https://folk.ntnu.no/michaeng/tdt4205/
michael.engel@ntnu.no

Assignments for Compiler Construction
Spring 2021

Theoretical Exercise Example Solution 2
Lexical Analysis

Please submit solutions on Blackboard by Friday, 12.02.2021 14:00h

2.1 Regular languages

a. A *palindrome* is a word (or set of words, ignoring whitespace, punctuation characters and capitalization) that reads the same when read from left to right and from right to left.

Simple examples are "*rotor*" and "*madam*", but you can also come up with sophisticated palindromes such as the Finnish word "*saippuakuppinippukauppias*" and sentences like "*A man, a plan, a canal – Panama*" (however, this example only works when ignoring any whitespace and punctuation characters).

Are palindromes regular languages? Justify your answer (note that we don't expect some formal proof here, rather some sort of intuition).

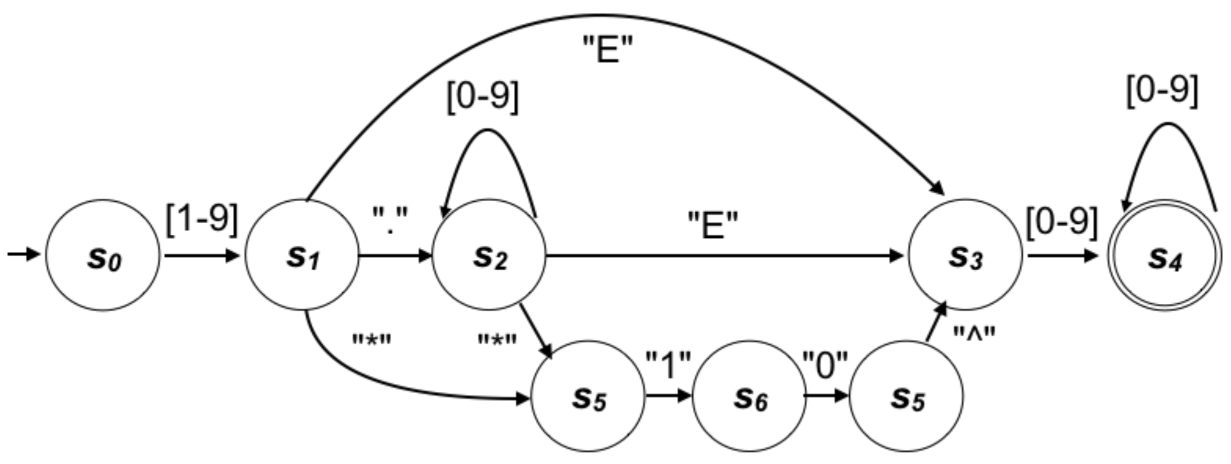
Detecting palindromes requires some sort of memory mechanism – you have to remember the first n characters and replay them in reverse order, e.g. using a stack.
This is only possible using recursion, so regular languages are not sufficient to detect palindromes.

b. In scientific notation, all numbers are written in the form $m \times 10^n$ or, as shorthand, mEn. The integer n is called the *order of magnitude* and the real number m is called the *significand* or *mantissa*.

In **normalized notation**, the exponent is chosen so that the absolute value (modulus) of the significand *m* is at least 1 but less than 10.

Examples for normalized notation are Avogadro's constant = $6.022E23$ or the speed of light $c = 2.99792458E8\frac{m}{s}$.

Draw a deterministic finite automaton (DFA) which accepts numbers in **normalized notation** (no units required) using either the "E" or the "10^n" notation, the latter written as "`*10^n`".



c. Create a regular expression that generates the language over the alphabet {a, b} of all strings containing an even number of bs.

Examples of strings in the language:
ϵ
a
aa
aaa
abba
bb
bab
abbaa
aababbb

Examples of strings *not* in the language:
b
abbb
babab
abbbabab
bbaaba

Hint: make sure that your regular expression generates bs in pairs.

The idea here is to always create two b's with an arbitrary (including zero) number of a's before, after or in between the b's:

```
(a*ba*ba*)*
```

We need the first a* to generate strings such as "abb" or "abbbb".
We need the middle a* to generate e.g. "abaaaaaba" or "bab".
And we need the final a* to generate e.g. "abbaa".

d. Create a regular expression that generates the language over the alphabet {a, b} of all strings in which

$n \bmod m = 2$

where n is the length of the string. In other words, all strings in the language have a length of 2, or 5, or 8, or 11, etc.

Examples of strings in the language:
ab
bb
baabb
aaaaa
abbbbbba

Examples of strings *not* in the language:
ϵ
a
abb
baaa
babbba

Hint: generate two initial letters, and expand the string by appending chunks of length 3.

```
[a|b][a|b] ([a|b][a|b][a|b])*
```

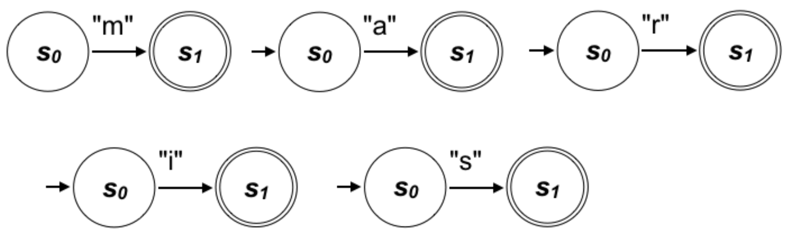
The first part [a|b][a|b] generates exactly two characters that can have all of the valid combinations aa, ab, ba and bb.
Similarly, the second part always generates three characters at the same time and this partial regexp can repeat an arbitrary number of times, including zero, due to the Kleene star, so the length of the string are 2, 5, 8, 11, ...

2.2 NFAs and DFAs

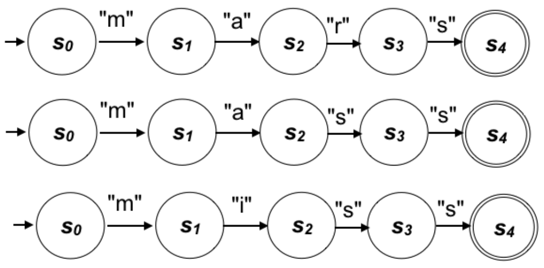
a. Construct an NFA that accepts the following regular expression:

```
(mars | mass | miss)
```

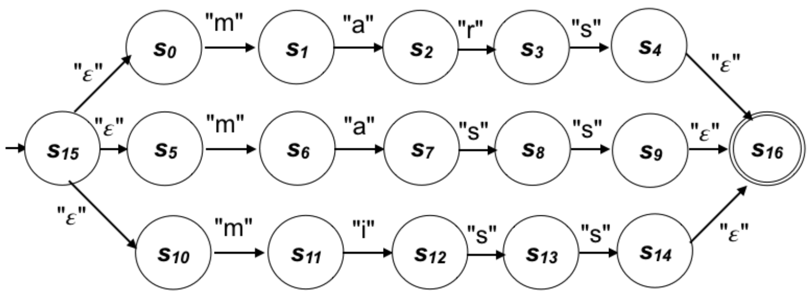
First, we construct simple scanners for each of the included letters:



Then, we combine the simple scanners into ones that detect each word separately:



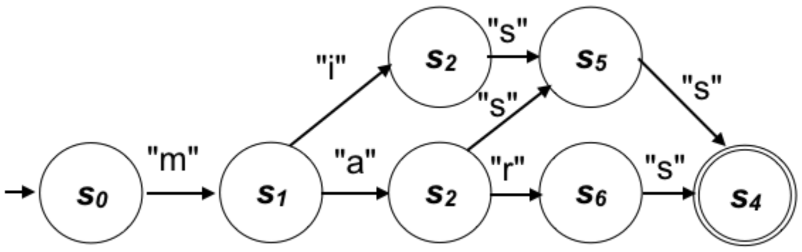
Finally, we combine these using ε-transitions:



b. Convert your NFA to a DFA using the subset construction algorithm.

- Initialization:
  $q_0 \leftarrow \varepsilon\text{–closure}(\{s_{15}\}) = \{s_{15}, s_0, s_5, s_{10}\};$
  $Q_0 \leftarrow q_0 = \{s_{15}, s_0, s_5, s_{10}\};$

A manually constructed DFA looks like this:



2.3 DFA minimization

Use the table method (Myhill-Nerode) to minimize the DFA given in Fig. 1. Show the steps performed, the related changes to the table and the final minimized DFA.

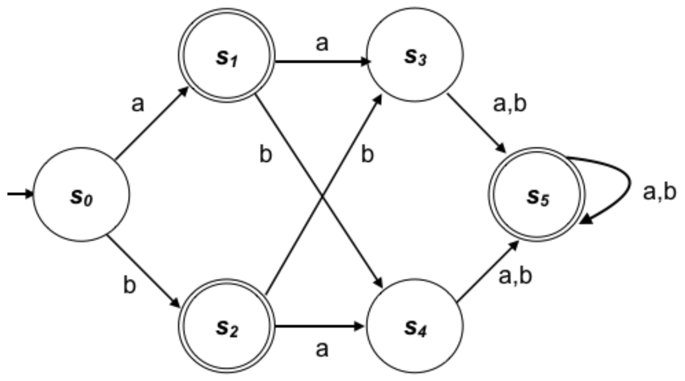


Figure 1: Non minimized DFA

- Construct a table with all pairs of states of the DFA. Mark all pairs (p, q) of states where p ∈ F and q ∉ F or vice versa. **Here: s_1, s_2 and s_5 are accepting states, the others are not.**

	s0	s1	s2	s3	s4	s5
s0						
s1	X					
s2	X					
s3		X	X			
s4		X	X			
s5	X			X	X	

- If there are any unmarked pairs (p, q) such that [δ(p, x), δ(q, x)] is marked, then mark [p, q] (here 'x' is an arbitrary input symbol) – new markings shown in red:

(s3, s0), x=a	**(s3, s0), x=b**	**(s4, s0), x=a**	**(s4, s0), x=b**
(s3, a) = s5	(s3, b) = s5	(s4, a) = s5	(s4, b) = s5
(s0, a) = s1	(s0, b) = s2	(s0, a) = s1	(s0, b) = s2

(s2, s1), x=a	**(s2, s1), x=b**	**(s1, s5), x=a**	**(s1, s5), x=b**
(s2, a) = s4	(s2, b) = s3	(s1, a) = s3	(s1, b) = s4
(s1, a) = s3	(s1, b) = s4	(s5, a) = s5	(s5, b) = s5

(s5, s2), x=a	**(s5, s2), x=b**	**(s4, s3), x=a**	**(s4, s3), x=b**
(s5, a) = s5	(s5, b) = s5	(s4, a) = s5	(s4, b) = s5
(s2, a) = s4	(s2, b) = s3	(s3, a) = s5	(s3, b) = s5

- This results in new markings at positions (s1, s5), (s2, s5):

	s0	s1	s2	s3	s4	s5
s0						
s1	X					
s2	X					
s3		X	X			
s4		X	X			
s5	X	X	X	X	X	

- Repeat until no more markings can be made:

(s3, s0), x=a	**(s3, s0), x=b**	**(s4, s0), x=a**	**(s4, s0), x=b**
(s3, a) = s5	(s3, b) = s5	(s4, a) = s5	(s4, b) = s5
(s0, a) = s1	(s0, b) = s2	(s0, a) = s1	(s0, b) = s2

(s2, s1), x=a	**(s2, s1), x=b**	**(s4, s3), x=a**	**(s4, s3), x=b**
(s2, a) = s4	(s2, b) = s3	(s4, a) = s5	(s4, b) = s5
(s1, a) = s3	(s1, b) = s4	(s3, a) = s5	(s3, b) = s5

- This results in new markings at positions (s3, s0), (s4, s0):

	s0	s1	s2	s3	s4	s5
s0						
s1	X					
s2	X					
s3	X	X	X			
s4	X	X	X			
s5	X	X	X	X	X	

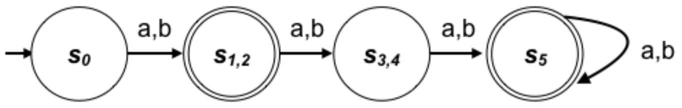
- Repeat until no more markings can be made:

(s2, s1), x=a	**(s2, s1), x=b**	**(s4, s3), x=a**	**(s4, s3), x=b**
(s2, a) = s4	(s2, b) = s3	(s4, a) = s5	(s4, b) = s5
(s1, a) = s3	(s1, b) = s4	(s3, a) = s5	(s3, b) = s5

- No new markings could be made in this iteration ⇒ terminate

  The resulting unmarks elements imply that state s1 ≈ s2 and state s3 ≈ s4.

  The resulting automaton looks like this:


s0
a,b
s1,2
a,b
s3,4
a,b
s5
a,b