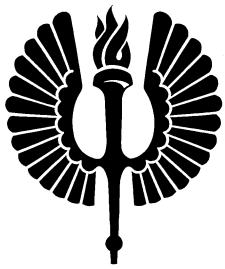


ON THE CENTRALIZER OF A RATIONAL LANGUAGE AND THE FIXED POINT METHOD

Petri Salmela

Licentiate thesis
Turku Centre for Computer Science
September 2005

UNIVERSITY OF TURKU
DEPARTMENT OF MATHEMATICS
FIN-20014 TURKU
FINLAND

UNIVERSITY OF TURKU
Department of Mathematics

SALMELA, PETRI On the centralizer of a rational language and the fixed point method

Licentiate thesis, 66 pages
Mathematics
September 2005

In this work we consider the commutation of rational languages. Especially we study the centralizer, the maximal language commuting with a given language and Conway's problem connected to it. J.H. Conway asked in 1971 (Regular algebra and Finite Machines, Chapman Hall,1971), if the centralizer of a rational language is also rational. The question remained unanswered for a long time, until just recently M. Kunc gave it the negative answer.

We shall present the fixed point approach introduced by J. Karhumäki (Challenges of Commutation - An advertisement, in: FCT 2001, R. Freivalds, (ed.), Lecture Notes in Computer Science 2138, Springer-Verlag, New York 2001, pp. 15-23). This approach is an iterative method for finding the centralizer of a given language as a maximal fixed point of certain language operation. For most of rational languages this method gives the result rather soon, but on some cases it leads to an infinit computation.

First we introduce some required tools and notations. Next we give a few results on the centralizer in general and in some special cases. Finally we study, how the fixed point approach behaves on some selected examples. We discuss few cases where approach is successful and also some cases for which the approach fails. We show that the fixed point approach can fail even when the given language is finite.

As a computing tool, we have used the computer program called Grail+. This program is developed in the University of Western Ontario, Canada, and improved in the University of Turku.

Keywords: rational languages, commutation, the centralizer, fixed point approach, Conway's problem.

Contents

1 Introduction

In this work we will consider a problem related to the commutation of two rational languages. For two single words commutation is very well known. Two words commute if and only if they are powers of a common word. Generally there is quite much knowledge on equations on words [16, 1]. However, very little is known about equations on languages. Many cases seem to lead to hard and undecidable problems. To see this contrast between word equations and language equations, we can note, that the satisfiability problem for word equations is decidable, by the result of S. Makanin [17]. This problem is equivalent with a corresponding problem for finite sets of word equations and with the same problem for rational sets of word equations, see Culik and Karhumäki [5]. W. Plandowski proved, that the satisfiability problem for words is even in PSPACE [20]. On the other hand, in [9] Karhumäki and Lisovik prove, that the satisfiability problem of rational systems of equations over finite sets of words is undecidable. For more about language equations in general, see [15].

The commutation equation $XY = YX$ is one special case of language equations. It has extremely simple formulation, but has appeared to be the source of very hard problems. Our focus is on one of them, so called Conway's problem [4]. This problem concentrates on the *centralizer* $\mathcal{C}(X)$ of a given language X, i.e. the largest language commuting with the language X. In other words, we are interested in the maximal solution Y of the commutation equation, with fixed language X. Conway's problem asks, if the centralizer of the rational language is also rational. This problem was recently solved by M. Kunc [14], who gave a negative answer for the question, and even in a very strong form. In his breakthrough result he proved, that the centralizer

of a rational language does not need to be even recursively enumerable. Not even for finite languages.

There has been several different approaches to attack Conway's problem. In his PhD Thesis [19] I. Petre used results on formal power series. In [13] Karhumäki and Petre discuss so called branching point approach on Conway's problem. So called fixed point approach is studied in [12] and [6]. This fixed point approach is used also in this work. It can be used for finding the centralizer of rational language in most cases.

The method, which Kunc uses is closely related to so called TAG systems of Post. In TAG systems one starts with some initial word and proceeds by using some given rules for rewriting. In the case of centralizer, the initial word w is chosen from the centralizer of language X and the rewriting rules are $w \mapsto (xw)y^{-1}$ and $w \mapsto y^{-1}(wx)$ where words x and y are from X. Kunc represents his proof as a game between an attacker and a defender. The attacker tries to prove that selected initial word is not in the centralizer and the defender tries to show that it is. The attacker chooses words x and the defender chooses words y. The attacker wins, if repetitive rewriting leads to a word which is for sure not in the centralizer, i.e. when the defender can not choose any y in X. The defender is the winner in the case that game continues forever. Now the initial word w is in the centralizer if and only if the defender has a winning strategy, i.e. if he can keep game going on, whatever the attacker does.

We will give a solution to Conway's problem on so called *singular languages* in the finite case. We will also use the same method to give a partial solution for 4-element sets. Next we will give the definition of the fixed point approach.

Finally we go through some examples to see, what kind of centralizers we can have for rational languages and how fixed point approach works in different cases. We study these cases mostly by implementing the method with *Grail+*-software [22], but in most cases we also prove the correctness of the result manually. In some cases we also illustrate the iterative steps by hand step by step.

The Grail+ software is developed in the University of West Ontario in Canada. It is a set of command line tools implementing several operations on finite automata, finite languages and regular expressions. It also supports UNIX-like piping of output

of one operation to the input of another. Special thanks go to *Arto Lepistö* from University of Turku, who has greatly improved the Grail+-software by optimizing algorithms and adding several additional and needed operations.

In the last chapter we study the behavior of the fixed point method in the case where the algorithm does not halt. In these cases the program will keep computing step after step getting more and more complex results as iteration steps. We will analyze some example cases, both finite and infinite, to see that they exist and to find out why the method does not halt.

2 Notations

We will start by defining some notation we will use in this work. For more detailed definitions we refer to [16].

Through this work we will mainly use Σ to deduce the *alphabet*, a (finite) set of letters. As usual Σ^* and Σ^+ are the free monoid and the free semigroup generated by Σ. For a language L the language L^* is the monoid $\cup_{i\geq 0} L^i$ and L^+ is the semigroup $\cup_{i\geq 1} L^i$.

Small letters $a, b, c, d, \ldots$ (with or without subscripts) are mainly considered as letters of the alphabet Σ and letters s, t, u, v, w, x, y, z (with or without subscripts) are used for words, i.e. strings of letters. The empty word is expressed with the symbol 1.

Capital letters are used as symbols for languages, the language X being usually the one which we are studying. A language which does not contain the empty word 1, is called 1-*free*.

Notation $|X|$ refers to the cardinality of the set X and notation $|w|$ means the length of the word w. The word $u \in \Sigma^*$ is called a *prefix* or *left factor* (resp. *suffix* or *right factor*) of the word $w \in \Sigma^*$ if $w = uv$ (resp. $w = vu$) for some word $v \in \Sigma^*$. We also use notation $u = wv^{-1}$ (resp. $u = v^{-1}w$), which refers to erasing of the word v from word w from right (resp. left). The corresponding notation is also extended for languages.

$$\begin{aligned} L_2^{-1}L_1 &= \{u_2^{-1}u_1 \mid u_1 \in L_1, u_2 \in L_2\} \\ L_1L_2^{-1} &= \{u_1u_2^{-1} \mid u_1 \in L_1, u_2 \in L_2\}. \end{aligned}$$

We use notation $\mathrm{Pref}_1(w)$ for the prefix of length 1 of a non-empty word w, i.e. the first letter of the word. Similarly notations $\mathrm{Pref}_n(w)$ and $\mathrm{Suf}_n(w)$ mean the prefix and the suffix of length n, of word w with length at least n respectively. We extend these notations in a natural way for languages

$$\begin{aligned}\mathrm{Pref}_1(X) &= \{\mathrm{Pref}_1(w) \in \Sigma \mid w \in X, |w| \geq 1\}\\ \mathrm{Pref}_n(X) &= \{\mathrm{Pref}_n(w) \in \Sigma^* \mid w \in X, |w| \geq n\}\\ \mathrm{Suf}_n(X) &= \{\mathrm{Suf}_n(w) \in \Sigma^* \mid w \in X, |w| \geq n\}.\end{aligned}$$

Notations $\mathrm{Pref}(X)$ and $\mathrm{Suf}(X)$ refer to sets of all prefixes and suffixes of arbitrary length.

We will say that a word $u \in \Sigma^*$ is a *proper prefix* of a word $w \in \Sigma^*$, if there exists a word $v \in \Sigma^+$ such that $w = uv$. We also call u a proper prefix of a language X, if u is a proper prefix of some word in X. In this case it is possible, that u is also in X itself. With this notation, for example a word aba is a proper prefix of a language $\{aba, ababb\}$.

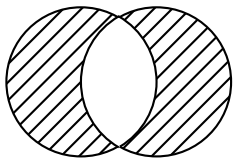
The notation w^R is used for the *reverse* of the word w. This means that if $w = a_1a_2\cdots a_n$ for some $a_i \in \Sigma$, then $w^R = a_n\cdots a_2a_1$, i.e. w^R is the same word as w, but written from right to left. Also this notation is extended in a natural way to sets of words

$$X^R = \{w^R \mid w \in X\}.$$

A language X is called *periodic*, if all of its words are powers of the same word, i.e. if $X \subseteq u^*$ for some word $u \in \Sigma^*$.

In many places in this work we have used infix operator $+$ to stand for the union of two languages. This notation is used mainly in rational expressions and can be justified by the fact, that powerset 2^{Σ^*} with catenation and union forms a semiring. The more traditional set theoretic symbol $\cup$ is also used in some places.

The infix operator Δ will be used as shorthand for the *symmetric difference* $A\Delta B = (A \cup B) \setminus (A \cap B) = (A \setminus B) \cup (B \setminus A)$ of two sets A and B.



3 Centralizer

We will take a look on commutation of two languages. We will define the centralizer of a language and prove some properties of it. Finally we will formulate a problem, introduced by J.H. Conway, which was open for a long time, and was just recently solved by M. Kunc [14].

3.1 Commutation of languages

Commutation of languages X and Y is defined with a simple equation $XY = YX$. However, it seems to be difficult to find general rules when languages do commute. It is well known result in combinatorics on words [16], that for two words $u, v \in \Sigma^*$ equation $uv = vu$ holds if and only if there exists some word $t \in \Sigma^*$ such that $u = t^i$ and $v = t^j$ for some integers i and j. Any general rules for commutation of entire languages are not likely to be found.

If languages X and Y commute, what does it mean for single words of these languages? For all words $x_1 \in X$ and $y_1 \in Y$ there exists such words $x_2 \in X$ and $y_2 \in Y$ that

$$x_1 y_1 = y_2 x_2$$

and such words $x_3 \in X$ and $y_3 \in Y$ that

$$y_1 x_1 = x_3 y_3.$$

Example 3.1. Languages $X = \{a, aaa, b, ba, ab, aba\}$ and $Y = X \cup \{aa\}$ commute.

First we see clearly, that $XX \subseteq XY$. Secondly, since

$$\begin{aligned} aa \cdot a &= a \cdot aa & aa \cdot aaa &= aaa \cdot aa \\ aa \cdot b &= a \cdot ab & aa \cdot ba &= a \cdot aba \\ aa \cdot ab &= aaa \cdot b & aa \cdot aba &= aaa \cdot ba, \end{aligned}$$

we get that $\{aa\}X \subseteq XY$. These together imply $YX = (X \cup \{aa\})X \subseteq XY$. Since X and Y are "symmetric", in the sense that $X = X^R$ and $Y = Y^R$, we obtain

$$\begin{aligned} YX &\subseteq XY \\ (YX)^R &\subseteq (XY)^R \\ X^R Y^R &\subseteq Y^R X^R \\ XY &\subseteq YX. \end{aligned}$$

Therefore $XY = YX$.

3.2 Centralizer and its properties

Next we fix one of the languages X and Y and focus on the maximal language commuting with it.

Definition 3.2. There are two *centralizers* for language X over alphabet Σ. The $*$-centralizer, or *monoid centralizer*, $\mathcal{C}_*(X)$ is the maximal subset of monoid Σ^* commuting with language X. Similarly the $+$-centralizer, or *semigroup centralizer*, $\mathcal{C}_+(X)$ is the maximal subset of semigroup Σ^+ commuting with X.

First we have to make sure, that these centralizers really exist for all X, and that they are unique. For any language X there always exists some language commuting with X, since for any $n \geq 0$ the equation $X^n X = X X^n$ holds, and especially $X^* X = X X^*$. Also the semigroup Σ^+ includes always some subsets commuting with X. If $1 \notin X$, at least X^+ commutes with X, and if $1 \in X$, even $\Sigma^+ X = \Sigma^+ = X\Sigma^+$.

Now the union of all languages commuting with X

$$\bigcup \{A \subseteq \Sigma^* \mid AX = XA\}$$

commutes also with X, since the distributive law holds. The fact, that this union includes all languages commuting with X, means that it is the maximal language commuting with X, i.e. the centralizer $\mathcal{C}_*(X)$. Now the centralizer $\mathcal{C}_*(X)$ is unique and always exists.

The uniqueness and existence of the centralizer $\mathcal{C}_+(X)$ is proven similarly by replacing Σ^* by semigroup Σ^+.

We must note that the notion of centralizer is here slightly different from the usual meaning of centralizer in algebra. In algebra the centralizer is defined using element-wise commutation and the centralizer of given element x is the set of all elements commuting with it. In the case of languages, i.e. in the semiring 2^{Σ^*} this would mean the set $\mathcal{COM}(X) = \{Y \subseteq \Sigma^* \mid XY = YX\}$, using the notation of [19]. However we will call the greatest element of the set $\mathcal{COM}(X)$ the monoid centralizer of the language X. In his book [4] Conway introduced the centralizer originally with the name *normalizer*, which is not very accurate either.

We will now show some properties of the centralizer.

Theorem 3.3. *For each language X, $\mathcal{C}_*(X)$ is a monoid and $\mathcal{C}_+(X)$ is a semigroup.*

Proof. The language A is a monoid (resp. semigroup) if and only if $A = A^*$ (resp. $A = A^+$). We will prove the claim for the $*$-centralizer. For the $+$-centralizer the claim is proven similarly.

First we show by induction on n, that $\mathcal{C}_*(X)^n X = X\mathcal{C}_*(X)^n$ for every $n \geq 0$. First of all

$$\mathcal{C}_*(X)^0 X = \{1\}X = X = X\{1\} = X\mathcal{C}_*(X)^0.$$

On the other hand, if the claim holds when $n \leq k$, we get

$$\mathcal{C}_*(X)^{k+1}X = \mathcal{C}_*(X)^k\mathcal{C}_*(X)X = \mathcal{C}_*(X)^k X\mathcal{C}_*(X) = X\mathcal{C}_*(X)^{k+1}.$$

So $\mathcal{C}_*(X)^n X = X\mathcal{C}_*(X)^n$ for every $n \geq 0$ and by the distributivity law $\mathcal{C}_*(X)^* X = X\mathcal{C}_*(X)^*$. Since $\mathcal{C}_*(X)$ is maximal, we must have $\mathcal{C}_*(X)^* \subseteq \mathcal{C}_*(X)$. Inclusion $\mathcal{C}_*(X) \subseteq \mathcal{C}_*(X)^*$ holds trivially. This means that

$$\mathcal{C}_*(X)^* = \mathcal{C}_*(X)$$

and so the centralizer $\mathcal{C}_*(X)$ is a monoid. □

Theorem 3.4. *If $1 \in X$, then $\mathcal{C}_*(X) = \Sigma^*$ and $\mathcal{C}_+(X) = \Sigma^+$.*

Proof. If $1 \in X$, then $\Sigma^* X = \Sigma^* = X\Sigma^*$. The other claim is proven similarly. □

We will simply use notation $\mathcal{C}(X)$ for the centralizer of X, if there is no risk of confusion. We will mainly consider the $+$-centralizer. Results for $*$-centralizers seem to be in most cases either trivial or obtained similarly as for $+$-centralizers. However the connection between these two centralizers is not totally clear yet. There are four cases depending on which one of these centralizers we consider and if the empty word 1 is or is not in X.

The cases with $1 \in X$ are those of Theorem 3.4. The case $1 \notin X$ with $+$-centralizer is the case considered here. Most of the results for $+$-centralizers, considered in this work, can be applied almost analogously to the $*$-centralizers, with the difference that the base set is Σ^* instead of Σ^+. This doesn't necessarily mean, that $\mathcal{C}_*(X) = \mathcal{C}_+(X) \cup \{1\}$. For example if $X = \{a, ab, ba, bb\}$, then $\mathcal{C}_*(X) = \Sigma^*$ and $\mathcal{C}_+(X) = \Sigma^+ \setminus \{b\}$, as noted in [12]. Generally it is not known, if problems on these two centralizers can be reduced to each others.

Theorem 3.5. *The centralizer of X is also the centralizer of X^+, i.e.*

$$\mathcal{C}(X) = \mathcal{C}(X^+).$$

Proof. Since $C(X)$ commutes with X, it clearly, by induction, commutes also with X^n for every $n \geq 1$. That means, by distributivity law, that it commutes also with $X^+ = \bigcup_{n \geq 1} X^n$. So we have $\mathcal{C}(X) \subseteq \mathcal{C}(X^+)$.

On the other hand we know, that $X \subseteq X^+ \subseteq \mathcal{C}(X^+)$ and that $\mathcal{C}(X^+)$ is semigroup. Hence

$$X\mathcal{C}(X^+) \subseteq X^+\mathcal{C}(X^+) = \mathcal{C}(X^+)X^+ = \underbrace{\mathcal{C}(X^+)X^*}_{\subseteq \mathcal{C}(X^+)} \cdot X \subseteq \mathcal{C}(X^+)X.$$

Similarly $\mathcal{C}(X^+)X \subseteq X\mathcal{C}(X^+)$, so $X\mathcal{C}(X^+) = \mathcal{C}(X^+)X$. Since the maximality of $\mathcal{C}(X)$, we must also have inclusion

$$\mathcal{C}(X^+) \subseteq \mathcal{C}(X).$$

□

The centralizer is easily defined, but it is not always an easy task to find it. We can start by setting some upper and lower bounds for the centralizer.

Theorem 3.6. *Let $X \subseteq \Sigma^+$. Then for the centralizer $\mathcal{C}(X)$ holds the equation*

$$X^+ \subseteq \mathcal{C}(X) \subseteq \mathrm{Pref}(X^+) \cap \mathrm{Suf}(X^+). \tag{1}$$

Since we are using $+$-centralizer, we interpret $\mathrm{Pref}(X^+)$ and $\mathrm{Suf}(X^+)$ as 1-free prefix and suffix of X^+.

Proof. The first inclusion is clear, since X^+ always commutes with X. The other inclusion can be seen as inclusions in both $\mathrm{Pref}(X^+)$ and $\mathrm{Suf}(X^+)$ separately. We will take look on the prefix case, the suffix case being similar. For every integer $n \geq 1$ and words $z_1 \in \mathcal{C}(X)$, $x_1, \ldots, x_n \in X$ there exists words $z_2, \ldots, z_{n+1} \in \mathcal{C}(X)$ and $y_1, \ldots, y_n \in X$ such, that

$$z_i x_i = y_i z_{i+1},$$

where $1 \leq i \leq n$. This means that equation

$$z_1 x_1 \cdots x_n = y_1 \cdots y_n z_{n+1}$$

holds. Since $1 \notin X$, the length of every word y_i is at least one and so for n big enough we have $|y_1 \cdots y_n| \geq |z_1|$. Hence $z_1 \in \mathrm{Pref}(X^+)$ for every word z_1 in the centralizer and so $\mathcal{C}(X) \subseteq \mathrm{Pref}(X^+)$. □

We can also give some more accurate bounds for the centralizer. For example the language

$$S = \{w \in \Sigma^+ \mid wX \subseteq XX^+ \text{ and } Xw \subseteq XX^+\} \tag{2}$$

commutes with X and is hence included in the centralizer. This can be seen as follows. The definition of S gives us $X^+ \subseteq S$. Also by the definition $SX \subseteq XX^+ \subseteq XS$ and $XS \subseteq XX^+ = X^+X \subseteq SX$, so $XS = SX$.

This means

$$X^+ \subseteq S \subseteq \mathcal{C}(X). \tag{3}$$

In most cases the centralizer of the rational language X is either X^+ or S. The previous case is more precisely the case where $X^+ = S = \mathcal{C}(X)$. However there also exist languages having centralizer larger than S.

Example 3.7. As an example of the case where inclusion $S \subseteq \mathcal{C}(X)$ is proper, we could take a look on a language $X = \{a, ab, ba, bb\}$. The centralizer is $\mathcal{C}(X) = \Sigma^+ \setminus \{b\}$, but $\mathcal{C}(X) \neq S$ since for example $bab \in \mathcal{C}(X)$, but $bab \cdot bb \notin XX^+$, which implies that $bab \notin S$.

The definition of S was given as a set of words having a certain property. It can also be given as a rather simple formula. This formula can be easily implemented with software.

Theorem 3.8. *The language S can be represented as*

$$S = \Sigma^+ \setminus \left((\Sigma^+ \setminus XX^+)X^{-1} \cup X^{-1}(\Sigma^+ \setminus XX^+)\right). \tag{4}$$

Proof. We start from the definition of S in formula (2). Since

$$w \in S \iff (\forall a \in X)(wa \in XX^+ \wedge aw \in XX^+),$$

we have

$$\begin{aligned} w \notin S &\iff (\exists a \in X)(wa \notin XX^+ \vee aw \notin XX^+) \\ &\iff (\exists a \in X)(wa \in \Sigma^+ \setminus XX^+ \vee aw \in \Sigma^+ \setminus XX^+). \end{aligned}$$

The complement of S can now be given as

$$\begin{aligned} \Sigma^+ \setminus S &= \{w \in \Sigma^+ \mid (\exists a \in X)(wa \in \Sigma^+ \setminus XX^+)\} \\ &\quad \cup \{w \in \Sigma^+ \mid (\exists a \in X)(aw \in \Sigma^+ \setminus XX^+)\} \\ &= (\Sigma^+ \setminus XX^+)X^{-1} \cup X^{-1}(\Sigma^+ \setminus XX^+). \end{aligned}$$

This gives us the language S as a complement

$$S = \Sigma^+ \setminus \left((\Sigma^+ \setminus XX^+)X^{-1} \cup X^{-1}(\Sigma^+ \setminus XX^+)\right).$$

□

All operations used in this formula are such, that the rationality of the language is preserved.

Using the binary alphabet $\Sigma = \{a, b\}$ in our examples does not limit the generality, since every finite alphabet can be encoded to binary alphabet preserving the centralizer.

Theorem 3.9. *Let $\Sigma = \{a_1, a_2, \ldots, a_n\}$ be a finite alphabet. With a properly chosen encoding ψ we can encode arbitrary language X over alphabet Σ to the binary alphabet $\{a, b\}$ so, that*

$$\psi(\mathcal{C}(X)) = \mathcal{C}(\psi(X)).$$

Proof. Let's have $\Gamma = \{ab^i a | i = 1, 2, \ldots, n\}$ as a set of code words. For encoding we use the morphism

$$\psi : \Sigma^* \to \Gamma^*, \quad \psi(a_i) = ab^i a.$$

The image of the language $X \subseteq \Sigma^+$ is $\psi(X) \subseteq \Gamma^+$. Now $\mathcal{C}(\psi(X)) \subseteq \Gamma^+$, since $\mathcal{C}(\psi(X)) \subseteq (\mathrm{Pref}(\psi(X)^+) \cap \mathrm{Suf}(\psi(X)^+)) \setminus \{a\} \subseteq \Gamma^+$. Since $\mathcal{C}(X)X = X\mathcal{C}(X)$, the equation

$$\psi(\mathcal{C}(X))\psi(X) = \psi(\mathcal{C}(X)X) = \psi(X\mathcal{C}(X)) = \psi(X)\psi(\mathcal{C}(X))$$

holds and so $\psi(\mathcal{C}(X)) \subseteq \mathcal{C}(\psi(X))$.

On the other hand ψ is injection and so

$$\begin{aligned} \psi^{-1}(\mathcal{C}(\psi(X)))X &= \psi^{-1}(\mathcal{C}(\psi(X)))\psi^{-1}(\psi(X)) = \psi^{-1}(\mathcal{C}(\psi(X))\psi(X)) \\ &= \psi^{-1}(\psi(X)\mathcal{C}(\psi(X))) = X\psi^{-1}(\mathcal{C}(\psi(X))), \end{aligned}$$

i.e. $\psi^{-1}(\mathcal{C}(\psi(X))) \subseteq \mathcal{C}(X)$ and hence $\mathcal{C}(\psi(X)) \subseteq \psi(\mathcal{C}(X))$.

So over all $\mathcal{C}(\psi(X)) = \psi(\mathcal{C}(X))$. □

This means, that if we are studying the centralizer of the language X over a finite arbitrary alphabet, we can encode X into a binary alphabet and study the centralizer of the encoded language.

However, as our next example shows, not all encodings are suitable.

Example 3.10. Let $\Sigma = \{a, b, c, d\}$ be an alphabet, which we encode to binary alphabet $\{e, f\}$ by using the mapping

$$\psi : \Sigma^* \to \{e, f\}^*, \, \psi(a) = ee, \psi(b) = ef, \psi(c) = fe, \psi(d) = ff.$$

If we study the language $X = \Sigma = \{a, b, c, d\}$, we have $\mathcal{C}(X) = X^+ = \Sigma^+$ as its centralizer and the encoded language is

$$\psi(X) = \{ee, ef, fe, ff\} = \{e, f\}^2.$$

However the centralizer of the image $\psi(X)$ is not the same as the image of the centralizer $\mathcal{C}(X)$, since

$$\mathcal{C}(\psi(X)) = \{e, f\}^+ \neq (\{e, f\}^2)^+ = \psi(\mathcal{C}(X)).$$

In this case the crucial reason, why this encoding does not work, is the fact, that the set of code words is not a primitive set, but a power of a smaller set of words.

We may also consider, what happens to the centralizer, if we map the language with a morphism φ.

Theorem 3.11. *Let Σ and Γ be two alphabets and $\varphi : \Sigma^* \to \Gamma^*$ a morphism. If $X \subseteq \Sigma^+$, then $\varphi(\mathcal{C}(X)) \subseteq \mathcal{C}(\varphi(X))$.*

Proof. The proof is straightforward. Since φ is a morphism,

$$\varphi(X)\varphi(\mathcal{C}(X)) = \varphi(X\mathcal{C}(X)) = \varphi(\mathcal{C}(X)X) = \varphi(\mathcal{C}(X))\varphi(X)$$

and this implies

$$\varphi(\mathcal{C}(X)) \subseteq \mathcal{C}(\varphi(X)).$$

□

If the centralizer of language X is not X^+, we obtain following result.

Theorem 3.12. *Let $X \subseteq \Sigma^+$ and $X^+ \subset \mathcal{C}(X)$. If the set $\{\varphi(a) \mid a \in \Sigma\}$ is a code, i.e. the morphism φ is injective, then we have also proper inclusion $\varphi(X)^+ \subset \mathcal{C}(\varphi(X))$.*

Proof. Let us choose a word $w \in \mathcal{C}(X) \setminus X^+$. The image $\varphi(w)$ is now clearly, by Theorem 3.11, in the centralizer of $\varphi(X)$. If the word $\varphi(w)$ was in the language $\varphi(X)^+ = \varphi(X^+)$, this would yield the equation $\varphi(w) = \varphi(w')$ with some word $w' \in X^+$. This, however, gives us a nontrivial identity over alphabet $\{\varphi(a) \mid a \in \Sigma\}$, which was chosen to be a code. As a conclusion we can say that $\varphi(w) \in \mathcal{C}(\varphi(X)) \setminus \varphi(X)^+$ and this implies that the inclusion $\varphi(X)^+ \subseteq \mathcal{C}(\varphi(X))$ is proper. □

3.3 Conway's problem

Concerning the centralizer, there exists several problems that have been open for a long time. One of them is so called Conway's problem introduced on 1971.

Conway's problem. *Is the centralizer of a rational language always rational?*

This problem was introduced by John Horton Conway more than 30 years ago [4] and remained open until it was solved just recently by M. Kunc [14]. For the rational languages it was also an open question, whether or not the centralizer was even recursive or recursively enumerable. Even if only finite languages were studied, this problem remained open. Kunc finally gave an answer also for these questions, and the answer was "no". The centralizer can be non-RE even for finite language. This result concerns both semigroup and monoid centralizers. In his paper Kunc gives an example of such a finite, but rather complicated language.

The positive answer has been proven on certain special cases, for example for codes [8], when $|X| \leq 2$ in [3], or when $|X| = 3$ in [8].

About the centralizer of a rational language we know from [12], that its complement is recursively enumerable. The fixed point approach, introduced later, will give us the semialgorithm for recognizing the complement of the centralizer. For a word in the centralizer this semialgorithm is however not guaranteed to halt.

4 Singular languages

In this chapter we consider the centralizer of a finite singular language. The notion of (left) singular languages was introduced by Ratoandromanana [21]. The language X is called *(left) singular* if there exists a word $v \in X$ such that $v\Sigma^* \cap (X \setminus \{v\})\Sigma^* = \emptyset$. We also say, that this word v is *singular in* X. This means, that v is not a proper prefix of any other word in X and none of words in X is a proper prefix of v. Third way to express this is that v is incomparable in X with respect to the (proper) prefix relation.

In [8] Karhumäki, Latteux and Petre define a slightly different notion. For a language $X \subseteq \Sigma^+$ they call word $x \in X$ *weakly singular* in X if $xX^* \cap (X \setminus \{x\})X^* = \emptyset$. The language X is called *weakly singular* if it has a weakly singular word x.

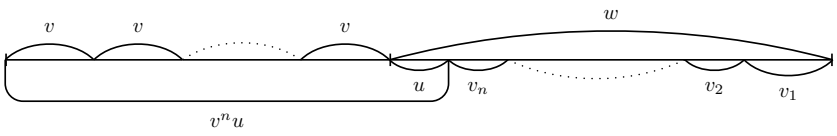
The notions of weakly singular and singular languages are related to each other in the same way as notions of code and prefix code are to each others. In fact the language L is a prefix code if and only if every word in L is singular in L and the language L is a code if and only if every word in L is weakly singular in L.

Theorem 4.1. *Let the word v be singular in X. If $XY = YX$ and $w \in Y$, then for some integer $n \geq 0$ there exists words $t \in X^n$ and $u \in \mathrm{Suf}(X)$ such, that $w = ut$ and $uX^n \subseteq Y$.*

Proof. If $w \in Y$ and v is singular in X , then equation $XY = YX$ implies $vw \in YX$ and $vwv_1^{-1} \in Y$ for some $v_1 \in X$. Repeating this n times we get

$$v^n w(v_n \cdots v_2 v_1)^{-1} \in Y, \qquad v_i \in X,$$

where $t = v_n \cdots v_2 v_1$ and $w = ut$. Then $v^n u \in Y$ for some integer $n \geq 0$ and word $u \in \mathrm{Suf}(X) \cap \mathrm{Pref}(w)$. Since v is singular in X, we see that for every $s \in X^n$



$$v^n us \in YX^n = X^nY \implies us \in Y.$$

In other words, $uX^n \subseteq Y$. □

Since $\mathcal{C}(X)$ is a semigroup, we have even $uX^nX^* \subseteq \mathcal{C}(X)$.

For every proper suffix $u_i \in \mathrm{Suf}(X)$ (including the empty word 1) there either exists a minimal integer n_i, for which $u_iX^{n_i} \subseteq \mathcal{C}(X)$, or $u_iX^n \not\subseteq \mathcal{C}(X)$ for every integer $n \geq 0$. The Theorem 4.1 implies that every word $w \in \mathcal{C}(X)$ belongs to a set $u_iX^{n_i}X^*$, where u_i is such that the integer n_i is minimal.

We conclude the following result from [6].

Theorem 4.2. *A finite singular language X has a rational centralizer. Moreover the centralizer is even finitely generated.*

Proof. If the language X is finite, then the set of its proper suffixes is also finite. If we use the notation I for the set of indices i of those suffixes u_i which have the minimal number n_i mentioned above, then we can say

$$\begin{aligned} \mathcal{C}(X) &= \bigcup_{i\in I} u_iX^{n_i}X^* \\ &= \underbrace{\left(\bigcup_{i\in I} u_iX^{n_i}\right)}_{=G} X^* \\ &= GX^* \end{aligned}$$

Here the language G is finite and $X \subseteq G$, since if $u_0 = 1$, then $n_0 = 1$ and hence $u_0X^{n_0} = 1 \cdot X = X \subseteq G$.

Since $\mathcal{C}(X)$ is semigroup and X is included in G, we get

$$\mathcal{C}(X) = \mathcal{C}(X)^+ = (GX^*)^+ = (X + G)^+ = G^+.$$

□

Example 4.3. In the language $X = \{a, bb, aba, bab, bbb\}$ the word bab is singular. In this case the set of proper suffixes is $\{1, a, b, ab, ba, bb\}$. We will discuss all of these words separately.

$u_0 = 1 :\ 1 \cdot X \subseteq \mathcal{C}(X) \implies n_0 = 1$

$u_1 = a :\ a \in X \subseteq \mathcal{C}(X) \implies n_1 = 0$

$u_2 = b :\ b \cdot a^n \cdot a \notin X\mathcal{C}(X) = \mathcal{C}(X)X \implies b \cdot a^n \notin \mathcal{C}(X), \quad \forall n \in \mathbb{N}.$
This means that $2 \notin I$.

$u_3 = ab :\ a \cdot ab \cdot (bab)^n \notin \mathrm{Suf}(X^+) \implies aab(bab)^n \notin \mathcal{C}(X)$
$\implies aab(bab)^n \notin X\mathcal{C}(X) \implies ab(bab)^n \notin \mathcal{C}(X), \quad \forall n \in \mathbb{N}.$
Hence $3 \notin I$.

$u_4 = ba :\ ba \cdot a^n \cdot a \notin X\mathcal{C}(X) \implies ba \cdot a^n \notin \mathcal{C}(X), \quad \forall n \in \mathbb{N},$
and hence $4 \notin I$.

$u_5 = bb :\ bb \in X \subseteq \mathcal{C}(X) \implies n_5 = 0.$

Hereby $I = \{0, 1, 5\}$ and $G = \bigcup_{i \in I} u_i X^{n_i} = 1 \cdot X + a + bb = X$. This gives us the finitely generated centralizer

$$\mathcal{C}(X) = GX^* = XX^* = X^+.$$

Example 4.4. As the second example we consider the language $X = \{a, ab, ba, bb\}$, for which we already mentioned that the centralizer is $\mathcal{C}(X) = \Sigma^+ \setminus \{b\} \neq X^+$. The set of proper suffixes is $\{1, a, b\}$. We conclude:

$u_0 = 1 :\ 1 \cdot X \subseteq \mathcal{C}(X) \implies n_0 = 1.$

$u_1 = a :\ a \in X \subseteq \mathcal{C}(X) \implies n_1 = 0.$

$u_2 = b :\ b \notin \mathcal{C}(X) \implies n_2 \neq 0$
$b \cdot a \in X \subseteq \mathcal{C}(X)$,
$b \cdot ab \in \mathcal{C}(X)$,
$b \cdot ba \in X^2 \subseteq \mathcal{C}(X)$
$b \cdot bb \in \mathcal{C}(X)$
$bX \subseteq \mathcal{C}(X) \implies n_2 = 1$

Hence $I = \{0, 1, 2\}$ and $G = 1 \cdot X + a + bX = \{a, ab, ba, bb, bab, bbb\}$. So the centralizer is indeed

$$\mathcal{C}(X) = GX^* = G^+ = \{a, ab, ba, bb, bab, bbb\}^+ = \Sigma^+ \setminus \{b\}.$$

5 Conway's problem for 4-element sets

As we mentioned, Conway's problem has previously been proven to have positive answer for languages with at most three elements, [3] and [8]. In some sense these cases always have a trivial centralizer, since it is always either ρ^+ for some primitive word ρ or X^+, where X is the language considered.

Conway's problem for four element sets is much more complicated, since in this case the centralizer doesn't have to be in one of the forms mentioned before. One example, which was already mentioned before, is the language $X = \{a, ab, ba, bb\}$ which has the centralizer $\mathcal{C}(X) = \Sigma^+ \setminus \{b\} = \{a, ab, ba, bb, bab, bbb\}^+$. There are also other such examples, which can not be obtained as morphic images of each others. Namely all languages of form

$$X_n = \{a, bb, ab(bb)^n, (bb)^n ba\}, \quad n \geq 0,$$

have the centralizer $\mathcal{C}(X_n) = \{a, bb, ab(bb)^n, (bb)^n ba, (bb)^n bab(bb)^n, bbb(bb)^n\}^+$.

We analyze languages with four elements and find the solution on Conway's problem for most of those. First we recall a result from [13].

We say that the language X is *branching*, if it has at least two words starting with different letters.

Theorem 5.1. *For any non-periodic language X, with $1 \notin X$, there exists a branching language X' such that the centralizer $\mathcal{C}(X)$ is rational if and only if the centralizer $\mathcal{C}(X')$ is rational.*

Proof. If X is branching then we choose $X' = X$.

If X is not branching, we can assume, that $X = aX_1$ for some letter $a \in \Sigma$ and some language X_1. Since $\mathcal{C}(X) \subseteq \mathrm{Pref}(X^+)$, we know that $\mathcal{C}(X) = aY$ for some $Y \subseteq \Sigma^*$ and so

$$aX_1aY = X\mathcal{C}(X) = \mathcal{C}(X)X = aYaX_1.$$

This means that also $X_1aYa = YaX_1a$, i.e. $Ya \subseteq \mathcal{C}(X_1a) = Za$ for some language $Z \in \Sigma^*$. Further this yields $Y \subseteq Z$.

If we do the same reasoning for the language X_1a and the centralizer $\mathcal{C}(X_1a) = Za$, we see that

$$aZ \subseteq \mathcal{C}(aX_1) = aY.$$

From this we obtain $Z \subseteq Y$ and together with $Y \subseteq Z$ we have $Y = Z$. This means that also $Ya = Za = \mathcal{C}(X_1a)$. Now we have the result

$$\mathcal{C}(aX_1) = aY = a((Ya)a^{-1}) = a((Za)a^{-1}) = a(\mathcal{C}(X_1a)a^{-1}).$$

This means that the centralizer of the language $X = aX_1$ is rational if and only if the centralizer of X_1a is rational. If X_1a is branching, then we choose $X' = X_1a$. If not, we can apply this method so many times, that the result will be branching. The branching language will be finally reached at some point, since the language X is non-periodic. □

Prefix-relations between words in the language X can be represented as a graph. In this graph an arrow from word u to the word v means, that u is a prefix of v. For example the prefix-graph of the language $X = \{a, aa, ab, bb, aba, bba\}$ is presented in the Figure 1.

For the language $X = \{\alpha, \beta, \gamma, \delta\}$ with four elements there exists nine different possible prefix-graphs (up to renaming of elements). See Figure 2.

If the language is periodic, i.e. $X \subseteq \rho^+$ for some primitive word ρ, then the centralizer is ρ^+, essentially due to [13]. However, if the language is not periodic, then the first four cases, the cases where the graph is a tree with one root, can be reduced to the last five cases by Theorem 5.1. The last four cases all have singular elements and by Theorem 4.2 they have finitely generated centralizers. Now only one of these nine cases, the fifth one, needs still to be considered.

By Theorem 5.1 we can assume, that α and γ start with different letters, i.e. $\mathrm{Pref}_1(\alpha) \neq \mathrm{Pref}_1(\gamma)$. Assume $\alpha = au$ and $\gamma = bv$ for some $a, b \in \Sigma$ and $u, v \in \Sigma^*$. We can consider this case as three different subcases.

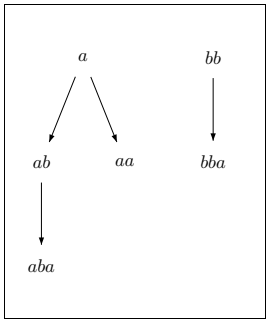


Figure 1: An example of a prefix-graph.

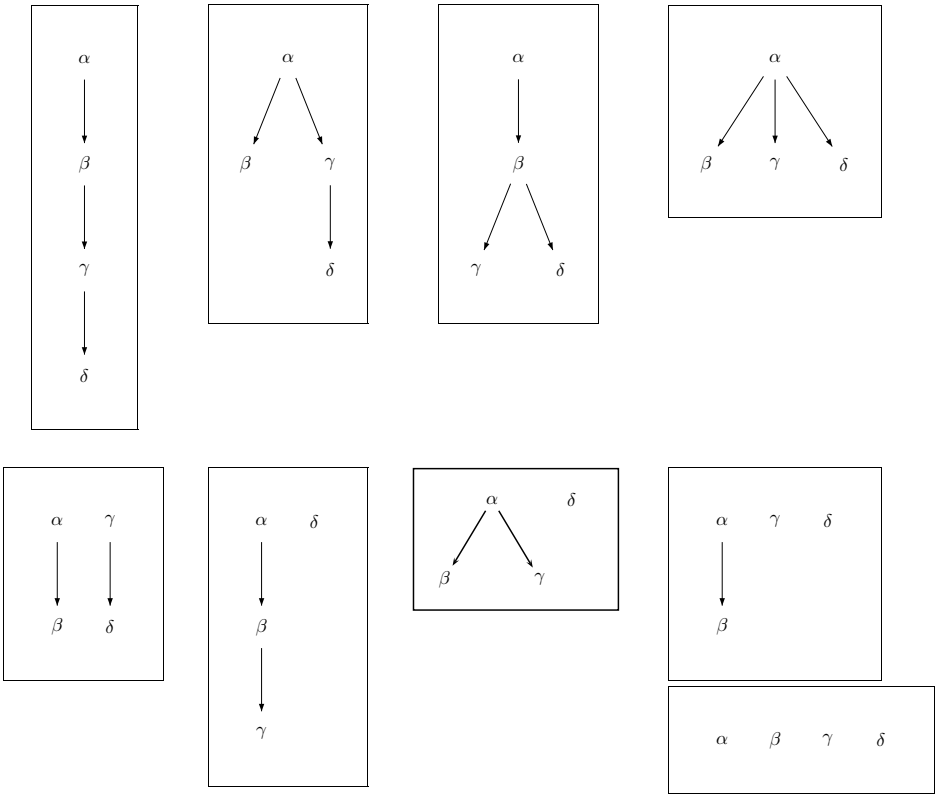


Figure 2: Prefix-graphs of all 4-element languages.

First of all we assume that $\beta = \alpha bx = aubx$ and $\delta = \gamma ay = bvay$ for some $x, y \in \Sigma^*$. Now we can use words α and γ in the same way as we used the singular word in Theorem 4.2. Let $w \in \mathcal{C}(x)$. If $\mathrm{Pref}_1(w) = a$, then for some integer n, $z \in \mathrm{Pref}(X^+) \cap \mathrm{Suf}(X)$ and $v_1, \ldots, v_n \in X$ we have $w = zv_1 \cdots v_n$ and $\alpha^n z \in \mathcal{C}(X)$. This implies that $zX^n \subseteq \mathcal{C}(X)$, since all words in $\alpha^n z X^n$ have unique left factor α^n in X^n. Similarly, if $\mathrm{Pref}_1(w) = b$, we can use the word $\gamma = bv$ instead of the word α. As in Theorem 4.2 this implies that $\mathcal{C}(X)$ is finitely generated.

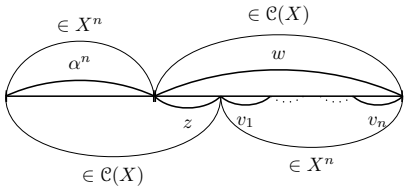


Figure 3: $\alpha^n w = \alpha^n z \cdot v_1 \cdots v_n \in X^n \mathcal{C}(X) = \mathcal{C}(X) X^n$

In the second subcase we assume, that $\beta = \alpha ax = auax$ and $\delta = \gamma by = bvby$. In this case we have to use both words α and γ. If $\mathrm{Pref}_1(w) = a$, then for some integer n we have $w = zv_1 \cdots v_n$ and $\gamma(\alpha\gamma)^{\lfloor \frac{n}{2} \rfloor} z \in \mathcal{C}(X)$ or $(\alpha\gamma)^{\frac{n}{2}} z \in \mathcal{C}(X)$, depending on the parity of n. Similarly, if $\mathrm{Pref}_1(w) = b$, for some n we have $w = zv_1 \cdots v_n$ and $\alpha(\gamma\alpha)^{\lfloor \frac{n}{2} \rfloor} z \in \mathcal{C}(X)$ or $(\gamma\alpha)^{\frac{n}{2}} z \in \mathcal{C}(X)$. Now these words have again unique left factors in X^n yielding once more that $zX^n \subseteq \mathcal{C}(X)$. Like before, the centralizer of X is finitely generated.

The third subcase, where both β and δ continue with the same letter after their prefixes α and γ, is harder. If, for example, $\alpha = au$, $\gamma = bv$, $\beta = aubx$ and $\delta = bvby$ for some $a \neq b$ and $u, v, x, y \in \Sigma^*$, then all words $w \in \mathcal{C}(X) \cap a\Sigma^*$ can easily be handled using α as before, but then for words $w \in \mathcal{C}(X) \cap b\Sigma^*$ the case remains still open.

6 Fixed point approach

In this chapter we introduce a method for finding the centralizer of a given language. For a rational language X this so called fixed point approach, see [7] and [6], can be implemented with a computer using some software such as *Grail+* [22].

The main idea is to have a mapping $\varphi : \wp(\Sigma^+) \to \wp(\Sigma^+)$, which has the centralizer $\mathcal{C}(X)$ as its maximal fixed point. Our mapping is such, that when we start with language X_0 including the centralizer and iterate by using that mapping repeatedly, we eventually get the centralizer.

Let X be a given 1-free language. Let us define a sequence of languages X_i by setting recursively

$$X_0 = \mathrm{Pref}(X^+) \cap \mathrm{Suf}(X^+) \tag{5}$$

and

$$X_{i+1} = X_i \setminus \left(X^{-1}(XX_i\Delta X_iX) \cup (XX_i\Delta X_iX)X^{-1}\right), \qquad i \geq 0. \tag{6}$$

We will use notation

$$Z_0 = \bigcap_{i\geq 0} X_i \tag{7}$$

for the infinite intersection of all languages X_i.

Theorem 6.1. *Language Z_0 is the centralizer $\mathcal{C}(X)$ of given language X.*

Proof. Since, for every $i \geq 0$, we get language X_{i+1} by taking some elements away from the previous language X_i, it is clear, that $X_{i+1} \subseteq X_i$. Theorem 3.6 gives, that $\mathcal{C}(X) \subseteq X_0$. Now if $\mathcal{C}(X) \subseteq X_i$ for some index i, then

$$\mathcal{C}(X)X = X\mathcal{C}(X) \subseteq XX_i \cap X_iX,$$

and hence

$$\mathcal{C}(X)X \cap (XX_i\Delta X_iX) = \emptyset,$$

and further

$$\mathcal{C}(X) \cap \left(X^{-1}(XX_i\Delta X_iX) \cup (XX_i\Delta X_iX)X^{-1}\right) = \emptyset.$$

This means that also $\mathcal{C}(X) \subseteq X_{i+1}$. By induction $\mathcal{C}(X) \subseteq X_i$ for every index $i \geq 0$ and hence the centralizer is also included in the intersection of languages X_i, i.e. $\mathcal{C}(X) \subseteq Z_0$.

On the other hand $Z_0X = XZ_0$ and $Z_0 \subseteq \mathcal{C}(X)$ by the maximality of $\mathcal{C}(X)$. If Z_0X and XZ_0 were not equal, then there would exist a word $w \in Z_0$, such that either $wX \not\subseteq XZ_0$ or $Xw \not\subseteq Z_0X$. By symmetry, assume the previous. By the definition of Z_0 this would mean, that beginning from some index k there would be $wX \not\subseteq XX_i$, when $i \geq k$. However $w \in Z_0 \subseteq X_i$ for every $i \geq 0$, especially for k, and hence $X_kX \neq XX_k$. This would mean that $w \in (XX_k\Delta X_kX)X^{-1}$ and hence $w \notin X_{k+1}$ and $w \notin Z_0$, which contradicts the assumption.

This proves, that $Z_0 = \mathcal{C}(X)$. □

The formula (6) gives us now the mapping

$$\varphi : \wp(\Sigma^+) \to \wp(\Sigma^+),\ \varphi(Y) = Y \setminus \left(X^{-1}(YX\Delta XY) \cup (YX\Delta XY)X^{-1}\right). \quad (8)$$

The fixed points of this mapping are exactly those languages Y fulfilling the condition $YX\Delta XY = \emptyset$, i.e. the languages satisfying the equation $XY = YX$. Hence the centralizer $\mathcal{C}(X)$, as the maximal language commuting with X, is the maximal fixed point of mapping φ.

The centralizer of X can be found by using this mapping iteratively, until $XX_i\Delta X_iX = \emptyset$, i.e. until $X_{i+1} = X_i$. For many rational languages X this gives the centralizer after just few (2–5) iteration steps. However the centralizer can't be reached with a finite number of steps in all cases. Unfortunately there are cases that reach the centralizer only as the limit given in formula (7).

If the centralizer of a rational language is reached in a finite number of steps, then it is a finite intersection of languages achieved from rational language with rational

operations, and hence it is rational itself. However, if the centralizer is achieved only as the limit, then there is no guarantee that the rationality is preserved, since the infinite intersection might lose the rationality. For example languages

$$A_i = \{a^{2^k} \mid 1 \leq k \leq i\} \cup a^{2^i}a^*, \qquad i = 1, 2, 3, \ldots$$

are all rational, but their intersection

$$A = \bigcap_{i=1}^{\infty} A_i = \{a, a^2, a^4, a^8, a^{16}, \ldots\} = \{a^{2^i} \mid i = 1, 2, 3, \ldots\}$$

is not.

This all means, that the fixed point approach does not give final answer for the question on the rationality of the centralizer of rational language. For the complement of the centralizer of a rational language we, however, obtain the following result.

Theorem 6.2 ([10], [12]). *The complement of the centralizer $\mathcal{C}(X)$ of the rational language X is recursively enumerable.*

Proof. The fixed point approach gives us a semialgorithm, which tells us, whether the given word is in the language $\Sigma^+ \setminus \mathcal{C}(X)$. This semialgorithm halts for every input word from the language $\Sigma^+ \setminus \mathcal{C}(X)$, since for each such word there exists an index $k \geq 0$ for which $w \notin X_i$, whenever $i \geq k$. However, if the input word is in the centralizer, we can not be sure that the procedure halts. □

If the algorithm halts, then the situation is simple. In the case that algorithm does not halt, we can still make some conclusions as follows.

We know about the language S introduced in Chapter 3, that

$$X^+ \subseteq S \subseteq \mathcal{C}(X).$$

Since S can be computed by using the formula in Theorem 3.8, we can compare X^+ and S. If these are different languages, then we know that $X^+ \subset \mathcal{C}(X)$ and at least X^+ is not the centralizer.

Another way tho study the centralizer is to study so called branching points. First we define two sets connected to the language X. Let Σ be an alphabet including

at least two letters and X a 1-free language over this alphabet. Now we call word $w \in \mathrm{Pref}(X^+)$ a *branching point*, if $wa, wb \in \mathrm{Pref}(X^+)$ for two distinct letters a and b. The branching point w is said to be *critical*, if $w \notin X^+$. For given language X we denote the set of all branching points with B and the set of critical points with C. As in Chapter 5, we say that language X is branching, if it has words starting with different letters. The next result tells us something about the question whether X^+ is the centralizer of X.

Theorem 6.3 ([13]). *Let $X \subseteq \Sigma^+$ be a finite branching language. If $C = \emptyset$, then $\mathcal{C}(X) = X^+$.*

Proof. The centralizer is a semigroup and hence, if $z \in \mathcal{C}(X)$ and $x \in X \subseteq \mathcal{C}(X)$, then $zx \in \mathcal{C}(X)$. This implies also that $zx \in \mathrm{Pref}(X^+)$. Especially $z\,\mathrm{Pref}_1(X) \subseteq \mathrm{Pref}(X^+)$ implying $z \in B$. The word z is not critical, since we assumed that $C = \emptyset$. This means $z \in X^+$ and further $\mathcal{C}(X) = X^+$. □

Theorem 6.3 says that if the set of critical points of a finite language X is empty, then the centralizer of X is X^+. This claim can not be turned around, since there exists finite sets with nonempty set of critical points, but for which the centralizer is X^+. For example the language $X = \{a, bb, aba, bab, bbb\}$ is such a language. Here the word ab is critical, since $aba, abb \in X^+ \subseteq \mathrm{Pref}(X^+)$, but $ab \notin X^+$. However, in example 4.3 we showed that the centralizer of this language is X^+.

The recursive formula of fixed point approach can be slightly simplified. However the algorithm itself, its result or languages X_i don't chance at all.

Theorem 6.4. *The formula of the fixed point approach can be written without symmetric difference Δ in the form:*

$$X_{i+1} = X_i \setminus \left(X^{-1}\left(XX_i \setminus X_iX\right) \cup \left(X_iX \setminus XX_i\right)X^{-1}\right).$$

Proof. Symmetric differences in the original formula can be written as unions of ordinary differences

$$\begin{aligned} X_{i+1} &= X_i \setminus \left(X^{-1}(XX_i \Delta X_iX) \cup (XX_i \Delta X_iX)X^{-1}\right) \\ &= X_i \setminus \Big(X^{-1}\left(\left(XX_i \setminus X_iX\right) \cup \left(X_iX \setminus XX_i\right)\right) \\ &\qquad \cup \left(\left(XX_i \setminus X_iX\right) \cup \left(X_iX \setminus XX_i\right)\right)X^{-1}\Big). \end{aligned}$$

Right and left quotient can be taken from both parts of the union separately. Then we obtain

$$\begin{aligned} X_{i+1} &= X_i \setminus \big(X^{-1}(XX_i \setminus X_iX) \cup X^{-1}(X_iX \setminus XX_i) \\ &\qquad \cup (XX_i \setminus X_iX)X^{-1} \cup (X_iX \setminus XX_i)X^{-1}\big) . \end{aligned}$$

Next we notice, that since $X_iX \setminus XX_i$ does not include any words from XX_i and respectively $XX_i \setminus X_iX$ does not include words from X_iX, the left and right quotients give us

$$X^{-1}(X_iX \setminus XX_i) \cap X_i = (XX_i \setminus X_iX)X^{-1} \cap X_i = \emptyset.$$

Hence corresponding parts can be ignored in the formula and only the formula

$$X_{i+1} = X_i \setminus \big(X^{-1}\,(XX_i \setminus X_iX) \cup (X_iX \setminus XX_i)\,X^{-1}\big)$$

remains. □

If we now compare the mapping, used in the fixed point approach, in its new form

$$\varphi(Y) = Y \setminus \big(X^{-1}(XY \setminus YX) \cup (YX \setminus XY)X^{-1}\big)$$

with the formula of language S

$$S = \Sigma^+ \setminus \big(X^{-1}(\Sigma^+ \setminus XX^+) \cup (\Sigma^+ \setminus XX^+)X^{-1}\big) ,$$

we can see some similarities. Formulas are even more similar, if we write language S as

$$S = \Sigma^+ \setminus \big(X^{-1}(X\Sigma^+ \setminus X^+X) \cup (\Sigma^+X \setminus XX^+)X^{-1}\big) .$$

We can now think that mapping φ is a more accurate version of the formula of language S. The upper bound of the centralizer is dropped from Σ^+ to language Y, which usually is one of languages X_i. Also the language X^+ is replaced with Y. This means, that words are removed more warily. In the formula of S we might remove also some words from the centralizer.

Language X_{i+1} may also be given as a similar kind of set as S in formula (2).

Theorem 6.5. *The language X_{i+1} can be written in form*

$$X_{i+1} = \{w \in X_i \,|\, wX \subseteq XX_i \ \textit{and} \ Xw \subseteq X_iX\}$$

for each i in $\mathbb{N}$.

Proof. This claim is proven the same way as in Theorem 3.8. Let $i \in \mathbb{N}$ and

$$T = \{w \in X_i \,|\, wX \subseteq XX_i \ \text{and} \ Xw \subseteq X_iX\}.$$

Then

$$\begin{aligned}
w \in T &\iff w \in X_i \quad \text{and} \quad (\forall a \in X)(wa \in XX_i \ \wedge \ aw \in X_iX) \\
w \in X_i \setminus T &\iff w \in X_i \quad \text{and} \quad (\exists a \in X)(wa \notin XX_i \ \vee \ aw \notin X_iX) \\
&\iff w \in X_i \quad \text{and} \\
&\qquad (\exists a \in X)(wa \in X_iX \setminus XX_i \ \vee \ aw \in XX_i \setminus X_iX).
\end{aligned}$$

Hence

$$X_i \setminus T = \big((X_iX \setminus XX_i)X^{-1} \cup X^{-1}(XX_i \setminus X_iX)\big) \cap X_i$$

and

$$T = X_i \setminus \big((X_iX \setminus XX_i)X^{-1} \cup X^{-1}(XX_i \setminus X_iX)\big) = X_{i+1}.$$

□

7 Examples

Next we will discuss some rational languages as examples and try to find the centralizer for them. By these cases we prove, that there exists some languages in each of the classes $X^+ = S = \mathcal{C}(X)$, $X^+ \subset S = \mathcal{C}(X)$, $X^+ = S \subset \mathcal{C}(X)$ and $X^+ \subset S \subset \mathcal{C}(X)$. We also use these examples to illustrate, how the fixed point approach works, and to show some ways to find the centralizer by hand. Some of these languages are finite and some of them are infinite. We will also verify whether the centralizer is X^+, S or something else. These cases are discussed with different precision. Some of them are handled in more details and some of them are just determined with the computer.

First we introduce a property, which holds for every singular language X.

Theorem 7.1. *Let X be a language over the alphabet Σ. If X is (left or right) singular, then $X^+ = S$.*

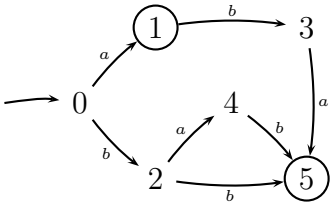
Proof. Let $w \in X$ and $t \in S \setminus X^+$. Then $wt \in XX^+$, i.e. word wt has a X-factorization such that the first factor is not w. This means, that the leftmost factor in this factorization is either proper prefix of w or some word wu, such that $t = us$ for some $s \in X^+$. Now if X is singular, we can choose w to be a singular word in X and we see that $S \setminus X^+$ must be empty.

The corresponding claim with suffixes can be proven similarly. □

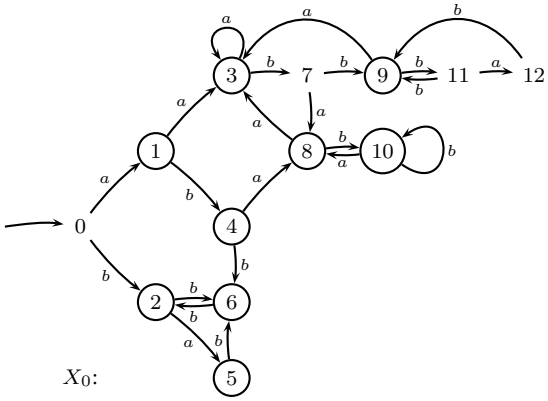
7.1 The case $X^+ = S = \mathcal{C}(X)$

Now we will discuss the finite language $X = \{a, bb, aba, bab\}$ as an example of a language having $X^+ = S = \mathcal{C}(X)$. Theorem 7.1 clearly implies, that $X^+ = S$, since the word bb is singular in X.

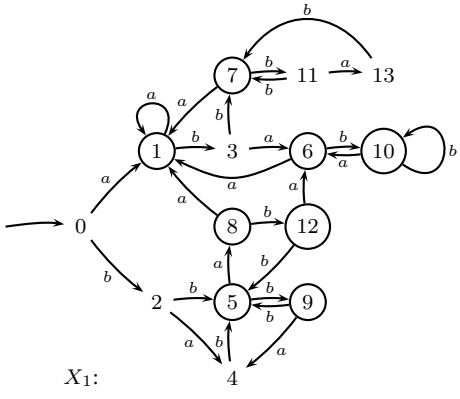
The centralizer is found with the fixed point approach and it is reached already on the third step. The languages X_i corresponding to the steps of iteration can be recognized by deterministic finite automata given in following graphs. The states of automata are enumerated beginning with the initial state 0. Final states are circled. The equality $X_3 = X^+$ was checked with a computer by comparing the minimized deterministic finite automata recognizing these languages.



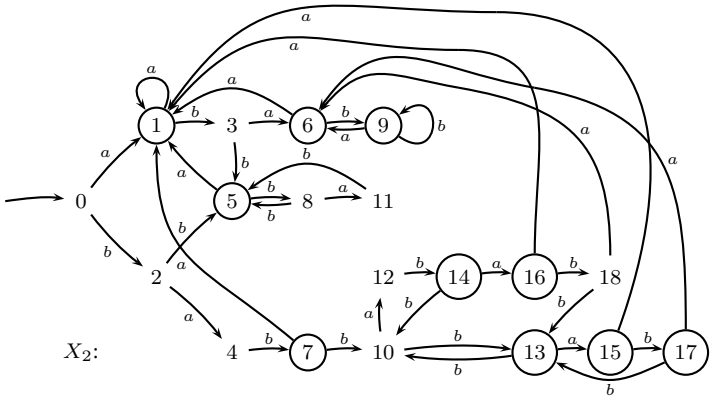
$X = \{a, bb, aba, bab\}$

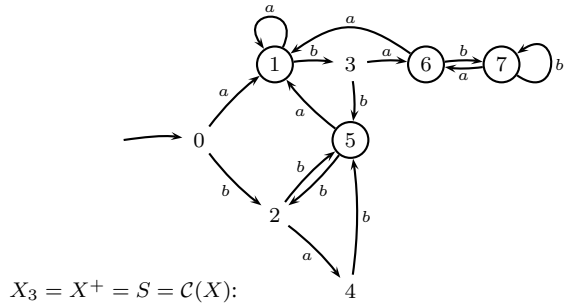


X_0:



X_1:





If we take a closer look at the automata, we can see, that during the iteration the states and transitions form some directed subgraphs which remain unchanged to the next steps. For example the graph formed by states 3,7,8,9,10,11 and 12 in automaton recognizing X_0 is preserved in next steps. Only the numbers of states change. The corresponding states in X_1 in respective order are 1,3,6,7,10,11 and 13, in X_2 they are states 1,3,6,5,9,8 and 11. While iteration goes on, the number of states usually grows, since we get more specific "rules" defining the words in the centralizer. On the other hand, when the centralizer is finally reached, the number of states often drops quite low, since in many cases the centralizers seem to be rather simple.

7.2 The case $X^+ \subset S = \mathcal{C}(X)$

Finding a finite example of language X having $X^+ \subset S$ seems to be difficult. We could not find any and we don't know if that is possible. That is the reason, why

we choose the infinite rational language $X = \Sigma a\Sigma^* a + b\Sigma^* b\Sigma$ over the alphabet $\Sigma = \{a, b\}$ as an example of language having this proper inclusion. The fixed point approach gives the centralizer of X in three steps. As the Table 1 shows, the number of the states does not drop on the last step as it did in the previous example.

	final st.	states	transitions
X	5	13	25
X_0	3	5	9
X_1	7	18	35
X_2	9	22	43
X_3	11	26	51
X_4	11	26	51
S	11	26	51
X^+	6	14	27

Table 1: Number of states, final states and transitions in automata recognizing the language $X = \Sigma a\Sigma^* a + b\Sigma^* b\Sigma$, the steps X_i of the algorithm and languages S and X^+.

Let's find the centralizer with the fixed point approach. This illustrates, step by step, how the fixed point approach proceeds and at the same time it also gives us the equality $S = \mathcal{C}(X)$. The first step is to find the starting point of the iteration, i.e. the language $X_0 = \mathrm{Pref}(X^+) \cap \mathrm{Suf}(X^+)$. We know, that always $X^+ \subseteq X_0$ so we start by finding the language $X_0 \setminus X^+$. The length of the shortest word in X^+ is 3. All words in X_0 shorter than this are a, b, aa, ba and bb.

Longer words can be discussed separately depending on the first letter of the word. Let $w \in X_0 \setminus X^+$ and assume that $w \in a\Sigma\Sigma^+$. Now, since the first letter is a, $w \notin \mathrm{Pref}(b\Sigma^* b\Sigma X^*)$ and hence $w \in \mathrm{Pref}(\Sigma a\Sigma^* aX^*)$ which means that $w \in aa\Sigma^+$. On the other hand $w \notin X$ implies, that $w \in aa\Sigma^* b$. Symmetrically, since $w \notin \mathrm{Suf}(X^*\Sigma a\Sigma^* a)$, there must hold $w \in \mathrm{Suf}(X^* b\Sigma^* b\Sigma)$, which implies $w \in aa\Sigma^* bb$. Additionally, since $aa\Sigma^* ab\Sigma^* bb \subseteq X^2$, the word w must be in language $aab^* a^* bb$. This language is entirely included in $X_0 \setminus X^+$.

If we next assume, that $w \in b\Sigma\Sigma^+$, we can make the following conclusions concerning the word w. Since $b\Sigma^*b\Sigma \subseteq X^+$, we get $w \in b\Sigma^*a\Sigma$. Further $w \notin \mathrm{Suf}(X^*b\Sigma^*b\Sigma)$ implying $w \in \mathrm{Suf}(X^*\Sigma a\Sigma^*a)$ and hence $w \in b\Sigma^*aa$. Now $baa + ba\Sigma^*aa \subseteq X^+$, which means that the second letter of w must be b, i.e. $w \in bb\Sigma^*aa$. In the next step we see, that the word in language $bb\Sigma^*aa$ is in the language X^+ if and only if it is of format $b\Sigma^*b\Sigma\Sigma a\Sigma^*a$, since clearly the beginning of the word must be in $b\Sigma^*b\Sigma$ and the last part in $\Sigma a\Sigma^*a$. So, the language we are interested in is $bb\Sigma^*aa \setminus b\Sigma^*b\Sigma\Sigma a\Sigma^*a$. We discuss the part Σ^* between the beginning bb and ending aa in sequences of three letters. After the beginning bb the next three letters must be either aab, abb or bab, since $bb\Sigma\Sigma a\Sigma^*aa + bbbbb\Sigma^*aa \subseteq b\Sigma^*b\Sigma\Sigma a\Sigma^*a$. Further, after aab there can be whichever of these three words, after abb there can be only abb and after bab only bab. Hence

$$w \in bb(aab)^*(abb)^*(1 + \Sigma + \Sigma^2)aa + bb(aab)^*(bab)^*(1 + \Sigma + \Sigma^2)aa.$$

For this language we must still discuss several different cases to find out which of them are included in X^+. For languages included in X^+ the part of format $b\Sigma\Sigma a$ is underlined.

$$\begin{aligned}
bb(aab)^*(abb)^*aa &\subseteq X_0 \setminus X^+ \\
bb(aab)^*(\underline{abb)^+\Sigma aa} &\subseteq X^+ \\
bb(aab)^*\Sigma aa &\subseteq X_0 \setminus X^+ \\
bb(aab)^*(\underline{abb)^+\Sigma\Sigma aa} &\subseteq X^+ \\
bb(\underline{aab)^+\Sigma\Sigma aa} &\subseteq X^+ \\
\underline{bb\Sigma\Sigma a}a &\subseteq X^+ \\
bb(aab)^*(\underline{bab)^+aa} &\subseteq X^+ \\
bb(aab)^*(\underline{bab)^+aaa} &\subseteq X^+ \\
bb(aab)^*(bab)^+baa &\subseteq X_0 \setminus X^+ \\
bb(aab)^*(\underline{bab)^+\Sigma\Sigma aa} &\subseteq X^+
\end{aligned}$$

Now we can conclude

$$\begin{aligned} b\Sigma\Sigma \cap (X_0 \setminus X^+) &= bb(aab)^*(abb)^*aa + bb(aab)^*\Sigma aa + bb(aab)^*(bab)^+baa \\ &= bb(aab)^*(abb)^*aa + bb(aab)^*a(abb)^*aa \\ &\quad + bb(aab)^*b(abb)^*aa \end{aligned}$$

and for the language X_0, the starting point of the iteration, we get the regular expression

$$\begin{aligned} X_0 &= \mathrm{Pref}(X^+) \cap \mathrm{Suf}(X^+) \\ &= X^+ + aab^*a^*bb \\ &\quad + bb(aab)^*(abb)^*aa \\ &\quad + bb(aab)^*a(abb)^*aa \\ &\quad + bbb(abb)^*aa \\ &\quad + a + b + aa + ba + bb. \end{aligned}$$

Next we start doing the iteration steps. In the step from X_0 to X_1 the following words on the left hand side are erased by the corresponding reasons on the right hand side.

a :	$a \cdot bbb \in X_0X$, but $abbb \notin XX_0$.
b :	$aaa \cdot b \in XX_0$, but $aaab \notin X_0X$.
aa :	$babba \cdot aa \in XX_0$, but $babbaaa \notin X_0X$.
bb :	$bb \cdot baaba \in X_0X$, but $bbbaaba \notin XX_0$.
aaa^nbb :	$aaa \cdot aaa^nbb \in XX_0$, but $aaaaaa^nbb \notin X_0X$. $(n \geq 0)$
aab^nbb :	$aab^nbb \cdot bbb \in X_0X$, but $aab^nbbbbb \notin XX_0$. $(n \geq 0)$
$bb(abb)^naa$:	$aabba \cdot bb(abb)^naa \in XX_0$, but $aab(bab)^i \cdot \overbrace{(bab)^jbaa}^{\in X} \notin X_0X$. $(n \geq 0, i+j = n+1)$
$bb(aab)^naa$:	$bb(aab)^naa \cdot baabb \in X_0X$, but $bba(aba)^{n+1}abb \notin XX_0$. $(n \geq 0)$

No other words are erased and the resulting language X_1 is

$$\begin{aligned} X_1 &= X^+ + aab^+a^+bb \\ &\quad + bb(aab)^+(abb)^+aa \\ &\quad + bb(aab)^*a(abb)^*aa \\ &\quad + bbb(abb)^*aa \\ &\quad + ba. \end{aligned}$$

In the same way, when stepping from X_1 to X_2, we erase the following words

ba :	$ba \cdot aaa \in X_1X$, but $baaaa \notin XX_1$.
$bbb(abb)^naa$:	$aaa \cdot bbb(abb)^naa \in XX_1$, but $aaabbb(abb)^naa \notin X_1X$. $(n \geq 0)$
$bb(aab)^naaa$:	$bb(aab)^naaa \cdot bbb \in X_1X$, but $bb(aab)^naaabbb \notin XX_1$. $(n \geq 0)$
$bbaabb(abb)^naa$:	$aaa \cdot bbaabb(abb)^naa \in XX_1$, but $aaabbaabb(abb)^n \notin X_1X$. $(n \geq 0)$
$bb(aab)^naabbaa$:	$bb(aab)^naabbaa \cdot bbb \in X_1X$, but $bb(aab)^naabbaabbb \notin XX_1$. $(n \geq 0)$

The remaining language is

$$\begin{aligned} X_2 &= X^+ + aab^+a^+bb \\ &\quad + bb(aab)^+(abb)^+aa \\ &\quad + bb(aab)^+aabb(abb)^+aa. \end{aligned}$$

Finally the step from X_2 to X_3 erases the last words not contained in the centralizer.

aab^nabb : $aab^nabb \cdot aaa \in X_2X$, but $aab^nbbaaa \notin XX_2$, since the left factor, contained in X, would be one of the words aab^na, aab^nabba or aab^nabbaa, but for the corresponding right factors $bbaaa, aa, a \notin X_2$.
$(n \geq 0)$

aab^naabb : $aab^naabb \cdot aaa \in X_2X$, but $aab^naabbaaa \notin XX_2$ similarly as previously, since $abbaaa, bbaaa, aa, a \notin X_2$.
$(n \geq 0)$

$aaba^nbb$: $bbb \cdot aaba^nbb \in XX_2$, but $bbbaaba^nbb \notin X_2X$ again similarly, since the right factor in X should be a word beginning with letter b, i.e. one of the words $ba^nbb, baaba^nbb$ or $bbaaba^nbb$, but the corresponding left factors $bbbaa, bb$ and b are not in X_2.
$(n \geq 0)$

$aabba^nbb$: $bbb \cdot aabba^nbb \in XX_2$, but $bbbaabba^nbb \notin X_2X$ exactly in the same way as in the other cases.
$(n \geq 0)$

This gives us the language

$$\begin{aligned} X_3 &= X^+ + aabbb^+aaa^+bb \\ &\quad + bb(aab)^+(abb)^+aa \\ &\quad + bb(aab)^+aabb(abb)^+aa. \end{aligned}$$

The fixed point method stops here, since $XX_3 = X_3X$. This can be seen for example

as follows

$$
\begin{aligned}
XX_3 &= X\left(X^+ + aabbb^+aaa^+bb + bb(aab)^+(abb)^+aa + bb(aab)^+aabb(abb)^+aa\right)\\
&= XX^+\\
&\quad +\Sigma a\Sigma^* a \cdot aabbb^+aaa^+bb\\
&\quad +b\Sigma^* b\Sigma \cdot aabbb^+aaa^+bb\\
&\quad +\Sigma a\Sigma^* a \cdot bb(aab)^+(abb)^+aa\\
&\quad +b\Sigma^* b\Sigma \cdot bb(aab)^+(abb)^+aa\\
&\quad +\Sigma a\Sigma^* a \cdot bb(aab)^+aabb(abb)^+aa\\
&\quad +b\Sigma^* b\Sigma \cdot bb(aab)^+aabb(abb)^+aa\\
&= XX^+\\
&\quad +\Sigma a\Sigma^* aaa \cdot bbb^+aaa^+bb\\
&\quad +b\Sigma^* b\Sigma aabb \cdot b^+aaa^+bb\\
&\quad +\Sigma a\Sigma^* abbaa \cdot b(aab)^*(abb)^+aa\\
&\quad +b\Sigma^* b\Sigma bb \cdot (aab)^+(abb)^+aa\\
&\quad +\Sigma a\Sigma^* abbaa \cdot b(aab)^*aabb(abb)^+aa\\
&\quad +b\Sigma^* b\Sigma bb \cdot (aab)^+aabb(abb)^+aa\\
&\subseteq XX^+.
\end{aligned}
$$

The inclusion $X_3X \subseteq XX^+$ can be seen similarly. This implies $X_3 \subseteq S$ and since always $S \subseteq \mathcal{C}(X) \subseteq X_i$, the equality $X_3 = S = \mathcal{C}(X)$ must hold. Additionally clearly $X^+ \neq X_3$, since for example $aabbbaaabb \in X_3 \setminus X^+$, and hence $X^+ \subset S = \mathcal{C}(X)$.

In practice, this all is of course done algorithmically with the computer.

7.3 The case $X^+ = S \subset \mathcal{C}(X)$

For this case it is again easy to find finite examples. One very simple example of a language, for which $X^+ = S$, but the centralizer is something different, is the language $X = \{aa\}$ over the alphabet $\Sigma = \{a\}$. The whole semigroup $\Sigma^+ = a^+$ clearly commutes with X, implying $\mathcal{C}(X) = a^+$. The language $X^+ = (aa)^+$ is

the language of all sequences of a of even length. Since the length of catenation of two words of even length is also even, all words in S have also even length. Hence $X^+ = S \subset \mathcal{C}(X)$.

We call the language $\rho(X)$ a *root* of the language X, if $X = \rho(X)^n$ for some integer n. A root of X is called *minimal*, if it is not a proper power of any other set. A minimal root is not necessarily unique. For example the set $X = \{a^i \mid 0 \leq i \leq 30, i \neq 1, 8, 11, 23\}$ has two different minimal roots, see [2]. If the set X has the unique minimal root, that is called *primitive root* of X. As in the example above, it holds generally, that if X has a proper root, then always

$$X^+ \neq \mathcal{C}(X),$$

since

$$X^+ \subset \rho(X)^+ \subseteq \mathcal{C}(X)$$

for some (minimal) root $\rho(X)$, since naturally $\rho(X)^+$ commutes with $X = \rho(X)^n$.

As another example of type $X^+ = S \subset \mathcal{C}(X)$, we choose the language $X = \{a, ab, ba, bb\}$, which was mentioned already in example 3.7. The fact that $S = X^+$ can be seen either by Theorem 7.1 or with a computer.

For $X = \{a, ab, ba, bb\}$ we have

$$\mathrm{Pref}(X) = \mathrm{Suf}(X) = \{a, b, ab, ba, bb\}.$$

Let's find the language X_0. First we see, that

$$\begin{aligned} \left(\Sigma^2\right)^+ = \{a\,a, ab, ba, bb\}^+ \subseteq X^+ \subseteq \mathrm{Pref}(X^+) \quad \text{and} \\ \left(\Sigma^2\right)^* \Sigma \subseteq X^*\Sigma \subseteq X^* \,\mathrm{Pref}(X) = \mathrm{Pref}(X^+). \end{aligned}$$

Hence $\Sigma^+ = (\Sigma^2)^+ + (\Sigma^2)^*\Sigma \subseteq \mathrm{Pref}(X^+) \subseteq \Sigma^+$ implying $\mathrm{Pref}(X^+) = \Sigma^+$. Corresponding result $\mathrm{Suf}(X^+) = \Sigma^+$ can be obtained similarly. This gives us

$$X_0 = \mathrm{Pref}(X^+) \cap \mathrm{Suf}(X^+) = \Sigma^+.$$

Next we find the centralizer as follows. We claim, that the centralizer is the language $Z = \Sigma^+ \setminus \{b\}$. This language can be divided into two parts depending on

the first letter of the word. Similarly the division can be done by the last letter. This means

$$Z = \Sigma^+ \setminus \{b\} = a\Sigma^* + b\Sigma^+ = \Sigma^* a + \Sigma^+ b.$$

The language Z commutes with X, since

$$ZX = a\Sigma^* X + b\Sigma^+ X = a \cdot \Sigma^* X + b\Sigma \cdot \Sigma^* X = (a + b\Sigma)\Sigma^* X \subseteq XZ$$

and

$$XZ = X\Sigma^* a + X\Sigma^+ b = X\Sigma^* \cdot a + X\Sigma^* \cdot \Sigma b = X\Sigma^*(a + \Sigma b) \subseteq ZX.$$

The centralizer is not the whole set $X_0 = \Sigma^+$, since b is not in it. This can be seen for example by the fact, that $ba \in X_0 X$, but $ba \notin XX_0$. Especially $X^+ \neq \mathcal{C}(X)$, since for example $bbb \in \mathcal{C}(X)$, but $bbb \notin X^+$. Over all we have

$$X^+ = S \subset \mathcal{C}(X).$$

Additionally we note that $\mathcal{C}_*(X) \neq \mathcal{C}_+(X) \cup \{1\}$, since $b \in \mathcal{C}_*(X) = \Sigma^*$.

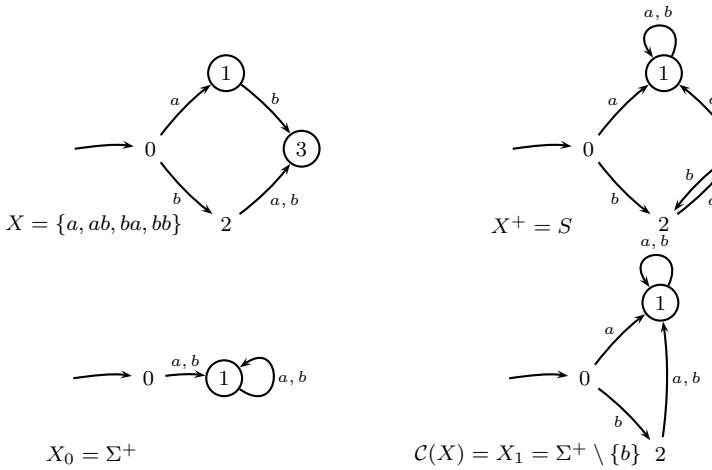


Figure 4: Minimized automata for languages X and X^+ and for iteration steps X_0 and $X_1 = \mathcal{C}(X)$.

7.4 The case $X^+ \subset S \subset \mathcal{C}(X)$

We will still search for a language X, for which proper inclusions $X^+ \subset S \subset \mathcal{C}(X)$ hold. For this case we, again, could not find a finite language as an example. Let's choose

$$X = a\Sigma^+b + b\Sigma^+a.$$

Now

$$X_0 = \mathrm{Pref}(X^+) \cap \mathrm{Suf}(X^+) = (a\Sigma^* + b\Sigma^*) \cap (\Sigma^*a + \Sigma^*b) = \Sigma^+ \cap \Sigma^+ = \Sigma^+.$$

The language X_1 is recognized by the automaton found by the computer and shown in Figure 5. The language given by this automaton can easily be expressed as rational

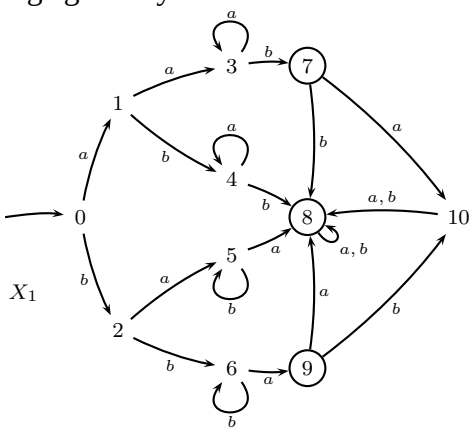


Figure 5: The automaton, which recognizes the language X_1 computed with a computer program *Grail+*.

expression

$$\begin{aligned} X_1 &= aaa^*b + aaa^*bb\Sigma^* + aaa^*ba\Sigma^+ + aba^*b\Sigma^* \\ &+ bbb^*a + bbb^*aa\Sigma^* + bbb^*ab\Sigma^+ + bab^*a\Sigma^*. \end{aligned} \tag{9}$$

The same language can also be given by using its complement as

$$X_1 = \Sigma^+ \setminus (a^+ + b^+ + aba^* + bab^* + aa^+ba + bb^+ab). \tag{10}$$

Also this can be easily seen from the same automaton. We get X_1 as an iteration step of the algorithm since

$$\begin{aligned} a^+ \cdot aab \cap XX_0 &= \emptyset \\ b^+ \cdot bba \cap XX_0 &= \emptyset \\ aba^* \cdot aab \cap XX_0 &= \emptyset \\ bab^* \cdot bba \cap XX_0 &= \emptyset \\ baa \cdot aa^+ba \cap X_0X &= \emptyset \\ abb \cdot bb^+ab \cap X_0X &= \emptyset, \end{aligned}$$

which means

$$a^+ + b^+ + aba^* + bab^* + aa^+ba + bb^+ab \subseteq X^{-1}(XX_0 \Delta X_0X) \cup (X_0X \Delta XX_0)X^{-1}.$$

The next step erases the rest of the words not in the centralizer. The automaton of the next iteration step is illustrated in the Figure 6. The regular expression of the

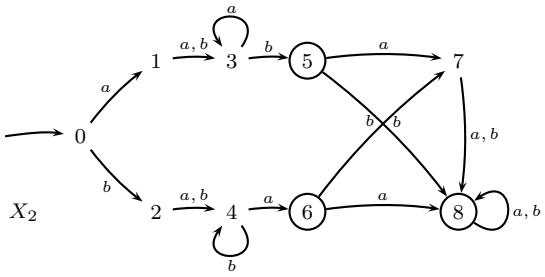


Figure 6: Automaton recognizing X_2. Computed with Grail+.

language X_2 is

$$\begin{aligned} X_2 &= a\Sigma a^*b + a\Sigma a^*ba\Sigma^+ + a\Sigma a^*bb\Sigma^* \\ &+ b\Sigma b^*a + b\Sigma b^*ab\Sigma^+ + b\Sigma b^*aa\Sigma^*. \end{aligned} \tag{11}$$

Also this step can be given as a complement of a rather simple language. From this representation we also see, which words are deleted from X_1, when taking this step.

$$\begin{aligned} X_2 &= \Sigma^+ \setminus (a^+ + b^+ + aba^* + bab^* + aa^+ba + bb^+ab + aba^*ba + bab^*ab) \\ &= X_1 \setminus (aba^*ba + bab^*ab). \end{aligned} \tag{12}$$

As a justification for erasing the language $aba^*ba + bab^*ab$ it is enough to see, that

$$\begin{aligned} aba^*ba \cdot baa \cap XX_1 &= \emptyset \quad \text{and} \\ bab^*ab \cdot abb \cap XX_1 &= \emptyset. \end{aligned}$$

These equalities hold, since if we separate from words in aba^*babaa and bab^*ababb the left factor which is in X, either aba^nb or aba^nbab and either bab^na or bab^naba, the remaining parts $abaa, aa, babb$ and bb are not in X_1, as can be seen by the formula (10).

Now X_2 is the centralizer $\mathcal{C}(X)$, since $XX_2 = X_2X$. We can see that

$$\begin{aligned} X_2X &= a\Sigma a^*b \cdot a\Sigma^+b + a\Sigma a^*b \cdot b\Sigma^+a + a\Sigma a^*ba\Sigma^+ \cdot a\Sigma^+b \\ &+ a\Sigma a^*ba\Sigma^+ \cdot b\Sigma^+a + a\Sigma a^*bb\Sigma^* \cdot a\Sigma^+b + a\Sigma a^*bb\Sigma^* \cdot b\Sigma^+a \\ &+ b\Sigma b^*a \cdot a\Sigma^+b + b\Sigma b^*a \cdot b\Sigma^+a + b\Sigma b^*ab\Sigma^+ \cdot a\Sigma^+b \\ &+ b\Sigma b^*ab\Sigma^+ \cdot b\Sigma^+a + b\Sigma b^*aa\Sigma^* \cdot a\Sigma^+b + b\Sigma b^*aa\Sigma^* \cdot b\Sigma^+a \end{aligned}$$

and here every term is included in XX_2. For example $a\Sigma a^*bb\Sigma^* \cdot a\Sigma^+b \subseteq XX_2$, since $a\Sigma a^*bb \cdot a\Sigma^+b \subseteq XX$ and by formula (12) $a\Sigma a^*b \cdot b\Sigma^+a\Sigma^+b \subseteq XX_2$. This way we get $X_2X \subseteq XX_2$ and since, $X = X^R$ and $X_2 = X_2^R$, it's easy to see, that

$$XX_2 = (X_2^RX^R)^R = (X_2X)^R \subseteq (XX_2)^R = X_2^RX^R = X_2X.$$

The inclusions $X^+ \subset S \subset \mathcal{C}(X)$ will be seen by choosing suitable example words. For example

$$abbbaX = abbba \cdot a\Sigma^+b + abbba \cdot b\Sigma^+a = abbb \cdot aa\Sigma^+b + abb \cdot bab\Sigma^+a \subseteq XX^+$$

and

$$Xabbba = a\Sigma^+b \cdot abbba + b\Sigma^+a \cdot abbba = a\Sigma^+bab \cdot bba + b\Sigma^+aa \cdot bbba \subseteq XX^+,$$

implying, that $abbba \in S$, but clearly $abbba \notin X^+$. On the other hand $abbaa \in \mathcal{C}(X)$, since the automaton in Figure 6 recognizes this word, but $abbaa \notin S$, since for example $abbaa \cdot baa \notin XX^+$.

8 Centralizer as the limit

We give an example about the case in which the fixed point method does not give the centralizer in any finite number of iteration steps. We also analyze the case to find the reason, why we get the centralizer only as the limit. The example is interesting, because it shows that even finite language X with only five words can lead the fixed point approach to an infinite computation. Moreover the centralizer of our example X is still only X^+. We also mention a couple of other cases leading to the infinite iteration loop.

8.1 Defining the case

In this example we use the alphabet $\Sigma = \{a, b\}$ and the finite language $X = \{a, bb, aba, bab, bbb\}$, which has already been considered in some earlier examples.

If we ask, whether $\mathcal{C}(X) = X^+$, we can try to apply two methods given in the end of Chapter 6. We can compare languages X^+ and S. If they were different, would the centralizer $\mathcal{C}(X)$ naturally also be different from X^+. However, if we compute S, we find that $S = X^+$. We see this also by using Theorem 7.1, since word bab is singular word in X.

Another way to test if $\mathcal{C}(X) = X^+$ is to check if the set of critical points C is empty. If $C = \emptyset$, we see that $\mathcal{C}(X) = X^+$, since all elements in centralizer are branching points. However in this case this approach does not tell us that centralizer is X^+, because for example word b is critical point. Word b is branching point, since $b \in \mathrm{Pref}(X^+)$ and $ba, bb \in \mathrm{Pref}(X^+)$. It is also critical, since $b \notin X^+$.

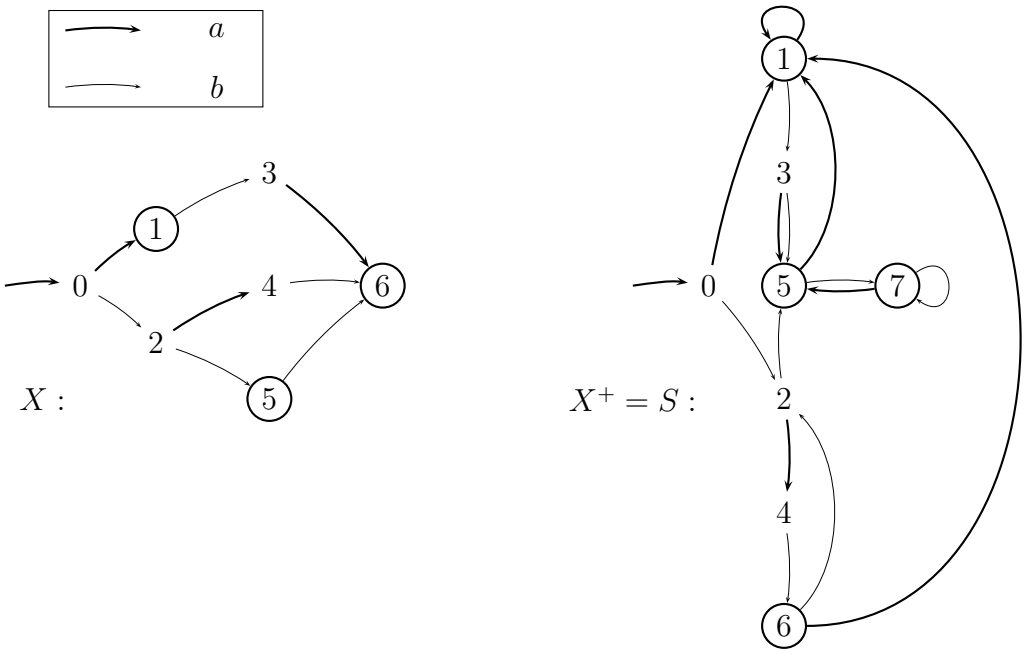


Figure 7: Finite automata recognizing languages X and X^+.

If we try to find the centralizer of X by using fixed point approach, it will turn out quite soon, that the method does not seem to give any result. The numbers of states in the minimized deterministic finite automata of languages X_i keep growing step after step. The corresponding automata grow with certain speed and pattern. So our language seems to be quite difficult to be handled by these methods.

8.2 Fixed point method with Grail+

Let's use Grail+ to produce automaton representing the language $X = \{a, bb, aba, bab, bbb\}$ and apply a few steps of the fixed point method to this automaton. We will examine the minimized automata of languages that we get as iteration steps of the method. We will try to find some common patterns from these automata. The automaton recognizing the first step X_0 of the iteration is given in Figure 8.

The numbers of states, final states and transitions for some iteration steps of language X are given in Table 2. From this table we see that after few steps the growth becomes constant. Every step adds six more states, three of them final, and eleven transitions.

	finalst.	states	transitions		finalst.	states	transitions
X_0	6	8	15	X_{20}	60	121	222
X_1	5	9	17	X_{21}	63	127	233
X_2	6	13	24	X_{22}	66	133	244
X_3	9	19	35	X_{23}	69	139	255
X_4	12	25	46	X_{24}	72	145	266
X_5	15	31	57	X_{25}	75	151	277
X_6	18	37	68	X_{26}	78	157	288
X_7	21	43	79	X_{27}	81	163	299
X_8	24	49	90	X_{28}	84	169	310
X_9	27	55	101	X_{29}	87	175	321
X_{10}	30	61	112	X_{30}	90	181	332
X_{11}	33	67	123	X_{31}	93	187	343
X_{12}	36	73	134	X_{32}	96	193	354
X_{13}	39	79	145	X_{33}	99	199	365
X_{14}	42	85	156	X_{34}	102	205	376
X_{15}	45	91	167	X_{35}	105	211	387
X_{16}	48	97	178	X_{36}	108	217	398
X_{17}	51	103	189	X_{37}	111	223	409
X_{18}	54	109	200	X_{38}	114	229	420
X_{19}	57	115	211	X_{39}	117	235	431

Table 2: Number of states, final states and transitions of automata corresponding to iteration steps for language X.

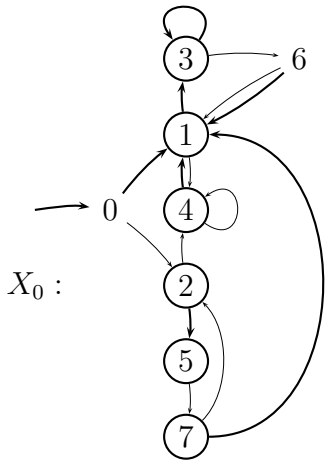


Figure 8: Automaton recognizing the language X_0, the first step of fixed point method

When we draw the automata corresponding to subsequent iteration steps X_i and X_{i+1}, as in figures 9 and 10, we can see clear pattern in the growth. From these figures we see that the automata representing languages X_5 and X_6 are built from two sequence of sets of three states. In the automaton of X_5 the states 2,4,6,8–18 and 20 are in the first sequence and states 1,3,5,19 and 21–28 in the other. In the automaton of X_6 both sequence have got additional set of three states. So there are six new states, including three new final states, and eleven new transitions. On every iteration step automata seem to use same pattern to grow. When number of iteration steps goes to infinity, lengths of both sequence also go to infinity. If the corresponding states are merged together, the result recognizes the language X^+. This of course does not prove anything, but from this we can make a guess that $\mathcal{C}(X) = X^+$.

8.3 The centralizer

Next we shall prove that the centralizer really is $\mathcal{C}(X) = X^+$. This was already shown in example 4.3, but following a different proof will also illustrate, how fixed point approach handles this case. This proof will also help to understand reasons why the computation becomes infinite. Here we will give the languages as regular expressions. We will apply the fixed point method on language X, but instead of making iteration steps one by one we will use the induction to make them all at once. The first thing is to find the starting point of the iteration, $X_0 = \mathrm{Pref}(X^+) \cap \mathrm{Suf}(X^+)$.

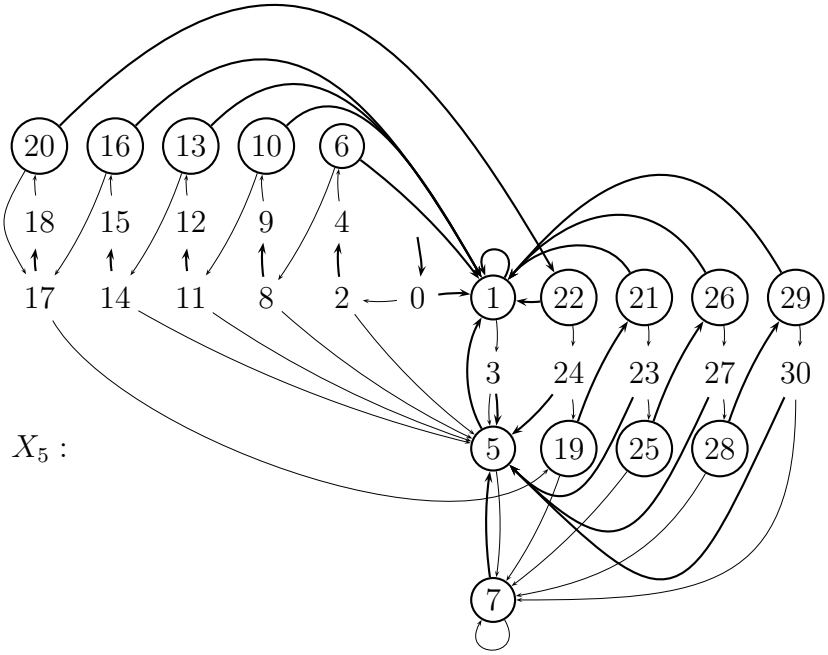


Figure 9: Automaton recognizing X_5.

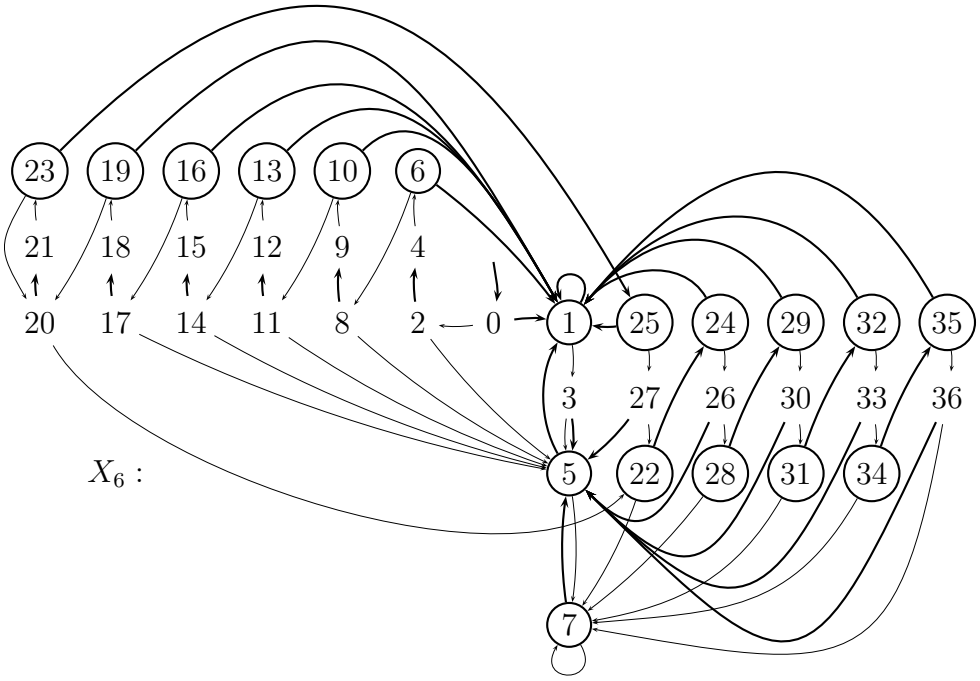


Figure 10: Automaton recognizing X_6.

Lemma 8.1. *The language X_0 can be expressed as*

$$X_0 = X^+ + (bab)^*b(bab)^* + (bab)^*ab(bab)^* + (bab)^*ba(bab)^*.$$

Proof. Let's mark

- $Y_1 = (bab)^*b(bab)^*$,
- $Y_2 = (bab)^*ab(bab)^*$ and
- $Y_3 = (bab)^*ba(bab)^*$.

The inclusion

$$X^+ + Y_1 + Y_2 + Y_3 \subseteq X_0$$

can be seen easily. Naturally $X^+ \subseteq X_0$, since X^+ is subset of both its own prefix and its own suffix. We get inclusion $Y_1 \subseteq X_0$ by noting, that

$$(bab)^*b(bab)^* = b(a\,bb)^*(bab)^* \subseteq \mathrm{Suf}(X)X^* = \mathrm{Suf}(X^+) \quad \text{and} \quad (13)$$

$$(bab)^*b(bab)^* = (bab)^*(bb\,a)^*b \subseteq X^*\,\mathrm{Pref}(X) = \mathrm{Pref}(X^+). \quad (14)$$

That means $Y_1 \subseteq \mathrm{Pref}(X^+) \cap \mathrm{Suf}(X^+) = X_0$. We can also see, that

$$(bab)^*b(bab)^* \cap X^+ = \emptyset,$$

since formula (14) gives the only way to represent words of $(bab)^*b(bab)^*$ in the form $X^*\,\mathrm{Pref}(X)$ and the word b in the end of it is in $\mathrm{Pref}(X)$, but not in X. Similarly from equations

$$(bab)^*ab(bab)^* = ab(bab)^* + ba(bb\,a)^*(bab)^+ = (bab)^*(a\,bb)^*ab$$

and

$$(bab)^*ba(bab)^* = ba(bb\,a)^*(bab)^* = (bab)^*ba + (bab)^+(a\,bb)^*ab$$

we get, that $Y_2, Y_3 \subseteq X_0$, and $Y_2 \cap X^+ = Y_3 \cap X^+ = \emptyset$.

The next step is to prove the inclusion on the other direction. Since $\mathrm{Pref}(X) = \mathrm{Suf}(X) = \{a, b, ab, ba, bb, aba, bab, bbb\} = X + \{b, ab, ba\}$, we get

$$X_0 = X^+ + \left(\{b, ab, ba\}X^* \cap X^*\{b, ab, ba\}\right).$$

The language X^+ is clearly subset of $X^+ + Y_1 + Y_2 + Y_3$, the right hand side of the claim, so let's have a look on words in $\{b, ab, ba\}X^*$. First of all $\{b, ab, ba\} \in Y_1 + Y_2 + Y_3$. Next we take words uvw, in which $u \in \{b, ab, ba\}$, $v \in X$ and $w \in X^*$, and try to find out what format they should be, if they are either in X^+ or in $X^+\{b, ab, ba\}$. If $u = ab$, we get with different values of v that

- $v = a \implies uvw = ab \cdot a \cdot w = aba \cdot w \in X^+$,
- $v = bb \implies uvw = ab \cdot bb \cdot w = a \cdot bbb \cdot w \in X^+$,
- $v = aba \implies uvw = ab \cdot aba \cdot w = a \cdot bab \cdot a \cdot w \in X^+$,
- $v = bab \implies uvw = ab \cdot bab \cdot w = (abb)abw$,
- $v = bbb \implies uvw = ab \cdot bbb \cdot w = a \cdot bb \cdot bb \cdot w \in X^+$.

Only the case $v = bab$ needs a closer look. This case is however quite easy, since abw, the end of the word $(abb)abw$, is still in the original form abX^*. That means we can similarly take left factors $v_i \in X$ out of word w until its empty. Recursively this means that

$$abX^* \cap X^*\{b, ab, ba\} \subseteq X^+ + ab(bab)^* \subseteq X^+ + Y_2.$$

Let's have $u = ba$, then

- $v = a \implies uvw = ba \cdot a \cdot w$,
- $v = bb \implies uvw = ba \cdot bb \cdot w = (bab)bw$,
- $v = aba \implies uvw = ba \cdot aba \cdot w$,
- $v = bab \implies uvw = ba \cdot bab \cdot w = (bab)abw$,
- $v = bbb \implies uvw = ba \cdot bbb \cdot w = bab \cdot bb \cdot w \in X^+$.

From these we see, since $b, ba, baa \notin X$, that v can be neither a nor aba. If it were, word uvw wouldn't be in language $X^+\{b, ab, ba\}$. The case $v = bbb$ is trivial; $uvw \in X^+$. This means, only cases $v = bb$ and $v = bab$ give something new. The case

$v = bab$ leads to situation similar to the previous one, where end of the word is $abw \in abX^*$, and we get

$$(ba)babX^* \cap X^*\{b, ab, ba\} \subseteq X^+ + (bab)ab(bab)^* \subseteq X^+ + Y_2.$$

If $v = bb$, we have $uvw = ba \cdot bbw = (bab)bw$ and the next part solves this case.

If $u = b$, then

- $v = a \implies uvw = b \cdot a \cdot w = ba \cdot w$,
- $v = bb \implies uvw = b \cdot bb \cdot w = bbb \cdot w \in X^+$,
- $v = aba \implies uvw = b \cdot aba \cdot w = bab \cdot a \cdot w \in X^+$,
- $v = bab \implies uvw = b \cdot bab \cdot w = (bba) \cdot b \cdot w$,
- $v = bbb \implies uvw = b \cdot bbb \cdot w = bb \cdot bb \cdot w \in X^+$.

The cases $v = a$ and $v = bab$ are the nontrivial ones. In the other cases clearly $uvw \in X^+ + Y_1 + Y_2 + Y_3$. If $v = bab$, then the end of word uvw is still of the form bX^*. Hence this part covers, not only words of format bX^*, but all of them in $(bba)^*bX^*$. Now

$$(bba)^*b = b(bab)^* \subseteq (bab)^*b(bab)^* = Y_1$$

and

$$b(bab)^* \cdot a = (bba)^*ba = ba + (bba)^*bb \cdot aba \subseteq Y_2 + X^+.$$

Also $b(bab)^+aw = (bb\,a)^+baw = (bb\,a)^*bb\,abaw \subseteq X^+$, so words baw are the only words needing further attention. As we saw before, these words can be in X^+ or in either $ba\,babX^* = (bab)abX^*$ or in $ba\,bbX^* = (bab)bX^*$. Previous leads to word in $X^+ + (bab)ab(bab)^*$ and latter gives recursively word in X^+, $(bab)^+b(bab)^*$ or $(bab)^+(bab)ab(bab)^*$.

Hence

$$bX^* \cap X^*\{b, ab, ba\} \subseteq X^+ + Y_1 + Y_2 + Y_3,$$

and so

$$X_0 = X^+ + (\{b, ab, ba\}X^* \cap X^*\{b, ab, ba\}) \subseteq X^+ + Y_1 + Y_2 + Y_3$$

holds. □

A more systematic way to find X_0 is to manipulate automata representing the corresponding language. We can construct iteration, prefix, suffix, union and intersection of automata and finally minimize and compare them. With a computer this can be done rather easily.

Next we must show that languages Y_1, Y_2 and Y_3 are not included in centralizer $\mathcal{C}(X)$ and hence we obtain $\mathcal{C}(X) = X^+$ from equation

$$X^+ \subseteq \mathcal{C}(X) \subseteq X^+ + Y_1 + Y_2 + Y_3.$$

Lemma 8.2. *The words b, ab and ba are not in $\mathcal{C}(X)$.*

Proof. It's easily seen, that $b \notin \mathcal{C}(X)$. Clearly

$$b \in (bab)^*b(bab)^* = Y_1 \subseteq X_0$$

and $a \in X$, so $ab \in XX_0$. However $ab \notin X_0X$ and hence $b \notin \mathcal{C}(X)$.
Similarly $ab \in (bab)^*ab(bab)^* = Y_2 \subseteq X_0$, $a \in X$ and $aab \in XX_0$, but $aab \notin X_0X$.
Finally $ba \in (bab)^*ba(bab)^* = Y_3 \subseteq X_0$, $a \in X$ and $baa \in X_0X$, but $baa \notin XX_0$.

Therefore $b, ab, ba \notin \mathcal{C}(X)$. □

Lemma 8.3. *The languages $ab(bab)^*$, $ba(bab)^*$ and $b(bab)^*$ are in the complement of the centralizer.*

$$ab(bab)^* \cap \mathcal{C}(X) = ba(bab)^* \cap \mathcal{C}(X) = b(bab)^* \cap \mathcal{C}(X) = \emptyset.$$

Proof. Let us choose an arbitrary word $w = ab(bab)^k$ $(k \geq 1)$ in the language $ab(bab)^*$. If we concatenate w with the word $a \in X$, we get the word $a \cdot ab(bab)^k \in XX_0$. In the end of this word there is only one factor in X, the word bab. However the rest of the word, $aab(bab)^{k-1}$, is not in X_0 since

$$a(a\,bb)^{k-1}ab \notin X_0 = X^+ + Y_1 + Y_2 + Y_3.$$

Therefore $ab(bab)^k \notin \mathcal{C}(X)$ $(k \geq 0)$. That means $ab(bab)^* \cap \mathcal{C}(X) = \emptyset$.

Next we choose an arbitrary word $ba(bab)^k \in ba(bab)^*$ $(k \geq 1)$. Now $ba(bab)^k \cdot bab \in X_0X$, but $ba(bab)^k(bab) = bab \cdot ab(bab)^k \notin X\mathcal{C}(X)$ since, as we just proved, $ab(bab)^k \notin \mathcal{C}(X)$. Therefore $ba(bab)^k \notin \mathcal{C}(X)$ $(k \geq 0)$ and $ba(bab)^* \cap \mathcal{C}(X) = \emptyset$.

Similarly, if we have word $b(bab)^k$ $(k \geq 1)$ in language $b(bab)^*$, we see that $a \cdot b(bab)^k \in XX_0$, but $ab(bab)^{k-1} \cdot bab \notin \mathcal{C}(X)X$. Hence also

$$b(bab)^* \cap \mathcal{C}(X) = \emptyset.$$

□

In the next step we will use induction.

Lemma 8.4. *The languages $(bab)^*b(bab)^*$, $(bab)^*ab(bab)^*$ and $(bab)^*ba(bab)^*$ are all in the complement of the centralizer. That is*

$$\begin{aligned}(bab)^*b(bab)^* \cap \mathcal{C}(X) &= \emptyset \\ (bab)^*ab(bab)^* \cap \mathcal{C}(X) &= \emptyset \\ (bab)^*ba(bab)^* \cap \mathcal{C}(X) &= \emptyset.\end{aligned}$$

Proof. The proof is similar for all of these three languages. We mark $v \in \{b, ab, ba\}$ and prove the claim for $(bab)^*v(bab)^*$. We prove, that for any integer $n, i \geq 0$ we have $(bab)^nv(bab)^i \notin \mathcal{C}(X)$. If $n = 0$, then $v(bab)^i \notin \mathcal{C}(X)$ for all $i \geq 0$.

Assume that $(bab)^nv(bab)^i \notin \mathcal{C}(X)$ for all $i \geq 0$, if $n \leq k$ for some integer k. If $n = k + 1$ we get

$$(bab)^{k+1}v(bab)^i \cdot (bab) \in X_0X,$$

but

$$(bab) \cdot (bab)^kv(bab)^{i+1} \notin X\mathcal{C}(X),$$

since induction assumption says $(bab)^kv(bab)^{i+1} \notin \mathcal{C}(X)$.

Therefore $(bab)^{k+1}v(bab)i \notin \mathcal{C}(X)$, and hence

$$Y_1 \cap \mathcal{C}(X) = Y_2 \cap \mathcal{C}(X) = Y_3 \cap \mathcal{C}(X) = \emptyset.$$

□

Theorem 8.5. *The centralizer of $X = \{a, bb, aba, bab, bbb\}$ is*

$$\mathcal{C}(X) = X^+.$$

Proof. The claim follows directly from the previous lemma, since

$$\mathcal{C}(X) = X_0 \cap \mathcal{C}(X) = (X^+ + Y_1 + Y_2 + Y_3) \cap \mathcal{C}(X) = X^+.$$

□

Corollary 8.6. The centralizer of the language $Y = \{a, bb, aba, abb, bab, bba, bbb\}$ coincides with that of

$$\mathcal{C}(Y) = \mathcal{C}(X) = X^+.$$

Proof. The languages X and Y generate the same semigroup $X^+ = Y^+$, since $X \subseteq Y \subseteq X^+$. Additionally, by Theorem 3.5, $\mathcal{C}(Y) = \mathcal{C}(Y^+) = \mathcal{C}(X^+) = \mathcal{C}(X) = X^+$. □

8.4 Analysis

Let's find out what really happens, when the fixed point method is applied to the language $X = \{a, bb, aba, bab, bbb\}$, and why the iteration does not stop.

Even if X is finite and its centralizer is just X^+, fixed point method does not give centralizer after finite number of steps. We could say that fixed point method doesn't succeed, because it is not capable to use induction. For example, if we write all words of $(bab)^*ab(bab)^*$ in the shape of infinite triangle, as in Figure 11, we could illustrate the fixed point method as follows.

$$\begin{array}{ccccccccc}
 & & & & ab & & & & \\
 & & & ab(bab) & & (bab)ab & & & \\
 & & ab(bab)^2 & & (bab)ab(bab) & & (bab)^2ab & & \\
 & ab(bab)^3 & & (bab)ab(bab)^2 & & (bab)^2ab(bab) & & (bab)^3ab & \\
ab(bab)^4 & & (bab)ab(bab)^3 & & (bab)^2ab(bab)^2 & & (bab)^3ab(bab) & & (bab)^4ab \\
\vdots & & \vdots & & \vdots & & \vdots & & \vdots
\end{array}$$

Figure 11: Language $(bab)^*ab(bab)^*$ as an infinite triangle

When fixed point method takes the first step from X_0 to X_1, the language $ab(bab)^*$ is deleted from X_0, as shown in Lemma 8.3. This means that the left-

most row of the triangle is deleted, as Figure 12 shows. Next steps delete the topmost rows one after another, since, as Lemma 8.4 implies, the existence of any word of form $(bab)^i ab(bab)^j$ in the language X_k requires the existence of both words $(bab)^{i-1}ab(bab)^j$ and $(bab)^i ab(bab)^{j-1}$ in the previous language X_{k-1}. Since the bottom of triangle is infinite, the iteration does not stop on finite number of steps. Same happens for languages $(bab)^*ba(bab)^*$ and $(bab)^*b(bab)^*$. Some parts of these triangles are common. For example, $(bab)ab(bab)^* = ba(bab)^+$.

We could try to make iteration faster and more effective by choosing X_0, the starting point of the iteration, smaller than $\mathrm{Pref}(X^+) \cap \mathrm{Suf}(X^+)$. X_0 should still of course include the centralizer. We could for example mark

$$\begin{aligned} B_p &= \{w \in \mathrm{Pref}(X^+) | w \,\mathrm{Pref}_1(X) \subseteq \mathrm{Pref}(X^+)\} \\ B_s &= \{w \in \mathrm{Suf}(X^+) | \,\mathrm{Suf}_1(X) w \subseteq \mathrm{Suf}(X^+)\}, \end{aligned}$$

where B_p clearly includes the centralizer and B_s is the similar set obtained by using suffix instead of prefix. Now we could choose

$$X_0 = B_p \cap B_s.$$

However in this case this does not change the iteration much, since

$$\big(\mathrm{Pref}(X^+) \cap \mathrm{Suf}(X^+)\big) \setminus (B_p \cap B_s) = ab(bab)^* + (bab)^*ba,$$

and this language was deleted already on step $X_0 \to X_1$ in the original iteration.

8.5 Other examples

There exists also rational languages, for which fixed point method doesn't stop and the centralizer is not X^+. One example is the language $X = a\Sigma^+b + b\Sigma^*ba$. If we compute X^+ and S with computer, we see that they are different and hence $X^+ \neq \mathcal{C}(X)$. Table 3 lists numbers of states for automata corresponding to the first steps of iteration, computed by Grail+. Table shows, that beginning from language X_7 every step increases number of states by eight. Closer look on corresponding automata tells that they stay mainly same, but certain parts of them grow with same sequence

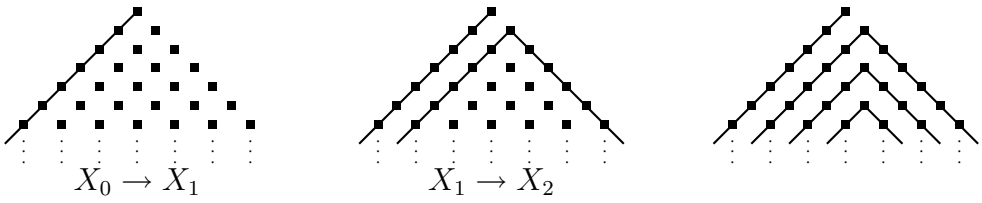


Figure 12: Deleting parts of $(bab)^*ab(bab)^*$ during iteration.

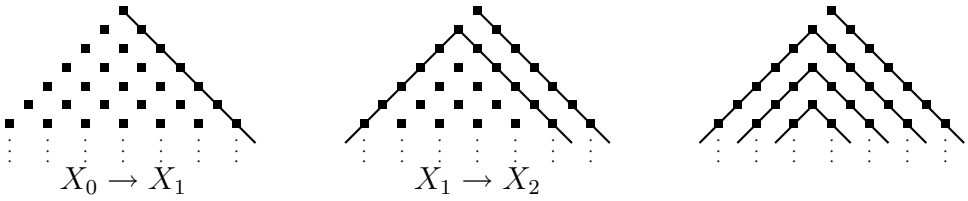


Figure 13: Deleting parts of $(bab)^*ba(bab)^*$ during iteration.

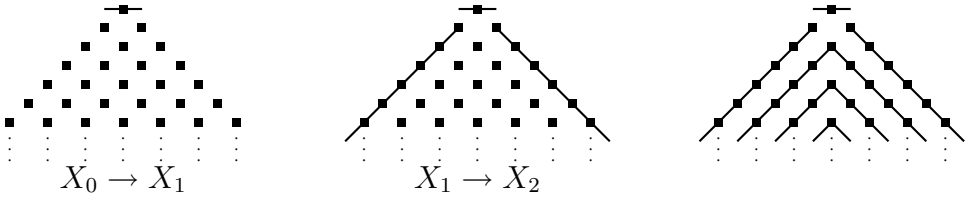


Figure 14: Deleting parts of $(bab)^*b(bab)^*$ during iteration.

	endstates	states	transitions		endstates	states	transitions
X	2	7	14	X_{12}	49	107	214
X_0	2	4	8	X_{13}	53	115	230
X_1	4	15	30	X_{14}	57	123	246
X_2	6	16	32	X_{15}	61	131	262
X_3	9	24	48	X_{16}	65	139	278
X_4	13	33	66	X_{17}	69	147	294
X_5	18	44	88	X_{18}	73	155	310
X_6	23	53	106	X_{19}	77	163	326
X_7	29	67	134	X_{20}	81	171	342
X_8	33	75	150	X_{21}	85	179	358
X_9	37	83	166	X^+	4	12	24
X_{10}	41	91	182	S	8	24	48
X_{11}	45	99	198	Z	10	28	56

Table 3: Numbers of states and transitions of automata corresponding to iteration steps of $X = a\Sigma^+b + b\Sigma^*ba$

on every step. Figure 15 shows the essential part of the automaton of X_{11}. This automaton has two separate "chains" with period of four states. On every step both of these chains get four additional states. Only the last sates of these chains don't follow the pattern. When the iteration goes forward, the ends of the chains go farther and farther. This means that words in $X_i \setminus X_{i+1}$ get longer, when i gets bigger. If i grows infinitely, the chains get infinite and can be replaced by a loop of four states. For example in automaton representing the language X_{11}, we could replace transition $98 \xrightarrow{b} 13$ with transition $98 \xrightarrow{b} 95$ and transition $93 \xrightarrow{b} 13$ with transition $93 \xrightarrow{b} 87$ and minimize the result. Let us use notation Z for the language recognized by this new automaton. Number of states in this automaton is also given in Table 3. We will prove that $Z = \mathcal{C}(X)$. Grail+ verifies that

$$X^+ \subset S \subset Z$$

and that Z commutes with X. Hence for sure

$$X^+ \subset S \subset Z \subseteq \mathcal{C}(X).$$

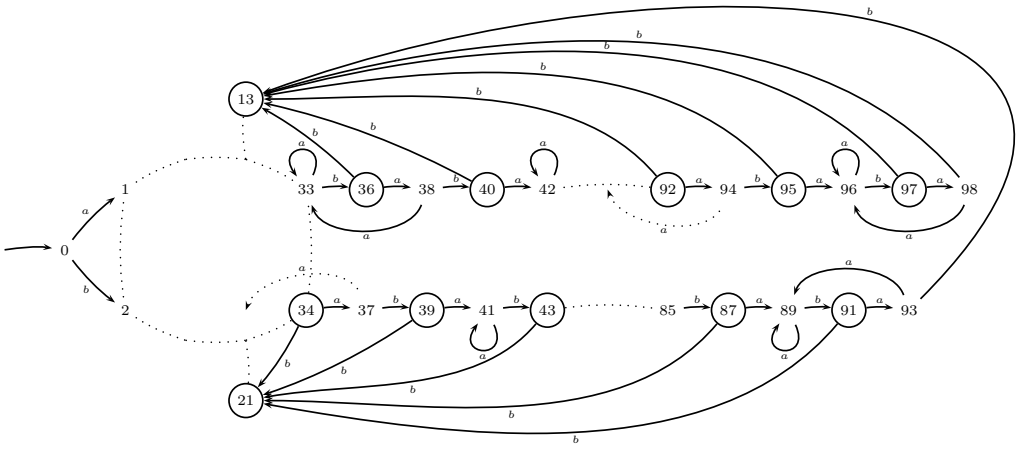


Figure 15: Essential parts of automaton representing X_{11}.

The equality $Z = \mathcal{C}(X)$ can be proven as follows.

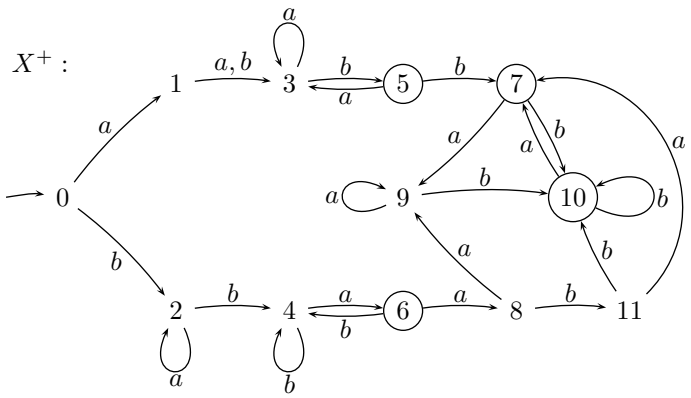


Figure 16: Automaton for X^+. ($X = a\Sigma^+b + b\Sigma^*ba$)

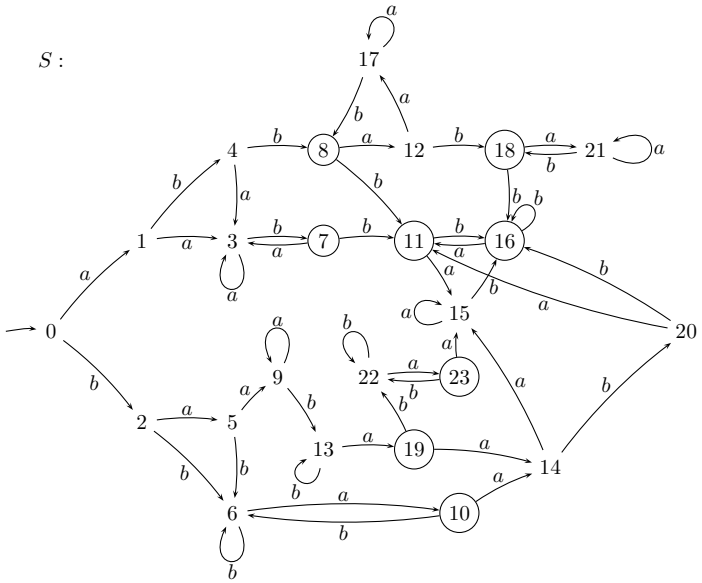


Figure 17: Automaton for S. ($X = a\Sigma^+b + b\Sigma^*ba$)

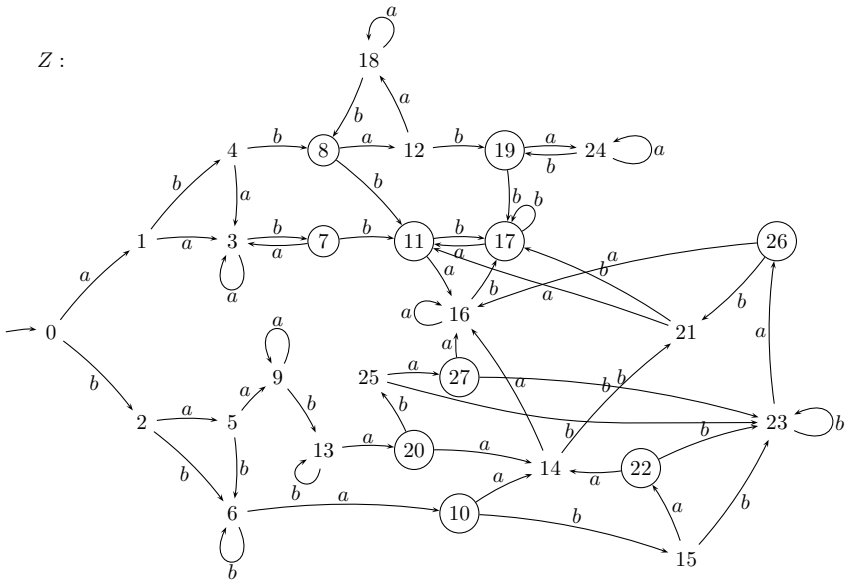


Figure 18: Automaton for $Z = \mathcal{C}(X)$. ($X = a\Sigma^+b + b\Sigma^*ba$)

Lemma 8.7. *Let* $A = aa^+(ba^+)^*$. *Then*

$$A \cap \mathcal{C}(X) = \emptyset.$$

Proof. First of all $aa^nba \in A$ ($n > 1$) and $aab \in X$. However

$$aab \cdot aa^nbaX^{-1} = \{aa\},$$

and since $aa \notin X_0 = \mathrm{Pref}(X^+) \cap \mathrm{Suf}(X^+)$ we have $aab \cdot aa^nba \notin \mathcal{C}(X)X$ and $aa^+ba \cap \mathcal{C}(X) = \emptyset$.

As the second step we see that $aa^+ba^+ba \cap \mathcal{C}(X) = \emptyset$, since

$$aab \cdot aa^+ba^+baX^{-1} = aa + aabaa^+.$$

Now $(aa + aabaa^+) \cap \mathrm{Suf}(X^+) = \emptyset$, which means that also $(aa + aabaa^+) \cap X_0 = \emptyset$.

Finally we use the induction to show, that if for some $k \geq 1$ we have $aa^+(ba^+)^nba \cap \mathcal{C}(X) = \emptyset$, whenever $n \leq k$, then

$$aab \cdot aa^+(ba^+)^{k+1}baX^{-1} = aa + aabaa^+(ba^+)^{\leq k} \subseteq \Sigma^+ \setminus \mathcal{C}(X).$$

Hence $aa^+(ba^+)^*ba \cap \mathcal{C}(X) = \emptyset$. Additionally, we can note that $\Sigma^*aa \cap \mathrm{Suf}(X^+) = \emptyset$, which gives us now

$$aa^+(ba^+)^* \cap \mathcal{C}(X) = \emptyset.$$

□

Now if we choose the language

$$Y_0 = X_0 \setminus A = (\mathrm{Pref}(X^+) \cap \mathrm{Suf}(X^+)) \setminus A$$

as the starting point of the fixed point approach, then the iteration stops already after few steps. Namely

$$Y_6 = Y_7 = Z$$

giving us

$$\mathcal{C}(X) = Z.$$

Now we can note, that we found this centralizer by choosing different initial value for fixed point method.

Notion 8.8. The result of the fixed point approach is not stable. Even if the fixed point method, with original starting point X_0, leads to infinite computation, it may still be possible to find the centralizer with a finite computation by choosing smaller starting point for iteration. This helps only, if we succeed to ”eliminate” the ”hard part” of the original starting point.

All languages leading to infinite iteration do not give so clearly periodic automata. For example, for the language $X = \Sigma a\Sigma^* a\Sigma + \Sigma b\Sigma^* b\Sigma$, the number of states in iteration steps grows in two phases. Every other step the addition is twelve and every other step eighteen states.

With the finite language $X = \{a, bb, aab, aba, abb, baa, bab, bba, bbb\}$ the rate of the growth does not become constant, at least during first fifty steps. Additionally, in every second step the number of states increases and every other it decreases. The decrease is usually half of the increase. If we take look on the steps as automata, we see still some kind of pattern as in Figure 19.

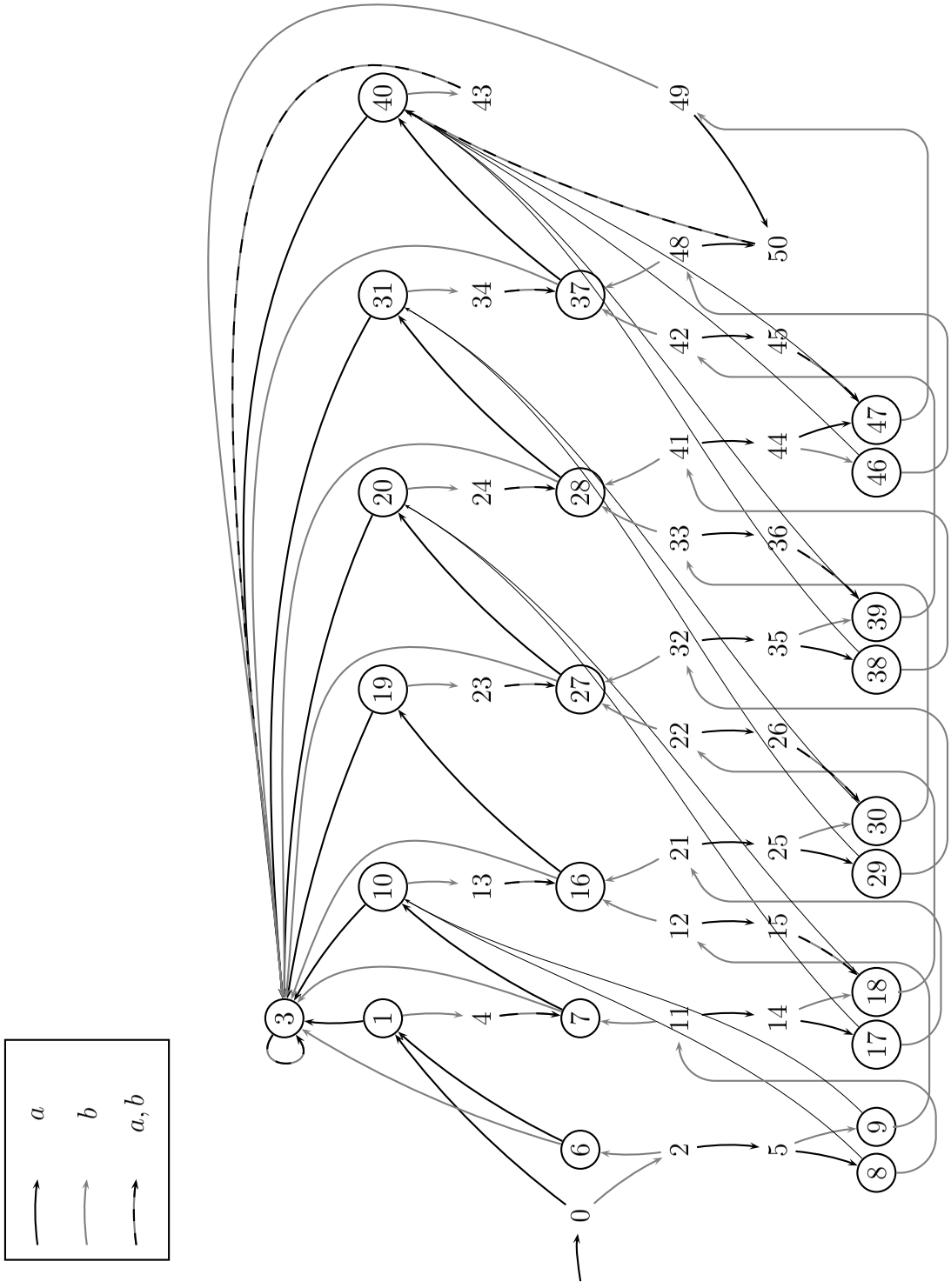


Figure 19: Language $X = \{a, bb, aab, aba, abb, baa, bab, bba, bbb\}$ / Iteration X_{12}

Since the language X is finite and for example the word baa is singular in X, we know that the centralizer is finitely generated. In fact, the centralizer is the language X^+, which can be proved as follows, with the same technique as we used in examples of Chapter 4.

The set of proper suffixes of $X = \{a, bb, aab, aba, abb, baa, bab, bba, bbb\}$ is $\{1, a, b, aa, ab, ba, bb\} = 1 \cup \Sigma \cup \Sigma^2$. Next we find variables n_i for corresponding suffixes u_i.

$u_0 = 1 :\ 1 \cdot X \subseteq \mathcal{C}(X) \implies n_0 = 1.$

$u_1 = a :\ a \in X \subseteq \mathcal{C}(X) \implies n_1 = 0.$

$u_2 = b :\ b \cdot a \notin X\mathcal{C}(X) \implies b \notin \mathcal{C}(X),$
$ba \cdot a \notin X\mathcal{C}(X) \implies ba \notin \mathcal{C}(X),$
$baab \cdot a \notin X\mathcal{C}(X) \implies baab \notin \mathcal{C}(X),$
and then by induction
$b(aab)^n \cdot a = baa \cdot (baa)^{n-1}ba \notin X\mathcal{C}(X) \implies b(aab)^n \notin \mathcal{C}(X) \quad \forall n > 0$
$\therefore (\forall n \geq 0) \quad b(aab)^n \notin \mathcal{C}(X)$
i.e. n_2 does not exist.

$u_3 = aa :\ aa \in X^+ \subseteq \mathcal{C}(X) \implies n_3 = 0.$

$u_4 = ba :\ ba(aba)^n \cdot aba = baa \cdot (baa)^n ba \notin X\mathcal{C}(X)$
$\implies ba(aba)^n \notin \mathcal{C}(X) \quad \forall n \geq 0$ (as in case $u_2 = b$).
$\implies n_4$ does not exist.

$u_5 = ab :$ Since $X^R = X$, we have also $\mathcal{C}(X)^R = \mathcal{C}(X)$ and hence $aba \cdot (aba)^n ab =$
$ab(aab)^n \cdot aab \notin \mathcal{C}(X)X \implies ab(aab)^n \notin \mathcal{C}(X) \quad \forall n \geq 0.$
$\implies n_5$ does not exist.

Now $I = \{0, 1, 3\}$, $n_0 = 1, n_1 = 0, n_3 = 0$ and we obtain the result

$$\begin{aligned} G &= \bigcup_{i \in I} u_i X^{n_i} = 1 \cdot X + a + aa = X \\ \mathcal{C}(X) &= GX^* = X^+. \end{aligned}$$

We can compare the sequence of steps on fixed point method to the sequence of decimal numbers of finite length. Let $X = a\Sigma^+b + b\Sigma^*ba$. We can think that the sequence of automata corresponding languages X_i is like the sequence

$$0, 12345; 0, 123412345; 0, 1234123412345; 0, 12341234123412345; \ldots$$

of decimal numbers. Both sequences grow periodically towards an infinite representation, but this limit can also be given with a finite representation. Same way as the limit $0, 12341234\ldots$ of the sequence of decimal numbers above has a finite representation $\frac{1234}{9999}$. Similarly the sequence obtained by the search for the centralizer of $X = \{a, bb, aab, aba, abb, baa, bab, bba, bbb\}$ may be compared to the sequence

$$0, 12798; 0, 1218798098; 0, 12127098; 0, 12121978068798; 0, 121212707698; \ldots,$$

of decimal numbers, which has $0, 121212\ldots = \frac{12}{99}$ as the limit. In this sequence we get one period in two steps, the number of digits both increases and decreases and the end of decimal numbers get more and more non-periodic digits, which do not have any effect on the limit.

References

[1] C. Choffrut, J. Karhumäki, Combinatorics on Words, in: Rozenberg, G., Salomaa, A. (eds.), Handbook of Formal Languages, Vol. 1, Springer-Verlag, (1997), 329–438.

[2] C. Choffrut, J. Karhumäki, On Fatou properties of rational languages, In: C. Martin-Vide and V. Mitrana (eds.), Where Mathematics, Computer Science, Linguistics and Biology Meet, Kluwer, Dordrecht 2000.

[3] C. Choffrut, J. Karhumäki, N. Ollinger, The commutation of finite sets: A challenging problem, Theoretical Computer Science 273, (2002), 69–79.

[4] J. H. Conway, Regular Algebra and Finite Machines, Chapman Hall, 1971.

[5] K. Culik II, J. Karhumäki, Systems of equations and Ehrefeucht's conjecture, Discr. Math. 43, (1983), 139–153.

[6] K. Culik II, J. Karhumäki, P. Salmela, Fixed Point Approach to Commutation of Languages, in N. Jonoska, Gh. Paun, G. Rozenberg (eds.), Aspects of Molecular Computing, – Essays dedicated to Tom Head on the occasion of His 70th Birthday, LNCS 2950, Festschrift, Springer–Verlag, Berlin Heidelberg, (2004), 119–131.

[7] J. Karhumäki, Challenges of Commutation - An advertisement, in: Fundamentals of Computation Theory 2001, R. Freivalds, (ed.), LNCS 2138, Springer–Verlag, New York, (2001), 15–23.

[8] J. Karhumäki, M. Latteux, I. Petre, The commutation with codes and ternary sets of words, Theory of Computing Systems (to appear); preliminary version in: Proceedings of STACS 2003, LNCS 2607, Springer–Verlag, Berlin Heidelberg, (2003), 74-84.

[9] J. Karhumäki, L. Lisovik, The equivalence problem for finite substitutions on ab^*c, with applications, IJFCS 14, (2003), 699–710.

[10] J. Karhumäki, I. Petre, Conway's problem for three-word sets, Theoretical Computer Science 289 (1), (2002), 705–725.

[11] J. Karhumäki, I. Petre, On the Centralizer of a Finite Set, in: U. Montari et al. (eds.), ICALP 2000, LNCS 1853, Springer–Verlag, Berlin Heidelberg, (2000), 536–546.

[12] J. Karhumäki, I. Petre, Two Problems on Commutation of Languages, in: G. Paun, G. Rozenberg and A. Salomaa (eds.), Current Trends in Theoretical Computer Science, The Challences of the New Century, World Scientific, Singapore, (2004), 477–494.

[13] J. Karhumäki, I. Petre, The Branching Point Approach to Conway's Problem, in: Formal and Natural Computing, W. Brauer et al. (Eds.), LNCS 2300, Springer–Verlag, Berlin Heidelberg, (2002), 69–76.

[14] M. Kunc, The Power of Commuting with Finite Sets of Words, manuscript, available at `http://www.math.muni.cz/~kunc/`. Extended abstract in Proceedings of STACS 2005, LNCS 3404, Springer–Verlag, Berlin Heidelberg, (2005), 569–580.

[15] E. Leiss, Language Equations, Springer, 1998.

[16] M. Lothaire, Combinatorics on words, Addison-Wesley, Reading, MA., 1983.

[17] G. S. Makanin, The problem of solvability of equations in a free semigroups, Mat. Sb. 103, (1977), 147–236; Math. USSR Sb. 32, (1977), 129–198.

[18] M. L. Minsky, Computation: Finite and Infinite Machines, Prentice-Hall, 1967.

[19] I. Petre, Commutation Problems on Sets of Words and Formal Power Series, PhD Thesis, University of Turku (2002).

[20] W. Plandowski, Satisfiability of word equations with constants is in PSPACE, Journal of the ACM 51 (3), (2004), 483–496.

[21] B. Ratoandromanana, Codes et motifs, RAIRO Inform. Theor., 23 (4), (1989) 425–444.

[22] Grail+ 3.0 -program, University of West Ontario, Canada, <URL:http://www.csd.uwo.ca/research/grail/grail.html>.