# Problems and their learning goals

## Case 0 Morse alphabet – Basic consepts

Goal: To learn the basic concepts like alphabet, strings, languages, consideration that all symbolic languages can be coded by binary alphabet. Practising the problem-based method.

## Case 1 Problem solving company – Modelling problems, their difficulty and solvability

Goal: Overview of modelling problems, their difficulty and solvability. The need for formal languages (descriptions), Chomsky hierarchy of languages, existence of unsolvable problems. (In a more profound way in the end of course.)

## Case 2 Search for mail addresses and precompiler for programming language – Regular expressions

Goal: To understand the idea of regular expressions and construct a describing expression for a given problem. Notice! In b) one should first consider, what is meant by legal definitions. The regular languages are sufficient tool for e.g. variable declarations, but not for checking correct use of parantheses. So this problem also shows the limits of regular languages!

## Case 3 Coffee automaton – Deterministic finite automata

Goal: To learn how to construct a deterministic finite automaton and represent it as a program.

## Case 4 Nondeterministic editor – Nondeterministic finite automata and determinization

Goal: To understand the idea of nondeterministic automata and how to determinize them.

## Case 5 Roman checking exams – Finite automata vs. regular expressions, $\epsilon$-automata.

Goal: To learn how to construct a finite automaton corresponding a regular expression and vice versa. Also $\epsilon$-automata and determinization and minimization are needed.

## Case 6 Grandma's rhyme – Pumping Lemma

Goal: To understand the limits of regular languages and how to recognize irregular languages. Pumping Lemma.

## Case 7 L-systems – grammars

Goal: To understand the idea of grammars, especially context-free grammars, and how to program simple grammar-based systems.

## Case 8 Parantheses parsing – Pushdown automaton

Goal: To learn to construct pushdown automata and how they can be programmed.

## Case 9 Arithmetic calculator – LL(1)-grammars, recursive parser

Goal: To understand the idea of LL(1)-grammars, how to transform nearly LL(1)-grammars into LL(1)-form and how to implement a recursive parser for it. Also the idea of "hidden" stack in the recursive programs.

## Case 10 General parser – the CYK-algorithm and Chomsky normal form

Goal: To understand how CYK-algorithm works and why Chomsky Normal Form is needed. How to modify an arbitrary context-free grammar into CNF.

## Extraproblems: Attribute grammar and parser tools

Goal: To study the idea of attribute grammars or test some parser tool.

## Case 11 Summing machine – basics of Turing machines

Goal: To understand the basic idea of Turing machines and how to construct them. Understanding that multiple tape machines are equivalent with standard TM's.

## Case 12 Library functions for TM's – deeper practice on Turing machines

Goal: To study more carefully, how to construct TM's for different purposes. Also consideration of equivalence between programs/TM's, functions/submachines.

## Case 13 Programming competition – Universal machines and universal languages, solvability proofs

Goal: To understand the idea of universal machines and universal languages. How to study properties of Turing machines? Introduction to the unsolvability of semantic properties.

## Case 14 Philosophical considerations

Goal: To consider the meaning of main results concerning solvability and Church-Turing theses. Limits of computation.

## Case 1 revisited – overview of the principles learnt

Goal: To create an overview, how we can use the techniques and principles learnt in this course to analyze the computability and difficulty of any given problem.