SEMINARARBEIT

Star height of rational languages

ausgeführt am

Institut für
Diskrete Mathematik und Geometrie
TU Wien

unter der Anleitung von

Associate Prof. Dipl.-Ing. Dr.techn. Stefan Hetzl

durch

Johannes Weiser BSc
Matrikelnummer: 11906087
Danhausergasse 7/10
1040 Wien

Wien, am 21. März 2023

Contents

1 Introduction

The aim of this work is to give an introduction to the star height of rational languages. In the first chapter, I present some results and definitions regarding rational languages and finite automata that are needed later on. After that I will introduce the star height itself as a measure of structural complexity of rational languages and show how the star height of a language L is linked to the complexity of automata that recognise L. In the fourth chapter we will answer the question whether there is an upper bound for the star height of languages of a fixed alphabet. Lastly, I will introduce the generalised star height as an alternative to the regular star height and present an important theorem regarding the existence and identification of languages with generalised star height strictly greater than 0.

2 Rational expressions and NFAs

The goal of this chapter is to define, what rational expressions are and to cite two theorems from [2], which together give us the result that exactly the same languages can be obtained using rational expressions and nondeterministic finite automata as defined in [2, pp. 51–53]. This allows us to later speak only about rational languages and look at them from different points of view. Furthermore, we will cite the pumping lemma for rational languages and define some special kinds of automata, which we will need later on.

First we define rational expressions:

Definition 2.1 (Rational expressions)**.** *Let Σ be an alphabet and $(0, 1, +, \cdot, *)$ functions of the respective arities $(0, 0, 2, 2, 1)$. Then*

(i) 0,1 and a are rational expressions for any $a \in \Sigma$.

(ii) If E and F are rational expressions over Σ, then E+F, $E \cdot F$ and E^* are rational expressions over Σ as well.

RatE(Σ^*) denotes the set of all rational expressions over Σ.

In order to avoid excessive use of parentheses, we agree on an operator precedence convention: $*$ has higher precedence than $\cdot$, which takes precedence over $+$.

Having defined rational expressions, we can now define the languages, which they induce:

Definition 2.2. *Let $E \in RatE(\Sigma^*)$ be a rational expression. We assign one language $L[E]$ to it using the following rules:*

(i) For atomic expressions:

$$L[0] = \emptyset, L[1] = \varepsilon \text{ and } L[a] = a \text{ for any } a \in \Sigma$$

(ii) For two rational expressions $E, F \in RatE(\Sigma^*)$:

a) $L[E + F] = \{L[E]\} \cup \{L[F]\}$

b) $L[E \cdot F] = \{L[E]\}\{L[F]\}$

c) $L[E^*] = \{L[E]\}^*$

Here ε denotes the empty word [1]. Although it is often mixed, we distinguish between rational expressions and the languages we obtain by using them. For example, in (ii)c there are two $*$ with different meanings: The first one is simply a unary operation on a rational expression, the second one is the well-known *Kleene star* applied to a language.

To show the important fact that the languages accepted by NFAs and the languages created by rational expressions are the same, we need one more definition.

[1]sometimes the empty word is referred to as 1_{A^*}

Definition 2.3.

(i) Let F be a family of languages over Σ^*. It is rationally closed if it is closed under $\cup, *$ and concatenation. That is,

$$\forall X, Y \in F: \; X \cup Y \in F, \; XY \in F, \; X^* \in F$$

(ii) Let F be a family of languages over Σ^*. We define the rational closure $Rat(F)$ as the smallest family of languages that is rationally closed and contains F .

(iii) $Rat(\Sigma^*)$ denotes the rational languages over Σ^* and is defined as the rational closure of the set of finite subsets of Σ^*.

Having defined rational languages, we can use them to establish an important connection between NFAs and rational languages.

Theorem 2.4 (Kleene)**.** *Let Σ be a finite alphabet. A language X over Σ^* is rational iff there is an NFA that recognizes it.*

Proof. Since this work focuses on other aspects of rational languages and this result is most likely already known by the reader, I will simply cite the proof from [2, pp. 87–89]. ■

Now we can make make an equally important, albeit a more obvious observation:

Theorem 2.5. *Let Σ be a finite alphabet. A language X over Σ^* is rational iff it can be produced by rational expression over Σ.*

Proof. Again we refer to [2, p. 126] for the proof. ■

With these two theorems, there is an obvious corollary.

Corollary 2.6. *Let Σ be a finite alphabet. The class of languages recognized by NFAs is identical to the class of languages obtained using rational expressions.*

Before we carry on with NFAs, there is one lemma, we will need later in Chapter 4.

Lemma 2.7 (Pumping Lemma)**.** *Let L be a rational language over Σ^*. Then there existes an integer $N \in \mathbb{N}$ such that for every word $f \in L$ and every factorisation $f = uv_1v_2 \cdots v_Nw$, where all the v_i are not empty, there are indices j and k such that the following holds:*

$$uv_1 \cdots v_j(v_{j+1} \cdots v_k)^* v_{k+1} \cdots v_N w \in L$$

Proof. For the proof we refer to [2, pp. 71–72]. ■

We now make some necessary definitions regarding automata that we will need later on.

Definition 2.8. *Let $A = (Q, \Sigma, E, I, T)$ be an NFA. We call A normal if the following criteria are met:*

(i) There is exactly one initial state i, which is not the destination of any transition of A.

(ii) There is exactly one final state t, which is not the source of any transition of A.

(iii) $i \neq t$

Now we can define trim automata:

Definition 2.9. *Let $A = (Q, \Sigma, E, I, T)$ be an NFA. A state $q \in Q$ is called accessible if there is a path from an initial state i that leads to q. A state $q \in Q$ is called co-accessible if there is a path from q to a final state t. If all the states of A are accessible, we call A accessible. If all the states of A are co-accessible, we call A co-accessible. If A is accessible and co-accessible, we call A trim.*

It is easy to understand that:

Lemma 2.10. *Every NFA A is equivalent to a trim automaton A_m, which is also finite.*

For later purposes we now need to make another definition and have a look at how we can determine the language that an automaton recognizes.

Definition 2.11. *$A = (Q, \Sigma, E, I, T)$ is called a generalised automaton if its transitions do not have to be labelled by letters from Σ but can also be labelled by any subsets of Σ^*.*

The first thing we observe is the following: If there are two edges going from a to b with $a, b \in Q$ with the labels E_1 and E_2 respectively, we can simply combine those edges into one edge going from a to b with the label $E_1 \cup E_2$. Therefore we can assume that for any pair $(a, b) \in Q^2$ there exists not more than one edge going from a to b.

Next we want to determine the language that is recognized by an NFA using generalised automata. Let $A = (Q, \Sigma, E, I, T)$ be an NFA. We add the states i, t to Q and the triples (i, ε, q_i) and (q_t, ε, t) to E for all $q_i \in I$ and $q_t \in T$. We now have the normal NFA $A' = (Q', \Sigma, E', \{i\}, \{t\})$. Now we pick one arbitrary state p in Q. Let $a_1, \ldots, a_n$ be all the states different from p such that $(a_i, e_i, p) \in E$, $b_1, \ldots, b_m$ all the states different from p such that $(p, l_i, b_i) \in E$ and L the expression such that $(p, L, p) \in E$ if there is such an L, otherwise $L = \emptyset$. Now we eliminate p from Q' and all the transitions that contain p from E'. In return we add the transitions $(a_i, e_i L^* l_k, b_k)$ for all i, k to E'. At last we can build the union over the labels of the edges if there now are several edges between two states. We are left with a general automaton that is obviously equivalent to A and A', but has not as many states. By repeating the elimination of steps we receive an automaton that only has two states i and t and only one transition (i, L_f, t). L_f is the language this automaton recognizes. This procedure is known as the *state elimination algorithm* or the *BMC algorithm* [2, pp. 97–99].

The language we receive from the algorithm in the paragraph above is obtained using only a finite number of unions, stars and concatenation. It is therefore rational. Thus the algorithm above constitutes a proof of one inclusion of Corollary 2.6 (i.e. recognizable is rational).

It is important to state that although the automaton recognizes exactly one language, this language can be described using several rational expressions. If we eliminate the states in a different order, we might get different expressions with the algorithm above. We will come back to that later in Section 3.3.

3 Complexity of rational languages

Having established the basic connection between NFAs and rational expressions, we can now consider rational languages from multiple perspectives. The goal of this chapter is to introduce two different measures of structural complexity of rational languages and show how they are connected. The main result is Eggan's theorem 3.12 stating that the complexity of rational expressions is in some sense transferred to the NFAs that recognize the same language.

3.1 Star height

In this section we introduce the star height for rational expressions and subsequently for rational languages as a measure of structural complexity.

Definition 3.1. *Let $E \in RatE(\Sigma^*)$ be an rational expression over Σ. Then the star height h[E] is defined by induction:*

(i) If $E = 0$, $E = 1$ or $E = a$ for any $a \in \Sigma$, then h[E]=0.

(ii) If $E = A + B$ or $E = A \cdot B$ for $A, B \in RatE(\Sigma^*)$, then $h[E] = \max\{h[A], h[B]\}$.

(iii) If $E = A^*$, then $h[E] = 1 + h[A]$.

An informal characterisation would be to say that the star height of an expression is the highest count of nested stars.

In order to motivate the next definition, we consider an example:

Example 3.2. *Let $E = \{a, b\}^*(b\{a, b\}^*)^* \in RatE(\{a, b\}^*)$. We see that $h[E] = 2$. Obviously, the language $L := L[E]$ contains all the words from $\{a, b\}^*$. That means, we can write L as $L[F]$ with $F = \{a, b\}^*$ and $h[F] = 1$. Moreover, it is evident that there is no rational expression G with $h[G] < 1$ and $L[G] = L$ since there are infinitely many words in L and this requires at least one $*$ being used in any rational expression that induces the same language. Therefore, 1 constitutes the minimum of all the star heights of rational expressions that induce L, that is*

$$1 = \min\{h[X] \mid X \in RatE(\{a, b\}^*) \wedge L[X] = L\}$$

Because of this ambiguity of the star heights of expressions that produce the same language, we make the following definition:

Definition 3.3. *Let $L \in Rat(\Sigma^*)$ be a rational language over Σ^*. The star height of L, written as $h[L]$ is defined as the minimum of the star heights of the rational expressions that induce L:*

$$h[L] = \min\{h[X] \mid X \in RatE(\Sigma^*) \wedge L[X] = L\}$$

We see that the language from our Example 3.2 has start height 1.

In Chapter 4 we will prove an important property of the star height. For now and in preparation for Eggan's theorem, the definition is enough.

3.2 Loop complexity

In this section we want to define the loop complexity of a given automaton. Before we do that, we have to circle back and have a look at automata as graphs. We start with defining the strongly connected components of a graph.

Lemma 3.4. *Let G=(V,E) be a directed graph. $\approx$ is a relation[1] on V and is defined by $x \approx y$ if one of the following two conditions is met:*

(i) $x = y$

(ii) $\exists v_1, \dots, v_n, w_1, \dots w_m : \ (x, v_1), (v_1, v_2), \dots, (v_n, y), (y, w_1), (w_1, w_2), \dots, (w_m, x) \in E.$

Then $\approx$ is an equivalence relation.

Proof. Obviously, $\approx$ is reflexive and symmetric. Furthermore, if there is a path a from x to y, a path b from y to x, a path c from y to z and a path d from z to y, then ac is a path from x to z and db is a path from z to x. This means, $\approx$ is also transitive and therefore an equivalence relation. ■

This result leads to the following definition:

Definition 3.5. *The equivalence classes induced by the relation from Lemma 3.4 are called strongly connected components of the graph G.*

More informally, one could say that the strongly connected components of a graph G are the parts of it where every node can be reached from every other node of the same part. Now we can define the *balls* of a graph G:

Definition 3.6. *Let $G = (V, E)$ be a directed graph and $C \in G/_\approx$ be a strongly connected component of G. We call C a ball of G if it contains at least one edge. That is, C is a ball if one the following two conditions is met:*

(i) C contains two or more nodes.

(ii) C contains only one node v and $(v, v) \in E$.

Having defined the terms strongly connected component and ball, we can finally define the loop complexity[2] of a graph G.

Definition 3.7. *Let G=(V,E) be a directed graph. For any $s \in V$ we write $G \setminus \{s\}$ for the directed graph $G' = (V \setminus \{s\}, \{(a, b) \in E \mid a \neq s \wedge b \neq s\}$. We define the loop complexity $lc(G)$ recursively:*

[1]Informally spoken: $x \approx y$ if there is a cycle that contains both x and y

[2]sometimes the loop complexity is referred to as the cycle rank

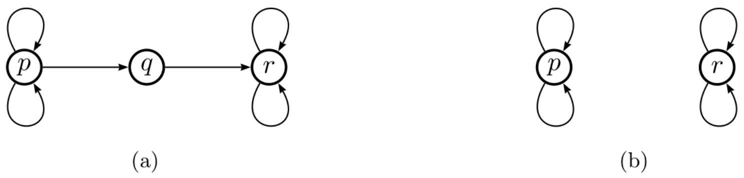

Figure 3.1: A graph and its balls [2, p. 160]

(i) $lc(G) = 0$ *if* G *contains no balls*

(ii) $lc(G) = \max\{lc(P) \mid P$ *is a ball of* $G\}$ *if* G *is not strongly connected*

(iii) $lc(G) = 1 + \min\{lc(G \setminus \{s\}) \mid s \in V\}$ *if* G *is strongly connected*

We have so far spoken mostly about graphs in this section. Since an automaton A is a directed graph, we can define the strongly connected components, the balls and the loop complexity accordingly.

3.3 Eggan's theorem

The goal of this section is to finally formulate and prove Eggan's theorem 3.12. This whole section closely follows [2, pp. 162–166]. In order to develop Eggan's theorem, we will need some definitions and lemmas. First we define the index of a rational expression E:

Definition 3.8. *The index* $i(E)$ *of a rational expression* E *is given by* $i(E) = h[E]$. *If* $e = (q_1, E, q_2)$ *is a transition of an automaton we define the index of* e *as* $i(e) = i(E)$.

Having defined the index of an expression, we can carry on and define the index of an automaton A relative to a total order ω on the states of A. We define the trace of omega over any subset R of Q as $\omega_{|R\times R}$. In the following, we write $\overline{\omega}$ for the greatest element of Q according to ω and ω for the trace of ω over $R \subseteq Q$ (and also $\overline{\omega}$ for the greatest element of R). A transition $e = (q_1, E, q_2)$ of an automaton is called adjacent to p if $q_1 = p$ or $q_2 = p$.

Definition 3.9. *Let* $A = (Q, \Sigma, E, I, T)$ *be a generalised automaton and* ω *a total order on* Q. *We define the index* $i(A, \omega)$ *inductively:*

(i) $i(A, \omega) = 0$ *if* A *is empty*

(ii) $i(A, \omega) = 1 + \max(\{i(e) \mid e$ *is adjacent to* $\overline{\omega}\} \cup \{i(A \setminus \overline{\omega}, \omega)\})$ *if* A *is a ball*

(iii) $i(A, \omega) = \max(\{i(e) \mid e$ *does not belong to a ball in* $A\} \cup \{i(P, \omega) \mid P$ *is a ball in* $A\})$ *if* A *is not a ball*

If we compare the definition of the loop complexity and the index of an automaton, we see the following for 'normal' (i.e. not generalised) automata:

Lemma 3.10. *Let* A *be an NFA, then*

$$lc(A) = \min\{i(A, \omega) \mid \omega \text{ is an order on } Q\}.$$

In the following we write $E_{BMC}(A,\omega)$ for the rational expression we receive from the state elimination algorithm applied to A in the order ω (from smallest state to greatest). We can now formulate the most central lemma in this chapter:

Lemma 3.11. *Let ω be a total order on the set of states Q of the automaton A. Then $i(A,\omega) = h[E_{BMC}(A,\omega)]$.*

Proof. We prove this lemma via induction on the number of states of Q. The first step of the BMC algorithm is to transform A to an equivalent automaton B by adding two states i and t. We extend ω by making i and t greater than the states in Q ($t > i$). Therefore $i(A,\omega) = i(B,\omega)$. We now take care of the base case where A has one state and B therefore three. In any case B itself is not a ball and not empty. We distinguish two cases (see Figure 3.2):

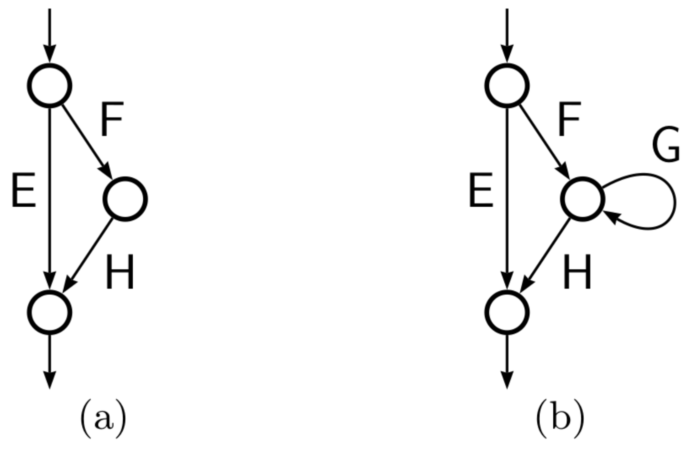

Figure 3.2: The base case [2, p. 165]

(a) B contains no balls: $i(B,\omega) = \max\{h[E], h[F], h[H]\} = h[E+F\cdot H] = h[E_{BMC}(B,\omega)]$

(b) B contains a ball: $i(B,\omega) = \max\{h[E], h[F], h[H], 1+h[G]\} = h[E + F\cdot G^* \cdot H] = h[E_{BMC}(B,\omega)]$

In both cases we get the desired result. Now let B be a normalised automaton with n+2 states and q the smallest state according to ω. Let P be the smallest ball strictly containing q and $P = B$ if there is no such ball. We define B' as the automaton we receive from the state elimination algorithm after eliminating q and P' as the 'image' of P in B'. We distinct two cases:

(i) There is no edge (q, G, q) in E. First we have to realize that all the edges in P' are either the same as in P (if they're not adjacent to q) or they are a product $F \cdot H$. In this case, we know $h[F \cdot H] = \max(h[F], h[H])$. This is important because now we can look at the definition of the index and switch P and P'. It is crucial to state that we only have to consider the case (*iii*) from Definition 3.9 because if P' is a ball, then (*ii*) and (*iii*) coincide due to the monotony of the index:

$$\begin{aligned}
i(P',\omega) &= \max(\{i(e) \mid e \text{ does not belong to a ball in } P'\} \cup \{i(Q,\omega) \mid Q \text{ ball in } P'\}) \\
&= \max(\max\{i(e) \mid e \text{ does not belong to a ball in } P'\}, \max\{i(Q,\omega) \mid Q \text{ ball in } P'\} \\
&= \max(\max\{i(e) \mid e \text{ does not belong to a ball in } P\}, \max\{i(Q,\omega) \mid Q \text{ ball in } P\} \\
&= i(P,\omega)
\end{aligned}$$

(ii) There is an edge (q, G, q) in E. Similarly to the case above, the edges in P' are either the same as in P or have the form $F \cdot G^* \cdot H$. In this case, we have to remember $h[F \cdot G^* \cdot H] = \max(h[F], h[G^*], h[H])$ and $i(\{q\},\omega) = 1 + h[G] = h[G^*]$. We then get:

$$\begin{aligned}
i(P',\omega) &= \max(\{i(e) \mid e \text{ does not belong to a ball in } P'\} \cup \{i(Q,\omega) \mid Q \text{ ball in } P'\}) \\
&= \max(\max\{i(e) \mid e \text{ does not belong to a ball in } P'\}, \max\{i(Q,\omega) \mid Q \text{ ball in } P'\} \\
&= \max(\max\{i(e) \mid e \text{ does not belong to a ball in } P\}, 1 + h[G], \\
&\quad \max\{i(Q,\omega) \mid Q \text{ ball in } P'\} \\
&= \max(\max\{i(e) \mid e \text{ does not belong to a ball in } P\}, i(\{q\},\omega), \\
&\quad \max\{i(Q,\omega) \mid Q \text{ ball in } P \text{ and Q is different from } \{q\}\} \\
&= i(P,\omega)
\end{aligned}$$

We can now again distinguish between two cases:

(i) $P = B$. Then $P' = B'$ and since $i(P,\omega) = i(P',\omega)$, $i(B,\omega) = i(B',\omega)$. This proves the induction and the whole lemma.

(ii) $P \neq B$. Having show the property $i(P,\omega) = i(P',\omega)$, we can use it to show that it holds for every ball containing q by induction. For the other balls it holds in any case since they are not affected by the algorithm. Thus for all balls Q in B we get that $i(Q,\omega) = i(Q',\omega)$ and therefore $i(B,\omega) = i(B',\omega)$.

■

We can now formulate the central theorem of this section:

Theorem 3.12 (Eggan's Theorem)**.** *Let L be a rational language over Σ.*

(i) For every trim NFA A that recognizes the language L there is a rational expression E such that $L[E] = L$ and $lc(A) = h[E]$.

(ii) For every rational expression E with $L[E] = L$ there exists an NFA A such such that A recognizes L and $h[E] = lc(A)$.

Proof. We can prove (*i*) by referring to Lemma 3.10 and Lemma 3.11. In order to prove (*ii*) we simply perform an induction over the depth of the expression E. The base case is clear: The empty word and every letter of the alphabet Σ is recognized by an acyclic automaton with loop complexity 0. If we consider the union of to expressions E and F and of the respective automata A_E and A_F, we clearly get the new loop complexity $lc(A_E \cup A_F)$ by taking the maximum of $lc(A_E)$ and $lc(A_F)$. The same happens if we concatenate two expressions. The last case is the star operator. Let E be a rational expression and A the corresponding automaton such that $h[E] = lc(A)$. We now consider E^* and take two copies of A: A' and A''. We concatenate them in the following way: We insert a state r that works as a bridge between the final states of A' and the initial states of A''. Moreover, we insert a state u that works as a bridge between the final states of A'' and the initial states of A'. We call the new automaton B and take u as the initial state and r and u as the final ones. Clearly B recognizes E^*. Obviously B is a ball. That means

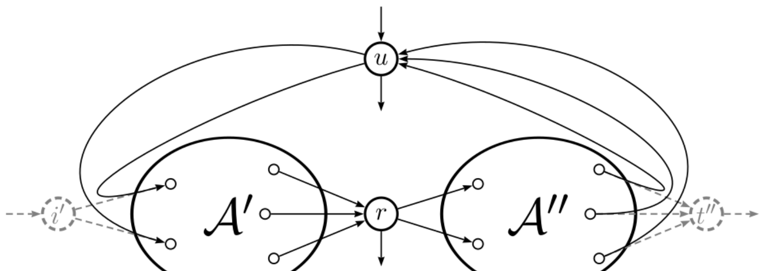

Figure 3.3: A', A'' and B [2, p. 162]

$lc(B) = 1 + \min\{lc(B \setminus s) \mid s \in B\}$. If we take $s = r$ the balls of $B \setminus s$ are the ones of A' and A'' and therefore A. That means $lc(B) \leq 1 + lc(A)$. On the other hand, we take any s in B, it still contains a copy of A and therefore $lc(B) \geq 1 + lc(A)$. Thus $lc(B) = 1 + lc(A) = 1 + h[E] = h[E^*]$, which concludes the induction. ∎

We receive an obvious corollary:

Corollary 3.13. *For any rational language L, it holds that*

$$h[L] = \min\{lc(A) \mid A \text{ is an NFA } \wedge L[A] = L\}.$$

Proof. Per definition $h[L] = \min\{h[E] \mid E \in RatE(\Sigma^*) \wedge L[E] = L\}$. Theorem 3.12 tells us that $\{h[E] \mid E \in RatE(\Sigma^*) \wedge L[E] = L\} = \{lc(A) \mid A \text{ is an NFA } \wedge L[A] = L\}$. ∎

4 The star height hierarchy

In one of the previous sections we have seen a rational language with the star height 1. Furthermore we can easily construct a language with the star height 0 if we consider the rational expression $a \in RatE(\{a,b\}^*)$. The question that naturally follows is if there is an upper bound for the star height of languages or if there are languages with an arbitrary star height. In this chapter we will answer this question positively by constructing languages that have star height $q \in \mathbb{N}$. Similarly to the last section, this chapter closely follows [2, pp. 167–169].

In the following we will only consider $\Sigma = \{a,b\}$ and for the rest of the chapter $q \in \mathbb{N}$ is fixed. We will now construct a language and show that it has star height q.

Definition 4.1. *Let W_q be the language over $\{a,b\}$ that only contains the words that fullfil*

$$|u|_a \equiv |u|_b \mod 2^q,$$

where $|y|_x$ is the number of occurrences of x in the word y.

We can formulate the following lemma

Lemma 4.2.

$$h[W_q] \leq q$$

Proof. We simply need to find an expression of star height q that produces W_q. To do that we look at an NFA that recognizes W_q. Having in mind that $|u|_a \equiv |u|_b \mod 2^q$, we can look at a ring automaton A_q with 2^q states $p_1, \dots, p_{2^q}$ and the transitions (p_1, b, p_{2^q}), (p_{2^k}, a, p_1), (p_i, a, p_k) and (p_k, b, p_i) for all $i - k = 1$ (see Figure 4.1 for the case $q = 3$). We can now calculate the loop complexity of this automaton. Since A_q is strongly connected, we have $lc(A_q) = 1 + \min\{lc(A_q \setminus \{s\}) \mid s \in A_q\}$. What ever s we take, we get one strongly connected 'chain' of length $2^q - 1$ as a result. Since A_q is symmetrical, we can take any s and without loss of generality, we choose s to be our initial state. We call this chain A'. Let's now look at the 'chains'. Let C be any chain in the form described above, but of any length $n = 2^p - 1$. With induction, we want to show that $lc(C) = p - 1$. If the length is 1, then C contains no balls and the loop complexity is zero. Let the length be $2^p - 1$. C is strongly connected, therefore $lc(C) = 1 + \min\{lc(C \setminus s) \mid s \in C\}$. Since the loop complexity is obviously monotone in the sense that $A \subseteq B \Rightarrow lc(A) \leq lc(B)$, we see that we have to eliminate the middle node and get the minimum by splitting the chain in half. By doing that, we get two chains of the length $2^{p-1} - 1$. Therefore, $lc(C) = 1 + p - 2 = p - 1$, which concludes the induction. In our case we have a chain A' of length $2^q - 1$. Thus $lc(A_q) = 1 + lc(A') = 1 + q - 1 = q$. Eggan's theorem now tells us that there is a rational expression E such that $L[E] = W_q$ and $h[E] = lc(A_q) = q$. ∎

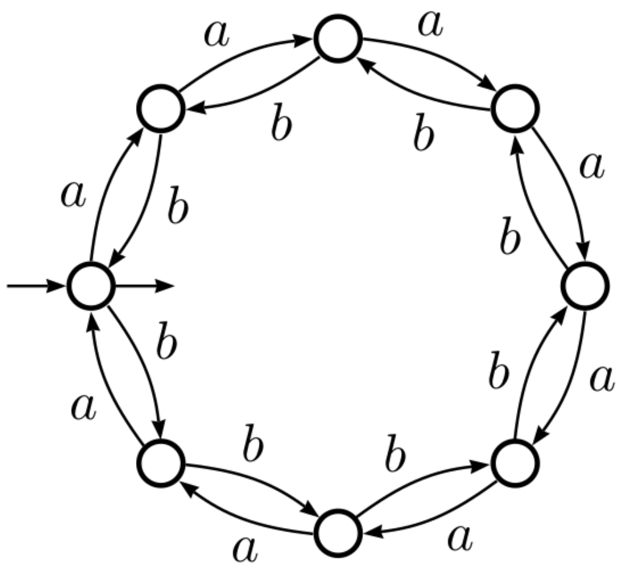

Figure 4.1: A ring automaton for $q = 3$ [2, p. 168]

We now still have to show that W_q has star height of at exactly q. To start, we define certain words, we will look at more closely in the following proof.

Definition 4.3. *Let $w_{n,k}$ be words over Σ defined inductively by*

(i) $w_{0,n} = ab$

(ii) $w_{k,n} = a^{2^k}(w_{k-1,n})^n b^{2^k}(w_{k-1,n})^n$

These words have a special property that we will need later:

Lemma 4.4. *Any left factor u and right factor v of the word $(w_{k,n})^n$ satisfy the equations:*

$$0 \le |u|_a - |u|_b \le 2^{k+1} - 1,\ 0 \le |v|_b - |v|_a \le 2^{k+1} - 1.$$

Proof. We prove this with induction on k. It is clear that the property holds for $(w_{0,n})^n = (ab)^n$. Now consider $(w_{k,n})^n = (a^{2^k}(w_{k-1,n})^n b^{2^k}(w_{k-1,n})^n)^n$. First, we notice that every $w_{k,n}$ contains an equal number of $a's$ and $b's$. That means it is equivalent if the property holds for $(w_{k,n})^n$ or for $w_{k,n}$. We can now distinguish a few cases:

(i) u is a left factor of a^{2^k}. Obviously $0 \le |u|_a - |u|_b \le 2^k - 0 \le 2^{k+1} - 1$.

(ii) u is a left factor of $a^{2^k}(w_{k-1,n})^n$ and not of a^{2^k}. Then $|u|_a = 2^k + l$ and $0 \le l - |u|_b \le 2^k - 1$. Thus $|u|_b \le l$ and $|u|_b \ge l - 2^k + 1$. Therefore $0 \le |u|_a - |u|_b$ and $|u|_a - |u|_b \le 2^k + l - l + 2^k - 1 = 2^{k+1} - 1$

(iii) u is a left factor of $a^{2^k}(w_{k-1,n})^n b^{2^k}$ and not of $a^{2^k}(w_{k-1,n})^n$. Then $|u|_a = 2^k + l$ and $|u|_b = l + m$ with $0 < m \leq 2^k$. We get that $0 \leq |u|_a - |u_b| \leq 2^{k+1} - 1$.

(iv) u is a left factor of $a^{2^k}(w_{k-1,n})^n b^{2^k}(w_{k-1,n})^n$ and not of $a^{2^k}(w_{k-1,n})^n b^{2^k}$. Then $|u|_a = 2^k + l + m$ and $|u|_b = l + 2^k + h$ with $0 \leq m - h \leq 2^k - 1$. Thus $0 \leq |u|_a - |u|_b = m - h \leq 2^{k+1} - 1$.

The proof for v is practically the same. Alternatively one could look at the words $x_{k,n} = b^{2^k}(x_{k-1,n})^n a^{2^k}(x_{k-1,n})^n$ with $x_{0,n} = ba$ and use the first case. ■

We now define a property for languages over $\{a, b\}$:

Definition 4.5. *Let L be a language over $\{a, b\}$. We say L satisfies (P_k) if there is an infinite number of values n such that $(w_{k,n})^n$ is a factor of at least one word of L. We define X_k as the set of languages L such that*

(i) $L \subseteq W_q$

(ii) L satisfies (P_k)

(iii) L has minimum star height of all languages that satisfy (i) and (ii).

We see that any language in X_0 has to be infinite and has therefore star height strictly greater than 0. On the other hand $(w_{k-1,n})^n$ is a factor of $(w_{k,n})$. Therefore, any language that satisfies P_k also satisfies P_l for $l \leq k$. Moreover, $(w_{q-1,i})^i \in W_q$ for all $i \in \mathbb{N}$. Therefore, W_q fulfils P_{q-1}. That's why, we can write the following inequality:

$$0 < h_0 \leq h_1 \leq \cdots \leq h_{q-1} \leq q, \tag{4.1}$$

where h_k is the common star height of the family X_k. We can now formulate the important lemma for this chapter:

Lemma 4.6. *Let h_k be the common star height of the languages of X_k for any k. Then it holds that $h_{k-1} < h_k$*

Proof. Let L be in X_k. Since L is rational it can be written as a finite union of languages F_j of the form

$$F_j = f_0 H_1^* f_1 \cdots H_m^* f_m. \tag{4.2}$$

There is only a finite number of $F_j's$, so there has to be at least one that satisfies (P_k), therefore we can assume that L itself is of the form (4.2). L has star height h_k, therefore there has to be one of the H_i that has star height $h_k - 1$ and none of them have a star height higher than that. We see that $f_0 f_1 \cdots f_m$ has to be in W_q and therefore every H_i must be a subset of W_q. Again, L is of minimal star height h_k and therefore none of the H_i can fulfil (P_k). We now have to show that one of them satisfies (P_{k-1}) to prove the lemma.

Since there are infinitely many $n's$ such that $(w_{k,n})^n$ is a factor of at least one word in L, these $n's$ get arbitrarily large. We can now apply the pumping lemma and conclude that there are infinitely many $n's$ and for each of these $n's$ there are infinitely many $l's$ such that $(w_{k,n})^l$ is a factor of at least one word of L. Since m from (4.2) is finite, there has to be at least one H_i^* that satisfies (P_k). Therefore, there are infinitely many $n's$ such that $(w_{k,n})^n$ is a factor of a word $r_n \in H_i^*$. Let us write r_n as the product $r_n = g_0 g_1 \cdots g_l$ with $g_j \in H_i$. We can now distinguish different cases:

(i) $w_{k,n}$ is a factor of one of the g_j. Then H_i fulfils (P_{k-1}) because $(w_{k-1,n})^n$ is a factor of $w_{k,n}$.

(ii) Otherwise: Let's look at

$$(w_{k,n})^2 = a^{2^k}(w_{k-1,n})^n b^{2^k}(w_{k-1,n})^n a^{2^k}(w_{k-1,n})^n b^{2^k}(w_{k-1,n})^n.$$

Again, we distinguish two cases:

a) b^{2^k} is a factor of a g_j and therefore $g_j = vb^{2^k}u$. Two cases:

i. Either v or u covers $(w_{k-1,n})^n$ completely. Then $(w_{k-1,n})^n$ is a factor of g_j and H_i fulfils (P_{k-1}).

ii. v is a right factor of $(w_{k-1,n})^n$ and u is a left factor of $(w_{k-1,n})^n$. We set

$$x = |g_j|_b - |g_j|_a,\ y = |u|_a - |u|_b,\ z = |v|_b - |v|_a.$$

Then $x = 2^k - (y - z)$. By Lemma 4.4 we get that $0 \leq x, y \leq 2^k - 1$ and therefore $1 - 2^k \leq x - y \leq 2^k - 1$. Thus $0 < x < 2^{k+1} \leq 2^q$. This means that $|g_j|_b - |g_j|_a \not\equiv 0 \mod 2^q$. Therefore, g_j cannot be in W_q and therefore not in H_i, which is a contradiction.

b) A left factor of b^{2^k} is a right factor of a g_j. That is $g_j = vb^r,\ r > 0$. Two cases:

(i) v covers $(w_{k-1,n})^n$ completely. Then H_i fulfils (P_{k-1}).

(ii) v is a right factor of $(w_{k-1,n})^n$. Then we define similarly as above

$$x = |g_j|_b - |g_j|_a,\ z = |v|_b - |v|_a.$$

Then $x = 2^r + z$ with $0 \leq z \leq 2^k - 1$. Since $r \leq k$, we get $0 < x < 2^{k+1} \leq 2^q$ and the same contradiction as above.

In any case, we obtain that H_i fulfils (P_{k-1}) and therefore $h_{k-1} < h_k$. ■

This leads to the conclusion that (4.1) can actually be written as

$$0 < h_0 < h_1 < \cdots < h_{q-1} \leq q \tag{4.3}$$

Since this is a strictly monotone sequence with q elements which are all strictly greater than zero, we can conclude that $h_{q-1} = q \leq W_q$. This leads to the following corollary:

Corollary 4.7. *For any $q \in \mathbb{N}$, there is a language L such that $h[L] = q$.*

5 Generalised star height

Instead of defining rational expressions like we did in Chapter 2, one can also define *generalised rational expressions* by including the complement. That leads to the so-called *generalised star height.* This chapter aims to elaborate on this alternative approach and present the important Theorem 5.11 as well as an important question that hasn't been answered yet.

Let us start with the definition of generalised rational expressions:

Definition 5.1. *Let* Σ *be an alphabet and* $(0, 1, +, \cdot, *, \bar{\cdot})$ *functions of the respective arities* $(0, 0, 2, 2, 1, 1)$*. Then*

(i) 0,1 and a are generalised rational expressions for any $a \in \Sigma$.

(ii) If E and F are generalised rational expressions over Σ*, then E+F,* $E \cdot F$*,* E^* *and* $\overline{E}$ *are generalised rational expressions over* Σ *as well.*

$GRatE(\Sigma^*)$ *denotes the set of all generalised rational expressions.*

The precedence of the operators stays the same with the addition that $* > \bar{\cdot} > \cdot$. Similarly to rational expressions, we can assign languages to generalized rational expressions:

Definition 5.2. *Let* $E \in GRatE(\Sigma^*)$ *be a generalised rational expression. We assign one language* $L[E]$ *to it using the following rules:*

(i) For atomic expressions:

$$L[0] = \emptyset, L[1] = \varepsilon \text{ and } L[a] = a \text{ for any } a \in \Sigma$$

(ii) For two generalised rational expressions $E, F \in GRatE(\Sigma^*)$*:*

a) $L[E + F] = \{L[E]\} \cup \{L[F]\}$

b) $L[E \cdot F] = \{L[E]\}\{L[F]\}$

c) $L[E^*] = \{L[E]\}^*$

d) $L[\overline{E}] = \Sigma^* \setminus L[E]$

The following states an important fact about rational languages, which is widely known today:

Lemma 5.3. *Rational languages are closed under complement.*

Proof. Let L be a rational language. Since the same languages can be obtained by using DFAs and NFAs, there exists a DFA $A = (Q, \Sigma, E, q_0, F)$ that recognizes L. We now consider the DFA $B = (Q, \Sigma, E, q_0, Q \setminus F)$. $E \subseteq Q \times \Sigma \times Q$ is a function mapping values from $Q \times \Sigma$ to Q. Therefore, all the words of Σ^* can be 'processed' by the automaton, but only the words of $\overline{L}$ stop at a final state $p \in Q \setminus F$. ■

This Lemma tells us why it is interesting to look at generalised rational expressions:

Corollary 5.4. *The languages obtained by using generalised rational expressions are the same as the ones we receive from rational expressions.*

Now we can assign the generalised star height to generalised rational expressions and the languages they induce.

Definition 5.5. *Let $E \in GRatE(\Sigma^*)$ be a generalised rational expression. The generalised star height $gsh[E]$ is defined by induction:*

(i) If $E = 0$, $E = 1$ or $E = a$ for any $a \in \Sigma$, then h[E]=0.

(ii) If $E = A+B$ or $E = A{\cdot}B$ for $A, B \in GRatE(\Sigma^*)$, then $gsh[E] = \max\{gsh[A], gsh[B]\}$.

(iii) If $E = \overline{A}$ for $A \in GRatE(\Sigma^*)$, then $gsh[E] = gsh[A]$

(iv) If $E = A^*$, then $gsh[E] = 1 + gsh[A]$.

For a rational language L, the generalised star height $gsh[L]$ is defined as

$$gsh[L] = \min\{gsh[E] \mid E \in GRatE(\Sigma^*) \wedge L[E] = L\}$$

5.1 The syntanctic monoid and the theorem of Schützenberger

Every language induces a certain algebraic structure, the syntactic monoid. The goal of this section is to define the syntactic monoid and present the theorem of Schützenberger, which states that certain properties of the syntactic monoid relate to the generalised star height of the language.

Definition 5.6. *Let L be a rational language over Σ. We define the relation $\approx_L$ over Σ^* by*

$$x \approx_L y :\Leftrightarrow (\forall u, v \in \Sigma^* uxv \in L \Leftrightarrow uyv \in L).$$

Lemma 5.7. *$\approx_L$ is an equivalence relation. Furthermore, for $M := \Sigma^*/_{\approx_L}$ it holds that*

(i) $[x][y] := [xy]$ is a well defined binary operation on M

(ii) $[\varepsilon][x] = [x][\varepsilon] = [x]$ for all $[x] \in \Sigma^*$.

Proof. The relation $\approx_L$ is obviously reflexive, symmetric and transitive. For the other properties:

(i) Let $[a] = [b]$ and $[x] = [y]$. Now let's take $u, v \in \Sigma^*$ and assume $uaxv \in L$. Since $a \approx_L b$, this is equivalent to $ubxv$ and because $x \approx_L y$, this is also equivalent to $ubyv \in L$. We get that $[a][x] = [b][y]$.

(ii) Obviously $u\varepsilon xv \in L \Leftrightarrow ux\varepsilon v \in L \Leftrightarrow uxv \in L$.

■

Corollary 5.8. $(\Sigma^*/_{\approx_L}, \cdot, [\varepsilon])$ *is a monoid.*

Definition 5.9. *We call the monoid from Corollary 5.8 the syntactic monoid from L.*

Definition 5.10. *A monoid $(M, \cdot, 1)$ is called aperiodic if for all $x \in M$ there exists an $n \in \mathbb{N}$ such that $x^n = x^{n+1}$.*

In chapter Chapter 4 we have seen that there are languages of an arbitrarily high star height. This begs the question if there are such languages for the generalised star height. To this day, this question could not be answered in full. However, Schützenberger proved that there are languages that possess the generalised star height 1. More precisely, he proved the next theorem. I will only cite this theorem and refer to [1, pp. 87–93] for the proof since it would be out of scope for this work.

Theorem 5.11. *A recognizable language L has generalised star height 0 if and only if its syntactic monoid is aperiodic.*

Example 5.12. *Let us look at the rational language $L = L[(a^2)^*]$. Obviously, $gsh[L] \leq 1$. If we consider $M := a^*/_{\approx_L}$, we see that there are exactly two elements $[a], [aa]$ (the equivalence class of all the words with an even number of a's and the one of those with an odd number). We see that*

$$[a][a] = [aa], \ [aa][aa] = [aa], \ [aa][a] = [a][aa] = [a].$$

Therefore, we conclude that $(M, \cdot, [aa]) \cong (\mathbb{Z}/_{2\mathbb{Z}}, +, 0)$. This means that $(M, \cdot, [aa])$ is not aperiodic (consider 1). Theorem 5.11 now tells us that $gsh[L] > 0$ and we see that $gsh[L] = 1$.

Bibliography

[1] J. E. Pin. *Varieties of Formal Languages*. London: North Oxford Academic Publishers, 1986.

[2] Jacques Sakarovitch. *Elements of Automata Theory*. Cambridge: Cambridge University Press, 2009.