School of Computing and Information Systems
COMP30026 Models of Computation Tutorial Week 12

28–30 October 2020

The plan

Sadly, the last tutorial session is here (but note that we plan to run a catch-up event before the exam). The last tutorial exercises are about computability, decidability, reduction and simulation. From here on, work to develop confidence in your own solutions. For the exam, you need to be able to read your solutions critically, to try to find errors or holes in your own work. You need to be able to judge your answers and decide, without spending too much time on it, when what you have is correct, so that you can move on.

The exercises

97. The following Turing machine D was written to perform certain manipulations to its input—it isn't intended as a recogniser for a language, and so we don't bother to identify an accept or a reject state. The machine stops when no transition is possible, and whatever is on its tape at that point is considered output.

    D's set of states is $\{q_0, q_1, q_2, q_3, q_4\}$, with q_0 being the initial state. The input alphabet is $\{1\}$ and the tape a x z ␣ ␣ s for 'blank', or absence of a proper symbol.

    $$\begin{array}{lcllcl} \delta(q_0, 1) & = & (& q_2, \mathtt{z}) & = & (q_4, 1, L) \\ \delta(q_0, \sqcup) & = & (q_4, \sqcup, L) & \delta(q_2, 1) & = & (\quad) \quad q_3, 1, R) \\ \delta(q_1, 1) & = & (q_1, \mathtt{x}, R) & \delta(q_2, \mathtt{x}) & = & (\quad \sqcup) \quad q_2, 1, L) \end{array}$$

    Draw D's diagram and determine what D doe

98. A 2-PDA is a pushdown automaton that has two stacks instead of one. In each transition step it may consume an input symbol, pop and/or push to stack 1, and pop an/or push to stack 2. It can also leave out any of these options (using ϵ moves) just like the standard PDA.

    In the Week 9 lecture we used the pumping lemma for context-free languages to established that the language $B = \{a^n b^n c^n \mid n \in \mathbb{N}\}$ is not context-free. However, B has a 2-PDA that recognises it. Outline in English or pseudo-code how that 2-PDA operates.

99. In fact, a 2-PDA is as powerful as a Turing machine. Outline an argument for this proposition by showing how a 2-PDA can simulate a given Turing machine. Hint: Arrange things so that, at any point during simulation, the two stacks together hold the contents of the Turing machine's tape, and the symbol under the tape head sits on top of one of the stacks.

100. Show that the halting problem for Turing machines is undecidable. More precisely, show that the language

    $$Halt_{TM} = \{\langle M, w\rangle \mid M \text{ is a Turing machine and } M \text{ halts when run on input } w\}$$

    is undecidable. Hint: Use reduction from A_{TM}, that is, show that if we did have a decider for $Halt_{TM}$ then we could also build a decider for A_{TM}.

101. Consider the problem of whether a given context-free grammar with alphabet $\{\mathtt{0},\mathtt{1}\}$ is able to generate a string in $\mathtt{1}^*$. Is that decidable? In other words, is the language

$$\{\langle G\rangle \mid G \text{ is a context-free grammar over } \{\mathtt{0},\mathtt{1}\} \text{ and } L(G)\cap \mathtt{1}^* \neq \emptyset\}$$

decidable? Hint: Consider known closure properties for context-free languages.

102. Here is how we can see that the class of decidable languages is closed under intersection. Let A and B be decidable languages, let M_A be a decider for A and M_B a decider for B. We construct a decider for $A \cap B$ as a Turing machine which implements this routine:

On input w: (1) Run M_A on input w and reject if M_A rejects. (2) Run M_B on input w and reject if M_B rejects; else accept.

Show that the class of decid

103. (Drill.) The class of Turing recog ection, and the construction we gave in the previous question, for decidable languages, can equally be used to prove this. (Convince yourself that the argument is still right, even though we now have no guarantee that M_A and M_B always terminate.)

The class of Turing recognisable languages is also closed under union, but we can't argue that the same way, by constructing a Turing machine which effectively runs M_A and M_B, one after the other. (Why not?)

Show that the clas r union.

104. (Drill.) Show that th omplement. Why can't we use the same argument to show that the class of Turing reco ble languages is closed under complement?

105. (Drill.) Show that the class of decidable languages is c tenation, as well as under Kleene star.

106. (Optional.) Consider the alphabet $\Sigma = \{0,1\}$. The set Σ^* consists of all the *finite* bit strings, and the set, while infinite, turns out to be countable. (At first this may seem obvious, since we can use the function $\mathit{binary} : \mathbb{N} \to \Sigma^*$ defined by

$$\mathit{binary}(n) = \text{ the binary representation of } n$$

as enumerator; however, that is not a surjective function, because the legitimate use of leading zeros means there is no unique binary representation of n. For example, both 101 and 00101 denote 5. Instead the idea is to list all binary strings of length 0, then those of length 1, then those of length 2, and so on: $\epsilon, 0, 1, 00, 01, 10, 11, 000, 001, 010, 011, 100, 101, \ldots$.)

Now consider instead the set $\mathcal{B}$ of *infinite* bit strings. Show that $\mathcal{B}$ is much larger than Σ^*. More specifically, use diagonalisation to show that $\mathcal{B}$ is not countable.