Universitat Rovira i Virgili
Facultat de Lletres
Departament de Filologies Romàniques

Languages Generated by Iterated Idempotencies

PhD Dissertation

Presented by
Peter LEUPOLD

Supervised by
Juhani KARHUMÄKI and Victor MITRANA

Tarragona, 2006

Supervisors

Professor Juhani Karhumäki

Department of Mathematics
University of Turku
20014 Turku
Finland

and

Professor Victor Mitrana

Grup de Recerca de Linguïstica Matemàtica
Universitat Rovira i Virgili
Pça. Imperial Tàrraco 1
43005 Tarragona
Catalunya
Spain

and

Faculty of Mathematics and Computer Science
Bucharest University
Str. Academiei 14
70109 Bucureşti
Romania

Foreword

Not being a big friend of rituals and formalities, I was thinking about leaving out the usual sermon of thank-yous that opens all the theses I have read. However, I now have strong doubts that I will explicitly express the gratitude I feel in the context of this thesis towards many people. Therefore I have decided to follow the tradition of listing them here anyway.

Foremost I want to thank both my supervisors. Victor Mitrana who not only introduced me to the topic of duplication but kept me active in the beginning by continuously inviting me to participate in his work. Juhani Karhumäki first suggested the generalization to idempotency and I have learned a great many things about birds and the Finnish outdoors in general on the various excursions, on which I could accompany him. Both have helped and guided me as much as I have let them, although I am not an asker of many questions.

Further thanks are due to many people, who have helped me over the last few years. The wonderful circumstances of the PhD School in Formal Languages and Applications were created by Carlos Martín Vide, who never seems to tire of searching for sources of funding. Masami Ito was a very kind and generous host during my stay in Kyoto. Matteo Cavaliere was a wonderful partner for both political discussions and scientific work during my first years in Tarragona. More than anyone else Rebeca Tomás Smith and Rafel Escoda Rosich have made me feel at home in Tarragona, and most of my Català I owe to them.

Among my co-authors on the topics of this thesis, those not yet mentioned above are José Sempere and Kayoko Shikishima-Tsuji. Also the interaction with them was important for my ideas to evolve to the state presented here. In this context also a number of anonymous referees should be mentioned, whose comments helped to greatly improve some of the work presented here.

Human beings do not live on air alone, even mathematicians need someone to support them economically. In my case this has mainly been done by the Spanish Ministry of Culture, Education and Sport under the Programa Nacional de Formación de Profesorado Universitario (FPU); it has also facilitated two stays in Turku and one in Kyoto.

Before this, the Spanish Foreign Ministry supported me under the programme BecasMAE. Further, I am thankful for travel grants to the conferences WORDS03 in Turku, DNA10 in Milano, and CANT06 in Lüttich as well as two short trips to Budapest, Szombathely, and Debrecen financed by a Hungarian-German Project headed by Manfred Kuflek. All these travels have taught me that, besides concentrated thinking and reading, listening to others and trying to explain your thoughts are essential to get ahead in mathematics.

Finally, I should also thank the beautiful land of Catalunya. With its marvels from the sunny beaches over the nearby mountain ranges to the Pyrenees it has very often successfully seduced me away from my work; but otherwise I would probably have had much less inspiration and motivation during the times of working. And with the abandoned herdsmen's shelters, the Serra de Montsant even provides perfect locations for meditating about intricate problems, be they of mathematical or other nature.

Tarragona, September 2006
Peter Leupold

Contents

0 Introduction

Theoretical Computer Science has developed and also adopted quite a number of significantly different fields. Among these, the work to be presented here belongs most to Formal Language Theory as it emerged from Noam Chomsky's definition of generative grammars in the 1950s. However, we will heavily use results and methods from two more fields, namely Combinatorics on Words and String-Rewriting Systems; both of these can be traced back to the work of Axel Thue in the beginning of the twentieth century, long before the advent of electrical computers and what is called computer science now.

To start with, we will present in Chapter 1 fundamental concepts from the three fields of Formal Language Theory, Combinatorics of Word and String-Rewriting Systems; all of these will be used in our later investigations and therefore constitute an indispensable basis for the remainder of this thesis.

Much of the current work in Formal Language Theory has been inspired by mechanisms observed in molecular biology. Most prominently, the computational power of recombinations occurring in DNA is investigated, when applying these operations on general strings. Also our work has its origin in such a DNA operation, namely in duplication.

Chapter 2 will outline the original motivation for introducing the formal language operation of duplication in context with other DNA-inspired string operations. Then its generalization to idempotency languages is described. A few spotlights are shed on the history of idempotencies in the parts of Algebra related to formal languages, most mentionable on the famous Burnside problem and the problem of non-counting classes. After this, the actual investigations on idempotency languages are presented.

Starting out from a few results on special cases treated in earlier work of other authors, we mainly focus on two types of questions. For one thing we try and determine, which relations are confluent. Secondly, we examine whether the languages generated by them are regular.

First off, we treat the most restricted variant, uniformly bounded

idempotencies. Here all rewrite rules must have the same length. This makes the problems quite resolvable, and the conditions for confluence and regularity are fully characterized for all possible combinations of parameters.

Already for the following variant, bounded idempotencies, where only an upper bound is imposed on the rules' length, more cases are left open. Finally, for unrestricted idempotency relations we present mainly results that carry over from the restricted cases. Interesting questions like the context-freeness of duplication languages remain open.

Contrary to the chronological development we then come from general idempotency languages to duplication languages in Chapter 3. Some results are presented, which have not been generalized to general idempotencies and which seem especially interesting in the context of the original motivation for duplication from DNA computing. But before that we try and shed some light on the reasons, why it is so difficult to determine, whether general duplication languages are context-free. Further a few decision problems related to duplication are treated and shown to be decidable.

Section 3.2 then introduces the concept of idempotency root. This is motivated by recalling the primitive root of words, then some results concerning duplication roots. The main interest is on the finiteness of roots and the decidability of this property.

In Section 3.3 we define a type of code, which is robust under uniformly bounded duplications in the sense that such duplications occurring in the code words do not affect the uniqueness of factorization. Among other things the conditions are characterized, under which infinite such codes exist, and the density of languages generated by these codes is investigated.

Finally, in Section 3.4 we examine the closure of the classes of regular and context-free languages under duplication in its differently length-bounded variants. Mainly bounded duplication is treated, for example the closures of regular and context-free languages under this operation is established.

A few concluding thoughts and a more detailed exposition of a small number of selected problems left open form the conclusion of this thesis.

The majority of the results that will be presented here has already been published in scientific journals and been presented at conferences. Because in the text we will not point out the place of publication of single results obtained by the current author, we now give an overview of where these can already be found in the literature. Of

course, slight improvements of proofs and presentation have been implemented in many places.

Sections 2.4 and 2.5 are based on an article accepted for publication in Theoretical Computer Science [53]. The following Section 2.6 is mainly based on an article accepted for publication in the Journal of Languages, Automata and Combinatorics [54], some of the results for three and more letters are again from the article about the bounded case [53].

The considerations on the general duplication language starting Chapter 3 are yet unpublished. Some of the following results on duplication represent parts of two articles in Discrete Applied Mathematics [56] and the LNCS volume dedicated to Tom Head [58]; many of the results in these two articles are, however, implied by more general ones stated already in Chapter 2.

The results concerning primitive roots in Section 3.2.1 constitute part of the work presented at WORDS 2003 in Turku [50], while the remainder of Section 3.2 is formed by a talk given at the Theorietag Automaten und Formale Sprachen of the Gesellschaft für Informatik in Wien [55].

Section 3.3 presents results on duplication codes accepted for publication in RAIRO Informatique Théorique [57] and in part presented earlier at WACAM 2005 in Turku [51]. Finally, the duplication closure of languages treated in Section 3.4 has been presented at Developments in Language Theory 2006 in Santa Barbara [47].

1 Formal Languages and Combinatorics of Words

Before coming to the actual topic of our treatise, we need to introduce the notions and tools we will employ. The results we will present in what follows belong mainly to Formal Language Theory. Therefore we now introduce the concepts from this field that will be referred to later on. There are, however, two more fields of investigation, the results of which we will use very frequently. These are Combinatorics of Words and the theory of string-rewriting systems.

The fundamental feature connecting these three fields is the concept of word as a sequence of symbols. Since single words are the focus of Combinatorics on Words, we will take this as our starting point. Then we move on to formal languages, i.e. sets of such words, and to string-rewriting systems.

All the notions particular to the three fields and needed in our investigations mentioned will now be defined. However, the reader is supposed to be familiar with basic mathematical terminology and notation as used in set theory, algebra, and propositional and predicate logic. References for further reading in each of the fields presented here are suggested in the respective sections.

1.1 Combinatorics of Words

The concept of word we will use deviates significantly from the one common in everyday use, where mainly words as in natural human languages are meant. In this context the concept usually comprises a semantic component. Thus Miller [71] states that words are "the building blocks of language," and in his linguistic approach he assumes every word to consist of three fundamental aspects: it is "a synthesis of a concept, an utterance, and a syntactic role."

This means that there is a concept in our mind, a phonetic sequence that in some way we associate to that idea, and finally there is a specific way to use this sequence in interaction with others to form a sentence. However, in a naive approach to words a human

being not polluted by prior exposition to such theories will most probably describe a word simply as a sequence of sounds, or –in its written form– as a sequence of letters, thus focusing on only the second one of the three aspects described by Miller.

When investigating combinatorial aspects of words, we also take this latter, basic point of view. While disregarding a word's possible use, place of occurrence, interaction with other words, meaning etc., we do partition it into its physical building blocks, its letters. Thus a word is simply a sequence of symbols over a given alphabet. Nothing exceeding this very abstract view is considered. Thus in common terms we are speaking about sequences rather than words, but the term *Combinatorics of Words* is well-established for this.

This concentration on the basic concept of sequentiality explains the wide applicability of results from combinatorics of words. To some extent human perception of the world is essentially sequential. If we take three-dimensional space as one fundamental dimension of our perception, then time is the second one. And time we perceive essentially as a temporal succession of observed events, states etc. — as a sequence. Depending on the aspect of the world we are considering, different features of such sequences are of interest; but in many cases combinatorics on words can be used to state them in an abstract manner and to establish some of their properties.

One of the central points of interest in this is the study of repetitions. They seem to be a feature of sequences which greatly attracts the attention of human beings. A repeated rhythm will stick out from other sounds, trees planted in patterns will catch our eye when looking at an otherwise irregular landscape etc.

Repetitions in linear sequences of symbols were first investigated by Thue at the beginning of the twentieth century [89, 90]. He determined whether certain repetitions are bound to occur, i.e. whether they are unavoidable in a long enough sequence over a limited alphabet of symbols. The most important results are the facts that over two letters no square-free word of length greater than three exists, while over three and more letters infinite square-free words can be constructed. A nice summary of his work was given by Berstel [8].

Another indicator towards the fundamentality of repetitions in sequences is the big number of times that his results have been discovered again in later years without knowledge of his work and in quite different contexts. Most prominently in this respect is certainly the work of Morse both in his mathematical studies [74] and his investigations on the possibility of endless chess games [75], more rediscoveries are listed in Berstel's article.

The standard reference for nearly all topics in Combinatorics of Words consists in the three books of Lothaire [61, 62, 63]. The Handbook of Formal Languages [81] contains a separate chapter on combinatorics. Also Berstel and Pin's book on infinite words contains many related results [10], as does the book on automatic sequences by Allouche and Shallit [4]. We now proceed to provide the definitions and concepts from this field that we will make use of later on.

1.1.1 Words and Periodicity

As already mentioned, for us a word is a sequence of symbols over a finite alphabet. This includes the word consisting of no symbol, which is called the *empty word* and denoted by λ. The words generated by the alphabet Σ together with catenation form the free monoid denoted by Σ^*.

The length of a finite word w is the number of not necessarily distinct symbols it consists of and is written $|w|$. The number of occurrences of a certain letter a in w is $|w|_a$. The set of all letters occurring in w is its alphabet $\mathrm{alph}(w)$. The i-th symbol we denote by $w[i]$. The notation $w[i \dots j]$ is used to refer to the part of a word starting at the i-th position and ending at the j-th position.

A word u is a *prefix* of w if there exists an $i \leq |w|$ such that $u = w[1 \dots i]$; if $i < |w|$, then the prefix is called *proper*. The set of all prefixes is $\mathrm{pref}(w)$. A *suffix* is a word u such that $u = w[i \dots |w|]$, and a *factor* is any word such that there exist i and j such that $u = w[i \dots j]$. A *scattered subword* of w, in contrast, is a word u for which there exist integers $i_1 < i_2 \cdots < i_{|u|}$ such that for all $j \in \{1, 2 \dots |u|\}$ there is $u[j] = w[i_j]$.

We now turn to periodicity; a word w has a positive integer k as a *period*, if for all i, j such that $i \equiv j \pmod{k}$ we have $w[i] = w[j]$, if both $w[i]$ and $w[j]$ are defined. In this case, w is said to be k-periodic. w is *weakly k-periodic*, if it fulfills this condition for $j = i + k$ instead of $i \equiv j \pmod{k}$. These two notions are equivalent. We write $p(w)$ for the minimal period of the word w and $P(w)$ for the set of all its periods.

A famous result dealing with periodicity is the following theorem, which in its original form is due to Fine and Wilf and can be found in several forms in the book of Lothaire [61]. However, we present it in a slightly different variant more apt to our needs later on.

Theorem 1.1.1. *If a word w has two periods k and ℓ, then also* $\gcd(k, \ell)$ *is a period of w.*

Occasionally we will also speak about infinite words, more exactly about right-infinite words. These have a starting point on the left-hand side, but on the right-hand side they continue forever. The set of all these word is denoted Σ^ω, and the exponent ω will denote infinitely iterated catenation to the right in general.

1.1.2 Special Types of Words

Via certain properties special types of words are defined. We do this already in natural language, for example with palindromes, which will be defined further down. Mainly, however, the types of words interesting to us will be defined by properties exclusively motivated from combinatorics.

A word is *primitive*, iff it is not a non-trivial (i.e. with exponent one) power of any word. Thus u is primitive, if $u = v^k$ implies $u = v$ and $k = 1$; this means that λ is not primitive, because, for example, $\lambda^4 = \lambda$. It is a well-known fact that for every non-empty word w there exists a unique primitive word p such that $w \in p^+$; this primitive word is called the *(primitive) root* of w and we will denote it by $\sqrt{w}$. The unique integer i such that $\sqrt{w}^i = w$ is called the *degree* of w.

The next property has been defined under numerous names, see also Section 3.2.2, we will only give the two most widely used ones here. A word is called *unbordered*, also called *non-overlapping*, iff none of its proper prefixes is also one of its suffixes; all other words are called *bordered* or *overlapping*.

For a word w, by w^R we denote its reversal, that is $w[|w|-1\ldots0]$. If $w = w^R$, the word is called a *palindrome*; the English words *mom* and *dad* or the German *Esse* are natural language examples of palindromes.

We now come to avoidability, which deals with the question, whether certain subsequences are found in a word. For a rational number r, a non-empty word w is a repetition of order r, iff there exists a word u such that w is a prefix of u^ω and $\frac{|u|}{|w|} = r$. For the integers 2 and 3 repetitions of the respective order are called *squares* and *cubes*. We will also use rational exponents to denote non-integer powers of words in the following way: $(aba)^{\frac{5}{3}} = abaab$.

Avoiding a certain repetition means not having any factor that constitutes such a repetition. Thus a word is called *r-free*, iff it does not contain any repetition of order r. If the word may contain repetitions of order r but not of any greater order, then we call it r^+-free.

These notions of avoidability are used also for infinite words.

Thue's pioneering work stated among other facts the fundamental results that over two letters there are no infinite square-free words, while there are 2^+-free words; over three letters, however, there exist infinite square-free words [89, 90].

1.2 Classical Formal Language Theory

When we look at sets of words rather than individual words, we take the step from Combinatorics of Words to Formal Language Theory. Here the most common questions concern the complexity of a given set of words – called a (formal) language – in terms of generating or accepting devices. There exist several classical such hierarchies, which we will introduce briefly. Further, we will present some important properties of selected classes of languages. Standard references for these results are the books by Salomaa [82] and Harrison [36] as well as the Handbook of Formal Languages [81].

1.2.1 Generative Devices

If the sets of words we deal with are called languages and not anything else, this is mainly due to the fact that they were first dealt with in a linguistic context. In the 1950s Noam Chomsky defined generative grammars in an attempt to formalize the mechanism, by which human beings produce utterances in their language. He introduced a hierarchy, grouping this type of grammars by the complexity of their rules.

While none of these classes were found to be completely adequate for the description of human languages, they did prove to be very useful in many other fields. Thus the mentioned hierarchy is still the standard reference point for determining the complexity of languages in Formal Language Theory.

A *(generative) grammar* in the sense of Chomsky is a quadruple $G = [\Sigma, N, S, P]$, where Σ is the alphabet of *terminals*, N is the alphabet of *non-terminals* disjoint from Σ, and $S \in N$ is the *start symbol*. The set P of *productions* or *rules* is a subset of $(\Sigma \cup N)^* \times (\Sigma \cup N)^*$.

With such a grammar G we associate a derivation relation $\Rightarrow_G$ as follows: for words $u, v \in (\Sigma \cup N)^*$ we have $u \Rightarrow_G v$ iff there exist factorizations $u = u_1u_2u_3$ and $v = u_1v_2u_3$ such that $(u_2, v_2) \in P$, i.e. by application of one rule we can transform u into v. Applying a rule means to find its left side as a factor in a given word and to replace it with the right side.

Let $\Rightarrow_G^+$ denote the transitive closure of this relation. Then the language generated by G is defined as $L(G) := \{w : w \in \Sigma^* \wedge S \Rightarrow_G^+ w\}$. This means that $L(G)$ consists of all the strings that are made up of only terminal symbols and that can be reached from the start symbol via the derivation relation.

There are several restricted types of generative grammars, which are of interest. A generative grammar $[\Sigma, N, S, P]$ is called

- *(left-)regular* iff all rules in P are of the form $[A, xB]$ for $A \in N$, $B \in N \cup \{\lambda\}$ and $x \in \Sigma$,
- *linear* iff all rules in P are of the form $[A, xBy]$ for $A \in N$, $B \in N \cup \{\lambda\}$, and $x, y \in \Sigma \cup \{\lambda\}$,
- *context-free* iff all rules in P are of the form $[A, u]$ for $A \in N$, and $u \in (\Sigma \cup N)^*$,
- *context-sensitive* or *non-decreasing* iff all rules in P are of the form $[v, u]$ for $u, v \in (\Sigma \cup N)^*$ with $|v| \leq |u|$.

The classes of languages generated by these types of grammars have the corresponding names. For the last one only the term context-sensitive is in use. They are denoted by *REG*, *LIN*, *CF*, and *CS* respectively. Further *FIN* denotes the class of finite languages, while generative grammars without restrictions generate the class *RE* of recursively enumerable languages. Now we state the result justifying the name Chomsky-Hierarchy for these classes.

Theorem 1.2.1. *$FIN \subset REG \subset LIN \subset CF \subset CS \subset RE$ and all these inclusions are proper.*

Another very convenient form of expressing regular languages comes from their definition as *rational* languages. These are the closure of the singleton sets containing the letters under union, catenation, and Kleene-star; this is called the rational closure. We now define *regular expressions* and their corresponding languages (their interpretations ϕ) as follows:

- if a is in Σ, then a is an expression; its interpretation is $\{a\}$;
- if e_1 and e_2 are expressions, so is $(e_1 \circ e_2)$; its interpretation is $\phi(e_1) \cdot \phi(e_2)$;
- if e_1 and e_2 are expressions, so is $(e_1 \vee e_2)$; its interpretation is $\phi(e_1) \cup \phi(e_2)$;

- if e is an expression, so is $(e)^*$; its interpretation is $\phi(e)^*$.

There are no other expressions. Every clause of the definition corresponds exactly to one part of the definition of rational closure.

In general, we will leave away the interpretation function and speak, for example, of the language ab^* instead of $\phi(ab^*)$; note that here the star has higher precedence than catenation and that we leave away the $\circ$ as well as some parentheses. Thus ab^* stands for $(a \circ (b^*))$. All these simplifications are standard in the literature and should not confuse the reader. Another standard abbreviation we will use is denoting singleton sets $\{a\}$ simply by their unique member a, if this cannot lead to confusion in the respective context.

1.2.2 Accepting Devices

While grammars are good for generating words, on might also for a given word want to find out, whether it belongs to a given language. This is known as the *word problem*, and in our context it is answered by acceptors of languages.

A device accepts a language, if it computes its characteristic function; this means it gets as an input a word, and as output it says YES, if this word belongs to the language in question, otherwise it outputs NO or runs forever. There is a very rich theory of this type of automata. In particular, there is for each class of languages from the Chomsky Hierarchy a class of automata, which accept exactly those languages. Here we introduce only the two types of automata that will play a role later on.

For the regular languages the corresponding devices are called *deterministic finite automata*. Such a DFA is a tuple $[Q, \Sigma, \delta, q_0, F]$. Q is the set of states, Σ the input alphabet. $q_0 \in Q$ is the start state, $F \subseteq Q$ is the set of final states. The transition function δ is a mapping $Q \times \Sigma \mapsto Q$. The function δ^* is its extension to $Q \times \Sigma^*$ such that $\delta^*(q, w) := \delta(\delta(\ldots\delta(\delta(q, w[1]), w[2])\ldots w[|w|-1]), w[|w|])$. The graphic idea behind this is that a reading head moves along the input word and changes its state according to the letters it finds. The word is accepted, if this ends in a final state.

Thus the language accepted by such a deterministic finite automaton A is defined as $L(A) := \{w : \delta^*(q_0, w) \in F\}$. The class of languages accepted by this type of device we denote by $L(DFA)$.

If δ is not a function, but can be any type of relation, then the device is called a *(non-deterministic) finite automaton*, FA. For a given pair of state and input letter there can be some choice for the following

state, and a word is accepted if there exists one computation that halts in a final state. The class of languages accepted is denoted by $L(FA)$. While non-deterministic finite automata are often more compact in the size of the state set for a given language, they are not more powerful than their deterministic counterparts and coincide with the regular languages.

Theorem 1.2.2. *$REG = L(DFA) = L(FA)$.*

Maybe the strongest factor limiting the power of finite automata is the fact that they do not have any explicit way of storing information, i.e. they do not have memory. When we add such a memory in the form of a (push-down) stack to them, we obtain a *push-down automaton*. These are tuples $[Q, \Sigma, \Gamma, \delta, q_0, \gamma_0, F]$, where Q, Σ, q_0, and F are as for finite automata. Γ is the stack alphabet, and γ_0 is the bottom-of-stack symbol. δ this time is a mapping $Q \times \Sigma \times \Gamma \mapsto Q \times \Gamma^*$.

The interpretation here is that the PDA reads in every step an input symbol and the top-most symbol on the stack. According to this it changes its state and can put an arbitrary string onto the top of the stack. The language accepted is defined analogously to the one for finite automata. Also here deterministic and non-deterministic automata are considered, and the non-deterministic PDAs correspond exactly to the context-free languages.

Theorem 1.2.3. *$CF = L(PDA)$.*

In contrast to the case for regular languages, here the deterministic version of automata does not have the same power as the non-deterministic one. The class of languages accepted by the former type of device are the *deterministic context-free languages*, *DCF*.

Theorem 1.2.4. *$DCF \subset CF$. $CF \setminus DCF \neq \emptyset$.*

Although context-sensitive and recursively enumerable languages will not play a big role in what follows, we want to mention here that they are accepted by *non-deterministic linear bounded automata* and *Turing machines* respectively.

1.2.3 Closure Properties and Miscellanea

Closure under an operation is in our context a property of language classes. A class is said to be closed under an operation, if the action of this operation on languages of the respective class results in a language, which belongs again to the same class. We consider

operations acting on one language as well as ones acting on two languages. We provide a short list of the ones important in our context. Most of them should be well-known from set theory.

- *Complement* is a unary operation denoted by $\overline{W}$,
- *union* is a binary operation denoted by $V \cup W$,
- *intersection* is a binary operation denoted by $V \cap W$,
- *intersection with regular languages* is a unary operation denoted by $W \cap REG$

for languages V, W. Closure under the last one of these operations means that $W \cap U$ is in the same class as W for all languages $U \in REG$. The complement is relative to the alphabet and is the set $\Sigma^* \setminus W$.

The classes from the Chomsky Hierarchy have the closure properties listed in Table 1.1, where Y signifies closure and N stands for the respective class not being closed.

	Compl	$\cup$	$\cap$	$\cap REG$
REG	Y	Y	Y	Y
LIN	Y	Y	Y	Y
CF	N	Y	N	Y
CS	Y	Y	Y	Y
RE	N	Y	Y	Y

Table 1.1: Closure Properties.

Another important property of language classes is the *decidability* of certain questions, most prominently of the word problem: given the language L and a word w, is it possible to find out with with an effective algorithm whether $w \in L$ is true? Without going into any detail, an effective algorithm here is any computation method in a complete model of computation like the Turing Machine. More about decidability can be found in the references given for Formal Language Theory in general and in more depth in the very entertaining book by Rozenberg and Salomaa [80].

In a more algebraic view of languages, often the relation $\sim_L$ over $\Sigma^* \times \Sigma^*$ for a language L plays an important role. It is called the *syntactic right-congruence* of L and is defined as follows:

$$u \sim_L v :\leftrightarrow \forall w \in \Sigma^* (uw \in L \leftrightarrow vw \in L).$$

This is obviously an equivalence relation. It is well-known that a language L is regular, if and only if the corresponding relation $\sim_L$ has a finite number of equivalence classes; this number is called the *index* of $\sim_L$. Its relation to regular languages follows from a theorem of Myhill.

Theorem 1.2.5. *A language L is regular, if and only if $\sim_L$ has finite index.*

This provides us with yet another characterization of regular languages after finite automata, regular grammars, and regular expressions. Next we state a property, which every regular language fulfills, but which also other languages can fulfill. Thus it is necessary but not sufficient for regularity and can mainly be used to show that a given language is not regular.

Lemma 1.2.6. *For every regular language L there exists an integer k such that every word $w \in L$ longer than k, has a factorization $w = w_1w_2w_3$ such that $w_2 \neq \lambda$, $|w_1w_2| \leq k$ and $w_1w_2^*w_3 \subseteq L$.*

This *pumping* property has its name from the fact that the factor w_2 can in some sense be pumped arbitrarily without obtaining words outside the language. A similar property exists for context-free languages. However, here the pumping occurs at two sites simultaneously. In this case many stronger versions like the Ogden-Lemma or the Interchange-Lemma are known, but for our purposes the basic and original version stemming from Bar-Hillel will suffice.

Lemma 1.2.7. *For every context-free language L there exists an integer k such that every word $w \in L$ longer than k, has a factorization $w = w_1w_2w_3w_4w_5$ such that $w_2w_4 \neq \lambda$, $|w_2w_3w_4| \leq k$ and $w_1w_2^iw_3w_4^iw_5 \in L$ for all $i \geq 0$.*

In some contexts the actual sequence of letters is not so essential, and we are interested only in the numbers in which the different letters occur in a word. Then we look only at vectors of dimension $|\Sigma|$, whose i-th component is the number of occurrences of the i-th letter in the corresponding word. This correspondence is established by the so-called *Parikh mapping* of a word w, which is defined as $\psi(w) := (|w|_{a_1}, |w|_{a_2}, \ldots, |w|_{a_{|\Sigma|}})$. It is extended in the canonical way to languages as $\psi(L) := \{\psi(w) : w \in L\}$. Note here that different words can be mapped to the same vector.

For sets of vectors over $\mathbb{N}^k$ there exists the notion of being *linear*, which means that such a set A can be generated from a finite number of vectors $r_0, r_1, \ldots, r_\ell \in \mathbb{N}^k$ such that $A = \{r_0 + m_1r_1 + \cdots + m_\ell r_\ell :$

$m_1, \ldots, m_\ell \in \mathbb{N}\}$. If a set is a finite union of linear sets, it is called *semi-linear*. A language is called semi-linear, iff its Parikh set is semi-linear. With this we can now state Parikh's theorem, which provides us with another necessary condition for context-freeness.

Theorem 1.2.8. *All context-free languages are semi-linear.*

There are several special classes of languages more that will occur in what follows and which are defined by different means than we have seen up to this point. A language L is called

- *non-counting*, iff there is an integer $i \geq 0$ such that for every $y \in \Sigma^+$ and $x, z \in \Sigma^*$, we have $xy^i z \in L$ iff $xy^{i+1}z \in L$,
- *dense*, iff for all $w \in \Sigma^*$ we have $\Sigma^* w \Sigma^* \cap L \neq \emptyset$,
- *bounded*, iff there exists a finite number of words $w_1, w_2, \ldots, w_k$ such that $L \subseteq w_1^* w_2^* \cdots w_k^*$,
- *slender*, iff there is a number k such that it never contains more than k words of any given length, or more exactly $|\Sigma^n \cap L| \leq k$ for all $n > 0$.

The class of regular non-counting languages is equal to the so-called *star-free* languages [68]. These are the languages obtainable from the an alphabet's letters by a finite number of applications of the operations union, intersection, concatenation, and complementation.

1.3 String-Rewriting Systems

Axel Thue can be seen not only as the founder of the field of Combinatorics on Words as explained in Section 1.1, but he also introduced what is today known under the name of rewriting system [91]. Named after him such systems acting on strings are sometimes called *semi-Thue* systems. Such a system consists basically of a set of rules, which are applied on a word containing the rule's left side by replacing this by the rule's right side. For example, the rule systems of generative grammars constitute an application of this type of system. We will call them by their most common name, string-rewriting systems, and now proceed to define them formally.

In our notation we mostly follow Book and Otto [12] and define a *string-rewriting system* R on Σ to be a subset of $\Sigma^* \times \Sigma^*$. Its single-step reduction relation is defined as $u \to_R v$ iff there exists $(\ell, r) \in R$

such that for some u_1, u_2 we have $u = u_1 \ell u_2$ and $v = u_1 r u_2$. We also write simpler just $\to$, if it is clear which is the underlying rewriting system. By $\overset{*}{\to}$ we denote the relation's reflexive and transitive closure, which is called the *reduction relation* or *rewrite relation*.

A string w is *irreducible* iff there is no rule $(\ell, r) \in R$ such that ℓ is a factor of w, i.e. no rule can be applied on w. The set of all the strings irreducible with respect to a string-rewriting system R is denoted by $IRR(R)$. An irreducible string v such that $u \overset{*}{\to} v$ is called a *normal form* of u.

We distinguish several special types of rewrite relations. Such a relation $\to$ is called *confluent*, iff for all $w, w_1, w_2 \in \Sigma^*$ always $w_1 \overset{*}{\leftarrow} w \overset{*}{\to} w_2$ implies the existence of some w' such that $w_1 \overset{*}{\to} w' \overset{*}{\leftarrow} w_2$. Here we use $w_1 \leftarrow w$ as a sometimes convenient way of writing $w \to w_1$.

Local confluence is given, iff for all $w, w_1, w_2 \in \Sigma^*$ always $w_1 \leftarrow w \to w_2$ implies the existence of some w' such that $w_1 \overset{*}{\to} w' \overset{*}{\leftarrow} w_2$. A still more local condition is the *diamond property*, which holds iff $w_1 \leftarrow w \to w_2$ implies the existence of some w' such that $w_1 \to w' \leftarrow w_2$. Its relation to general confluence is the following.

Proposition 1.3.1. *A string-rewriting system which fulfills the diamond property is confluent.*

Further, $\to$ is *noetherian* (also *terminating*), iff there is no infinite sequence $u_0, u_1, \ldots$ such that $u_i \to u_{i+1}$ for all $i \geq 0$. The relation is *convergent* iff it is both confluent and noetherian. For noetherian systems an analogous result holds for local confluence.

Proposition 1.3.2. *A string-rewriting system which is locally confluent and noetherian is confluent.*

Thus the diamond property implies confluence, which in turn implies local confluence. To see that the opposite of the first implication does not hold we provide a small example without rigorously proving it.

Example 1.3.3. The system $R = \{(a, aa), (b, bb), (abb, aaabbb)\}$ can rewrite a word abb in one step to $aabbb$. This result can also be reached by applying first (a, aa) via the three steps $abb \to aabb \to aaabb \to aaabbb$. It is rather easy to see that this system is confluent, since the first two rules can in this way simulate applications of the third one. However, R does not fulfill the diamond property as can be seen from the reduction described.

By imposing restrictions on the format of the rewriting rules, many special classes of rewriting systems can be defined. Following Hofbauer and Waldmann [39] we will call a rule (ℓ, r) *context-free* (*inverse context-free*), if $|\ell| \leq 1$ ($|r| \leq 1$). The class of rewriting-systems with only (inverse) context-free rules we denote by CF (InvCF). A system is *monadic*, if it is inverse context-free and for all its rewrite rules (ℓ, r) we have $|\ell| > |r|$. The class of monadic systems is denoted by mon.

1.4 Accepting Languages with String-Rewriting Systems

The main object of this treatise, idempotency languages, will be generated by rewrite relations over strings. Therefore it will sometimes be very convenient to use string-rewriting systems to determine their location in the Chomsky Hierarchy. For this reason we now introduce the McNaughton languages, which connect the Chomsky Hierarchy with string-rewriting systems.

It is a very natural idea to let a string-rewriting system accept a language in the following way: if a given input is reducible to a specific normal form, then it is part of the language; otherwise it is not. A mechanism of this type was first defined by McNaughton et al. [67], later investigated in more detail by Beaudry et al. [7]. Finally, Woinowski formalized this in so-called *Church-Rosser language systems* [95].

We do not need to use the entire formalism of these systems here and therefore simply say that a language $L \subseteq \Sigma^*$ is a McNaughton language of a finite string-rewriting system R, iff there exist an alphabet Γ containing Σ, strings $t_1, t_2 \in (\Gamma \setminus \Sigma)^* \cap IRR(R)$ and a letter $Y \in (\Gamma \setminus \Sigma) \cap IRR(R)$ such that for every word $w \in \Sigma^*$ we have $w \in L$ if and only if $t_1 w t_2 \overset{*}{\Rightarrow}_R Y$. This is denoted by $L \in R\text{-McNL}$. Note that one system can accept several languages with different strings t_1 and t_2.

A class of finite string-rewriting systems $\mathcal{S}$ defines its corresponding *McNaughton family of languages* $\mathcal{S}$-McNL in the canonical way such that $\mathcal{S}$-McNL consists of all languages accepted by at least one rewriting system from the class $\mathcal{S}$. Without restrictions, string-rewriting systems are computationally complete in this sense.

Theorem 1.4.1 ([7]). *The family of all McNaughton languages coincides with the class of recursively enumerable sets.*

Since we will mainly deal with regular and context-free languages, the following result is actually of more interest to us.

Theorem 1.4.2 ([7]). $Mon\text{-}McNL = CF.$

The idempotency relations we will use to generate languages are, of course, also interesting in this context. More exactly speaking, it is their inverses, which can reduce generated words back to the origin. These belong to a class of rewrite relations called the *length-reducing* ones; here essentially every left side of a rule must be longer than the corresponding right one. This class $\mathtt{lr}$ accepts in the McNaughton sense the *growing context-sensitive* languages, which are located properly between the classes of context-free and context-sensitive languages. Using confluent systems one obtains only a smaller class of languages, which is incomparable to the context-free languages.

Theorem 1.4.3 ([7]). $lr\text{-}McNL = GCSL$ *and* $lr\text{-}McNL \setminus con\text{-}lr\text{-}McNL \neq \emptyset.$

1.5 Variables

In what follows we will try to use variables in a systematic way, that is, the same variable should always denote the same type of entity. Before starting out, we want to provide this classification for variables, because it might make reading slightly easier at times.

a, b, c, d	will not denote variables, but the letters of our alphabets
i, j, k, l, m, n	integers
u, v, w, z	words
p, q	words that are in some sense primitive
x, y	single letters
r, s, t	used where there are not enough other lower case letters
L, U, V, W	sets of words, i.e. languages
Σ, Γ, N	alphabets
A, B, C, D, T	non-terminals of grammars
$f, g, h, \phi, \delta, \psi$	mappings

These variables might not always be explicitly quantified, while all other variables shall not be used without proper quantification.

2 Languages Generated by Iterated Idempotencies

Among the many variants of idempotency relations that we will now introduce, duplication was the one standing at the origin of all the work presented here. Further it is the main one having a strong motivation from outside of pure mathematics, namely it was first introduced in the context of DNA computing. We will briefly sketch the development from duplication languages to general idempotency languages in an informal manner, starting out with a survey of all DNA-inspired string operations. Then the languages generated by iterated idempotencies are formally defined.

After this, we provide some more background on idempotencies in the context of formal languages, namely on the Burnside and Brzozowski problems. Then we will summarize scattered results from several lines of research that have already treated idempotency languages, though under different names. With all these foundations laid and the scientific context described we then proceed to investigate the regularity of languages and the confluence of relations for the numerous variants of idempotencies.

2.1 From DNA to Generalized Idempotency

From the very beginning of electrical computers, miniaturization of their components has been a major aim of research. On this path, we have come from large condensers to today's microscopic integrated circuits on silicon chips, from computers occupying entire buildings to laptops and smart phones. The famous law of Moore predicts in its original form that every year the number of components per square inch on integrated circuits will double [73]; later he corrected the period of time from one to two years.

And up to now this has roughly held true; actually the number lies just in between the two predictions, as the number of circuits per square inch has doubled every 18 months approximately. However, at some point this miniaturization will come to an end due to physical limits — as far as we can see from the current state of knowledge,

electrical computers will always require conducting lines of many molecules in diameter and even more in length.

On the other hand, our need for computation seems to increase even faster than our computers' power; simulations of the Earth's climate, processing of astronomical data and many other tasks encounter their limits mainly in the capacities of the computers at their disposal. Thus it is only natural to search for fundamentally new ways of building computers, possibly using some of the smallest building blocks of our world that we know of, i.e. atoms, even single electrons, or at least molecules consisting of not very many atoms.

Besides the use of quantum mechanical effects, biochemical reactions seem to be the most promising candidate for this. A great number of theoretical models have been proposed, which make use of some special interaction among molecules. The most frequently employed molecule in this context is DNA. We will now survey the naturally occurring operations in DNA strands and sketch the way from these phenomena to the definition of idempotency languages.

2.1.1 String Operations Inspired by DNA

Already in its usual function as carrier of genetic information DNA acts as a very compact information storage. But beyond this, it exhibits many ways of rearranging itself, often in interaction between two strands, which one could interpret as a computation. Here we will not go into any biochemical detail. A reader familiar with DNA at the level presented in any high school book should be able to follow the presentation. We want to distinguish two different classes of DNA rearrangements.

Firstly, there is the Watson-Crick complementarity between the two strands aligned with each other in DNA. When the double helix is split into its two strands, these tend to align with complementary strands again. This was employed in the seminal work of the field by Adleman [3]. From the ways in which a certain set of strands align, he concludes whether a coded problem has a solution or not. Since this is not the path we will follow, we only mention one more way of using complementarity: Watson-Crick automata work on a tape with two complementary strands [34].

Secondly, certain changes can occur inside the strands, changing their sequence of bases. Here we can disregard the double-strand structure and rather see them as normal words. By iterating this type of operation, one obtains a language. Thus they are typically used as generating devices in the context of Formal Languages. Dassow and

Mitrana [24] as well as Searls [84] discuss different formal operations on strings related to the language of nucleic acids. Dassow et al. [27] give a nice overview of DNA-inspired operations on formal languages. In these articles, the following operations play a role:

Deletion is the removal of a factor from a word. It has mainly been investigated together with

Insertion, which is the operation that adds a new factor at an arbitrary position in a word. A summary of the power of different variants of so-called Insertion-Deletion Systems, which employ both operations, can be found in the book by Păun et al. on DNA Computing [76]

Inversion is the replacement of a factor by its mirror image. It has not proven very fruitful for computation so far.

Transposition moves a factor to a new position within the same word. It does not appear very apt for computation either.

Cross-over is an operation involving two strands. These are cut and then put back together in a crosswise manner, therefore the name. Figure 2.1 depicts the exact way, in which this happens. Strands u and v are cut into the pieces γ/δ and α/β respectively. Then these are attached cross-wise with each other. The result are

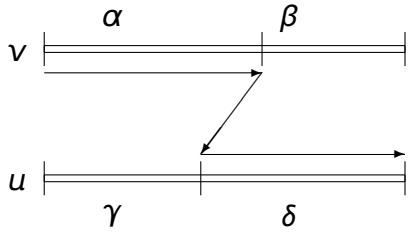

Figure 2.1: A scheme for crossing over

two new strings $\alpha\delta$ and $\gamma\beta$, where the arrows run along the first one of them. Most prominently the so-called Splicing Systems were motivated by this [37].

Duplication concludes our list. To this operation we will dedicate its own section, because from it duplication languages were derived; and these are the chronological origin of all the work presented here.

2.1.2 Duplication

One of the most frequent mutations among the genome rearrangements is gene duplication or the duplication of a segment of a chromosome [70]. This is the DNA operation, which has motivated our research. The definition of gene duplication as given in the MedTerms Online Medical Dictionary [69] is the following:

> Gene duplication: An extra copy of a gene. Gene duplication is a key mechanism in evolution. Once a gene is duplicated, the identical genes can undergo changes and diverge to create two different genes.
> . . .
> Duplications typically arise from an event termed unequal crossing-over (recombination) that occurs between misaligned homologous chromosomes during meiosis (germ cell formation). The chance of this event happening is a function of the degree of sharing of repetitive elements between two chromosomes. The recombination products of such an event are a duplication at the site of the exchange and a reciprocal deletion.

In the process of duplication, a stretch of DNA is duplicated to produce two adjacent copies, resulting in a tandem repeat. An interesting property of tandem repeats is that they make it possible to do "phylogenetic analysis" on a single sequence. This means, for example, to try to find the most likely duplication history, which then provides one with knowledge about possible earlier version of the gene.

Several mathematical models have been proposed for the production of tandem repeats including replication, slippage and unequal crossing over [59, 94, 83]. These models have been supported by biological studies [88, 93]. Modeling and simulation by Charlesworth et al. [18] suggest that very low recombination rates can result in very large numbers of copies and higher order repeats.

We now illustrate a possibility for obtaining tandem repeats via crossing over as was depicted in Figure 2.1. If the two strings involved are the same, then we have the scenario of Figure 2.2. Following the arrows we read the word $uvvw$ obtained from the orig-

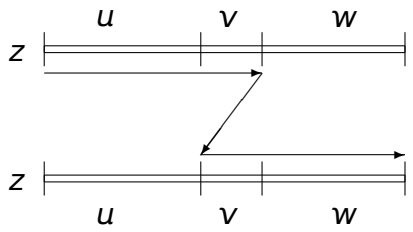

Figure 2.2: A scheme for duplication

inal uvw, which was cut at the beginning of v in one case and just after v in the other case.

In Formal Language Theory, this behaviour first inspired so-called duplication grammars [66], [72]: one starts with a given finite set of strings and produces new strings by copying specified substrings to certain places in a string, according to a finite set of duplication rules. This mechanism is studied from the generative power point of view. Also the context-free versions of duplication grammars are considered. Context-free duplication grammars formalize the hypothesis that duplications appear more or less at random within the genome in the course of its evolution.

Then Dassow et al. introduced languages generated from a word by iterated application of the duplication operation in the form of rewriting rules $u \rightarrow uu$ acting on arbitrary factors [26]. This line of research was further followed by Wang [92], and later also the restriction of the duplicated factors' length to a maximum or to one fixed length have been investigated [58], [56], [51], [57]. The main focus in all this work has been on determining whether the languages generated are regular or not.

2.1.3 Idempotency Relations and Languages

From an algebraic point of view, the basic feature underlying duplication is the idempotency $u \equiv u^2$, however read only from left to right. The first and second power on the left and right hand side respectively are motivated by the duplications observed in DNA strands. However, from a purely mathematical point of view there is no reason to restrict our attention only to this special case. Starting out from this thought, we will investigate the languages generated from one word by iterated application of generalized idempotency rules

$u^m \equiv u^n$ for arbitrary integers m and n; a rule here is the interpretation of $u^m \equiv u^n$ as a string-rewriting rule $u^m \to u^n$.

Following the spirit of the definition of duplication languages, we now proceed to define idempotency relations $\bowtie_m^n$, which rewrite repetitive factors of order m to factors of order n. Then the languages obtained by iterated application of these relations to a single word are introduced.

Definition 2.1.1. For an alphabet Σ the relation $\bowtie_m^n$ over $\Sigma^* \times \Sigma^*$ is defined for two natural numbers m and n as

$$u \bowtie_m^n v :\Leftrightarrow \exists z[z \in \Sigma^+ \wedge u = u_1 z^m u_2 \wedge v = u_1 z^n u_2].$$

With $(\bowtie_m^n)^*$ we denote the relation's reflexive and transitive closure and define the language it generates from a given word w as

$$w^{\bowtie_m^n} := \{u : w(\bowtie_m^n)^* u\}.$$

If the factor whose number of occurrences is changed is bounded in length or required to have a certain length k, then the corresponding relations are denoted by ${}^{\leq k}\bowtie_m^n$ and ${}^{=k}\bowtie_m^n$, formally defined as

$$u {}^{\leq k}\bowtie_m^n v :\Leftrightarrow \exists z[z \in \Sigma^+ \wedge u = u_1 z^m u_2 \wedge v = u_1 z^n u_2 \wedge |z| \leq k] \text{ and}$$

$$u {}^{=k}\bowtie_m^n v :\Leftrightarrow \exists z[z \in \Sigma^+ \wedge u = u_1 z^m u_2 \wedge v = u_1 z^n u_2 \wedge |z| = k].$$

We denote the languages generated by $w^{{}^{\leq k}\bowtie_m^n}$ and $w^{{}^{=k}\bowtie_m^n}$.

It is worth pointing out that k bounds the length of the factor z in the definition of bounded idempotency relations, and not the length of the rule application site z^m. The advantage of defining things in this fashion is that every combination of parameters results in a distinct relation.

Another point worth noting is that we do not define $\bowtie_m^n$ to be the relation $\{(z^m, z^n) : z \in \Sigma^+\}$. When we use both relations as string-rewriting systems, their rewrite relations actually turn out to be the same, namely $\bowtie_m^n$ itself. By our choice of the definition, the rewriting system and its rewrite relation coincide. Thus we can talk about properties of both of them, for example confluence and derivability, while using only one relation. However, in proofs and informal discussions we will often adopt the point of view that we apply a rule (z^m, z^n) rather than $(uz^m v, uz^n v)$, because only in the part consisting of z actual changes occur. While all results and argumentations hold for both versions of the definition, the reader might want to be aware

of this technicality.

A few simple examples shall give a first taste of how these definitions work. We will not prove their correctness here, though – this might be a nice exercise to become familiar with the way the idempotency rules in question work.

Example 2.1.2. Over two letters, duplications can generate just about any factor in any place as the example $(aba)^{\bowtie_1^2} = a\{a,b\}^*b\{a,b\}^*a$ shows. In the case of $(abcbcbab)^{=2}\bowtie_2^4 = a(bcbc)^+bab$ the rules can be applied only on square factors, and in $abcbcbab$ there are only two, which overlap and are even conjugates; thus only one of them needs to be considered. The language generated consists simply of the words reached by iterated catenation of this factor.

For length-reducing rules the languages generated are, of course, finite, like in the case of $(abcbabcbc)^{\bowtie_2^1} = \{abc, abcbc, abcbabc, abcbabcbc\}$; here in a first step either the prefix $(abcb)^2$ or the suffix $(bc)^2$ can be reduced, only the former case results in a word with another square, which can be reduced to abc. This example already shows that one word can in general be reduced to more than one normal form, i.e. the reduction is not converging towards a unique endpoint.

Already these few examples show that the languages generated by idempotency relations are very strongly connected to repetitions in their words. Depending on the parameters m and n, these repetitions are introduced, expanded, shortened or deleted by the rules. This connection explains, why the results from the field of avoidability presented in Section 1.1 build such an important foundation for our investigations.

Of course, our definition provides us with an infinite class of relations. But in the context of our investigations, big classes of such relations exhibit similar behaviour most of the time. We now present a first, rough classification according to the differences in nature of idempotency relations $\bowtie_m^n$ for different values of the parameters m and n. An intuitive characterization of the nature of the relations described is given with each class.

- $\bowtie_0^1$ is the insertion of arbitrary words at arbitrary places.
- $\bowtie_0^n$ for $n \geq 2$ is the insertion of words with some internal structure at arbitrary places.
- $\bowtie_1^2$ is the duplication of arbitrary factors of a word.

- $\bowtie_1^n$ for $n \geq 3$ is the replacement of arbitrary factors of a word by higher powers of these factors.
- $\bowtie_m^n$ for $2 \leq m < n$ increases the powers of factors already occurring in powers of two or higher; here rules can be applied only at very restricted sites.
- $\bowtie_m^n$ for $m \geq n$ do not increase the length of the underlying word and therefore result always in finite languages.

We will see that most of the results will treat not one single relation but one or more of these classes.

Before proceeding to present the results of our research, we will place the object of our work within Formal Language Theory and related fields of investigation.

2.2 Idempotencies and Related Languages

In presenting the background our investigations are founded on, we start out with the Burnside Problem, in which idempotencies play a central role. While this problem still deals with groups in general, the non-counting classes already represent a pure formal language problem. After these we mention stuttering language, which are almost equal to some of our idempotency languages. Finally we will present some results, which actually already treat idempotency languages, just under different names; these are duplication languages, languages generated by copy systems, and insertion and deletion closures.

2.2.1 The Burnside Problem

Idempotencies have already received a great deal of interest through a problem stated by Burnside in 1902 [16]: Is every group, which satisfies the identity $x^r = 1$ and has a finite set of generators, finite? To understand this questions, we take a short excursion into algebra. A group is a structure $[A, \circ]$ consisting of a set A together with a binary operation $\circ : A \times A \to A$ over this set; this operation we will call multiplication. It is associative, and there is a neutral element 1 which fulfills $1 \circ x = x \circ 1 = 1$ for all $x \in A$. Further, for every $x \in A$ there exists an inverse element $y \in A$ such that $x \circ y = 1$. Thus a group satisfying $x^r = 1$ is one where the $(r-1)$-fold multiplication of any element with itself produces the neutral element.

A set B of generators of a group is a subset of A such that any element in A can be obtained by a finite number of multiplications of elements from B. As an example, all words can be generated by catenation of elements of the alphabet; note, however, that a set of words together with catenation does not form a group but only a monoid, because no inverses exist other than for the empty word. With this, all the concepts appearing in the statement of the Burnside problem are explained.

Burnside himself already gave a positive answer for $r \in \{1, 2, 3\}$. Since then many cases have been solved, others remain open. The first negative result was stated by Adian and Novikov in 1968 for all $r \geq 4381$ [2]. Later this was improved by Adian to all odd $r \geq 665$, for which the group generated is infinite [1]. In the 1980s and 1990s interest in the topic flared up again, a nice overview of the history and the results obtained in that period, when the problem was actually extended to semigroups, has been given by Dershowitz [28]. His exposition also includes the non-counting classes, which are the subject of the next section.

2.2.2 Non-Counting Classes

Another problem that received a great deal of interest is the regularity of non-counting classes, which constitutes one of the most famous problems concerning regular languages and has in part been open for over 30 years. A nice overview was published by Brzozowski [14], after whom the problem was also named Brzozowski's Problem.

Recall from Section 1.2.3 that a language L is called non-counting, iff there is an integer $i \geq 0$ such that for every $v \in \Sigma^+$ and $u, z \in \Sigma^*$, we have $uv^iz \in L$ iff $uv^{i+1}z \in L$. Derived from this was the question whether every equivalence class of the smallest congruence on Σ^* satisfying $v^i \sim v^{i+1}$ is regular.

An important result on the topic is the Theorem of Green and Rees [35], which treats the case where $i = 1$. It states that the relations $w \sim ww$ for a finite alphabet always have a finite number of equivalence classes. Stated in terms of our idempotency relations, this theorem has the following form.

Theorem 2.2.1. *The relation* $\bowtie_1^2 \cup \bowtie_2^1$ *over a finite alphabet has a finite number of equivalence classes.*

Thus there is a finite set of words W such that any word in Σ^+ can be reached from a word in W by duplicating and unduplicating factors. In the form presented in Lothaire's book on combinatorics of

words [61], the Theorem of Green and Rees even gives the number of equivalence classes as a function of the size of the alphabet. The minimal sizes of these sets increase very rapidly for bigger alphabets:

Alphabet size:	0	1	2	3	4	...
Minimal size of W:	1	2	7	160	332381	...

This is contrasted by the fact that in general every square-free word defines a separate equivalence class for $w^m \equiv w^n$ where $m, n \geq 2$. Thus over at least three letters these relations have an infinite number of equivalence classes.

Later, de Luca and Varricchio [64] proved that for all $i \geq 5$ the relation corresponding to $v^i \equiv v^{i+1}$ has a finite number of equivalence classes. This leaves open only the cases 2, 3, and 4.

At about the same time, the problem was extended from $v^i \sim v^{i+1}$ to $v^m \sim v^{m+n}$ under the name of *free Burnside semigroups*, which are already very similar to our idempotency languages. A survey of the results obtained in this line of research until 2001 has been published by do Lago and Simon [29].

2.2.3 Stuttering Languages

Another field, where we find notions related to our idempotency languages is concurrency theory, namely where *linear temporal logic* is used to specify concurrent programs. Originally, here one deals with the repetition of letters in a word, and with languages containing any word pumped up by repeating letters from another word contained in them; these describe processes, which differ in at most the number of times a state may adjacently repeat. The word *cccaaab* would be such a pumped version of *cab*. This is of interest in linear temporal logic, because certain classes of formulas cannot distinguish between words equivalent in this sense.

In natural (spoken) language this phenomenon is known as stuttering. Therefore the name of *stutter*-closure of a language is used, for example, by Peled et al. [77], who study the closure of ω-languages under this operation. In our notation the upward stutter-closure of a word w would be $w^{=1\bowtie_1^2}$.

But not only this rather simple case plays a role. Kucera and Strejcek generalize letter-stuttering to subword-stuttering, where factors can be repeated to an arbitrary degree [49] – they use the term subword for what we call factor. What they use to distinguish the expressiveness of different types of formulas would be subsets of the

languages $w^{\bowtie_m^{m+1}}$ in our context. Since the languages are only used to obtain results very different in nature to our interests, we will not go into any more detail on this topic.

2.2.4 Known Results About Special Cases

As mentioned already in Section 2.1, the most intensively investigated case of idempotency-generated languages so far seems to be the duplication closure, i.e. the case of languages generated by rules $u \to u^2$. First off we present two regular cases. The first one stems from the initial article, where duplication languages were introduced.

Proposition 2.2.2 ([26]). *For every word $w \in \{a, b\}^*$ and the language $w^{\bowtie_1^2}$ is regular.*

Also, uniformly bounding the length of duplications results in regular languages, independently of the size of the alphabet.

Proposition 2.2.3 ([56]). *For every word w and integer $k \geq 0$ the language $w^{=k\bowtie_1^2}$ is regular.*

Over an alphabet of more than two letters we can get beyond regularity in the general and even in most of the length-bounded cases.

Proposition 2.2.4 ([58]). *For every integer $k \geq 4$ the language $(abc)^{\leq k\bowtie_1^2}$ is not regular.*

These cases of non-regularity were shown by refinements in the proof techniques used for obtaining the chronologically first result of this kind.

Proposition 2.2.5 ([92]). *The language $(abc)^{\bowtie_1^2}$ is not regular.*

These results raise the question about an upper bound for the complexity of the languages generated by bounded and general duplication. In the bounded case, context-freeness of the languages generated has been proved; in the general case it remains an open problem.

Proposition 2.2.6 ([58]). *For every every word w and integer $k \geq 0$ the language $w^{\leq k\bowtie_1^2}$ is context-free.*

It must be mentioned here that some of these results were already obtained earlier in investigations dealing with so called copy systems. Obviously the work on duplication has so far been done

without any knowledge of this field. These copy systems are actually defined in exactly the same way as our idempotency languages for $\bowtie_1^2$, only the symbol for the relation differs. Ehrenfeucht and Rozenberg wrote the initial article on copy systems [32] and proved a result implying Proposition 2.2.5. In a following article [13], Bovet and Varricchio did the same for Proposition 2.2.2.

Another special case of idempotency languages is that of arbitrary insertion or deletion of factors, which correspond to the relations $\bowtie_0^1$ and $\bowtie_1^0$ respectively. These have been investigated under the names of 1-*insertion* and *deletion*. A compilation of the results obtained can be found in the book by Ito [45]. There, n-insertion of a word u into a word v for a positive integer n is defined as

$$u \triangleright^{[n]} v := \{v_1 u_1 v_2 u_2 \dots v_n u_n v_{n+1} : u = u_1 u_2 \dots u_n \wedge v = v_1 v_2 \dots v_n v_{n+1}\}.$$

This is extended to languages U and V in the following way:

$$U \triangleright^{[n]} V := \bigcup_{u \in U, v \in V} u \triangleright^{[n]} v.$$

Then it is proved that for regular U and V also $U \triangleright^{[n]} V$ is always regular. Since obviously $\Sigma^* \triangleright^{[1]} \{w\} = w^{\bowtie_0^1}$ we obtain the following result.

Proposition 2.2.7. *For every every word w the language $w^{\bowtie_0^1}$ is regular.*

The deletion of one language from another is defined via the deletion of words $u \longrightarrow v := \{u_1 u_2 : u = u_1 v u_2\}$ such that $U \longrightarrow V := \bigcup_{u \in U, v \in V} u \longrightarrow v$. From the result that the deletion of a regular language from a regular language is again regular we can derive the following result in our context.

Proposition 2.2.8. *For every regular language L the language $\{u : w \bowtie_1^0 u \wedge w \in L$ is regular.*

This does not imply directly the regularity of $w^{\bowtie_1^0}$, but will be useful later on in its proof.

Finally, we also want to mention that in the field of DNA computing similar mechanisms have been investigated under the name of Insertion-Deletion system [76]. Using only insertion or only deletion also here amounts to applying an idempotency rule. However, while some variants without any deletion operations were considered, always context-sensitive insertion has been in the focus of attention. Therefore it seems that all existing results cannot help in our context.

2.3 General Observations

It has already been stated that one of our main objectives will be finding out how complex languages generated by idempotency relations are with respect to the classes of the Chomsky Hierarchy and related language classes. We start by giving a very general upper bound for this complexity, which applies in all the length-bounded cases.

Proposition 2.3.1. *For all integers $k, m, n \geq 0$, every word w, and a condition $c \in \{\leq k, = k\}$ the language $w^{c}\bowtie_{m}^{n}$ is always growing context-sensitive.*

Proof. For $m \geq n$ all languages are finite. The other cases are proven via the McNaughton characterization of languages. Note that all of the relations are strictly length-increasing for $n > m$. Therefore their inverse relations are strictly length-decreasing. Take any such relation as a string-rewriting system and add the rule (w, Y) for some symbol Y that is not in the alphabet of w. With empty strings t_1 and t_2 from the definition of McNaughton languages this system obviously accepts $w^{c}\bowtie_{m}^{n}$. Thus this language is in $\mathtt{lr\text{-}McNL}$ which is equal to the class of growing context-sensitive languages by Theorem 1.4.3. □

Since the class of growing context-sensitive languages is not so well-known, we mention here a few facts about them, which have been established. In contrast to the case of general context-sensitive languages, the membership problem is decidable in polynomial time for this class [23]. Further, it is closed under union, catenation, iteration, intersection with regular languages, λ-free and inverse homomorphisms; thus the growing context-sensitive languages form an abstract family of languages [15].

The question is, of course, how tight this upper bound is. In many cases the languages generated are much simpler, namely regular. However, we will see that there are also cases, where it is unknown whether they are context-free or not. There the upper bound for their complexity provided here is actually the best one known.

We also want to state a relation with a class of languages mentioned in Section 2.2.2, the non-counting languages. Their definition exhibits some obvious parallels to that of the relation $\bowtie_{m}^{m+1}$. Clearly m is such a constant that $xy^{i}z \in w^{\bowtie_{m}^{m+1}}$ iff $xy^{i+1}z \in w^{\bowtie_{m}^{m+1}}$. This allows us to directly conclude the following.

Proposition 2.3.2. *For every $m \geq 0$ and every word w the language $w^{\bowtie_{m}^{m+1}}$ is non-counting.*

2.4 Uniformly Bounded Idempotency

The first variant of idempotency-generated languages we will deal with is the one where the idempotencies are restricted most: all words defining idempotency rules must have the same length. This implies serious limitations for the languages generated; for example, their words can only be of certain lengths: the language $(ababa)^{=2\bowtie_2^5} = ababa((ba)^3)^*$, for example, consists only of words of lengths $5 + 6i$ for integers i. Therefore it does not come as a surprise that we will find confluence and regularity in the majority of cases.

2.4.1 Confluence

Before we actually investigate confluence, we will now state a useful property of uniformly bounded idempotency languages. It simplifies the construction of regular expressions for such a language and thus will be used implicitly further down.

Lemma 2.4.1. *Let $k, m, n > 0$ with $n \geq m$ and let the word $w \in \Sigma^*$ have period k. Then $w^{=k\bowtie_m^n} = w[1 \dots |w| - k](w[|w| - k + 1 \dots |w|]^{n-m})^+$.*

Proof. We prove the claim by induction on the number of rewrite rules that have been applied to obtain a word in $w^{=k\bowtie_m^n}$. Clearly the induction basis $w \in w[1 \dots |w| - k](w[|w| - k + 1 \dots |w|]^{n-m})^+$ holds. So let $w_1 \,{}^{=k}\!\bowtie_m^n\, w_2$ with $w_1 \in w[1 \dots |w| - k](w[|w| - k + 1 \dots |w|]^{n-m})^+$. Then w_2 can be obtained from w_1 by application of one idempotency rule on a factor v^m of w_1 with $|v| = k$. So v has period k. Therefore the period k of the word w_1 is preserved, and of course the last k letters of w_1 also remain unchanged. Thus we have $w_2 \in w[1 \dots |w| - k](w[|w| - k + 1 \dots |w|]^{n-m})^+$. Together with trivial length considerations for the exponent $(n - m)$ this suffices to prove the claim. □

Now we will see that all length-increasing uniformly bounded idempotency relations are confluent.

Lemma 2.4.2. *For $k, m, n \geq 0$ with $n \geq m$ the relation $\,{}^{=k}\!\bowtie_m^n$ is confluent.*

Proof. It is known that the diamond property implies confluence [6]. Therefore it suffices to show that this property $w_1 \leftarrow u \rightarrow w_2 \Rightarrow$

$\exists v(w_1 \to v \leftarrow w_2)$ holds for the relation ${}^{=k}\bowtie^{n}_{m}$. So let two words w_1 and w_2 be direct successors of another word u.

If the factors in u, where the rules are applied, do not overlap, then obviously in both cases the respectively other rule can be applied afterwards and one arrives at a common v. So let two application sites r^m and s^m overlap in u. Without restriction of generality let r^m occur first from the left, and call u' the factor from the start of r^m till the end of s^m such that $u = u_1 u' u_2$ for some $u_1, u_2 \in \Sigma^*$.

Now we can interpret the application of $r^m \to r^n$ as the insertion of r^{n-m} just in front of u'; equally $s^m \to s^n$ amounts to the insertion of s^{n-m} just after u'. Since application of these rules leaves u' unchanged, the two derivations

$$u_1 u' u_2 \to u_1 r^{n-m} u' u_2 \to u_1 r^{n-m} u' s^{n-m} u_2$$

and

$$u_1 u' u_2 \to u_1 u' s^{n-m} u_2 \to u_1 r^{n-m} u' s^{n-m} u_2$$

are possible, and the fact that they result in the same word concludes our proof. □

So all the length-increasing variants are confluent. For length-reducing rules, however, this is true only in some cases.

Lemma 2.4.3. *For $k \geq 2$ the relation ${}^{=k}\bowtie^{0}_{1}$ is not confluent.*

Proof. Let w be a word of length $k+1$. Then the parameters of the relation allow the application of a rewrite rule exactly on two sites: w's prefix and suffix of length k; these will leave the last, respectively the first letter of w as irreducible remainder, and these are in general not equal. □

Lemma 2.4.4. *For $k \geq 2$, $m > n$, and $n \geq 1$ the relation ${}^{=k}\bowtie^{n}_{m}$ is confluent.*

Proof. As in the proof of Lemma 2.4.2 it suffices to show that the diamond property holds, i.e. $w_1 \leftarrow u \to w_2 \Rightarrow \exists v(w_1 \to v \leftarrow w_2)$ for the relation ${}^{=k}\bowtie^{n}_{m}$.

Since $m > n$, rewrite rules reduce repetitive factors to ones of lower repetitiveness but at least one copy of the repeated word of length k remains, because $n \geq 1$. Therefore the diamond property holds obviously, if the application sites of two rewrite rules do not overlap by more than k symbols.

If, on the other hand, there are two powers of order m overlapping in more than k symbols, then the entire sequence has period k, and

thus the application of either rule results in the same word, thus already $w_1 = w_2$. □

Now we are able to fully characterize the conditions under which uniformly bounded idempotency relations are confluent. Lemmata 2.4.2, 2.4.3, and 2.4.4 leave open only the cases where $k = 1$ and $k = 0$. But for these cases confluence is obvious for any m and n.

Proposition 2.4.5. *The relation* ${}^{=k}\bowtie_m^n$ *is confluent except for the case where* $k \geq 2$, $m = 1$, *and* $n = 0$.

2.4.2 Regularity

If we deal with words over an alphabet of only one letter, then, as one might expect, the strict restriction to uniform length of the rules results in the languages generated being rather simple, namely ultimately periodic and therefore regular. This result is implied by the later one on two-letters; we still prove it explicitly, because the proof is easier in this case and it provides us with a concrete expression for the language generated.

Proposition 2.4.6. *Over a one-letter alphabet* $\{a\}$ *for every nonempty word* w *and integers* $k, m, n \geq 0$ *the language* $w^{=k}\bowtie_m^n$ *is regular.*

Proof. If $m \geq n$, then the language generated is finite and thus also regular. For $m < n$ there exists only one possible rewrite rule, namely $(a^k)^m \to (a^k)^n$, and with every application exactly $k \cdot (n - m)$ copies of the letter a are inserted. The place of application does not matter since catenation is commutative over just one letter. Thus $w^{=k}\bowtie_m^n = w(a^{k\cdot(n-m)})^*$. □

While in most cases also for bigger alphabets the languages generated remain regular, the proofs of this will be somewhat more involved. For the rest of this section we will assume an alphabet Σ containing at least two letters. It is still rather easy to see that insertion of arbitrary words generates only regular languages, see also Proposition 2.2.7, where, however, unrestricted insertion is treated.

Proposition 2.4.7. *For every word* w *and an integer* $k \geq 0$ *the language* $w^{=k}\bowtie_0^1$ *is regular.*

Proof. In this case, at any point arbitrary words from the set Σ^k can be inserted into the original word. Thus the language generated is described by the regular expression $\phi := (\Sigma^k)^* w[1](\Sigma^k)^* w[2](\Sigma^k)^* \ldots (\Sigma^k)^* w[|w|](\Sigma^k)^*$.

The only consideration necessary to see this is the following: let some word $u = u_1u_2$ be inserted, and later a second one v between the two factors u_1 and u_2; choose the factorization v_1v_2 of v for which $|u_1v_1| = |v_2u_2| = k$. Then the same word would have been reached by first inserting u_1v_1, and then v_2u_2 just behind it. Thus insertions of one factor inside another need not be considered and catenation of factors from Σ^n in the way described in ϕ suffices to generate the entire language $w^{=k\bowtie_0^1}$. □

When n becomes greater than 1, instead of arbitrary words we insert words, which already have some internal structure, namely they are squares, cubes etc., i.e. they are always non-primitive. Then the insertion cannot be replaced by simple catenation and we obtain also non-regular languages.

Example 2.4.8. Let $L \subset \{a, b\}^*$ be the language generated from λ by insertion of squares of words of length 2, i.e. $L = \lambda^{=2\bowtie_0^2}$. Then we show that $L \cap (bbaa)^+(aabb)^+ = \{(bbaa)^n(aabb)^n : n \geq 0\}$, and this language is clearly not regular.

Every word in $\{(bbaa)^n(aabb)^n : n \geq 0\}$ can be generated from λ by first putting b^4, then a^4 in its center, and so on.

On the other hand, every word in $(bbaa)^+(aabb)^+$ and therefore also every word in $L \cap (bbaa)^+(aabb)^+$ contains only one square of a word of length 2, namely the a^4 in the center. Removing it, b^4 forms a unique such square. Thus a reduction to λ is possible only if the numbers of $bbaa$ and $aabb$ correspond, and this shows that all words in this intersection must belong to the set $\{(bbaa)^n(aabb)^n : n \geq 0\}$.

This example does not represent some special case, rather non-regularity always holds over an alphabet of at least two letters, more precisely speaking the languages generated are not even linear.

Proposition 2.4.9. *For every word w and integers $k \geq 2$, and $n \geq 2$ the language $w^{=k\bowtie_0^n}$ is not linear but context-free.*

Proof. Analogously to the language obtained by intersection in Example 2.4.8 we can always filter out a non-linear component over two letters. So for an arbitrary relation $^{=k}\bowtie_0^n$ let us consider the language

$$\lambda^{=k\bowtie_0^n} \cap (b^2a^{nk-1})^+(ab^{nk-2})^+(ba^{nk-1})^+(ab^{nk-1})^+$$

obtained by intersection of $\lambda^{=k\bowtie_0^n}$ with a regular language. This results in the non-linear

$$L = \{(b^2a^{nk-1})^i(ab^{nk-2})^i(ba^{nk-1})^j(ab^{nk-1})^j : i, j \geq 0\}.$$

The reasoning for seeing this is the same as in Example 2.4.8. Clearly L is a subset of the intersection by derivations

$$\lambda \to b^{nk} \to b^2a^{nk}b^{nk-2} \to b^2a^{nk-1}b^{nk}ab^{nk-2} \to \cdots$$

for the first component, and by an analogous derivation for the second component.

To see that the language obtained by intersection is contained in L, we observe that all words in $(b^2a^{nk-1})^+(ab^{nk-2})^+(ba^{nk-1})^+(ab^{nk-1})^+$ are in the intersection, iff they have λ as a normal form under the relation ${}^{=k}\bowtie_n^0$. For obtaining the normal form of any word in $(b^2a^{nk-1})^+(ab^{nk-2})^+(ba^{nk-1})^+(ab^{nk-1})^+$ the only applicable rule is $a^nk \to \lambda$, which is applicable on two sites. Application at either site creates b^nk there and applying $b^nk \to \lambda$ takes us back into the original language. At no stage any rule transgressing the border between the two first and two last iterations is possible. So the reduction goes independently in both components, and the word can only be reduced to λ if the exponents are as in L.

Now we show the inclusion of languages $w^{=k}\bowtie_0^n$ in CF by sketching the construction of a context-free grammar generating $w^{=k}\bowtie_0^n$ for some word w and non-negative integers k and n. It has only two nonterminals S and T. For the start symbol S, there is the unique rule $(S, Tw[1]Tw[2]T \ldots Tw[|w|]T)$. The rest of rules consists of the set $\{(T, T(x_1Tx_2T \ldots Tx_kT)^n : x_1, x_2, \ldots, x_k \in \Sigma)\}$ and the deleting rule (T, λ). It should be rather obvious that this grammar generates exactly the desired language with the ubiquitous T permitting insertion at any position, while it can be deleted wherever no further insertions occur. □

Proposition 2.4.10. *For every nonempty word w, integers $k, n \geq 0$, and $m \geq 1$ the language $w^{=k}\bowtie_m^n$ is regular.*

Proof. Because different parameters can result in a quite different behaviour of the relations ${}^{=k}\bowtie_m^n$, we distinguish several cases.

Case 1: $m \geq 1$ and $n \leq m$. The relation is length-reducing or the identity, the resulting language is obviously finite and therefore regular.

Case 2: $n = 2, m = 1$. This is the special case of uniformly bounded duplication and is therefore covered by Proposition 2.2.3.

Case 3: $n > 2, m = 1$. The crucial fact to note is that the applications of the idempotency rules can be done strictly from left to right; i.e. it can be done in a way such that at most the last k positions produced in the last step are affected in the following one. To see this it suffices to recall that according to Lemma 2.4.5 the relation is confluent here, and as shown in the lemma's proof it even fulfills the diamond property.

This implies that every word $u \in w^{=k\bowtie^n_m}$ can be constructed by successive applications of idempotency rules in such a way that at any stage it can be factored as rst, where rs is already a prefix of u, s will be replaced by s^n in the next step, and st is a suffix of the original word w. This tells us that for any prefix u' of a word in $w^{=k\bowtie^n_m}$ there exists a word v, which is a suffix of w such that $u'v \in w^{=k\bowtie^n_m}$. It remains to show that this allows us to give a bound for the number of equivalence classes of the syntactic congruence $\sim$ for the language $w^{=k\bowtie^n_m}$.

All words u such that there exists no v such that $uv \in w^{=k\bowtie^n_m}$ constitute one such class C. Now let such a factor v exist, i.e. u is a prefix of a word in the language. As shown above, this word can be constructed from left to right.

This means that there exists a word v fulfilling the above property, which is at the same time a suffix of w except for maybe its first $(n-m)k-1$ letters produced in the last application of an idempotency rule. Of course, there are only finitely many suffixes of w and only finitely many words of length $(n-m)k-1$. As the possible right contexts of all equivalence classes of $\sim$ (except for C) have to contain at least one such suffix, their number is bounded exponentially by the number of suffixes of w and the number $(n-m)k-1$, to be more exact, $|\Sigma|^{|w|+(n-m)k-1}$ is a bound. Therefore the syntactical congruence is of finite index and by Theorem 1.2.5 the language $w^{=k\bowtie^n_m}$ is regular.

Case 4: $n > m, m = 2$. First let us look at the rules $u^2 \to u^n$ as insertions of u^{n-2} between the two original occurrences of u. This illustrates that idempotency rules affecting factors overlapping by no more than k symbols can be looked at independently. Further, note that due to the fixed length of such words u, every border between letters in the original word w can be center of at most one relevant factor uu.

Now we construct the regular expression R from w as follows. Going from left to right, every square uu with $|u| = k$ is replaced by $u(u^{n-2})^*u$. Clearly the language described by R is a subset of $w^{=k\bowtie^n_m}$.

However, two squares of length $2k$ overlapping in more than k letters might allow applications of idempotency rules in ways not described by this expression.

To see that this is not the case, we first notice the fact that two such factors uu and vv overlapping in more than k letters imply that u and v are conjugates, because v is an internal factor of uu. This means that the entire factor of w spanning these two squares has period k. Therefore it does not matter, whether u^{n-2} or v^{n-2} is inserted at the respective place, the result is the same, see also Lemma 2.4.1. Thus this case is described by R, too, and consequently exactly the language $w^{=k}\bowtie_m^n$ is described and therefore regular.

Case 5: $n > m, m > 2$. Essentially the same reasoning as in Case 4 applies.

□

2.5 Bounded Idempotency

Compared to uniformly bounded rules, general length-bounded rules allow many more possibilities, namely to have the application site for one inside the site for another rule. This feature makes the languages generated non-regular in many cases, and also confluence is not always given.

2.5.1 Confluence

In the case of bounded insertion, we can establish confluence rather easily.

Proposition 2.5.1. *For all $k, n \geq 1$ the relation ${}^{\leq k}\bowtie_0^n$ is confluent.*

Proof. Let $u, v \in w^{\leq k}\bowtie_0^n$ for a word w. This means that u and v can both be obtained from w by inserting n-th powers of words of length no greater than k between the letters of w. So, marking the original letters of w by underlining them, we have $u = u_1\underline{w[1]}u_2\underline{w[2]}\dots u_{|w|}\underline{w[|w|]}u_{|w|+1}$ and $v = v_1\underline{w[1]}v_2\underline{w[2]}\dots v_{|w|}\underline{w[|w|]}v_{|w|}$ for some words $u_1, u_2 \dots u_{|w|+1}, v_1, v_2, \dots v_{|w|+1} \in \Sigma^*$. Now clearly

$$u_1v_1\underline{w[1]}u_2v_2\underline{w[2]}\dots u_{|w|}v_{|w|}\underline{w[|w|]}u_{|w|+1}v_{|w|+1} \in u^{\leq k}\bowtie_0^n \cap v^{\leq k}\bowtie_0^n,$$

which proves the confluence of ${}^{\leq k}\bowtie_0^n$. □

For ${}^{\leq k}\bowtie_1^n$ confluence depends on the length bound, as the following two propositions will show.

Proposition 2.5.2. *For all $k < 3$ and $n \geq 1$ the relation ${}^{\leq k}\bowtie_1^n$ is confluent.*

Proof. We show that the diamond property holds, i.e. $w_1 \leftarrow u \rightarrow w_2 \Rightarrow \exists v(w_1 \rightarrow v \leftarrow w_2)$. For $k < 2$ this is obvious. The same is true for $k = 2$ if the two application sites of the rules do not overlap. The few possible cases for $k = 2$ can now be checked in an exhaustive manner to have the diamond property.

For $n \geq 2$ the derivations $abc \rightarrow (ab)^n c \rightarrow (ab)^n c(bc)^{n-1} \leftarrow a(bc)^{n-1} \leftarrow abc$ treats the case of two rules with left sides of length two. If they are of length one and two, then and $ab \rightarrow ab^n \rightarrow (ab)^n b^{n-1} \leftarrow (ab)^n \leftarrow ab$ proves the diamond property. Of course, if not all of the letters involved are different, then things become even easier. □

The argumentation shows that, informally speaking, for non-confluence it has to be possible to have the application site of one rule properly inside the one of the other. The shortest possible lengths for this are one and three, and these already suffice, however only over at least three letters.

Proposition 2.5.3. *For all $k \geq 3$ and $n \geq 2$ the relation ${}^{\leq k}\bowtie_1^n$ is not confluent over an alphabet of three or more letters.*

Proof. From the word $ab^{k-2}c$ one can obtain in one step $u = ab^{k+n-3}c$ and also $v = (ab^{k-2}c)^n$. Notice that v contains an occurrence of a after one of c, and thus all words obtained by application of further rules will do so.

At the same time in u the unique occurrences of a and c are separated by at least $k - 1$ letters b. Thus no application of a rule from ${}^{\leq k}\bowtie_1^n$ can include as well a as c. Since this central block of b is conserved, no word with an a after a c can be reached. Thus $u^{\leq k}\bowtie_1^n \cap v^{\leq k}\bowtie_1^n = \emptyset$, which proves our claim. □

Over a smaller alphabet, also the duplication corresponding to the parameters from Proposition 2.5.3 is still confluent.

Proposition 2.5.4. *Over a two-letter alphabet, the relation ${}^{\leq k}\bowtie_1^2$ is confluent for all $k \geq 1$.*

Proof. The cases where $k = 1$ are obvious. So let us suppose that we have $u \overset{*}{\leftarrow} w \overset{*}{\rightarrow} v$. Notice that all words in $w^{\leq k}\bowtie_1^2$ start and end with the same letters, let them be a and b respectively. Then a characteristic feature of every such word is its number of changes from a to b. Let this number be i for u and j for v. Unless they are equal, without restriction of generality let i be the greater number. We now start from the word v and select any occurrence of ab in it. This we duplicate $i - j$ times, the resulting word v' now has i changes from a to b, just as u.

In a next step we look at u and v' and compare the length of the initial blocks of a. In the shorter one we duplicate the initial a so often, that the block of a becomes as long as the other one. Then the same is done for the first block of b and so on for all blocks. Clearly the resulting word is in $u^{\leq k}\bowtie_1^2 \cap v^{\leq k}\bowtie_1^2$, which proves the confluence of ${}^{\leq k}\bowtie_1^2$. Note that we have used only rules, where $k = 1$ or $k = 2$. □

To show the confluence of ${}^{\leq k}\bowtie_1^n$ for greater n, the construction method used for $n = 2$ cannot be applied. Unlike in the construction in the proof of Proposition 2.5.4, two blocks of one letter cannot be made to have the same length in general as the following lemma shows.

Lemma 2.5.5. *Let $w \in \{a, b\}^*$, $k \geq 1$ and $n > 2$. For all words $u \in w^{\leq k}\bowtie_1^n$ the number of changes from a to b, the number of changes from b to a, and the numbers $|u|_a$ and $|u|_b$ are constant modulo $(n - 1)$.*

Proof. If the site of a rule application contains no change from a to b, then the number of such changes for the entire word stays the same. If, on the other hand, it contains i changes, they will be replaced by $n \cdot i$ ones, the number increases by $(n - 1) \cdot i$. Similarly, a rule whose left side contains i letters a replaces them by $n \cdot i$ new ones, also here the number increases by $(n - 1) \cdot i$. □

Nonetheless we conjecture that these relations are still confluent, but some different reasoning will be necessary.

Proposition 2.5.6. *For all $k \geq 3$, $m \geq 2$, $k > m$ and $n > m$ the relation ${}^{\leq k}\bowtie_m^n$ is not confluent.*

Proof. We start from the word $(a^m b)^m$. The entire word is the left side of a rule resulting in $(a^m b)^n$, which has more than m letters b. On the other hand the rule $a^m \to a^n$ can be applied to any of the blocks a^m; if this is done to any of these blocks except the first one, it is quite

clear that after this it will only be possible to apply rules producing more a. Thus the number of letters b will always remain lower than n, which suffices to prove our claim. □

2.5.2 Regularity

As for confluence, also the question of regularity of the languages generated is much more interesting with just a general length bound compared to the uniform one. The one-letter case remains simple, though.

Proposition 2.5.7. *Over a one-letter alphabet $\{a\}$ for every nonempty word w and integers $k, m, n \geq 0$ the language $w^{\leq k}\bowtie_m^n$ is regular.*

Proof. With a reasoning very much along the lines of the proof of Proposition 2.4.6 we can see that for $m < n$

$$w^{\leq k}\bowtie_m^n = w(a^{(n-m)})^*(a^{2\cdot(n-m)})^* \cdots (a^{k\cdot(n-m)})^*.$$

□

For a greater alphabet the language generated is also regular, if we look at the insertion of words with no inner structure u^n for $n \geq 2$.

Proposition 2.5.8. *For every word w and integer $k \geq 0$ the language $w^{\leq k}\bowtie_0^1$ is regular, and further $w^{\leq k}\bowtie_0^1 = w^{\leq 1}\bowtie_0^1$ for $k \geq 1$.*

Proof. The case of $k = 0$ is trivial. For greater k always insertions of length one, i.e. of single letters are possible at any position, and between the letters of the original word any word can be generated. Thus any word in $w^{\leq k}\bowtie_0^1$ can be generated by insertions of length only one and the resulting language consists exactly of all the words having w as a scattered subword — a condition that can easily be checked by a finite automaton. □

Along quite similar lines as originally used by Wang for unbounded duplication [92] we will now prove that for many relations the languages generated are not regular.

Proposition 2.5.9. *Over an alphabet of three letters, for every word w and integers $k \geq 1$ and $n \geq 2$ the language $w^{\leq k}\bowtie_0^n$ is not regular.*

Proof. We prove that $\lambda^{\leq k}\bowtie_0^n$ is not regular. First we show that for every square-free word u there exists a word v such that $uv \in \lambda^{\leq k}\bowtie_0^n$. It is rather straight-forward to construct uv in a way that produces

one letter of u in every step, while v will consist of all the letters produced, which do not form part of u. We start with λ and first insert $u[1]^n$, then after the first letter of u insert $u[2]^n$ etc. In this v takes up all the letters not needed for u, but which are produced by the rules. By this method we obtain an upper bound on the length of the smallest such v, namely $|v| \leq |u|(n-1)$, because exactly $n-1$ letters of v are produced in every step.

Now we establish a lower bound on the length of words v such that $uv \in \lambda^{\leq k}\bowtie_0^n$. Since u is square-free, every insertion can produce at most $2k-1$ symbols of it, otherwise there would be a square in u. It is also impossible for letters after the $(n-1)$-st position to become part of u later by insertions in front of them: then these would leave a square within u. So still in the optimal case of always producing $2k-1$ letters of u in every step, we have that $|v| \geq \frac{|u|}{2k-1}((n-2)k+1)$.

Summarizing, for every square-free word u there exists a word v such that $uv \in \lambda^{\leq k}\bowtie_0^n$, and for the shortest such v we have $\frac{|u|}{2k-1}((n-2)k+1) \leq |v| \leq |u|(n-1)$, where the lower bound is optimal. Now, over three letters there exists an infinite square-free word. Let $u_1, u_2, u_3 \ldots$ be a sequence of prefixes of such a word with $\frac{|u_{i+1}|}{2k-1}((n-2)k+1) > |u_i|(n-1)$ and let v_i be the shortest word such that $u_i v_i \in \lambda^{\leq k}\bowtie_0^n$ for all $i \geq 1$. Then clearly $u_j v_i \notin \lambda^{\leq k}\bowtie_0^n$ for all $j > i$. This means that the equivalence classes of the u_i in the syntactical congruence of $\lambda^{\leq k}\bowtie_0^n$ are pairwise different, so there is an infinite number of such classes. According to Theorem 1.2.5 the language $\lambda^{\leq k}\bowtie_0^n$ cannot be regular. □

Before we treat the cases, where $m = 1$, we compile some properties of the underlying relations, which will then allow us to prove the non-regularity of several cases.

Lemma 2.5.10. *Over a two-letter alphabet $\{a, b\}$ for every 2^+-free word u starting with ab and every integer $n > 1$ there exists a word v, such that $uv \in (ab)^{\leq 3}\bowtie_1^n$ and $|v| \leq (3(n-1)+2)(|u|-2)$.*

Proof. u being 2^+-free implies that there is no factor xxx for any letter $x \in \Sigma$. Thus the alphabet's containing only two elements guarantees that after at most 2 positions in u letters repeat, i.e. for every position in u its letter is repeated at most three positions later. Thus we can construct a word having u as a prefix in the following way: starting from ab, always one letter more of u is constructed per step. We take the shortest suffix z of the already constructed part of u starting with the next letter needed. On it we apply the rule $z \to z^n$

putting the required letter in the position. As exposed above, the maximum length of z is 3 and thus all rules belong to ${}^{\leq 3}\bowtie_1^n$. This process takes exactly $|u|-2$ steps, and in each one at most $3(n-1)+2$ additional letters are introduced, which proves the length bound on v. □

However, the length of the word v in Lemma 2.5.10, i.e. in some sense the amount of garbage produced during the generation of u, cannot be reduced to arbitrarily small numbers.

Lemma 2.5.11. *Over a two-letter alphabet $\{a,b\}$ for every 2^+-free word u starting with ab and every integer $n \geq 3$ there exists no word v, such that $uv \in (ab)^{{}^{\leq 3}\bowtie_1^n}$ and $|v| \leq \frac{|u|-2}{2k}$.*

Proof. u is obtained from ab by the application of rules $z \to z^n$. Since u is 2^+-free and $n \geq 3$ every such rule must produce at least one additional symbol outside of u, therefore contributing to v. At the same time each rule produces at most $2k$ letters of u such that at least $\frac{|u|-2}{2k}$ rules must be applied. Therefore there are at least $\frac{|u|-2}{2k}$ symbols in v. □

Lemma 2.5.12. *Over a three-letter alphabet $\{a,b,c\}$ for every square-free word u starting with abc and every integer $n > 1$ there exists a word v, such that $uv \in (abc)^{{}^{\leq 4}\bowtie_1^n}$, and for the shortest such word we have $\frac{|u|-3}{7} \leq |v| \leq (4(n-1)+3)(|u|-3)$.*

Proof. uv can be constructed starting from abc in a way very similar to that of the proof of Lemma 2.5.10. Only here between two consecutive occurrences of the same letter in u there can be three other letters, because 3 is the length of the longest square-free word over two letters. Therefore the longest z such that rules $z \to z^n$ are applied is 4 letters long, and that gives us the upper bound on the length of v. The lower bound is obtained in a manner analogous to the proof of Lemma 2.5.11. □

Proposition 2.5.13. *Over a two-letter alphabet for every word w and integers $k, n \geq 3$ the language $w^{{}^{\leq k}\bowtie_1^n}$ is not regular, while $w^{{}^{\leq k}\bowtie_1^2}$ is.*

Proof. The non-regularity of $w^{{}^{\leq k}\bowtie_1^2}$, i.e. for the case of duplication, is already stated in Proposition 2.2.4. For $n \geq 3$ Lemmata 2.5.10 and 2.5.11 show us that for every 2^+-free word u starting with ab and every integer $n \geq 3$ there exists a word v, such that $uv \in (ab)^{{}^{\leq 3}\bowtie_1^n}$ and $\frac{|u|-2}{2k} \leq |v| \leq (3(n-1)+2)(|u|-2)$, i.e. the length of a minimal v is bounded from above and below.

Now we take an infinite 2^+-free word starting with ab and produce a sequence of prefixes $(u_i)_{i\geq 1}$ such that $\frac{|u_{i+1}-2|}{2k} > (|u_i|-2)2k$. Then the v_i from the construction in the proof of Lemma 2.5.10 is such that $u_i v_i \in w^{\leq k}\bowtie_1^n$, while $u_{i+1}v_i \notin w^{\leq k}\bowtie_1^n$ due to length considerations. Therefore all our words u_i are pairwise in different equivalence classes of the syntactical right congruence of $w^{\leq k}\bowtie_1^n$, and by Theorem 1.2.5 the language cannot be regular. □

With more than two letters also the special case of $n = 2$ is not regular any more.

Proposition 2.5.14. *Over a three-letter alphabet for every word w and integers $k \geq 4$ and $n > 1$ the language $w^{\leq k}\bowtie_1^n$ is not regular.*

Proof. A construction analogous to the proof of Proposition 2.5.13 using the length bounds from Lemma 2.5.12 proves this statement. For more details the reader can also consult the proof for the special case of bounded duplication [58]. □

Having found lower bounds for the interesting cases where $m = 1$, we can now also state an upper bound, which determines the exact place of languages $w^{\leq k}\bowtie_m^n$ in the Chomsky Hierarchy, also for $m > 1$. We provide here a proof completely different from the original one, where a complicated PDA was constructed to accept $w^{\leq k}\bowtie_m^n$ [53]. The proof provided here is shorter, more elegant, and easier to understand.

Proposition 2.5.15. *For every word w, and for integers $k, m, n \geq 0$ the language $w^{\leq k}\bowtie_m^n$ is context-free.*

Proof. We will transform words from Σ^+ into a redundant representation, where every letter contains also the information about the $k \cdot m - 1$ following ones. This way rewrite rules from ${}^{\leq k}\bowtie_m^n$ can be simulated by ones with a left side of length only one. Their inverses are monadic. Thus the McNaughton characterization of languages provides us with the context-freeness of the language generated and consequently of $w^{\leq k}\bowtie_m^n$.

First off we define the mapping $\phi : \Sigma^+ \mapsto ((\Sigma \cup \{\square\})^{k\cdot m})^+$ as follows. We delimit with $(\ldots)$ letters from $(\Sigma \cup \{\square\})^{k\cdot m}$ and with $[\ldots]$ factors of the word w as usual. The image of a word u is

$$u \mapsto (w[1 \ldots k\cdot m])(w[2 \ldots k\cdot m+1])\cdots(w[|w|-k\cdot m+1 \ldots |w|])\cdot (w[|w|-k\cdot m+2 \ldots |w|]\square)\cdots(w[|w|]\square^{k\cdot m-1}).$$

Thus every letter contains also the information about the $k \cdot m$ following original ones from the original word u, at the end of the word letters are filled up with the space symbol $\square$. This encoding can be reversed by a letter-to-letter homomorphism h defined as $h(x) := x[1]$ if $x[1] \in \Sigma$, for the other case we select some arbitrary letter a and set $h(x) := a$ if $x[1] = \square$; the latter case will never occur in out context. It is clear that $h(\phi(u)) = u$ for words from Σ^*. Both mappings are extended to languages in the canonical way such that $\phi(L) := \{\phi(u) : u \in L\}$ and $h(L) := \{h(u) : u \in L\}$.

Now we define the string-rewriting system R over the alphabet $(\Sigma \cup \{\square\})^{k \cdot m}$ as follows:

$$R := \{((u^m v), \phi(u^n v')[1 \ldots |\phi(u^n v')| - m \cdot k - 1]) : u \in \Sigma^{\leq k} \wedge v' \in \Sigma^* \wedge v \in v' \cdot \{\square^*\} \wedge |u^m v| = k \cdot m\}.$$

A letter $[u^m v]$ is replaced by the image of $u^n v$ minus the suffix of letters that contain $\square$. This way application of rules from R keeps this space symbol only in the last letters of our words. It should be rather clear that $\phi(w^{\leq k}\bowtie_m^n) = \{u : \phi(w) R^* u\}$ or, in other words $w^{\leq k}\bowtie_m^n = h(\{u : \phi(w) R^* u\})$.

To determine the complexity of the language generated we now consider the string-rewriting system $S := R^{-1} \cup \{(\phi(w), Y)\}$, where Y is a letter not occurring in Y. This systems accepts as a McNaughton language $\phi(w^{\leq k}\bowtie_m^n)$. As it is monadic, the language is context-free by Proposition 1.4.2. Since context-free languages are closed under letter-to-letter homomorphisms, also $w^{\leq k}\bowtie_m^n = h(\phi(w^{\leq k}\bowtie_m^n))$ is context-free. □

The properties of languages generated by bounded idempotency which we stated for the non-regularity proofs earlier also allow us to conclude that in many cases the inclusions $w^{\leq k}\bowtie_1^n \subset w^{\leq k+1}\bowtie_1^n$ are proper. For this, however, we first need to recall the notion of circular pattern avoidance. A word w is said to be *circular square-free*, iff it is square-free and so are all its conjugates. This means that one can arrange the word in a circle with the first letter following the last, and nowhere along the circle there is a square. We explicitly state an immediate consequence of this definition.

Lemma 2.5.16. *For a circular square-free word w the word ww contains no square shorter than ww itself.*

Circular cube-freeness is defined analogously. It is known that over a three letter alphabet there exist circular square-free words of any

length greater than 17, and over two letters there exist circular cube-free words of any given length [22].

Proposition 2.5.17. *For every word w over two letters all inclusions $w^{\leq k}{}_{\bowtie_1^n} \subset w^{\leq k+1}{}_{\bowtie_1^n}$ are proper for $n \geq 3$ and $k \geq 2$.*

Proof. From Lemma 2.5.10 we know that for every 2^+-free word u starting with ab and every integer $n \geq 2$ there exists a word v, such that $uv \in (ab)^{\leq 3}{}_{\bowtie_1^n}$. At some point of w a change from one letter to another must occur. So there we can construct any circular cube-free word. Let us construct such a word u of length $k+1$ for some fixed k. In the next step we can apply here the rule $u \to u^n$.

The resulting factor u^n can also be produced by shorter rules, but Lemma 2.5.10 also shows that there is a lower bound on the number of additional symbols produced in this process. Thus by further applying the rule $u \to u^n$ we can reach a word, where the block of u^+ is so long in relation to the rest of the word, that it is impossible to produce the same word only with rules where the left side is not longer than k, since by a generalization of Lemma 2.5.16 to blocks u^n instead of just u^2 no shorter rule can have been applied anywhere within this block. □

Proposition 2.5.18. *For every word w over three or more letters there exists a k_w such that all inclusions $w^{\leq k}{}_{\bowtie_1^n} \subset w^{\leq k+1}{}_{\bowtie_1^n}$ are proper for $n \geq 2$ and $k \geq k_w$; if w has a factor abc, then $k_w = 18$ will work.*

Proof. k_w will be the maximum of two values. One is the smallest number such that with rules of left sides of this length we can produce a factor of the form abc in w. For example, for the word $aabbbbbbccc$ the value is 9: with one rule we can produce $aabbbbbbcaabbbbbbccc$ and with another one $aabbbbbbc\mathbf{abc}aabbbbbbccc$. The second value is 18, since starting from this length there exist circular square-free words of any given length.

From Lemma 2.5.12 we know that over a three-letter alphabet for every square-free word u starting with abc and every integer $n \geq 2$ there exists a word v, such that $uv \in (abc)^{\leq 4}{}_{\bowtie_1^n}$. Thus also every circular square-free word can be constructed. Starting from lengths of 18, such a word always exists, and starting from k_w we also can suppose that a word in $w^{\leq k}{}_{\bowtie_1^n}$ contains a factor of the form abc. Since Lemma 2.5.12 also provides a lower bound on the length of the additional v, which is produced, the same proof technique as for Proposition 2.5.17 applies. □

2.6 General Idempotency

When dropping all restrictions on the idempotencies, a fundamental difference to the cases treated up to this point is that we have infinitely many rewrite rules, whereas so far, due to the length restrictions, there have been only finitely many. In some simple cases there are finite sets equivalent in generating power, but not in general as already shown by Propositions 2.5.17 and 2.5.18.

So we will first look at another restriction than length bounds, namely at smaller alphabets. For one and two letters, many questions can still be answered. After this we list a number of results for the completely unrestricted cases, most of which carry over more or less directly from previous ones.

2.6.1 The One-Letter-Case

As one might expect, the one-letter case does not hold any surprises. Also without any length bound, relations are always confluent, and languages are always regular.

Proposition 2.6.1. *Over a one-letter alphabet all relations* $\bowtie_m^n$ *are confluent.*

Proof. It is easy to see that the diamond property holds, i.e. $w_1 \leftarrow u \rightarrow w_2 \Rightarrow \exists v(w_1 \rightarrow v \leftarrow w_2)$ for the relation $\bowtie_m^n$, and this implies confluence [12]. Looking at a word as a unary number, applying a rule from $\bowtie_m^n$ amounts always to adding for $n > m$ and to subtracting for $n < m$, and both these operations are associative.

The only problematic case is the subtraction of two numbers, whose sum is greater than the original number. For this, consider a rule $(a^k)^m \rightarrow (a^k)^n$ for some positive integer k. It reduces the number of letters by $(m-n)k$. Thus in every rule application a multiple of $m-n$ is removed. Via the rule $a^m \rightarrow a^n$ we can arrive at the same result in k steps. Any word w is reduced in this way to the irreducible word a^ℓ, where ℓ is the remainder when dividing $|w|$ by $m-n$. Thus a^ℓ is the unique normal form and confluence is given. □

Proposition 2.6.2. *Over a one-letter alphabet* $\{a\}$ *for every nonempty word* w *and integers* $m, n \geq 0$ *the language* $w^{\bowtie_m^n}$ *is regular.*

Proof. We assume that $|w| \geq m$, otherwise no rule can be applied, and $w^{\bowtie_m^n}$ is trivially regular, because it is finite. For $n \leq m$ the language generated is finite and therefore also regular. For $n > m$ we have the rule $a^m \rightarrow a^n$; taken just by itself it generates the language

$w(a^{n-m})^*$ starting from a word w. Applying any other rule $a^{km} \to a^{kn}$ for some $k > 1$ adds $k(n-m)$ letters a, thus the result is already in $w(a^{n-m})^*$. Therefore $w^{\bowtie_m^n} = w(a^{n-m})^*$. □

2.6.2 Confluence over Two Letters

While the step from one letter to two makes things significantly more complicated, things remain more tractable than for the case of still larger alphabets.

Proposition 2.6.3. *Over a two-letter alphabet all relations $\bowtie_0^n$ and $\bowtie_1^n$ are confluent for all n.*

Proof. For $m = 0$ the diamond property holds for $\bowtie_m^n$. Rule application amounts to the insertion of a factor u^n, and inserting two such factors can obviously be done independently of each other.

For $m = 1$ two rule applications are obviously independent, if their sites do not overlap. In the case of an overlap such that not one site is completely inside the other, it suffices to realize that expansions $u \to u^n$ preserve prefixes and suffixes, so the two rules can still be applied independently. The remaining case is the one of two rules $v \to v^n$ and $z \to z^n$ where $z = z_1 v z_2$. Also here confluence is given as shown by the following

$$z \to z^n \to z_1 v^n z_2 z^{n-1} \overset{n-1}{\to} (z_1 v^n z_2)^n \leftarrow z_1 v^n z_2 \leftarrow z.$$

Thus confluence is always given, though for $\bowtie_1^n$ the diamond property does not hold. □

Proposition 2.6.4. *Over a two-letter alphabet relations $\bowtie_m^n$ are not confluent for $m < n$ and $m \geq 2$.*

Proof. For such a relation look at the word $(aba(ab)^{m-1})^m$. It is as a whole an m-th power and can therefore be rewritten in one step to $w = (aba(ab)^{m-1})^n$, which contains n blocks of more than one consecutive letters a. On the other hand, around the border of to adjacent factors $aba(ab)^{m-1}$ of the original word we have the factor $(ab)^m$, which can be rewritten to $(ab)^n$. But then no rewriting spanning the entire word will be possible any more, because neither the initial ab nor the final $(ab)^{m-1}$ cannot be expanded. Thus we obtain by further application of rules from $\bowtie_m^n$ the language $ab[a(ab)^m)((ab)^{n-m})^*]^{m-1}a(ab)^{m-1})$, all of whose words have only m blocks of more than one consecutive letters a. Therefore w is not in this language, which suffices to prove our claim. □

Thus all cases of length-increasing relations are treated, and we come to the cases of length-reducing relations where confluence is equivalent to convergence toward a unique normal form. The first result holds even for alphabets of any size.

Proposition 2.6.5. *Relations* $\bowtie_m^0$ *are confluent only for* $m \leq 1$.

Proof. $\bowtie_1^0$ is trivially confluent, since here every word can be reduced to λ, and this is the only irreducible word for this relation. For greater m consider the word $(aab)^m(ab)^{m-1}$. It can be reduced by the rules $(aab)^m \to \lambda$ and $(ab)^m \to \lambda$ to $(ab)^{m-1}$ and $(aab)^{m-1}$ respectively. Both of these are irreducible, which proves our claim. □

Proposition 2.6.6. *For* $m > n \geq 1$ *over a two-letter alphabet relations* $\bowtie_m^n$ *are confluent for* $m = n + 1$.

Proof. The confluence of $\bowtie_2^1$ we can deduce indirectly from the fact that the only square-free words over two letters are λ, a, b, ab, ba, aba, and bab. $\bowtie_2^1$ can reduce every word to one of these, and the combination of first and last letter together with the total number of distinct letters uniquely identify the seven square-free words. At the same time these three properties are invariant under the application of rules from $\bowtie_2^1$. Thus all words derived from an original word can eventually be reduced to the same square-free word, which proves confluence.

We now proceed to the case $\bowtie_3^2$. For noetherian relations confluence is equivalent to local confluence, so we will show only that the latter property holds. As always there is no problem for local confluence, if two application sites of rules do not overlap; the rules can be applied independently. For overlapping sites we will distinguish several cases. Let the two applicable rules be $uuu \to uu$ and $vvv \to vv$ with $|u| \geq |v|$.

If the overlap includes no more than a square of each of the two cubes, where the rules are applied, then rule application is still independent. If, on the other hand, vvv is completely inside of one factor u, then it occurs in all three factors u. Let u' be the word obtained from u by applying $vvv \to vv$. Then we can either go from uuu to uu and in two steps to $u'u'$, or we can go in three steps from uuu to $u'u'u'$ and then to $u'u'$; so no matter which rule we apply first, we can arrive at $u'u'$. Thus the only interesting cases are, where v^3 transgresses both borders between the u.

If the central v includes the border between two u and $3|v| \leq |u|$, then two factors v^3 occur at the border between two u. Thus

there are a prefix p and a suffix s of v such that $v = ps$ and $u = spswpsp$ for some $w \in \Sigma^*$ as we can see from the central u. Thus $uuu = spswpspspswpspspswpsp$. Application of $vvv \to vv$ in two sites results in $spswpspswpspswpsp = sp(swpsp)^3$, which can be reduced to $sp(swpsp)^2$. This is the same word we obtain by first reducing $u^3 = (spswpsp)^3$ to $spswpspspswpsp$ and then applying $pspsps \to psps$ in the center. If the border between two u is inside one of the lateral v, then a similar argumentation applies; if the borders between u and v coincide, then things are even easier.

So it remains to treat the case that $3|v| > |u|$. Here both borders between the factors u form part of the occurrence of vvv, and thus v is overlapping. Let us look at the position of these borders relative to the v. If they occur at the same position in both v, then $|u| = 2|v|$, so $u^3 = \hat{v}^6$ for some conjugate $\hat{v}$ of v. Then also $u^2 = \hat{v}^4$. Thus applying the rule $uuu \to uu$ results in $\hat{v}^4$. Applying, on the other hand, $vvv \to vv$ produces $\hat{v}^5$, because of the conjugacy of v and $\hat{v}$. Then we can continue with one step of $\hat{v}\hat{v}\hat{v} \to \hat{v}\hat{v}$ to $\hat{v}^4$, which is the same word that was obtained in the first case.

So let the borders between the u occur at different points in the two v. We distinguish two cases: first, let $2|v| > |u|$. There is a factor of v^3 inside u^3 and $|3v| > |v| + |u|$. Therefore Theorem 1.1.1 applies showing that u^3 has period $|v|$. Thus u^3 must again be equal to $\hat{v}^6$ for some conjugate $\hat{v}$ of v.

It remains to treat the case where $2|v| < |u|$. From the position of the factors v relative to the borders we can see that v is overlapping and can be written in the form rtr for nonempty words r and t. The overlap r is partitioned into two words w_2 and w_1 by the border between the u such that $r = w_2w_1$. We can now write u^3 as $w_1trrtrrtrtrrtrrtrtrrtrrtw_2$. Applying first $vvv \to vv$ in the center results in $w_1trrtrrtrtrrtrrtrrtw_2$, which is equal to $w_1trr(trrtr)^3rtw_2$ and can therefore be reduced further to $w_1trr(trrtr)^2rtw_2$. But this can be written $w_1trrtrrtw_2w_1trrtrrtw_2 = u^2$ and is therefore equal to the result of applying $uuu \to uu$ to start with.

Thus also $\bowtie_3^2$ is confluent. Relations $\bowtie_m^n$ for $m \geq 4$ can be shown to be confluent in analogous manner for all cases, where v^m does not touch more than two factors u. If it does touch three or more such factors, then we have a common stretch of length greater than $2|u|$ with periods both $|u|$ and $|v|$. By Theorem 1.1.1 we see that u^m and v^m share the period $\gcd(|u|, |v|)$, and confluence follows easily. □

As soon as the difference between m and n becomes greater than

one, all relations $\bowtie_m^n$ are not confluent any more.

Proposition 2.6.7. *For $m > n$ over a two-letter alphabet relations $\bowtie_m^n$ are not confluent for $m \geq n + 2$.*

Proof. For a given relation $\bowtie_m^n$ with $m \geq n + 2$ let us look at the word $(a(ab)^{n+1})^m(ab)^{m-n-1}$. It can be reduced to the irreducible $(a(ab)^{n+1})^{m-1}aab^n$ and to $(a(ab)^{n+1})^n(ab)^{m-n-1}$, which can also be written $(a(ab)^{n+1})^{n-1}a(ab)^m$ and can be reduced further to $(a(ab)^{n+1})^{n-1}a(ab)^n$. These two normal forms are clearly different from each other, which suffices to prove our claim. □

Thus we have fully characterized the conditions under which relations $\bowtie_m^n$ are confluent over alphabets of one and two letters. Table 2.6.2 displays these results graphically.

$m \setminus n$	0	1	2	3	4	5	6	⋯
0	+	+	+	+	+	+	+	⋯
1	+	+	+	+	+	+	+	⋯
2	-	+	+	-	-	-	-	⋯
3	-	-	+	+	-	-	-	⋯
4	-	-	-	+	+	-	-	
5	-	-	-	-	+	+	-	
6	-	-	-	-	-	+	+	
⋮	⋮	⋮	⋮	⋮			⋱	⋱

Table 2.1: Confluence of $\bowtie_m^n$ over a two-letter alphabet. + and − denote confluence and non-confluence, respectively.

2.6.3 Regularity over Two Letters

For a two-letter alphabet the cases of insertion and deletion, i.e. languages $w^{\bowtie_0^1}$ and $w^{\bowtie_1^0}$, are both regular. This is known from work on insertion and deletion closure of regular languages, which has been summarized by Ito [45]. We now show that also the insertion of squares results in regular languages, while for cubes and words of higher powers regularity is not given any more. To prove the regularity of $w^{\bowtie_0^n}$, we first reduce it to a simpler case.

Proposition 2.6.8. *For a nonempty word w and all integers $k \geq 3$ we have $w^{\leq k}\bowtie_0^n = w^{\leq 2}\bowtie_0^n$ and consequently $w^{\bowtie_0^n} = w^{\leq 2}\bowtie_0^n$.*

Proof. We first show that $\lambda^{\leq 2}\bowtie_0^2 = \mathbb{E}$, where the language $\mathbb{E}$ consists of all words, which have an even number of both a and b. Let $R \subset {}^{\leq 2}\bowtie_0^2$ be the string-rewriting system $\{\lambda \to aa, \lambda \to bb, \lambda \to abab, \lambda \to baba\}$. Application of rules from both R and R^{-1} preserves the defining properties of $\mathbb{E}$. Since the same is true for rules $\lambda \to a^4$ and $\lambda \to b^4$, we have $\lambda^{\leq 2}\bowtie_0^2 \subseteq \mathbb{E}$.

To see that the inclusion holds also in the other direction, take an arbitrary word from $\mathbb{E}$. Apply rules $aa \to \lambda$ and $bb \to \lambda$ as often as possible. The resulting word will be either from $(abab)^*$ or $(baba)^+$. Thus it can be reduced to λ via rules $abab \to \lambda$ or $baba \to \lambda$ respectively. But if R^{-1} can reduce the word to λ, then R can generate it from λ. this proves $\lambda^{\leq 2}\bowtie_0^2 = \mathbb{E}$.

For all $k \geq 2$ we have $R \subset {}^{\leq k}\bowtie_0^2$ and also $R \subset \bowtie_0^2$, and all of these relations preserve even numbers of both the letters a and b. Since already R produces all of these words, all other rules are unnecessary in the sense that they do not add generative power. As a final observation note that insertions can take place only between the letters of the original word and thus $w^{\bowtie_0^n} = \lambda^{\bowtie_0^n} w[1]\lambda^{\bowtie_0^n} w[2]\lambda^{\bowtie_0^n} \dots w[|w|]\lambda^{\bowtie_0^n}$. The same holds for the bounded versions of square-insertion, and this proves the proposition. □

The regularity of $w^{\bowtie_0^n}$ for $n \leq 2$ follows almost immediately.

Proposition 2.6.9. *For a nonempty word w and an integer $n \leq 2$ the language $w^{\bowtie_0^n}$ is regular.*

Proof. As just mentioned, the case $n = 1$ was treated already in earlier work [45], $n = 0$ is trivial. After the proof of Proposition 2.6.8 for a word $w = x_1 x_2 \dots x_r$ of r letters it is straight-forward to see that $w^{\bowtie_0^n} = \mathbb{E}x_1\mathbb{E}x_2\mathbb{E}\dots x_r\mathbb{E}$. Since $\mathbb{E}$ is regular, also $w^{\bowtie_0^n}$ is regular. □

Before establishing the non-regularity of $\bowtie_0^n$ for $n \geq 3$ we state an important property of these relations.

Lemma 2.6.10. *Over a two-letter alphabet $\{a, b\}$ for every 2^+-free word u starting with ab and every integer $n \geq 3$ the shortest word v such that $uv \in \lambda^{\bowtie_0^n}$ fulfills $(n-2)|u| \leq |v| \leq (n-1)|u|$.*

Proof. That $(n-1)|u|$ is an upper bound is immediate by applying the rule $\lambda \to u^n$. For seeing that $(n-2)\frac{|u|}{2}$) is a lower bound consider the last rule $\lambda \to z^n$ applied in the generation of uv. It must produce at least $(n-2)|z|$ letters of v, otherwise a repetition of order greater than two would form part of u. In the optimal case $2|z|$ letters of u

are produced. Since every prefix of a 2^+-free word is also 2^+-free the same must be true for all rules applied before. Thus all in all at least $(n-2)|u|$ letters are produced for v. □

The upper bound is tight only for a few very short words like ab. To see why for longer words it is never reached let us look at a different construction establishing the same bound. We construct u by first applying the rule $\lambda \to u[1]^n$. Let $u[1] = a$, then either $u[2]$ or $u[3]$ must be b, because u is 2^+-free. We apply $\lambda \to b^n$ after the first or second letter respectively. Then the next a is inserted and so on. In the worst case we produce $n-1$ extra letters for every letter of u as above. However, this is only the case if in u the two letters alternate after every position. In a 2^+-free word this is possible for at most 4 positions, because a factor $ababa$ has repetitiveness $\frac{5}{2}$ already. Thus for words longer than 4 there is always a way to construct uv with $|v| < (n-1)|u|$

With these preliminaries stated we can prove non-regularity of $\bowtie_0^n$ for $n \geq 3$ by a refinement of a method originally developed by Wang [92].

Proposition 2.6.11. *For a nonempty word w and an integer $n \geq 3$ the language $w^{\bowtie_0^n}$ is in general not regular.*

Proof. We show that $\lambda^{\bowtie_0^n}$ is not regular. From Lemma 2.6.10 we see that the shortest word v, such that $uv \in \lambda^{\bowtie_0^n}$ for a 2^+-free u is such that $(n-2)|u| \leq |v| \leq (n-1)|u|$. We construct a series of 2^+-free words $(u_i)_{i\geq 1}$ such that $(n-2)|u_{i+1}| > (n-1)|u_i|$. This is possible since there exists an infinite 2^+-free word.

Consider now the corresponding series $(v_i)_{i\geq 1}$ of shortest words such that always $u_i v_i \in \lambda^{\bowtie_0^n}$. From the length bounds for the v_i it is clear that $u_i v_j$ cannot be in $\lambda^{\bowtie_0^n}$ for $i > j$. Thus the infinitely many u_i are in pairwise different equivalence classes of the syntactic congruence, which implies that there is an infinite number of such classes. By Theorem 1.2.5 the language $\lambda^{\bowtie_0^n}$ is not regular.

□

Now we establish similar length bounds for relations $\bowtie_1^n$, which will then allow us to prove their non-regularity along similar lines.

Lemma 2.6.12. *Over a two-letter alphabet $\{a, b\}$ for every 2^+-free word u starting with ab and every integer $n \geq 3$ there exists a word v, such that $uv \in (ab)^{\bowtie_1^n}$ and $|v| \leq (3(n-2)+2)(|u|-2)$.*

Proof. u being 2^+-free implies that there is no factor xxx for any letter $x \in \Sigma$. Thus the alphabet's containing only two elements guarantees that after at most 2 positions in u letters repeat, i.e. for every position in u its letter is repeated at most three positions later. Thus we can construct a word having u as a prefix in the following way: starting from ab, always one letter more of u is constructed per step. We take the shortest suffix z of the already constructed part of u starting with the next letter needed. On it we apply the rule $z \to z^n$ putting the required letter in the position. As exposed above, the maximum length of z is 3. This process takes exactly $|u|-2$ steps. In each step at $|z|(n-1) \leq 3(n-1)$ new letters are introduced. At least one of them forms part of u, and thus at most $3(n-2)+2$ additional letters are introduced, which proves the length bound on v. □

Lemma 2.6.13. *Over a two-letter alphabet $\{a,b\}$ for every 2^+-free word u starting with ab and every integer $n \geq 3$ there exists no word v, such that $uv \in (ab)^{\bowtie_1^n}$ and $|v| \leq \log_2(|u|/2)$.*

Proof. u is obtained from ab by the application of rules $z \to z^n$. Since u is 2^+-free and $n \geq 3$, every such rule must produce at least one additional symbol outside of u, therefore contributing to v. At the same time each rule produces at most $2|\ell|$ letters of u, where ℓ is the rule's left side. Thus at least $\log_2(|u|/2)$ rules must be applied, since our starting word has length 2 and each idempotency rule can at most double the length of the subword of u already produced. Consequently, v is at least $\log_2(|u|/2)$ symbols long. □

Proposition 2.6.14. *Over a two-letter alphabet for every word w and integers $n \geq 3$ the language $w^{\bowtie_1^n}$ is not regular, while $w^{\bowtie_1^2}$ is.*

Proof. The regularity of $w^{\bowtie_1^2}$, i.e. for the case of duplication, was proven by Dassow et al. [26].

For $n \geq 3$ Lemmata 2.6.12 and 2.5.10 show us that for every 2^+-free word u starting with ab and every integer $n \geq 3$ there exists a word v, such that $uv \in (ab)^{\leq 3 \bowtie_1^n}$ and $\log_2(|u|/2) < |v| \leq (3(n-1)+2)(|u|-2)$, i.e. the length of a minimal v is bounded from above and below.

Now we take an arbitrary infinite 2^+-free word starting with ab and produce a sequence of prefixes $(u_i)_{i \geq 1}$ such that $(3(n-1)+2)(|u_i|-2) < \log_2(|u_{i+1}|/2)$. Then the v_i from the construction in the proof of Lemma 2.6.12 are by their construction such that $u_i v_i \in w^{\leq k \bowtie_1^n}$ and $|v_i| \leq (3(n-1)+2)(|u_i|-2)$. By Lemma 2.5.10 we see that $u_{i+1} v_i \notin w^{\leq k \bowtie_1^n}$, because the shortest word v such that $u_{i+1} v \in w^{\leq k \bowtie_1^n}$

is such that $|v| > \log_2(|u_{i+1}|/2)$. But by our choice of the u_i we have $|v_i| < \log_2(|u_{i+1}|/2)$.

Therefore all our words u_i are in pairwise different equivalence classes of the syntactic right-congruence of $w^{\leq k}\bowtie_1^n$, and therefore the language cannot be regular. □

We mention here that the regularity of $w^{\bowtie_1^2}$ was also proven by Ito et al. [47] along quite different lines from those of Dassow et al. [26]. The key result there is the following, which is similar in nature to Proposition 2.6.8, and which will be treated in more detail in Section 3.4.

Proposition 2.6.15. *Over an alphabet of two letters we have* $w^{\leq k}\bowtie_1^2 = w^{\leq 2}\bowtie_1^2$ *and consequently* $w^{\bowtie_1^2} = w^{\leq 2}\bowtie_1^2$ *for all words* w *and for* $k \geq 2$.

Regularity trivially holds for relations $\bowtie_m^n$ with $n \leq m$, which generate only finite languages. Therefore the only interesting cases left are those where $2 \leq m < n$. An interesting observation is that in the cases treated so far the languages generated are regular exactly in those cases, where an equivalent finite system of rewrite rules generates the same language. We strongly conjecture that this holds for relations $\bowtie_m^n$ where $2 \leq m < n$, and we also think that only regular languages are generated by these relations. However, a trivial length bound on the left side of rules such as the length of the original word does not hold as the following example illustrates.

Example 2.6.16. Let w be the word $a^{10}b^3a^3b^3a^{10}b^3$, which has length 32. Consider the language $w^{\bowtie_3^4}$. By application of rules $a^3 \to a^4$ and $b^3 \to b^4$ we can arrive from w at the word $(a^{10}b^{10})^3$; from here by application of the rule $(a^{10}b^{10})^3 \to (a^{10}b^{10})^4$ we obtain words with more than 3 changes from a to b within the word. It is quite clear that with shorter rules such words cannot be obtained, and therefore $w^{=k}\bowtie_3^4 \neq w^{\bowtie_3^4}$ for $k < 60$.

Thus the question of regularity remains to be answered for an interesting class of idempotency languages, and, of course, analogous questions can be considered for alphabets of three or ore letters. For the non-regular variants is remains to determine, whether they are context-free or not. Another interesting question is, whether local confluence always implies general confluence for the idempotency relations considered here.

2.6.4 Confluence

In the cases of deletions and insertions increasing the alphabet size does not matter so much and we can still establish the confluence of the corresponding relations.

Proposition 2.6.17. *The relations* $\bowtie_0^n$ *for* $n \geq 0$ *and* $\bowtie_1^0$ *are confluent.*

Proof. The confluence of $\bowtie_1^0$ is trivial, because here every word can be reduced to λ and this is the only irreducible word. For the confluence of $\bowtie_0^n$, on the other hand, the same proof as for the bounded case in Lemma 2.5.1 applies. □

Proposition 2.5.3 states that for all $k \geq 3$ and $n \geq 2$ the relation ${}^{\leq k}\bowtie_1^n$ is not confluent over an alphabet of three or more letters. When dropping the length bound, the proof technique used there cannot be applied any more. The only thing we can state here is local confluence for relations $\bowtie_1^n$, which might indicate that general confluence holds.

Proposition 2.6.18. *The relations* $\bowtie_1^n$ *for* $n \geq 2$ *are locally confluent.*

Proof. So let $u \leftarrow w \rightarrow v$. As usual, unless one application site is properly inside the other, the diamond property holds. Otherwise we can factorize $w = w_1w_2w_3w_4w_5$ such that without loss of generality $u = w_1(w_2w_3w_4)^nw_5$ and $v = w_1w_2w_3^nw_4w_5$. Then via rules (w_3, w_3^n) and $(w_2w_3^nw_4, (w_2w_3^nw_4)^n)$ we obtain $u \xrightarrow{n} w_1(w_2w_3^nw_4)^nw_5 \leftarrow v$, which proves our claim. □

Proposition 2.6.19. *The relations* $\bowtie_m^0$ *are not confluent for* $m \geq 2$*.*

Proof. We consider the word $a^mb(a^{m-1}b)^{m-1}$. It contains two factors, which are powers of order m, namely a^m and $(a^{m-1}b)^m$. Reducing the first one results in $b(a^{m-1}b)^{m-1}$, reducing the second one results in a; both are irreducible, and thus the reduction relation is not confluent. □

Proposition 2.6.20. *The relations* $\bowtie_m^1$ *are not confluent for* $m \geq 2$*.*

Proof. For the case $n = 2$ already Example 2.1.2 provides the appropriate counterexample of $(abcbabcbc)^{\bowtie_2^1} = \{abc, abcbc, abcbabc\}$. This can be generalized by using for a given n the word $(abcb)^mc(bc)^{m-2}$, which can be reduced to the two irreducible words abc and $(abcb)^{m-1}abc$. □

2.6.5 Regularity

As already stated in Section 2.2.4, the cases of insertion and deletion, i.e. languages $w^{\bowtie_0^1}$ and $w^{\bowtie_1^0}$, are both regular, see Propositions 2.2.7 and 2.2.8. Non-regularity can be established in several cases in similar ways to the proofs for bounded idempotencies, only the length bounds change. We first fix these in a few lemmata.

Lemma 2.6.21. *Over a two-letter alphabet $\{a, b\}$ for every 2^+-free word u starting with ab and every integer $n > 1$ there exists a word v, such that $uv \in (ab)^{\bowtie_1^n}$ and $|v| \leq (3(n-1)+2)(|u|-2)$.*

Proof. The same construction as in the proof of Lemma 2.5.10 applies. □

While the upper bound carries over, the lower bound is significantly lower than the one for the bounded case stated in Lemma 2.6.22. The length bound provided is not tight, but suffices for our purposes.

Lemma 2.6.22. *Over a two-letter alphabet $\{a, b\}$ for every 2^+-free word u starting with ab and every integer $n \geq 3$ there exists no word v, such that $uv \in (ab)^{\leq 3 \bowtie_1^n}$ and $|v| \leq \log_2(|u|/3)$.*

Proof. u is obtained from ab by the application of rules $z \to z^n$. Since u is 2^+-free and $n \geq 3$ every such rule must produce at least one additional symbol outside of u, therefore contributing to v. At the same time each rule produces at most $2|\ell|$ letters of u, where ℓ is the rule's left side. Thus at least $\log_2(|u|/3)$ rules must be applied, since our starting word has length 3 and each idempotency rule can at most double the length of the subword of u already produced. Consequently, v is at least $\log_2(|u|/3)$ symbols long. □

Lemma 2.6.23. *Over a three-letter alphabet $\{a, b, c\}$ for every square-free word u starting with abc and every integer $n > 1$ there exists a word v, such that $uv \in (abc)^{\bowtie_1^n}$ and $\log_2(|u|/3) \leq |v| \leq (|u|-3)(4(n-1)+3)$.*

Proof. uv can be constructed starting from abc as in the proof of Lemma 2.5.12. Only the lower bound for the length here corresponds to the one from Lemma 2.6.22. □

Proposition 2.6.24. *Over a two-letter alphabet for every word w and integers $n \geq 3$ the language $w^{\bowtie_1^n}$ is not regular, while $w^{\bowtie_1^2}$ is.*

Proof. The regularity of $w^{\bowtie_1^2}$, i.e. for the case of duplication, was proven by Dassow et al. [26]. For $n \geq 3$ Lemmata 2.6.21 and 2.6.22 allow a proof completely analogous to the one of Proposition 2.5.13. □

With more than two letters, also here the special case of $n = 2$ is not regular any more; the proof can again be done by the same method as for bounded idempotencies and the length bounds from Lemma 2.6.23.

Proposition 2.6.25. *Over a three-letter alphabet for every word w and an integer $n > 1$ the language $w^{\bowtie_1^n}$ is not regular.*

With this we close this section and also this chapter. The results on unbounded cases are much less than for the bounded ones; mainly we have only those ones that carry over in some way from the case of bounded length. Thus much remains to be done in this direction, but the problems left open seem vary hard in general.

3 Duplication

The special case of duplication was the origin of the investigations on idempotency languages as presented so far. Also it is the case with most motivation from a practical point of view, namely from the duplications occurring in DNA strands as presented in Section 2.1. Therefore there exist some results, which have not been generalized to general idempotencies and also some results, which seem to be of interest only for duplication like the duplication codes defined further down. This chapter collects results of this type.

First off, we dedicate a section to the discussion of the general duplication language and the reasons, why it is so hard to prove its non-context-freeness. Then some properties of duplication languages are presented, which have not been stated in the preceding chapters; mainly they concern related decidability questions.

The next section will introduce the concept of duplication root; first its motivation from other concepts of root of a word will be explained, then a number of results are presented. Following this, we investigate a special type of code that is resistant to duplications occurring in its code words. Finally, we apply duplication not just to single words but to entire languages; here we mainly focus on the question, whether this preserves regularity and context-freeness.

3.1 General Duplication

Since in the current chapter we will speak almost exclusively about relations $\bowtie_m^n$ where $m = 1$ and $n = 2$, we introduce a simpler notation omitting these two redundant parameters. The symbol $\heartsuit$ seems quite appropriate for the duplication operation, because viewed from bottom to top it goes from one origin to two equal halves. Thus we will henceforth write $\heartsuit$ instead of $\bowtie_1^2$, $\heartsuit^{\leq k}$ instead of ${}^{\leq k}\bowtie_1^2$ and $\heartsuit^k$ instead of ${}^{=k}\bowtie_1^2$; this way we also save the equality sign in the latter relation. The languages generated from a word by the respective rewrite relations are denoted by $w^{\heartsuit}$. $w^{\heartsuit \leq k}$, and $w^{\heartsuit k}$.

3.1.1 Context-Freeness

We will try to shed some light on the reasons for the complicatedness of the problem of determining whether general duplication languages are context-free. The main tools in formal language theory for proving a language non-context-free are pumping lemmata and Parikh's Theorem about semi-linear languages. The latter holds for all idempotency languages in a very straight-forward manner. The Parikh sets are actually not just semi-linear but even linear in the algebraic sense of the term, which, however, is different from the meaning for formal languages.

Proposition 3.1.1. *For every word $w \in \Sigma^*$ the language $w^\heartsuit$ is semi-linear.*

Proof. For all letters $x \in \Sigma^*$ there are rules (x, xx) in $\heartsuit$. Thus any letter occurring in w can be duplicated increasing its number of occurrences by one. This way we generate the following Parikh set:

$$\left\{ \psi(w) + \sum_{x \in \mathrm{alph}(w)} \ell_x \cdot \psi(x) : \ell_x \in \mathbb{N} \text{ for all } x \in \mathrm{alph}(w) \right\}.$$

It it obvious that the Parikh vectors of any word obtained from w by longer duplications are already in this set. Thus it is equal to $\psi(w^\heartsuit)$. Since this set is linear, the language $w^\heartsuit$ is semi-linear. □

Thus Parikh's Theorem 1.2.8 does not provide us with means to show that $w^\heartsuit$ is in general not context-free.

Neither can pumping lemmata like Lemma 1.2.7 provide us with any way to easily prove the non-context-freeness of duplication languages. If a word w can be factorized as $w_1 w_2 w_3 w_4 w_5$, then by definition all words $w_1 w_2^i w_3 w_4^i w_5$ are in $w^\heartsuit$ for $i \geq 1$ by rules $(w_2, w_2 w_2)$ and $(w_4, w_4 w_4)$; the only hope might be the case, where $i = 0$.

So both the Parikh Theorem and the pumping lemmata seem to be fulfilled by duplication languages because of their density, i.e. because they contain so many words. We will now show that duplication languages are indeed very dense also in the formal meanings of the word. Recall that density for a language means to contain any word as a factor in one of the language's words. With a construction similar to the ones used in Lemma 2.5.10 and the following ones, we can show that duplication languages are dense.

Proposition 3.1.2. *Every language $w^\heartsuit$ is dense over the alphabet $alph(w)$.*

Proof. Let alph(w) be $\{a_1, a_2, \ldots, a_\ell\}$ and without restriction of generality let the letters occur in the order starting from a_1 with a_ℓ being the letter with the latest first occurrence in w. We now give a method to construct an arbitrary word u letter by letter in the position just following the first occurrence of a_ℓ. For this let us put a marker θ just after this position.

$u[1]$ is in alph(w) and has an occurrence left of θ. Now duplicate the factor starting in such an occurrence and reaching until θ. This will leave the letter $u[1]$ just after the marker θ. Then we move the marker one position to the right and repeat the procedure for $u[2]$. In this manner we will finally arrive at the entire word u, and thus any word can occur as a factor in $w^\heartsuit$, which proves our claim. □

There is also another notion of density for languages. The function $n \mapsto |\Sigma^n \cap n|$ is called the density function of L. The maximum growth such a function can have is exponential, and this is reached by duplication languages.

Proposition 3.1.3. *The densities of $w^\heartsuit$ and its complement grow exponentially for alph(w) > 1.*

Proof. If alph(w) > 1, then somewhere in w there is a factor xy for letters x and y distinct from each other. At this place we can construct any word u in $x\{x,y\}^*y$ in the following way: let u have ℓ changes from the letter x to y; then duplicate xy until reaching $(xy)^\ell$. Now by rules (x, xx) and (y, yy) we can multiply each of the letters to the number, in which it occurs in the respective block and we obtain u.

So let w_1w_2 be the factorization of w such that the last letter of w_1 is the x from above. Then we have shown that $w_1\{x,y\}^*w_2$ is a subset of $w^\heartsuit$. Since the density function of $\{x,y\}^*$ is $\lambda i.2^i$, the density function of $w_1\{x,y\}^*w_2$ is at least $\lambda i.2^{i-|w|}$ for all values greater than $|w|$. Thus also the density function of $w^\heartsuit$ must grow exponentially.

For the complement of $w^\heartsuit$ things are rather obvious. Let x be a letter different from $w[1]$ and let y be a letter different from the last one of w. Then $x\Sigma^*y$ is a subset of the complement of $w^\heartsuit$, and already its density grows exponentially. □

These results explain in part, why the pumping lemmata and the Parikh Theorem fail to prove duplication languages non-context-free. Intuitively speaking, they cannot find an appropriate gap in $w^\heartsuit$, because these languages are so dense.

3.1.2 Decidability Questions

When a new class of languages is defined, one of the first things to be investigated is always, which of their properties are decidable. This section states a few decidability results for duplication languages. The first one actually shows that being a duplication language is a decidable property for regular languages. In the proof we use several of the properties we have established in prior sections.

Proposition 3.1.4. *Given a regular language L one can algorithmically decide whether or not L is an unbounded duplication language.*

Proof. The algorithm works as follows:

(i) We find the shortest string $z \in L$, for regular languages this can be done algorithmically. If there are several strings in L of the length of z, then L is not an unbounded duplication language.

(ii) We now compute the cardinality of $\mathrm{alph}(z)$.

(iii) If $|\mathrm{alph}(z)| \geq 3$, then there is no w such that $L = w^{\heartsuit}$, see Proposition 2.6.25.

(iv) If $|\mathrm{alph}(z)| = 1$, then L is an unbounded duplication language if and only if $L = \{a^{|z|+m} \mid m \geq 0\}$, where $\mathrm{alph}(z) = a$.

(v) If $|\mathrm{alph}(z)| = 2$, $z = z_1 z_2 \dots z_n$, $z_i \in \mathrm{alph}(z)$, $1 \leq i \leq n$, L is an unbounded duplication language if and only if

$$L = z_1^+ e_1 z_2 e_2 \dots e_{n-1} z_n^+, \tag{3.1}$$

where

$$e_i = \begin{cases} z_{i+1}^*, & \text{if } z_i = z_{i+1} \\ \{z_i + z_{i+1}\}^*, & \text{if } z_i \neq z_{i+1} \end{cases}$$

for all $1 \leq i \leq n-1$. Note that one can easily construct a deterministic finite automaton recognizing the language in the right-hand side of equation (3.1).

The condition used in step (v) was provided in the initial article about duplication languages by Dassow et al. [26]; the condition for step (iv) follows from it and is almost trivial at any rate. □

Now we come to a few more special decision problems that mainly concern relations between two words and the duplication languages generated by them.

Proposition 3.1.5. *The following problems are algorithmically decidable for unbounded duplication languages:*

Membership: Given u and v, is u in $v^\heartsuit$?

Inclusion: Given u and v, does $u^\heartsuit \subseteq v^\heartsuit$ hold?

Equivalence: Given u and v, does $u^\heartsuit = v^\heartsuit$ hold?

Regularity: Given u, is $u^\heartsuit$ a regular language?

Proof. Clearly, the membership problem is decidable by generating all words in $v^\heartsuit$, which are not longer than u. and inclusion can be reduced to it, because we have $u^\heartsuit \subseteq v^\heartsuit$ iff $u \in v^\heartsuit$.

Clearly $u^\heartsuit = v^\heartsuit$ holds only if, $|u| = |v|$ and thus $u = v$. In conclusion, $u = v$ iff $u^\heartsuit = v^\heartsuit$. This implies that the equivalence problem is decidable in linear time by simply deciding the equality of the two given words.

The regularity can again be decided using Proposition 2.6.25: if $|\mathrm{alph}(u)| \geq 3$, then $u^\heartsuit$ cannot be regular, otherwise it definitely is. □

3.2 Roots

As mentioned in the introductory Section 2.1, it is interesting for the phylogenetic analysis of a DNA sequence in a genome to reconstruct its duplication history. This means to determine what original sequence it might have come from via iterated duplications. In general, $w^{\bowtie_2^1}$ is the set of candidates for a sequence w. Since our objective is not so much phylogenetic analysis, but the language theoretic investigation of the duplication operation, we will however only look at the primitives that can be obtained in this way. This type of research follows a tradition of reducing a word to something primitive called its root.

In Formal Language Theory several concepts of *root* have been defined. The most common one is probably the one of *primitive root*. It is based on the fact that for every non-empty word w there exists a unique primitive word p such that $w \in p^+$; this unique p is called the root of w [61, 10]. The concept of root was generalized to languages in the canonical way: the root of a language is the set of roots of all the words contained in this language.

We will now illustrate the use of the notion of primitive root in a few exemplary results. Then we provide a short and informal overview of other notions of root and then define idempotency roots in the same spirit. Following this, we investigate the same questions for the case

of duplication that have been addressed for the primitive roots of languages.

3.2.1 Primitive Roots

Primitiveness of words is a concept widely used, for example, in the theory of codes [10]. As already stated in Section 1.1 a word is primitive, iff it is not a non-trivial power of any word. The primitive word p such that $w \in p^+$ is unique for every word w. Based on this, the primitive root $\sqrt{w}$ of a non-empty word w is defined to be the primitive word p such that $w \in p^+$.

This definition is extended from words to languages in the canonical way such that $\sqrt{L} := \{\sqrt{w} : w \in L\}$. The main focus in investigations on the primitive roots of languages was on decision problems related to the finiteness and regularity of the root [40, 42, 60].

As an example for interest motivated from another point, Head [38] proposed a way of visualizing a language in a discrete, two-dimensional coordinate system with Q in some order on one axis and the words' degree on the second axis. Due to the uniqueness of the primitive root, we have a one-to-one correspondence between words and points in the plane. For this, languages with finite roots are especially interesting, because they can be represented within finite width.

In the regular case, there is a characterization of the languages with finite or infinite root exists by means of so-called root terms [42]. Further, Lischke [60] has shown that already regular languages can have almost arbitrarily complicated roots. We now take a look at the same question for the next class in the Chomsky Hierarchy, the context-free languages.

Proposition 3.2.1. *All context-free languages with finite primitive root are regular.*

Proof. Let $\{p_1, p_2, \ldots, p_n\}$ be the finite root of a context-free language L. Then for $i \in \{1, \ldots, n\}$ the languages $L_i := \{u : u \in L \wedge \sqrt{u} = p_i\} = L \cap p_i^*$ are also context-free (by the closure of context-free languages under intersection with regular languages) and disjoint (by the uniqueness of the primitive root). We have $L = \bigcup_{i=1}^{n} L_i$.

Now we restrict our attention to just one fixed L_i and define a homomorphism ϕ_i as $\phi(p_i) := a$ for some letter a. Since L_i is context-free, also $\phi_i(L_i)$ is context-free by the closure of context-free languages under homomorphisms. Further we know from a theorem of Harrison

[36] that over a one-letter alphabet the regular and context-free languages coincide. Thus $\phi_i(L_i)$ is even regular. Finally, because regular languages are closed also under inverse homomorphisms considering the ϕ_i^{-1} shows that all the constituting languages L_i are regular. Summarizing we see that L is a finite union of such such regular L_i; therefore L itself is regular. □

We will now use this fact to design a decision procedure for the question, whether the root of a context-free language is finite or not. To this end we first collect a few useful results.

Lemma 3.2.2. *Every language with finite primitive root is slender.*

Proof. Let $\{p_1, p_2, \ldots, p_n\}$ be the language's root. Every p_i^* contains at most one word of any given length. Therefore $L = \bigcup_{l=1}^{n} p_l^*$ contains at most n words of any given length. □

Ilie [43], [44] and Raz [78] have both shown that slenderness is a decidable property for context-free languages. Further they have also provided effective procedures to compute a decomposition of those languages, which are slender, into finite numbers of paired loops, the term for languages of the form $\{w_1 w_2^i w_3 w_4^i w_5 : i \in \mathbb{N}\}$. We will now investigate in more detail the properties of such paired loops, and this will then allow us to tackle the decidability problem mentioned above.

Lemma 3.2.3. *For every factorization $w = w_1 w_2 w_3 w_4 w_5$ of a word $w \in \Sigma^*$ the (paired loop) language $L = \{w_1 w_2^i w_3 w_4^i w_5 : i \in \mathbb{N}\}$ either contains infinitely many primitive words, or $\sqrt{L}$ consists of just one word, that is $L \subseteq \sqrt{w}^*$.*

Proof. The degree of a word is invariant under cyclic permutation. Thus we can in the following consider words $w_2^i w_3 w_4^i w_5 w_1$ instead of working with the original $w_1 w_2^i w_3 w_4^i w_5$. We will call these words $w(i)$ and the resulting language L'. If we suppose that the root of L' is finite, then there is a primitive word p from this root such that the language $p^* \cap L'$ is infinite and therefore for arbitrarily large n we can find $i \geq n$ such that $w(i) \in p^*$.

Now with Theorem 1.1.1 for some large enough i we see that p is also the root of w_2, because w_2^i is always a prefix of p^ω. Then also w_3 is a prefix of p^ω. Similarly the root of w_4 must be a conjugate of p, and $w_5 w_1$ is a suffix of ${}^\omega p$. It is clear that $|w_1 w_3 w_5|$ must be divided by $|p|$ or be zero, because p is the root of infinitely many $w(i)$. But then we must have $w_3 w_5 w_1 \in p^*$. Adding words w_2 in front will not

change this and neither will adding words w_4 (i.e. words, whose root is a conjugate of p) in the specified place do so, because both have period and length $|p|$. Therefore all words $w(i)$ have the root p.
□

A second look at the last part of the proof also allows us to state another result without further proof.

Lemma 3.2.4. *In every paired loop* $\{w_1 w_2^i w_3 w_4^i w_5 : i \in \mathbb{N}\}$ *with finite (i.e. singleton) root, the lengths of* w_2 *and* w_4 *are both multiples of* $|\sqrt{w_1 w_2 w_3 w_4 w_5}|$ *and the language described is regular.*

Proof. In the case of a singleton root in Lemma 3.2.3 in every step from i to $i+1$ the degree of the word is increased by a constant number, more exactly by $|w_2 w_4|/|\sqrt{w}|$. Thus the entire language has the form

$$\sqrt{w}^{|w_1 w_3 w_5|/|\sqrt{w}|}(\sqrt{w}^{|w_2 w_4|/|\sqrt{w}|})^*$$

and is regular. This uses the fact that concatenation is commutative for words with equal roots. □

Our considerations up to this point in combination with Ilie's and Raz's results allow us now to provide a different decision procedure for the question treated by Horváth and Ito.

Theorem 3.2.5. *For any context-free language it is decidable, whether its primitive root is finite.*

Proof. First we decide whether the given context-free language is slender. If not so, then according to Lemma 3.2.2 its root is infinite. Otherwise we compute the paired loops it consists of. Now it is easy to find the root of the defining word of each one.

If for each one the iterated sections (that is the respective w_2 and w_4) have as lengths multiples of the respective roots' lengths, then by Lemma 3.2.4 all the paired loops are regular and at the same time the given language's root is finite. Otherwise the given language has infinite root, because there is already one of the paired loops, which contains infinitely many primitive words by Lemma 3.2.3 and is a subset of the language under consideration. □

From this proof we see further that for context-free (in this case regular) languages with finite primitive root, this root can be effectively constructed as the paired loops can be constructed. The final extraction of the (singular) root of each loop is then trivial. This was not stated in the earlier work by Horváth and Ito, although such a construction could also be realized based on their method of proof.

3.2.2 Other Roots

In combinatorics of words not only integer powers of words have been considered, but also rational powers. Thus $ababb^{\frac{7}{5}} = ababbab$. The primitive words under this notion are the ones whose shortest period is equal to their length; these are the non-empty words w such that for rational r the equality $w = u^r$ implies $r = 1$ and $u = w$. In the literature numerous terms have been used for them; most commonly they have been called unbordered [19] or non-overlapping [85], but also dipolar [86], primary [61], d-primitive [85], and aperiodic [41] words.

Analogous to the primitive root, Horvath and Ito defined the *periodicity root* of a word w to be the shortest word u such that w is a prefix of u^ω [41]; alternatively it can be characterized as the prefix of length of the shortest period, which is where the name is motivated from. The same notion of root was used under the simple name of root by Carpi and de Luca [17].

Another variation of the primitive root is treated by Krawetz [48]. He defines the root of a language L not only to consist of all primitive words p such that $p^+ \cap L \neq \emptyset$, but drops the condition of primitiveness:

$$root(L) := \{w : \exists n[n \geq 1 \wedge w^n \in L]\}.$$

The main focus of his investigations is on the change of state complexity effected by this operation on regular languages.

Fazekas [33] defines the *scattered root* of a word derived from the *shuffle root* of a set of words, which was introduced by Berstel and Boasson [9]. If a word w can be reached by shuffling some other word u several times with itself, and if u is primitive under this notion, then u is the scattered root of w.

Further, notions of root have been defined along similar lines also for languages instead of single words. Shyr calls R a root of the language L, if there exists an integer i such that $L = R^i$ [85]; a variation of this is the notions of premotif, where R has to be such that $L = \bigcup_{i \in \mathbb{I}} R^i$ for a set $\mathbb{I}$ of integers [5].

3.2.3 Idempotency Roots

As we have seen, all the mentioned notions of root reduce a word to another one, which is primitive or elementary under some notion. For an idempotency relation $\bowtie^n_m$ the primitive words are the ones which do not contain any repetition of order n, more formally it is the

set $IRR(\bowtie_n^m)$; we want to emphasize here that it is not $IRR(\bowtie_m^n)$. To obtain such a word, we can iteratively apply rewriting rules from the inverse relation $\bowtie_n^m$. Of course, this makes sense only if $n > m$ such that the inverse relation is noetherian, and this process ends at some point. Therefore we will assume for the remainder of this section that $n > m$ for all idempotency relations $\bowtie_m^n$ in question without explicitly stating this every time.

Another problem lies in the fact that unlike all the notions of root defined above, the result is not always unique, but in general only for convergent relations $\bowtie_n^m$; in all other cases the root can be a set of words. With these things in mind we define the idempotency root as follows.

Definition 3.2.6. For $n > m$ the $\bowtie_m^n$*-root* of a non-empty word w is

$$\sqrt[\bowtie_m^n]{w} := IRR(\bowtie_n^m) \cap w^{\bowtie_n^m}.$$

As usual, this notion is extended in the canonical way from words to languages such that

$$\sqrt[\bowtie_m^n]{L} := \bigcup_{w \in L} \sqrt[\bowtie_m^n]{w}.$$

The roots $\sqrt[=k\bowtie_m^n]{w}$ and $\sqrt[\leq k\bowtie_m^n]{w}$ are defined in completely analogous ways, and also these are extended to entire languages in the canonical way.

First off we notice that an analogue to Proposition 3.2.1 does not hold for any version of idempotency roots.

Proposition 3.2.7. *For $m \leq n$ there are languages L in $CF \setminus REG$ for which $\sqrt[\bowtie_m^n]{L}$ is finite. The same holds for $\sqrt[=k\bowtie_m^n]{L}$ and $\sqrt[\leq k\bowtie_m^n]{L}$.*

Proof. Consider the language $L = \{a^\ell b^\ell : \ell > 0\}$, which is context-free but not regular. Then $\sqrt[\bowtie_m^n]{L} = \{a^\ell b^\ell : m \leq \ell < n\}$, also $\sqrt[\leq k\bowtie_m^n]{L} = \{a^\ell b^\ell : m \leq \ell < n\}$. Finally $\sqrt[=k\bowtie_m^n]{L} = \{a^\ell b^\ell : km \leq \ell < kn\}$. □

In some sense this shows that iteration of idempotencies can create more complicated structures from a finite set than iteration of concatenation. Intuitively, the reason for this that catenation only adds to the end of a word, while here we obtain nested structures. Of course, there are also non-context-free languages with finite primitive root, even non-enumerable ones like $(ab)^K$, where K is some non-enumerable set of numbers; but this language cannot be created by iterated catenation of ab, only very selected words from $(ab)^*$ are taken.

The primitive words have received their name from being primitive under the notion of catenation and the related root. Also for our roots there are primitive words, namely those that do not have any repetition of order n for $\bowtie_m^n$. We now take at look at the complexity of the sets of all such words; these are exactly the roots of Σ^*. In the length-bounded cases these are rather simple.

Proposition 3.2.8. *For all positive m, n the languages $\sqrt[=k]{\bowtie_m^n \Sigma^*}$ and $\sqrt[\leq k]{\bowtie_m^n \Sigma^*}$ are regular.*

Proof. We consider the complement of the respective languages, that is the language of all words containing a repetition of order n and length exactly or maximally k. This language can be recognized by a non-deterministic finite automaton, which operates in the following way: it just reads the input string, and at some point guesses that a repetition of length k (or shorter) and of order n starts. Then it stores the next k letters in its states and matches them $n-1$ times against the following k letters. If this match is successful, then the rest of the input is read, and the word is accepted. In all other cases the input is rejected.

Clearly this automaton accepts the complement of the respective root of Σ^*, which therefore is regular. Because the regular languages are closed under complementation, also the root itself is regular. □

For the unbounded case, the languages of irreducible words are not regular any more, they are not even context-free.

Proposition 3.2.9. *For all positive m, n the language $\sqrt[\bowtie_m^n]{\Sigma^*}$ is not context-free.*

Proof. Every context-free language L must fulfill the Pumping Lemma 1.2.7; this means that if it is infinite, then there exists some word $w \in L$ with a factorization $w = w_1 w_2 w_3 w_4 w_5$ with $w_2 w_4 \neq \lambda$ such that $\{w_1 w_2^i w_3 w_4^i w_5 : i \geq 0\} \subset L$. As a consequence of this, for every infinite context-free language there is no bound on the degree of repetitiveness of factors of the words it contains. Thus none of these languages can be $\sqrt[\bowtie_m^n]{\Sigma^*}$ for $n \geq 2$. □

We want to mention here that also for the complement of $\sqrt[\bowtie_1^2]{\Sigma^*}$ non-context-freeness has been established. This was a long-standing open problem, which was independently solved by Ross and Winklmann [79] and by Rozenberg and Ehrenfeucht [31].

3.2.4 Finiteness of the Duplication Root

In some way, all the roots described can be seen as generating sets for the given language, though not in a strict sense, because they usually generate larger sets. Still, one of the main questions about generating sets in algebra seems especially interesting also here: does there exist a finite generating set? Or in our context: is the root finite? Trivially, duplication roots are finite over two letters.

Proposition 3.2.10. *Over a two-letter alphabet for every language L its duplication root $\sqrt[\heartsuit]{L}$ is finite.*

Proof. It is well-known that over an alphabet of two letters there exist only six non-empty square-free words. Since $\sqrt[\heartsuit]{L}$ contains only square-free words, it must be finite. □

As in most cases for confluence and regularity, things become more difficult over three or more letters. Let us first define the *letter sequence* $\mathrm{seq}(u)$ of a word u as follows: any word u can be uniquely factorized as $u = x_1^{i_1} x_2^{i_2} \cdots x_\ell^{i_\ell}$ for some integers $\ell \geq 0$ and $i_1, i_2, \ldots, i_\ell \geq 1$ and for letters $x_1, x_2, \ldots, x_\ell$ such that always $x_j \neq x_{j+1}$; then $\mathrm{seq}(u) := x_1 x_2 \cdots x_\ell$. Intuitively speaking, every block of several adjacent occurrences of the same letter is reduced to just one occurrence.

We now collect a few elementary properties that connect a word's letter sequence with duplication and duplication roots.

Lemma 3.2.11. *If for two words $u, v \in \Sigma^*$ we have $seq(u) = seq(v)$, then there exists a word w such that $u(\heartsuit^{-1})^* \; w \; \heartsuit^* v$, i.e. both u and v are reducible to w via unduplications.*

Proof. This is immediate, since every word can be reduced to its letter sequence via rules (xx, x) for $x \in \Sigma$. Thus our statement can be satisfied by setting $w = \mathrm{seq}(u)$. □

Now we state a result that links the letter sequence and the duplication root of a word in a fundamental way.

Lemma 3.2.12. *If for two words $u, v \in \Sigma^*$ we have $seq(u) = seq(v)$, then also $\sqrt[\heartsuit]{u} = \sqrt[\heartsuit]{v} = \sqrt[\heartsuit]{seq(u)}$.*

Proof. Via rules (xx, x) for all $x \in \Sigma$ we can obviously go from u to $\mathrm{seq}(u)$. Therefore we have $\sqrt[\heartsuit]{\mathrm{seq}(u)} \subseteq \sqrt[\heartsuit]{u}$. So it remains to show the converse inclusion, and $\sqrt[\heartsuit]{\mathrm{seq}(u)} = \sqrt[\heartsuit]{u}$ will then imply our statement.

Let us suppose there exists a word $z \in \sqrt[\heartsuit]{u}$, which is not contained in $\sqrt[\heartsuit]{\mathrm{seq}(u)}$. As already stated there exists a reduction from u to $\mathrm{seq}(u)$ using only rules (xx, x) for $x \in \Sigma$. Application of these rules preserves the letter sequence of a word. There is also a reduction from u to z via rules from $\heartsuit^{-1}$. Let us look at one specific reduction of this type. As all possible reductions from u to $\mathrm{seq}(u)$ via rules (xx, x) it starts in u, too. At some point –possibly already in the first step– it uses for the first time a rule (ww, w) with $|w| \geq 2$ and results in a word z'. Here this reduction becomes different from the ones to $\mathrm{seq}(u)$.

Because $\mathrm{seq}(w)^2$ is a subsequence of the letter sequence of the word, where this rule is applied, $\mathrm{seq}(w)^2$ is a factor of $\mathrm{seq}(u)$. Thus we can apply a rule $(\mathrm{seq}(w)^2, \mathrm{seq}(w))$ there and obtain the word $\mathrm{seq}(z')$. By Lemma 3.2.11 z' is reducible to $\mathrm{seq}(z')$, and it is still reducible to z. So we can repeat our reasoning. Because the reduction from u to z is finite, this process will terminate and show that there is a word v reachable from both z and $\mathrm{seq}(u)$ via rules from $\heartsuit^{-1}$.

But $z \in \sqrt[\heartsuit]{u}$ is irreducible under this relation, and thus we must have $v = z$. Now $\mathrm{seq}(u)(\heartsuit^{-1})^* z$ shows that $z \in \sqrt[\heartsuit]{\mathrm{seq}(u)}$. Since this contradicts our assumption, there can be no word in $\sqrt[\heartsuit]{u} \setminus \sqrt[\heartsuit]{\mathrm{seq}(u)}$, and this concludes our proof. □

In the proof, the word $\mathrm{seq}(z')$ is obtained by rules, whose left sides are not longer than the one of the simulated rule (ww, w). Therefore the same argumentation works for bounded duplication.

Corollary 3.2.13. *If for two words $u, v \in \Sigma^*$ and an integer k we have $seq(u) = seq(v)$, then also $\sqrt[\heartsuit^{\leq k}]{u} = \sqrt[\heartsuit^{\leq k}]{v} = \sqrt[\heartsuit^{\leq k}]{seq(u)}$.*

Without further considerations, we also obtain a statement about the finiteness of the root of a language.

Corollary 3.2.14. *A language L has finite duplication root, iff $\sqrt[\heartsuit]{seq(L)}$ is finite.*

If a language does not have a finite duplication root, then this root can not be of any given complexity. There is a gap between finite and context-free languages, in which no duplication root can be situated.

Proposition 3.2.15. *If a language has a context-free duplication root, then its duplication root is finite.*

Proof. For infinite regular and context-free languages the pumping lemmata 1.2.6 and 1.2.7 hold. Since a duplication root consists only of square-free words, no such language can fulfill these lemmata. □

Already for the bounded case this does not hold any more. For example, for any $k \geq 1$ we can use a circular square-free word w of length greater than k. Then we have $\sqrt[\heartsuit \leq k]{w^+} = w^+$, and this language is regular.

It is quite clear how the iteration of the union of several singleton sets can generate a regular language with infinite root; for the simplest case of this type consider $\{a, b, c\}^+$. We will now illustrate with an example that there are also regular languages constructed exclusively by concatenation and iteration, which have an infinite duplication root.

Example 3.2.16. From the introductory Example 2.1.2 we can see that the root of the word $u = abcbabcbc$ consists of the two words $u_1 = abc$ and $u_2 = abcbabc$. Let ρ be the morphism, which simply renames letters according to the scheme $a \to b \to c \to a$. Then $\rho(u)$ has the two roots $\rho(u_1)$ and $\rho(u_2)$; similarly, $\rho(\rho(u))$ has the two roots $\rho(\rho(u_1))$ and $\rho(\rho(u_2))$.

We will now use this ambiguity to construct a word w such that $\sqrt[\heartsuit]{w^+}$ is infinite. This word over the four-letter alphabet $\{a, b, c, d\}$ is

$$w = ud\rho(u)d\rho(\rho(u))d = abcbabcbc \cdot d \cdot bcacbcaca \cdot d \cdot cabacabab \cdot d.$$

Thus the duplication root of w contains among others the three words

$$\begin{aligned} w_a = &\ abc \cdot d \cdot bca \cdot d \cdot cabacab \cdot d \\ w_b = &\ abc \cdot d \cdot bcacbca \cdot d \cdot cab \cdot d \\ w_c = &\ abcbabc \cdot d \cdot bca \cdot d \cdot cab \cdot d, \end{aligned}$$

which are square-free. We now need to recall that a morphism h is called square-free, iff $h(v)$ is square-free for all square-free words w. Crochemore has shown that a uniform morphism h is square-free, iff it is square-free for all words of length 3, [20]. Here uniform means that all images of single letters have the same length, which is given in our case.

The morphism we define now is $\varphi(x) := w_x$ for all $x \in \{a, b, c\}$. Thus to establish the square-freeness of φ, we need to check this property for the images of all square-free words up to length 3. These

are

$$\begin{aligned}
\varphi(aba) &= abcdbcadcabacabdabcdbcacbcadcabdabcdbcadcabacabd\\
\varphi(abc) &= abcdbcadcabacabdabcdbcacbcadcabdabcbabcdbcadcabd\\
\varphi(aca) &= abcdbcadcabacabdabcbabcdbcadcabdabcdbcadcabacabd\\
\varphi(acb) &= abcdbcadcabacabdabcbabcdbcadcabdabcdbcacbcadcabd\\
\varphi(bab) &= abcdbcacbcadcabdabcdbcadcabacabdabcdbcacbcadcabd\\
\varphi(bac) &= abcdbcacbcadcabdabcdbcadcabacabdabcbabcdbcadcabd\\
\varphi(bca) &= abcdbcacbcadcabdabcbabcdbcadcabdabcdbcadcabacabd\\
\varphi(bcb) &= abcdbcacbcadcabdabcbabcdbcadcabdabcdbcacbcadcabd\\
\varphi(cac) &= abcbabcdbcadcabdabcdbcadcabacabdabcbabcdbcadcabd\\
\varphi(cab) &= abcbabcdbcadcabdabcdbcadcabacabdabcdbcacbcadcabd\\
\varphi(cba) &= abcbabcdbcadcabdabcdbcacbcadcabdabcdbcadcabacabd\\
\varphi(cbc) &= abcbabcdbcadcabdabcdbcacbcadcabdabcbabcdbcadcabd,
\end{aligned}$$

where, of course, the images of all words shorter than three are contained in them. All the twelve words listed here are indeed square-free as an eager reader can check, and thus φ is square-free.

Now let t be an infinite square-free words over the letters a, b and c. Then $\varphi(\mathrm{pref}(t))$ is an infinite set of square-free words. From the construction of φ we know that for any word z of length i we can reach $\varphi(z)$ from w^i by undoing duplications. Therefore $\varphi(\mathrm{pref}(t)) \subseteq \sqrt[\heartsuit]{w^+}$, whence also the latter set is infinite.

Thus even very simple languages can have rather complicated roots. In the case of uniformly bounded duplication roots, though, the regular languages are closed under the root operation.

Proposition 3.2.17. *If $L \in REG$, then also $\sqrt[\heartsuit^k]{L} \in REG$ for all $k \geq 1$.*

Proof. If a language L is regular, then it can be generated by a regular grammar $G = (N, \Sigma, S, P)$, which has only rules of the forms (A, xB) and (A, x) for non-terminals A and B and $x \in \Sigma$; for simplicity we ignore the possible rule (S, λ) to generate the empty word. From this grammar we construct another one that generates $\sqrt[\heartsuit^k]{L}$.

The new grammar's set of non-terminals is $N' = \{A_w : A \in N \wedge w \in \Sigma^{\leq 2k}\}$. The rule set is derived from P in the following way. The rules from $\{(A_w, B_{wx}) : (A, B) \in P \wedge |w| < 2k-1\}$ go in parallel to those of P, but store the letters generated in the non-terminals' index instead of actually generating them. When the index reaches length $2k$, the oldest letters are finally put out, when new ones come in.

$$\{(A_w, w[1]B_{w[2\ldots 2k]x}) : (A, xB) \in P \wedge |w| = 2k \wedge w[2\ldots 2k]x \in IRR((\heartsuit^k)^{-1})\}.$$

Only if the index would become a square of length k, then half of this

square is deleted, instead of putting anything out.

$$\{(A_w, B_{w[1\ldots k+1]}) : (A, xB) \in P \wedge |w| = 2k \wedge w[2\ldots 2k]x \notin IRR((\heartsuit^k)^{-1})\}.$$

The rules from

$$\{(A_w, B_{w[1\ldots k]}) : (A, xB) \in P \wedge |w| = 2k-1 \wedge w[1\ldots k] = w[k+1\ldots 2k-1]x\}$$

take care of the case that upon filling the index alreadz a k-square is produced. From the terminating rules of P we derive the sets

$$\{(A_w, wx) : (A, x) \in P \wedge wx \text{ is not a } k-\text{square}\}$$

and

$$\{(A_w, w[1\ldots k]) : (A, x) \in P \wedge wx \text{ is a } k-\text{square}\}.$$

These do not conform with our definitnion of regular grammar, because more than one letter is generated in one step; but since allowing this still keeps the language generated regular, we use this simpler way for conciseness.

This new grammar obviously generates the words that also G generates, only leaving out all squares of length $2k$ that occur when going from left to right. The argumentation that showed the confluence of $(\heartsuit^k)^{-1}$ in the proof of Lemma 2.4.4 also shows that in this way all the words in $\sqrt[\heartsuit k]{L}$ are reached. □

The grammar constructed for $\sqrt[\heartsuit k]{L}$ uses a similar idea as the algorithm for deciding the question "$u \in v^{\heartsuit k}$?," which we gave in an earlier article [56]. The effective closure of regular languages under uniformly bounded duplication can be used to decide the problem of the finiteness of the root for the uniformly bounded case.

Corollary 3.2.18. *For regular languages it is decidable, whether their uniformly bounded duplication root is finite.*

Proof. From the proof of Proposition 3.2.17 we see that from a regular grammar for a language L a regular grammar for the language $\sqrt[\heartsuit k]{L}$ can be constructed. This construction method is effective. Since the finiteness problem is decidable for regular languages, it can then also be decided for $\sqrt[\heartsuit k]{L}$. □

3.3 Duplication Codes

A central problem in DNA computation is to find a good encoding, which will facilitate the desired computation but will not exhibit any undesirable behavior. For example, strands can form secondary structures, or strands might simply align in ways not foreseen, if code words are not chosen carefully. A more detailed discussion of this can be found in the current author's work on the use of partial words for the coding problem [52].

As mentioned earlier, duplication is a rather frequent rearrangement in DNA sequences. From this comes the idea to devise a type of code, which is robust against duplications occurring in its words. This is what we will do for the case of uniformly bounded duplication. After collecting some properties of k-dup primitive words, we will provide the definition, characterize the conditions under which infinite codes of this type exist, and finally we will investigate more closely, what kinds of languages are generated by k-dup codes.

3.3.1 k-dup Primitive Words

As the properties of primitive words are frequently used in investigations about conventional codes, also in work about duplication codes a type of primitive words plays an important role. These are the ones primitive or irreducible under the respective relation. In this sense we call *k-dup primitive* all the words in the language $IRR((\heartsuit^k)^{-1})$.

We now proceed to collect some properties of this type of word and of uniformly bounded duplication roots in general. If not stated otherwise, we assume k to be an integer greater than zero in these investigations.

From the fact that the relation $(\heartsuit^k)^{-1}$ is confluent, see Lemma 2.4.4, the following property follows almost immediately.

Lemma 3.3.1. $\sqrt[\heartsuit^k]{uv} = \sqrt[\heartsuit^k]{\sqrt[\heartsuit^k]{u} \cdot \sqrt[\heartsuit^k]{v}}$.

The simpler equation $\sqrt[\heartsuit^k]{uv} = \sqrt[\heartsuit^k]{u} \cdot \sqrt[\heartsuit^k]{v}$ does not hold true in general. A trivial counterexample is $a = \sqrt[\heartsuit^1]{a \cdot a} \neq \sqrt[\heartsuit^1]{a} \cdot \sqrt[\heartsuit^1]{a} = aa$.

Lemma 3.3.2. *If for an k-dup primitive word u there is $\sqrt[\heartsuit^k]{uu} = u$, then also $\sqrt[\heartsuit^k]{u^+} = u$ holds.*

Proof. Let $\sqrt[\heartsuit^k]{uu} = u$. Then $\sqrt[\heartsuit^k]{uuu} = \sqrt[\heartsuit^k]{u \sqrt[\heartsuit^k]{uu}} = \sqrt[\heartsuit^k]{uu} = u$. For larger powers the same principle can be applied, and consequently we obtain by induction that $\sqrt[\heartsuit^k]{u^+} = u$, because other words cannot be roots due to the root's uniqueness. □

Lemma 3.3.3. *If w is a word of length k, then $w^{\heartsuit k} = w^*$.*

Proof. The inclusion $w^* \subseteq w^{\heartsuit k}$ is obvious for $|w| = k$. The other inclusion we show by induction on the number of duplications necessary to obtain a word $u \in w^{\heartsuit k}$ from w. Clearly, by one duplication only ww can be obtained, and it is in w^*. Now suppose that $v \in w^*$ and $u \in v^{1\heartsuit k}$. Then the factor $v[i \dots i+k-1]$ to be duplicated is a conjugate of w. Therefore there is a j such that $v[j+1 \dots i+k-1]v[i \dots j] = w$ and $v[1 \dots j], v[j+1 \dots |v|] \in w^*$. Now $v[i \dots i+k-1]^2 = v[i \dots j]wv[j+1 \dots i+k-1]$, and thus also $u \in w^*$. □

By quite similar reasoning, we obtain another related result, which shows that duplications (and just as well unduplications) preserve periods, which divide their length.

Lemma 3.3.4. *If a word w has a period ℓ, which divides k, then all words in $w^{\heartsuit k}$ and in $\sqrt[\heartsuit k]{w}$ have period ℓ, too.*

Proof. For $\sqrt[\heartsuit k]{w}$ the statement is completely trivial, because removing factors of length ℓ from a word with period ℓ maintains this period. For $w^{\heartsuit k}$ we prove the claim by induction. Of course, w has period ℓ by assumption. Now, suppose some word u has period ℓ, and a factor u of length k is duplicated starting from position i. The resulting word is $w[1 \dots i+k-1]uw[i+k \dots |w|] = w[1 \dots i-1]uuw[i+n \dots |w|]$. Now $w[1 \dots i-1]u$ and $uw[i+n \dots |w|]$ are a prefix and suffix of w, therefore have period ℓ. Since at the point of catenation they agree on the k letters of u to both sides, also the catenation has period ℓ. This, together with the fact that $w^{\heartsuit k} = \{w\}$ for words shorter than k, suffices to prove the claim. □

Now we turn our attention to cases, where a word and some of its powers have the same root. This is not always the case and thus has some implications for the structure of the underlying word.

Lemma 3.3.5. *If $\sqrt[\heartsuit k]{ww} = \sqrt[\heartsuit k]{w}$ for some word w, then $|w|$ is a multiple of k.*

Proof. $\sqrt[\heartsuit k]{ww} = \sqrt[\heartsuit k]{w}$ implies $\sqrt[\heartsuit k]{\sqrt[\heartsuit k]{w}\sqrt[\heartsuit k]{w}} = \sqrt[\heartsuit k]{w}$. This means that one can get from $\sqrt[\heartsuit k]{w}$ to $\sqrt[\heartsuit k]{w}\sqrt[\heartsuit k]{w}$ via duplications of length k. Because every such duplication increases the word's length by k, $|\sqrt[\heartsuit k]{w}|$ must be a multiple of k. Because also w can be reached from $\sqrt[\heartsuit k]{w}$ via duplications of length k, its length must be a multiple of k, too. □

For general powers w^ℓ with $\ell > 2$ this is not true; for example, whenever ℓ is a multiple of k there are trivial counterexamples over a one-letter alphabet. Before we can make a more general statement, we prove an auxiliary lemma.

Lemma 3.3.6. *If $w[1 \dots k]$ is a k-square-free prefix of w, then $w[1 \dots k-n+1]$ is a prefix of $\sqrt[\heartsuit k]{w}$.*

Proof. If $w[1 \dots k]$ is a square-free prefix of w, then the first k-square in w can start at position $k-2n+2$. Suppose that unduplicating this square creates a new one starting at a position closer to the beginning, say m. Then $w[m \dots k-2n]w[k-2n+1 \dots k-n+1]w[k+1 \dots m+n-k-1]$ is a k-square.

This implies that $w[m \dots k-2n]$ is a suffix of $w[k-2n+1 \dots k-n+1]$ and $w[k+1 \dots m+n-k-1]$ is a prefix of it, in fact $w[k-2n+1 \dots k-n+1] = w[m \dots k-2n]w[k+1 \dots m+n-k-1] = w[k-n+2 \dots k+1]$. But this shows that the square starting at position m was already there in the original word, which contradicts our assumptions. □

This bound is tight as shown by the example of $\sqrt[\heartsuit 3]{babaaba} = baba$, where the longest 3-square-free prefix has length 6, and the root has length $4 = 6-3+1$. Of course, the same reasoning applies from the end of the word.

Corollary 3.3.7. *If $w[k \dots |w|]$ is a square-free suffix of w, then $w[k+n-1 \dots |w|]$ is a suffix of $\sqrt[\heartsuit k]{w}$.*

Now we are ready to make a statement about the case where general powers of a word have the same root as the word itself.

Lemma 3.3.8. *If $\sqrt[\heartsuit k]{w^k} = \sqrt[\heartsuit k]{w}$ for some word w and some integer $\ell \geq 2$, then ℓ has a period of w as divisor.*

Proof. First notice that due to the uniqueness of the root, one can undo first all duplication within the different factors w of w^ℓ. By Lemma 3.3.4 this would not change the fact that k divides a period of w. Thus, without restriction of generality we can suppose that w is k-square-free. For words shorter than k, $\sqrt[\heartsuit k]{w^\ell} = \sqrt[\heartsuit k]{w}$ can never hold; for $|w| = k$, obviously always $\sqrt[\heartsuit k]{w^\ell} = \sqrt[\heartsuit k]{w}$ and also k trivially is a period of w. Therefore we can also suppose $|w| > k$ in the following.

Because w is k-square-free any duplication to be undone in w^ℓ must cross a border in between two of the factors w. Further, we suppose that the first k-square in w^ℓ involves at most the last k letters of the first factor w. This means that the entire word w remains

unchanged by unduplicating this square, and thus by Lemma 3.3.6 it remains unchanged in the whole process of arriving at $\sqrt[\heartsuit k]{w}$. If the first square starts earlier, the same reasoning will work from the end of the word, and the last factor w will remain unchanged.

The only case, where neither is true is the occurrence of a third power uuu such that the central u includes the border between the ws. If w has length at least $2k$, then these blocks do not overlap each other, we can just delete the initial k letters of each w and proceed with the resulting word; this preserves squares uu and also preserves any period not longer than k. For shorter w, since $|w| > k$ also $\ell > 2$, and the factors uuu overlap. This implies that the entire word w has period $|u|$. Since $|u| = k$, with Lemma 3.3.4 the initial claim is proven in this case.

Summarizing the reasoning to this point, we can now assume that w is k-square-free, $|w| > n$, and that the first k-square in w^ℓ involves at most the last k letters of the first factor w. This means we can cancel the last copy of each occurrence of this k-square within each of the second to ℓ-th factor w, and arrive at a new word $w(w')^{\ell-1}$. Since we started at the left-most k-square, according to Lemma 3.3.6 this process can be continued, until we arrive at a word w', which is shorter than k – under the assumption $\sqrt[\heartsuit k]{w^\ell} = \sqrt[\heartsuit k]{w}$ this must be possible, because we must be able to arrive at a word of length only $|w|$. If the length of w' is a divisor of k, then $(w')^{\ell-1}$ has a period dividing k, and by Lemma 3.3.4 also $w^{\ell-1}$, which proves our initial claim.

If the length of w' is not a divisor of k, there still must be an k-square in $(w')^{\ell-1}$. Since it has period k and also period $|w'| < k$, by the Theorem of Fine and Wilf 1.1.1 w' has period $gcd(k, |w'|)$, which by definition divides k, and again our initial claim follows with Lemma 3.3.4. □

3.3.2 k-dup Codes

We now proceed to define the central notion of this section, the *k-duplication code*, or short k-dup code. It is closely oriented after the definition of a conventional code, only instead of the catenation of words we investigate the catenation of their k-duplication sets. Recall that a set of words W is a conventional code, if for two integers n, l and words $u_0, \ldots, u_n, v_0, \ldots, v_l \in W$ the equation

$$u_0 u_1 \ldots u_n = v_0 v_1 \ldots v_l$$

implies that $n = l$ and all $u_i = v_i$ for $0 \leq i \leq n$. Analogously, in the sense described above, we call W an *k-dup code*, if

$$u_0^{\heartsuit k} u_1^{\heartsuit k} \dots u_n^{\heartsuit k} \cap v_0^{\heartsuit k} v_1^{\heartsuit k} \dots v_l^{\heartsuit k} \neq \emptyset$$

implies that $n = l$ and $u_i = v_i$ for $0 \leq i \leq n$. From the definition it is clear that every k-dup code is also a code in the conventional sense, because always $w \in w^{\heartsuit k}$. The converse is not true. However, one can easily see that there is a stronger relation to conventional codes.

Proposition 3.3.9. *A set of words W is an k-dup code, if and only if $W^{\heartsuit k}$ is a conventional code.*

Proof. Suppose there is an k-dup code W such that the set $W^{\heartsuit k}$ is not a code. This means there are two integers n, l and words $u_0, \dots, u_n, v_0, \dots, v_l$ from the set $W^{\heartsuit k}$ such that $u_0 u_1 \cdots u_n = v_0 v_1 \cdots v_l$. But this implies that $u_0^{\heartsuit k} u_1^{\heartsuit k} \dots u_n^{\heartsuit k} \cap v_0^{\heartsuit k} v_1^{\heartsuit k} \dots v_l^{\heartsuit k} \neq \emptyset$, and because W is an k-dup code, $n = l$ and all $u_i = v_i$ for $0 \leq i \leq n$.

The converse implication is true by definition. □

This also resolves some possible doubts about the definition of k-dup codes. According to the definition, the sequence of words $u_0 u_1 \dots u_n$ such that $w \in u_0^{\heartsuit k} u_1^{\heartsuit k} \dots u_n^{\heartsuit k}$ must be unique for any word w; this, however, still might admit some ambiguity as to the actual factorization of w. Different combinations of words from the sets $u_i^{\heartsuit k}$ might provide factorizations of w. Proposition 3.3.9 shows that this is impossible, because if $W^{\heartsuit k}$ is a code, then all factorizations over this set are by definition unique.

The conventional code thus associated to an k-dup code we will call its associated code. It is worth noting that it is never a prefix or suffix code, if the k-dup code is non-trivial, i.e. it contains at least one word longer than k. The definition of k-dup code can be given also in a stronger form, considering two lists of equal length only. The necessary property is stated in the following proposition.

Proposition 3.3.10. *For any set of words W that is not an k-dup code, there are a natural number m and words $u_0, u_1, \dots u_m$ and $v_0, v_1, \dots v_m$ all from W such that $u_0^{\heartsuit k} u_1^{\heartsuit k} \dots u_m^{\heartsuit k} \cap v_0^{\heartsuit k} v_1^{\heartsuit k} \dots v_m^{\heartsuit k} \neq \emptyset$ and not $u_i = v_i$ for all $i \leq m$.*

Proof. For any set of words W, which is not an k-dup code, there are by definition natural numbers k and l, and there are words $u_0, u_1, \dots u_n$ and $v_0, v_1, \dots v_l$ all from W such that

$$u_0^{\heartsuit k} u_1^{\heartsuit k} \dots u_n^{\heartsuit k} \cap v_0^{\heartsuit k} v_1^{\heartsuit k} \dots v_l^{\heartsuit k} \neq \emptyset$$

and not $u_i = v_i$ for all $i \leq \min\{k, l\}$. Now we set $m := n + l$ and looking at the set

$$u_0^{\heartsuit k} u_1^{\heartsuit k} \dots u_n^{\heartsuit k} v_0^{\heartsuit k} v_1^{\heartsuit k} \dots v_l^{\heartsuit k} \cap v_0^{\heartsuit k} v_1^{\heartsuit k} \dots v_l^{\heartsuit k} u_0^{\heartsuit k} u_1^{\heartsuit k} \dots u_n^{\heartsuit k},$$

which is also non-empty, we see that the proposition's statement is true. □

Example 3.3.11. As a first example of an k-dup code, we look at the set $\{aba\}$, which is a 2-duplication code. Clearly $(aba)^{\heartsuit 2} = a(ba)^*$. Thus any catenation of words from $(aba)^{\heartsuit 2}$ has two consecutive a exactly at the borders between the catenated words; this provides the unique factorization of these catenations.

An k-dup code W is *maximal*, if for all words w from $\Sigma^+ \setminus W$ the set $W \cup \{w\}$ is not an k-dup code. The 2-duplication code $\{aba\}$ from Example 3.3.11 is not maximal, as we can add, for example, the word $bbabb$ to it, and the result is still a 2-duplication code.

Example 3.3.12. In a trivial manner, $\{a, b\}$ is a maximal k-dup code over two letters for any $k > 1$, because no duplication is involved and $\{a, b\}$ is a maximal conventional code generating the entire set Σ^*. In the same way any finite maximal conventional code with its longest word of length n is a maximal k-dup code for any $k > n$ (see also Propostion 3.3.14 further down).

In the case of conventional codes, any code can be made maximal by subsequent, possibly infinite addition of more words [10]. The same reasoning works also in the case of duplication codes.

Proposition 3.3.13. *Every k-dup code over an alphabet Σ is contained in a maximal k-dup code over Σ.*

Proof. We consider the partial order on all k-dup codes created by set inclusion. By Zorn's Lemma suffices to show that any chain in this partial order has a least upper bound, which is again an k-dup code. Clearly, the union of all the chain's elements is its least upper bound. Let us call it W.

If W were not an k-dup code, then there would be a positive integer k and words $w_0, w_1, \dots, w_n, w'_0, w'_1, \dots, w'_n$ from W, such that

$$w_0^{\heartsuit k} w_1^{\heartsuit k} \dots w_n^{\heartsuit k} \cap w_0'^{\heartsuit k} w_1'^{\heartsuit k} \dots w_n'^{\heartsuit k} \neq \emptyset.$$

Since W is the union of all elements of the chain, there is a maximal element in this chain containing all these w_i and w'_i. The chain, how-

ever consists only of k-dup code, and thus this situation is impossible, also W is an k-dup code. □

We now compile some properties of words contained in k-dup codes. The first one is obvious, because for words shorter than k the duplication language generated contains only the original words itself.

Proposition 3.3.14. *A code of words all shorter than k is an k-dup code.*

Further, for every word w of length k, we have $w \cdot w \in w^{\heartsuit k}$. Therefore containment of w in an k-dup code would result in two distinct factorizations of ww. Thus we can state a condition that necessarily makes a set of words not an k-dup code.

Proposition 3.3.15. *An k-dup code cannot contain any word of length k.*

Similarly, for every word $w = a^k a^m$ with $m > 0$, there exists an integer $n > 1$, for example $n = 1 + lcm(m, k)$, such that $w^n \in w^{\heartsuit k}$.

Proposition 3.3.16. *An k-dup code cannot contain any word longer than k from the set a^+ for a letter a.*

So the only words really interesting are the ones of lengths greater than that of the duplications and with at least two letters. Of course, the duplication roots of the words involved play an important role.

Proposition 3.3.17. *An k-dup code cannot contain two words with the same k-duplication root.*

Proof. This is a direct consequence of the confluence of uniformly bounded duplication, which was stated in Lemma 2.4.2. □

After many conditions for a set of words not to be a code, we now state a property that makes a set of two words an k-dup code in a non-trivial way.

Proposition 3.3.18. *If uu, vv, uv and vu are k-square-free and longer than k, then $\{u, v\}$ is a k-dup code.*

Proof. If uu, vv, uv and vu are all k-square-free and longer than k, then all words in $\{u, v\}^*$ are k-dup-primitive. Thus every such word $w_0 w_1 \dots w_n$ is the unique root of any word in $w_0^{\heartsuit k} w_1^{\heartsuit k} \dots w_n^{\heartsuit k}$, if all w_i are from the set $\{u, v\}$. Suppose now that some word has two such factorizations into words from $\{u, v\}^{\heartsuit k}$. If they are distinct, then

they result in two distinct duplication roots as just exposed. This is in contradiction to the uniqueness of the root. Therefore no word can have two distinct factorizations of this type, and $\{u, v\}$ is an k-dup code. □

In a straight-forward manner, the argumentation from the proof of Proposition 3.3.18 can be generalized to any number of words. We state this only for a finite number, though also an infinite set of words can be treated the same way.

Corollary 3.3.19. *Let $W = \{w_0, w_1, \ldots w_k\}$ be a set of words all longer than k and such that all words in W^2 are k-square-free. Then W is an k-dup code.*

3.3.3 Infinite Duplication Codes

Of course, there are infinite conventional codes. After Proposition 3.3.9, however, it is not self-evident that also infinite duplication codes exist. As we will see, this depends on the size of the alphabet and on the length of the duplications. We start with a negative result, i.e. with a case where no infinite dup code exists.

Proposition 3.3.20. *There is no infinite* 1*-dup code over a two-letter alphabet.*

Proof. Let W be a 1-dup code over the alphabet $\{a, b\}$. Suppose that W contains a word w that starts with a and ends with b. If there is another word u from W with the same properties, then let n be the number of changes from a to b in u and let l be the same number for w.

We now start from the word $(ab)^{(n \cdot l)}$ and duplicate the initial a so often, that the initial block of a is as long as the longer one from u and w. Then the same is done for the first block of b etc. comparing the length of these blocks in the entire word generated to the respective lengths in u^l and w^n. Clearly the resulting word is in both $w^{\heartsuit 1}$ and $u^{\heartsuit 1}$. Thus W is a code only if $u = w$, and any 1-dup code can contain at most one word starting with a and ending with b.

For words starting with b and ending with a the argumentation is the same, for words starting and ending with the same letter (a or b), a very similar line of thought works. As there are only four possibilities of different first/last letter combinations, and for every one at most one word can be in W, no 1-dup code can be infinite. □

From the proof we immediately see an even tighter bound for the size of a 1-dup code. Namely that over a two-letter alphabet there is no 1-dup code consisting of more than four words, because there are only four possible combinations of first and last letter. This, however, is not yet optimal. In fact, the maximum number of words in a 1-dup code is only one – considerations only slightly more intricate than above show this.

Corollary 3.3.21. *Over a two-letter alphabet there is no* 1*-dup code consisting of more than one word.*

Proof. An argumentation analogous to that of the proof of Proposition 3.3.20 works for any two words having a change of letter inside. Only, if they do not start and end with the same letters the construction gets slightly more intricate, some “padding" at the start and end may be necessary.

This leaves only words over just one letter as candidates for a second word in a 1-dup code. But by Proposition 3.3.15 neither a nor b are possible, longer words are excluded by Proposition 3.3.16. □

The situation changes, when we increase the size of the alphabet. Already three letters suffice to construct an infinite code.

Proposition 3.3.22. *Over a three-letter alphabet, there exist infinite* 1*-dup codes.*

We prove this by providing an example for such a code.

Example 3.3.23. The language $W = (ab)^+c$ is an infinite 1-dup code. First off, we note the fact that the duplication of a single letter can never change the number of letter-changes (from a to b, a to c etc.) in a given word. From this, a 1-dup code factorization for every word w from $W^{\heartsuit k}$ can be found by splitting it after every block of c. Further, the number of changes from a to b uniquely determines the word from W, from which the respective factor originated.

Proposition 3.3.24. *There exist infinite* k*-dup codes for* $k \geq 2$*.*

Again, we provide examples for such codes. The first one uses three letters, the second one only two. Thus, for duplications longer than one there is not the same distinction between two- and three-letter alphabets as exhibited by Propositions 3.3.20 and 3.3.22.

Example 3.3.25. Let W be the set of k-dup-primitive words over the alphabet $\{a, b\}$, which are longer than k. Then the set $U := c \cdot W \cdot c$ is an infinite k-dup code. It is clearly infinite. Further, all the sets

$u^{\heartsuit k}$ for a u from U are disjoint, because the root is unique. Finally, all words begin and end with a c, and thus their catenation has two consecutive c at the border. Because duplications of length greater than one cannot create two consecutive c starting from a word in U, the factorizations are unique in a similar manner as above in Example 3.3.11, and U is an k-dup code.

Example 3.3.26. The language $W = a(abb)^+$ is a 2-dup code. To see this, consider the effects of possible 2-duplications on a word from W: $aa \to aaaa$, $ab \to abab$, and $bb \to bbbb$. All of them preserve the number of blocks of the same letter of length greater than one in the original word – for this, one needs to look also at the letters immediately preceding and following the duplicated factor.

Because in W^+ all W-factors of a word start with aa and this is the only occurrence of aa, the W-factorization is unique. Further, for every positive integer the word from W having this number of bb-blocks is unique. Thus it is easy to reconstruct from any word in $W^{\heartsuit 2}$ its unique 2-dup factorization by separating the word at the beginning of every (maximal) block of a, which is longer than one.

Summarizing the results of this section and adding a few trivial considerations for one-letter alphabets, we obtain the following theorem, which fully characterizes the conditions under which infinite duplication codes exist.

Theorem 3.3.27. *There exist infinite k-dup codes over an n-letter alphabet, if and only if $k, n \geq 2$ or if $k = 1$ and $n \geq 3$.*

Since all the examples for infinite k-dup codes provided in this section have been regular, it is worth stating that this is not necessarily so. We give an example for a non-context-free k-dup code over four letters. This leaves open the question, whether such codes exist also over two or three letters. Again, the answer might also be parameterized by the duplications' length.

Proposition 3.3.28. *For any $k \geq 1$, there exists an infinite k-dup code which is not regular.*

Proof. Let T_3 be the infinite set of all square-free words over a three-letter alphabet, which is known to be a non-context-free language [79]. However, the still non-context-free language $W = \{d\}T_3\{d\}$ is an k-dup code for every $n \geq 1$. □

3.3.4 Languages Generated by Duplication Codes

An interesting concept in relation with codes is the denseness of the languages they generate. Informally speaking, denseness means that any word appears as a factor of some word in the generated language. Recall that a language $L \subset \Sigma^*$ is called *dense*, if for all words $w \in \Sigma^*$ we have $\Sigma^* w \Sigma^* \cap L \neq \emptyset$.

The constructions used to prove that the languages generated from one word by general duplication [92] and by (non-uniformly) bounded duplication [56] show that in most cases those languages are also dense; for example the occurrence of a factor abc sufficed to guarantee this over the corresponding three-letter alphabet. For uniformly bounded duplications, however, these construction techniques cannot be applied.

Proposition 3.3.29. *There exists an infinite k-dup code W, such that $(W^{\heartsuit k})^*$ is not dense.*

Proof. As shown in Example 3.3.25, the language $W = cUc \subseteq \{a, b, c\}^*$ is an infinite k-dup code, where U is the set of k-dup primitive words over $\{a, b\}$. Following the argumentation showing that this is so, we also see that words from $(W^{\heartsuit k})^*$ do not contain any factor ccc. Thus $(W^{\heartsuit k})^*$ is not dense. □

On the other hand, density of W, or even of W^* guarantees the density of $(W^{\heartsuit k})^*$. These observations raise the question, whether there is an k-dup code W, such that W^* is not dense, but its associated language $(W^{\heartsuit k})^*$ is dense. If we require only W not to be dense, then there are trivial solutions like Σ itself, which is an k-dup code for any $n > 1$ and generates entire Σ^*.

The most prominent result concerning conventional codes in this respect is that density is given if and only if a code is maximal [10]. We now present two somewhat contrasting results, the first showing that there are always infinitely many k-dup primitive words not in the root of the language; then we will see that this still allows the languages generated to be dense.

Proposition 3.3.30. *For all k-dup codes W, the set $\heartsuit_k \setminus \sqrt[\heartsuit k]{(W^{\heartsuit k})^*}$ is infinite.*

Proof. First we notice that the set $\heartsuit_k$ of all k-primitive words is always infinite. Thus, if $\sqrt[\heartsuit k]{(W^{\heartsuit 1})^*}$ is not infinite, the proposition is true. In the contrary case, $\sqrt[\heartsuit k]{(W^{\heartsuit 1})^*}$ contains an infinite set U, which consists of words longer than $2n + 2$. For such a word u we now look

at the words $v = u[1 \dots \left\lfloor \frac{|u|}{2} \right\rfloor]$ and $w = u[\left\lfloor \frac{|u|}{2} \right\rfloor + 1 \dots |u|]$, which are both k-dup primitive, just as u.

If there existed words $v_1, w_1 \in (W^{\heartsuit k})^*$ such that $\sqrt[\heartsuit k]{v_1} = v$ and $\sqrt[\heartsuit k]{w_1} = w$, then also $v_1 w_1$ would be in $(W^{\heartsuit k})^*$. Then by the confluence of uniformly bounded duplication, see Lemma 2.4.2, there would also exist words $v_2, w_2 \in \Sigma^*$ such that $v_2 \in v^{\heartsuit k} \cap v_1^{\heartsuit k}$ and $w_2 \in w^{\heartsuit k} \cap w_1^{\heartsuit k}$. But this implies that $v^{\heartsuit k} w^{\heartsuit k} \cap u^{\heartsuit k} \neq \emptyset$. Therefore for at least one of v and w no word can be in W that has this root, otherwise W would not be an k-dup code. Neither can this word be composed by shorter ones, the same argumentation would apply. This provides us with one word in the set $\heartsuit_k \setminus \sqrt[\heartsuit k]{(W^{\heartsuit k})^*}$.

Thus it remains to construct an infinite sequence of such words providing us with pairwise different words from $\heartsuit_k \setminus \sqrt[\heartsuit k]{(W^{\heartsuit k})^*}$. For an infinite k-dup-code W, already $\sqrt[\heartsuit k]{W}$ is infinite by Proposition 3.3.17. Thus we can find an infinite sequence $(u_i)_{i \in \mathbb{N}}$ of words in $\sqrt[\heartsuit k]{W}$ such that always $|u_i| > |u_{i-1}| + 2$, which satisfies the requirements stated.

For a finite set W, we pick a word $w \in W$, which has no period that divides k. Then by Lemma 3.3.8 the sequence of $u_i := w^{2^n}$ works. If all words in W have periods dividing k, then we take $u_i := (vv)^{2^n}$ for such a word $v \in W$. Now, if vv still had a period dividing k, then v^{n+1} could be reduced via k-unduplications to v, and consequently v cannot be in an k-dup code. Therefore vv has no period dividing k, and can be used just as w above. □

Proposition 3.3.31. *There exists an infinite* 1*-dup code* W *over any alphabet* Σ *with three or more letters, such that* $(W^{\heartsuit 1})^*$ *is dense.*

Proof. Recall that $\heartsuit_1$ is the language of all 1-square-free words. Now we choose an arbitrary non-empty, unbordered word w from $\heartsuit_1$ with $w[1] = b$, $w[|w|] = a$ and $|w| > 1$. We set $\heartsuit_1' := \heartsuit_1 \setminus (\Sigma^* w \Sigma^* \cup a\Sigma^* b)$. Note that $\heartsuit_1'$ is only infinite over an alphabet with three or more letters.

Then $W := \heartsuit_1' \cdot w$ is a 1-dup code, because $\heartsuit_1'^{\heartsuit 1}$ and $w^{\heartsuit 1}$ are disjoint. Thus any word from $(W^{\heartsuit 1})^*$ is uniquely factorized into words from $W^{\heartsuit 1}$ by separating them after any occurrence of a factor from $w^{\heartsuit 1}$. Note that different occurrences of w cannot overlap, because the word is unbordered.

It remains to show that $(W^{\heartsuit 1})^*$ is dense. Intuitively speaking, $\heartsuit_1$ is the language one obtains from Σ^* by condensing all blocks of the same letter within a word to length one. From these words any other word can be obtained by doing the appropriate 1-duplications. Therefore it suffices to show that for all $u \in \heartsuit_1$ we have $\Sigma^* u \Sigma^* \cap W^* \neq \emptyset$.

For words u not containing a factor w this is true, because they are already contained in W.

We first look at words not starting with a. For such a word u not containing one factor w, there is a factorization $u = u_1 w u_2$. Here it is crucial to note that $u_2[1] \neq a$, otherwise u would contain a 1-square, the same for $u_1[|u_1|] \neq b$. But now we have $u_1 w, u_2 w \in W$, and thus $u_1 w, u_2 w \in W^*$ with a factor u. For words with more occurrences of factors w analogous factorizations can be found.

Thus, all 1-square-free words not starting with a are prefixes of words in W^*. With the observation that words av are factors of the corresponding wvw we conclude the proof. □

Another interesting questions is, whether the step from W to $(W^{\heartsuit k})^*$ increases the complexity of the language. For regular languages no increase in complexity can be observed.

Theorem 3.3.32. *The language $W^{\heartsuit k}$ is regular for every regular language W and $n \geq 1$.*

Proof. Let $W \subseteq \Sigma^*$ be a regular language and let $W = W_1 \cup W_2$, where $W_1 = \{x \in W \mid |x| \leq k\}$, and $W_2 = W \setminus W_1$. Obviously, $W^{\heartsuit k} = W_2^{\heartsuit k} \cup W_1$. Assume that W_2 is recognized by the DFA (deterministic finite automaton) $A = (Q, \Sigma, \delta, q_0, F)$. We construct the DFA

$$A' = (Q', \Sigma, \delta', \langle q_0, \lambda \rangle, F'),$$

where

$$\begin{aligned} Q' &= \{\langle q, x \rangle \mid q \in Q, x \in \Sigma^*, |x| \leq k\} \\ F' &= \{\langle q, x \rangle \mid q \in F, x \in \Sigma^*, |x| = k\} \end{aligned}$$

and the transition mapping δ' is defined as follows:

$$\delta'(\langle q, x \rangle, a) = \begin{cases} \langle \delta(q, a), xa \rangle, & \text{if } |x| < k \\ \langle \delta(q, a), Suf_k(xa) \rangle, & \text{if } |x| = k. \end{cases}$$

Here $Suf_k(z)$ denotes the suffix of z of length k. Clearly, the automaton A' recognizes the same language as A does, namely W_2.

We now recall a result from earlier work [56], which is very useful for the last part of our proof. For two words x, y over an alphabet Σ such that $y \in x^{\heartsuit k}$ and p a positive integer, we write $x \bowtie_{(p,n)} y$ if $x = tuv$, $|t| = p - 1$, $|u| = n$ and $y = tuuv$. Then the following result is known:

Proposition 3.3.33 ([56]). *If* $x = x_1 \bowtie_{(p_1,n)} x_2 \bowtie_{(p_2,n)} x_3 \bowtie_{(p_3,n)} \ldots x_r \bowtie_{(p_r,n)} w$ *for some* p_i, $1 \leq i \leq r$, *then* $x = y_1 \bowtie_{(q_1,n)} y_2 \bowtie_{(q_2,n)} y_3 \bowtie_{(q_3,n)} \ldots y_r \bowtie_{(q_r,n)} w$ *holds for some* $q_1 \leq q_2 \leq \ldots \leq q_r$. *Furthermore, for each* $i \in [r-1]$, *either* $q_i = q_{i+1}$ *or* $q_{i+1} - q_i > n$ *holds.*

By this, if one adds a loop labeled by x to any state $\langle q, x\rangle \in Q'$ with $|x| = k$, one gets an automaton (not necessarily deterministic) which accepts the language $W_2^{\heartsuit k}$ and we are done. □

Since the family of regular language is closed under Kleene closure, we obtain as an immediate corollary a statement about the languages generated by regular k-dup codes.

Corollary 3.3.34. *The language generated by a regular* k*-dup code* W, $k \geq 1$, *is still regular.*

3.4 Closure of Language Classes under Duplication

In some of the earlier work on special cases of idempotency relations, rather than their effect on single words their effect on entire languages was studied. Namely Propositions 2.2.7 and 2.2.8 in Section 2.2.4, which deal with insertion and deletion, in their original statements establish closure properties of the class of regular languages; i.e. the rewrite relations $\bowtie_0^1$ and $\bowtie_1^0$ were shown to preserve regularity. We now investigate the same topic for the case of duplication. The results will concern mainly bounded duplication.

3.4.1 Closure of Regular Languages

We start out with the closure of regular languages. Here the size of the alphabet will play an important role, and first we treat the three-letter case, where closure is not given in most of the cases. All results for this alphabet size also carry over to bigger alphabets.

It is known that the 4-bounded duplication closure of the word abc is not regular [58]. As one can see from the original proof, duplications longer than 4 do not affect the construction used, and therefore the result extends to longer bounds. Thus the class of regular languages is not closed under n-bounded duplication for $n \geq 4$, since singular sets are of course regular.

Proposition 3.4.1. *For $n \geq 4$ the class of regular languages is not closed under n-bounded duplication.*

On the other hand, it is trivial to see that 1-bounded duplication preserves regularity: the only possible change in the original word is that every letter a can be blown up to any word from a^+. We now take a look at the two cases in between, that is length-bounds of 2 and 3.

We now fix some notation, which will be convenient in the proof that follows. For a right-syntactic congruence $\sim_L$ we denote the set of all possible right contexts of a word u by $\sim_L(u) := \{w : uw \in L\}$. By $[u]_{\sim_L}$ we denote the congruence class of u; notice that for all $u_1, u_2 \in [u]_{\sim_L}$ we have $\sim_L (u_1) = \sim_L (u_2)$.

Proposition 3.4.2. *The class of regular languages is closed under 2-bounded duplication.*

Proof. Let L be a regular language, and $\sim_L$ the corresponding right-syntactic congruence. The right-syntactic congruence $\sim_{L^{\heartsuit \leq 2}}$ we will

denote more simply by $\sim$. We will show that the number of congruence classes of $\sim$ is bounded by a function of the number of congruence classes of $\sim_L$.

First notice that always $(\sim_L(u))^{\heartsuit\leq 2} \subseteq \sim(u)$, i.e. if v is a possible right context of u in L, then all words in $v^{\heartsuit\leq 2}$ are possible right contexts of u in $L^{\heartsuit\leq 2}$. If the two sets are not equal, this can be caused only by some duplication transgressing the border between u and v. Duplications of length one cannot do this, thus the only possibility is one of length two affecting the last letter of u and the first letter of v.

If the two letters are the same, say a, then the result will be a^4, which could have been obtained also by duplicating twice the a in v, so the result is in $v^{\heartsuit\leq 2}$. If the two letters are distinct, say a and b, then the result of the duplication will be $abab$. If the following letter in v is an a, then we could have obtained the same by duplicating the prefix ba of v, so the result is in $v^{\heartsuit\leq 2}$.

Otherwise the result will be $ababc$ for some letter c different from a. The resulting right context is not in $v^{\heartsuit\leq 2}$, so in this case a new congruence class for u is created in $\sim$. More duplications on the right side will not lead to new classes, because now we have bab following the final a of u. The number of such constellations of two different letters at the border with a different one from the first one following is bounded by the total number of letters in the alphabet. Thus every congruence class of $\sim_L$ results in a finite number of congruence classes for $\sim$, except possibly for the one of words not being a prefix of a word in L.

Therefore it remains to show that the u, which are not prefixes of a word in L but are prefixes of a word in $L^{\heartsuit\leq 2}$, do not generate an infinite number of new congruence classes. So let $uv \in L^{\heartsuit\leq 2}$. If there exists $u'v'$ that $u \in u'^{\heartsuit\leq 2}$ and $v \in v'^{\heartsuit\leq 2}$, then we are done. Otherwise in the generation of uv from $u'v'$ there is a duplication transgressing the border between the two words.

Similarly as above, this is interesting only in the configuration $ca|b$, where $|$ denotes the border between u' and v' (or rather between the two intermediate words generated from them). The result of this duplication is $caba|b$. Let us call the word on the left u''. No further duplications transgressing the border can be necessary, since $(caba)^{\heartsuit\leq 2}b^{\heartsuit\leq 2} = (cabab)^{\heartsuit\leq 2}$. Thus for all words u here we have either $[u]_\sim = [u']_\sim$ or $[u]_\sim = [u'']_\sim$. Thus also here the increase of the index of $\sim$ compared to $\sim_L$ preserves finiteness, and thus the resulting language is regular by Theorem 1.2.5, if the original language was regular. □

It appears possible to extend this proof technique to 3-bounded duplication under use of the fact that over two-letters the longest square-free word has length 3. While we leave this case open here, over an alphabet of only two letters things are not as complicated. To see this we first state a result that relates bounded and unbounded duplication. This will then allow us to state the closure of regular languages under these variants of duplication.

For the remainder of this section, $\rightarrow$ will denote the derivation relation of the string-rewriting system $R = \{a \rightarrow aa, b \rightarrow bb, ab \rightarrow abab, ba \rightarrow baba\}$, which generates the language $w^{\heartsuit \leq 2}$ for any word $w \in \{a, b\}$.

Lemma 3.4.3. *For every word $u \in \{a, b\}^*$ we have $ab \xrightarrow{*} abubab$, $ab \xrightarrow{*} abuaab$, and $ab \xrightarrow{*} abuab$.*

Proof. We prove this statement by induction on the length of u. For $|u| = 0$ the three derivations

$$\begin{aligned} ab &\xrightarrow{ab \rightarrow abab} abab \xrightarrow{b \rightarrow bb} abbab &&= abubab \\ ab &\xrightarrow{ab \rightarrow abab} abab \xrightarrow{a \rightarrow aa} abaab &&= abuaab \\ ab &\xrightarrow{ab \rightarrow abab} abab &&= abuab \end{aligned}$$

show us that the lemma holds. So let us suppose it holds for all words, which are shorter than a number n. Any word u of length n has a factorization either as va or vb for a word v of length $n-1$. For this word v the Lemma holds by our assumption. But then for $u = va$ the derivations

$$\begin{aligned} ab &\xrightarrow{*} abvab \xrightarrow{ab \rightarrow abab} abvabab &&= abubab \\ ab &\xrightarrow{*} abvaab \xrightarrow{a \rightarrow aa} abvaaab &&= abuaab \\ ab &\xrightarrow{*} abvaab &&= abuab \end{aligned}$$

and for $u = vb$ the derivations

$$\begin{aligned} ab &\xrightarrow{*} abvbab \xrightarrow{b \rightarrow bb} abvbbab &&= abubab \\ ab &\xrightarrow{*} abvbab \xrightarrow{a \rightarrow aa} abvbaab &&= abuaab \\ ab &\xrightarrow{*} abvbab &&= abuab \end{aligned}$$

show us that the lemma holds also for u and thus for all words. □

Proposition 3.4.4. *Over an alphabet of two letters we have $w^{\heartsuit \leq n} = w^{\heartsuit \leq 2}$ and consequently $w^{\heartsuit} = w^{\heartsuit \leq 2}$ for all words w and for $n \geq 2$.*

Proof. From Lemma 3.4.3 we know that $ab \xrightarrow{*} abuab$ holds for every word u, and applying this to the initial factor ab in abu we obtain $abu \xrightarrow{*} abuabu$. Just interchanging the letters a and b everything still is valid, and thus we see that also $bau \xrightarrow{*} baubau$ holds.

Now we prove that $aau \xrightarrow{*} aauaau$. If $u \in a^*$, then the statement is obviously true. Otherwise there is at least one b in u, and therefore u can be factorized as $u = a^m bv$ for some word v and an integer $m \geq 0$. Now the derivation

$$aau = aa^m(ab)v \overset{\text{Lemma 3.4.3}}{\longrightarrow} aa^m abvabv \xrightarrow{*} aaa^m bvaaa^m bv = aauaau$$

shows that the statement above holds. Interchanging the letters again provides us with the dual statement $bbu \xrightarrow{*} bbubbu$.

Because any word z longer than 1 has to start with either ab, ba, aa, or bb, this shows that we can always obtain by duplications of length at most 2 the word zz from z and thus $w^{\heartsuit \leq n} \subseteq w^{\heartsuit \leq 2}$. On the other hand, every duplication relation $\heartsuit^{\leq n}$ for $n \geq 2$ includes the relation $\heartsuit^{\leq 2}$ and so does $\heartsuit$. This suffices to prove that for all $n > 1$ we have $w^{\heartsuit \leq n} = w^{\heartsuit \leq 2}$, and $w^{\heartsuit} = w^{\heartsuit \leq 2}$ immediately follows from this, because in any derivation the length of duplications used is bounded. □

Combining the results of this section we are now able to state the closure of regular languages under duplication.

Proposition 3.4.5. *The class of regular languages over two-letter alphabet is closed under n-bounded duplication and under general duplication.*

Proof. Proposition 3.4.2 states that regular languages are closed under 2-bounded duplication over any alphabet, and from Proposition 3.4.4 we see that in the two-letter case for any $n > 1$ the n-bounded and general duplication operations are equivalent to the 2-bounded one. □

3.4.2 Closure of Context-Free Languages

When we speak about context-free languages, there is no difference between alphabets of size 2 and 3. It is already known that languages $w^{\heartsuit \leq n}$ are always context-free [58]. By further refining the push-down automaton used in that proof, we can establish the closure of context-free languages under bounded duplication.

Proposition 3.4.6. *The class of context-free languages is closed under bounded duplication.*

Proof. We will show this by constructing a Push-Down Automaton in a way rather analogous to the one used in earlier work for the bounded duplication closure of a single word [58]. There the PDA reduces the results of duplications uu to their origin u and matches the reduced string against the original word. Here, we also have to simulate a second PDA accepting the context-free input language. This can be done, because of the two components reducing duplications and accepting the original language, the latter one does not need to access the stack ever, while the first one is working. With this sketch of the proof idea we now proceed to the technical details.

We start out from a PDA M, which accepts the language L. Let the PDA be $M = [Q, \Sigma, \Gamma, \varphi, q_o, \bot]$, where Q is the set of states, Σ the tape alphabet, and Γ the stack alphabet. $\varphi : Q \times (\Sigma \cup \{\lambda\}) \times \Gamma \to Q \times \Gamma^*$ is the state transition function; i.e. we allow transitions without reading input and we always take the topmost symbol off the stack replacing it by an arbitrary number of stack symbols. q_0 is the start state, and $\bot$ marks the stack's bottom. The acceptance mode does not really need to be specified, since any common acceptance condition will carry over to the new PDA.

We now define the PDA A, which accepts $L^{\heartsuit \leq n}$. The state set is $S := Q \times (\underline{\Sigma} \cup \Sigma)^{\leq n} \times \Sigma^{\leq n}$, where $\underline{\Sigma} := \{\underline{a} : a \in \Sigma\}$ is a marked copy of the tape alphabet. States $s \in S$ we will denote in the way $s = q|_v^u$, where $q \in Q$, $u \in (\underline{\Sigma} \cup \Sigma)^{\leq n}$ is called the *match*, and $v \in \Sigma^{\leq n}$ the *memory*; then $q_0|_\lambda^\lambda$ is the start state of S. The stack alphabet is $\Gamma' := \Gamma \cup (\underline{\Sigma} \cup \Sigma)^{\leq n}$. The tape alphabet Σ and bottom-of-stack marker $\bot$ are as for M. What remains to be defined is the transition function δ. We first define the part

$$\delta(q|_\lambda^\lambda, x, \gamma) := (q'|_\lambda^\lambda, \alpha) \text{ where } \varphi(q, x, \gamma) = (q', \alpha) \tag{3.2}$$

for $x \in \Sigma \cup \{\lambda\}$, $\gamma \in \Gamma$, and $\alpha \in \Gamma^*$. We see that when guess and memory are empty, A works just as M; we will see that these are the only transitions changing the component from Q of A's states. Thus the simulation of M and the undoing of duplications, which uses match and memory leaving the component from Q unchanged, are done more or less independently. The next kind of transition makes a guess that the following letters on the input tape are the result of a duplication. Transitions

$$\delta(q|_v^u, x, \gamma) := (q|_v^w, u\gamma)$$

are defined for any words $u \in (\Sigma \cup \underline{\Sigma})^{\leq n}$ and $v, w \in \Sigma^{\leq n}$. Whatever is in the match is put on the stack to continue processing later. Note that the word u is put on the stack as a single symbol.

Next A checks whether the input continues with ww. This is done by matching the guess twice against the input, which is read, the first time underlining it in the guess, then undoing this. When both are matched, our PDA should continue as if there was one occurrence of w left on the input tape. However, both are already read. Thus we put w into the memory and read from there as if it was the input tape. Since in this construction the contents of the memory are thought to be situated in front of the input tape contents, nothing is ever read from the input tape, while the memory is not empty. For both situations all transitions are defined in parallel.

The variables used in the definition are quantified as follows: $q \in Q$, $x \in \Sigma$, $u, v, z \in \Sigma^*$, $\gamma \in \Gamma'$, $\beta \in \Gamma$, and $w \in \underline{\Sigma}^* \cdot \Sigma^* \cup \Sigma^* \cdot \underline{\Sigma}^*$ with $|w| \leq n$. Further, all catenations of words and letters are supposed to be no longer than n, and underlining a word from Σ^* shall signify the corresponding word over $\underline{\Sigma}$ obtained by underlining all the individual letters.

$$\begin{array}{lll}
\delta(q|_{\lambda}^{\underline{zxu}}, x, \gamma) := (q|_{\lambda}^{\underline{zxu}}, \gamma) & \text{and} & \delta(q|_{\underline{xv}}^{\underline{zxu}}, \lambda, \gamma) := (q|_{v}^{\underline{zxu}}, \gamma) \\
\delta(q|_{\lambda}^{\underline{xu}}, x, \gamma) := (q|_{\lambda}^{\underline{xu}}, \gamma) & \text{and} & \delta(q|_{\underline{xv}}^{\underline{xu}}, \lambda, \gamma) := (q|_{v}^{\underline{xu}}, \gamma) \\
\delta(q|_{\lambda}^{\underline{zxu}}, x, \gamma) := (q|_{\lambda}^{\underline{zxu}}, \gamma) & \text{and} & \delta(q|_{\underline{xv}}^{\underline{zxu}}, \lambda, \gamma) := (q|_{v}^{\underline{zxu}}, \gamma) \\
\delta(q|_{\lambda}^{\underline{zx}}, x, w) := (q|_{zx}^{w}, \lambda) & \text{and} & \delta(q|_{\underline{xv}}^{\underline{zx}}, \lambda, w) := (q|_{zxv}^{w}, \lambda) \\
\delta(q|_{\lambda}^{\underline{zx}}, x, \beta) := (q|_{zx}^{\lambda}, \beta) & \text{and} & \delta(q|_{\underline{xv}}^{\underline{zx}}, \lambda, \beta) := (q|_{zxv}^{\lambda}, \beta)
\end{array}$$

Finally, also the simulation of M must be possible, when the memory is not empty. Thus for $x \in \Sigma$ we define the analogue to transitions defined in 3.2 for reading from the tape:

$$\delta(q|_{\underline{xv}}^{\lambda}, \lambda, \gamma) := (q'|_{v}^{\lambda}, \alpha) \text{ where } \varphi(q, x, \gamma) = (q', \alpha).$$

There are no other transitions than the ones defined above. We now prove that $L^{\heartsuit \leq n} \subseteq L(A)$. For this, one observation is essential, whose truth should be immediately comprehensible after what we have already said about the way that A works.

Lemma 3.4.7. *If from a state $q|_{\lambda}^{u}$ with vw next on the working tape and γ on the stack there exists an accepting computation for A, then from $q|_{v}^{u}$ with w next on the working tape and γ on the stack there*

also exists an accepting computation.

With this we can prove $L^{\heartsuit \leq n} \subseteq L(A)$ by induction on the number of duplications used to reach a word $w \in L^{\heartsuit \leq n}$ from a word $u \in L$. While neither u nor the number need to be unique, they both must exist for all words in $L^{\heartsuit \leq n}$. So let u be a word such that $w \in u^{\heartsuit \leq n}$ via $k+1$ duplications. Then there exists a word u' reachable from u via k duplications such that $u' \heartsuit^{\leq n} w$.

Let us suppose that all words, which can be generated by k duplications from words in L, are accepted by A; then $u' \in L(A)$, and there exists an accepting computation of A for u', let us call it Ξ. Further let i, ℓ be integers such that the duplication of the factor of length ℓ starting at position i in u' results in w, i.e. $w = u'[1 \dots i-1]u'[i \dots i+\ell-1]^2 u'[i+\ell \dots |u'|]$. Obviously A can on input w follow the computation Ξ on the prefix $u'[1 \dots i-1]$. Let us call the configuration reached in the step before reading the next input letter ξ and let its state be s. Then in s the memory is empty, otherwise A would not read from the input tape.

Now instead of following Ξ further, we guess the duplication of $u'[i \dots i+\ell-1]$ and reduce it in the manner described above. At the end of this process we will have reached a state equal to s except for the fact that its memory contains $u'[i \dots i+\ell-1]$. On the tape we have left $u'[i+\ell \dots |u'|]$. By Lemma 3.4.7 there is an accepting computation for this configuration if there is one for ξ. Since Ξ is such an accepting computation, also w is accepted by A.

Further, A can obviously simulate any computation of M and thus $L(M) \subseteq L(A)$, i.e. all words reachable by zero duplications are in $L(A)$. Thus also the basis for our induction is given and we have $L^{\heartsuit \leq n} \subseteq L(A)$.

We do not prove in detail that $L(A) \subseteq L^{\leq n}$. The two parts of A, the one deterministically reducing duplications and the one simulating the original PDA M work practically independently, as the corresponding state sets are disjoint and separated by the match being filled or not. From these facts $L(A) \subseteq L^{\leq n}$ should be comprehensible rather easily. □

Of course, the same construction works for any finite set of factors that can be duplicated, and we immediately obtain a corollary.

Corollary 3.4.8. *The class of context-free languages is closed under the operation of uniformly bounded duplication.*

For general duplication this proof technique does not apply, because over three letters there is no n such that $(abc)^{\heartsuit} = (abc)^{\heartsuit \leq n}$.

In fact, n-bounded duplication grows more powerful with every increase of n. Here we will use the following two notions: a word w is *square-free*, if it does not contain any non-empty factor of the form $uu = u^2$; w is *circular square-free*, if the same holds true for w written along a circle, or equivalently if ww contains no square shorter than itself.

Proposition 3.4.9. *For two integers m and n with $17 < m < n$ the inclusion $(abc)^{\heartsuit \leq m} \subset (abc)^{\heartsuit \leq n}$ is proper.*

Proof. First we show that for every square-free word u over three letters starting with abc there exists a word v, such that $uv \in (abc)^{\heartsuit \leq k}$ for $k \geq 4$. This word is constructed from left to right in the following manner. The first three letters are abc and thus do not need to be constructed.

The fourth letter is created by going from the third letter left to the last occurrence of this desired letter. Since abc is a prefix of the word all three letters do have such an occurrence. Now the factor from this rightmost occurrence to the third letter is duplicated. In this way the fourth letter of the new word becomes the desired one. Then we move to the fifth letter, obtain it by duplicating the factor reaching back till its rightmost occurrence, and so on.

The last occurrence of any letter in the part of u already constructed can be at most four positions from the last, because there are only two more different letters and the longest square-free word over two letters has length three. Of course, if in some step more than one letter of u is produced, the process can advance to the next wrong one without further duplications.

We will illustrate this construction with a short example. From abc we construct $abcbacb$ as a prefix. Underlining signals the factor duplicated to obtain the following word, the horizontal bar signals the end of the prefix of $abcbacb$ constructed at the respective point. $a\underline{bc} \rightarrow \underline{abcb}|c \rightarrow abc\underline{ba}|bcbc \rightarrow abcbacb|abcbc$

We now establish some bounds for the number of additional symbols produced. Since abc is already there, $|u| - 3$ letters need to be constructed. In every step at most $2k - 1$ letters of u can be constructed, because u is square-free; thus at least one letter is added to v. At the same time at most $2k - 1$ letters are added to v, since no useless duplications are done. Thus we have $|u| - 3 \leq |v| \leq (|u| - 3)(2k - 1)$. Of course, every circular square-free word is square-free and can be constructed in this way, too. Starting from lengths of 18, such a word always exists [21].

Now we construct in this way a circular square-free word w of

length n as a prefix of a word wv' in $(abc)^{\heartsuit \leq n}$. We can expand this prefix to w^i in $i-1$ steps for any given $i \geq 1$ by the rule $w \to ww$, so all $w^i v'$ are in $(abc)^{\heartsuit \leq n}$. Further, w^i contains no squares shorter than $2n$, because w is circular square-free. Thus for constructing the same prefix in $(abc)^{\heartsuit \leq m}$ also the bounds $|w^i|-3 \leq |v| \leq (|w^i|-3)(2m-1)$ for the corresponding suffix v apply. For big enough i the shortest such v will be longer than v'. Thus such a $w^i v'$ cannot be in $(abc)^{\heartsuit \leq m}$, while it is in $(abc)^{\heartsuit \leq n}$. □

Concluding Thoughts

Lately many new theoretical models of computation have either been inspired by observation of one or more phenomena occurring in biochemistry, or even state as their primary objective the proposal of ways to design a computing device working on a molecular level. Also duplication languages are a mathematical construct that originates from a behavior of DNA strands.

This does not mean that our investigations had in any way as their objective the construction of of a biochemical computing device — rather we have defined an idealized version of the naturally occurring duplication: it can act anywhere on factors of any length and structure; and when talking about general idempotencies, we have gone even farther away from anything that might be immediately applicable in DNA computing or other fields.

Seeing what kind of research is called for by politics and the economy these days, and what kind is not, one might take this abstraction as a deficiency and rather call for restrictions to the duplication operation, which make it more realistic. A four letter alphabet like in DNA and restriction of the cutting sites to the docking sites of certain enzymes might go in this kind of direction.

However, we have intentionally chosen a somewhat contrary path. For one thing, even such supposedly realistic restrictions would still leave the model highly abstract, and the probability of creating a model actually applicable would remain minuscule. The chances that some day somebody will construct a biochemical device for computation using some of the results presented in this thesis would probably not have increased much.

On the other hand, a set of restrictions like mentioned above above would have deprived our concepts of their generality. With the definition of idempotency languages as it is stated here, the results obtained may be of interest in any domain, where rewriting within sequences occurs in any of the ways described – be that insertion, duplication, deletion or any other. This generality can only be achieved by holding the definition as slender and unspecialized as possible.

Even more importantly, we hope to have demonstrated that the theory of languages generated by idempotencies contains a great

number of problems interesting in themselves by the mere virtue of their mathematical beauty. For example there is the fact that in all three variants we have regular and non-regular versions; for uniformly bounded, bounded and unbounded idempotency languages. In each variant the conditions for regularity are fundamentally different, which shows the richness of the structure of the language classes generated. With a uniform bound, insertion generates non-regular languages, while for bounded and general idempotencies it is almost trivially regular; on the other hand uniformly bounded duplication is regular, while in the general case we do not even know if it can be shown to be context-free.

The coincidence of this richness with the simplicity and elementarity of our definitions is what has made research on idempotency languages so fascinating and satisfying, while at the same time a great number of interesting problems remain open and call for further work. We will now conclude this thesis by highlighting a selected few of these on the following pages.

Selected Problems Left Open

We conclude by listing some interesting but unresolved questions from the fields treated in this thesis. Some of these have been stated explicitly already in the preceding chapters, others not. The concise compilation here will certainly be helpful for anyone in search of some interesting questions to work on in the field of idempotency languages.

- Our choice in how to apply the idempotency rules in generating languages is not the only way this can be done. We have just followed the spirit of the original definition for duplication by Dassow et al. [26]. For example, idempotency is in general based on equality, which is symmetric. So one might look at the effects of applying rules in both possible directions, i.e. increasing and decreasing length. Over two letters, there would only be seven duplication languages then, corresponding to the seven square-free words. Over a one-letter alphabet there would be exactly n uniformly n-bounded duplication languages. The Theorem of Green and Rees even states that for alphabets of any finite size there would be only a finite number of duplication languages [35], and the non-counting classes in general constitute cases of idempotency languages in the sense described here. Without length restrictions several results have already been established as mentioned in Section 2.2.2, but it seems that for our type of length-restrictions no work in this direction has been done.

- As described in the motivation for duplication, the DNA operation of duplication takes two strings uvw and creates $uvvw$. But here also uw is created at the same time; of course, in the physical world the additional letters of the second v cannot appear out of nowhere. This second product was disregarded in the definition of the duplication operation. We could as well investigate a variant, where both form part of the language generated and are processed further. In some sense this would be an operation preserving the total number of symbols involved, whereas

our idempotency rules add and delete symbols in arbitrary numbers.

- The problem, which has attracted most attention in our context is certainly the context-freeness of general duplication languages over three letters. However, no decisive advances have been made up to this point. This is a property the problem shares with another one, which has received even more attention: the context-freeness of the language of primitive word, which was first mentioned by Dömösi, Horváth and Ito in 1991 [30] and which remains unresolved despite the fact that many a good mathematician has spent many an hour on trying to solve it. Stated in the terminology of Section 3.2 the question is, whether $\sqrt{\Sigma^+}$ is context-free.

  As shown in Section 3.1.1 for duplication, in both cases the languages under question are very dense, which accounts for the fact that they fulfill all known pumping properties. Unfortunately, pumping properties are almost the only easy-to-apply test for non-context-freeness. Thus it may very well be that some fundamentally new results about context-free languages are required before answers can be found.

- Also when investigating the confluence of the idempotency relations considered, the status of general duplication remains unclear. As shown in the proof of local confluence in Proposition 2.6.18, almost the diamond property holds — but already the two steps in one of the converging derivations could in principle make a relation non-confluent. We still conjecture that for duplication this is not the case.

  There is also a bounded class of idempotency relations, whose confluence remains an open problem. For relations ${}^{\leq k}\bowtie_1^n$ for $n \geq 3$ the proof technique of the case $n = 2$ from Proposition 2.5.4 does not apply, and therefore some other type of reasoning would need to be found.

- Example 3.2.16 provides a rather simple language of the form w^+ such that its duplication root $\sqrt[\heartsuit]{w^+}$ is finite. The proof of this is not immediate, w is quite long, and we need an alphabet of four letters. It is an interesting problem to find a shorter example, in the ideal case one, whose optimality can be proven. And, of course, the question is, whether the size of the alphabet can be reduced to three. We suspect that this is possible, while for the size of two we know that all duplication roots are finite.

As a matter of fact, not even an example of a word over three letters with more than two duplication roots is known. While such an example can rather certainly be found by computer search, the more challenging task would be to bound the maximal number of roots as a function of the original word's length - bound it both from above and below. This would probably give us more insight in the structures that can be produced by nested duplications.

- Another problem left open in Section 3.2 is the decidability of the finiteness of the duplication root of regular languages. While it is often said that "all problems about regular languages are decidable," we strongly doubt that also this question is decidable. The reason for this lies in the power of duplication. We have seen that it can generate non-regular, possibly non-context-free languages from a single word. Even more importantly, Proposition 3.2.9 shows that questions about the roots of regular languages implicitly also deal with non-context-free languages, because the root of Σ^* is not context-free. Therefore the common rule just mentioned might not apply in this case, because implicitely non-regular languages are involved. The same might even be true for the length-bounded case.

Of course, many other problems have been left open throughout this thesis. But this list is by no means intended to be complete; it merely picks out a few problems that seem especially interesting according to the author's very personal judgment. And this judgment tells him to conclude the list at this point and thus conclude the entire thesis.

Bibliography

[1] S.I. Adian: *The Burnside Problem and Identities in Groups*. Springer-Verlag, Berlin, 1979 (translated from Russian).

[2] S.I. Adian and P.S. Novikov : *Infinite periodic groups I, II, III*. In: Izv. Akad. Nauk SSSR Ser. matem., v. 32 , No. 1, 2, 3 (1968), pp. 212–244, 251–524, 709–731; English translation in Math. USSR Izv., 2, 1968.

[3] L.M. Adleman: *Molecular Computation of Solutions to Combinatorial Problems*. In: Science 226, 1994, pp. 1021–1024.

[4] J.-P. Allouche and J. Shallit: *Automatic Sequences*. Cambridge University Press, Cambridge, 2003.

[5] J.M. Autebert, L. Boasson, and M. Latteux: *Motifs et bases de langages*. In: RAIRO Informatique Théorique et Applications 23(4), 1989, pp. 379–393.

[6] F. Baader and T. Nipkow: *Term Rewriting and All That*. Cambridge University Press, Cambridge, 1998.

[7] M. Beaudry, M. Holzer, G. Niemann and F. Otto: *McNaughton Families of Languages*. In: Theoretical Computer Science 290, 2003, pp. 1581–1628.

[8] J. Berstel: *Axel Thue's Work on Repetitions in Words*. In: 4th FPSAC, Publications du LaCIM 11, 1992, pp. 65–80.

[9] J. Berstel and L. Boasson: *Shuffle Factorization is Unique*. In: Theoretical Computer Science, Vol. 273, 2002, pp. 47–67.

[10] J. Berstel and D. Perrin: *Theory of Codes*. Academic Press, Orlando, 1985.

[11] R.V. Book (ed.): *Formal Language Theory — Perspectives and Open Problems.* Academic Press, New York, 1980.

[12] R. Book and F. Otto: *String-Rewriting Systems*. Springer, Berlin, 1988.

[13] D.P. Bovet and S. Varricchio: *On the Regularity of Languages on a Binary Alphabet Generated by Copying Systems*. In: Information Processing Letters 44, 1992, pp. 119–123.

[14] J. Brzozowski: *Open Problems about Regular Languages.* In: [11].

[15] G. Buntrock and K. Lorys: *On Growing Context-Sensitive Languages*. In: Lecture Notes in Computer Science 623 (19th ICALP), 1992, pp. 77–88.

[16] W. Burnside: *On an Unsettled Question in the Theory of Discontinuous Groups*. In: Quarterly Journal of Pure and Applied Mathematics, Vol. 33, 1902, pp. 230–238.

[17] A. Carpi and A. de Luca: *Semi-Periodic Words and Root-Conjugacy*. In: Theoretical Computer Science 292, 2003, pp. 111–130.

[18] B. Charlesworth, P. Sniegowski and W.Stephan: *The Evolutionary Dynamics of Repetitive DNA in Eukaryotes.* In: Nature 371, 1994, pp. 215–220.

[19] C. Choffrut and J. Karhumäki: *Combinatorics on Words*. In: [81].

[20] M. Crochemore: *Sharp Caracterizations of Squarefree Morphisms*. In: Theoretical Computer Science 18, 1982, pp. 221–226.

[21] J.D. Currie: *There are Ternary Circular Square-free Words of Length n for* $n \geq 18$. In: Electronic Journal of Combinatorics, 9(1) N10, 2002.

[22] J.D. Currie and S.D. Fitzpatrick: *Circular Words Avoiding Patterns*. In: DLT 2002, Lecture Notes in Computer Science 2450, Springer-Verlag, Berlin, 2003.

[23] E. Dahlhaus and M. K. Warmuth: *Membership for Growing Context-Sensitive Grammars is Polynomial.* In: Journal of Computer and System Sciences, 33(3), 1986, pp. 456–472

[24] J. Dassow, V. Mitrana: *On Some Operations Suggested by the Genome Evolution.* In: R. Altman, K. Dunker, L. Hunter and T. Klein (eds): Pacific Symposium on Biocomputing'97, 1997, pp. 97–108.

[25] J. Dassow, V. Mitrana: *Self Cross-Over Systems.* In: Gh. Păun (ed): Computing with Bio-Molecules, Springer, Singapore, 1998, 283–294.

[26] J. Dassow, V. Mitrana and Gh. Păun: *On the Regularity of Duplication Closure*. In: Bull. EATCS 69, 1999, pp.133–136.

[27] J. Dassow, V. Mitrana and A. Salomaa: *On Some Operations Suggested by the Genome Evolution*. In: Theoretical Computer Science 270(1-2), 2003, pp. 701–738.

[28] N. Dershowitz: *Semigroups Satisfying* $x^{m+n} = x^n$. In: Lecture Notes in Computer Science, Vol. 656, Springer-Verlag, Berlin, 1993, pp. 307–314.

[29] A. do Lago and I. Simon: *Free Burnside Semigroups*. In: RAIRO Informatique Théorique et Applications 35(6), 2001, pp. 579–595.

[30] P. Dömösi, S. Horváth and M. Ito: *On the Connection between Formal Languages and Primitive Words*. In: Analele Univ. din Oradea, Fasc. Mat., 1991, pp 59–67.

[31] A. Ehrenfeucht and G. Rozenberg : *On the Separating Power of EOL Systems*. In: RAIRO Informatique Théorique et Applications, Vol. 17, No. 1, 1983, pp. 13–22.

[32] A. Ehrenfeucht and G. Rozenberg: *On Regularity of Languages Generated by Copying Systems*. In: Discrete Applied Mathematics 8, 1984, pp. 313–317.

[33] S.Z. Fazekas: *Scattered Roots of Words*. Manuscript 2006.

[34] R. Freund, Gh. Păun, G. Rozenberg and A. Salomaa: *Watson-Crick Finite Automata*. In: Proceedings of the Third Annual DIMACS Symposium on DNA Based Computers, Philadelphia, 1994, pp. 535–546.

[35] J.A. Green and D. Rees: *On Semigroups in Which $x^r = x$*. In: Proceedings of the Cambridge Philosophical Society, Vol. 48, 1952, pp. 35–40.

[36] M.A. Harrison: *Introduction to Formal Language Theory*. Reading, Mass., 1978.

[37] T. Head: *Formal Language Theory and DNA: an Analysis of the Generative Capacity of Specific Recombinant Behaviours.* In: Bull. Math. Biology 49, 1987, pp. 737–759.

[38] T. Head: *Visualizing Languages Using Primitive Powers*. In: Words, Semigroups, and Transductions, World Scientific, New Jersey, 2001, pp. 169–180.

[39] D. Hofbauer and J. Waldmann: *Deleting String-Rewriting Systems Preserve Regularity*. In: Theoretical Computer Science 327, 2004, pp. 301-317.

[40] S. Horváth and M. Ito: *Decidable and Undecidable Problems of Primitive Words, Regular and Context-free Languages*. In: Journal of Universal Computer Science, Nov. 1999.

[41] S. Horváth and M. Ito: *On Some Properties of the Periodicity Degree*. Presentation at AFL'02, Debrecen, 2002.

[42] S. Horváth, P. Leupold and G. Lischke: *Roots and Powers of Regular Languages*. In: Lecture Notes in Computer Science 2450, DLT 2002, Springer-Verlag, Berlin, 2002, pp. 220–230.

[43] L. Ilie: *On a Conjecture about Slender Context-Free Languages*. In: Theoretical Computer Science 132(1-2), 1994, pp. 427–434.

[44] L. Ilie: *On Lengths of Words in Context-Free Languages*. In: Theoretical Computer Science 242(1-2), 2000, pp. 327–359.

[45] M. Ito: *Algebraic Theory of Automata and Languages*. World Scientifc, New Jersey, 2004.

[46] M. Ito, L. Kari, and G. Thierrin: *Insertion and Deletion Closure of Languages*. In: Theoretical Computer Science 183, 1997, pp. 238–247.

[47] M. Ito, P. Leupold, and K. Shikishima-Tsuji: *Closure of Language Classes under Bounded Duplication*. In: Lecture Notes in Computer Science 4036, DLT 2006, Springer, Berlin, 2006, pp. 238–247.

[48] B. Krawetz: *Monoids and the State Complexity of the Operation root(L)*. Master's Thesis, University of Waterloo, Ontario, Canada, 2003.

[49] A. Kucera and J. Strejcek: *The Stuttering Principle Revisited.* In: Acta Informatica 41(7/8), 2005, pp. 415–434.

[50] P. Leupold: *Some Properties of Context-Free Languages Related to Primitive Words*. In: Preproceedings of Words'03, TUCS General Publications, Turku, 2003.

[51] P. Leupold: *n-Bounded Duplication Codes*. Proceedings of the ICALP-Workshop on Words, Avoidability, Complexity, Turku 2004. Technical Report 2004-07, Laboratoire de Recherche en Informatique d'Amiens, Amiens 2004.

[52] P. Leupold: *Partial Words for DNA Coding*. In: Proceedings of DNA 10, LNCS 3384, Springer, Berlin, 2005, pp. 224-234.

[53] P. Leupold: *Languages Generated by Iterated Idempotencies*. Accepted for publication in Theoretical Computer Science.

[54] P. Leupold: *General Idempotency Languages over Small Alphabets*. Accepted for publication in Journal of Automata, Languages and Combinatorics.

[55] P. Leupold: *Duplication Roots – Extended Abstract*. In: Proceedings Theorietag Automaten und Formale Sprachen, TU Wien, Wien, 2006, pp. 91–93.

[56] P. Leupold, C. Martín Vide and V. Mitrana: *Uniformly Bounded Duplication Languages*. In: Discrete Applied Mathematics 146 (3), 2005, pp. 301–310.

[57] P. Leupold and V. Mitrana: *Uniformly Bounded Duplication Codes*. Accepted for publication in RAIRO Informatique Théorique.

[58] P. Leupold, V. Mitrana and J. Sempere: *Languages arising from gene repeated duplication*. In: Aspects of Molecular Computing. Essays Dedicated to Tom Head on the Occasion of his 70th Birthday. LNCS 2950, Springer Verlag, Berlin, 2004, pp. 297–308.

[59] G.Levinson, G. Gutman: *Slipped-Strand Mispairing: a Major Mechanism for DNA Sequence Evolution.* In: Molec. Biol. Evol. 4, 1987, pp. 203–221.

[60] G. Lischke: *The Root of a Language and its Complexity*. Proceedings of Developments in Language Theory 2001, Lecture Notes in Computer Science 2295, Springer-Verlag, pp. 272–280.

[61] M. Lothaire: *Combinatorics on Words*. Addison-Wesley, Reading, MA, 1983.

[62] M. Lothaire: *Algebraic Combinatorics on Words*. Cambridge University Press, Cambridge, 2002.

[63] M. Lothaire: *Applied Combinatorics on Words*. Cambridge University Press, Cambridge, 2005.

[64] A. de Luca and S. Varricchio: *On Noncounting Regular Classes.* In: Theoretical Computer Science 100, 1992, pp. 67–104.

[65] R.C. Lyndon and M.P. Schützenberger: *On the Equation $a^M b^N = c^P$ in a Free Monoid*. Michigan Math. Journal 9, 1962, pp. 289–298.

[66] C. Martín-Vide and Gh. Păun: *Duplication Grammars.* In: Acta Cybernetica 14, 1999, pp. 101–113.

[67] R. McNaughton, P. Narendran and F. Otto: *Church-Rosser Thue Systems and Formal Languages*. In: Journal of the ACM 35, 1988, pp. 324–344.

[68] R. McNaughton and S. Papert: *Counter-Free Automata*. MIT Press, Cambridge, Massachusetts, 1971.

[69] *MedTerms* Online Medical Dictionary: http://www.medterms.com/

[70] A. Meyer: *Molecular Evolution: Duplication, Duplication.* In: Nature 421, 2003, pp. 31–32.

[71] G.A. Miller: *The Science of Words*. Freeman, New York, 1991.

[72] V. Mitrana and G. Rozenberg: *Some Properties of Duplication Grammars.* In: Acta Cybernetica 14, 1999, pp. 165–177.

[73] G. Moore: *Crumming More Components onto Integrate Circuits*. In: Electronics Magazine, April 19th, 1955.

[74] M. Morse: *Recurrent Geodesics on a Surface of Negative Curvature*. In: Transactions American Mathematical Society 22, 1921, pp. 84–100.

[75] M. Morse: *A Solution of the Problem of Infinite Play in Chess*. In: Bulletin of the American Mathemetical Society 44, 1938, pp. 84–100.

[76] Gh. Păun, G. Rozenberg and A. Salomaa: *DNA Computing – New Computing Paradigms*. Springer Verlag, Berlin, 1998.

[77] D. Peled, T. Wilke, and P. Wolper: *An Algorithmic Approach for Checking Closure Properties of ω-Regular Languages.* In: Theoretical Computer Science 195(2), 1998, pp. 183–203.

[78] D. Raz: *Length Considerations in Context-Free Languages*. In: Theoretical Computer Science 183(1), 1997, pp. 21–32.

[79] R. Ross and K. Winklmann: *Repetitive Strings are not Context-Free*. In: RAIRO Informatique Théorique et Applications 16(3), 1982, pp. 191–199.

[80] G. Rozenberg and A. Salomaa: *Cornerstones of Undecidability*. Prentice-Hall, Englewood Cliffs, 1994.

[81] G. Rozenberg and A. Salomaa (eds.): *Handbook of Formal Languages*. Springer-Verlag, Berlin, 1997.

[82] A. Salomaa: *Formal Languages*. Academic Press, Orlando, 1973.

[83] C. Schlotterer, D. Tautz: *Slippage Synthesis of Simple Sequence DNA.* In: Nucleic Acids Res. 20, 1992, pp. 211–215.

[84] D. B. Searls: *The Computational Linguistics of Biological Sequences.* In: L. Hunter (ed): Artificial Intelligence and Molecular Biology, AAAI Press/MIT Press, Menlo Park, CA/Cambridge, MA, 1993, pp. 47–120.

[85] H.J. Shyr: *Free Monoids and Languages*. Hon Min Book Company, Taichung, 1991.

[86] H.J. Shyr and G. Thierrin: *Monogenic e-Closed Languages and Dipolar Words*. In: Discrete Mathematics 126, 1994, pp. 339–348.

[87] H.J. Shyr and S.S. Yu: *Non-Primitive Words in the Language p^+q^+*. In: Soochow J. Math., Nr. 4, 1994.

[88] M. Strand, T. Prolla, R. Liskay and T. Petes: *Destabilization of Tracts of Simple Repetitive DNA in Yeast by Mutations Affecting DNA Mismatch Repair.* In: Nature 365, 1993, pp. 274–276.

[89] A. Thue: *Über unendliche Zeichenreihen*. In: Norske Videnskabers Selskabs Skrifter Mat.-Nat. Kl. (Kristiania), 7, 1906, pp. 1–22.

[90] A. Thue: *Über die gegenseitige Lage gleicher Teile verschiedener Zeichenreihen*. In: Norske Videnskabers Selskabs Skrifter Mat.-Nat. Kl. (Kristiania), 1, 1912, pp. 1–67.

[91] A. Thue: *Probleme über die Veränderung von Zeichenreihen nach gegebenen Regeln*. In: Norske Videnskabers Selskabs Skrifter Mat.-Nat. Kl. (Kristiania), 10, 1914, 34 pp.

[92] M.-W. Wang: *On the Irregularity of the Duplication Closure*. In: Bull. EATCS 70, 2000, pp. 162–163.

[93] M. Weitzmann, K. Woodford and K. Usdin: *DNA Secondary Structures and the Evolution of Hyper-Variable Tandem Arrays.* In: Journal of Biological Chemistry 272, 1997, pp. 9517–9523.

[94] R.Wells: *Molecular Basis of Genetic Instability of Triplet Repeats.* In: J. of Biological Chemistry 271, 1996, pp. 2875–2878.

[95] J.R. Woinowski: *The Context-Splittable Normal Form For Church Rosser Language Systems*. In: Information and Control 183, 2003, pp. 245-274.