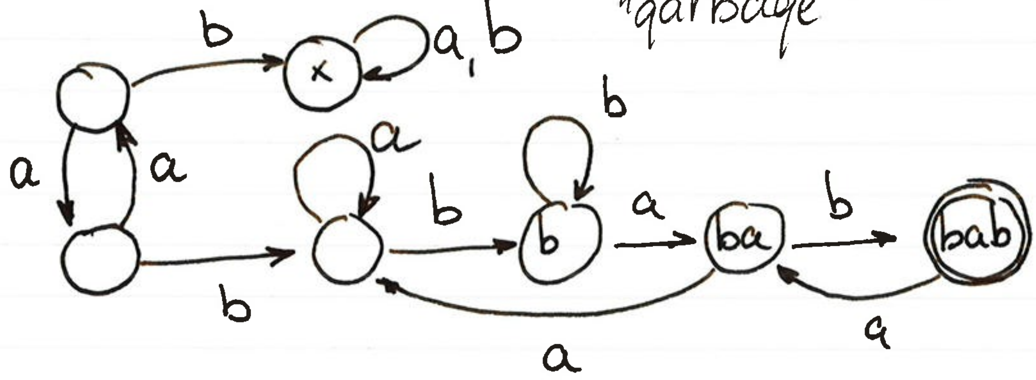
tentamen FI2 jan '20 uitwerkingen (en commentaar)

(1)



even aantal a's — suffix bab

(2) transitietabel (verzamelingen toestanden)

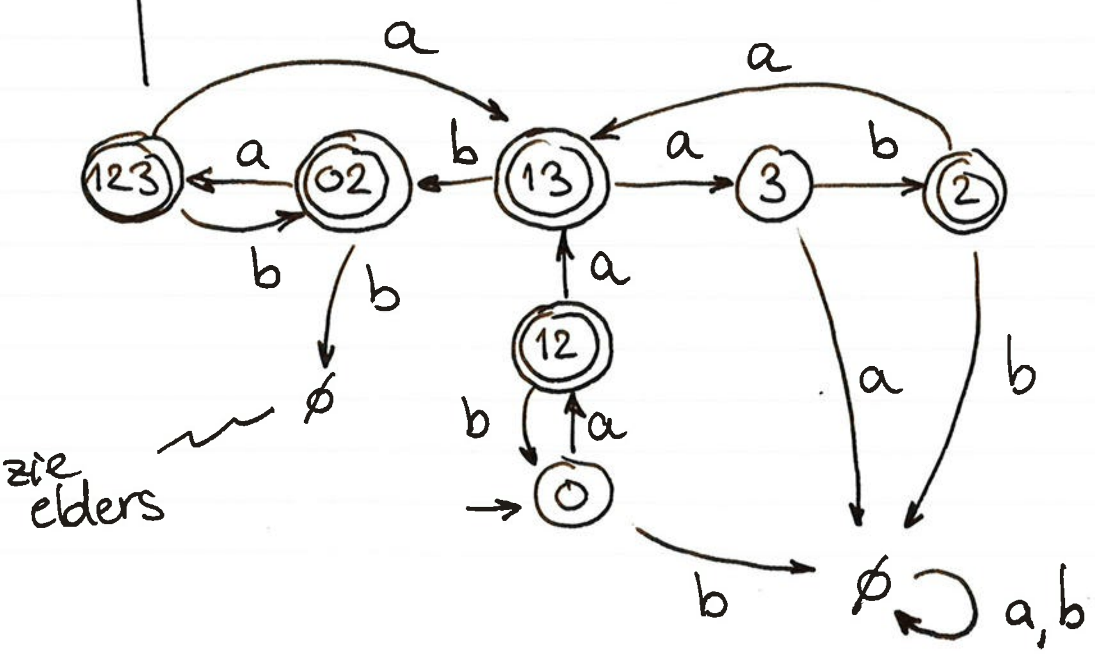
eigenlijke verzamelingen

	a	b
→ 0	12	– (leeg)
12	13	0
13	3	02
3	–	2
02	123	–
2	13	–
123	13	02
–	–	–

begintoestand {0}

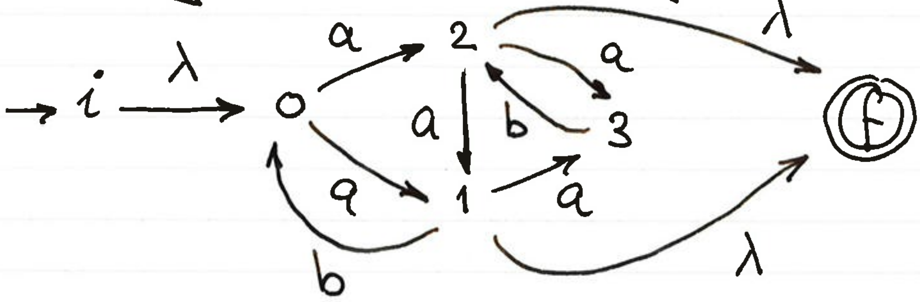
accepterend wanneer we 1 of 2 bereiken

(misschien) overbodig, maar hier het diagram



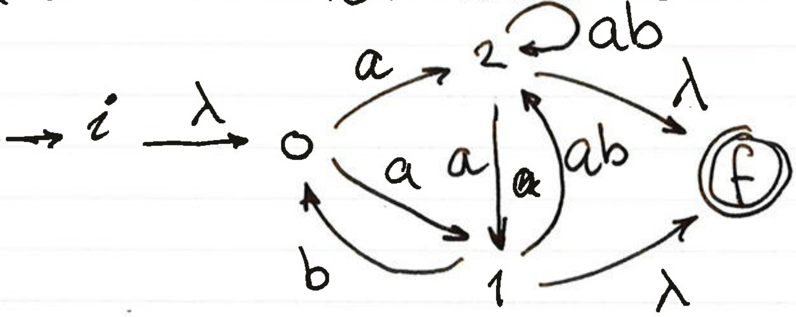
③ toestandeliminatie

* voeg eerst nieuw begin en eind to

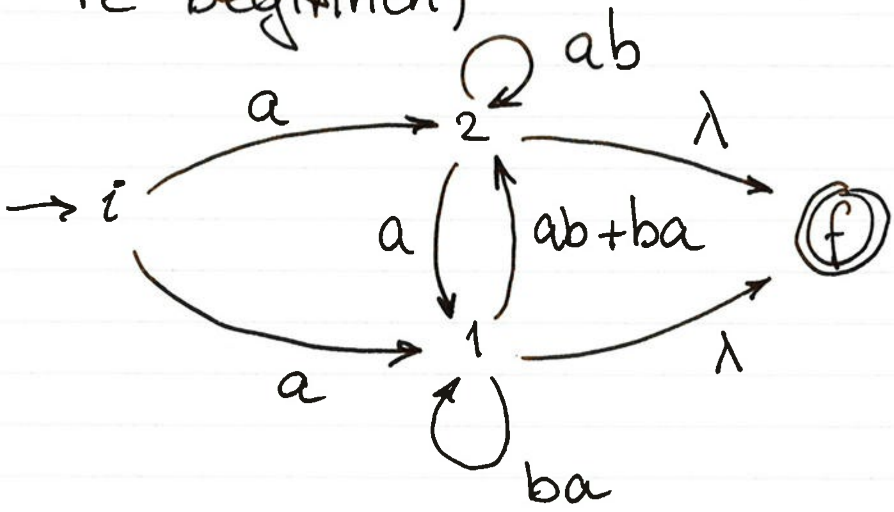


(als je dat niet doet raken er onderweg accepterende toestanden kwijt.)

* elimineer toestand 3

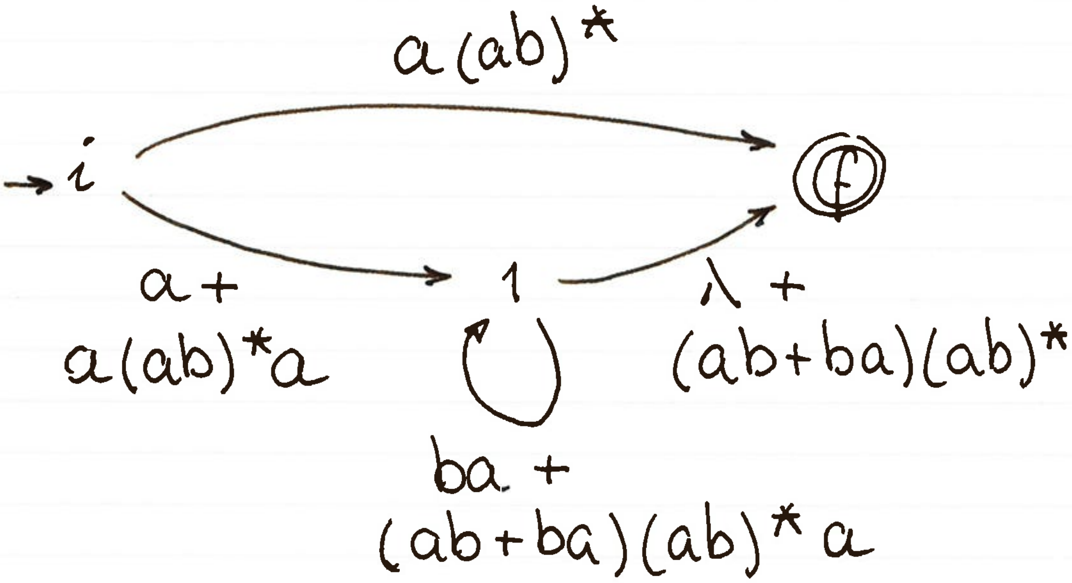


* elimineer toestand 1
(de volgorde is niet belangrijk, maar het lijkt eenvoudiger om met 'simpele' toestanden te beginnen)



③ vervolg

* elimineer toestand 2



als we uit eindelijk de laatste toestand 1 elimineren krijgen we één tak met de uiteindelijke expressie

$$a(ab)^* + (a + a(ab)^*a) \cdot (ba + (ab+ba)(ab)^*a)^* \cdot (\lambda + (ab+ba)(ab)^*)$$

(3) alternatief: algebraisch
modelleer de automaat met vergelijkingen

P_i: taal van alle woorden beginnend in (i)

$$P_0 = aP_1 + aP_2$$
$$P_1 = aP_3 + bP_0 + \lambda \qquad \text{(accepterend!)}$$
$$P_2 = aP_1 + aP_3 + \lambda$$
$$P_3 = bP_2$$

substitueer P_3

$$P_1 = abP_2 + bP_0 + \lambda$$
$$P_2 = aP_1 + abP_2 + \lambda$$

gebruik Arden op P_2

$$P_2 = (ab)^*(aP_1 + \lambda)$$

substitueer P_2

$$P_0 = aP_1 + a(ab)^*(aP_1 + \lambda)$$
$$= (a + a(ab)^*a)P_1 + a(ab)^*$$

$$P_1 = ab(ab)^*(aP_1 + \lambda) + bP_0 + \lambda$$

gebruik Arden op P_1

$$P_1 = (ab(ab)^*a)^* + bP_0 + ab(ab)^* + \lambda$$

vul nu P_1 in P_0 in, en gebruik Arden

(etc.)

④ gegeven de transitietabel (van de opgave)

δ:

	a	b
1	2	3
2	4	5
3	4	1
4	4	5
5	3	4

	1	2	3	4
2	0			
3	-	0		
4	0	-	0	
5	1	0	1	0

bepaal de paren toestanden (p,q) die niet equivalent zijn.

allereerst zijn dat paren waarvan de ene wel en de andere niet accepterend is.
dit zijn {1,3,5} vs. {2,4}
die markeer ik met 0

kijk nu of we nieuwe paren moeten markeren

$1 \xrightarrow{a} 2 \quad 5 \xrightarrow{a} 3$

omdat (2,3) gemarkeerd, markeren we (1,5)

$3 \xrightarrow{a} 4 \quad 5 \rightarrow 3$

idem, markeer (3,5)

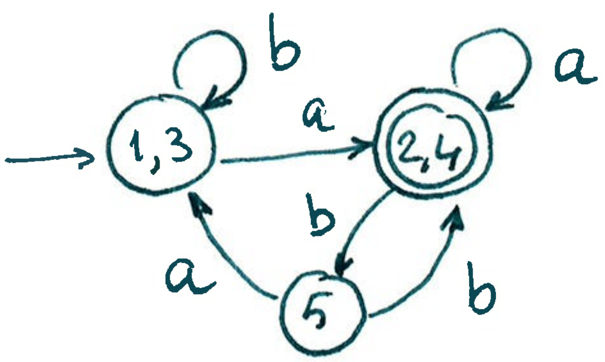
bv. (1,3) wordt niet gemarkeerd, want

$1 \xrightarrow{a} 2 \quad 3 \xrightarrow{a} 4$ en (2,4) is ok

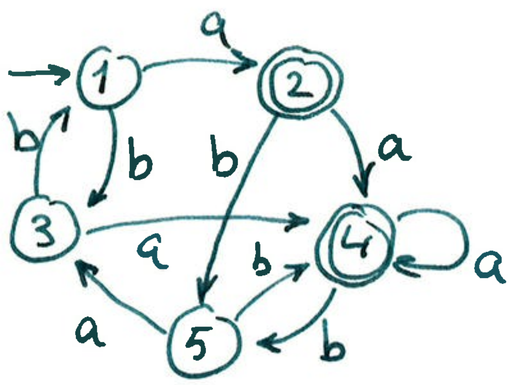
$1 \xrightarrow{b} 3 \quad 3 \xrightarrow{b} 1$ en (3,1) is ok

we houden de equivalente (ongemarkeerde) paren (1,3) en (2,4) over

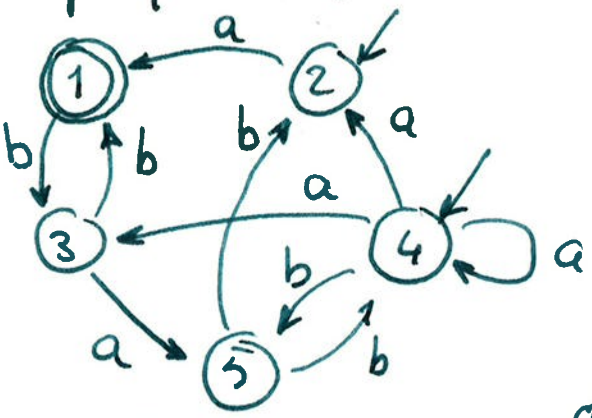
dus



④ toegift – extra
heb ik de methode van Brzozowski behandeld?
ik kwam dat een aantal maal tegen.
voor de grap

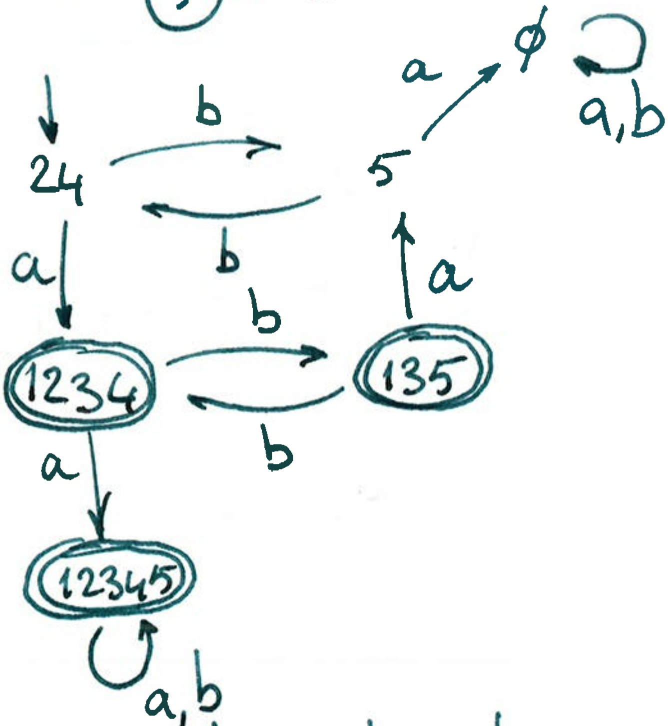


1. spiegelbeeld

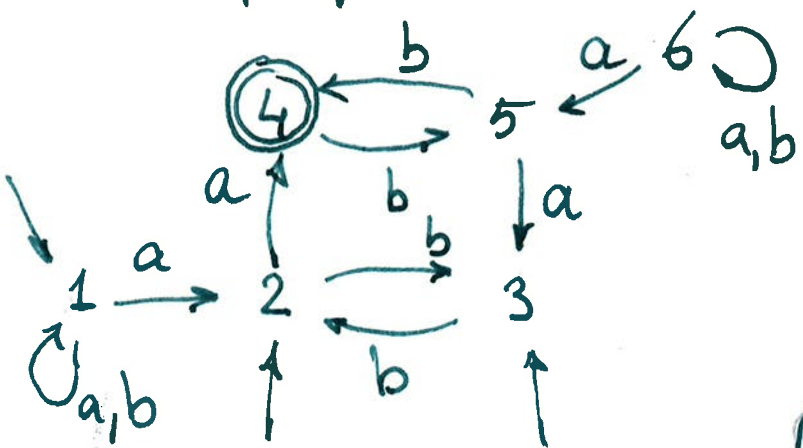


2. deterministisch
(subset)

	a	b
2,4	1234	5
1234	12345	135
5	–	24
12345	12345	12345
135	5	1234
–	–	–

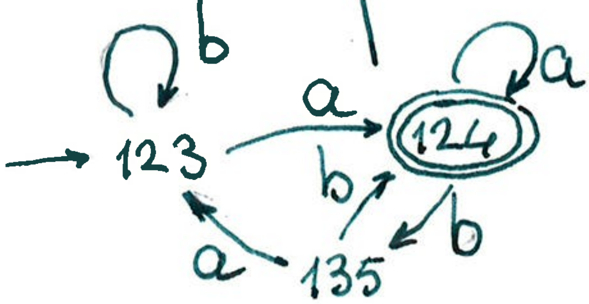


(hernoemen en)
3. spiegelbeeld



4. subsetconstructie
(again)

	a	b
123	124	123
124	124	135
135	123	124



(klaar)

5 a

idee: als twee delen van een string gelyke lengte moeten hebben, zullen die gelyktydig gegenereerd gaan worden. voorbeelden:

palindromen $S \to aSa \mid bSb \mid \lambda$

AnBn $S \to aSb \mid \lambda$

by de taal K

aaabbabbbb $i_1 = 3 \quad j_1 = 2 \quad i_2 = 1 \quad j_2 = 4$

$$
\begin{aligned}
S &\to aSb \mid A \mid B \\
A &\to aSa \mid X \\
B &\to bSb \mid X \\
X &\to bSa \mid \lambda
\end{aligned}
$$

buitenste
$j_1 < i_2$
$j_1 > i_2$
binnenste

b een woord $a^{i_1} b^{j_1} a^{i_2} b^{j_2}$ zit in $a^* b^*$ als $j_1 = 0$ òf $i_2 = 0$

we vinden dan woorden

$a^{i_1} a^{i_2} b^{j_2}$ met $i_1 = i_2 + j_2$ (vanwege K)

of $a^{i_1} b^{j_1} b^{j_2}$ $i_1 + j_1 = j_2$

dat is eenvoudiger dan het eerste onderdeel

$$
\begin{aligned}
S &\to aSb \mid A \mid B \\
A &\to aAa \mid \lambda \\
B &\to bBb \mid \lambda
\end{aligned}
$$

er geldt kennelyk $K \cap a^* b^* = (aa)^*$ ~~AeqB~~ AnBn $\cup$ ~~AeqB~~ AnBn $(bb)^*$

ik denk zelfs dat de doorsnede regulier is:

$K \cap a^* b^* = \{ a^i b^j \mid i + j \text{ is even} \}$

⑥ neem aan dat K regulier is.
dan bestaat er een constante p
zodanig dat voor elke string $z \in K$, met $|z| \geq p$
we een opdeling kunnen maken

$$z = uvw$$

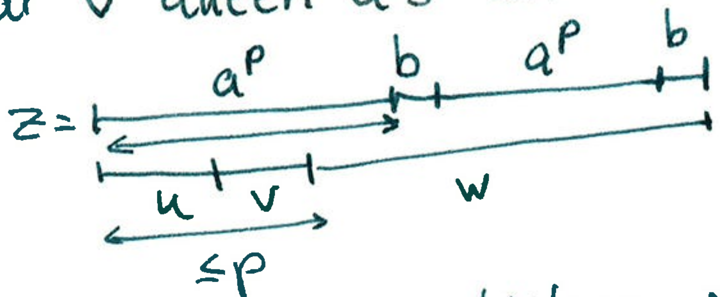
met $|v| \neq 0$ (tenminste iets pompen)
$|uv| \leq p$ (aan het begin)
en $uv^i w \in K$ voor elke $i \geq 0$

n.b. z moet afhangen van de constante p

we laten zien dat voor $z = a^p b a^p b$
zo'n opdeling niet bestaat
(en daarom geldt het pomplemma niet voor
K, en is K dus niet regulier).

(natuurlijk $z \in K$ want $p+1 = p+1$
en ook $|z| \geq p$, dus z voldoet om te
mogen pompen.)

als we schrijven $z = uvw$ met $|uv| \leq p$
dan bevat v alleen a's uit het eerste deel



dus gaan we pompen en kijken we naar
uv^0w dan is dat woord van de vorm
$a^q b a^p b$ met $q < p$ omdat er a's
weggehaald zijn.
dit woord ~~a~~ $uv^0w = a^q b a^p b \notin K$
want $q+1 \neq p+1$. (klaar).

bij ⑥

Standaard fouten

- het kiezen van een vaste string

  "$z = aaabaaab$"

  niet juist, alleen woorden langer dan de pompconstante worden gepompt. die waarde is gerelateerd aan het aantal toestanden van de automaat, en dat kan willekeurig groot zijn (dus groter dan 8 in het ~~z~~ voorbeeld van deze z).

- het kiezen van een vaste opdeling

  "neem $v = ab$"

  fout, want we laten zien dat pompen niet gaat - hoe we het ook proberen!

  bv. $(aab)^*$ is regulier, en de herhaling is duidelijk als we zelf $v = ab$ kiezen krijgen we inderdaad rare dingen.

(7) start met de λ-producties

$$N_1 = \{ A \in V \mid A \to \lambda \in P \}$$

bepaal dan herhaald de variabelen waarvan de rechterkant al nullable is

$$N_{i+1} = N_i \cup \{ A \in V \mid A \to \alpha \in P \text{ met } \alpha \in N_i^* \}$$

stop wanneer geen nieuwe variabelen gevonden worden, dus als $N_{i+1} = N_i$.
(dat gebeurt altyd, want er zyn natuurlyk eindig veel variabelen).

let op: nullable werkt niet alleen via keten-producties, bv. S is nullable by:

$$S \to AB \quad A \to BB \quad B \to \lambda$$

⑧ uit een blad met louter producties
wordt ik niets wyzer – graag wat uitleg

$$\left.\begin{array}{l} S \to aX \mid byX \\ X \to S \mid xy \mid \lambda \\ y \to XbS \mid X \end{array}\right\}\ \text{oorspronkelyk}$$

de nullable variabelen zyn
X (vanwege $X \to \lambda$) en y (via $y \to X$)
verwyder nu de λ-regels

$$S \to aX \mid byX \mid a \mid by \mid bX \mid b$$
$$X \to S \mid xy \mid x \mid y$$
$$y \to XbS \mid X \mid bS$$

de ketenregels zyn
$X \to S,\ X \to X,\ X \to y,\ y \to X$
voeg de regels van S toe aan die van X
en vervolgens die van X en y aan elkaar.
(nu zyn X en y gelyk, en kunnen we ze vervangen door één symbool – ik doe dat niet, maar schryf alleen de regels voor X)

$$S \to aX \mid byX \mid a \mid by \mid bX \mid b$$
$$X \to xy \mid \underbrace{xbS \mid bS}_{\text{via } y} \mid \underbrace{aX \mid byX \mid a \mid by \mid bX \mid b}_{\text{via } S}$$

voer variabelen in voor de terminalen

$$S \to AX \mid ByX \mid a \mid By \mid BX \mid b$$
$$X \to xy \mid XBS \mid BS \mid AX \mid ByX \mid a \mid By \mid BX \mid b$$
$$A \to a \qquad B \to b$$

⑧ vervolg

splits de regels die te lang zyn, dat zyn

$S \to BYX$
$X \to XBS \mid BYX$

door de nieuwe regels

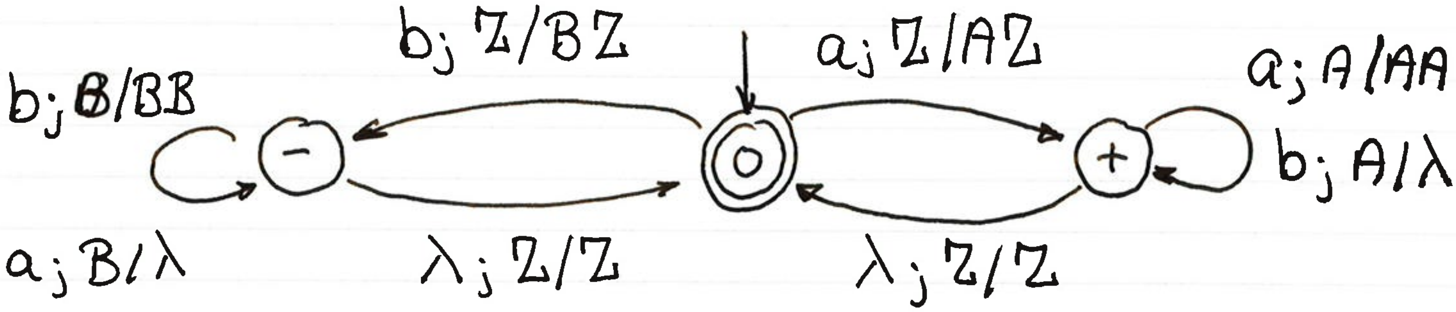
$S \to BP$
$X \to XQ \mid BP$
$P \to YX$
$Q \to BS$

(alle overige regels blyven staan).

nu zyn we Chomsky bestendig.

ps. het is niet nodig een speciale beginregel $S_0 \to S$ te introduceren (dat hebben jullie zeker van internet?)

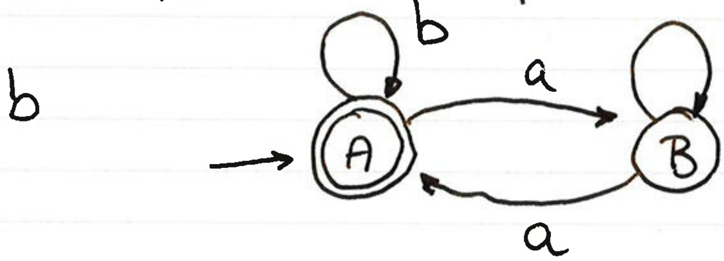
(9) a   op de stapel $n_a(w) - n_b(w)$ byhouden



by lege stapel, terug naar nul

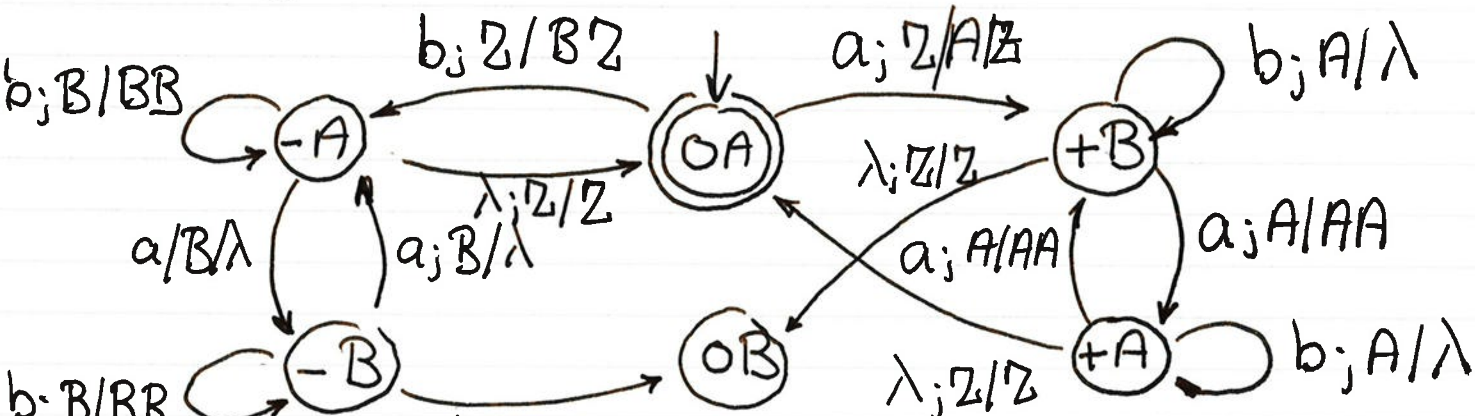
A erby/af

(eigenlyk kan voor ⊖ en ⊕ dezelfde toestand gekozen worden, omdat ± al door het stapelelement bepaald wordt.)

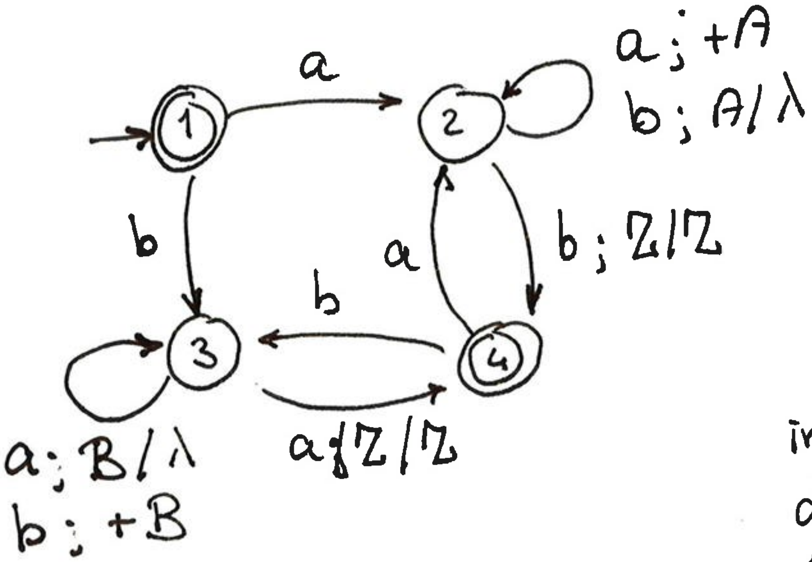


(dat kan echt in twee toestanden -natuurlyk)

productconstructie: leest letters tegelykertyd, als stapelautomaat λ-overgang, blyft eindige automaat dezelfde toestand



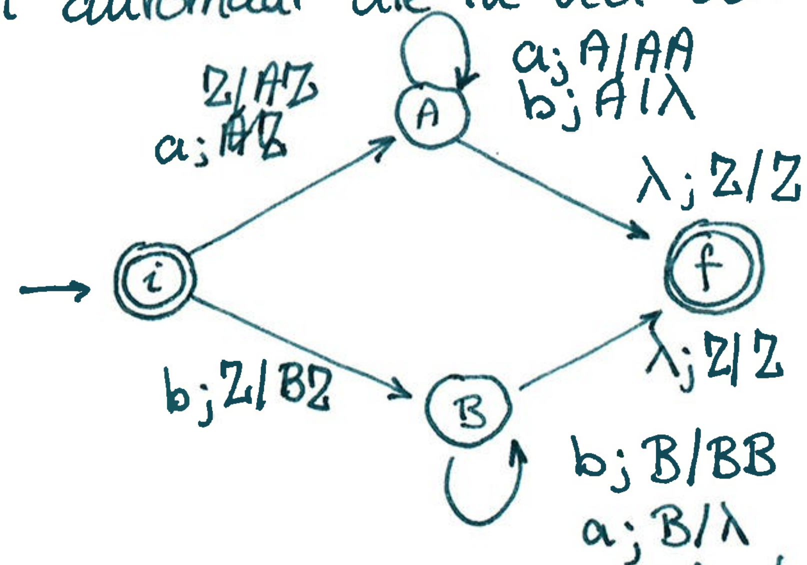
9 a toegift.
kan ook zonder λ-overgangen
(zag ik bij een van de uitwerkingen)



in ③ en ④ geldt dat de Z onderop de stapel ook meetelt bij het bijhouden van het aantal.

---

een automaat die ik veel ben tegengekomen:



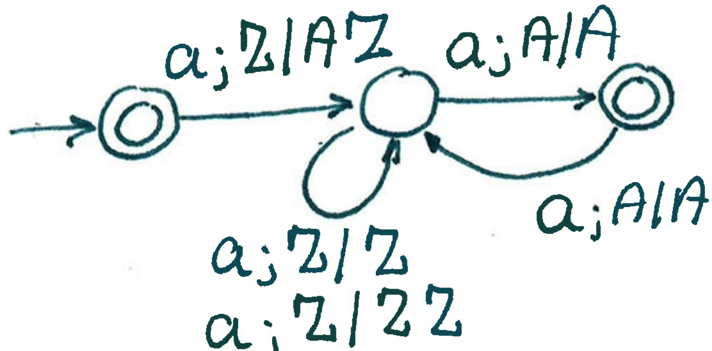
het probleem daarmee is dat als we met a beginnen altijd er meer a's dan b's gelezen moeten worden – de string abba wordt niet geaccepteerd,
dus in toestand f weer opnieuw beginnen!

⑩ niet waar.

de reden is dat determinisme per toestand bekeken wordt, en niet per berekening.

de volgende automaat voor de taal $(aa)^*$

zet eerst een extra A op de stapel, en bereikt dan een niet-det toestand (omdat er bij Z op de stapel twee instructies zijn), maar omdat A op de stapel staat kan de automaat slechts één stap kiezen.



(het is niet zo dat een deterministische stapelautomaat precies één stap heeft, zoals bij eindige automaten: het kan zijn dat er geen stap mogelijk is - de automaat "blokkeert")

wegens slecht gemaakt en ook verwarring over de interpretatie (wat is uniek, bedoel je accepterende berekening ...) niet meegeteld, (behalve af en toe een beetje bonus).