Languages, automata and computation II
Tutorial 4 – Learning theory. Logic

Winter semester 2024/2025

Learning regular languages

During the lecture it was shown Angluin's DFA learning algorithm. Recall that in Angluin's learning framework, Learner can ask two kind of queries: Membership queries ("is word $w \in \Sigma^*$ in the language?" answer: yes / no) and equivalence queries ("is the language recognised by a given DFA A?" answer: yes / no + counter-example word witnessing inequivalence). Note that the size n of the minimal automaton to be learned is not known by Learner in advance, however it is fixed by Teacher before the learning algorithm can start (otherwise Teacher can cheat and always answers "no" continually increasing n). In this tutorial we explore what happens when we question one or more of the aspects of this framework.

In the sequel, we fix a binary alphabet $\Sigma = \{a, b\}$. Recall that a (total) DFA is a tuple $A = (\Sigma, Q, q_I, F, \delta)$, where Q is a finite set of states, q_I is the initial state, $F \subseteq Q$ is the set of final states, and $\delta : Q \times \Sigma \to Q$ is the transition function. The *size* of a DFA A is the number of its states. The *complexity* of a learning algorithm is a function $f : \mathbb{N} \to \mathbb{N}$ s.t. every automaton of size $n \in \mathbb{N}$ can be learned with at most $f(n)$ queries.

Exercise 1 (146)**.** Consider a modified learning framework where Learner can only ask equivalence queries (and no membership queries). Moreover, to an equivalence query, Teacher only answers yes / no (i.e., without inequivalence counter-examples).

1. Design a learning algorithm of exponential complexity.
2. Show that one cannot do better. Does knowing the size n of Teacher's automaton help Learner?

The modification of the exercise above where Teacher additionally needs to provide inequivalence counter-examples yields the same outcome, but now the lower bound is harder to design. It was given as an open problem in [**?**], later solved in [**?**].

Exercise 2. (147) Consider a modified learning framework where Learner can only ask membership queries and a single equivalence query at the end.

1. Show that there is no learning algorithm.

2. What happens if Learner knows in advance the size n of the automaton to be learned?

3. In the previous situation, is there a polynomial learning algorithm?

Exercise 3 (148)**.** Consider a modified learning framework where Learner can ask membership and equivalence queries, however Teacher does not provide any counter-example to the equivalence queries.

1. Is there a polynomial time learning algorithm in this case? Does knowing the size n of Teacher's automaton help Learner?

2. What happens if Teacher provides only negative counter-examples (i.e., counter-examples only of the kind $w \in L(A) \setminus L$)? And only positive ones $(L \setminus L(A))$?

First-order logic

Free monoid

Fix a finite alphabet $\Sigma = \{a, b\}$. Consider the structure

$$(\Sigma^*, \cdot, a, b, \varepsilon)$$

where the *domain* is Σ^* (the free monoid generated by Σ), "$\cdot$" is the concatenation operation (the monoid operation), and a, b, ε are constants representing themselves. We say that a language of words $L \subseteq \Sigma^*$ is *first-order definable* in this structure if there is a first-order formula $\varphi(x)$ with a single free variable x s.t. the set of words satisfying $\varphi(x)$ is exactly L.

Problem 1. Show that the language a^*b^* is first-order definable in the structure M.

Problem 2. Show that the language of words with an even number of a's is first-order definable in the structure M. *Hint: Use a larger alphabet* $\Sigma' = \Sigma \cup \{\$\}$.

Problem 3. Show that the language $a^n b^n$ is first-order definable in the structure M.

Problem 4. Show that every recursively enumerable language is first-order definable in the structure M.

Quantifier elimination

Problem 5. Show that the structure $(\mathbb{Q}, \leq)$, where "$\leq$" is the natural order of the rational numbers, admits quantifier elimintion.