Formal Languages and Automata Theory
List of Projects

If you need the book [1], contact me directly.

Students who decide to work on a project must announce the lecturer on the chosen topic until the Friday of Week 5 (sharp deadline). The project consists of either:

- an implementation of an algorithm solving various problems (Section 1), or
- a theoretical topic of other applications of formal languages and automata (Section 2)

The project is individual or in team of maximum 2 students. Every project must be presented during the lecture/seminar (15 minutes Latex Beamer presentation, presentation template will be handed in later). Prior to the presentation, it must be uploaded on Google Classroom (code to be handed in later).

1 Practical Projects

Project 1 Evaluation of arithmetical expressions using two stacks. – *Ovidiu Muresan*

Project 2 Generation and evaluation of arithmetical expressions using polish notation. – *Mihai Bontea*

Project 3 *Simulation of an DFA.* A deterministic finite automaton (DFA) is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where Q is a finite set called the states, Σ is a finite set called the alphabet, $\delta : Q \times \Sigma \to Q$ is the transition function, $q_0 \in Q$ is the start state, and $F \subset Q$ is the set of accept states. We design a DFA and give a string as input. The program should check the validity of that string and displays the states that is encountered by the input string.

Inputs to the program:

1. The number of states that the DFA will contain.
2. Here we take the Σ the set of alphabets constant as $0, 1$ they might be altered or increased changing the course of the program.
3. Now we define the next states for every state for both alphabets 0 and 1; and if a state is final state or not.
4. $q_0 \in Q$ the initial state is taken as input and we have our DFA ready.
5. Next we take one string after another and check the validity of that string and the program shoes the number of states the machine traversed for that input string.

Program output:

1. Acceptance of the input string.
2. States traversed by the machine for that input string.

Project 4

(1) Given a grammar G, specify its type.

(2) Given the grammars G_1, ..., G_n, construct the grammars which generate $L(G_1) \cup ... \cup L(G_n)$ (union), $L(G_1) \cdot ... \cdot L(G_n)$ (product), $L(G_1)^*$, ..., $L(G_n)^*$ (Kleene closure) (you should use the algorithm developed at (1)).

Project 5 Construction of an DFA equivalent to a given regular expression. – *Patrick Lenis & Paul Cvasa*

Project 6 DFA minimization (see lecture notes on the website and Lecture 14 from [2]). *Zaharia Cristian& Andrada Popa*

Project 7 Write a program that reads a deterministic finite automaton from a file and, using (f)lex (http://flex.sourceforge.net/), (a) tests whether the automaton is deterministic, (b) tests whether the language used by the automaton is void, and if not finds a word accepted by the language, (c) simulates the automaton on the given word (printing the transitions/states)[1]. – *Alexandru Lazea & Andrada Surpeteanu*

Project 8 Consider the following puzzle: "On one side of a river are three humans, one big monkey, two small monkeys, and one boat. Each of the humans and the big monkey are strong enough to row the boat. The boat can fit one or two bodies (regardless of size). If at any time at either side of the river the monkeys outnumber the humans, the monkeys will eat the humans. How do you get everyone on the other side of the river alive?" Show that the language of solutions to the puzzle is regular. Write a finite automaton for the puzzle to a file (perhaps using a script if you need it). Using the previously written program find and print a solution to the puzzle. The printing should be done to be "understandable by humans"[2]. – *Dragos Ursan & Tugmeanu Andrei*

Project 9 Applications with regular expressions. Consider the Facebook metrics available at `http://archive.ics.uci.edu/ml/datasets/Facebook+metrics#` → `Data Folder`. Use regular expressions for computing certain statistics. You can use the paper mentioned at the link for some interesting ones. – *Adrian Tura*

Project 10 Applications with regular expressions. Consider the travel reviews metrics available at `http://archive.ics.uci.edu/ml/datasets/Travel+Reviews` → `Data Folder`. Use regular expressions for computing certain statistics, for example the two statistics from `http://ataspinar.com/2016/01/21/sentiment-analysis-with-bag-of-words/`, section Data Collection. – *Rosian Pop & Darco Murja*

Project 11 NFA for text search. (see, e.g. Chapter 2.4.2 from [1]). – *Rares Istoc*

Project 12 DFA for recognizing a set of keywords. (see, e.g. Chapter 2.4.3 from [1]). –

Project 13 Given a grammar, find its type. – *Dora Calea & Liviu-Marian Sopon*

Project 14 Convert a left linear grammar to a right linear one (see lecture notes). – *Robert-Cristian Ciama & Raul Filea*

Project 15 Algorithm converting an NFA to a DFA (see lecture notes). – *Sorina-Aurelia Dumitru & Sirona Miulescu*

[1] From Gabriel Istrate lecture notes
[2] From Gabriel Istrate lecture notes

Project 16 Algorithm for conversion to Chomsky normal form (see, e.g. Chapter 7.1.5 from [1]). –

Project 17 Simulation of deterministic pushdown automata. – *Fineas Silaghi*

Project 18 Algorithm converting a right-linear grammar into the equivalent, left-linear one. –

Project 19 Regular Expressions to Automata conversion. (see, e.g. Chapter 3.2.3 from [1] or lecture notes) – *Stefan Jieanu*

Project 20 DFA to regular expressions conversion. (see, e.g. Chapter 3.2.1 from [1] or lecture notes)– *Dan Cojocaru*

Project 21 Algorithm converting an ε-NFA to a DFA (see lecture notes). –

2 Theoretical Projects

Project 1 The Turing machine. (Chapter 8.2 from the book [1]) – *Catalin Tiu*

Project 2 Intractable Problems. Problems solvable in polynomial time. Example. (see e.g. Chapter 10.1.1-10.1.2 from the book [1]). – *Marius Bijan & Teodora Brotea*

Project 3 Intractable Problems. Problems solvable in nondeterministic polynomial time. Example (see e.g. Chapter 10.1.3-10.1.5 from the book [1]). – *Alexandru Predus*

Project 4 Algebraic Laws for Regular. Example (see e.g. Chapter 3.4 from the book [1]). You should also include a concrete example, e.g. Exercise 3.4.3. – *Darian Voda*

Project 5 Normal forms for context-free languages. Example (see e.g. Chapter 7.1 from the book [1]). – *Alexandru Manafu & Denisa Cotuna*

Project 6 The Pumping Lemma for context-free languages. Example (see e.g. Chapter 7.2 from the book [1]). – *Raluca Chis & Simona Tuns*

Project 7 Closure Properties of context-free languages. Example (see e.g. Chapter 7.3 from the book [1]). – *Monea Sebastian*

Project 8 Decision properties of context-free languages. Example (see e.g. Chapter 7.4 from the book [1]). – *Andrei Andronache*

Project 8 Intractable problems. The classes $\mathcal{P}$ and $\mathcal{NP}$ (see e.g. Chapter 10. from the book [1])) – *Nina Cocea*

References

[1] John E. Hopcroft, Rajeev Motwani, and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation (3rd Edition)*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2006.

[2] Dexter C. Kozen. *Automata and Computability*. Springer-Verlag, Berlin, Heidelberg, 1st edition, 1997.