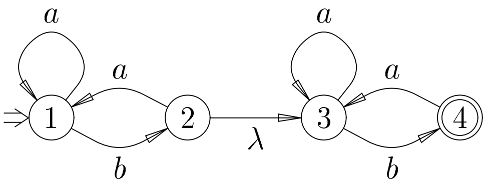
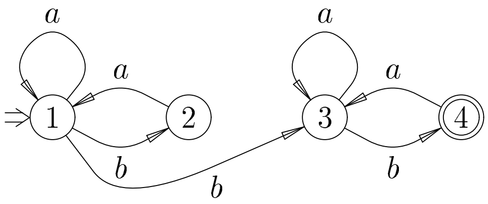
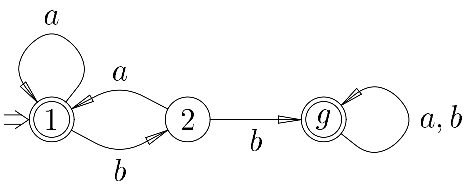
1) **a** $L(\mathcal{A}) \cdot L(\mathcal{A})$. Verbind twee exemplaren van de originele automaat door een λ-tak, zoals aangegeven. Verwijder dan de λ-tak, bijvoorbeeld met de $e\lambda$-methode.





Complement van $L(\mathcal{A})$. De automaat is deterministisch, totaal maken volstaat. Voeg dus een 'garbage'-toestand toe. Wissel dan eindtoestanden om met niet-eindtoestanden.



b Een reguliere expressie voor $L(\mathcal{A})$ is $(a+ba)^* \cdot b$, voor $L(\mathcal{A}) \cdot L(\mathcal{A})$ dus $(a+ba)^* \cdot b \cdot (a+ba)^* \cdot b$. Voor het complement van $L(\mathcal{A})$ vinden we eenvoudig $(a+ba)^* + (a+ba)^* \cdot bb \cdot (a+b)^*$

c We bereiken één verzamelingsvariabele door ons te realiseren dat de automaat slechts twee toestanden heeft. De toestand is of 1 (in verzameling X) of 2 (niet in verzameling X). Het is dan niet meer nodig om een partitie te eisen.

$$
\begin{array}{lll}
(\exists X)[& (\forall i)(\text{first}(i) \to (a(i) \wedge i \in X) \vee (b(i) \wedge i \notin X) & \text{eerste stap} \\
\wedge & (\forall i)(\forall j)(\text{next}(i,j) \to & \text{transitie} \\
 & \qquad\qquad (i \in X \wedge a(j) \wedge j \in X) \vee & \\
 & \qquad\qquad (i \in X \wedge b(j) \wedge j \notin X) \vee & \\
 & \qquad\qquad (i \notin X \wedge a(j) \wedge j \in X)\ ) & \\
\wedge & (\forall i)(\text{last}(i) \to i \notin X) & \text{acceptatie} \\
] & &
\end{array}
$$

We kunnen zelfs zonder verzamelingsvariabelen toe, door naar de taal zelf te kijken (en dus niet paden in de automaat te coderen, zoals de standaard methode doet). Een woord behoort tot de taal precies als het op een b eindigt en er geen twee opeenvolgende b's zijn:

$$(\exists i)[\ \text{last}(i) \wedge b(i)\ ] \ \wedge\ \neg(\exists i)(\exists j)[\ \text{next}(i,j) \wedge b(i) \wedge b(j)\ ]$$

De formule voor het complement van de taal vinden we door een formule voor de taal te ontkennen.

2) Maak eerst de grammatica λ-vrij. Duidelijk is A de enige verdwijnende niet-terminaal. Nieuwe producties:

$S \to AB \mid B, \quad S \to Sb, \quad A \to AbBa \mid bBa, \quad B \to bBc, \quad B \to bc.$

Werk bijvoorbeeld van achter naar voor, dus 1:B, 2:A en 3:S.

1: Rechterzijden producties voor B beginnen met een terminaal. Klaar.

2(sub): Er zijn geen producties $A \to B\alpha$.

2(rec): Verwijder recursie voor A. De A-producties worden:

$A \to bBa \mid bBaX, \quad$ met toegevoegd $X \to bBaX \mid bBa$

2(sub): Er zijn geen producties $B \to A\alpha$.

3(sub): Substitueer in de producties $S \to AB \mid B$.

$S \to bBaB \mid bBaXB \mid bBc \mid bc$, verder houden we $S \to Sb$.

3(rec): Verwijder links-recursie voor S:

$S \to bBaB \mid bBaXB \mid bBc \mid bc$, $S \to bBaBY \mid bBaXBY \mid bBcY \mid bcY$, verder $Y \to bY \mid b$.

3(sub): Er zijn geen producties $A \to S\alpha$ of $B \to S\alpha$.

Rest ons een aantal a's en c's aan de rechterzijde te vervangen door nieuwe niet-terminalen P en Q:

$S \to bBPB \mid bBPXB \mid bBQ \mid bQ \mid bBPBY \mid bBPXBY \mid bBQY \mid bQY$,
$Y \to bY \mid b$,
$A \to bBP \mid bBPX$,
$X \to bBPX \mid bBP$,
$B \to bBQ \mid bQ$,
$P \to a$, $Q \to c$.

Hierna kunnen de producties $A \to bBP \mid bBPX$ verwijderd worden, omdat A niet meer bereikbaar is.

3) **a** We moeten de b's en c's tegelijk genereren, en dan is er geen ruimte om tegelijkertijd ook bij te houden of $k + m \equiv 0 \bmod 3$. Dat lukt alleen als we de drie mogelijkheden voor $k, m \bmod 3$ uitsplitsen.

$S \to A_0B_0 \mid A_1B_2 \mid A_2B_1$,
$A_0 \to \lambda \mid aA_2$, $A_1 \to aA_0$, $A_2 \to aA_1$,
$B_0 \to \lambda \mid bB_2c$, $B_1 \to bB_0c$, $B_2 \to bB_1c$.

Er geldt bijvoorbeeld dat $A_k \Rightarrow^* a^\ell$, met $\ell \equiv k \bmod 3$.

aa De stelligheid waarmee hierboven gesuggereerd wordt dat de keuze voor $k+m \equiv 0 \bmod 3$ aan het begin gemaakt moet worden (met het kiezen van een produktie $S \to A_iB_j$) is misplaatst. We kunnen wel degelijk eerst de a's genereren en daarna de b's en c's tegelijk.

$S \to aaaS \mid T \mid abbTcc \mid aabTc$,
$T \to bbbTccc \mid \lambda$.

b Stel $K_2 \in \mathsf{CF}$. Dan is er een constante n als in het pomplemma voor cf-talen (Stelling 4.42 – volgens de nummering op moment van schrijven). Kies $z = a^{3n}b^{3n}c^{3n}$, dan $z \in K_2$ (met $k = m = 3n$, $k+m = 6n$ een 3-voud) en $|z| = 9n > n$. Nu moet er dus een opdeling $z = uvwxy$ zijn die op de juiste wijze gepompt kan worden.

Vanwege $|vwx| \leq n$ is vwx een deelwoord van $a^{3n}b^{3n}$ of van $b^{3n}c^{3n}$.

Bekijk nu deze twee mogelijkheden:

i. vwx is een deelwoord van $a^{3n}b^{3n}$. Nu zal uv^2wx^2y nog steeds $3n$ c's bevatten, maar meer a's of meer b's. Dus $uv^2wx^2y \notin K_2$ (want in K_2 is het aantal a's hoogstens het aantal c's en het aantal b's gelijk aan het aantal c's.)

ii. vwx is een deelwoord van $b^{3n}c^{3n}$. Nu zal $uwy = uv^0wx^0y$ nog steeds $3n$ a's bevatten, maar minder b's en/of c's. Dus $uv^0wx^0y \notin K_2$. (want in K_2 is het aantal b's niet minder dan het aantal a's en het aantal c's ook.)

Conclusie: in elk van beide gevallen wordt niet aan de eigenschap voldaan dat het gepompte woord in de taal moet zitten. Tegenspraak. K_2 is niet context-vrij.

4) **a** Vanwege de eis tot determinisme kunnen we de gevonden grammatica niet tot automaat omwerken, want de producties voor S leveren een niet-deterministische keuze. Een voorstel:

$(p_0, a, Z) \mapsto (p_1, Z)$, $(p_1, a, Z) \mapsto (p_2, Z)$, $(p_2, a, Z) \mapsto (p_0, Z)$, tel a's modulo 3.

$(p_0, b, Z) \mapsto (q_1, \Box Z)$, $(p_1, b, Z) \mapsto (q_2, \Box Z)$, $(p_2, b, Z) \mapsto (q_0, \Box Z)$,
bij b's stapelen, we blijven tellen.

$(q_0, b, \Box) \mapsto (q_1, \Box\Box)$, $(q_1, b, \Box) \mapsto (q_2, \Box\Box)$, $(q_2, b, \Box) \mapsto (q_0, \Box\Box)$,

$(q_0, c, \Box) \mapsto (r, \lambda)$, een c op het juiste moment

$(r, c, \Box) \mapsto (r, \lambda)$, blijf afstapelen bij c's

$(r, \lambda, Z) \mapsto (q_{\text{fin}}, \lambda)$, bodem bereikt, we mogen accepteren.

Begintoestand p_0. Eindtoestanden zijn p_0 (om te accepteren bij $m = 0$!) en q_{fin}. Stapelalfabet $\Gamma = \{Z, \Box\}$, initieel stapelsymbool $Z_{\text{in}} = Z$.

b $K_1 \notin \mathsf{DSA_e}$, want K_1 bevat λ en bijvoorbeeld a^3, en is daarmee niet prefix-vrij (Lemma 5.18).

K_1' zit wel in $\mathsf{DSA_e}$, de automaat hierboven gegeven voldoet: bij het bereiken van q_{fin} is de stapel leeggemaakt.

Alternatief: $K_1 \in \mathsf{DSA_f}$, en daarmee ook K_1' want die ontstaat uit K_1 door doorsnijding met een reguliere taal (Stelling 5.25). Merk op dat K_1' bovendien prefix-vrij is, en dus in $\mathsf{DSA_e}$ volgens de karakterisatie van Stelling 5.21.

5) Als R een reguliere taal is, dan kunnen we bepalen of $L(G) \cap R = \emptyset$. Dit volgt uit het feit dat CF (effectief) gesloten is onder doorsnijding met reguliere talen (Stelling 4.52). Daarom is $L(G) \cap R$ weer context-vrij, zodat we leegheid kunnen testen volgens Stelling 4.56.

Evenzo kunnen we testen of $L(G) \subseteq R$, dat is namelijk het geval als $L(G) \cap R^c = \emptyset$, waarbij R^c het complement van R is. Ook R^c is regulier, want $\mathsf{REP} = \mathsf{REG}$ is gesloten onder complement.

a Beslis of $L(G) \cap R_1 = \emptyset$, met R_1 de reguliere taal $a^*b^*a^*$.

b Beslis of $L(G) \subseteq R_1$, met R_1 als hiervoor.

c De taal $R_2 = \{a^k b^m a^n \mid 0 \leq k, m, n \leq 100\}$ is eindig en dus regulier (Stelling 3.23). Beslis of $L(G) \subseteq R_2$ als boven. Om ook $R_2 \subseteq L(G)$ te verifiëren gaan we voor elk element w van R_2 apart na of $w \in L(G)$, volgens Stelling 4.53. Dit lukt in dit geval omdat R_2 een eindige taal is!