Tutorial Sheet

Table of contents

1 Subjective Questions:

1. Define the following terms:
   a) Automaton
   b) Alphabet
   c) String
   d) Language
2. Given an alphabet $\Sigma = \{a, b\}$, list some valid strings over Σ.
3. Explain why we need a simpler way of discussing computing machines.
4. Describe the concept of an automaton and its role as a mathematical model of a computing device.
5. What are the reasons for building models in computing? Explain the significance of finite automata as abstractions of computers.

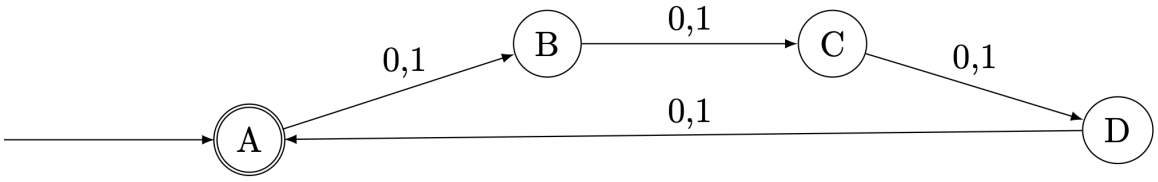
6. Define an alphabet and a string over an alphabet Σ. Provide examples.
7. Given an alphabet Σ = {a, b, c}, list some valid strings in the language of palindromes over Σ.

2 Objective Questions:

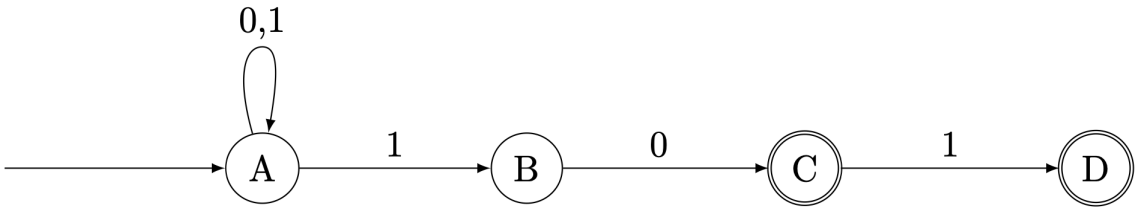
1. Define an NFA and explain how it differs from a DFA in terms of determinism.
2. Define the formal structure of a DFA and NFA (hint: look at the 5 tuple structure I disucssed in the lecture).

3 Design DFA

Q1. What language is accepted by the following DFA?



Q2.What language is accepted by the following NFA?



Q3. Construct a DFA that accepts binary strings of any length, except 3.

Q4. Let Σ = {a, b} and let L = {baa}. Design a DFA for L.

Q5. Let Σ = {a, b} and let L = { w Σ* | w = and the first and last character of w are the same }. Design a DFA for L.

Q6. Let Σ = {a, b} and let L = { w Σ* | w is a nonempty string whose characters alternate between a's and b's }. Design a DFA whose language is L.

4 Design NFA

Q7. Let $\Sigma = \{a, b, c\}$ and let L = { w $\in \Sigma^*$ | w ends in cab }. Design an NFA for L.

Q8. Let $\Sigma = \{a, b, c\}$ and let L = { w $\in \Sigma^*$ | some character in Σ appears at most twice in w }. Design an NFA for L.

Q9. Let $\Sigma = \{a, b\}$ and let L = { w $\in \Sigma^*$ | the third-from-last character of w is a }. Design an NFA for L. Your NFA should use at most four states.

5 More Practice Questions

Q10. Give NFAs with the specified number of states recognizing each of the following languages. In all cases, the alphabet is $\Sigma = \{0, 1\}$.

- Q10.1. The language { w $\in \Sigma*$ | w ends with 00 } with three states.
- Q10.2. The language { w $\in \Sigma*$ | w contains the substring 0101, i.e., w = x0101y for some x, y $\in \Sigma*$ } with five states.
- Q10.3. The language { w $\in \Sigma*$ | w contains at least two 0s, or exactly two 1s } with six states.
- Q10.4. The language { } with one state.

5.1 More simple/fun Questions

> ⚠ Warning
>
> **Important**: I will give you solutions to these questions but they may be slightly wrong, you need to figure out what is wrong and fix that [1].

1. **Heat Wave in The Netherlands**: Design a DFA to determine if a heat wave has occurred in The Netherlands (the definition of a heatwave is that the temperature should be greater than 25°C for 5 days straight).

L = { w in {<= 25, > 25}* | w { contains at least five consecutive } > 25 }

2. **Password Strength**: Design a DFA that accepts passwords that are at least 3 characters long, contain at least one uppercase letter, one lowercase letter, and one digit.

[1] Trust me this is better than having no solutions!

L = { w in {a-z, A-Z, 0-9}* | w { contains at least one uppercase, one lowercase, one digit, and has length } >= 3 }

3. **Elevator Button Presses**: Design an NFA where the elevator goes to the 5th floor only if the 2nd and 4th-floor buttons are pressed consecutively.

L = { w in {1,2,3,4,5}* | w { contains 2 followed by 4} }

4. **Traffic Light Sequence**: Design a DFA that represents the sequence of traffic lights (Red -> Green -> Yellow -> Red).

L = { w in {R, G, Y}* | w { follows the pattern } RGY }

5. **Coffee Machine**: Design a DFA that accepts sequences where a user selects a coffee type, adds sugar (optional), and then presses the start button.

L = { w in {coffee, sugar, start}* | w { starts with coffee and ends with start} }

6. **Library Book Return**: Design an NFA that accepts if a book is returned either on or before its due date or within two days after its due date.

L = { w in {on-time, late}* | w { is either on-time or late up to two days} }

7. **ATM Transactions**: Design a DFA that accepts sequences where a user inserts a card, enters a PIN, selects an account, chooses a transaction type, and then takes the cash or completes the transaction.

L = { w in {card, PIN, account, transaction, cash}* | w { starts with card and ends with cash or transaction} }

8. **Music Playlist Shuffle**: Design a DFA that shuffles songs without repeating the same song until all songs in the playlist have been played.

L = { w in {song_1, song_2, … song_n}* | w { does not repeat any song until all songs have been played} }