Introduction to Theoretical Computer Science

Coursework 1

Due 12:00 on Friday 24.10.2025.

This coursework is formative. Submission on the LEARN page. Grading via Grade-Scope.

1 Regular languages

Recall that in the lectures we showed that the class of regular languages was closed under union, sequential composition, and Kleene closure.

(a) The complement of a language L, written $\overline{L}$, is every string not in L, i.e. $\Sigma^* \setminus L$. Show that the regular languages are closed under complement. [*4 marks*]

(b) Using (a), or other arguments, show that the regular languages are closed under intersection. The intersection of two languages $L_1 \cap L_2$ is the set of all strings that are in both L_1 and L_2, that is $\{w \mid w \in L_1 \wedge w \in L_2\}$. [*4 marks*]

(c) An *all-NFA* is a variant of an NFA where a string w is only accepted if *all* states reached on word w are final, i.e. $\delta^*(q_0, w) \subseteq F$. Show that there is an all-NFA that recognises a language if and only if it is regular. [*4 marks*]

(d) Prove that $L = \{\mathtt{0}^n\mathtt{1}^m\mathtt{2}^{m-n} \mid m \geq n \geq 0\}$ is not regular. You may use any of the three methods used in lectures: the Pumping Lemma, the Myhill-Nerode theorem, or using (or proving) closure properties of the regular languages to reduce the problem to a known non-regular language. As an extension exercise, you may wish to try multiple methods. [*4 marks*]

2 Context-free languages

Consider the CFG G:

$$S \to \mathtt{a}S \mid \mathtt{a}S\mathtt{b}S \mid \varepsilon$$

(a) Informally characterise $\mathcal{L}(G)$. *[4 marks]*

(b) Show that it is *ambiguous* by finding a string for which you can construct two parse trees and two leftmost derivations. *[4 marks]*

(c) Find an unambiguous grammar for $\mathcal{L}(G)$. *[4 marks]*

(d) Give a pushdown automaton that recognises $\mathcal{L}(G)$. *[4 marks]*

(e) Some $w \in \{\mathtt{a}, \mathtt{b}\}^*$ have unique parse trees in G. What are those strings w? Give an efficient test to tell whether w has this property. The test "try all parse trees to see how many yield w" is not adequately efficient. *[4 marks]*

3 Reductions for undecidability

In lectures, we considered the Halting Problem H, the Looping Problem L, and the Uniform Halting Problem UH. Now we consider the Universal Looping problem UL: given a machine M, does M loop on all inputs R?

(a) Show, by reduction from L, that UL is undecidable. *[4 marks]*

(b) Show, by constructing a suitable machine, that UL is co-semi-decidable. (*Hint:* interleaving.) *[4 marks]*

We often want to know whether a program, or even just a function/method in a program, correctly implements its specification. Can we write programs to check this?

Recall that we say a machine computes a function f if, when started with n in R_0, it halts with $f(n)$ in R_0.

Take f to be the factorial function $f(n) = n!$.

Let the decision problem *Fac* be the (codes of) the register machines that compute f.

(c) **Construct** a reduction from H to *Fac*, and so show that *Fac* is undecidable. *[6 marks]*

Hint: you need to start with an arbitrary program, for which we want to know whether it halts, and end up with an 'is it a factorial function?' problem. Probably you won't much care what the arbitrary program actually computes.