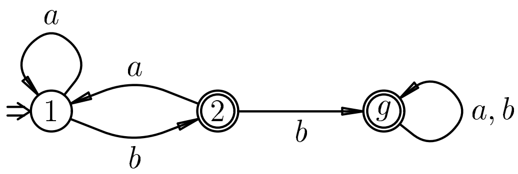
1) **a** $\mathcal{A} = (Q, \Delta, \delta, I, F)$, waarbij $Q = Q_1 \times Q_2$, $\Delta = \Delta_1 \cap \Delta_2$, $I = I_1 \times I_2$, $F = F_1 \times F_2$, en $((p_1, p_2), a, (q_1, q_2)) \in \delta$ desdals $(p_1, a, q_1) \in \delta_1$ en $(p_2, a, q_2) \in \delta_2$.

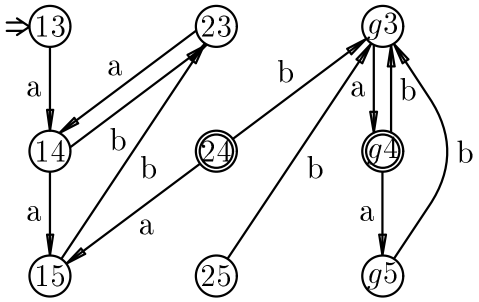
b $L(\mathcal{A}) \neq \emptyset$ desdals er een pad is van een begin- naar een eindtoestand. Bereikbare toestanden laten zich makkelijk berekenen.

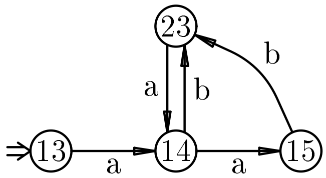
c In het algemeen geldt $B \subseteq A$ desdals $A^c \cap B = \emptyset$. Het gevraagde algoritme construeert dus een automaat voor het complement van $L(\mathcal{A}_1)$ (totaal deterministisch maken, eindtoestanden aanpassen) en construeert vervolgens de doorsnede met $\mathcal{A}_2$, test tenslotte de leegheid van deze automaat.

d Voeg een 'garbage'-toestand g aan $\mathcal{A}$ toe om de automaat totaal te maken, de automaat is al deterministisch. Complementeer $\mathcal{A}$.



De 'formele' doorsnede constructie levert $3 \cdot 3 = 9$ toestanden op (hieronder links). Een meer praktische aanpak begint bij de begintoestand en levert alleen de bereikbare toestanden op (rechts). Geen van de bereikbare toestanden is eindtoestand, de doorsnede is dus leeg, en daarmee is $L(\mathcal{B}) \subseteq L(\mathcal{A})$.





Een alternatieve methode, maar meer ad hoc, is het aangeven van een simulatie van $\mathcal{B}$ door $\mathcal{A}$. We geven aan hoe elk (accepterend) pad van $\mathcal{B}$ nagedaan kan worden met een (accepterend) pad van $\mathcal{A}$. Deze simulatie kunnen we weergeven als een eindige automaat, die er toevallig (?) uitziet als de rechter automaat hierboven.

e Nu wordt $(2, 3)$ een bereikbare eindtoestand. Geaccepteerde woorden, zoals ab, zijn element van $L(\mathcal{B}) - L(\mathcal{A})$, dus $L(\mathcal{B}) \not\subseteq L(\mathcal{A})$.

2) **b** Neem aan dat K context-vrij is. K is dus 'pompbaar', laat n de bijbehorende constante uit het pomplemma zijn.

Kies nu $z = a^n\$b^n\$a^nb^n$ (of $z = \$a^nb^n\$a^nb^n$). Omdat $z \in K$ en $|z| \geq n$ moeten we een opdeling $z = uvwxy$ kunnen vinden die aan de eisen van het pomplemma voldoet. Het pompgedeelte vx kan niet het symbool \$ bevatten omdat anders niet uv^0wx^0y tot K kan behoren (wegens een tekort van \$'s).

Elk woord in K bevat vanzelfsprekend gelijke aantallen a's voor en na de tweede \$, idem voor de b's. De enige manier om bij pompen aan deze eigenschap te blijven voldoen is om de segmenten a^n (en b^n) in gelijke aantallen aan de pompstukken vx mee te laten doen. Dit is in tegenspraak met de eis dat

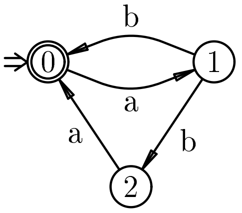
$|vwx| \leq n$: v en x kunnen niet meer dan n symbolen uit elkaar liggen, tussen de segmenten a^n (en idem b^n) liggen meer dan dit aantal symbolen.

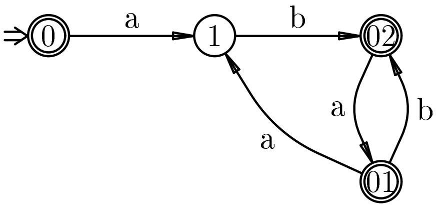
Uit deze tegenspraak volgt dat z niet binnen K te pompen is, daarmee voldoet K niet aan het pomplemma, en is K niet context-vrij.

3) **a** $S \to BB \quad S \to SP \quad P \to XA$
$A \to BQ \quad Q \to YR \quad R \to AZ \quad A \to b$
$B \to AY$
$X \to a \quad Y \to b \quad Z \to c$

b verwijder recursie voor S:
$S \to BB \quad S \to BBX \quad X \to aA \quad aAX$
doe niets bij A:
$A \to BbAc \quad A \to b$
substitueer A in B:
$B \to BbAcb \quad B \to bb$
en verwijder dan recursie voor B:
$B \to bb \quad B \to bbY \quad Y \to bAcb \quad Y \to bAcbY$
Uiteindelijk: substitueer omgekeerd B in S en in A:
$S \to bbB \quad S \to bbYB \quad S \to bbBX \quad S \to bbYBX$
$A \to bbbAc \quad A \to bbYbAc \quad A \to b$ we hadden nog:
$X \to aA \quad aAX$
$B \to bb \quad B \to bbY \quad Y \to bAcb \quad Y \to bAcbY$

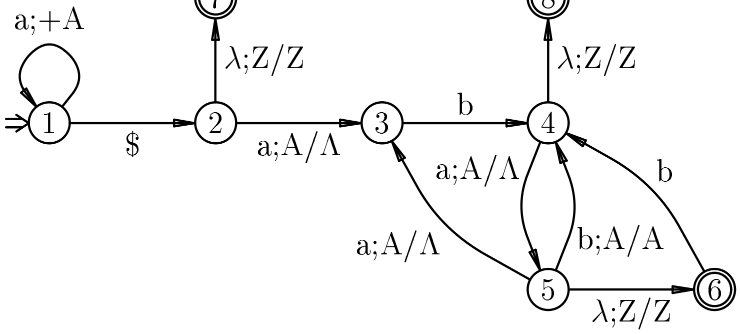
4) $L_4 = \{ab, aba\}^*$. *deterministische* stapelautomaat voor $a^*\%L_4$.
Eindige automaat voor L_4, en vervolgens deterministisch gemaakt:





Construeer daarmee een stapelautomaat voor $a^*\%L_4$. Initieel stapelsymbool Z, daarnaast een symbool A, dat gebruikt zal worden om a's te tellen.

De lastigheden doen zich voor bij toestand 5, we onderscheiden daar of de stapel nul A's bevat (we kunnen dan accepteren, maar óók verder met een b) of dat er nog A's opstaan (we móeten dan verder, hetzij met a hetzij met b).



5) $K_5 = \{ab, aab, aaab\}^*$, $L_5 = \{ab, aab\}^*$. Een woord uit $K_5\%L_5$ dat vaak over het hoofd gezien wordt is $abaabaaab\%aabaabaab$, de aantallen a's links en rechts kloppen, maar niet precies met de woordgrenzen. We geven twee mogelijke grammatica's voor $K_5\%L_5$.

i. We proberen zo evenwichtig mogelijk woorden uit de talen K_5 en L_5 te genereren, opgebroken in deelwoorden $ab, aab, aaab$.

Het verschil in aantallen a's wordt bijgehouden in een niet-terminaal: S, A, B, C, X representeren respectievelijk $0, 1, 2, 3, -1$. Bij S, X genereren we een stukje uit K_5 ('erbij'), bij A, B, C genereren we een stukje uit L_5 ('eraf').

$S \to \$$
$S \to abA \quad S \to aabB \quad S \to aaabC$
$A \to Sab \quad A \to Xaab$
$B \to Aab \quad B \to Saab$
$C \to Bab \quad C \to Aaab$
$X \to abS \quad X \to aabA \quad X \to aaabB$

ii. Een mogelijke rechts-lineaire grammatica voor K_5 heeft de variabelen S_i, $0 \leq i \leq 3$, axioma S_0, en de producties

$S_0 \to \Lambda$
$S_i \to aS_{i+1}, \quad i = 0, 1, 2$
$S_i \to bS_0, \quad i = 1, 2, 3$

Evenzo een *links*-lineaire grammatica voor L_5, om de taal 'achterstevoren' te genereren.

$S_0 \to \Lambda$
$S_j \to S_{j-1}a, \quad i = 1, 2$
$S_0 \to S_jb, \quad j = 1, 2$

We maken een grammatica voor $K_5\%L_5$ door de beide talen 'parallel' te genereren, b's onafhankelijk van elkaar (die worden niet geteld), a's uit beide talen synchroon (om gelijke aantallen te waarborgen). Vergelijk de 'product-constructie'.

Variabelen S_{ij}, $0 \leq i \leq 3$, $0 \leq j \leq 2$, axioma S_{00}, en producties

$S_{00} \to \$$
$S_{ij} \to bS_{0j}, \quad i = 1, 2, 3,\ j = 0, 1, 2$
$S_{i0} \to S_{ij}b, \quad i = 0, 1, 2, 3,\ j = 1, 2$
$S_{ij} \to aS_{i+1j-1}a, \quad i = 0, 1, 2,\ j = 1, 2$

Uitgeschreven zijn dit 24 producties.