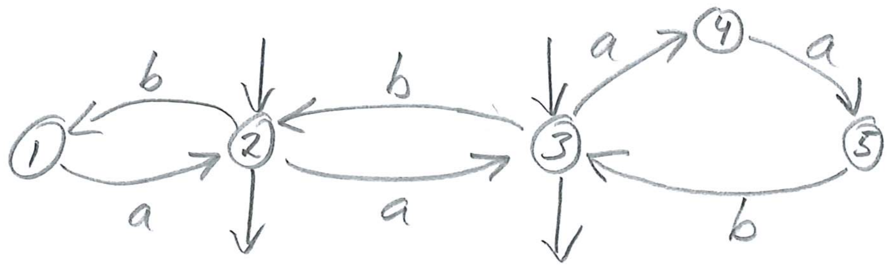
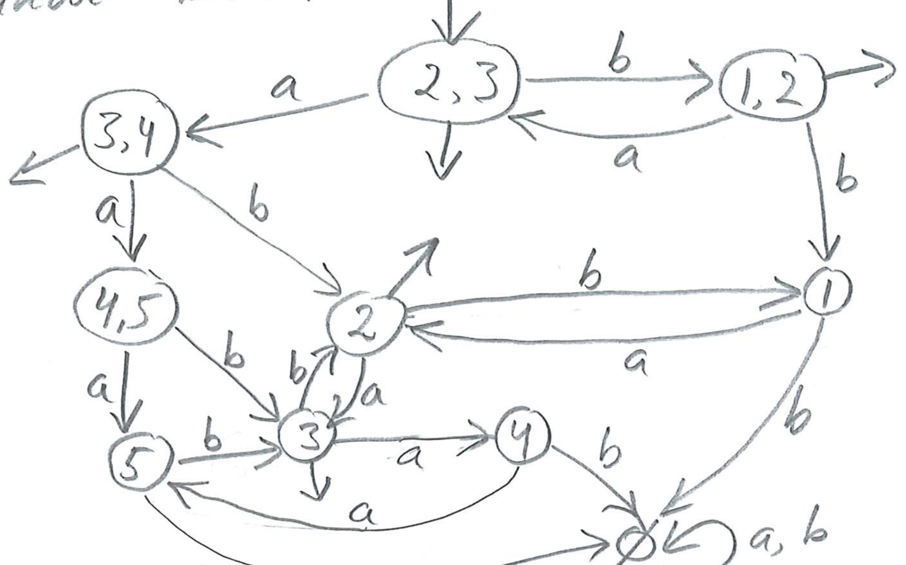
# Algoritmik 2020-10-20

## Svar/lösningsförslag

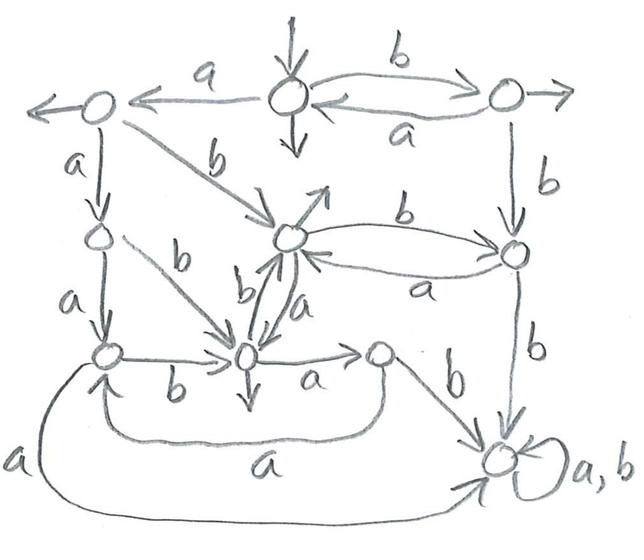
1. Först görs NFA:n icke-glupsk och tillstånden numreras:



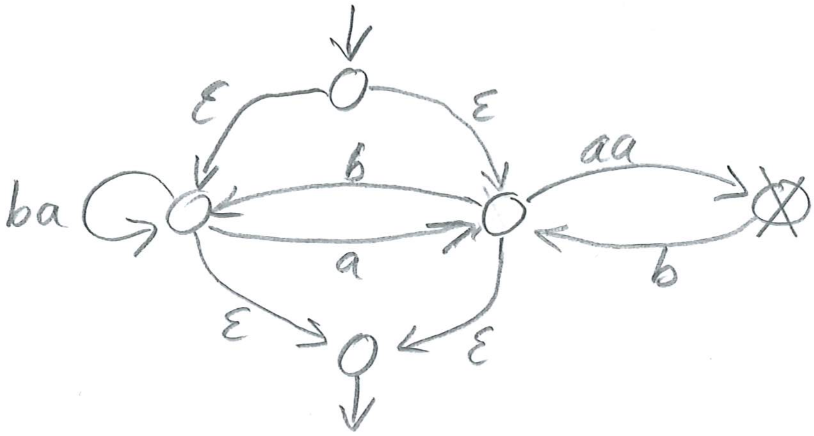

Med delmängdsalgoritmen får vi nu följande DFA:



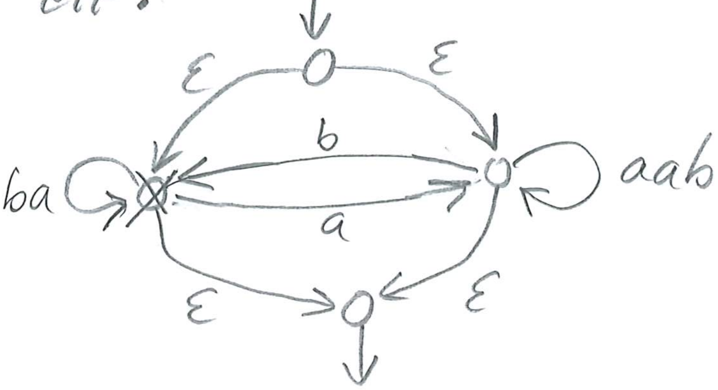

Tillsnyggat:

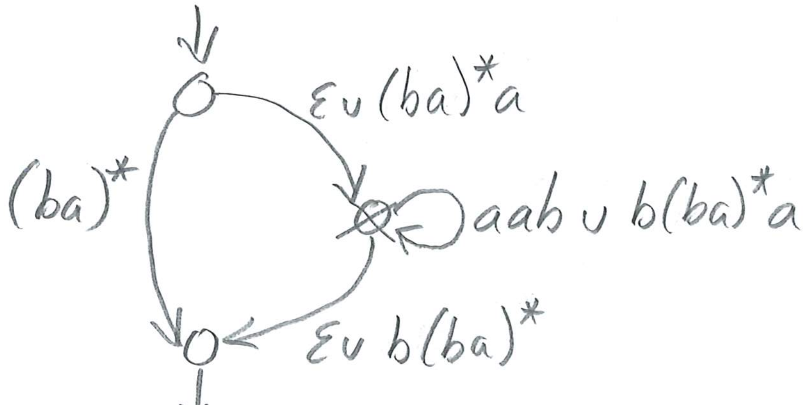


2. Först läggs nytt start- och accepterande tillstånd till:




Sedan elimineras alla gamla tillstånd, ett för ett:



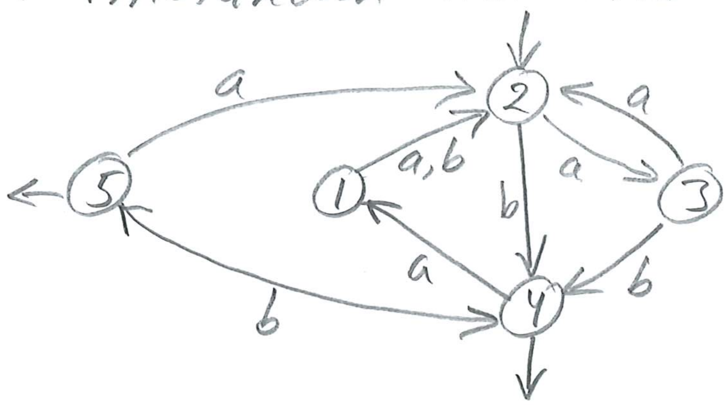




Obs! Jag gör vissa förenklingar på en gång


Det sista reguljära uttrycket beskriver språket som NFA:n accepterar.

3. DFA:n i uppgiften liknar den i duggan från september 2020, <u>men</u> de accepterande tillstånden är <u>inte</u> detsamma vilket gör att lösningen blir annorlunda här.

Om tillstånden numreras enligt figur



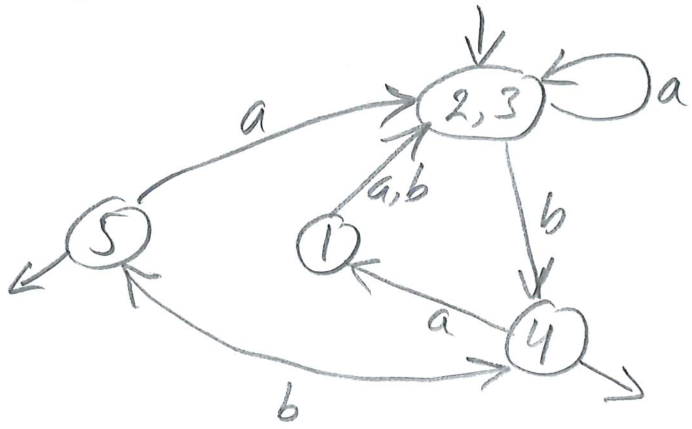

så kan tillståndsövergångarna beskrivas av denna tabell (som gör nästa steg enklare):

| | 1 | 2 | 3 | 4 | 5 |
|---|---|---|---|---|---|
| a | 2 | 3 | 2 | 1 | 2 |
| b | 2 | 4 | 4 | 5 | 4 |

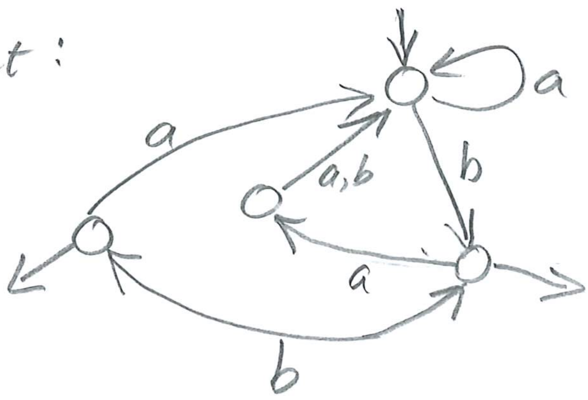
Sedan särskiljandealgoritmen:

| nivå | uppdelning |
|---|---|
| 1 | {1, 2, 3} {4, 5} |
| 2 | {1} {2, 3} {4, 5} |
| 3 | {1} {2, 3} {4} {5} |
| 4 | {1} {2, 3} {4} {5} |

När två nivåer ser likadana ut så terminerar algoritmen och vi får följande minimala DFA:




Tillsnyggat:




4. $L_2$ är reguljärt. Eftersom reguljära språk är slutna under komplement så räcker det att visa att språket $\overline{L_2} = \{w \in \{a,b\}^* :$ w har <u>minst tre</u> förekomster av ab <u>eller</u> <u>minst två</u> förekomster av bb $\}$

är reguljärt. $\overline{L_2}$ har följande reguljära uttryck: $(a \cup b)^* ab (a \cup b)^* ab (a \cup b)^* ab (a \cup b)^*$
$\cup\ (a \cup b)^* bb (a \cup b)^* bb (a \cup b)^* \cup (a \cup b)^* bbb (a \cup b)^*$

⑤

$L_1$ är *inte* reguljärt. Bevis med särskiljandesatsen. Notera först att strängen $(ab)^m b^n$ har $m$ förekomster av ab och $n$ förekomster av bb.

Låt $A = \{(ab)^m : m \in \mathbb{N}\}$ vilket uppenbarligen är en *oändlig* mängd.
Vi ska visa att om $x, y \in A$ och $x \neq y$ så särskiljs $x$ och $y$ av $L_1$. Då följer från särskiljandesatsen att $L_1$ inte är reguljärt.

Låt $x, y \in A$ och antag att $x \neq y$.
Då finns $m, n \in \mathbb{N}$ så att $m \neq n$, $x = (ab)^m$ och $y = (ab)^n$. Om $z = b^m$ så $xz = (ab)^m b^m \in L_1$ och $yz = (ab)^n b^m \notin L_1$ (eftersom $m \neq n$), vilket visar att $x$ och $y$ särskiljs av $L_1$.

*Kommentar:* Man kan också använda pumpsatsen för reguljära språk och välja t.ex. $u = (ab)^N$, $w = b^N$, $v = \varepsilon$, där $N$ ges av pumpsatsen. Om $w = xyz$ och $y \neq \varepsilon$ så $uxy^0zv \notin L_1$, men mer detaljer måste förstås ges för full poäng.

5 (a). abccbc accepteras och här är en körning:

| tape | stack | tillstånd (numrerade från vänster till höger) |
|---|---|---|
| abccbc | $\varepsilon$ | 1 |
| abccbc | Y | 2 |
| abccbc | xY | 2 |
| abccbc | xxY | 3 |
| abccbc | xY | 4 |
| abccbc | Y | 4 |
| abccbc | $\varepsilon$ | 1 |
| abccbc | Y | 2 |
| abccbc | xY | 3 |
| abccbc | Y | 4 |
| abccbc | $\varepsilon$ | 1 |

(Obs! Ibland har jag följt två övergångar på en gång.)

---

aabcccc accepteras <u>inte</u> för hur man än försöker läsa av strängen så "fastnar" man med ett 'c' kvar på tapen som inte kan avläsas för att det saknas 'x' överst på stacken.

(b) $L(M) = L^*$ där $L = \{a^n b^m c^{n+m} : n, m \in \mathbb{N}\}$.

(c)
$$S \to SS \mid X \mid \varepsilon$$
$$X \to aXc \mid Y \mid \varepsilon$$
$$Y \to bYc \mid \varepsilon$$

6. Strängen $1^8$ accepteras och här är en körning:

\#11111111
11111111 ($R_1 R_{\bar{x}}$ kors)
1x111111 ($x R_{\bar{x}}$ kors)
1x1x1111 ($R_1 x$ kors)
1x1x1111 ($R_{\bar{x}}$ kors)
1x1x1x11 ($R_1 x$ kors)
1x1x1x11 ($R_x$ kors)
1x1x1x1x ($R_1 x$ kors)
1x1x1x1x\# ($R_{\bar{x}}$ kors)

\#1x1x1x1x ($L_\#$ kors)
⋮
1x1x1x1x
⋮
1xxx1x1x
⋮
1xxx1xXX
1xxx1xxx\#
\#1xxx1xxX
⋮
1xxx1xxx
⋮
1xxxxxxx\#
\#1xxxxxxx
⋮
\#1xxxxxxX\# (terminerar)

Strängen $1^6$ accepteras <u>inte</u>, för TM:en går igenom den icke-blanka delen av tapen om och om igen och skriver över <u>varannan</u> etta med 'x'. För att TM:en inte ska hamna i en oändlig sökning så måste inputsträngen ha formen $1^{(2^n)}$ (alltså $2^n$ ettor) för något $n \in \mathbb{N}$.

(b) TM:en accepterar språket $\{1^{(2^n)} : n \in \mathbb{N}\}$. Se motivering i del (a).

7. $L_4$ är reguljärt och därmed sammanhangsfritt. Detta eftersom $L_4 = \{a,b\}^*$ som är reguljärt, med reguljärt uttryck $(a \cup b)^*$. För om $w \in \{a,b\}^*$ så låt $x = w$ och $n = 0$ och vi får $w = a^n x a^n$.

$L_3$ är sammanhangsfritt men <u>inte</u> reguljärt. En CFG för $L_3$:

$S \to aSa \mid b$.

Att $L_3$ inte är reguljärt kan t.ex. visas genom att visa att mängden $A = \{a^n b : n \in \mathbb{N}\}$ särskiljs av $L_3$, men jag gör inte detaljerna.

$L_5$ är <u>inte</u> sammanhangsfritt och därmed inte heller reguljärt.
Bevis med pumpsatsen (för CFL).

1. $L_5$ är oändligt för $a^n b a^n \in L_5$ för varje $n \in \mathbb{N}$.
2. Antag att $L_5$ är en CFL. (dvs ett sammanhangsfritt språk).
3. Låt $K$ vara givet av pumpsatsen.
4. Välj $w = a^K b^K b a^K b^K$ så $|w| \geq K$ och $w \in L_5$.
5. Antag att $w = uvxyz$, $vy \neq \varepsilon$ och $|vxy| \leq K$. Då är $vxy$ en delsträng av $a^K b^K$, $b^K b a^K$ eller $a^K b^K$.

I samtliga fall så gäller följande har $uvyz$ formen $a^n b^{m+1} a^i b^j$ och minst ett av följande gäller:

$n < i$, $i < n$, $m < j$ eller $j < m$.

I samtliga fall är det omöjligt att skriva $a^n b^{m+1} a^i b^j$ på formen $xbx$ för något $x \in \{a,b\}^*$, så $uvyz \notin L_5$.

6. Punkt 5 motsäger pumpsatsen så $L_5$ kan inte vara en CFL.

8. (a) Nej, ingen TM kan avgöra problemet.
Låt $\Omega = \{L : L$ är ett accepterbart språk som innehåller minst 20 strängar$\}$.

Enligt definitionen av $\Omega$ så är alla språk i $\Omega$ accepterbara. $\Omega$ är inte tom, för om $L$ beskrivs av $a^*$ så innehåller $L$ oändligt många strängar och är reguljärt, därmed accepterbart, och sålunda $L \in \Omega$. Men om $L = \emptyset$ så innehåller $L$ ingen sträng och är accepterbart. Så $\Omega$ innehåller inte alla accepterbara språk.

Från Rices sats följer att ingen TM kan avgöra, för godtycklig TM M, om $L(M) \in \Omega$ (dvs. om $L(M)$ innehåller minst 20 strängar).

(b) Detta problem är avgörbart. Jag beskriver en informell algoritm som avgör problemet och enligt Church-Turings tes finns en TM som avgör det.

ALGORITM: Låt en DFA M vara given som input. Undersök om det går att följa tillståndsövergångar från starttillståndet till ett accepterande tillstånd på så sätt att ett tillstånd besöks två gånger. Om det är möjligt så accepterar M oändligt många strängar så vi svarar 'ja'. Om det inte är möjligt så finns bara ändligt många sätt att ta sig från starttillståndet till ett accepterande tillstånd och varje "sätt" motsvarar en sträng som accepteras. Räkna nu alla sätten och svara 'ja' eller "nej" beroende på om det blev minst 20 eller ej.

(c) Problemet är avgörbart.
Observera först att inom 20 steg kan en TM bara besöka högst 20 rutor på tapen så om en TM M accepterar en sträng w inom 20 steg så accepteras även ett prefix till w som har längd $\leq 20$.

<u>ALGORITM</u>: Låt en TM M vara given som input. Gör en lista på alla strängar (över M:s inputalfabet) med längd högst 20. Det finns bara ändligt många. Kör nu M i (högst) 20 steg på var och en av strängarna i listan. Om M terminerar inom 20 steg för någon av strängarna så svara 'ja', annars 'nej'.

9. Låt M vara en PDA. Varje PDA-accepterbart språk är sammanhangsfritt. Så det finns en CFG G sådan att $L(G) = L(M)$.
För varje CFG så finns en (t.ex.) top-down parser med samma språk. Så det finns en top-down parser N s.a. $L(N) = L(G) = L(M)$.
Men en top-down parser har bara två tillstånd och vi är klara.