**School of Computing and Information Systems**
**COMP30026 Models of Computation Tutorial Week 10**

2–6 October 2017

## The plan

Reminder: Sipser's excellent introduction to regular languages is available under "Readings Online". Some of this week's exercises are taken from there. Make sure that you solve the exercises for this week (and Week 9) to get on top of automata theory. Question 68 is carried over from Week 9.

## The exercises

68. Give regular expressions for the following languages.

    (a) $\{w \mid w$ begins with a `1` and ends with a `0`$\}$

    (b) $\{w \mid w$ contains the substring `0101`$\}$ (so $w = x\texttt{0101}y$ for some strings $x$ and $y$)

    (c) $\{w \mid w$ has length at least 3 and its third symbol is `0`$\}$

    (d) $\{w \mid$ the length of $w$ is at most 5$\}$

    (e) $\{w \mid w$ is any string except `11` and `111`$\}$

    (f) $\{w \mid$ every odd position of $w$ is a `1`$\}$

    (g) $\{w \mid w$ contains at least two `0`s and at most one `1`$\}$

    (h) $\{\epsilon, \texttt{0}\}$

    (i) The empty set

    (j) All strings except the empty string

69. A *palindrome* is a string that reads the same forwards and backwards. Use the pumping lemma and/or closure results to prove that these languages are not regular:

    (a) $A = \{\texttt{0}^n\texttt{1}^n\texttt{2}^n \mid n \geq 0\}$

    (b) $B = \{\texttt{a}^i\texttt{ba}^j \mid i > j \geq 0\}$

    (c) $C = \{w \in \{\texttt{a}, \texttt{b}\}^* \mid w$ is not a palindrome$\}$

70. String $s$ is a *suffix* of string $t$ iff there exists some string $u$ (possibly empty) such that $t = us$. For any language $L$ we can define the set of suffixes of strings in $L$:

$$\mathit{suffix}(L) = \{x \mid x \text{ is a suffix of some } y \in L\}$$

    Let $A$ be any regular language. Show that $\mathit{suffix}(A)$ is a regular language. Hint: Think about how a DFA for $A$ can be transformed to recognise $\mathit{suffix}(A)$.

71. Consider the language

$$A = \left\{ w \;\middle|\; \begin{array}{l} w \text{ contains an even number of } \texttt{a}\text{s and an odd} \\ \text{number of } \texttt{b}\text{s and does not contain the substring } \texttt{ab} \end{array} \right\}$$

    over the alphabet $\Sigma = \{\texttt{a}, \texttt{b}\}$. Give a regular expression for $A$. (You may first want to think of a different way of expressing $A$ in English.)

72. Construct a five-state DFA which recognises $A$ from the previous question.

73. In general it is difficult, given a regular expression, to find a regular expression for its complement. However, it can be done, and you have been given all the necessary tricks and algorithms. This question asks you to go through the required steps for a particular example.

    Consider the regular language $(\mathtt{ba}^*\mathtt{a})^*$. Assuming the alphabet is $\Sigma = \{\mathtt{a}, \mathtt{b}\}$, we want to find a regular expression for its complement, that is, for

    $$L = \{w \in \{\mathtt{a}, \mathtt{b}\}^* \mid w \text{ is not in } (\mathtt{ba}^*\mathtt{a})^*\}$$

    To complete this task, go through the following steps.

    (a) Construct an NFA for $(\mathtt{ba}^*\mathtt{a})^*$. Two states suffice.

    (b) Turn the NFA into a DFA using the subset construction method.

    (c) Do the "complement trick" to get a DFA $D$ for $L$.

    (d) Reflect on the result: Wouldn't it have been better/easier to apply the "complement trick" directly to the NFA?

    (e) Turn DFA $D$ into a regular expression for $L$ using the NFA-to-regular-expression translation shown in the lecture on regular expressions (Lecture 15).

74. The Week 10 Grok Worksheet asks you to first complete a DFA simulator, written in Haskell, and then to apply it. This is of interest because of Assignment 2, which asks you to construct certain state machines and submit them, in the form of Haskell representations. If you get the simulator to work, this will allow you to test your DFAs thoroughly before submission.