. We want to shortly present those.

Regular languages: Generalization into two directions come to mind: Replace the free monoid X^* by an arbitrary monoid or by special monoids like free groups or free commutative monoids. We already pointed that out and we will investigate two important cases later in this book.

The second generalization concerns the free (path) category. By adding a second operation we get simply computable, infinite categories. The paths in graphs are then replaced by trees or nets representing the morphisms of these new free category.

Rational languages: Instead of the free monoid one can use arbitray monoids. We already made some statements about this. We now already that this generalization will coincide with a corresponding generalization of the regular sets.

In an even stronger generalization the monoid can be replaced by other algebras, that is one adds other operations in addition to the monoid operation. The question arises if in this case the strong relation between the rattional and regular sets will be conserved.

Recognizable languages: Here also two different directions for generalization come to mind. First, one can keep monoids but replace finite monoids by simply computable infinite monoids.

A second generalization concerns the replacement of monoids by other algebras. Especially the free monoid can be replaced by free algebras and the finite monoid by finite algebras.

Right-linear languages: Here one generalizes by using general Chomsky grammars (see definition 1, chapter 1.6). This last generalization will be in our focus later.

Our main theorem motivates yet another generalization:

Abstract Families of Languages (AFL): The AFL-theory generalizes the notion of the family of rational languages. This is done by introducing the notion of **Kleene-algebra** which is based on the operations of union, complex product and Kleene-star.

A set $\mathcal{A} \in Pot(X_\infty^*)$ is called a **full AFL**, if $\mathbb{A}$ is closed under the operations of union, complex product, Kleene-star, homomorphisms and inverse homomorphisms and intersection with regular languages.

The idea is the following:

All these operations are simple, therefore using these operations only "simple sets" mya be constructed from some given set of "simple sets". It should be possible to decide the common computer science questions for all elements in an AFL if one can decide them for the generating basic sets.

In this book we cannot treat all these topics. We will not develop the AFL-theory.

The interested reader is referred to the book by Jean Berstel [**?**] which handles this theory in great detail.

5. The rational transducer

In this section we extend the finite automaton with an output which leads to:

DEFINITION 5.1.

$$\mathcal{T} = (G, X, S, F, \alpha, \beta)$$

is called a **finite transducer** *with input alphabet X and output alphabet Y if (G, X, S, F, α) is a finite automaton, $G = (V, E)$, and $\beta = (\mathcal{W}(G), Y^*, \beta_1, \beta_2)$ is a functor.*

$\mathcal{T}$ is called **length-preserving** if $|\alpha(e)| = |\beta(e)|$ for all edges $e \in E$.

$\mathcal{T}$ computes the **finite transduction**

$$\begin{aligned} \tau(\mathcal{T}) \quad = \quad & \{(u, v) \in X^* \times Y^* \mid \text{there exists a path } w \in \mathcal{W}(G)(S, F) \\ & \text{with } \alpha(w) = u \text{ and } \beta(w) = v\} \end{aligned}$$

Remark: For each length-preserving transducer $\mathcal{T}$ there exists a transducer $\mathcal{T}'$ with $|\alpha(e)| = |\beta(e)| = 1$ such that $\tau(\mathcal{T}) = \tau(\mathcal{T}')$.

A finite transduction can equivalently be written as

$$\tau(\mathcal{T})(u) = \beta(\alpha^{-1}(u) \cap \mathcal{W}(G)(S, F)), \ u \in X^*$$

We also speak of **rational transduction** instead of finite transduction.

We give an example for a rational transducer:

Let $\mathcal{T} = (G, \{x\}, \{a, b\}, \{1\}, \{3, 5\}, \alpha, \beta)$ be given by the graph

FIGURE

The edges are labeled with the input/output symbols.

$\mathcal{T}$ realizes the transduction $\tau(\mathcal{T})$ defined by

$$\tau(\mathcal{T})(x^n) = \begin{cases} a^n & \text{if } n \equiv 1 \bmod 2 \\ b^n & \text{if } n \equiv 0 \bmod 2 \end{cases}$$

LEMMA 5.1 (closure under inversion). *Let $\tau(\mathcal{T})$ be a rational transduction. Then $(\tau(\mathcal{T}))^{-1}$ is also a rational transduction.*

Proof: Let

$$\mathcal{T} = (G, X, Y, S, F, \alpha, \beta)$$

be a finite tranductor. Then

$$\mathcal{T}' = (G, Y, X, S, F, \beta, \alpha)$$

is a finite transducer and

$$\tau(\mathcal{T}') = (\tau(\mathcal{T}))^{-1}$$

We now define the image of a language under rational transduction:

DEFINITION 5.2. *Let $L \subset X^*$, then*

$$\begin{aligned} \tau(\mathcal{T})(L) \quad := \quad & \{v \in Y^* \mid \textit{there exists } u \in L \textit{ with } (u, v) \in \tau(\mathcal{T})\} \\ := \quad & \bigcup_{u \in L} \tau(\mathcal{T})(u) \end{aligned}$$

The following theorem holds:

THEOREM 5.1 (Regular languages are closed under rational transductions). *Let $L \in REG(X^*)$ be a regular language and $\mathcal{T}$ be a finite transducer, then the image of L under τ is a regular language over Y^*:*

$$\tau(\mathcal{T})(L) \in REG(Y^*)$$

Proof: By definition, the image of L can be presented as

$$\tau(\mathcal{T})(L) = \beta\big(\alpha^{-1}(L) \cap \mathcal{W}(G)(S,F)\big)$$

α may be continued to a monoid homomorphism from E^*, the free monoid over the edge set of graph G, into X^*.

We already know:

$$\begin{array}{rc}
\alpha^{-1}(L) \in REG(E^*) & \text{Theorem 2, chapter 2.3} \\
\mathcal{W}(G)(S,F) \in REG(E^*) & \text{Lemma 4, chapter 2.1} \\
\Rightarrow \alpha^{-1}(L) \cap \mathcal{W}(G)(S,F) \in REG(E^*) & \text{Theorem 1, chapter 2.1} \\
\Rightarrow \beta(\alpha^{-1}(L) \cap \mathcal{W}(G)(S,F)) \in REG(Y^*) & \text{Theorem 7, chapter 2.1}
\end{array}$$

The next theorem gives an insight into the power of rational transductions.

THEOREM 5.2. *For each $L \in REG(Y^*)$ there exists a length-preserving transducer $\mathcal{T}_L$ with*

$$\tau(\mathcal{T}_L)(\{x\}^*) = L$$

Proof: $L \in REG(Y^*) \Rightarrow$ there exists a finite automaton $\mathcal{B} = (G, Y, S, F, \beta)$, $G = (V, E)$ with $L = \beta(\mathcal{W}(G)(S,F))$. For $X = \{x\}$ set $\alpha(e) = x$ for all edges $e \in E$.

Then $\alpha^{-1}(X^*) \cap \mathcal{W}(G)(S,F) = \mathcal{W}(G)(S,F)$, from which follows

$$\beta(\alpha^{-1}(X^*) \cap \mathcal{W}(G)(S,F)) = L$$

THEOREM 5.3. *If $\tau_1 : X^* \to Y^*$ and $\tau_2 : Y^* \to Z^*$ are finite, length-preserving transductions, then $\tau_1 \circ \tau_2 : X^* \to Z^*$ is a finite transduction.*

Proof: We want to illustrate the proof geometrically as shown in the following figure:

$$\begin{tikzcd}
 & & E_3^* \cap R_3 \arrow[dl, "\chi"'] \arrow[dr, "\rho"] & & \\
 & E_1^* \cap R_1 \arrow[dl, "\alpha^{(1)}"'] \arrow[dr, "\beta^{(1)}"] & & E_2^* \cap R_2 \arrow[dl, "\alpha^{(2)}"'] \arrow[dr, "\beta^{(2)}"] & \\
X^* \arrow[rr, "\tau_1"] & & Y^* \arrow[rr, "\tau_2"] & & Z^*
\end{tikzcd}$$

(Translator's remark: I slightly changed names to make the proof more readable)

We have the following situation:

$$\tau_1(u) = \beta^{(1)}\big({\alpha^{(1)}}^{-1}(u) \cap R_1\big)$$

and

$$\tau_2(v) = \beta^{(2)}\big({\alpha^{(2)}}^{-1}(v) \cap R_2\big)$$

with $u \in X^*, v \in Y^*$ and $R_1 \subset E_1^*, R_2 \subset E_2^*$ regular languages and monoid homomorphisms

$$\alpha^{(1)} : E_1^* \to X^*, \quad \beta^{(1)} : E_1^* \to Y^*, \quad \alpha^{(2)} : E_2^* \to Y^*, \quad \beta^{(2)} : E_2^* \to Z^*$$

Let

$$E_3 := \{(e_1, e_2) \mid \beta^{(1)}(e_1) = \alpha^{(2)}(e_2)\}$$

and

$$R_3 := \chi^{-1}(R_1) \times \rho^{-1}(R_2) = \mathcal{W}()(S_1 \times S_2, F_1 \times F_2)$$

with monoid homomorphisms $\chi : e_3^* \to E_1^*$ and $\rho : E_3^* \to E_2^*$.

Now let $(u, v) \in \tau_1$ and $(v, w) \in \tau_2$. Then there exists paths $\omega_1 \in E_1^* \cap R_1$ and $\omega_2 \in E_2^* \cap R_2$ such that

$$u = \alpha^{(1)}(\omega_1), \quad v = \beta^{(1)}(\omega_1) = \alpha^{(2)}(\omega_2), \quad w = \beta^{(2)}(\omega_2)$$

The "parallel" path $\omega = (\omega_1(1), \omega_2(1)) \cdot (\omega_1(n), \omega_2(n))$ is an element of $E_3^* \cap R_3$ and $\chi(\omega) = \omega_1,\ \rho(\omega) = \omega_2$ (without loss of generality the length-restriction from the remark after definition 1 should hold here too).

Together this means

$$(u, v) \in \rho \circ \beta^{(2)}\big(\alpha^{(1)^{-1}} \circ \chi^{-1}(X^*) \cap R_3\big)$$

which means $(u, v) \in \tau_3$.

To the opposite, let $(u, v) \in \tau_3$. Then there exists a path $\omega \in E_3^* \cap R_3$ with $\chi(\alpha^{(1)})(\omega) = u$ and $\rho(\beta^{(2)})(\omega) = v$.

Let $\omega_1 = \chi(\omega)$ and $\omega_2 = \rho(\omega)$ ("projections"), then we get

$$\omega_1 \in E_1^* \cap R_1, \qquad \omega_2 \in E_2^* \cap R_2$$

By construction of E_3 it follows

$$\beta^{(1)}(\omega_1) = \alpha^{(2)}(\omega_2)$$

Corollary 5.1. *Length-preserving transductions with composition form a category.*

We generalize now our statement by allowing α and β to be arbitrary functors to X^* resp. Y^*.

Next, we investigate the case $|\alpha(w)| \leq |w|$ and $|\beta(w)| \leq |w|$. Afterwards we will reduce the general case to this special case.

Notation: Functors fulfilling the condition above are called **non-expanding**.

Theorem 5.4 (Composition of non-expanding transductions is a transduction). *Let $\mathcal{T}_i = (G_i, X_i, S_i, F_i, \alpha^{(i)}, \beta^{(i)}),\ G_i = (V_i, E_i)$ be transducers with non-expanding functors $\alpha^{(i)}$ and $\beta^{(i)},\ i = 1, 2$.*

If $Y_1 = X_2$ (the input alphabet of the second is the output alphabet of the first transducer), then the composition $\tau(\mathcal{T}_1) \circ \tau(\mathcal{T}_2)$ is a transduction.

Proof: Our assumption is described by the following figure:

$$
\begin{tikzcd}
 & E_1^* \cap \mathcal{W}(G_1)(S_1,F_1) \arrow[dl, "\alpha^{(1)}"'] \arrow[dr, "\beta^{(1)}"] & & E_2^* \cap \mathcal{W}(G_2)(S_2,F_2) \arrow[dl, "\alpha^{(2)}"'] \arrow[dr, "\beta^{(2)}"] & \\
X_1^* \arrow[rr, "\tau(\mathcal{T}_1)"] & & Y_1^* \arrow[rr, "\tau(\mathcal{T}_2)"] & & Y_2^*
\end{tikzcd}
$$

Remark: It should be clear that the addition of ϵ-loops ($e \in E, \alpha(e) = \beta(e) = \epsilon$) in the graph does not change the computed transduction.

We add to each state of $\mathcal{T}_1$ and $\mathcal{T}_2$ such a loop.

Because this does not change the computed transductions, this also is the case for the composition of those.

We define the vertex set of the composed transducer as $V := V_1 \times V_2$. The edge set is given by

$$E := \{(e_1, e_2) \in E_1 \times E_2 \mid \alpha^{(2)}(e_2) = \beta^{(1)}(e_1)\}$$

where

$$
\begin{aligned}
Q((e_1, e_2)) &= (Q(e_1), Q(e_2)) \\
Z((e_1, e_2)) &= (Z(e_1), Z(e_2))
\end{aligned}
$$

With the projections $\pi^i : E^* \to E_i^*,\ i = 1, 2$ we get the commutative diagram:

$$
\begin{tikzcd}
 & E^* \arrow[dl, "\pi^1"'] \arrow[dr, "\pi^2"] \arrow[dd, "\sigma"] & \\
E_1^* \arrow[dr, "\beta^{(1)}"'] & & E_2^* \arrow[dl, "\alpha^{(2)}"] \\
 & Y_1^* &
\end{tikzcd}
$$

and it holds: for $v \in \beta^{(1)}(E_1^*) \cap \alpha^{(2)}(E_2^*)$ there exists a path $u \in E^*$ with $\sigma(u) = v$.

Define start and final state sets as follows:

$$
\begin{aligned}
S &= S_1 \times S_2 \\
F &= F_1 \times F_2
\end{aligned}
$$

Then the composed transducer

$$\mathcal{T} = (G, X_1, Y_2, S, F, \pi^1 \circ \alpha^{(1)}, \pi^2 \circ \beta^{(2)})$$

computes the composition of transductions τ_1 and τ_2:

$$\tau(\mathcal{T}) = \tau(\mathcal{T}_1) \circ \tau(\mathcal{T}_2)$$

(Exercise).

In analogy to the corresponding result for finite automata it holds:

Lemma 5.2. *For each transducer*

$$\mathcal{T} = (G, X, Y, S, F, \alpha, \beta)$$

there exists a transducer

$$\mathcal{T}' = (G', X, Y, S', F', \alpha', \beta')$$

with

$$\forall e \in E(G') : \alpha'(e) \in X \cup \{\epsilon\}$$
$$\forall e \in E(G') : \beta'(e) \in Y \cup \{\epsilon\}$$
$$card(S') = 1$$
$$card(F') = 1$$

which computes the same transduction as $\mathcal{T}$.

Proof: Let $\mathcal{T}$ be an arbitrary transducer. If an edge $e \in E$ has an input or output label longer than 1, that is $e \in E$ with $|\alpha(e)| = k > 1$ or $|\beta(e)| = l > 1$ and $m := max(k, l)$, then a path of length m is split as follows:

$$\xrightarrow{e_1}\xrightarrow{e_2}\dots\xrightarrow{e_m}$$

(1) $|\alpha'(e_i)| \leq 1$

(2) $|\beta'(e_i)| \leq 1, \quad i = 1, \dots, m$

(3) $\alpha(e) = \alpha'(e_1 \cdots e_m)$ and $\beta(s) = \beta'(e_1 \cdots e_m)$

For the resulting transducer $\mathcal{T}'$ it holds

$$\tau(\mathcal{T}) = \tau(\mathcal{T}')$$

The construction for making this transducer initial (single start state) is completely analog to the construction in lemma 1 in chapter 2.1.

From lemma 2 and theorem 4 we get directly

Theorem 5.5 (composition of rational transductions is a rational transduction). *For rational transductions τ_1 and τ_2 the composition $\tau_1 \circ \tau_2$, if defined, is also a rational transduction.*

Remark: An equivalent definition of rational transductions can be given using matrices. These are a compact description for transductions which one gets by putting the output words that belong to a fixed input word into the matrix, for all state changes.

Multiplication of matrices corresponds to concatenation of paths in the graph of the transducer and to the union of sets of output words of these paths.

The interested reader should refer to [?] for more information.

6. Homomorphy and equivalence of finite automata

We consider again the finite automaton from chapter 2.1 and investigate the question of structural relationship between finite automata.

We tackle the question how finite automata are related which accept the same language. We will formalize this by the existence of certain functors between finite automata.

For making statements about decidability wrt. accepted languages it is an advantage to be able to determine a "simplest"" finite acceptor. We will prove the existence of such a "simplest" automaton.

First, we consider some simple decidability questions answered by the following theorems:

THEOREM 6.1. *Let $\mathcal{A}$ be a finite automaton. Then the questions $L_{\mathcal{A}} = \emptyset$? and $L_{\mathcal{A}} = X^*$? are decidable.*

Proof:

"$L_{\mathcal{A}} = \emptyset$?":

Because of $L_{\mathcal{A}} = \emptyset \Leftrightarrow \mathcal{W}(G)(S, F) = \emptyset$ this is a decidable problem (reachability in a finite graph). The formal proof is up to the reader.

"$L_{\mathcal{A}} = X^*$?":

We gave in chapter 2.1 an effective procedure for constructing a deterministic, finite automaton $\mathcal{A}'$ from an automaton $\mathcal{A}$. From this deterministic automaton, one can construct an automaton $\mathcal{B}$ accepting the complement of $L_{\mathcal{A}}$. From $L_{\mathcal{B}} = \emptyset \Leftrightarrow L_{\mathcal{A}} = X^*$, and 1) follows the claim.

DEFINITION 6.1. *$\mathcal{A}$ is called* **weakly-equivalent** *to an automaton $\mathcal{B}$ if the accepted languages are equal:*

$$L_{\mathcal{A}} = L_{\mathcal{B}}$$

THEOREM 6.2 (decidability of weak equivalence). *For finite automata $\mathcal{A}$ and $\mathcal{B}$ it is decidable if $L_{\mathcal{A}} = L_{\mathcal{B}}$.*

Proof: Without loss of generality we may assume that $\mathcal{A}$ and $\mathcal{B}$ are complete and deterministic.

From $\mathcal{A}$ and $\mathcal{B}$ using our constructions we can directly give automata for

$$\bar{L_{\mathcal{A}}}, \bar{L_{\mathcal{B}}}, \bar{L_{\mathcal{A}}} \cap L_{\mathcal{B}}, L_{\mathcal{A}} \cap \bar{L_{\mathcal{A}}}$$

and also for

$$L_{\mathcal{C}} := (\bar{L_{\mathcal{A}}} \cap L_{\mathcal{B}}) \cup (L_{\mathcal{A}} \cap \bar{L_{\mathcal{A}}})$$

Because $L_{\mathcal{A}} = L_{\mathcal{B}} \Leftrightarrow L_{\mathcal{C}} = \emptyset$ which is decidable (theorem 1) we get the proof of theorem 2.

We consider now the homomorphy between finite automata. Structure-preserving mappings between finite automata are called **automata homomorphisms**.

DEFINITION 6.2. *Let $\mathcal{A}, \mathcal{A}'$ be finite automata. A functor $\beta = (\beta', \beta'')$ is called an* **automata homomorphism**, *if*

$$\beta' = (\mathcal{W}(G), \mathcal{W}(G'), \beta_1', \beta_2')$$

is a functor,

$$\beta'' : X^* \to X'^*$$

is a monoid homomorphism, and the axioms (A1) and (A2) hold:

(A1) *The following diagram commutes:*

$$\begin{tikzcd}
\mathcal{W}(G) \arrow[r, "\alpha"] \arrow[d, "\beta'"] & X^* \arrow[d, "\beta''"] \\
\mathcal{W}(G') \arrow[r, "\alpha'"] & X'^*
\end{tikzcd}$$

(A2) $\beta'_1(S) = S'$ *and* $\beta'_1(F) = F'$.

β *is called* **length-preserving** $\Leftrightarrow \beta'_2(e) \in E(G')$ *for each edge* $e \in E(G)$.

β *is called* **non-expanding** $\Leftrightarrow \beta'_2(e) \in E(G') \cup \{1_e \mid e \in E(G')\}$

In the following we will always identify β' with β.

It holds:

THEOREM 6.3. *Given an automata homomorphism* $\beta : \mathcal{A} \to \mathcal{A}'$. *If*

$$\beta_2(\mathcal{W}(G)(S, F)) = \mathcal{W}(G')(S', F')$$

then $\beta''(L_{\mathcal{A}}) = L_{\mathcal{A}'}$.

Proof:

"$\subset$":

Let $u \in L_{\mathcal{A}}$, then there exists a path $\pi \in \mathcal{W}(G)(S, F)$ labelled with u, i.e. $\alpha(\pi) = u$. Because $\beta_2(\pi) \in \mathcal{W}(G')(S', F')$ by assumption, it follows $\alpha'(\beta_2(\pi)) \in L_{\mathcal{A}'}$.

Because of the commutativity of the diagram it follows $\alpha'(\beta_2(\pi)) = \beta''(\alpha(\pi))$.

$$\Rightarrow \beta''(L_{\mathcal{A}}) \subset L_{\mathcal{A}'}$$

"$\supset$":

Let $u' \in L_{\mathcal{A}'}$. Then there exists a path $\pi' \in \mathcal{W}(G')(S', F')$ that is labeled with u': $\alpha'(\pi') = u'$.

Because β_2 is surjective on $\mathcal{W}(G')(S', F')$ it follows: there exists a path π with $\beta_2(\pi) = \pi'$ and because β is a functor it follows $\alpha(\pi) \in L_{\mathcal{A}}$.

By definition, $\beta''(\alpha(\pi)) = \alpha'(\beta_2(\pi)) = \alpha'(\pi') = u'$, therefore $u' \in \beta''(L_{\mathcal{A}})$.

$$\Rightarrow \beta''(L_{\mathcal{A}}) \supset L_{\mathcal{A}'}$$

Together this proves

$$L_{\mathcal{A}'} = \beta''(L_{\mathcal{A}})$$

DEFINITION 6.3 (Reduction, closed homomorphism). *Let* $\mathcal{A}$ *and* $\mathcal{A}'$ *be finite automata.*

A homomorphism $\beta : \mathcal{A} \to \mathcal{A}'$ *is called a* **reduction** *from* $\mathcal{A}$ *to* $\mathcal{A}'$ *if*

$$\beta_2(\mathcal{W}(G)(S, F)) = \mathcal{W}(G')(S', F')$$

and $\beta'' = id_{X^*}$.

β *is a* **length-preserving reduction**, *if it is a reduction and the homomorphisms is length-preserving.*

β is a **non-expanding reduction***, if it is a reduction and the homomorphisms is non-expanding.*

β is called a **closed homomorphism** *or short:* **closed***, if it is a homomorphism and it holds*

$$\beta_1^{-1}(S') = S \quad \wedge \quad \beta_1^{-1}(F') = F$$

In analogy, one defines a **closed reduction***.*

We immediately get the following corollary to theorem 3:

Corollary 6.1.

$$\beta : \mathcal{A} \to \mathcal{A}' \textit{ reduction } \Rightarrow L_{\mathcal{A}} = L_{\mathcal{A}'}$$

Our goal will be to transfer equivalent automata into each other using chains of reductions.

To reach this goal, we will need a number of lemmata.

Lemma 6.1. *Let $\mathcal{A}, \mathcal{B}, \mathcal{C}$ be complete, deterministic finite automata and let $\beta_{\mathcal{A}} : \mathcal{C} \to \mathcal{A}$ and $\beta_{\mathcal{B}} : \mathcal{C} \to \mathcal{B}$ be closed reductions. Then there exists a finite automaton $\mathcal{D}$ and reductions $\gamma_{\mathcal{A}}$ and $\gamma_{\mathcal{B}}$ such that the following diagram commutes:*

$$\begin{tikzcd}
& \mathcal{C} \arrow[dl, "\beta_{\mathcal{A}}"'] \arrow[dr, "\beta_{\mathcal{B}}"] \arrow[dd, dashed, "\delta"] & \\
\mathcal{A} \arrow[dr, dashed, "\gamma_{\mathcal{A}}"'] & & \mathcal{B} \arrow[dl, dashed, "\gamma_{\mathcal{B}}"] \\
& \mathcal{D} &
\end{tikzcd}$$

Proof: The idea of the proof is as follows: We create from $\mathcal{C}$ a new automaton $\mathcal{D}$ by putting those vertices and edges into classes which will be identified under the reductions $\beta_{\mathcal{A}}$ or $\beta_{\mathcal{B}}$. In that way we will get the reduction δ. The difficulty that arises in identifying the edges is that the labelling of the edges has to be respected.

To realize out idea we define equivalence relations on the edges and vertices of the graph $G_{\mathcal{C}} = (V_{\mathcal{C}}, E_{\mathcal{C}})$ of $\mathcal{C}$:

Let $v_1, v_2 \in V_{\mathcal{C}}$ be vertices (states). Define

$$\begin{aligned} v_1 \underset{\mathcal{A}}{\equiv} v_2 \quad &:\Longleftrightarrow \quad \beta_{\mathcal{A}}(v_1) = \beta_{\mathcal{A}}(v_2) \\ v_1 \underset{\mathcal{B}}{\equiv} v_2 \quad &:\Longleftrightarrow \quad \beta_{\mathcal{B}}(v_1) = \beta_{\mathcal{B}}(v_2) \end{aligned}$$

In analogy we define such an equivalence relation for the edges of the graph $G_{\mathcal{C}}$.

We define now an equivalence relation on the vertices (states) of the automaton $\mathcal{C}$. For $v, v' \in V_{\mathcal{C}}$:

$$\begin{aligned} v \equiv v' \quad :\Leftrightarrow \quad & \text{there is a chain of states } v = v_1, v_2, \ldots, v_k = v' \text{ with} \\ & v_i \underset{\mathcal{A}}{\equiv} v_{i+1} \quad \text{or} \quad v_i \underset{\mathcal{B}}{\equiv} v_{i+1}, \quad i = 1, \ldots, k \end{aligned}$$

Obviously this defines an equivalence relation on the vertex (state) set $V_{\mathcal{C}}$. We do this in the same way for the edges.

We get equivalence classes

$$\begin{aligned}[v] &:= \{v' \in V_{\mathcal{C}} \mid v \equiv v'\} \\ [e] &:= \{e' \in E_{\mathcal{C}} \mid e \equiv e'\}\end{aligned}$$

We define the sets of equivalence classes by

$$\begin{aligned}\bar{V} &:= \{[v] \mid v \in V_{\mathcal{C}}\} \\ \bar{E} &:= \{[e] \mid e \in E_{\mathcal{C}}\}\end{aligned}$$

Claim: With $Q([e]) := [Q(e)]$ and $Z([e]) := [Z(e)],\ e \in E_{\mathcal{C}}$ we get well-defined source and target mappings for the edges of the graph of the automaton $\mathcal{C}$.

Proof. Let $e' \in [e]$. To prove that $Q(e') \in [Q(e)]$, we use induction over the length of the chain $e = e_1, \ldots, e_k = e'$ where $s_i \underset{\mathcal{A}}{\equiv} s_{i+1}$ or $s_i \underset{\mathcal{B}}{\equiv} s_{i+1}$ for all $i = 1, \ldots, k-1$.

It is sufficient to show that the claim is correct for each single step. Because $\beta_{\mathcal{A}}$ and $\beta_{\mathcal{B}}$ are functors, it holds

$$e_i \underset{\mathcal{A}}{\equiv} e_{i+1} \Rightarrow Q(e_i) \underset{\mathcal{A}}{\equiv} Q(e_{i+1})$$

and

$$e_i \underset{\mathcal{B}}{\equiv} e_{i+1} \Rightarrow Q(e_i) \underset{\mathcal{B}}{\equiv} Q(e_{i+1})$$

From this it follows that Q and Z are well-defined on the equivalence classes.

Let $\bar{G} = (\bar{V}, \bar{E})$ be the graph defined by the sets of vertex and edge equivalence classes. Define the labelling $\alpha_{\mathcal{D}} : \bar{E} \to X^*$ for the edges of this graph by

$$\alpha_{\mathcal{D}}([e]) := \alpha_{\mathcal{C}}(e)$$

This mapping is well-defined.

Proof: Because $\beta_{\mathcal{A}}$ is a reduction, it holds:

$$\alpha_{\mathcal{A}}(\beta_{\mathcal{A}}(e) = \beta''(\alpha_{\mathcal{C}}(e)) = \alpha_{\mathcal{C}}(e)$$

This means that for each $e' \in [e]_{\mathcal{A}}$ we have $\alpha_{\mathcal{A}}(\beta_{\mathcal{A}}(e')) = \alpha_{\mathcal{C}}(e)$. This our claim is correct if $e \underset{\mathcal{A}}{\equiv} e'$. The same holds for $e \underset{\mathcal{B}}{\equiv} e'$. Together we get that this holds for all $e \equiv e'$ so $\alpha_{\mathcal{D}}$ is well-defined.

To complete the definition of the automaton $\mathcal{D}$ we have to define start and final state sets.

$$\mathcal{D} := (\bar{G}, X, [S_{\mathcal{C}}], [F_{\mathcal{C}}], \alpha_{\mathcal{D}})$$

with final state set $[F_{\mathcal{C}}] := \{[f] \mid f \in F_{\mathcal{C}}\}$.

Claim: The canonical mapping $e \to [e],\ v \to [v]$ is a reduction.

Proof: Exercise (Hint: use that automaton is complete and deterministic)

Finally we define the reductions $\gamma_{\mathcal{A}}$ and $\gamma_{\mathcal{B}}$ by

$$\begin{aligned}\gamma_{\mathcal{A}}(v) &:= [\beta_{\mathcal{A}}^{-}1](v), \quad v \in V_{\mathcal{A}} \\ \gamma_{\mathcal{B}}(v) &:= [\beta_{\mathcal{B}}^{-}1](v), \quad v \in V_{\mathcal{B}} \\ \gamma_{\mathcal{A}}(e) &:= [\beta_{\mathcal{A}}^{-}1](e), \quad e \in E_{\mathcal{A}} \\ \gamma_{\mathcal{B}}(e) &:= [\beta_{\mathcal{B}}^{-}1](e), \quad e \in E_{\mathcal{B}}\end{aligned}$$

□

Remark: The requirement of our lemma, that the automata have to be complete, is essential, as the following example shows:

FIGURE

It holds: $L_{\mathcal{A}} = L_{\mathcal{B}} = L_{\mathcal{C}} = \{a^n \mid n \in \mathbb{N}\} \cup \{b^n \mid n \in \mathbb{N}\}$.

From our construction it follows $4 \equiv 2 \equiv 1 \equiv 3$ and the automaton $\mathcal{D}$ has the form

FIGURE

But $L_{\mathcal{D}} = \{a, b\}^* \neq L_{\mathcal{C}}$. It follows:

There exists no reduction from $\mathcal{C}$ to $\mathcal{D}$ because $\mathcal{A}$, $\mathcal{B}$ and $\mathcal{C}$ are not complete *and* deterministic.

The motivation for lemma 1 can be easily seen in the following diagram:

Let

$$\begin{tikzcd}
 & \mathcal{A}_2 \arrow[dl] \arrow[dr] & & \mathcal{A}_4 \arrow[dl] \arrow[dr] & & \mathcal{A}_6 \arrow[dl] \\
\mathcal{A}_1 & & \mathcal{A}_3 & & \mathcal{A}_5 &
\end{tikzcd}$$

be a chain of reductions. Then we can construct automata $\mathcal{A}_7, \mathcal{A}_8$ and reductions like

$$\begin{tikzcd}
 & \mathcal{A}_2 \arrow[dl] \arrow[dr] & & \mathcal{A}_4 \arrow[dl] \arrow[dr] & & \mathcal{A}_6 \arrow[dl] \\
\mathcal{A}_1 \arrow[dr, dashed] & & \mathcal{A}_3 \arrow[dl, dashed] & & \mathcal{A}_5 \arrow[ddll, dashed] & \\
 & \mathcal{A}_7 \arrow[dr, dashed] & & & & \\
 & & \mathcal{A}_8 & & &
\end{tikzcd}$$

such that the diagram can be "shortened" to

$$\begin{tikzcd}
\mathcal{A}_1 \arrow[dr] & & \mathcal{A}_6 \arrow[dl] \\
 & \mathcal{A}_8 &
\end{tikzcd}$$

LEMMA 6.2. *Let $\gamma_{\mathcal{A}} : \mathcal{A} \to \mathcal{D}$ and $\gamma_{\mathcal{B}} : \mathcal{B} \to \mathcal{D}$ be length-preserving reductions.*

Then there exists an automaton $\mathcal{C}$ and reductions $\beta_{\mathcal{A}}$ and $\beta_{\mathcal{B}}$ such that the following diagram commutes:

$$\begin{tikzcd}
 & \mathcal{C} \arrow[dl, dashed, "\beta_{\mathcal{A}}"'] \arrow[dr, dashed, "\beta_{\mathcal{B}}"] \arrow[dd, dashed] & \\
\mathcal{A} \arrow[dr, "\gamma_{\mathcal{A}}"'] & & \mathcal{B} \arrow[dl, "\gamma_{\mathcal{B}}"] \\
 & \mathcal{D} &
\end{tikzcd}$$

PROOF. We construct the automaton

$$\mathcal{C} = (G_{\mathcal{C}}, X, S_{\mathcal{C}}, F_{\mathcal{C}}, \alpha_{\mathcal{C}})$$

with vertex (state) set

$$V_{\mathcal{C}} = V_{\mathcal{A}} \times V_{\mathcal{B}}$$

and edge set

$$E_{\mathcal{C}} = \{(e, e') \in E_{\mathcal{A}} \times E_{\mathcal{B}} \mid \gamma_{\mathcal{A}}(e) = \gamma_{\mathcal{B}}(e')\}$$

start states

$$S_{\mathcal{C}} = S_{\mathcal{A}} \times S_{\mathcal{B}}$$

final states

$$F_{\mathcal{C}} = F_{\mathcal{A}} \times F_{\mathcal{B}}$$

and labelling

$$\alpha_{\mathcal{C}}((e, e')) = \alpha_{\mathcal{A}}(e)$$

$\beta_{\mathcal{A}}$ and $\beta_{\mathcal{B}}$ are the projections onto $\mathcal{A}$ and $\mathcal{B}$.

Claim: $\beta_{\mathcal{A}}$ and $\beta_{\mathcal{B}}$ are reductions.

Without loss of generality we show the claim for $\beta_{\mathcal{A}}$ only.

(1) $\alpha_{\mathcal{A}}(\beta_{\mathcal{A}}((e, e'))) = \alpha_{\mathcal{A}}(e) = \beta''_{\mathcal{A}}(\alpha_{\mathcal{C}})(e, e') = \beta''_{\mathcal{A}}(\alpha_{\mathcal{A}})(e)$

With $\beta''_{\mathcal{A}}$ the equation above holds.

(2) $\beta_{\mathcal{A}}(S_{\mathcal{C}}) = S_{\mathcal{A}}$ and $\beta_{\mathcal{A}}(F_{\mathcal{C}}) = F_{\mathcal{A}}$ holds by definition

(3) $\beta_{\mathcal{A}}\Big(\mathcal{W}(G_{\mathcal{C}})(S_{\mathcal{C}}, F_{\mathcal{C}})\Big) \stackrel{!}{=} \mathcal{W}(G_{\mathcal{A}})(S_{\mathcal{A}}, F_{\mathcal{A}})$

Let $\pi \in \mathcal{W}(G_{\mathcal{A}})(S_{\mathcal{A}}, F_{\mathcal{A}})$ be an accepting path. We have to show that there exists an accepting path $\pi' \in \mathcal{W}(G_{\mathcal{C}})(S_{\mathcal{C}}, F_{\mathcal{C}})$ with $\beta_{\mathcal{A}}(\pi') = \pi$.

Because $\gamma_{\mathcal{A}}$ is a reduction, there exists an accepting path $\pi_{\mathcal{D}} \in \mathcal{W}()\mathcal{D}(S_{\mathcal{D}}, F_{\mathcal{D}})$ with $\gamma_{\mathcal{A}}(\pi) = \pi_{\mathcal{D}}$.

Further there exists an accepting path $\pi_{\mathcal{B}} \in \mathcal{W}(G_{\mathcal{B}})(S_{\mathcal{B}}, F_{\mathcal{B}})$ with $\gamma_{\mathcal{B}}(\pi_{\mathcal{B}}) = \pi_{\mathcal{D}}$, because $\gamma_{\mathcal{B}}$ is a reduction (and therefore is surjective).

From the paths π and $\pi_{\mathcal{B}}$ we construct the path we are searching for:

$$\pi = e_1 \cdots e_n \qquad \pi_{\mathcal{D}} = \gamma_{\mathcal{A}}(e_1) \cdots \gamma_{\mathcal{A}}(e_n) = \gamma_{\mathcal{B}}(\pi_{\mathcal{B}})$$

Because $\gamma_{\mathcal{A}}$ and $\gamma_{\mathcal{B}}$ are length-preserving it follows

$$\pi_{\mathcal{B}} = e'_1 \cdots e'_n$$

with $e'_i \in E_{\mathcal{B}}$.

Now create the path $\pi_{\mathcal{C}} = (e_1, e'_1) \cdots (e_n, e'_n) \in \mathcal{W}(G_{\mathcal{C}})(S_{\mathcal{C}}, F_{\mathcal{C}})$.

$$\Rightarrow \mathcal{W}(G_{\mathcal{A}})(S_{\mathcal{A}}, F_{\mathcal{A}}) \subset \beta_{\mathcal{A}}(\mathcal{W}()\mathcal{C}(S_{\mathcal{C}}, F_{\mathcal{C}}))$$
$$\Rightarrow \mathcal{W}(G_{\mathcal{A}})(S_{\mathcal{A}}, F_{\mathcal{A}}) = \beta_{\mathcal{A}}(\mathcal{W}()\mathcal{C}(S_{\mathcal{C}}, F_{\mathcal{C}}))$$

□

We want to treat again the question from the beginning of this section: How do automata relate which accept the same language?

First, we want to render more precisely: How does a non-deterministic automaton $\mathcal{A}$ relate to the complete, deterministic automaton $\mathcal{A}$'?

We remember the constructions from the first section.

At first, we want to start with an ϵ-free automaton with $\alpha(e) \in X$ for each edge e.

Consider the step which makes this automaton initial (single start state). The construction was as follows:

FIGURE

Let $\mathcal{A}_1$ be the initial automaton with start state Z_0.

We construct an automaton $\mathcal{A}_1'$ by adding for each start state $i \in S_{\mathcal{A}}$ a state $i' \in V_G$ and setting $S_{\mathcal{A}'} = \{1', \ldots, k'\}$:

FIGURE

The homomorphism $\phi : \mathcal{A}_1' \to \mathcal{A}$ is defined by $\phi_1(i) = \phi_1(i') = i,\ i = 1, \ldots, k$ and the identity otherwise. Then ϕ is a reduction.

The homomorphism $\phi' : \mathcal{A}_1' \to \mathcal{A}$ is defined by $\phi_1'(i') = Z_0,\ i = 1, \ldots, k$ and $\phi_1'(i) = i$. ϕ' is also a reduction such that the step of make an automaton initial can be described by the following reduction diagram:

$$\begin{tikzcd} & \mathcal{A}_1' \arrow[dl, "\phi"'] \arrow[dr, "\phi'"] & \\ \mathcal{A} & & \mathcal{A}_1 \end{tikzcd}$$

The construction of the final states is totally similar.

As the next step we want to make our initial automaton $\mathcal{A}_1$ complete.

This is realized by introducing a new vertex (state) ω to which from all "incomplete" vertices new edges with the corresponding label will be drawn. Each edge starting from ω also ends in ω.

Let's name the automaton which results from this step with $\mathcal{A}_2$. Obviously we get a reduction $\phi'' : \mathcal{A}_1 \to \mathcal{A}_2$.

Now we make $\mathcal{A}_2$ deterministic. The resulting automaton shall be named $\mathcal{A}'$.

We construct an automaton $\mathcal{A}_2'$ by constructing edges of the following form:

$$e := (\{p_1, \ldots, p_r\}, p_i) \xrightarrow{x} (\{q_1, \ldots, q_s\}, q_j)$$

if $p_i \xrightarrow{e'} q_j \in E_{\mathcal{A}_{\in}}$ and $\alpha(e') = x$ for all $i = 1, \ldots, r,\ j = 1, \ldots, s$ in the construction of the deterministic automaton.

We get homomorphisms

$$\tilde{\phi} \text{ with } \tilde{\phi}(e) := (p_i \xrightarrow{x} q_j)$$

and

$$\hat{\phi} \text{ with } \hat{\phi}(e) := (\{p_1, \ldots, p_r\} \xrightarrow{x} \{q_1, \ldots, q_s\})$$

for all edges e as described above.

Both homomorphisms are reductions.

If we compose all these reductions, we get the following diagram:

$$\begin{tikzcd}
& \mathcal{A}_1' \arrow[dl] \arrow[dr] & & & & \\
\mathcal{A} & & \mathcal{A}_1 \arrow[dr] & & \mathcal{A}_2' \arrow[dl] \arrow[dr] & \\
& & & \mathcal{A}_2 & & \mathcal{A}'
\end{tikzcd}$$

Because the composition of reductions is also a reduction, we can change this diagram as follows:

$$\begin{tikzcd}
& \mathcal{A}_1' \arrow[dl] \arrow[dr] & & \mathcal{A}_2' \arrow[dl] \arrow[dr] & \\
\mathcal{A} & & \mathcal{A}_2 & & \mathcal{A}'
\end{tikzcd}$$

Because all reductions in this diagram are length-preserving, we can complete this diagram to:

$$\begin{tikzcd}
& & \mathcal{C} \arrow[dl, dashed] \arrow[dr, dashed] & & \\
& \mathcal{A}_1' \arrow[dl] \arrow[dr] & & \mathcal{A}_2' \arrow[dl] \arrow[dr] & \\
\mathcal{A} & & \mathcal{A}_2 & & \mathcal{A}'
\end{tikzcd}$$

and finally to

$$\begin{tikzcd}
& \mathcal{C} \arrow[dl] \arrow[dr] & \\
\mathcal{A} & & \mathcal{A}'
\end{tikzcd}$$

Remark: All the reductions that we constructed for make an automaton initial, complete and deterministic have been length-preserving. In chapter 2.1 we additionally made a construction which transforms an automaton with labelling $\alpha(e) \in X^*$ into an automaton with $\alpha(e) \in X$. We had to split edges and make the automaton ϵ-free. We can represent this also by reduction diagrams but in that case the reductions are are not length-preserving (non-expanding) anymore.

The consequence is that lemma 2 cannot be applied anymore.

We now describe these reductions informally.

(1) Splitting of edges:

$$\xrightarrow{x_1 \cdots x_k} \qquad \xrightarrow{\phi} \qquad \xrightarrow{x_1} \;\cdots\; \xrightarrow{x_k}$$

Here, an edge is mapped to a path. The reduction properties are fulfilled.

(2) Removal of ϵ-edges after insertion of bridges:

FIGURE

$\phi(e_1') = e_1 \cdot e_2, \quad Q(e_1') = Q(e_1), \quad Z(e_1') = Z(e_2), \quad \phi(e_2) = e_2$, identity otherwise.

Here the properties of a reduction are also fulfilled.

Let's visualize the complete construction graphically:

$$\begin{tikzcd}
 & & & & \tilde{\mathcal{A}} \arrow[dl, dashed] \arrow[ddrr, dashed] & & & \\
 & & & \mathcal{A}_2'' \arrow[dl] \arrow[dr] & & & & \\
\mathcal{A} \arrow[dr] & & \mathcal{A}_2 \arrow[dl] & & \mathcal{A}_2' \arrow[dr] & & \mathcal{A}_3'' \arrow[dl] \arrow[dr] & \\
 & \mathcal{A}_1 & & & & \mathcal{A}_3 & & \mathcal{A}'
\end{tikzcd}$$

This diagram can be simplified after our previous considerations and using the categorial closure of length-preserving reductions into:

$$\begin{tikzcd}
 & & & \tilde{\mathcal{A}} \arrow[dl] \arrow[dr] & \\
\mathcal{A} \arrow[dr] & & \mathcal{A}_2 \arrow[dl] & & \mathcal{A}' \\
 & \mathcal{A}_1 & & &
\end{tikzcd}$$

We can now concretize our initial goal: describing the relation between two equivalent automata.

We want to achieve the following:

$L_{\mathcal{A}} = L_{\mathcal{B}}$, then there should exist a diagram

FIGURE

We specify for the complete, deterministic automaton a normal-form and define

Definition 6.4 (reduced automaton). *A complete, deterministic automaton is called* **reduced** $\iff$ *each reduction* $\rho : \mathcal{A} \to \mathcal{B}$ *is a monomorphism (injective homomorphism).*

Lemma 6.3. *For each complete, deterministic finite automaton* $\mathcal{A}$ *there exists a reduced automaton* $\mathcal{A}_r$ *and a closed reduction* $\rho : \mathcal{A} \to \mathcal{A}_r$.

If $\mathcal{A}_r$ *does not contain superfluous elements, then it is unique up to isomorphism.*

Proof. We prove first that every two automata which are reduced and belong to $\mathcal{A}$ are isomorphic.

Let $\rho_i : \mathcal{A} \to \mathcal{A}_i,\ i = 1, 2$ be closed reductions. We can apply lemma 1 and get the following situation:

There exist a reduced automaton $\mathcal{A}_r$ and reductions $\psi_i : \mathcal{A}_i \to \mathcal{A}_r$ with

$$\begin{tikzcd} & \mathcal{A} \arrow[dl, "\rho_1"'] \arrow[dr, "\rho_2"] & \\ \mathcal{A}_1 \arrow[dr, "\psi_1"'] & & \mathcal{A}_2 \arrow[dl, "\psi_2"] \\ & \mathcal{A}_r & \end{tikzcd}$$

By assumption, $\mathcal{A}_1$ and $\mathcal{A}_2$ are reduced, therefore ψ_1 and ψ_2 are monomorphisms (injective) and ρ_1 and ρ_2 are surjective, $\mathcal{A}_1$ and $\mathcal{A}_2$ are complete and deterministic

$\Rightarrow$ the images under ψ_1 and ψ_2 are equal

$\Rightarrow \psi_1, \psi_2$ are isomorphisms on the images. $\square$

It remains to prove that for each automaton $\mathcal{A}$ there exists a reduces automaton $\mathcal{A}_r$.

This follows from the fact that the automata are finite and is shown by the following construction.

Remark: For simplicity (without loss of generality) we want to assume that our automata in the following do not contain superfluous vertices (states).

We consider now a restriction of the syntactic congruence defined in chapter 1.2.

Definition 6.5. *$u, v \in X^*$ are called* **congruent modulo** *L, written as*

$$u \equiv_L v :\iff \textit{ for all } w \in X^* \textit{ holds } \quad u \cdot w \in L \iff v \cdot w \in L$$

Notation: $[w]_L := \{v \mid w \equiv_L v\}$

$$X^*/L := \{[w]_L \mid w \in X^*\}$$

With this notatation it holds

Lemma 6.4.

$$card(X^*/L) \textit{ is finite } \iff L \in REG(X^*)$$

Proof. "$\Rightarrow$":

One sees immediately: $w \equiv_L w' \Rightarrow w \cdot x \equiv_L w' \cdot x$

We construct the graph $G' = (X^*/L, E)$ with edge set

$$E = \{([w], [wx]) \mid w \in X^*,\ x \in X\}$$

which by assumption is finite.

Set $S := \{[e]\},\ F := \{[w] \mid w \in L\}$ and define the labelling α by $\alpha(([w], [wx])) := x$.

We obtain a finite automaton

$$\mathcal{A} = \mathcal{A}(L) = (G', X, S, F, \alpha)$$

with the property $L_{\mathcal{A}} = L$ (exercise).

"$\Leftarrow$":

(Translator remark: Made the proof more readable)

Let $L \in REG(X^*)$ be a regular language, then there exists a finite acceptor $\mathcal{A}$ for L. Assume, that $\mathcal{A}$ is complete and deterministic with graph $G = (V, E)$.

Consider paths π and π' both starting at some start state of $\mathcal{A}$

$$Q(\pi) \in S,\ Q(\pi') \in S$$

with labels being congruent modulo L:

$$\alpha(\pi) \equiv_L \alpha(\pi')$$

For any pair of paths ω, ω' that can be appended to π and π' and which have equal labels, that is

$$Q(\omega) = Z(\pi),\ Q(\omega') = Z(\pi'),\ \alpha(\omega) = \alpha(\omega')$$

it holds:

$$Z(\omega) \in F \iff Z(\omega') \in F$$

(Otherwise, we would have for example $Z(\omega) \in F,\ Z(\omega') \notin F$. But from this would follow $\alpha(\pi) \cdot \alpha(\omega) \in L$ and $\alpha(\pi') \cdot \alpha(\omega) \notin L$ which violates our assumption.)

Now assume to the opposite that for each pair of paths ω, ω' that can be appended to π and π' and $\alpha(\omega) = \alpha(\omega')$ it holds

$$Z(\omega) \in F \iff Z(\omega') \in F$$

Then for any pair of paths π, π' with

$$Z(\pi) = Q(\omega),\ Z(\pi') = Q(\omega'), \quad Q(\pi) \in S,\ Q(\pi') \in S$$

it follows

$$\alpha(\pi) \equiv_L \alpha(\pi')$$

This gives us a mapping that assigns to each vertex (state) $v \in V$ a congruence class, namely $[\alpha(\pi)]$.

From this we get: $card(X^*/L)$ is finite, especially $card(X^*/L) \leq card(V)$. □

In our lemma we even proved a bit more:

Let $\rho_1 : V \to X^*/L$ with $\rho_1(v) = [u]_L$, if $\pi \in \mathcal{W}(G)(S,V)$ where $Z(\pi) = v$ and $\alpha(\pi) = u$ be the mapping from the lemma. Then it holds

LEMMA 6.5. *ρ_1 induces a reduction from $\mathcal{A}$ to $\mathcal{A}(L)$.*

PROOF. Let $e \in E$ be an edge with $Q(e) = v, Z(e) = v'$ and $\alpha(e) = x$, then the mapping ρ_2 with

$$\rho_2(e) = e' \in E(G'),\ Q(e') = [u] = \rho_1(v),\ Z(e') = [ux] = \rho_1(v')$$

induces a reduction. □

We get immediately

THEOREM 6.4. *If $\mathcal{A}$ is a complete, deterministic finite automaton, then there exists a closed reduction*

$$\rho : \mathcal{A} \to \mathcal{A}_L$$

PROOF. Let ρ be the mapping defined by ρ_1 and ρ_2. It holds:

(1) ρ is a homomorphism

(2) $\rho_1^{-1}([L]) = F, \qquad \rho_1^{-1}([\epsilon]) = S$

(3) ρ is surjective!

ρ_1: clear, because to each $v \in V$ there is a corresponding congruence class.

ρ_2 is surjective because $\mathcal{A}$ is complete. $\square$

Remark: ρ is even length-preserving.

COROLLARY 6.2. *$\mathcal{A}(L)$ is reduced. (Exercise)*

With all this we can set $\mathcal{A}_r := \mathcal{A}(L)$ and we have proved lemma 3 completely.

The question remains: how do equivalent, reduced, complete and deterministic finite automata relate?

THEOREM 6.5. *If $\mathcal{A}$ and $\mathcal{B}$ are reduced, complete and deterministic finite automata which accept the same language, then they are isomorphic.*

PROOF. If $\mathcal{A}$ and $\mathcal{B}$ are reduced, then $\mathcal{A} \cong \mathcal{A}(L) \cong \mathcal{B}$. $\square$

COROLLARY 6.3. *If $\mathcal{A}$ is equivalent to $\mathcal{B}$ then there exists a chain of reductions*

$$\mathcal{A} \longrightarrow \mathcal{A}_1 \longleftarrow \mathcal{A}_2 \longleftarrow \mathcal{C} \longrightarrow \mathcal{B}_1 \longleftarrow \mathcal{B}$$

PROOF. We look at the construction from figure TODO:

$$\begin{tikzcd}
 & & & & & \mathcal{C} \arrow[dll, dashed] \arrow[drr, dashed] & & & & & \\
 & & & \tilde{\mathcal{A}} \arrow[dl] \arrow[dr] & & & & \tilde{\mathcal{B}} \arrow[dl] \arrow[dr] & & & \\
\mathcal{A} \arrow[r] & \mathcal{A}_1 & \mathcal{A}_2 \arrow[l] & & \mathcal{A}' \arrow[dr] & & \mathcal{B}' \arrow[dl] & & \mathcal{B}_2 \arrow[r] & \mathcal{B}_1 & \mathcal{B} \arrow[l] \\
 & & & & & \mathcal{A}_r & & & & &
\end{tikzcd}$$

$\mathcal{A}_r$ exists by theorems 4 and 5. $\mathcal{C}$ exists by lemma 2. $\square$

LEMMA 6.6. *Let $\gamma_{\mathcal{A}} : \mathcal{C} \to \mathcal{A}$, $\gamma_{\mathcal{B}} : \mathcal{C} \to \mathcal{B}$ be length-preserving reductions.*

Then there exists an automaton $\mathcal{A} \times \mathcal{B}$ and reductions

$$\pi_{\mathcal{A}} : \mathcal{A} \times \mathcal{B} \to \mathcal{A}$$
$$\pi_{\mathcal{B}} : \mathcal{A} \times \mathcal{B} \to \mathcal{B}$$

such that the following diagram commutes:

$$\begin{tikzcd}
 & \mathcal{C} \arrow[dl, "\gamma_{\mathcal{A}}"'] \arrow[dr, "\gamma_{\mathcal{B}}"] \arrow[dd, "\delta"] & \\
\mathcal{A} & & \mathcal{B} \\
 & \mathcal{A} \times \mathcal{B} \arrow[ul, "\pi_{\mathcal{A}}"] \arrow[ur, "\pi_{\mathcal{B}}"'] &
\end{tikzcd}$$

PROOF. Let

$$G := (V_{\mathcal{A}} \times V_{\mathcal{B}}, \{(e, e') \in E_{\mathcal{A}} \times E_{\mathcal{B}} \mid \alpha_{\mathcal{A}}(e) = \alpha_{\mathcal{B}}(e')\})$$

Then the finite automaton $\mathcal{A} \times \mathcal{B}$ is defined by

$$\mathcal{A} \times \mathcal{B} := (G, X, S_{\mathcal{A}} \times S_{\mathcal{B}}, F_{\mathcal{A}} \times F_{\mathcal{B}}, \alpha)$$

Let $\delta(v) = (\gamma_{\mathcal{A}}(v), \gamma_{\mathcal{B}}(v)),\ v \in V_{\mathcal{C}}$, and $\delta(e) = \gamma_{\mathcal{A}}(e), \gamma_{\mathcal{B}}(e)),\ e \in E_{\mathcal{C}}$.

δ is a homomorphism from $\mathcal{C}$ to $\mathcal{A} \times \mathcal{B}$.

Now $\gamma_{\mathcal{A}} = \delta \circ \pi_{\mathcal{A}}$ and $\gamma_{\mathcal{B}} = \delta \circ \pi_{\mathcal{B}}$ are reductions

$\Rightarrow \pi_{\mathcal{A}}$ and $\pi_{\mathcal{B}}$ are reductions.

□

With lemma 6 we get

THEOREM 6.6. *If $\mathcal{A}$ is equivalent to $\mathcal{B}$, then $\mathcal{A}$ can be transformed into $\mathcal{B}$ by the following chain of reductions:*

$$\mathcal{A} \longrightarrow \mathcal{A}_1 \longleftarrow \mathcal{A}_2 \xleftarrow{\pi_{\mathcal{A}}} \mathcal{A}_2 \times \mathcal{B}_2 \xrightarrow{\pi_{\mathcal{B}}} \mathcal{B}_2 \longrightarrow \mathcal{B}_1 \longleftarrow \mathcal{B}$$

PROOF. Our constructions above. □

We can now give a **decision algorithm** for the equivalence of finite automata $\mathcal{A}$ and $\mathcal{B}$.

(1) From $\mathcal{A}$ and $\mathcal{B}$ construct the automata $\mathcal{A}_2$ and $\mathcal{B}_2$.

(2) Check if $\mathcal{A}_2 \xleftarrow{\pi_{\mathcal{A}}} \mathcal{A}_2 \times \mathcal{B}_2 \xrightarrow{\pi_{\mathcal{B}}} \mathcal{B}_2$ are reductions. If this is the case, $L_{\mathcal{A}} = L_{\mathcal{B}}$.

We want to specify the **complexity** of this decision algorithm:

As measure of complexity of $\mathcal{A}$ we define

$$\|\mathcal{A}\| := \sum_{e \in E} |\alpha(e)|$$

"Splitting" the edges:	$O(\|\mathcal{A}\|)$
Creating the bridges:	$(card(E))^2$
Transitive closure of ϵ-graph	$O(\|\mathcal{A}\|^3)$
Construction of $\mathcal{A} \times \mathcal{B}$:	$\|\mathcal{A}\| \cdot \|\mathcal{B}\|$
Test for reduction	
1. Removal of superfluous states:	$O(\|\mathcal{A}\|^3)$
2. Surjectivity test on the rest:	$O(2^{\|\mathcal{A}\|})$

The test for surjectivity takes most of the time and dominates the complexity of this decision algorithm. Our algorithm takes the same time as the other algorithms from the literature [**?**], because we start from non-deterministic automata.

Remark: If we would have required for our automata $\mathcal{A}$ and $\mathcal{B}$ that $\alpha_{\mathcal{A}}(e) \in X$ and $\alpha_{\mathcal{B}}(e) \in X$, then the diagram from the corollary to theorem 5 could have been simplified to $\mathcal{A} \longleftarrow \mathcal{C} \longrightarrow \mathcal{B}$. For the compexity of our decision algorithm this however would be insignificant.

Exercises:

(1) Let $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$ be finite automata, $\beta_{\mathcal{A}} : \mathcal{C} \to \mathcal{A}$, $\beta_{\mathcal{B}} : \mathcal{C} \to \mathcal{B}$ reductions and $\underset{\mathcal{A}}{\equiv}, \underset{\mathcal{B}}{\equiv}$ the equivalence relations from the proof of lemma 1.

Show: $(\underset{\mathcal{A}}{\equiv} \subset \underset{\mathcal{B}}{\equiv}) \Rightarrow$ there exists a reduction $\beta : \mathcal{B} \to \mathcal{A}$ such that the following diagram commutes:

$$\begin{tikzcd} & \mathcal{C} \arrow[dl, "\beta_{\mathcal{A}}"'] \arrow[dr, "\beta_{\mathcal{B}}"] & \\ \mathcal{A} & & \mathcal{B} \arrow[ll, dashed, "\beta"] \end{tikzcd}$$

(2) Let $\mathcal{A}$ be a complete, deterministic finite automaton, $\mathcal{B}$ a finite automaton and $\beta : \mathcal{A} \to \mathcal{B}$ a homomorphism with $\beta(E_{\mathcal{A}}) = E_{\mathcal{B}}$. Show:

- β is a reduction
- The given requirements are necessary.

7. Regular sets in $X^{(*)}$, REG($X^{(*)}$)

In the first section of this chapter we looked at $REG(X^*)$, the class of regular languages over the free monoid X^*.

We replace now the free monoid by an important special case of a monoid, namely the polycyclic monoid $X^{(*)}$ which is the syntactic monoid of the Dyck-language of the alphabet $X \cup X^{-1}$.

We represent the elements of $X^{(*)}$ by their reduced words (see chapter 1.3).

Definition 7.1. *Let $|w|$ be the reduced word of $w \in (X \cup X^{-1})^*$ with respect to the polycyclic monoid $X^{(*)}$ and let $L \subset (X \cup X^{-1})^*$ be a language. Then*

$$|L| := \{|w| \mid w \in L,\ |w| \neq 0\}$$

is the set of reduced words of L.

Definition 7.2.

$$\begin{aligned} |REG|(X^{(*)}) \quad := \quad & \{L \subset (X \cup X^{-1})^* \mid \textit{there exists } L' \in REG(X^{(*)}) \\ & \textit{with } L = \{|w| \mid [w] \in L'\}\} \end{aligned}$$

Our goal is to show the inclusion

$$|REG|(X^{(*)}) \subset REG((X \cup X^{-1})^*)$$

This inclusion does not tell that the word problem of $X^{(*)}$ is rational, or in other words that $[w]$ is a rational set. But the structure of $REG(X^{(*)})$ is determined for the biggest part by $REG((X \cup X^{-1})^*)$.

Of course it is

$$|REG|(X^{(*)}) \neq REG((X \cup X^{-1})^*)$$

But we can completely characterize $|REG|(X^{(*)})$ as a subset of $REG((X \cup X^{-1})^*)$.

To do so, we prove at first a number of lemmata.

Lemma 7.1.

$$L \subset REG((X \cup X^{-1})^*) \Rightarrow |L| \in REG((X \cup X^{-1})^*)$$

Proof. Let

$$\mathcal{A} = (G, X \cup X^{-1}, S, F, \alpha)$$

be a finite automaton accepting L.

We construct in several steps an automaton $\mathcal{A}'$ accepting $|L|$.

This construction will play an exceptional role in later chapters.

Without loss of generality we may assume that for each edge of our graph $G = (V, E)$ holds: $\alpha(e) \in X \cup X^{-1} \cup \{\epsilon\}$.

Step 1: If there is a path $\pi \in \mathcal{W}(G)(v, v')$ with $|\alpha(\pi)| = \epsilon$, then we add a new edge $e : v \to v'$ to E and set $\alpha(e) := \epsilon$. ϵ-loops will be removed.

We do this for each such pair $(v, v') \in VxV$ and get a graph G_1 with corresponding labelling α.

Step 2: For edges $e \in E$ with $\alpha(e) = \epsilon$ and $e' \in E$ with $\alpha(e') \neq \epsilon$ and $Z(e) = Q(e')$ we add an edge e'' with $Q(e'') := Q(e)$, $Z(e'') := Z(e')$ and $\alpha(e'') := \alpha(e')$.

If $Z(e') = Q(\tilde{e})$ with $Z(\tilde{e}) \in F$ is a final state and $\alpha(\tilde{e}) = \epsilon$, then add edge e'' with $Q(e'') := Q(e)$, $Z(e'') := Z(\tilde{e})$ and $\alpha(e'') := \alpha(e')$.

(Translator's remark: Insert figure to clarify this)

If we do this for all such pairs of edges (e, e'), we get a graph G_2 with labelling α as described.

Step 3: We remove all edges $e \in G_2$ where $\alpha(e) = \epsilon$. The resulting graph shall be G_3 with labelling α.

Because of our construction we get:

$$L_{\mathcal{A}3} = \{v \mid v \text{ is reduced word of } w \in L_{\mathcal{A}} \text{ wrt. } x \cdot x^{-1} = 1\}$$

Step 4: We interrupt all paths with a label containing $x \cdot y^{-1},\ x \neq y$. To do so, we duplicate certain vertices and edges of G_3.

Let v be a vertex which is the target of an edge e with $\alpha(e) \in X$ and is the source of an edge e' with $\alpha(e') \in X^{-1}$.

We replace vertex v by vertices v and v^+ and add edges as follows:

All edges e_1 with $Z(e_1) = v$ and $\alpha(e_1) \in X$ will be rerouted to vertex v^+. Edges e_2 with $Z(e_2) = v$ and $\alpha(e_2) \in X^{-1}$ are left unchanged.

All edges e_3 with $Q(e_3) = v$ and $\alpha(e_3) \in X$ will be duplicated into e_3^+ and e_3^-. One of them leads from v^+ and the other from v to $Z(e_3)$.

The labelling is defined by $\alpha(e_3^+) = \alpha(e_3^-) = \alpha(e_3)$.

Edges e_4 with label $\alpha(e_4) \in X^{-1}$ and $Q(e_4) = v$ stay unchanged.

The following figure shows the construction:

FIGURE

We do this with every such vertex of G_3. Tge resulting graph shall be G'. With start and final state sets $S' := S$ and $F' := F \cup \{e^+ \mid e \in F\}$ we get a finite automaton

$$\mathcal{A}' = (G', X \cup X^{-1}, ', F', \alpha')$$

It then holds: $L_{\mathcal{A}}' = |L|$!

We observe that $L_{\mathcal{A}}'$ contains only reduced words which are different from zero. Additionally, the reversal of the construction of G' from G_3 is a functor which preserves the labelling of the edges.

Therefore $L_{\mathcal{A}}' \subset |L|$.

Let to the opposite π be a path in $\mathcal{W}(G_3)$ with label $\alpha(\pi) = |\alpha(\pi)|$, then there is no $(+,-)$-sequence in the label.

That means the path π stays unchanged in G' as long as the label of the edges is in X^{-1}.

If the label changes, the path switches to the $+$-vertices and stays "positive" until its target. If we denote this path that belongs to π by w', then $\alpha(\pi) = \alpha'(\pi')$ from which follows $|L| \subset L_{\mathcal{A}}'$.

Together we get $|L| = L_{\mathcal{A}'}$. $\square$

In this proof we even showed more which is summarized in the following corollary.

COROLLARY 7.1. *To each path* $\pi = e_1 \cdots e_k \in \mathcal{W}(G)(S,-),\ e_i \in E$ *with* $\alpha(e_k) \neq \epsilon$ *and* $\alpha(\pi) \neq \epsilon$ *there is assigned a unique* ϵ*-free path* $\pi' \in \mathcal{W}(G_3)(S,-)$ *with* $\alpha(\pi) = \alpha(\pi')$.

Additionally, we can define a functor $\nu : \mathcal{W}(G_3) \to \mathcal{W}(G)$ *such that the following holds:*

For each accepted word $u \in L$ *there exists an accepting path* $\pi = e_1 \cdots e_n \in \mathcal{W}(G_3)(S,F)$ *with* $e_i \in E_3,\ \alpha(e_i) \neq \epsilon$ *such that* $\alpha(\nu(\pi)) = u$

Proof: Exercise.

Our construction show even more: the number of paths π with $\alpha(\pi) = u$ for a given $u \in |L|$ is the same in $\mathcal{W}(G_3)$ as in $\mathcal{W}(G')$. Therefore this number is in $\mathcal{W}(G')$ less or equal to the number in $\mathcal{W}(G_2)$. A similar statement holds for $\mathcal{W}(G_1)$ and $\mathcal{W}(G)$.

This fact, which will be of importance later in this book, is summarized in the following corollary. We define

DEFINITION 7.3.

$$INDEX(|\mathcal{A}|, u) := card(\{\pi \in \mathcal{W}(G)(S,F) \mid |\alpha(\pi) = u\})$$

COROLLARY 7.2.

$$INDEX(|\mathcal{A}|, u) \geq INDEX(|\mathcal{A}'|, u),\ u \in L_{\mathcal{A}}{}'$$

Remark: Because $L_{\mathcal{A}}{}' = |L_{\mathcal{A}}{}'|$ it holds $INDEX(|\mathcal{A}'|, u) = INDEX(\mathcal{A}', u)$.

Now we can prove the announced theorem.

THEOREM 7.1.

$$|REG|(X^{(*)}) = \{|L| \mid L \in REG((X \cup X^{-1})^*)\}$$

PROOF. The inclusion "$\supset$" is clear.

"$\subset$": Every functor $\alpha : \mathcal{W}(G) \to X^{(*)}$ can be extended to a functor $\alpha' : \mathcal{W}(G) \to (X \cup X^{-1})^*$ by defining $\alpha'(e) := |u|$ for each edge $e \in E$ and $\alpha(e) = [u]$.

α' is continued to a functor and for the finite automaton

$$\mathcal{A}' = (G', X \cup X^{-1}, S', F', \alpha')$$

it holds:

$$L_{\mathcal{A}} = \{[u] \mid u \in L_{\mathcal{A}}{}'\}$$

and with lemma 1 we get:

$$L_{\mathcal{A}} = \{[u] \mid u \in |L_{\mathcal{A}}{}'|\}$$

which means

$$|L_{\mathcal{A}}| = \{|u| \mid u \in L_{\mathcal{A}}{}'\}$$

□

The next theorem gives an even more precise characterization for $|REG|(X^{(*)})$.

THEOREM 7.2. *Let $\mathcal{A}$ be a finite automaton*

$$\mathcal{A} = (G, X \cup X^{-1}, S, F, \alpha)$$

where $G = (V, E)$ and $card(E) = c$. Then there exist c pairs (L_i^+, L_i^-) with

$$L_i^+ \in REG(X^*), \qquad L_i^- \in REG((X^{-1})^*)$$

such that

$$|L_{\mathcal{A}}| = \bigcup_{i=1}^{c} L_i^- \cdot L_i^+$$

PROOF. The proof follows directly from the construction of the automaton $\mathcal{A}'$.

If V' is the set of the vertices v which have an associated vertex v^+ in G', then every path $\pi \in \mathcal{W}(G')(S', F')$ can be uniquely split into a product $\pi_1 \cdot \pi_2$ where the target $Z(\pi_1)$ is such a vertex v.

Therefore one can write

$$|L_{\mathcal{A}}| = \bigcup_{v \in V} L^-_{\mathcal{A}(S,v)} \cdot L^+_{\mathcal{A}(v,F)}$$

□

COROLLARY 7.3.

$$INDEX(\mathcal{A}', u) = \sum_{\substack{v \in V' \\ u_1 \cdot u_2 = u}} INDEX(\mathcal{A}'(S, v), u_1) \cdot INDEX(\mathcal{A}'(v, F), u_2)$$

LEMMA 7.2. *Let $L = L_{\mathcal{A}}{}^+ \subset X^*$ and $L' = L_{\mathcal{A}}{}^- \subset (X^{-1})^*$. Then there exist $L_1^+, \ldots, L_p^+ \in REG(X^*)$, $L_1, \ldots, L_q^- \in REG((X^{-1})^*)$ with*

$$|L \cdot L'| = \bigcup_{i=1}^{p} L_i^+ \ \cup \ \bigcup_{j=1}^{q} L_j^-$$

Here $p, q \leq card(F(\mathcal{A}'))$.

PROOF. We construct $\mathcal{A} = \mathcal{A}^+ \circ \mathcal{A}^-$ with $L_{\mathcal{A}} = L \cdot L'$ and thereafter $\mathcal{A}'$ with $L_{\mathcal{A}'} = |L_{\mathcal{A}}|$.

Because for every reduced word $u \in (X \cup X^{-1})^*$ it holds $u = u^- \cdot u^+$ where $u^- \in (X^{-1})^*, u^+ \in X^*$ the first part of the lemma is proven.

From our construction of $\mathcal{A}'$ it follows that the set $F(\mathcal{A}')$ of final states are divided into sets from $(X^{-1})^*$ and sets from X^*. From that follows the rest of the lemma.

□

We want to investigate in the following the set of regular sets which can be defined with fixed graph and show two simple but important properties of these sets.

Let $G = (V, E)$ be a graph and $\alpha : E \to (X \cup X^{-1})^*$. We may assume that $\alpha(e) \in X \cup X^{-1} \cup \{\epsilon\}$.

We define

$$\mathcal{R}(G) := \{L_{\mathcal{A}(\tilde{S}, \tilde{F})} \mid \mathcal{A} = (G, X \cup X^{-1}, S, F, \alpha),\ \tilde{S} \subset V,\ \tilde{F} \subset V\}$$

For $c := card(V)$ then it trivially holds $card(\mathcal{R}(G)) \leq 2^{2 \cdot c}$.

Remark: The set $\mathcal{R}(G)$ is in general not closed under union but it contains a simple base for the closure $< \mathcal{R}(G) >$ under union.

For shorter notation we set $\mathcal{R}(\tilde{S},\tilde{F}) := L_{\mathcal{A}(\tilde{S},\tilde{F})}$.

LEMMA 7.3.

$$\mathcal{R}(\tilde{S},\tilde{F}) = \bigcup_{\substack{v\in\tilde{S}\\ v'\in\tilde{F}}} \mathcal{R}(v,v')$$

PROOF. Exercise for the reader. □

Let $\mathcal{A}'$ be the automaton constructed as in lemma 1. We construct a graph $G' = (V', E')$ from the graph $G = (V, E)$. Define

$$\mathcal{B}(G') := \{L_{\mathcal{A}'(v,v')} \mid v, v' \in V'$$

LEMMA 7.4. *Let $L^+ \subset X^*, L^- \subset (X^{-1})^*$ and $L^+, L^- \in \mathcal{B}(G')$. Then $|L^+ \cdot L^-|$ is contained in the union-closure $< \mathcal{B}(G') >$ of $\mathcal{B}(G')$.*

PROOF. Let $L^+ = L_{\mathcal{A}'(v,v')}$ and $L^- = L_{\mathcal{A}'(q,q')}$. We create an automaton

$$\mathcal{A}'' := \mathcal{A}'(v,v') \circ \mathcal{A}'(q,q')$$

with $L_{\mathcal{A}''} = L^+ \cdot L^-$. □

We get $\mathcal{A}''$ by connecting an instance of $\mathcal{A}'(q,q')$ using an ϵ-edge which leads from v' to q with another instance. Now all words $u \in L_{\mathcal{A}''}$ have the form $u = u^+ \cdot u^-$. We are only interested in the reduced words.

We have the cases:

$$|u^+ \cdot u^-| = \begin{cases} u_1^+ \\ 0 \\ u_1^- \end{cases}$$