LUDWIG-MAXIMILIANS-UNIVERSITÄT MÜNCHEN

CHAIR OF THEORETICAL COMPUTER SCIENCE AND THEOREM PROVING

Formal Languages and Automatas

Ruben Triwari

Seminar Paper on Formal Languages and Automatas
for the course “Algebra & Computer Science”

Supervisor: Prof. Dr. Jasmin Blanchette

Advisor: Xavier Genereux

Submission Date: August 15, 2024

Disclaimer

I confirm that this seminar paper is my own work and I have documented all sources and material used.

Munich, August 15, 2024 Author

Abstract

Abstract... **Keywords**: Math

Contents

1 Introduction

2 Backround

First of all, we need to define some structures and ideas, that will be really important in this paper. We will define formal languages in two ways and then automatas arcordingly. A formal language is just a set of words, where words are a series of characters from an alphabet.

Definition 1 *An Alphabet is an abritrary non empty finite set denoted with* Σ

Example 2.1 *Simple examples for Alphabets would be:*

1. $\Sigma = \{x\}$
2. $\Sigma = \{a,b,c,d,\ldots z\}$
3. $\Sigma = \{1,2,3,\ldots n\}$ *where* $n \in \mathbb{N}$

This definition is rather general, because an alphabet Σ can really be anything. We could also define an Alphabet as subset of the natural numbers $\Sigma \subset \mathbb{N}$. This is equalivant to the previous definition, becaus we can always find a one-to-one map between a finite amount of objects and the natural numbers. In other words we can always enumerate a finite amount of objects. The above definition results in more readable words, thus resulting in more readable problems. Furhtermore its unintuitive to have words full of numbers and languages full of number series. That is why im chosing this definition, like most of the literature.
Now with our first building block defined we can define words, which are just a series of characters in an alphabet.

Definition 2 *Let* Σ *be an fixed alphabet, then w is a series of characters out of* Σ*.*

$$w = (a_1, a_2, \ldots, a_n) \in \Sigma^n$$

We will denote it by:

$$w = a_1 a_2 \ldots a_n$$

The empty word which contains no characters, and is denoted by ε*.*

Example 2.2 *Simple examples for words would be:*

1. $\Sigma = \{a,b,c\} \rightsquigarrow w_1 = abc$
2. $\Sigma = \{x\} \rightsquigarrow w_2 = xxxx$
3. $\Sigma = \{1,2,3\} \rightsquigarrow w_3 = 112233$
4. $\Sigma = \{1,2,3\} \rightsquigarrow w_4 = \varepsilon$

The empty word ε is analogous to the empty set in Set theory. It has a lot uf uses, but one obvious is that now one can define a Monoid over words $(M, \cdot, \Sigma, \varepsilon)$. M is a set of words and multiplication is concetanation of words, then ε is the neutral element of the Monoid. Which brings us to the next definition.

Definition 3 *Let Σ be an fixed alphabet and w, v words with characters out of Σ. Then the contenation is defined by:*

$$w \cdot v := wv$$

sometimes just written as wv. With powers recursively defined by:

$$w^0 := \varepsilon$$
$$w^1 := w$$
$$w^{n+1} := w^n \cdot w$$

In simple terms a concatenation of two words is just a new word with the two words chained togehter.
Now we can again build on words to define langauges.

Definition 4 *Let Σ be an fixed alphabet, then a formal language is a set of words, with characters in Σ.*

Example 2.3 *Simple examples for formal languages would be:*

1. $\Sigma = \{a,b,c\} \rightsquigarrow L = \{aaa, bbb, ccc, abc\}$
2. $\Sigma = \{x\} \rightsquigarrow L = \{x, xx, xxx, xxxx\}$
3. $\Sigma = \{1,2,3\} \rightsquigarrow L = \{11, 22, 33, 123, \varepsilon\}$

This definition is really clear and simple. We will see a less intuitive way to define language shortly. Obviously the simpler one has some downsides, which we will discuss later in the paper.
The kleene star * is a really powerful operator, to define it we need to first define contenation for languages.

Definition 5 *Let Σ be an fixed alphabet and L, V languages with words constructed out of Σ. Then the contenation of formal languages is defined by:*

$$L \cdot V := \{l \cdot v | l \in L \wedge v \in V\}$$

sometimes just written as LV. With powers defined recursively:

$$L^0 := \{\varepsilon\}$$
$$L^1 := L$$
$$L^{i+1} := \{l_i \cdot l | l_i \in L^i \wedge l \in L\}$$

In simple terms we are creating a new language by compbining every word of the first language with every word of the second. Now we are able to define the kleene star operator.

Definition 6 *Let Σ be an fixed alphabet and L a language with words constructed out of Σ. Then the kleene star of a formal language is defined by:*

$$L^* := \bigcup_{i \in \mathbb{N}_0} L^i = L^0 \cup L^1 \cup L^2 \cup \ldots$$

In simple terms it generates all possible permutation of abriatary length of a formal language. This is really powerful, because Σ^* is the biggest language one can construct from the alphabet Σ. Thus every language L generated by Σ is just a subset of Σ^*.
Unintuively, we can also write a formal language as a formal power series.

Definition 7 *Let Σ be an fixed alphabet, then a formal language can be written as formal power series L:*

$$L : \Sigma^* \to \mathbb{B}$$

where

$$\mathbb{B} = \{0,1\}$$

$$L = \sum_{w \in \Sigma^*} (L,w)w \ \texttt{where} \ (L,w) \in \mathbb{B}.$$

Note: *The set of formal power series is denoted by $\mathbb{B}\langle\langle\Sigma^*\rangle\rangle = \{\Sigma^* \to \mathbb{B}\}$*
Notation: *If the language consist only out of one character or the empty word $a \in \Sigma \cup \{\varepsilon\}$, then the formal power series would be:*

$$L = h \in \mathbb{B}\langle\langle\Sigma^*\rangle\rangle$$

where

$$h : \Sigma^* \to \mathbb{B}$$

$$h(w) = \begin{cases} 1 & \text{if } w = a \\ 0 & \text{otherwise} \end{cases}.$$

In this case we just write $L = a$.

This looks like a sum, we call it series, but it is actually a map from words to zero or one. The sum is a way to write a function definition: Infront of every word $w \in \Sigma^*$ in the series, there is a $x \in \mathbb{B}$, which is zero or one. So we can interpret every element in the series as an assignment: $w \mapsto x$. All words that map to one, are in the language, the others are not. So we can write a formal language as a function $\Sigma^* \to \mathbb{B}$.
The representation with sets has the andvantage that a formal language will inherit all operations on sets like complement, union, and intersection. Those operations are really useful, thus we will define similar on the power series version.

Definition 8 *Let Σ be an fixed alphabet and $L, V \in \mathbb{B}\langle\langle\Sigma^*\rangle\rangle$, where*

$$L = \sum_{w \in \Sigma^*} (L,w)w \quad \text{and} \quad V = \sum_{w \in \Sigma^*} (V,w)w.$$

Addition and multiplication on formal power series is defined as follows:

$$L + V := \sum_{w \in \Sigma^*} ((L,w) + (V,w))w \quad \text{and} \quad L \cdot V := \sum_{\substack{u,v \in \Sigma^* \\ w = uv}} ((L,u) \cdot (V,v))w$$

This definition is not recursive, because the right addition and muliplication operater are from the Boolean Algebra $\mathbb{B} = \{0,1\}$.

$+$	0	1
0	0	1
1	1	1

$\times$	0	1
0	0	0
1	0	1

With powers defined recursively:

$$\begin{aligned} L^0 &:= \varepsilon \\ L^1 &:= L \\ L^{i+1} &:= \sum_{\substack{u,v\in\Sigma^* \\ w=uv}} ((L^i,u)\cdot(L,v))w, \end{aligned}$$

Kleene Star can now also be defined for formal power series:

$$L^* := L^0 + L^1 + L^2 + L^3 + \cdots = \sum_{i\in\mathbb{N}_0} L^i.$$

With these operation defined our two versions of formal languages are powerful tools, which will be compared in the following sections. In order to conpare them we will also need finite automatas. Simillar to formal languages there are multiple ways one can define finite automatas. In this paper we will define two. One is more suitable for combining it with the set version of formal languages and the other one is more siutable for the formal power series representation. First we will define the one more suitable for the set version.

Definition 9 *A finite automata is a quintuple $M = (Q,\Sigma,\delta,q_0,F)$, where*

1. *Q is a finite set of states,*
2. *Σ is an alphabet, all inputs are constructed from,*
3. *$\delta : Q\times\Sigma \to Q$ is the transition function, carrying the automata from one state to another,*
4. *$q_0 \in Q$ is the initial state of the automata,*
5. *$F \subset Q$ are the accepting states of the automata.*

In simple terms, an finite automata gets an input string and a starting state. Then it works through the input string, starting at the first charater, carrying the starting state to another according to the transition function. Exactly the same procedure, for the second input string character and the remainding characters. If this process ends up in an accepting state, the automata accepts the input string. Consequently, the finite automata generates a language, namely the words that the automata accepts. The definition above is one possible way to encode a finite automata. Like with graphs we can encode the transition function of an finite automata, with just a matrix.

Definition 10 *Let $M = (Q,\Sigma,\delta,q_0,F)$ be an abitrary finite automata, where $Q = \{s_1,s_2,\dots s_n\}$. Frthermore, we need a helper function:*

$$\delta' : \mathbb{N}\times\mathbb{N} \to \mathbb{B}\langle\langle\Sigma^*\rangle\rangle$$

where

$$\delta'(i,j) = \begin{cases} \alpha, & \text{if } \exists \alpha \in \Sigma : \delta(s_i, \alpha) = s_j \\ 0, & \text{otherwise} \end{cases}.$$

Then the finite automata M can be encoded with a Matrix $A \in \mathbb{B}\langle\langle\Sigma^*\rangle\rangle^{n\times n}$ *in the following way:*

$$A = \begin{pmatrix} \delta'(1,1) & \delta'(1,2) & \delta'(1,3) & \dots & \delta'(1,n) \\ \delta'(2,1) & \delta'(2,2) & \delta'(2,3) & \dots & \delta'(2,n) \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ \delta'(n,1) & \delta'(n,2) & \delta'(n,3) & \dots & \delta'(n,n) \end{pmatrix}.$$

We can denote a state transition $s_i \to s_j$ *with* $s_i A_{ij} = s_j$.
Note: *We definined* $A \in \mathbb{B}\langle\langle\Sigma^*\rangle\rangle^{n\times n}$ *and not* $A \in \Sigma^{n\times n}$*, because otherwise the defintion is not closed under matrix multiplication, nor matrix addition.*
The kleene star of a Matrix can intuively be defined as:

$$A^* = E_n + A + A^2 + A^3 \cdots = \sum_{i\in\mathbb{N}_0} A^i$$

where

$$E_n \in \mathbb{B}\langle\langle\Sigma^*\rangle\rangle^{n\times n}$$

$$(E_n)_{ij} := \begin{cases} \varepsilon & \text{if } i = j \\ 0 & \text{otherwise} \end{cases} \in \mathbb{B}\langle\langle\Sigma^*\rangle\rangle$$

Now we can write our finite automata in a shorter way:

$$M' = (\Sigma, A, q_0, F)$$

This definition only works, because we have finite sets. Note that we can omit the state set, since as mentioned above, there always exist a one-to-one map between a set with finite objects and a subset of the natural numbers. The first object gets the number one, the second the number two, and so forth. The new definition of the state transition function works like a lookup table, when one wants to know what characters map state i to state j, its a simple lookup: M_{ij}.

Example 2.4 *Here is a simple automata depicted, where the circles are states , the arrows are transitions, the start is indicating the initial state, and the double lined circle is the accepting state.*

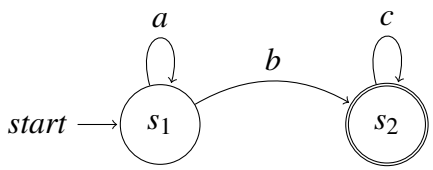
$$M_{tuple} = (\{s_1, s_2\}, \{a, b, c\}, \delta_0, s_1, \{s_2\}), M_{matrix} = (\{a, b, c\}, A, s_1, \{s_2\})$$

where

$$\delta_0(s_1, a) = s_1, \;\; \delta_0(s_1, b) = s_2, \;\; \delta_0(s_2, c) = s_2$$

and

$$A = \begin{pmatrix} a & b \\ 0 & c \end{pmatrix} \in \mathbb{B}\langle\langle\Sigma^*\rangle\rangle^{2\times 2}.$$



The resulting accepting formal language is:

$$\textit{Set represntation: } L = \{a^n bc^n : n \in \mathbb{N}\}$$

$$\textit{Formal power series represntation: } L = a^* \cdot b \cdot c^* = a^* bc^*$$

3 Formal Languages: Sets vs. Formal Power Series

In this chapter we will discuss the obvious diffrences and simillarities of both representations. First we will give intuition, why one might use a seires to represent a function definition. Secondly, we will map the formal power series version into the set version and the other way around, to get a better understanding of both. Lastly, we will discuss why formal power series are more powerful than just sets.
The intuition behind the notation of formal power series is quite simple. The operation defined on the formal power series, which are maps, are really close to a series. For addition, formal power series and normal series are quite simillar, namely defined pointwise.
In the following are $L, V \in \mathbb{B}\langle\langle\Sigma^*\rangle\rangle$ and A, B series.

$$L + V = \sum_{w \in \Sigma^*} ((L,w) + (V,w))w \quad \leftrightarrow \quad A + B = \sum_{n \in \mathbb{N}} (a_n + b_n)x^n$$

The multiplication defined on formal power series are again really close to the cauchy product of series. In the following are $L, V \in \mathbb{B}\langle\langle\Sigma^*\rangle\rangle$ and A, B series.

$$L \cdot V = \sum_{\substack{u,v \in \Sigma^* \\ w = uv}} ((L,u) \cdot (V,v))w \quad \leftrightarrow \quad A \cdot B = \sum_{\substack{i,j \in \mathbb{N} \\ n = i+j}} (a_i \cdot b_j)x^n$$

Consequently, it is quite handy to use a formal power series as function definition, because the operations seem natural with series. Despite the notation being quite confusing at first glance.
Next we will show that bot representations of a formal language are equivalent, by mapping one to another. To be sure that both representation are actually equivalent, we need to show there exist an bijective map, between formal power series and the set of words. This map needs to preserve the structure, that means the order of function application and operator application must be irrelevant. Such maps are called isomporphism and structures for that a isomorphism exist are called isomporphic to each other. Thus we want tho show that both representation are isomorphic.

Proof: Let Σ be an fixed alphabet, then we claim that:

$$\mathbb{B}\langle\langle\Sigma^*\rangle\rangle \cong \mathscr{P}(\Sigma^*)$$

In order to show those are isomorphic, we provide a map $\phi : \mathbb{B}\langle\langle\Sigma^*\rangle\rangle \to \mathscr{P}(\Sigma^*)$ and its inverse $\phi^{-1} : \mathscr{P}(\Sigma^*) \to \mathbb{B}\langle\langle\Sigma^*\rangle\rangle$ and show that ϕ is bijective and preservers structure, formally:
Let $L, V \in \mathbb{B}\langle\langle\Sigma^*\rangle\rangle$, then for ϕ must hold:

$$\phi(L+V) = \phi(L) \cup \phi(V) \tag{1}$$
$$\phi(L \cdot V) = \phi(L) \cdot \phi(V) \tag{2}$$
$$\phi(L^*) = \phi(L)^* \tag{3}$$

We define ϕ and its inverse as follows:

$$\phi(L) = \{w \in \Sigma^* : (L,w) = 1\}$$

$$\phi(L)^{-1} = h_L$$

where

$$h_L : \Sigma^* \to \mathbb{B}\langle\langle\Sigma^*\rangle\rangle$$

$$h_L(w) = \begin{cases} 1, & \text{if } w \in L \\ 0, & \text{otherwise} \end{cases}$$

To prove that ϕ is bijective, it must hold

$$\phi^{-1}(\phi(L)) = L$$

and

$$\phi(\phi^{-1}(L)) = L.$$

First we show $\phi^{-1}(\phi(L)) = L$
Let $L \in \mathbb{B}\langle\langle\Sigma^*\rangle\rangle$ abitrary but fixed, with

$$L = \sum_{w \in \Sigma^*} (L,w)w.$$

Then

$$\begin{aligned} \phi^{-1}(\phi(L)) = \phi^{-1}(\{w \in \Sigma^* : (L,w) = 1\}) &= \begin{cases} 1 & \text{if } w \in \{w \in \Sigma^* : (L,w) = 1\} \\ 0 & \text{otherwise} \end{cases} \\ &= \begin{cases} 1 & \text{if } (L,w) = 1 \\ 0 & \text{otherwise} \end{cases} \\ &= \sum_{w \in \Sigma^*} (L,w)w = L. \end{aligned}$$

Next we need to show that $\phi(\phi^{-1}(L)) = L$.
Let $L \in \mathscr{P}(\Sigma^*)$ abitrary but fixed, then

$$\phi(\phi^{-1}(L)) = \phi(h_L) = \{w \in \Sigma^* : (h_L,w) = 1\} = \{w \in \Sigma^* : h_L(w) = 1\} = \{w \in \Sigma^* : w \in L\} = L.$$

Consequently, ϕ is a one-to-one correspondence between formal power series and set of words. We are not done yet, we still need to prove that ϕ preserves structure, thus we need to show $(1), (2)$, and (3). Starting with number one:

$$\textbf{(1)} \quad \phi(L+V) = \phi(L) \cup \phi(V)$$

In the following let $L, V \in \mathbb{B}\langle\langle \Sigma^* \rangle\rangle$ arbitrary but fixed. Then

$$\begin{aligned}
\phi(L+V) &= \{w \in \Sigma^* : (L,w) + (V,w) = 1\} \\
&= \{w \in \Sigma^* : (L,w) = 1 \vee (V,w) = 1\} \\
&= \{w \in \Sigma^* : w \in \{x \in \Sigma^* : (L,x) = 1\} \vee w \in \{x \in \Sigma^* : (V,x) = 1\}\} \\
&= \{w \in \Sigma^* : w \in \phi(L) \vee w \in \phi(V)\} = \phi(L) \cup \phi(V).
\end{aligned}$$

Continuing with two:

$$\textbf{(2)} \quad \phi(L \cdot V) = \phi(L) \cdot \phi(V)$$

In the following let $L, V \in \mathbb{B}\langle\langle \Sigma^* \rangle\rangle$ arbitrary but fixed. Then

$$\begin{aligned}
\phi(L \cdot V) &= \{uv \in \Sigma^* : (L,u) \cdot (V,v) = 1\} \\
&= \{uv \in \Sigma^* : (L,u) = 1 \wedge (V,v) = 1\} \\
&= \{uv \in \Sigma^* : u \in \{w \in \Sigma^* : (L,w) = 1\} \wedge v \in \{w \in \Sigma^* : (V,w) = 1\}\} \\
&= \{uv \in \Sigma^* : u \in \phi(L) \wedge v \in \phi(V)\} = \phi(L) \cdot \phi(V).
\end{aligned}$$

Last but not least, three:

$$\textbf{(3)} \quad \phi(L^*) = \phi(L)^*$$

In the following let $L \in \mathbb{B}\langle\langle \Sigma^* \rangle\rangle$ arbitrary but fixed. Then

$$\phi(L^*) = \phi(\sum_{i \in \mathbb{N}_0} L^i) \overset{\textbf{(1)}}{=} \sum_{i \in \mathbb{N}_0} \phi(L^i) \overset{\textbf{(2)}}{=} \sum_{i \in \mathbb{N}_0} \phi(L)^i = \phi(L)^*$$

Thus ϕ is an isomorphism and our claim follows. □

Now we have shown that both definition are equivalent, but why should one use the less natural formal power series definition? The anwser to this, as so often in mathematics, is **abstraction**. Finite automata are accepting formal languages, thus we can say they generate a formal language. When abstracting from finite automata's to weighted automata's which have a weight or cost between each transition, the generated object is not longer a language, but a map that assigns a weight or cost to each word. Generally, this weight or cost can be seen as a semiring (a Ring that does not have necessarily an additive inverse), which are objects that behave like numbers, we will denote an arbitrary semiring with $\mathbb{S}$. Those maps, that get accepted by an weighted automata, have the signature $L : \Sigma^* \to \mathbb{S}$ and are indeed a formal power series over $\mathbb{S}$, consequently $L \in \mathbb{S}\langle\langle \Sigma^* \rangle\rangle$. To Summarize, finite automata generate formal power series over $\mathbb{B}$ and weighted automata generate formal power series over $\mathbb{S}$. So formal power series are in fact an abstraction of formal languages.

4 Kleene's theorem (Power series, Matrix Automata)

In this chapter we will examine, what kind of languages get accepted by an finite automata. In order to do that, we will prove that all languages that get accepted by an finite automata are rational and every rational language can be generated by an finite automata (kleene's theorem). A rational language is defined inductively.

Definition 11 *Let Σ be an fixed alphabet, then a rational language is defined as follows:*

1. *A language $L = a \in \mathbb{B}\langle\langle\Sigma^*\rangle\rangle$ with a single character $a \in \Sigma$ is said to be rational.*
2. *The empty word ε is rational.*
3. *The sum of two rational Languages $L_1, L_2 \in \mathbb{B}\langle\langle\Sigma^*\rangle\rangle$ are again rational $L = L_1 + L_2$.*
4. *The product of two rational Languages $L_1, L_2 \in \mathbb{B}\langle\langle\Sigma^*\rangle\rangle$ are again rational $L = L_1 \cdot L_2$.*
5. *The kleene star of a rational language $L \in \mathbb{B}\langle\langle\Sigma^*\rangle\rangle$ is again rational $V = L^*$.*

So we start with just one character and then building with our operations step by step a language. To show kleene's theorem we have to prove two direction. First we will prove that finite automata only accept rational languages. Afterwards, we will prove that every rational language is accepted by an finite automata. In an effort to do that, we need to understand certain properties of the Matrix encoding of finite automatas. Raising the power of two of a Matrix is defined by:
Let $A \in \mathbb{B}\langle\langle\Sigma^*\rangle\rangle^{n\times n}$ be an arbitrary matrix, then

$$(A^2)_{ij} = \sum_{k=1}^{n} A_{ik}A_{kj}.$$

As mentioned before, this matrix is a transistion function of an finite automata, where we can look up, what character carry one state in another. In A_{ij} is the character that carry i to j, thus multiplying A by itself, gives us a new Matrix with all words with length two, given as input and i is the starting node would carry i to j in two steps. In the formula above we see that we are trying to go throug all nodes $k \in \{1,2,3,\dots n\}$ to end up in j and then uniting them together with addition.
Generally, we can have a matrix $A \in \mathbb{B}\langle\langle\Sigma^*\rangle\rangle^{n\times n}$ risen to the power of $n \in \mathbb{N}$:

$$(A^n)_{ij} = \sum_{k_1=1}^{n}\sum_{k_2=1}^{n}\dots\sum_{k_{n-1}=1}^{n} a_{ik_1}a_{k_1k_2}a_{k_2k_3}\dots a_{k_{n-1}j}$$

This is not beatiful at all, but the idea stays exactly the same: Now we have in $(A^n)_{ij}$ all words that carry i to j in n steps, brute forcing throug all possible paths. Now to get all possible paths of arbitrary length we just add all together:

$$E_n + A + A^2 \cdots = \sum_{i\in\mathbb{N}} A^i = A^*$$

Yielding, that in $(A^*)_{ij}$ are all possible words, carrying the automata from i to j. If i is the starting node and j the accepting node, the accepting language of the automata is just $(A^*)_{ij}$. That means, A^* stores all accepting languages of an automata, with transition matrix A. With those important properties, we can now move on to the first prove.

Proof:

claim: All finite automata only accept rational languages.

Let $A \in \mathbb{B}\langle\langle\Sigma^*\rangle\rangle^{n\times n}$ be a transition function of an arbitrary finite automata. A is rational, because the transition function consists only out of characters. Since A^* stores every accepting language of that finite autotamata, for every possible initial state and accepting state, we just need to prove that all entries in A^* are rational languages. We will prove this statement via induction over the number of states, namely the dimension of the square matrix $n \in \mathbb{N}$.
Base case: $n = 1$
Thus

$$A \in \mathbb{B}\langle\langle\Sigma^*\rangle\rangle^{1\times 1} = \mathbb{B}\langle\langle\Sigma^*\rangle\rangle.$$

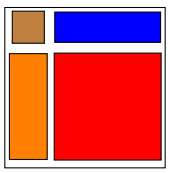
We know that $A \in \mathbb{B}\langle\langle\Sigma^*\rangle\rangle$ is rational, but that also means that A^* is per definition rational, which finishes the base case.
Inudction step: $n \to n+1$
We can assume the claim holds for $n \in \mathbb{N}$ and need to prove it also holds for $n+1$. we can assume that a transition matrix with dimenion $n \times n$ has only rational entries.
Let $A \in \mathbb{B}\langle\langle\Sigma^*\rangle\rangle^{(n+1)\times(n+1)}$ be a transition function of an arbitrary finite automata. Our goal is it to reduce the matrix A by one dimension. In order to do that we devide the matrix in four blocks:

$$A = \begin{pmatrix} A(1,1) & A(1,2) \\ A(2,1) & A(2,2) \end{pmatrix}$$

where

$$A(1,1) \in \mathbb{B}\langle\langle\Sigma^*\rangle\rangle^{1\times 1}, A(2,1) \in \mathbb{B}\langle\langle\Sigma^*\rangle\rangle^{n\times 1}, A(1,2) \in \mathbb{B}\langle\langle\Sigma^*\rangle\rangle^{1\times n}, A(2,2) \in \mathbb{B}\langle\langle\Sigma^*\rangle\rangle^{(n-1)\times(n-1)}$$

visually:



We can interpret our block matrix as a transition matrix with four entries, thus having two states. Our new two states consists of the original states in the following way:

$$\mathbf{1} = \{1\} \text{ and } \mathbf{2} = \{2,3,4,\ldots n+1\}$$

□

5 Maximum Gap Lemma

6 Not all Formal languages are regular

7 Kleene Schützenberg theorem and weigthed automatas

8 Conclusion

8.1 Citation

The citation method follows the author-year system. Place reference is in the text, footnotes should only be used for explanations and comments. The following notes are taken from the *language* bibliography template from `ron.artstein.org`:

The *Language* style sheet makes a distinction between two kinds of in-text citations: citing a work and citing an author.

- Citing a work:
  - Two authors are joined by an ampersand (&).
  - More than two authors are abbreviated with *et al.*
  - No parentheses are placed around the year (though parentheses may contain the whole citation).
- Citing an author:
  - Two authors are joined by *and*.
  - More than two authors are abbreviated with *and colleagues*.
  - The year is surrounded by parentheses (with page numbers, if present).

To provide for both kinds of citations, `language.bst` capitalizes on the fact that `natbib` citation commands come in two flavors. In a typical style compatible with `natbib`, ordinary commands such as `\citet` and `\citep` produce short citations abbreviated with *et al.*, whereas starred commands such as `\citet*` and `\citep*` produce a citation with a full author list. Since *Language* does not require citations with full authors, the style `language.bst` repurposes the starred commands to be used for citing the author. The following table shows how the `natbib` citation commands work with `language.bst`.

Command	Two authors	More than two authors
`\citet`	Hale & White Eagle (1980)	Sprouse et al. (2011)
`\citet*`	Hale und White Eagle (1980)	Sprouse and colleagues (2011)
`\citep`	(Hale & White Eagle 1980)	(Sprouse et al. 2011)
`\citep*`	(Hale und White Eagle 1980)	(Sprouse and colleagues 2011)
`\citealt`	Hale & White Eagle 1980	Sprouse et al. 2011
`\citealt*`	Hale und White Eagle 1980	Sprouse and colleagues 2011
`\citealp`	Hale & White Eagle 1980	Sprouse et al. 2011
`\citealp*`	Hale und White Eagle 1980	Sprouse and colleagues 2011
`\citeauthor`	Hale & White Eagle	Sprouse et al.
`\citeauthor*`	Hale und White Eagle	Sprouse and colleagues
`\citefullauthor`	Hale und White Eagle	Sprouse and colleagues

Authors of *Language* articles would typically use `\citet*`, `\citep`, `\citealt` and `\citeauthor*`, though they could use any of the above commands. There is no command for giving a full list of authors.

8.2 Bibliography

The bibliography of this template includes the references of the *language* stylesheet as a sample bibliography.

List of Figures

List of Tables

References

BUTT, MIRIAM, und WILHELM GEUDER (Hg.) 1998. *The projection of arguments: Lexical and compositional factors*. Stanford, CA: CSLI Publications.

CROFT, WILLIAM. 1998. Event structure in argument linking. In Butt & Geuder, 21–63.

DONOHUE, MARK. 2009. Geography is more robust than linguistics. Science e-letter, 13 August 2009. URL `http://www.sciencemag.org/cgi/eletters/324/5926/464-c`.

DORIAN, NANCY C. (Hg.) 1989. *Investigating obsolescence*. Cambridge: Cambridge University Press.

GROPEN, JESS; STEVEN PINKER; MICHELLE HOLLANDER; RICHARD GOLDBERG; und RONALD WILSON. 1989. The learnability and acquisition of the dative alternation in English. *Language* 65.203–57.

HALE, KENNETH, und JOSIE WHITE EAGLE. 1980. A preliminary metrical account of Winnebago accent. *International Journal of American Linguistics* 46.117–32.

HASPELMATH, MARTIN. 1993. *A grammar of lezgian*. Walter de Gruyter.

HYMES, DELL H. 1974a. *Foundations in sociolinguistics: An ethnographic approach*. Philadelphia: University of Pennsylvania Press.

HYMES, DELL H. (Hg.) 1974b. *Studies in the history of linguistics: Traditions and paradigms*. Bloomington: Indiana University Press.

HYMES, DELL H. 1980. *Language in education: Ethnolinguistic essays*. Washington, DC: Center for Applied Linguistics.

MINER, KENNETH. 1990. Winnebago accent: The rest of the data. Lawrence: University of Kansas, MS.

MORGAN, TJH; NT UOMINI; LE RENDELL; L CHOUINARD-THULY; SE STREET; HM LEWIS; CP CROSS; C EVANS; R KEARNEY; I DE LA TORRE; ET AL. 2015. Experimental evidence for the co-evolution of hominin tool-making teaching and language. *Nature communications* 6.

PERLMUTTER, DAVID M. 1978. Impersonal passives and the unaccusative hypothesis. *Berkeley Linguistics Society* 4.157–89.

POSER, WILLIAM. 1984. *The phonetics and phonology of tone and intonation in Japanese*. Cambridge, MA: MIT Dissertation.

PRINCE, ELLEN. 1991. Relative clauses, resumptive pronouns, and kind-sentences. Paper presented at the annual meeting of the Linguistic Society of America, Chicago.

RICE, KEREN. 1989. *A grammar of Slave*. Berlin: Mouton de Gruyter.

SALTZMAN, ELLIOT; HOSUNG NAM; JELENA KRIVOKAPIC; und LOUIS GOLDSTEIN. 2008. A task-dynamic toolkit for modeling the effects of prosodic structure on articulation. *Proceedings of the 4th International Conference on Speech Prosody (Speech Prosody 2008)*, Campinas, 175–84. URL `http://aune.lpl.univ-aix.fr/~sprosig/sp2008/papers/3inv.pdf`.

VAN DER SANDT, ROB A. 1992. Presupposition projection as anaphora resolution. *Journal of Semantics* 9.333–77.

SINGLER, JOHN VICTOR. 1992. Review of Melanesian English and the Oceanic substrate, by Roger M. Keesing. *Language* 68.176–82.

SPROUSE, JON; MATT WAGERS; und COLIN PHILLIPS. 2011. A test of the relation between working memory capacity and syntactic island effects. *Language*, to appear.

STOCKWELL, ROBERT P. 1993. Obituary of Dwight L. Bolinger. *Language* 69.99–112.

SUNDELL, TIMOTHY R. 2009. Metalinguistic disagreement. Ann Arbor: University of Michigan, MS. URL `http://faculty.wcas.northwestern.edu/~trs341/papers.html`.

TIERSMA, PETER M. 1993. Linguistic issues in the law. *Language* 69.113–37.

WILSON, DEIRDRE. 1975. *Presuppositions and non-truth-conditional semantics*. London: Academic Press.

YIP, MOIRA. 1991. Coronals, consonant clusters, and the coda condition. *The special status of coronals: Internal and external evidence*, hrsg. von Carole Paradis und Jean-François Prunet, 61–78. San Diego, CA: Academic Press.