The University of Melbourne
School of Computing and Information Systems
COMP30026 Models of Computation

Assignment 2, 2017

Released: 22 September. Deadline: 20 October at 23:00

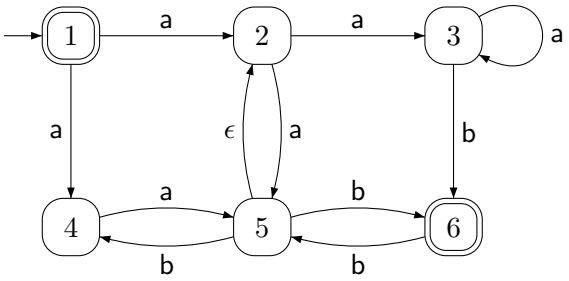
Purpose

To improve and consolidate your understanding of regular and context-free languages, and of finite-state automata and related computational devices. To develop skills in analysis and formal reasoning about complex concepts, and to practise writing down formal arguments.

Five challenges

Your answers to the first four should be submitted via Grok, using the module called "COMP30026 Assignment 2". Your answer to the fifth should be submitted via Turnitin on the LMS.

1. Construct a 5-state DFA d (with alphabet $\{\mathsf{a}, \mathsf{b}\}$) that is equivalent to the following NFA. The DFA must be complete, that is, its transition function is total, and it must have five states.



2. Assume the alphabet is $\{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$ and let L_e be the set of strings in which every substring of length 3 has exactly one c. To picture this, assume you have a cardboard sheet with a rectangle cut out, and the resulting "window" has a width that allows you to see 3 consecutive symbols through. If you take a string w from L_e of length 3 or more, and you slide this window along w, you will see exactly one c at any time. (If w has length 2 or less, there is no such constraint.) Formally,

$$L_e = \{w \in \Sigma^* \mid \forall x, y, z \in \Sigma^* (w = xyz \Rightarrow (|y| = 3 \Rightarrow y \text{ contains exactly one } \mathsf{c}))\}$$

   These are examples of strings in L_e: a, cc, bbc, $\mathsf{cabcbbca}$, $\mathsf{acaacbacbb}$. And these are examples of strings not in L_e: caaa, $\mathsf{bbcaacbcaac}$. Construct a DFA d (with alphabet $\{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$) that recognises L_e.

3. Let L be a language over alphabet Σ. Suppose we take the strings in L that have length greater than 0, and for each string we remove the first symbol, so we are left with

$$\mathit{tails}(L) = \{w \mid vw \in L, v \in \Sigma, \text{ and } w \in \Sigma^*\}$$

   (a) Let L_f be the regular language $(\mathsf{ab} \cup \mathsf{ba})^*(b \cup \epsilon)$. Construct a DFA f (with alphabet $\{\mathsf{a}, \mathsf{b}\}$) that recognises L_f.

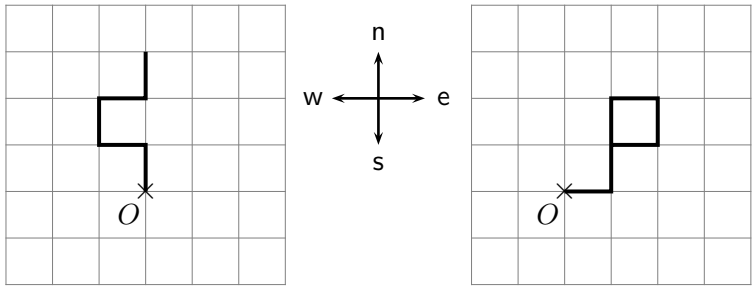
   (b) The language $L_g = \mathit{tails}(L_f)$ is regular (in fact, for every regular language R, $\mathit{tails}(R)$ is regular). Construct a DFA g (with alphabet $\{\mathsf{a}, \mathsf{b}\}$) for L_g.

4. A *Post machine* is similar to a push-down automaton (PDA), but instead of a stack it has a queue, into which its input is loaded. A special marker, '$', is placed at the end of the queue. After this, there is no other access to the original input, only to the queue (just like a Turing machine only has access to its tape). Symbols can only be added to the end of the queue, and removed from the front. Unlike a PDA, a Post machine is *deterministic*. A single move depends on the state and the symbol currently at the front of the queue. The move has three components: the new state, an indication of whether or not to remove the front symbol from the queue, and a symbol (or possibly a caret, ˆ, indicating nothing) to add to the rear.

   (a) Construct a Post machine p (with alphabet $\{\mathsf{a}, \mathsf{b}, \mathsf{c}\}$) which halts on all input and recognises
   $$L_p = \{\mathsf{a}^i\mathsf{b}^j\mathsf{c}^k \mid i, j, k > 0 \wedge i + j = k\}$$

   (b) (Harder.) Construct a Post machine q (with alphabet $\{\mathsf{a}, \mathsf{b}\}$) which halts on all input and recognises
   $$L_q = \{ww^{\mathcal{R}} \mid w \in \{\mathsf{a}, \mathsf{b}\}^*\}$$
   Here $w^{\mathcal{R}}$ denotes the string w reversed, so L_q consists of all the even-length palindromes over $\{\mathsf{a}, \mathsf{b}\}$, including the empty string ϵ. Hint: invent additional marker symbols to place in the queue.

5. A floor-polishing robot is located on an infinite floor, at a starting point O. It is programmed to make random walks across the floor, but in such a way that it always returns to O in finite time. The robot can only move in one-meter steps, one step at a time. Moreover, it can only make 90 degree turns. So each walk can be expressed by a finite string over the alphabet $\{\mathsf{e}, \mathsf{n}, \mathsf{s}, \mathsf{w}\}$, where e means "East", n "North", s "South", and w "West". Two walks are shown below, one that takes the robot 3 meters to the north of its starting point O, the other taking it back to O. (Note that the robot is allowed cross its path and/or travel repeatedly on the same segment.)



nwnen enneswsw

   A walk that makes the robot return to O is called *valid*. Let $L_r \subseteq \{\mathsf{e}, \mathsf{n}, \mathsf{s}, \mathsf{w}\}^*$ be the set of strings that correspond to valid robot walks. For example, nwnen $\notin L_r$, whereas enneswsw $\in L_r$.

   (a) Complete this definition: $L_r = \{x \in \{\mathsf{e}, \mathsf{n}, \mathsf{s}, \mathsf{w}\}^* \mid x \text{ contains } \ldots\}$.

   (b) Show that L_r is not context-free.

   (c) Give two context-free grammars G and G', so that $L_r = L(G) \cap L(G')$.

How to represent and submit your DFAs and Post machines

For testing we will use two emulators written in Haskell, one for DFAs and one for Post machines. The Grok Worksheets for Weeks 10 and 11 will include exercises to make you familiar with these Haskell programs, and you will see how DFAs and Post machines are to be represented as Haskell expressions. This will enable you to start developing solutions to Challenges 1–4 and test these solutions carefully. Their submission will be via the Grok "COMP30026 Assignment 2" module. This module opens on Friday 13 October, late in the day.

Both emulators allow the machines' transition functions to be partial. The DFA emulator will automatically "complete" your DFA if needed. This makes it easier to write DFAs, for example for Challenge 2. However, it is important to remember that, by definition, the transition functions are *total* functions, irrespective of what input to the machines might look like.

Submission and assessment

Some of the problems are harder than others. All should be solved and submitted by students individually. Your solution will count for 10 marks out of 100 for the subject. Each question is worth 2 marks. Marks are primarily allocated for correctness, but elegance and how clearly you communicate your thinking can also be taken into account.

The deadline is Friday 20 October at 23:00. Late submission will be possible, but a late submission penalty will apply: a flagfall of 2 marks, and then 1 mark per 12 hours late.

For Challenges 1 to 4, submit well-formed Haskell expressions on Grok, in the required format, as defined in the Week 10 and 11 worksheets. Name the expressions `d`, `e`, `f`, `g`, `p` and `q`, after the machines they represent. It is your responsibility to test these expressions for well-formedness and correctness. Note that on Grok, "saving" your file does not mean submitting it for marking. To submit, you need to click "mark". You can submit as many times as you like. What gets marked is the last submission you have made before the deadline.

As mentioned, the Grok "COMP30026 Assignment 2" module will be accessible from Friday 13 October. We encourage you to start completing the challenges well before that. The Week 10 and 11 Grok Worksheets will help you get to the point where you can develop and test your solutions carefully, so that they are ready for submission in Week 12.

For Challenge 5, submit a PDF document via the LMS. This document should be no more than 1 MB in size. If you produce an MS Word document, it must be exported and submitted as PDF, and satisfy the space limit of 1 MB. We also accept *neat* hand-written submissions, but these must be scanned and provided as PDF. If you scan your document, make sure you set the resolution so that the generated document is no more than 1 MB. Also please don't scan it upside down or sideways, as we mark it in front of a screen. As usual, I will leave the LaTeX source for this document in the content area where the PDF version is. For the automaton on page 1 I have used the `gastex` package, annotating the `gastex` code for easier understanding.

Make sure that you have enough time towards the end of the assignment to present your solutions carefully, in particular for Challenge 5. A nice presentation is sometimes more time consuming than solving the problems. Start early; time that you put in early usually turns out to be more productive than a last-minute effort.

Individual work is expected, but if you get stuck, email Matt or Harald a precise description of the problem, bring up the problem at the lecture, or (our preferred option) use the LMS discussion board. The COMP30026 LMS discussion forum is both useful and appropriate for this; soliciting help from sources other than the above will be considered cheating and will lead to disciplinary action.

Harald Søndergaard
21 September 2017