Monash University
Faculty of Information Technology
2^{nd} Semester 2023

FIT2014
Exercises 3
Regular Languages, Inductive Definitions, Finite Automata, Kleene's Theorem I

There may not be time in class to cover all these exercises, so please attempt them *before* the class and make sure you ask questions about the ones you did not manage to solve confidently.

Remember that classes are not just about giving answers: you'll get these anyway, on Moodle, after the week of the class. The main aim is to *make you think* about the material covered during the lectures and, in particular, to generate discussion about some of the more interesting and challenging concepts.

The Supplementary Exercises are there for extra practice, and it is recommended that you attempt a selection of these, although you will not have time to cover them during the class.

ASSESSED PREPARATION: Question 5.
You must provide a serious attempt at this entire question at the start of your tutorial.

---

1. Write down all the words of length less than or equal to 8 described by the following regular expression.

$$(\mathbf{ab} \cup \mathbf{ba})(\mathbf{aa} \cup \mathbf{bb})^*\mathbf{ab}$$

2. For each of the following languages construct a regular expression defining each of the following languages over the alphabet $\{a, b\}$.

(i) All words that consist of exactly two letters.

(ii) All words that contain exactly two *b's* or exactly three *b's*, not more.

(iii) All strings that end in a double letter.

(iv) All strings that do not end in a double letter.

(v) All strings that have exactly one double letter in them.

(vi) All strings that have an even length.

(vii) All strings in which the letter b is never tripled. This means that no word contain the string bbb.

3. This question is inspired by the abstract game Zendo[1][2]

One round of the game has the following structure.

One player is known as the Master. When playing in the tutorial class, your Tutor will play this role to begin with. The Master does the following:

1. devise a regular expression, which we'll denote by R, and keep it secret;

[1]designed by Kory Heath (`http://www.koryheath.com/zendo/`), published by Looney Labs (`https://www.looneylabs.com/games/zendo`) on New Year's Eve, 1999

[2]Thanks to FIT2014 tutor Ben Jones for the idea for this question.

2. devise a string x which matches R, and another string y which does not match R;
3. reveal x and y to the other players, indicating which matches R and which does not.

The other players are each called Students. They do the following.

1. Each Student devise a string z and reveals it to everyone;
2. The Master tells everyone whether each student's string z matches R or not (but does not reveal R).
3. A Student — or group of Students working together — may guess what R is. Let S be the regular expression they guess.
4. If the guess S is right ($S = R$), then the Student wins this round, and the round ends.
   if the guess is wrong, then the Master devises and reveals to everyone a new string w, different from all previously revealed strings, on which R and S disagree (i.e., w matches R but not S, or matches S but not R).
5. Go back to Step 1.

Try this activity with a group of your fellow students. A group size of up to half-a-dozen should work ok.

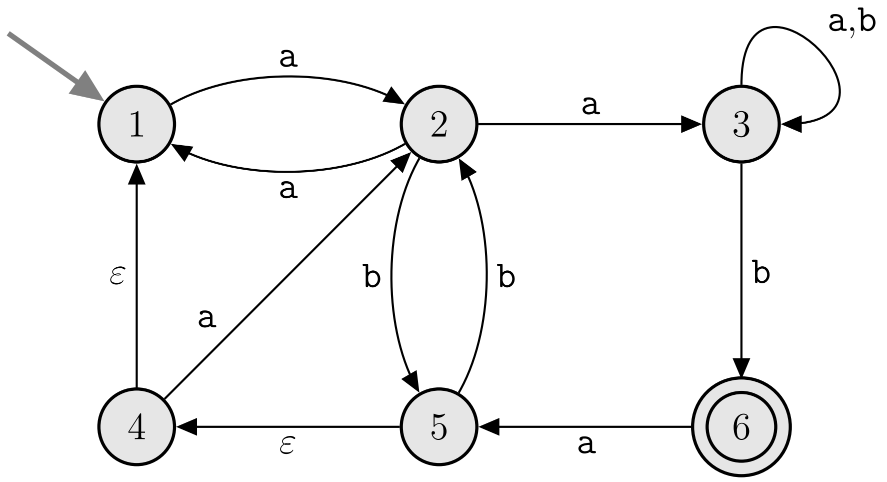
Some questions to consider as you do so:

- What sorts of regular expressions do you find work well for the Master?
- What sorts of regular expressions are easiest for the Students to guess?
- What modifications do you need to make to the rules, to make the game work better?

4. Build a Finite Automaton that accepts only those words that do not end in *ba*.

5.

Consider the following Nondeterministic Finite Automaton.



We will build up a proof, by induction on n, that

for all n, the string `(aaab)`^{n} is accepted by this NFA. (*)

(a) If the NFA is in state 1, what states can it be in after reading `aaab`?

(b) If the NFA is in state 6 and it then reads `aaab`, what states can it be in when it finishes reading that string?

(c) Is the string `aaab` accepted by the NFA? Why or why not?

Let $n \geq 1$.

(d) Suppose the string `(aaab)`^{n} is accepted by the NFA. What happens to the string `(aaab)`$^{n+1}$, i.e., does the NFA accept or reject it? Why?

(e) Explain how to put (some of) your answers together to make a proof by induction of (*).

6. Find a Nondeterministic Finite Automaton which accepts the languages defined by the following Regular Expressions:

(i) $(a \cup b)^*(aa \cup bb)$

(ii) $((a \cup b)(a \cup b))^*$

(iii) $(aa \cup bb)^*$

(iv) $(ab)^*aa(ba)^*$

(v) $(ab \cup ba)(aa \cup bb)^*(ab \cup ba)$

7. Convert each of the Nondeterministic Finite Automata you found in the previous question into Finite Automata.

Supplementary exercises

8. Write down all the words described by the following regular expression.

$(\mathbf{aa} \cup \mathbf{bb})(\mathbf{ba} \cup \mathbf{ab})(\mathbf{aa} \cup \mathbf{bb})$

9. Write down all the words of length 6 described by the following regular expression.

$(\mathbf{aa} \cup \mathbf{bb})*$

10. A programmer used the following regular expression to describe dates where the day, month, and year are separated by "/".

$([\mathbf{0}-\mathbf{9}][\mathbf{0}-\mathbf{9}] \cup [\mathbf{0}-\mathbf{9}])/([\mathbf{0}-\mathbf{9}][\mathbf{0}-\mathbf{9}])/[\mathbf{0}-\mathbf{9}]^+$

Give an example of a valid date not described by this regular expression, and an invalid date described by this regular expression.

11. Construct a regular expression to describe times for the twenty-four hour clock, where the hour, minute, and second are separated by ":".

12. In the book "*The C programming Language*", by B.W. Kernighan and D.M. Ritchie, a floating point number is defined as an optional sign, a string of numbers possibly containing a decimal point, and an optional exponent field containing an **E** or **e** followed by a possible signed integer. Construct a regular expression which describes a floating point number.

13.

Consider the recursive definition of regular expressions given in lectures.

Suppose we dropped subparts (i) and (ii) of part 3 of the definition (so we can't group or concatenate any more), but kept all other parts of the definition. What "regular expressions" would we get? What "regular languages" would result?

Challenge: Think about the effect of dropping other parts of the definition. Is any part of the definition redundant? If so, which one, and why? For any *essential* part of the definition, find a regular expression which would no longer satisfy the definition if that part were dropped.

14. In Linux, consult the man page for **grep** or **egrep** to learn the basics of how to use these utilities. (You may find **sed** and **wc** useful for this exercise too.)

Find a file of all English words, which may often be found in Unix/Linux systems in a location like `/usr/dict/words` or `/usr/share/dict/words`, or can easily be found on the web. You could also use one that you constructed in Lab 0.

Write a regular expression to match any vowel.

Write a regular expression to match any consonant.

Write a regular expression to match any word with no vowel or 'y'. Use **grep**, or one of its relatives, to find how many such words there are in your list.

Similarly, determine how many words have no consonants.

Write a regular expression to match any word whose letters alternate between consonants and vowels. What fraction of English words have this property?

What is the longest run of consonants in an English word? The longest run of vowels?

Can you use this tool to find a list of all English palindromes?

15. This question is about **one-dimensional Go**: see Tute 1, Q13.

Define

$\ell_{B,n}$	:=	the number of **legal** Go positions on the n-vertex path graph, where vertex n is **Black**.
$\ell_{W,n}$	:=	the number of **legal** Go positions on the n-vertex path graph, where vertex n is **White**.
$\ell_{U,n}$	:=	the number of **legal** Go positions on the n-vertex path graph, where vertex n is **Uncoloured**.
$a_{B,n}$	:=	the number of **almost legal** Go positions on the n-vertex path graph, where vertex n is **Black**.
$a_{W,n}$	:=	the number of **almost legal** Go positions on the n-vertex path graph, where vertex n is **White**.

(a) State the values of $\ell_{B,1}$, $\ell_{W,1}$, $\ell_{U,1}$, $a_{B,1}$, and $a_{W,1}$.
(b) Derive recursive expressions for $\ell_{B,n+1}$, $\ell_{W,n+1}$, $\ell_{U,n+1}$, $a_{B,n+1}$, and $a_{W,n+1}$, in terms of $\ell_{B,n}$, $\ell_{W,n}$, $\ell_{U,n}$, $a_{B,n}$, and $a_{W,n}$.
(c) How many of these quantities do you really need to keep, for each n, in order to work out the values for $n+1$?
(d) How do you work out the total number of legal positions on the n-vertex path graph, from $\ell_{B,n}$, $\ell_{W,n}$, $\ell_{U,n}$, $a_{B,n}$, and $a_{W,n}$?

16. This is a follow-up question to Q15, on one-dimensional Go.
(a) Write a regular expression for the set of strings that represent legal positions.
(b) Write a regular expression for the set of strings that represent almost legal positions.
(c) Does there exist a regular expression for the set of strings that represent positions that are neither legal nor almost legal?

17. In the lectures regular expressions were defined by a recursive definition. Give a recursive definition for arithmetic expressions, which use integers, the operators $+$, $-$, $/$, $*$, and brackets, ().

18. The language **PALINDROME** over the alphabet $\{a, b\}$ consists of those strings of a and b which are the same read forward or backward. Give a recursive definition for the language **PALINDROME**.

19. Construct a Finite Automaton that accepts only words with b as the second letter.

20. Construct a Finite Automaton that only accepts the words baa, ab, and abb and no other strings longer or shorter.

21. Build a Finite Automaton that accepts only those words that have *more* than four letters.

22. Build a Finite Automaton which only rejects the string aba.

23. Build a Finite Automaton that accepts only those words that have an even number of substrings ab.

24. Build a Finite Automaton that accepts precisely the strings of decimal digits that represent positive integers (with no leading 0s) that are multiples of 3.

25. Consider the Finite Automaton represented by the following table.

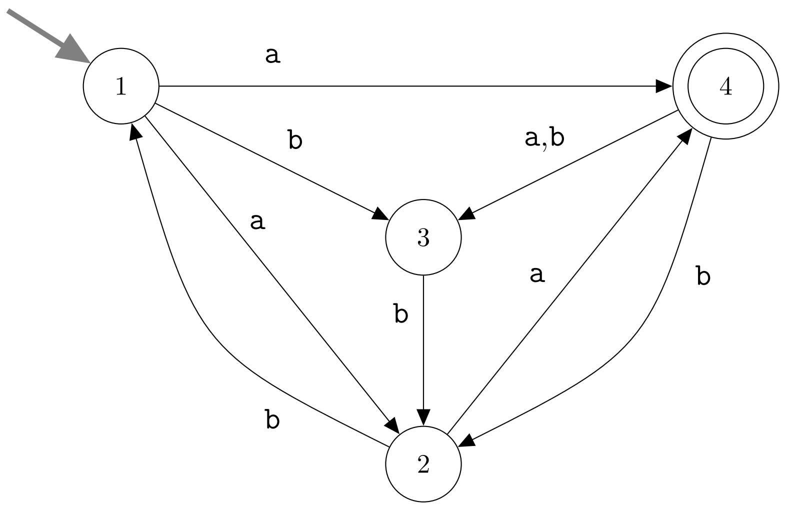
	state	`a`	`b`
Start	1	2	1
Final	2	1	2

Prove, by induction on the string length, that this FA accepts every string in which `a` occurs an *odd* number of times.

26.

From FIT2014 Final Exam, 2nd semester 2016:

Consider the following Nondeterministic Finite Automaton (NFA).



(a) What are the possible states that this NFA could be in, after reading the input string `abba`?

(b) Prove, by induction on n, that for all positive integers n, the string $(\texttt{abba})^n$ is accepted by this NFA. (This string is obtained by n repetitions of `abba`.)