Formele Talen en Automaten

Project

Bart Middag
3de bachelor informatica
Academiejaar 2013-2014

1. REGULIER OF NIET-REGULIER?

a) Niet-regulier:

Om te beginnen zien we dat de definitie van L nergens een volgorde vastlegt. Bijgevolg is er een symmetrie in deze taal: als xyz een woord is in deze taal, is zyx ook een bestaand woord. Bijgevolg geldt dat $L^R = L$ – we moeten dus enkel een bewijs of een tegenvoorbeeld geven voor de taal L.

Ook merken we op dat een 1 en een 2 bijna volledig gelijk behandeld worden in deze taal. Voor het volgende bewijs kunnen we deze twee symbolen dus samennemen in een abstract symbool b, waarbij we afspreken dat een opeenvolging van dit symbool, bijvoorbeeld b^5, ofwel 12121 ofwel 21212 kan zijn. Het symbool 0 noemen we in dit bewijs a. Met deze afspraken kunnen we deze taal als volgt schrijven: $L = \{w \in \{a,b\}^* : \#_b(w) > 2\#_a(w)\}$. We zien een sterk verband tussen het aantal a's en het aantal b's. We verwachten dus dat de taal niet-regulier zal zijn.

We stellen dat de taal wel regulier is. Enkel het verband tussen het aantal a's en het aantal b's zal het niet-regulier zijn opleveren. We kunnen de definitie van de taal dus nogmaals vereenvoudigen:

$$L' = L \cap a^*b^*$$

Dan is $L' = \{a^i b^j : j > 2i\}$. Aangezien we aangenomen hebben dat L regulier is, is L' ook regulier. We passen nu het pumping lemma toe op L'. Dit is duidelijk een oneindige taal.

Volgens het pumping lemma bestaat er een $k \geq 1$ zodanig dat voor iedere $w \in L'$ met $|w| \geq k$, er strings x,y,z bestaan zodat $w = xyz, |xy| \leq k, |y| > 0$ en $w_q = xy^q z \in L'$ voor iedere $q \in \mathbb{N}$.

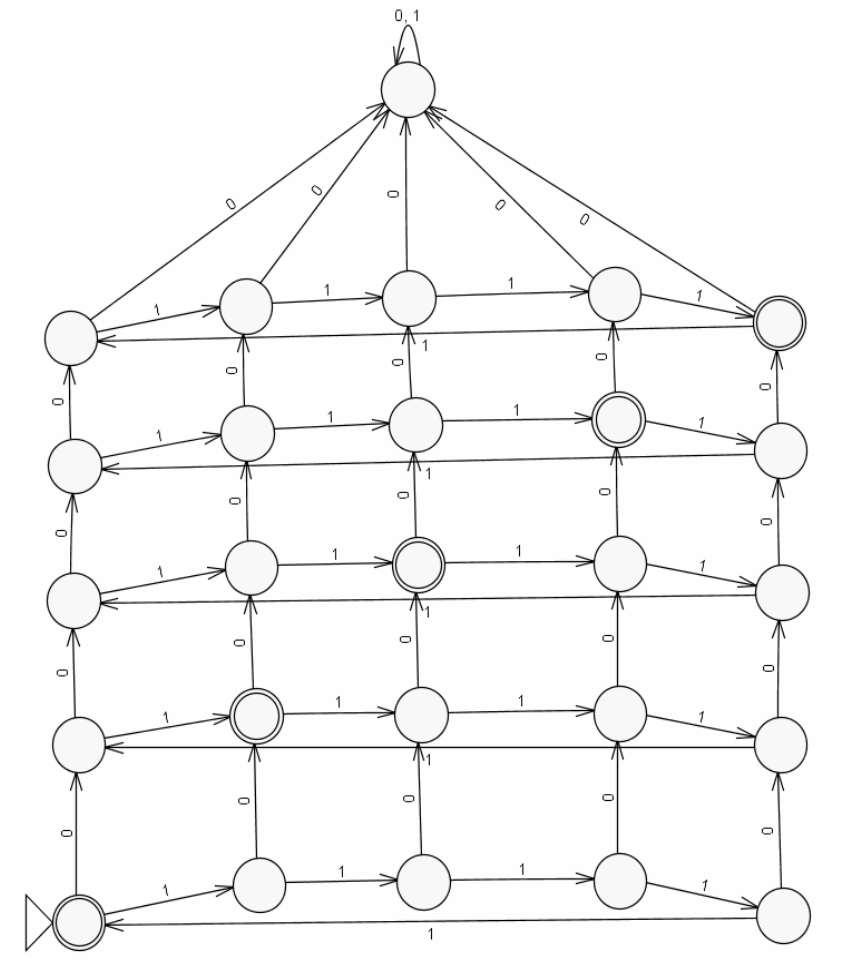
Definieer nu $w = a^k b^{2k+1} \in L'$. Dan is $|w| \geq k$, dus bestaan er strings x, y, z zodat $w = xyz, |xy| \leq k$ en $|y| > 0$. Omdat $|xy| \leq k$, komen x en y in de eerste k symbolen van w voor, dus $x = a^{|x|}$ en $y = a^{|y|}$ met $|y| > 0$. We noteren $|y|$ vanaf nu als p.

Er geldt dat $w_q = a^{|x|}(a^p)^q a^{k-p-|x|} b^{2k+1} \in L'$ voor iedere $q \in \mathbb{N}$. Als $q = 2$, dan is $w_2 = a^{k+p} b^{2k+1}$. w_2 is echter geen element van L', aangezien $2k + 1 \not> 2(\mathrm{k} + \mathrm{p})$ $(p > 0)$. Het pumping lemma geeft dus een tegenstrijdigheid.

L' en L zijn dus niet-regulier.

b) Regulier:

Deze taal is regulier – ze kan immers voorgesteld worden door de finite-state machine op Figuur 1. Bij invoer van 1 overloopt de machine een horizontale rij van states. De enige accepting state op deze rij is die die hoort bij het aantal nullen. Telkens als er een 0 wordt ingevoerd, zal de machine naar een hogere rij springen. Zodra er meer dan 4 nullen zijn ingegeven, is de invoer ongeldig en zal deze nooit meer geaccepteerd worden.



Figuur 1: Automaat van opgave 1b. In elke toestand links is $\#_1(w)\%5 = 0$. Rechts is $\#_1(w)\%5 = 4$. Onderaan is $\#_2(w)\%5 = 0$ – helemaal bovenaan is dit 5. Dit is niet mogelijk in de taal, dus deze toestand zal nooit geaccepteerd worden.

2. CONTEXTVRIJ OF NIET-CONTEXTVRIJ?

a) Contextvrij:

Er zijn maximaal 6 a's, het aantal c's is maximaal het aantal a's en de aantallen van b en d zijn samen gelijk aan het aantal van c. Het is duidelijk dat dit een eindige taal is. We weten dat alle eindige talen regulier zijn, en dat alle reguliere talen contextvrij zijn. Deze taal is dus contextvrij.

b) Niet-contextvrij:

We zien gemakkelijk dat opdat deze definitie zou gelden, $\#_b(w)$ een veelvoud moet zijn

van $\#_c(w)$. Als $\#_c(w) = 0$, mag a onbeperkt voorkomen. Als echter $\#_c(w) \neq 0$ moet $\#_a(w) = \frac{\#_c(w)+1}{\#_c(w)}\#_b(w)$. Door dit sterk en complex verband tussen a, b en c verwachten we dat deze taal niet-contextvrij zal zijn. We zullen dit aantonen door middel van het pumping lemma.

Stel dat de taal L wel contextvrij is. We leggen eerst een volgorde op de letters in onze woorden. Definieer de taal L' als $L' = L \cap c^*cb^*a^*$. We kunnen dit ook als volgt schrijven:

$$L' = \{c^m b^n a^{\frac{m+1}{m}n}, n \geq 0 \text{ en } n \text{ is deelbaar door } m \text{ met } m > 0\}$$

Aangezien we aangenomen hebben dat L contextvrij is, is L' ook contextvrij. We passen dus het pumping lemma toe op de taal L'. Het is duidelijk dat deze ook een oneindige taal is.

Volgens het pumping lemma bestaat er een $k \geq 1$ zodanig dat voor iedere $w \in L'$ met $|w| \geq k$, er strings u, v, x, y, z bestaan zodanig dat

$$w = uvxyz, |vxy| \leq k, |vy| > 0$$

en $w_q = uv^q xy^q z \in L'$ voor iedere $q \in \mathbb{N}$.

Definieer nu $w = c^k b^k a^{\frac{k+1}{k}k} \in L'$. Dan is $|w| \geq k$. Dus bestaan er strings u, v, x, y, z zodat $w = uvxyz, |vxy| \leq k, |vy| > 0$. Omdat $|vxy| \leq k$, betekent dit dat v en y hoogstens k symbolen van elkaar liggen. We zien dat w uit drie regio's bestaat:

$$w = cc\ b..b\ a\ ....a$$
$$|1|\ |\ 2\ |\ |\ \ 3\ \ |$$

Er zijn nu verschillende mogelijkheden in welke regio's v en y liggen.

- v is niet-leeg en ligt in meer dan één regio. Indien we v oppompen, zit de resulterende string niet meer in L': de volgorde van de letters is niet meer correct.
- y is niet-leeg en ligt in meer dan één regio. Indien we y oppompen, zit de resulterende string niet meer in L': de volgorde van de letters is niet meer correct.

We mogen aannemen dat zowel v en y in één regio liggen (niet noodzakelijk dezelfde), maar het is nog mogelijk dat v of y leeg is (als één van de twee leeg is, dan zeggen we dat het in dezelfde regio ligt als de andere string).

- (1,1) Zowel v als y liggen in regio 1. Één van de twee strings kan leeg zijn, maar de string vy is niet leeg. Dit betekent dat minstens één van de k c's in de regio de string v of y is. We kunnen dus c's bijpompen of c's wegpompen, maar in ieder geval blijft de verhouding tussen het aantal a's en het aantal b's hetzelfde, terwijl deze zou moeten veranderen naargelang het aantal c's. Bijvoorbeeld: $w_2 \notin L'$.
- (2,2) Zowel v als y liggen in regio 2. Één van de twee strings kan leeg zijn, maar de string vy is niet leeg. We kunnen nu b's bijpompen of wegpompen zodat $\#_a(w) \neq \frac{k+1}{k}\#_b(w)$. Bijvoorbeeld: $w_2 \notin L'$.
- (3,3) Zowel v als y liggen in regio 2. Één van de twee strings kan leeg zijn, maar de string vy is niet leeg. We kunnen nu a's bijpompen of wegpompen zodat $\#_a(w) \neq \frac{k+1}{k}\#_b(w)$. Bijvoorbeeld: $w_2 \notin L'$.

- (1,2) v ligt in regio 1 en y ligt in regio 2. Beide strings zijn niet leeg. Door bij te pompen kunnen we het aantal b's groter maken dan het aantal a's, wat duidelijk niet geldig is. Bijvoorbeeld: $w_5 \notin L'$.
- (2,3) v ligt in regio 2 en y ligt in regio 3. Beide strings zijn niet leeg. Door bij of weg te pompen zal de verhouding tussen het aantal b's en het aantal a's niet meer gelijk zijn aan $\frac{k+1}{k}$ (met k het aantal c's).
  Als $\#_c(w) = m < \frac{k}{2}$, dan zou er een keuze van strings zijn waardoor de verhouding toch gelijk zou blijven: de string v met lengte m aan het einde van regio 2 en de string y met lengte $m + 1$ aan het begin van regio 3. Omdat $|vxy| \leq k$ is dit echter geen geldige keuze als $m > \frac{k}{2}$, zoals in het gekozen woord w.
  Dus: $w_2 \notin L'$.
- (1,3) v ligt in regio 1 en y ligt in regio 3. Beide strings zijn niet leeg. Dit is niet mogelijk, omdat er k b's zijn. Als beide strings niet leeg zouden zijn, is dit tegenstrijdig met $|vxy| \leq k$.

In elk geval vinden we een tegenstrijdigheid: het pumping lemma is dus niet geldig voor L'. L' is dus geen contextvrije taal.
L' was echter de doorsnede van L en een reguliere taal – als L dus contextvrij zou zijn, zou L' dit normaal ook zijn. We kunnen dus besluiten dat L een niet-contextvrije taal is.

3. OPSTELLEN EN BEWIJZEN VAN EEN CONTEXTVRIJE GRAMMATICA

Vereenvoudiging van de opgave:

- Er mogen om het even hoeveel 2'en staan in elk woord, en op om het even de welke plaats.
- $\#_0(w) = 1 + \frac{\#_1(w)}{2}$. Dat betekent dat $\#_1(w)$ een even getal moet zijn.

Definitie:

We definiëren de grammatica $G = (\{S, A, B\}, \{0, 1, 2\}, R, S)$, met de productieregels

$$S \to BA0B$$
$$A \to 1B1BA0B|\varepsilon$$
$$B \to 2B|\varepsilon$$

Bewijs:

We zullen nu bewijzen dat $L(G) = L$, dus dat de grammatica correct is.

- Is $\boldsymbol{L(G) \subseteq L}$?
  Stel $S \Rightarrow_G^n z$. Met inductie naar n bewijzen we $f(z)$, wat we als volgt definiëren:
  $$f(z) = "z \in \{0,1,2\}^* \text{ en } \#_0(z) = 1 + \frac{\#_1(z)}{2} \text{ en alle 1'en komen voor alle 0'en}"$$
  Indien $n = 0$, dan is $z = S$. Voor dit geval is $f(z)$ waar.
  Indien $n > 0$, is $n = m + 1$. Stel $S \Rightarrow_G^m z' \Rightarrow_G z$. Noem de laatste regel die we toepassen om z te bekomen R. Uit de inductiehypothese volgt dat $f(z')$ geldt.
  Er zijn nu verschillende mogelijkheden voor de laatste regel R:

- De laatste regel $\boldsymbol{R}$ is gelijk aan $\boldsymbol{S \to BA0B}$:
  Men kan eenvoudig controleren dat $f(z)$ geldig is: S wordt enkel gebruikt als startsymbool. Als dit de laatste regel is, betekent dat dat het woord z enkel zal bestaan uit $BA0B$ – dit bevat één 0 en geen enkele 1, en voldoet zo aan $f(z)$.

- De laatste regel $\boldsymbol{R}$ is gelijk aan $\boldsymbol{A \to 1B1BA0B}$:
  In het woord z' wordt er ergens één A vervangen door $1B1BA0B$. Er komen dus twee 1'en en één 0 bij. We rekenen uit:
  $$\#_0(z') = 1 + \frac{\#_1(z')}{2}$$
  $$\Leftrightarrow \#_0(z') + 1 = 1 + \frac{\#_1(z') + 2}{2}$$
  $$\Leftrightarrow \#_0(z) = 1 + \frac{\#_1(z)}{2}$$
  Het aantal 0'en en 1'en is dus nog steeds geldig. Ook is het duidelijk dat binnen $1B1BA0B$ alle 1'en voor alle 0'en komen. Het symbool A wordt gebruikt in twee productieregels: $S \to BA0B$ en $A \to 1B1BA0B$. Als de productieregel die het woord z' genereerde $S \to BA0B$ was, is het duidelijk dat alle 1'en voor alle 0'en blijven – dan is z immers van de vorm $B1B1BA0B0B$. Als de productieregel die het woord z' genereerde $A \to 1B1BA0B$ was, blijven ook alle 1'en voor alle 0'en – als we de regel $A \to 1B1BA0B$ immers substitueren in $A \to 1B1BA0B$, krijgen we een regel van de vorm $A \to 1B1B1BA0B0B$, en binnen $1B1B1BA0B0B$ komen ook alle 1'en voor alle 0'en. $f(z)$ is dus waar.

- De laatste regel $\boldsymbol{R}$ is gelijk aan $\boldsymbol{B \to 2B}$:
  Dit verandert niets aan de positie of aantallen van bestaande 0'en of 1'en, dus zolang $f(z')$ waar was, is $f(z)$ waar.

- De laatste regel $\boldsymbol{R}$ is gelijk aan $\boldsymbol{A \to \varepsilon}$ of $\boldsymbol{B \to \varepsilon}$:
  Dit verandert niets aan de positie of aantallen van bestaande 0'en of 1'en, dus zolang $f(z')$ waar was, is $f(z)$ waar.

Stel nu dat $z \in L(G)$. Dan is $S \Rightarrow_G^n z$ en bevat z enkel terminalen. Omdat $f(z)$ waar is en omdat z enkel terminalen bevat, voldoet z aan de voorwaarden voor de taal L. We besluiten dus dat $z \in L$. We hebben dus net aangetoond dat $L(G) \subseteq L$.

- Is $\boldsymbol{L \subseteq L(G)}$?
  We stellen eerst en vooral een concretere definitie op van L:
  $$L = \{w : w = 2^*(12^*12^*)^n(02^*)^n02^*\}$$
  Met deze definitie van w hebben we meteen alle mogelijkheden van posities en aantallen van 2's in w – er liggen immers geen beperkingen op het gebruik van 2's (we spreken af dat een 2^* in een groep tussen haakjes die herhaald wordt in verschillende aantallen kan voorkomen in de herhalingen van die groep). Als we de relatief onbelangrijke 2'en even weglaten om een meer leesbare formule te krijgen, zien we het volgende:

$$w' = (11)^n(0)^n0$$

Deze formule legt dus enkel vast dat alle 1'en voor alle 0'en moeten komen, en dat $\#_0(w) = 1 + \frac{\#_1(w)}{2}$. De formule is dus correct.

$$w = 2^*(12^*12^*)^n(02^*)^n02^*$$

We zien dat in deze formule alle 2'en voorkomen in het formaat 2^*: nul of meer 2's. We vervangen alle 2^*'s in deze formule eerst door een symbool dat ook nul of meer 2's genereert:

$$w = B(1B1B)^n(0B)^n0B$$
$$\text{met } B \to 2B|\varepsilon$$

Voor de duidelijkheid van het volgende deel van het bewijs duiden we 2 regio's aan op in de formule:

$$w = B(1B1B)^n(0B)^n0B$$
$$\quad | \quad 1 \quad || \quad 2 \quad |$$

Vervolgens zien we dat als we n groter laten worden, er telkens $1B1B$ zal bijkomen na regio 1 en $0B$ vóór regio 2. We vervangen regio 1 en regio 2 dus samen door het symbool A dat deze regio's genereert.

$$w = BA0B$$
$$\text{met } A \to 1B1BA0B|\varepsilon$$
$$\text{met } B \to 2B|\varepsilon$$

Als we nu w laten genereren door het symbool S, is dit exact onze grammatica G! Dit betekent dat $w \in L(G)$. Dus, $L \subseteq L(G)$.

We hebben bewezen dat $L(G) \subseteq L$ en $L \subseteq L(G)$, dus $L(G) = L$. De grammatica is dus correct.

4. GESLOTEN OF NIET GESLOTEN?

a) Niet gesloten:

De reguliere talen zijn niet gesloten onder $f(L)$. Als we bijvoorbeeld de reguliere taal $L = \{w \in \{a,b\}^*\}$ beschouwen, zien we dat

$$f(L) = \{w \in \{a,b\}^* : w \text{ kan opgedeeld worden in 2 strings } x = x_1x_2 \dots x_n$$
$$\text{en } y = y_1y_2 \dots y_{2n}, \text{ en } w = x_1y_1x_2y_2 \dots x_ny_nx_1y_{n+1}x_2y_{n+2} \dots x_ny_{2n}\}$$

We kunnen elk woord dan als volgt in 2 delen van lengte $2n$ opdelen:

$$w = pq \text{ met } p = x_1y_1x_2y_2 \dots x_ny_n \text{ en } q = x_1y_{n+1}x_2y_{n+2} \dots x_ny_{2n}$$

Het is duidelijk dat we om q op te stellen, kennis nodig hebben van de lengte van x en van alle tekens van x, die we willekeurig hebben gekozen om p op te stellen.
Er is dus geheugen nodig, en omdat n onbepaald is, kan men dit probleem ook niet omzeilen door alle gevallen apart voor te stellen in één grote finite-state machine: $f(L)$ is dus niet-regulier. Met andere woorden, de reguliere talen zijn niet gesloten onder $f(L)$.

b) Gesloten:

De reguliere talen zijn gesloten onder $g(L)$. Dit zullen we bewijzen aan de hand van een algoritme dat een eindige deterministische herkenner (edh) voor L omzet in een edh voor $g(L)$.
Gegeven een edh $M = (Q, \Sigma, \delta, q_0, F)$ voor L.

We definiëren een *garbage state* als een toestand die foute input opvangt. Deze toestand heeft voor elke input een transitie naar zichzelf, en geen enkele transitie naar andere toestanden.

Voor alle $q \in F$ (dit zijn alle accepting states):
  Voor alle $\delta(q, \alpha)$ met $\alpha \in \Sigma$ (dit zijn alle transities die vertrekken uit q):
    Bestaat er een *garbage state*? Zoja, noemen we deze g, anders maken we een nieuwe *garbage state* aan die we g noemen.
    Verander het doel van de transitie $\delta(q, \alpha)$ naar g.
  Verwijder alle onbereikbare toestanden uit Q en uit F, en alle transities die eruit vertrekken. (Hier zijn genoeg gekende algoritmes voor, deze stap zullen we niet in detail beschrijven)

We bekomen nu een edh M' voor $g(L)$. Als dit algoritme correct is, is $g(L)$ dus ook een reguliere taal, en zijn de reguliere talen gesloten onder $g(L)$. We bewijzen dus nog dat het algoritme correct is.

$g(L)$ is de verzameling van alle woorden van L waarvan er geen echte prefixen zijn die zelf ook in L zitten. Stel $w, x \in L$ en x is een echt prefix van w. Om w te verwerken, moet de edh van L dan eerst de volledige input van x verwerken voor het de rest van w kan verwerken. Omdat $x \in L$, weten we dat de edh na het verwerken van x in een accepting state terechtgekomen is, waarna het *verdergaat* om de resterende input van w te verwerken. Er is dus een transitie van een accepting state naar een andere toestand, waarna er uiteindelijk opnieuw een accepting state bereikt wordt.
Omgekeerd, als er een transitie van een accepting state q_1 naar een andere toestand is, en als er via die transitie uiteindelijk opnieuw een accepting state q_2 bereikt kan worden, zal het woord gevormd door de input te verwerken tot aan q_1 een prefix zijn van het woord gevormd door de input te verwerken tot aan q_2.

Van een woord w in L zijn er geen echte prefixen die zelf ook in L zitten als en slechts als bij het verwerken van w, de edh van L nooit van een accepting state naar een andere toestand gegaan is. Met andere woorden, $g(L)$ is L zonder alle woorden waarvoor de edh van L van een accepting state naar een andere toestand moet gaan om ze te verwerken. Het zijn enkel deze woorden die het algoritme uitsluit: het algoritme verandert de “doeltoestand” van alle transities die uit alle accepting states vertrekken naar een *garbage state* die nooit geaccepteerd zal worden. Hierna worden onbereikbare toestanden verwijderd, omdat deze geen rol meer kunnen spelen in het verwerken van woorden van de taal $g(L)$.

Het is duideijk dat $g(L) = L(M')$. Het algoritme is correct. De reguliere talen zijn dus gesloten onder $g(L)$.

5. MINIMALE EINDIGE DETERMINISTISCHE HERKENNER

Niet-deterministische herkenner:

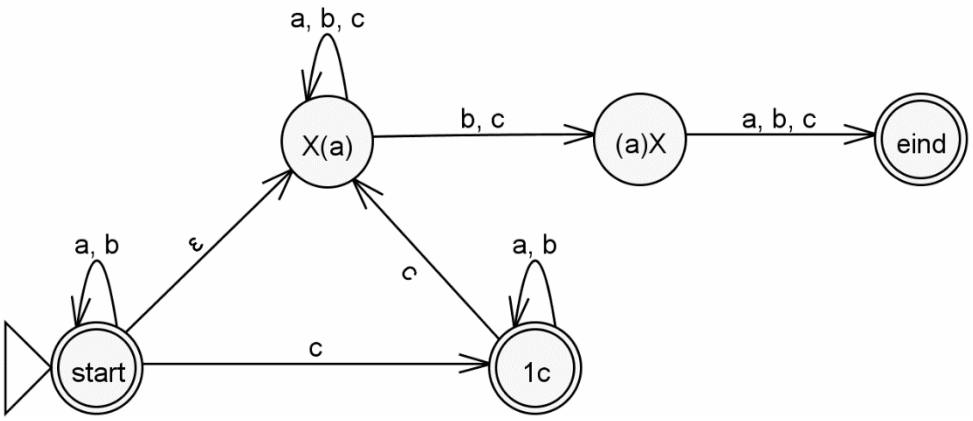
We zetten de formule die de taal L definieert eerst om naar een andere vorm om onze redenering duidelijk te maken:

$$\text{de voorlaatste letter van } w \text{ is een } a \implies \#_c(w) \leq 1$$
$$\equiv \#_c(w) > 1 \implies \text{de voorlaatste letter van } w \text{ is geen } a$$

Dit maakt het gemakkelijker om een herkenner op te stellen.

We stellen eerst een niet-deterministische herkenner op. Het resultaat zien we op Figuur 2. Vanaf er 2 c's zijn, moet deze een pad van transities volgen waarbij de voorlaatste letter geen a kan zijn. Dit pad kan ook direct gevolgd worden. We beschrijven de redenering die gebruikt werd om de niet-deterministische herkenner op te stellen:

- start: Er is nog geen enkele c aanwezig. De input is correct.
- 1c: In deze toestand is er zeker nog maar 1 c aanwezig. De input is correct.
- X(a): Er kan meer dan 1 c voorkomen, mogelijk zelfs gevolgd door een a.
- (a)X: Als er een a voorkwam in X(a) is deze nu de voorlaatste letter van de input.
- eind: De voorlaatste letter is nu zeker geen a.



Figuur 2: Niet-deterministische herkenner voor L

Omzetting naar een deterministische herkenner:

We passen het geziene algoritme toe om deze ndh om te zetten naar een edh.

We bepalen eerst $eps(q)$ voor iedere toestand q:

$$eps(start) = \{start, X(a)\}$$
$$eps(1c) = \{1c\}$$
$$eps\big(X(a)\big) = \{X(a)\}$$
$$eps\big((a)X\big) = \{(a)X\}$$
$$eps(eind) = \{eind\}$$

Definieer $s \coloneqq eps(start) = \{start, X(a)\}$.

We construeren nu δ.

- $\delta(s, a) = ?$
  $start$ wordt onder a afgebeeld op $start$
  $X(a)$ wordt onder a afgebeeld op $X(a)$
  Dus, $\delta(s, a) \coloneqq s$.
- $\delta(s, b) = ?$
  $start$ wordt onder b afgebeeld op $start$
  $X(a)$ wordt onder b afgebeeld op $X(a), (a)X$
  Definieer $t_0 \coloneqq eps(start) \cup eps\big(X(a)\big) \cup eps\big((a)X\big) = \{start, X(a), (a)X\}$ en $\delta(s', b) \coloneqq t_0$.
- $\delta(s, c) = ?$
  $start$ wordt onder c afgebeeld op $1c$
  $X(a)$ wordt onder c afgebeeld op $X(a), (a)X$
  Definieer $t_1 \coloneqq eps(1c) \cup eps\big(X(a)\big) \cup eps\big((a)X\big) = \{1c, X(a), (a)X\}$ en $\delta(s, c) \coloneqq t_1$.

- $\delta(t_0, a) = ?$
  $start$ wordt onder a afgebeeld op $start$
  $X(a)$ wordt onder a afgebeeld op $X(a)$
  $(a)X$ wordt onder a afgebeeld op $eind$
  Definieer $t_2 \coloneqq \{start, X(a), eind\}$ en $\delta(t_0, a) \coloneqq t_2$.
- $\delta(t_0, b) = ?$
  $start$ wordt onder b afgebeeld op $start$
  $X(a)$ wordt onder b afgebeeld op $X(a), (a)X$
  $(a)X$ wordt onder b afgebeeld op $eind$
  Definieer $t_3 \coloneqq \{start, X(a), (a)X, eind\}$ en $\delta(t_0, b) \coloneqq t_3$.
- $\delta(t_0, c) = ?$
  $start$ wordt onder c afgebeeld op $1c$
  $X(a)$ wordt onder c afgebeeld op $X(a), (a)X$
  $(a)X$ wordt onder c afgebeeld op $eind$
  Definieer $t_4 \coloneqq \{1c, X(a), (a)X, eind\}$ en $\delta(t_0, c) \coloneqq t_4$.

- $\delta(t_1, a) = ?$
  $1c$ wordt onder a afgebeeld op $1c$
  $X(a)$ wordt onder a afgebeeld op $X(a)$
  $(a)X$ wordt onder a afgebeeld op $eind$
  Definieer $t_5 \coloneqq \{1c, X(a), eind\}$ en $\delta(t_1, a) \coloneqq t_5$.
- $\delta(t_1, b) = ?$
  $1c$ wordt onder b afgebeeld op $1c$
  $X(a)$ wordt onder b afgebeeld op $X(a), (a)X$
  $(a)X$ wordt onder b afgebeeld op $eind$
  Definieer $\delta(t_1, b) \coloneqq t_4$.
- $\delta(t_1, c) = ?$
  $1c$ wordt onder c afgebeeld op $X(a)$
  $X(a)$ wordt onder c afgebeeld op $X(a), (a)X$
  $(a)X$ wordt onder c afgebeeld op $eind$

Definieer $t_6 \coloneqq \{X(a), (a)X, eind\}$ en $\delta(t_1, c) \coloneqq t_6$.

- $\delta(t_2, a) = ?$
  $start$ wordt onder a afgebeeld op $start$
  $X(a)$ wordt onder a afgebeeld op $X(a)$
  $eind$ wordt onder a afgebeeld op niets
  Dus, $\delta(t_2, a) \coloneqq s$.
- $\delta(t_2, b) = ?$
  $start$ wordt onder b afgebeeld op $start$
  $X(a)$ wordt onder b afgebeeld op $X(a), (a)X$
  $eind$ wordt onder b afgebeeld op niets
  Dus, $\delta(t_2, b) \coloneqq t_0$.
- $\delta(t_2, c) = ?$
  $start$ wordt onder c afgebeeld op $1c$
  $X(a)$ wordt onder c afgebeeld op $X(a), (a)X$
  $eind$ wordt onder c afgebeeld op niets
  Dus, $\delta(t_2, c) \coloneqq t_1$.

- $\delta(t_3, a) = ?$
  $start$ wordt onder a afgebeeld op $start$
  $X(a)$ wordt onder a afgebeeld op $X(a)$
  $(a)X$ wordt onder a afgebeeld op $eind$
  $eind$ wordt onder a afgebeeld op niets
  Dus, $\delta(t_3, a) \coloneqq t_2$.
- $\delta(t_3, b) = ?$
  $start$ wordt onder b afgebeeld op $start$
  $X(a)$ wordt onder b afgebeeld op $X(a), (a)X$
  $(a)X$ wordt onder b afgebeeld op $eind$
  $eind$ wordt onder b afgebeeld op niets
  Dus, $\delta(t_3, b) \coloneqq t_3$.
- $\delta(t_3, c) = ?$
  $start$ wordt onder c afgebeeld op $1c$
  $X(a)$ wordt onder c afgebeeld op $X(a), (a)X$
  $(a)X$ wordt onder c afgebeeld op $eind$
  $eind$ wordt onder c afgebeeld op niets
  Dus, $\delta(t_3, c) \coloneqq t_4$.

- $\delta(t_4, a) = ?$
  $1c$ wordt onder a afgebeeld op $1c$
  $X(a)$ wordt onder a afgebeeld op $X(a)$
  $(a)X$ wordt onder a afgebeeld op $eind$
  $eind$ wordt onder a afgebeeld op niets
  Dus, $\delta(t_4, a) \coloneqq t_5$.
- $\delta(t_4, b) = ?$
  $1c$ wordt onder b afgebeeld op $1c$

$X(a)$ wordt onder b afgebeeld op $X(a), (a)X$
$(a)X$ wordt onder b afgebeeld op $eind$
$eind$ wordt onder b afgebeeld op niets
Dus, $\delta(t_4, b) \coloneqq t_4$.

- $\delta(t_4, c) = ?$
  $1c$ wordt onder c afgebeeld op $X(a)$
  $X(a)$ wordt onder c afgebeeld op $X(a), (a)X$
  $(a)X$ wordt onder c afgebeeld op $eind$
  $eind$ wordt onder c afgebeeld op niets
  Dus, $\delta(t_4, c) \coloneqq t_6$.

- $\delta(t_5, a) = ?$
  $1c$ wordt onder a afgebeeld op $1c$
  $X(a)$ wordt onder a afgebeeld op $X(a)$
  $eind$ wordt onder a afgebeeld op niets
  Definieer $t_7 \coloneqq \{1c, X(a)\}$ en $\delta(t_5, a) \coloneqq t_7$.
- $\delta(t_5, b) = ?$
  $1c$ wordt onder b afgebeeld op $1c$
  $X(a)$ wordt onder b afgebeeld op $X(a), (a)X$
  $eind$ wordt onder b afgebeeld op niets
  Dus, $\delta(t_5, b) \coloneqq t_1$.
- $\delta(t_5, c) = ?$
  $1c$ wordt onder c afgebeeld op $X(a)$
  $X(a)$ wordt onder c afgebeeld op $X(a), (a)X$
  $eind$ wordt onder c afgebeeld op niets
  Definieer $t_8 \coloneqq \{X(a), (a)X\}$ en $\delta(t_5, c) \coloneqq t_8$.

- $\delta(t_6, a) = ?$
  $X(a)$ wordt onder a afgebeeld op $X(a)$
  $(a)X$ wordt onder a afgebeeld op $eind$
  $eind$ wordt onder a afgebeeld op niets
  Definieer $t_9 \coloneqq \{X(a), eind\}$ en $\delta(t_6, a) \coloneqq t_9$.
- $\delta(t_6, b) = ?$
  $X(a)$ wordt onder b afgebeeld op $X(a), (a)X$
  $(a)X$ wordt onder b afgebeeld op $eind$
  $eind$ wordt onder b afgebeeld op niets
  Dus, $\delta(t_6, b) \coloneqq t_6$.
- $\delta(t_6, c) = ?$
  $X(a)$ wordt onder c afgebeeld op $X(a), (a)X$
  $(a)X$ wordt onder c afgebeeld op $eind$
  $eind$ wordt onder c afgebeeld op niets
  Dus, $\delta(t_6, c) \coloneqq t_6$.

- $\delta(t_7, a) = ?$
  $1c$ wordt onder a afgebeeld op $1c$

$X(a)$ wordt onder a afgebeeld op $X(a)$
Dus, $\delta(t_7, a) \coloneqq t_7$.

- $\delta(t_7, b) = ?$
  $1c$ wordt onder b afgebeeld op $1c$
  $X(a)$ wordt onder b afgebeeld op $X(a), (a)X$
  Dus, $\delta(t_7, b) \coloneqq t_1$.
- $\delta(t_7, c) = ?$
  $1c$ wordt onder c afgebeeld op $X(a)$
  $X(a)$ wordt onder c afgebeeld op $X(a), (a)X$
  Dus, $\delta(t_7, c) \coloneqq t_8$.

- $\delta(t_8, a) = ?$
  $X(a)$ wordt onder a afgebeeld op $X(a)$
  $(a)X$ wordt onder a afgebeeld op $eind$
  Dus, $\delta(t_8, a) \coloneqq t_9$.
- $\delta(t_8, b) = ?$
  $X(a)$ wordt onder b afgebeeld op $X(a), (a)X$
  $(a)X$ wordt onder b afgebeeld op $eind$
  Dus, $\delta(t_8, b) \coloneqq t_6$.
- $\delta(t_8, c) = ?$
  $X(a)$ wordt onder c afgebeeld op $X(a), (a)X$
  $(a)X$ wordt onder c afgebeeld op $eind$
  Dus, $\delta(t_8, c) \coloneqq t_6$.

- $\delta(t_9, a) = ?$
  $X(a)$ wordt onder a afgebeeld op $X(a)$
  $eind$ wordt onder a afgebeeld op niets
  Definieer $t_{10} \coloneqq \{X(a)\}$ en $\delta(t_9, a) \coloneqq t_{10}$.
- $\delta(t_9, b) = ?$
  $X(a)$ wordt onder b afgebeeld op $X(a), (a)X$
  $eind$ wordt onder b afgebeeld op niets
  Dus, $\delta(t_9, b) \coloneqq t_8$.
- $\delta(t_9, c) = ?$
  $X(a)$ wordt onder c afgebeeld op $X(a), (a)X$
  $eind$ wordt onder c afgebeeld op niets
  Dus, $\delta(t_9, c) \coloneqq t_8$.

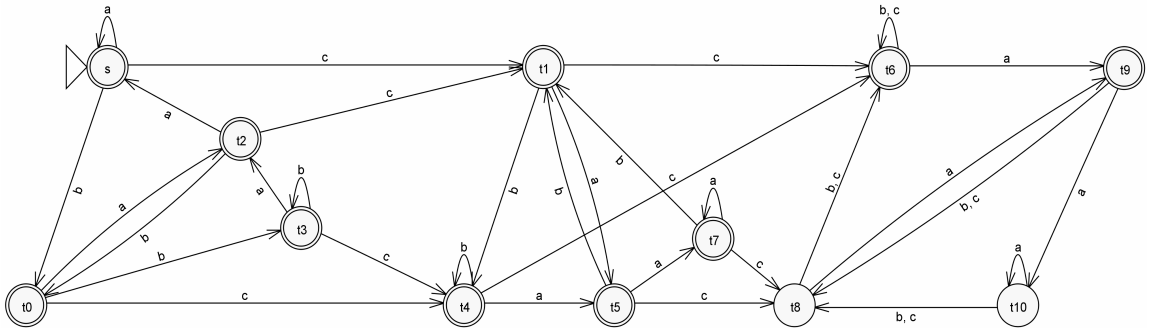
- $\delta(t_{10}, a) = ?$
  $X(a)$ wordt onder a afgebeeld op $X(a)$
  Dus, $\delta(t_{10}, a) \coloneqq t_{10}$.
- $\delta(t_{10}, b) = ?$
  $X(a)$ wordt onder b afgebeeld op $X(a), (a)X$
  Dus, $\delta(t_{10}, b) \coloneqq t_8$.

- $\delta(t_{10}, c) = ?$
  $X(a)$ wordt onder c afgebeeld op $X(a), (a)X$
  Dus, $\delta(t_{10}, c) \coloneqq t_8$.

We vatten even samen:

$$
\begin{aligned}
s &= \{start, X(a)\} \\
t_0 &= \{start, X(a), (a)X\} \\
t_1 &= \{1c, X(a), (a)X\} \\
t_2 &= \{start, X(a), eind\} \\
t_3 &= \{start, X(a), (a)X, eind\} \\
t_4 &= \{1c, X(a), (a)X, eind\} \\
t_5 &= \{X(a), (a)X, eind\} \\
t_6 &= \{X(a), (a)X, eind\} \\
t_7 &= \{1c, X(a)\} \\
t_8 &= \{X(a), (a)X\} \\
t_9 &= \{X(a), eind\} \\
t_{10} &= \{X(a)\}
\end{aligned}
$$

Alle states met $start$, $1c$ of $eind$ zijn accepting. Het resultaat is voorgesteld op Figuur 3.



Figuur 3: Niet-minimale deterministische herkenner voor L

Dit is duidelijk niet erg efficiënt – we zullen de deterministische herkenner dus minimaliseren.

Minimaliseren van de deterministische herkenner:

Eerst onderscheiden we twee soorten van toestanden: de niet-accepting states nemen we samen in A en de accepting states in B:

$$
\begin{aligned}
A &\coloneqq \{s, t_0, t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_9\} \\
B &\coloneqq \{t_8, t_{10}\}
\end{aligned}
$$

We zien dat $\delta(t_8, b) = t_6 \in A$ en $\delta(t_{10}, b) = t_8 \in B$, dus t_8 en t_{10} moeten apart gehouden worden.

$$
\begin{aligned}
A &\coloneqq \{s, t_0, t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_9\} \\
B &\coloneqq \{t_8\} \\
C &\coloneqq \{t_{10}\}
\end{aligned}
$$

$\delta(t_9, a) = t_{10} \in C$ – voor de rest is er geen enkele $q \in A$ waarvoor geldt dat $\delta(q, a) \in C$, dus deze moet apart gezet worden. Analoog zien we dat t_6 moet apart gezet worden omdat $\delta(t_6, a) = t_9$.

$$A \coloneqq \{s, t_0, t_1, t_2, t_3, t_4, t_5, t_7\}$$
$$B \coloneqq \{t_8\}$$
$$C \coloneqq \{t_{10}\}$$
$$D \coloneqq \{t_9\}$$
$$E \coloneqq \{t_6\}$$

We blijven A opsplitsen. We zien nu dat $\delta(t_5, c) \in B$ en $\delta(t_7, c) \in B$. Behalve deze twee is er geen enkele $q \in A$ waarvoor geldt dat $\delta(q, c) \in B$ – deze twee moeten dus uit A.

$$A \coloneqq \{s, t_0, t_1, t_2, t_3, t_4\}$$
$$B \coloneqq \{t_8\}$$
$$C \coloneqq \{t_{10}\}$$
$$D \coloneqq \{t_9\}$$
$$E \coloneqq \{t_6\}$$
$$F \coloneqq \{t_5, t_7\}$$

We blijven A opsplitsen. We zien nu dat $\delta(t_1, c) \in E$ en $\delta(t_4, c) \in E$. Behalve deze twee is er geen enkele $q \in A$ waarvoor geldt dat $\delta(q, c) \in E$ – deze twee moeten dus uit A.

$$A \coloneqq \{s, t_0, t_2, t_3\}$$
$$B \coloneqq \{t_8\}$$
$$C \coloneqq \{t_{10}\}$$
$$D \coloneqq \{t_9\}$$
$$E \coloneqq \{t_6\}$$
$$F \coloneqq \{t_5, t_7\}$$
$$G \coloneqq \{t_1, t_4\}$$

We kunnen A nu niet meer opsplitsen: er geldt:

$$\forall q \in A: \delta(q, a) \in A \wedge \delta(q, b) \in A \wedge \delta(q, c) \in G$$

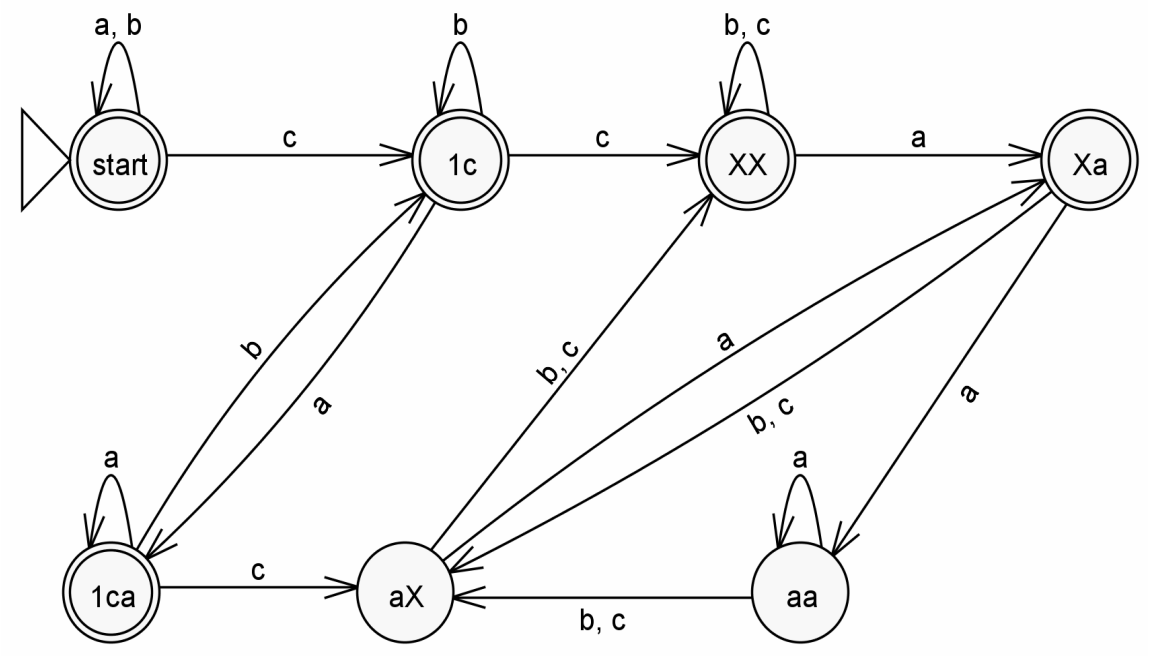
Analoog zien we dat we F en G ook niet meer moeten opsplitsen. We hebben dus een minimale edh bereikt. Om de oplossing duidelijker te maken, geven we de toestanden andere namen: $start \coloneqq A, 1c \coloneqq G, 1ca \coloneqq F, XX \coloneqq E, Xa \coloneqq D, aX \coloneqq B, aa \coloneqq C$. Het resultaat is weergegeven in Figuur 4.

Reguliere grammatica:

Aan de hand van de minimale edh die we bepaald hebben, definiëren we de grammatica $G = (\{S, A, B, C, D, E, F\}, \{a, b, c\}, R, S)$ met de productieregels:

$$S \to aS|bS|cA|\varepsilon$$
$$A \to aB|bA|cC|\varepsilon$$
$$B \to aB|bA|cD|\varepsilon$$
$$C \to aE|bC|cC|\varepsilon$$
$$D \to aE|bC|cC$$
$$E \to aF|bD|cD|\varepsilon$$
$$F \to aF|bD|cD$$

Het verband met de edh is duidelijk: S is afgeleid uit de toestand $start$, A uit $1c$, B uit $1ca$, C uit XX, D uit aX, E uit Xa en F uit aa.



Figuur 4: De minimale edh voor L. De redenering is duidelijk: in de toestand "start" is alles nog in orde, maar vanaf er 1 c is (toestand 1c), moeten we opletten. Als er een a bij komt, en dit is de laatste letter, zijn we in toestand 1ca. Zodra er nog een c bij komt, moeten we zorgen dat de voorlaatste letter geen a is (daarom was 1ca een aparte toestand): dit gebeurt in de 4 meest rechtse toestanden. De namen van deze toestanden geven de 2 laatste letters weer – een X betekent hierbij een b of een c.

Reguliere expressie:

We beschouwen de volgende reguliere expressie:

$$\big((a \cup b)^*(c \cup \varepsilon)(a \cup b)^*\big) \cup \big((a \cup b \cup c)^*(b \cup c)(a \cup b \cup c)\big)$$

We onderscheiden 2 soorten input die toegelaten zijn in L: er is maximaal één c in w, of de voorlaatste letter van w is geen a. Dat is precies wat deze reguliere expressie beschrijft.

6. WAAR OF VALS?

Vals:

We geven een tegenvoorbeeld:

$$L_1 = \{ab\}$$
$$L = \{a^n b^n : n \geq 0\}$$
$$L_2 = \{a^*b^*\}$$

We weten dat L_1 en L_2 beide reguliere talen zijn, maar L is geen reguliere taal. Toch geldt $L_1 \subseteq L \subseteq L_2$.

7. LOOPTRAJECT

a) Contextvrije grammatica:

We definiëren de grammatica $G = (\{S, B\}, \{a, b, c\}, R, S)$, met de productieregels

$$S \to aBaBaBa$$
$$B \to b^{45} | c^{45}$$

Reguliere grammatica:

Ja, er bestaat een reguliere grammatica die deze figuur voorstelt. De figuur is duidelijk eindig en kan met een eindig aantal stappen voorgesteld worden (in realiteit kan het draaien oneindig veel stappen lang zijn, maar dit kan opgelost worden door met een modulosysteem te werken zoals in vraag 1b). We kunnen dus gewoon alle gevallen apart voorstellen in een reguliere grammatica.

b) Contextvrije grammatica:

We definiëren de grammatica $G = (\{S\}, \{a, b, c\}, R, S)$, met de productieregel

$$S \to b^{45}ab^{45}aS \mid c^{45}ac^{45}aS \mid \varepsilon$$

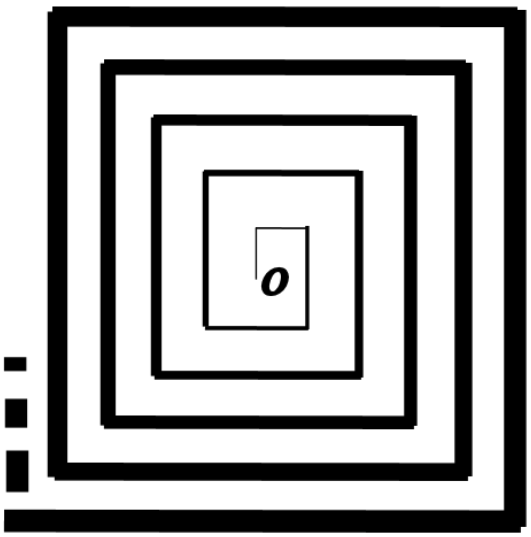
Beschrijving:

De startpositie is dan om het even de welke hoek van om het even het welk vierkant (tussen 4 achthoeken), en we kijken naar het midden van het vierkant toe. Met deze grammatica genereren we dan een pad dat telkens 2 keer links of 2 keer rechts afslaat, zodat we bij een volgend vierkant terecht komen, waar we hetzelfde doen. Zo kunnen we een achthoek tekenen of aan een andere achthoek beginnen. Uiteindelijk zal de productieregel $S \to \varepsilon$ gekozen worden: zo stopt het genereren en bekomen we een woord dat enkel bestaat uit terminalen. De figuur in de opgave is dan de combinatie van alle mogelijke woorden.

c) Geen contextvrije grammatica:

Ja, er bestaat een figuur waarvoor er geen contextvrije grammatica bestaat. Een voorbeeld is een rechthoekige spiraal die oneindig doorgaat, zoals op Figuur 5. Als we zouden beginnen op positie O zoals aangegeven op de figuur, kunnen we deze spiraal vormen met $ab^{90}ab^{90}aab^{90}aab^{90}aaab^{90}aaab^{90}$ ...

We zien dat het aantal a's steeds na 2 b^{90}'s toeneemt.



Figuur 5: Voor deze oneindige spiraal bestaat er geen contextvrije grammatica.

Als deze taal contextvrij zou zijn, zou er volgens het pumping lemma een $k \geq 1$ bestaan, zodanig dat voor iedere $w \in L$ met $|w| \geq k$, er strings u, v, x, y, z bestaan zodanig dat

$$w = uvxyz, |vxy| \leq k, |vy| > 0$$

en $w_q = uv^q xy^q z \in L$ voor iedere $q \in \mathbb{N}$.

Het is duidelijk dat deze strings niet bestaan: zodra we iets oppompen, zal er een stukje van de spiraal te lang of te kort zijn, of zal er plots in een andere richting gedraaid worden, en zal de figuur helemaal verstoord zijn.