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 11, 2024

Disclaimer

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

Munich, August 11, 2024 Author

Abstract

Abstract... **Keywords**: Math

Contents

1 Introduction

2 Backround

First of all, we need to define some simple 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 an isomorphism between a finite amount of objects and the natural numbers. In other words we can always number 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*. The empty word which contains no characters, 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:

$$w^n = \underbrace{w \cdot w \cdot \ldots w}_{n \text{ times}} \text{ where } n \in \mathbb{N}$$

and

$$w^0 = \varepsilon.$$

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:*

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

sometimes just written as LV. With powers defined recursively:

$$\begin{aligned} L^0 &:= \{\varepsilon\} \\ L^1 &:= L \\ L^{i+1} &:= \{l_i \cdot l | l_i \in L^i \vee l \in L\} \end{aligned}$$

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:*

$$L = \sum_{w \in \Sigma^*} (L,w)w \texttt{ where } (L,w) \in \mathbb{B} = \{0,1\}$$

The sum is iterating over all possible words $w \in \Sigma^*$ and (L,w) is an indicating function, which is one if w should be in the language and ohterwise zero. Thus all words with an one infront are in the language.
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 L,V formal languages, L and V can be written*

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

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

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

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:

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

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 $H(i, j) = \{\alpha \in \Sigma : \delta(s_i, \alpha) = s_j\}$ and two helper functions:*

$$w : \Sigma^{\times} \to \Sigma^*, \delta' : \mathbb{N} \times \mathbb{N} \to \Sigma^*$$

where

$$w(\{a_1, a_2, \dots, a_n\}) = a_1 a_2 \dots a_n$$

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

Then the finite automata M can be encoded with a Matrix $A \in (\Sigma^*)^{n \times n}$ 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 $s_i A_{ij} = s_j$. The kleene star of a Matrix can intuively be defined as:

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

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 an isomorphism 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.*

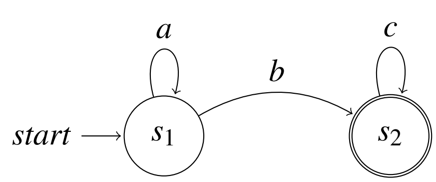
$$M_{tuple} = (\{s_1, s_2\}, \{a, b, c\}, \delta_0, s_1, \{s_2\}), M_{matrix} = (\{a, b, c\}, A, s_1, \{s_2\})$$

where

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

and

$$A = \begin{pmatrix} a & b \\ 0 & c \end{pmatrix}.$$



The resulting accepting formal language is:

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

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

3 Formal Languages: Sets vs. Formal Power Series

4 Maximum Gap Lemma

5 Pumping Lemma

6 Not all Formal languages are regular

7 Pumping Lemma vs. Maximum Gap Lemma

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

9 Kleene's theorem (Set, Tuple)

10 Kleene's theorem vs. Kleene's theorem

11 Kleene Schützenberg theorem and weigthed automatas

12 Conclusion

12.1 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.

12.2 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.