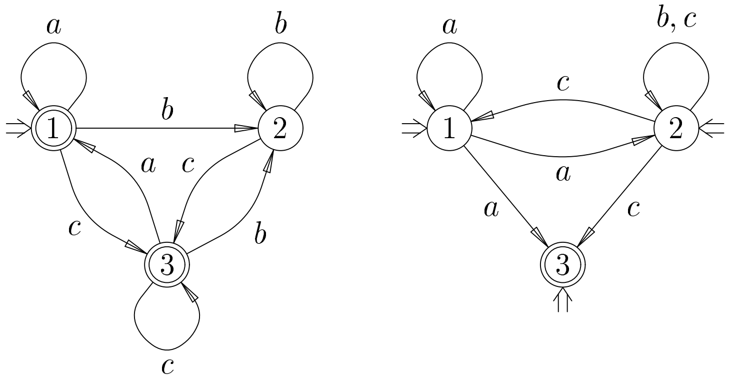
1) **a** Met behulp van de λe-methode of de $e\lambda$-methode worden respectievelijk de volgende automaten verkregen :



b Door goed naar de gegeven automaat te kijken is direct in te zien dat hij de taal $L(\mathcal{A}) = (a+b^*c)^*$ accepteert. Voor de zekerheid kun je de methode van Brzozowski en McCluskey toe passen op (bijvoorbeeld) de gegeven automaat (want dat is een gea) en dan krijg je, met verwijdervolgorde $\langle 2, 3, 1\rangle$ en met weglating van λ (zoveel mogelijk), de reguliere expressie $(a + b^*c)^*(\lambda + b^*c)$.

Als E een reguliere expressie is voor de taal $L(\mathcal{A})$, dan is uiteraard $E \cdot E$ een reguliere expressie voor de taal $L(\mathcal{A}) \cdot L(\mathcal{A})$. Dus, bijv., $(a + b^*c)^*(a + b^*c)^*$. (Ga dus niet eerst een automaat voor $L(\mathcal{A}) \cdot L(\mathcal{A})$ construeren en daarvan weer de reguliere expressie!)

c Een rechts-lineaire grammatica kan volgens het bewijs van Stelling 3.65 uit een eindige automaat (zonder λ-takken) geconstrueerd worden die één begintoestand heeft. Het makkelijkst is dus om de automaat te nemen die in **a** met de λe-methode verkregen is. Dan krijg je de rechts-lineaire grammatica $G = (\Sigma, \Delta, P, S)$ met $\Sigma = \{1, 2, 3, a, b, c\}$, $\Delta = \{a, b, c\}$, startsymbool $S = 1$ en producties $1 \to a1$, $1 \to b2$, $1 \to c3$, $1 \to \lambda$, $2 \to b2$, $2 \to c3$, $3 \to a1$, $3 \to b2$, $3 \to c3$, $3 \to \lambda$. Als je de automaat neemt die in **a** met de $e\lambda$-methode verkregen is, dan moet je eerst een nieuwe begintoestand (zeg S) nemen en de bijbehorende λ-takken verwijderen. Dan krijg je een rechts-lineaire grammatica met producties $S \to a1$, $S \to a2$, $S \to a3$, $S \to c1$, $S \to b2$, $S \to c2$, $S \to c3$, $S \to \lambda$, $1 \to a1$, $1 \to a2$, $1 \to a3$, $2 \to c1$, $2 \to b2$, $2 \to c2$, $2 \to c3$, $3 \to \lambda$.

2) **a** Intuitief moeten er twee posities i en j in de string zijn waartussen alleen b's staan (minstens één) en waarbuiten alleen a's staan (aan iedere kant minstens één). Dus, bijv.,

$$(\exists i, j)(i < j \wedge \neg\mathrm{first}(i)$$
$$\wedge(\forall k)((k < i \to a(k)) \wedge (i \leq k < j \to b(k)) \wedge (j \leq k \to a(k)))).$$

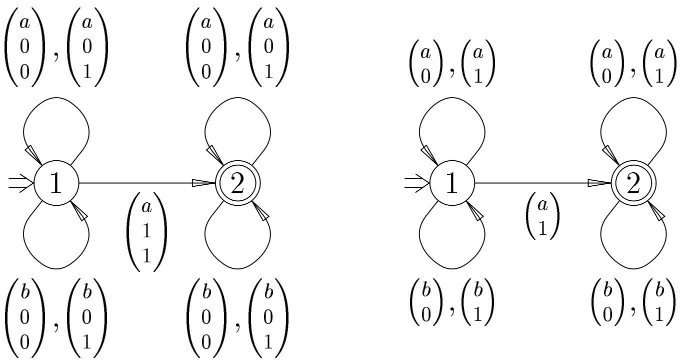
Hierin is i de positie van de eerste b en j de positie van de eerste a na de b's.

b Nu moet bovendien gezorgd worden dat het aantal b's tussen i en j even is. Neem dezelfde formule als boven, maar met een extra eis, als volgt: $(\exists i, j)(\cdots \wedge \phi(i, j))$ waarbij $\phi(i, j)$ de volgende formule is:

$(\exists X)(i \in X \wedge (\forall k, m)(\mathrm{next}(k, m) \to (k \in X \leftrightarrow m \notin X)) \wedge j \in X)$.

Intuitief is X de verzameling posities die op een even afstand van i liggen, vgl. Voorbeeld 3.49.

c,d De automaten voor de formules $i \in X \wedge a(i)$ en $(\exists i)(i \in X \wedge a(i))$ zijn, respectievelijk:



In de eerste automaat correspondeert de eerste bit met i en de tweede bit met X. De tweede automaat is uit de eerste verkregen door de i-bit overal te schrappen.

3) **a** Gebruik het standaard algoritme, met bijv. de volgorde S, A, B. Eerst substitutie voor $A \to SA$. Dat levert de producties $A \to AaBA \mid Aa \mid b$ voor A. Dan de links-recursie van A verwijderen, met behulp van een nieuwe niet-terminaal X. Dat levert de producties $A \to b \mid bX$ en $X \to aBAX \mid aX \mid aBA \mid a$. Daarna substitutie voor $S \to AaB$. Dat geeft de producties $S \to baB \mid bXaB$ voor S. En dan substitutie voor $B \to SB$, resulterend in $B \to baBB \mid bXaBB \mid Bb \mid ab$. Vervolgens de links-recursie van B verwijderen, met behulp van een nieuwe niet-terminaal Y. Dat levert $B \to baBB \mid bXaBB \mid ab \mid baBBY \mid bXaBBY \mid abY$ en $Y \to b \mid bY$. Alle rechterkanten van producties beginnen nu met een terminaal. We hoeven dus alleen nog de andere terminalen te vervangen door nieuwe niet-terminalen, bijv. a door E en b door D, met toevoeging van de producties $E \to a$ en $D \to b$. Uiteindelijk krijgen we zo de grammatica met producties

$S \to bEB \mid bXEB$,

$A \to b \mid bX$,

$X \to aBAX \mid aX \mid aBA \mid a$,

$B \to bEBB \mid bXEBB \mid aD \mid bEBBY \mid bXEBBY \mid aDY$,

$Y \to b \mid bY$,

$E \to a$, $D \to b$.

4) **a** Bijv. de cf grammatica met producties $S \to aSa$, $S \to bS$, $S \to Sb$, $S \to c$.

b Stel dat K_2 wél context-vrij is, en laat n de constante van het pomplemma zijn. Bekijk het woord $z = a^n b^n c a^n b^n$. Het is duidelijk dat $z \in K_2$ en dat $|z| = 4n + 1 > n$. Volgens het pomplemma zijn er dus woorden u, v, w, x, y zodanig dat $z = uvwxy$, $vx \neq \lambda$, $|vwx| \leq n$ en $uv^iwx^iy \in K_2$ voor alle $i \geq 0$. Het woord z bestaat uit vijf 'segmenten': a^n, b^n, c, a^n, b^n. Omdat $|vwx| \leq n$, is vwx een deelwoord van ófwel de eerste twee segmenten, ófwel de middelste drie segmenten, ófwel de laatste twee segmenten. We gaan deze drie gevallen na en bekijken steeds $z_0 = uwy$ (d.w.z. $i = 0$).

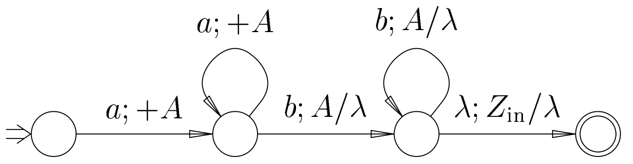
i. eerste twee segmenten: Dan is $z_0 = a^k b^l c a^n b^n$ met $k + l < 2n$. Dus ófwel $k < n$ en dan is in z_0 het aantal a's links van c niet gelijk aan het aantal a's rechts van c, ófwel $l < n$ en dan is in z_0 het aantal b's links van c niet gelijk aan het aantal b's rechts van c. In beide gevallen zit z_0 niet in K_2, in tegenspraak met het pomplemma.

ii. middelste drie segmenten: Als v of x een c bevat, dan bevat z_0 geen c. Als v en x geen c bevatten, dan is $z_0 = a^n b^k c a^l b^n$ met $k + l < 2n$. Als $k < n$, dan is in z_0 het aantal b's links van c niet gelijk aan het aantal b's rechts van c, en als $l < n$, dan is in z_0 het aantal a's links van c niet gelijk aan het aantal a's rechts van c. In alle drie de gevallen zit z_0 dus niet in K_2, weer in tegenspraak met het pomplemma.

iii. laatste twee segmenten: analoog aan geval i.

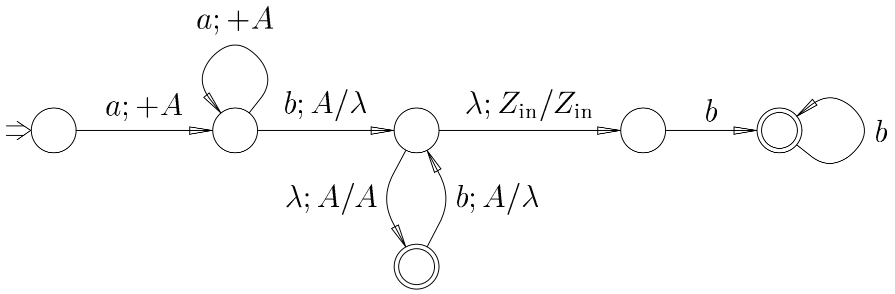
c Het complement van K_2 is $K_2^c = \{ucv \mid u, v \in \{a,b\}^*, \#_a(u) \neq \#_a(v) \text{ of } \#_b(u) \neq \#_b(v)\}$. Dus $K_2^c = K_1^c \cup h(K_1^c)$, waarbij h het homomorfisme is dat a en b verwisselt (d.w.z. $h(a) = b$ en $h(b) = a$). Omdat CF gesloten is onder vereniging (Stelling 4.51) en onder homomorfismen (Stelling 4.52), volstaat het nu dus te bewijzen dat K_1^c context-vrij is. Er geldt $K_1^c = R_1 \cup K_1 \cdot R_2 \cup R_2 \cdot K_1$ waarbij R_1 het complement is van de reguliere taal $(a+b)^*c(a+b)^*$ (en dus zelf ook regulier omdat de reguliere talen gesloten zijn onder complement: Stelling 3.20) en R_2 de reguliere taal $(a+b)^*a(a+b)^*$ is. Omdat (i) K_1 context-vrij is volgens **a**, (ii) alle reguliere talen context-vrij zijn (Stelling 4.7), en (iii) CF gesloten is onder vereniging en concatenatie (Stellingen 4.51 en 4.52), is K_1^c dus context-vrij.

5) **a** $K = L^c \cap a^*b^*$ waarbij $L = \{a^k b^k \mid k \geq 1\}$. De taal L zit in $\mathsf{DSA_f}$, omdat de volgende deterministische stapelautomaat L accepteert met eindtoestanden:



Omdat $\mathsf{DSA_f}$ gesloten is onder complement en onder doorsnijding met reguliere talen (Stelling 5.25), zit K dus ook in $\mathsf{DSA_f}$.

Een andere mogelijkheid is natuurlijk om een deterministische stapelautomaat te maken die K accepteert met eindtoestanden, bijv. de volgende:



b Stel dat K wél in REP zit. Omdat $L = K^c \cap a^*b^*$ (met de L uit **a**), en omdat REP gesloten is onder complement en doorsnede (Stelling 3.20), zit L dan ook in REP. Maar met behulp van het pomplemma is te bewijzen dat L niet in REP zit (zie Gevolg 3.19), een tegenspraak.

Het pomplemma is ook direct te gebruiken om aan te tonen dat K niet regulier is, maar dat is lastig. Neem $z = a^n b^{n+n!}$ (waarbij n de constante van het pomplemma is).

c K is niet prefix-vrij: bijv. abb en $abbb$ zitten allebei in K en abb is een prefix van $abbb$. Volgens Lemma 5.18 zit K dan niet in $\mathsf{DSA_e}$.