NTIN071 Automata and grammars

Jakub Bulín

Adapted from the Czech lecture slides by Marta Vomlelová.

Translation, minor modifications, and any errors are mine.

Contents

1 Introduction, Myhill-Nerode theorem **2**

1.1 About the course 2

2 Introduction **3**

1 Introduction, Myhill-Nerode theorem

1.1 About the course

Resources

The course is mostly based on the following two textbooks:

- *Hopcroft* et al: "Introduction to Automata Theory, Languages, and Computation" (3rd edition) – an online copy and several physical copies are available in the library
- *Sipser*: "Introduction to the theory of computation" (3rd edition) – a physical copy is in the library

see the course webpage for additional resources

The path to success

- *Study and self-test regularly (ideally each week).* Can you write down the definition/theorem/proof, fully and correctly?
- *Make your own notes.* The lecture notes are not fully self-contained.
- *Review after each lecture.* Complete your understanding.
- *Try to attend all lectures.* If you miss one, catch up before the next. Use office hours and the textbooks as needed.
- *Learn to work with the formalism,* comfortably and precisely.
- *Pay attention to the tutorial* in a similar way.

How to study?

I highly recommend this minicourse on effective learning:

https://www.samford.edu/departments/academic-success-center/how-to-study

Invest 35 minutes now, save many hours later!

Aims of the course

- get familiar with abstract models of computation
- be able to *formally* describe such models
- understand how minor changes can lead to huge difference in expressive power
- experience the unavoidability of undecidable problems
- a brief introduction to complexity theory (P, NP, and friends)
- prepare for NTIN090 Intro to Complexity and Computability
- also used in NSWI098 Compiler Principles, and in linguistics

Two levels of understanding: the *idea* behind a concept and the ability to *formalize* said concept

2 Introduction

Formal languages

A *language* L over an *alphabet* Σ is a set of *words* (finite strings) consisting of symbols (letters) from the alphabet.

Languages can represent:

- natural languages (words, well-formed sentences),
- programming languages (expressions, statements), document formats (XML,. . .)
- formal proofs
- possible strings of input or sensor readings of a machine, or
- *decision problems*, for example, $\Sigma = \{0, 1\}$ and

$$L = \{w \in \Sigma^\star \mid w \text{ encodes a CNF formula which is satisfiable}\}$$

Classifying languages

- Testing (membership of) words: how complex is the computing device needed? (*automata*)
- Generating words: how complex rules? (*grammars*)

$$\begin{tikzcd}
\boxed{\textit{Turing machines}} \arrow[r, leftrightarrow] & \boxed{\text{Formal grammars}} \\
\boxed{\text{Pushdown automata}} \arrow[r, leftrightarrow] & \boxed{\textit{Context-free grammars}} \\
\boxed{\textit{Finite automata}} \arrow[r, leftrightarrow] & \boxed{\text{Regular grammars}}
\end{tikzcd}$$

NB: Almost all languages have no such finite representation.