DM534 — Øvelser Uge 44

Introduktion til Datalogi, Efterår 2021

Jonas Vistrup og Rolf Fagerberg

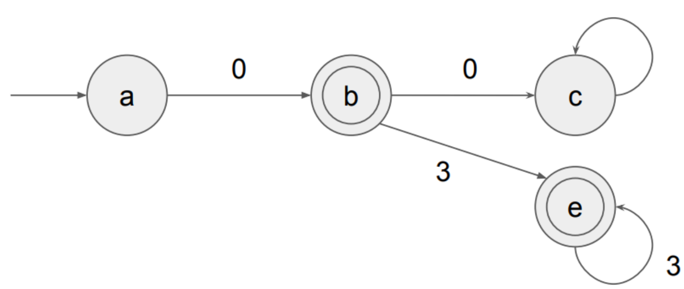
1 I

Start med at repetere følgende definitioner fra slides:

- En Deterministic Finite Automaton (DFA).
- At en DFA accepterer en streng.
- Sproget bestemt af en DFA.
- En Context-Free Grammar (CFG).
- At en CFG udleder (derives) en streng.
- Sproget bestemt af en CFG.

1.1

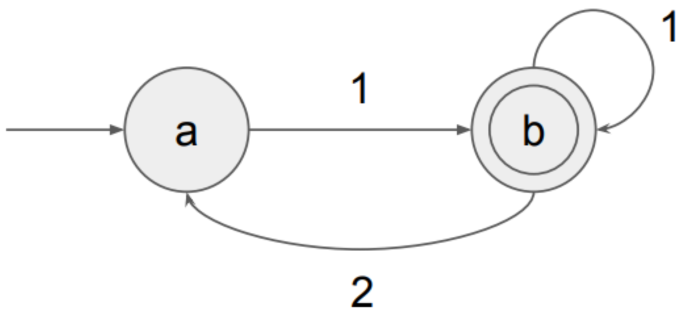
What is the language of the following DFA?



SVAR: 0 og så nul eller flere 3'er. Tilsvarende regex: 03*.

1.2

What is the language of the following DFA?

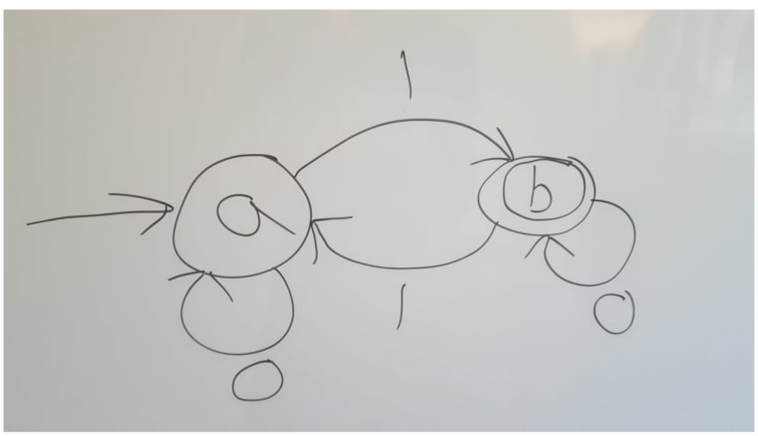


SVAR: 1 mindst en gang, hvis der er et 2, så skal det efterfølges af mindst et 1. Tilsvarende regex: 11*(211*)*.

1.3

Define a DFA that recognises the following language: All strings of 0s and 1s that contain an odd number of 1s and any number of 0s.

SVAR:

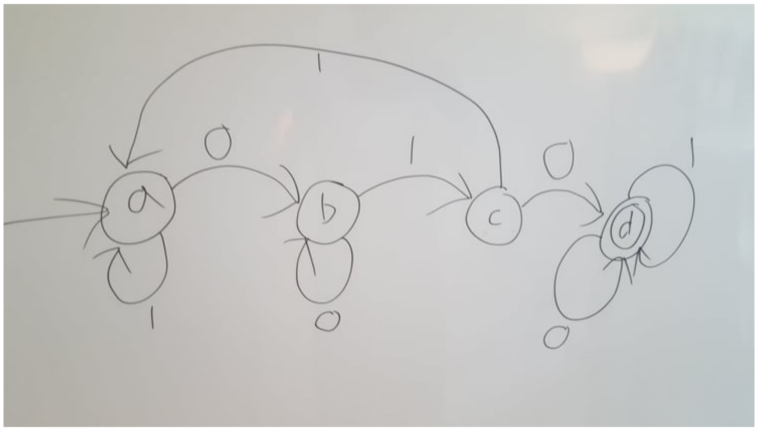


Tilsvarende regex: 0*1(0*10*1)*0*.

1.4

Define a DFA that recognises the following language: All strings of 0s and 1s that contain the string 010.

SVAR:

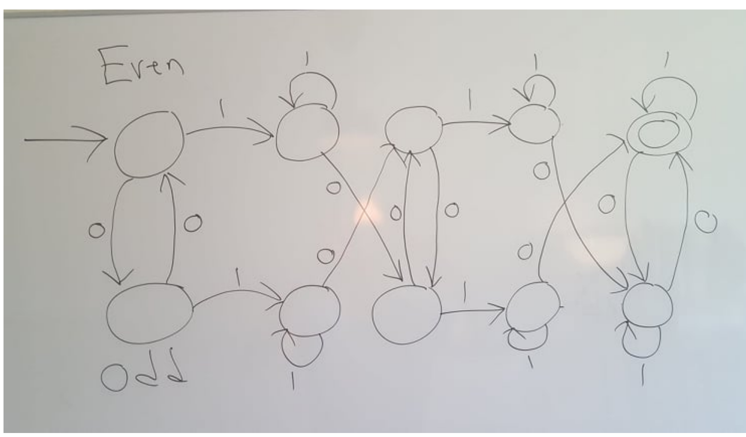


Tilsvarende regex: (0|1)*010(0|1)*.

1.5

Define a DFA that recognises the following language: All strings of 0s and 1s that contain at least two occurrences of 10 and an even number of 0s.

SVAR:



Den tilsvarende regex er lang, vi opbygger den derfor i moduler (dvs. nedenfor er EVEN, ODD, OO,... navne for del-regex'er). Mellemrum er kun medtaget for klarhedens skyld, de skal fjernes i den endelige regex.

EVEN = (1*01*0)*1*
ODD = 1*0 EVEN (Jvf. svaret i opgave 3, som er det samme, bare med 0 og 1 byttet om.)

OO = ODD 10 ODD 10 EVEN
OE = ODD 10 EVEN 10 ODD
EO = EVEN 10 ODD 10 ODD
EE = EVEN 10 EVEN 10 EVEN

Endelig regex: OO | OE | EO | EE

Idéen er en case baseret analyse: peg på de to 10 som strengen skal have, og se på antal 0 i resten af strengen (som falder i tre dele, hvor pariteten af den sidste bliver bestemt af pariteten af de to første, der er derfor $2 \times 2 = 4$ cases).

1.6

What is the language of the following CFG?

$$S \to ab$$
$$S \to SS$$

SVAR: $(ab)^+$. At least one repeat of the string ab.

1.7

Write two different derivations for the string 0001111 with the following CFG. (Same end result, but some different intermediate steps.)

$$S \to 0M1$$
$$M \to M1$$
$$M \to 0M$$
$$M \to 0$$
$$M \to 1$$

SVAR:

$$S \to 0M1 \to 00M1 \to 000M1 \to 000M11 \to 000M111 \to 0001111$$
$$S \to 0M1 \to 0M11 \to 0M111 \to 0M1111 \to 00M1111 \to 0001111$$

1.8

What is the language of the following CFG?

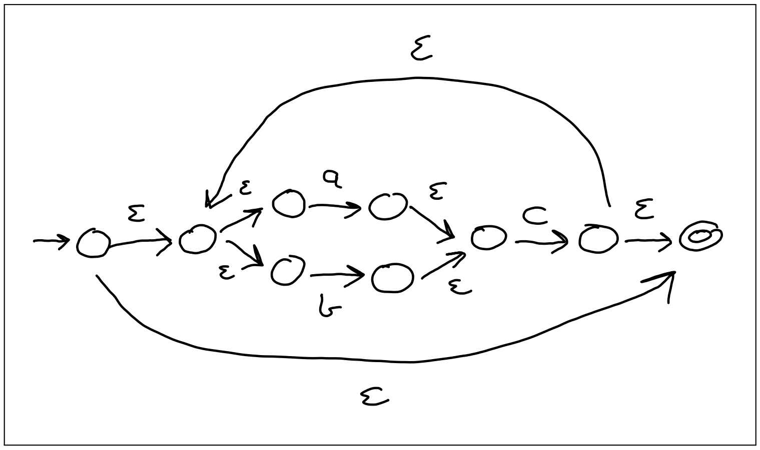
$$S \to 0MM1$$
$$M \to 0M$$
$$M \to 1M$$
$$M \to 0$$
$$M \to 1$$

SVAR: Any 0-1 string that starts with 0 and ends with 1 of at least length 4.

A

For det regulære udtryk $((\mathsf{a}|\mathsf{b})\mathsf{c})^*$, lav en ε-NFA med samme sprog ved at udføre Thompsons algoritme.

SVAR:



2 II

2.1

Define a CFG that recognizes the following language: All strings of 0s and 1s consisting of n 0s followed by n 1s. Examples: 0011 is OK, 1100 is not OK, 011 is not OK.

SVAR:

$$S \rightarrow 0S1$$

$$S \rightarrow 01$$

2.2

Define a DFA that recognises the same language of this CFG:

$$S \rightarrow 0M$$

$$S \rightarrow 1$$

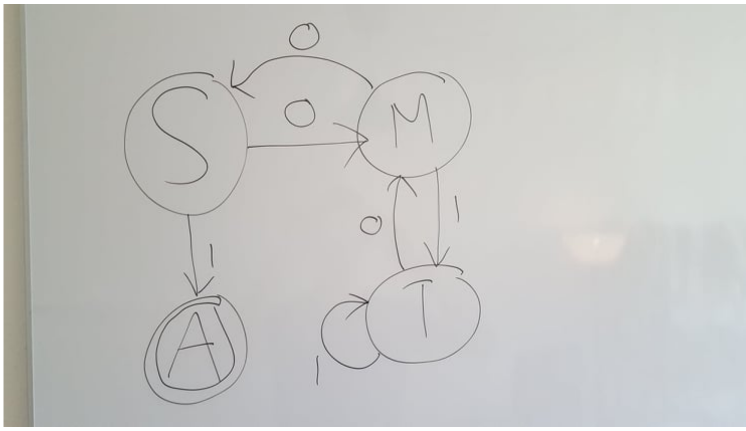
$$M \rightarrow 0S$$

$$M \rightarrow 1T$$

$$T \rightarrow 0M$$

$$T \rightarrow 1T$$

SVAR:



2.3

Define a CFG that recognises the following language: All strings of arithmetic additions that contain numbers, the + sign, and (balanced) parentheses. Examples: (0+1) is OK, (2+(3))+4 is OK, 2+3(2) is not OK.

SVAR:

$$S \to E$$
$$E \to (E)$$
$$E \to E + E$$
$$E \to N$$
$$N \to NN$$
$$N \to 0|1|2|3|4|5|6|7|8|9$$

B

Hvilket sprog beskriver hvert af følgende regulære udtryk?

(a) $(0|1)^*$

(b) $0^*|1^*$

(c) 0^*1^*

(d) $(0^*|1^*)^*$

(e) $(c^*)^*$

(f) ε^*

(g) $(\varepsilon^*)^*$

SVAR:

(a) Alle binære strenge (inkl. den tomme streng).

(b) Alle binære strenge som kun indeholder én type tegn, enten 0 eller 1 (inkl. den tomme streng).

(c) Alle binære strenge, som starter med nul eller flere 0 og derefter består af nul eller flere 1.

(d) Kleene star over sproget i (b). Da sproget i (b) indeholder strengene 0 og 1, indeholder sproget i (d) alle binære strenge. Omvendt, da sproget i (d) klart består af binære strenge, kan det ikke indeholde mere. Så de to sprog er ens. Kort sagt, svaret i (d) er: alle binære strenge.

(e) Da $\mathtt{c}^*$ indeholder strengen $\mathtt{c}$, indeholder $(\mathtt{c}^*)^*$ mindst $\mathtt{c}^*$. Omvendt består $(\mathtt{c}^*)^*$ klart af strenge af typen fra $\mathtt{c}^*$, så det kan ikke indeholde mere. Så $(\mathtt{c}^*)^*$ er lig $\mathtt{c}^*$. [Mere generelt gælder, med samme argument, at $(L^*)^* = L^*$ for alle sprog L.]

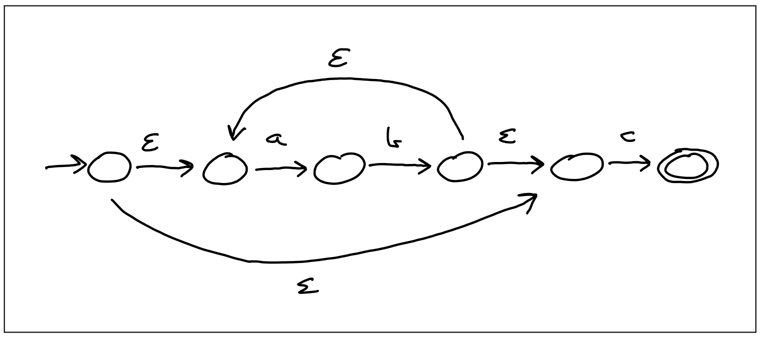
(f) Da $\varepsilon\varepsilon = \varepsilon$, giver Kleene star her ikke noget nyt. Så $\varepsilon^* = \varepsilon$ (altså sproget $\{\varepsilon\}$, der indeholder præcis den tomme streng og ikke flere strenge).

(g) Svaret er igen lig ε, dvs. sproget som indeholder præcis den tomme streng. Argument via (e) eller (f). [F.eks. via (f): $(\varepsilon^*)^* = \varepsilon^* = \varepsilon$.]

C

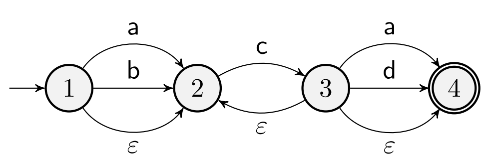
For det regulære udtryk $(\mathtt{ab})^*c$, lav en ε-NFA med samme sprog ved at udføre Thompsons algoritme.

SVAR:



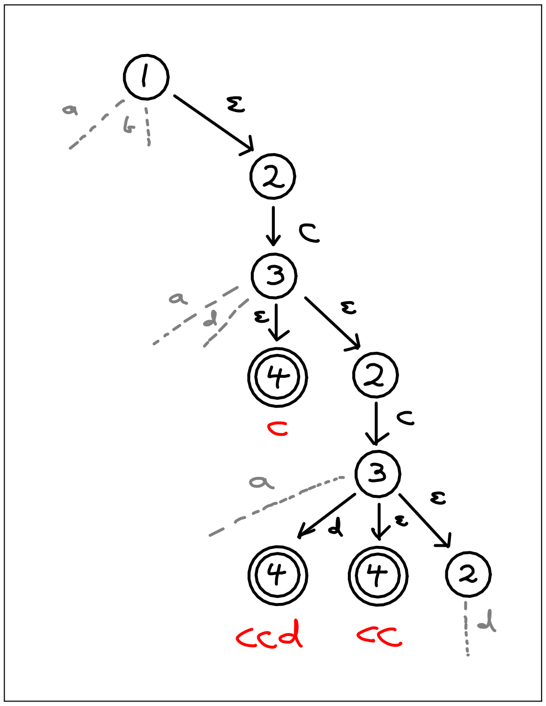
D

Vi ser på algoritmen CHECK, når den bruges på følgende ε-NFA:



Tegn et rekursionstræ for udførslen af algoritmen CHECK når input-strengen T er ccda. Hvilke strenge i starten af ccda matches?

SVAR:



Her vises alle iterationer i FOR-loop i algoritme, også dem som ikke giver anledning til rekursion - dem uden rekursion er vist i gråt. De matchede strenge i starten af `ccda` er vist i rødt.