Uitwerking huiswerkopgave Fundamentele Informatica 1 (I&E), najaar 2015

a) Een contextvrije grammatica G_1 voor $L_1 = \{ a^{i_1} b^{j_1} a^{i_2} b^{j_2} \dots a^{i_m} b^{j_m} \mid m \geq 0$ en voor $k = 1, \dots, m$ geldt dat $i_k \geq j_k \geq 1 \}$

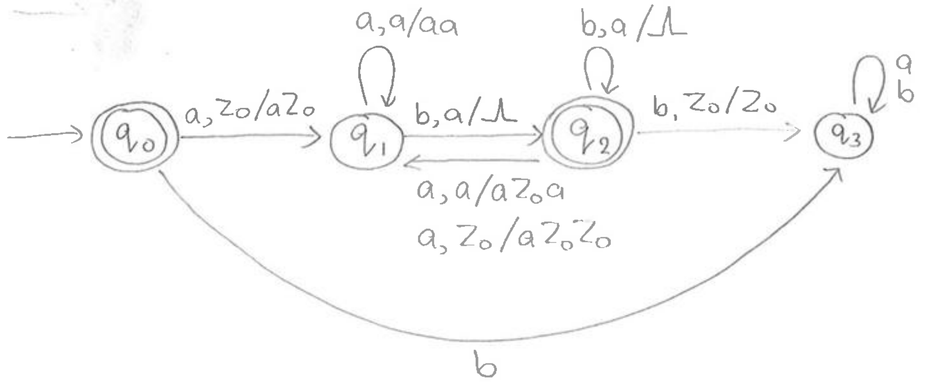
Neem de volgende producties voor G_1:

$S \to TS \mid \Lambda$ S is startsymbool met deze producties genereren we nul of meer variabelen T. Het aantal T's wordt m

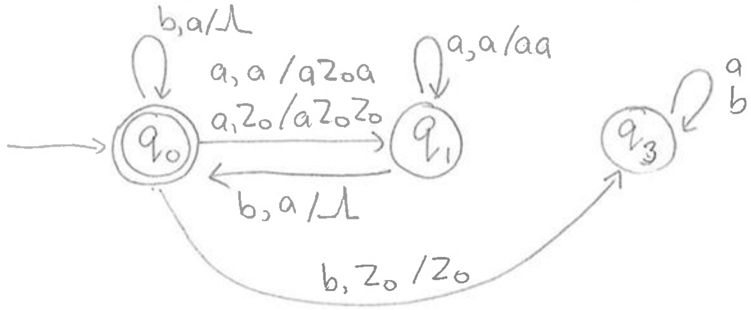
$T \to aTb \mid aT \mid ab$ T is verantwoordelijk voor $a^{i_k} b^{j_k}$, met $i_k \geq j_k \geq 1$.

Bij elke b moet ook een a gegenereerd worden (met $T \to aTb$ en $T \to ab$), er mogen extra a's gegeneerd worden (met $T \to a$) en er moet minstens een b gegenereerd worden ($j_k \geq 1$, met 'slotproductie' $T \to ab$).

b) Een PDA M_1 voor L_1:



Het kan ook met drie toestanden:



M_1 begint in q_0, en dat is een accepterende toestand, omdat $\Lambda \in L_1$. Wanneer er dan een a wordt gelezen, gaan we naar q_1, waar we net zoveel a's op de stapel zetten als dat we lezen. q_1 is niet accepterend, omdat $j_k \geq 1$ moet gelden, dus we moeten eindigen met een b. Als we dan een b lezen, gaan we terug naar de accepterende toestand q_0. Daar gaan we verder met b's lezen, waarbij we steeds een a afstapelen.

Als de a's op de stapel op zijn en we toch nog een b lezen (of een b lezen als eerst letter) gaan we van q_0 naar het afvoerputje q_3, waar we de rest van de invoer kunnen lezen, maar nooit meer kunnen accepteren: we hebben immers een k gehad met $i_k < j_k$.

Als we in q_0 weer een a lezen, laten we eventuele resterende a's op de stapel staan. We zetten er gewoon een nieuw symbool Z_0 bovenop en beginnen daar bovenop nieuwe a's te stapelen.

c] Een contextvrije grammatica G_2 voor L_2

Elke a die 'teveel' is in $a^{i_k}b^{j_k}$ moet aan het eind gecompenseerd worden met een b. Deze extra a's moeten dan niet meer vanuit T gegenereerd worden, omdat (in G_1) er een variabele S tussen T en het eind van de string staat. Hierdoor kan T niet bij het eind van de string komen, op ook een b toe te voegen. We genereren de extra a's daarom vanuit S:

$$S \to aSb \mid TS \mid \Lambda$$
$$T \to aTb \mid ab$$

T is nu puur verantwoordelijk voor $a^{j_k}b^{j_k}$ met $j_k \geq 1$: bij elke a genereert T ook een b (en omgekeerd).

S genereert (met $S \to aSb$) eerst extra a's links en compenserende b's rechts, voordat hij (met $S \to TS$) T introduceert. De nieuwe S die hierbij (bij $S \to TS$) ontstaat, kan dan de volgende $a^{i_k}b^{j_k}$ weer genereren, met compenserende b's.

S kan ook Λ worden, als we genoeg $a^{i_k}b^{j_k}$'s hebben, en ook meteen aan het begin, om de lege string te genereren.

Merk op dat de laatste $a^{i_k}b^{j_k}$ met compenserende b's rechtstreeks uit S te genereren is, zonder tussenkomst van T:

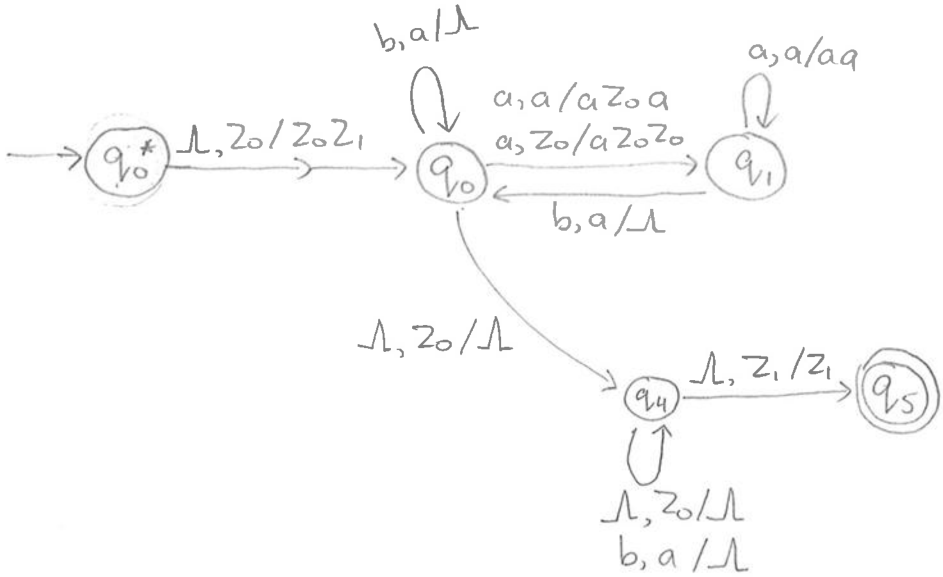
$S \Rightarrow^* a^{i_k}Sb^{i_k} \Rightarrow a^{i_k}b^{i_k} = a^{i_k}b^{j_k}$ met compenserende b's.

d] Bij de PDA M_1 van onderdeel (b) laten we de extra a's van elke $a^{i_k}b^{j_k}$ op de stapel staan. Nu moeten we ze aan het eind er alsnog afhalen, bij het lezen van de 'compenserende' b's.

Daarbij halen we ook de tussenliggende Z_0's van de stapel.

Om te weten wanneer we de laatste Z_0 te pakken hebben (zodat ook alle a's van de stapel zijn) en we kunnen accepteren, zetten we aan het begin een speciaal symbool Z_1 onder op de stapel

We krijgen dan de volgende stapelautomaat M_2:



Toestand q_0^* hoeft niet accepterend te zijn, want we kunnen de lege string Λ accepteren via q_0, q_4 en q_5.
In toestand q_4 halen we de stapel leeg: de Z_0's gaan ervanaf zonder iets te lezen, de a's bij het lezen van compenserende b's.
De PDA is niet-deterministisch in toestand q_0: gaan we naar q_1 of naar q_4 als we Z_0 op de stapel zien. De reden om met een Λ-transitie van q_0 naar q_4 te gaan (waardoor het niet-determinisme ontstaat), is dat het mogelijk is dat er helemaal geen a's meer op de stapel staan onder de Z_0 die we zien, bijvoorbeeld na het lezen van $a^3b^3a^2b^2a^4b^4$.
Op dat moment bevat de stapel $\begin{matrix}Z_0\\Z_0\\Z_0\\Z_0\\Z_1\end{matrix}$, en moeten we geen b meer lezen.

en niet met een b-transitie

Een alternatief zou zijn om met speciale toestanden of speciale stapelsymbolen bij te houden of er onder de bovenste Z_0 op de stapel minstens een a staat. Dan kun je een b-transitie gebruiken in plaats van een Λ-transitie om de overgang te maken naar het leegmaken van de stapel bij het lezen van compenserende b's.
Op die manier kunnen we een deterministische PDA krijgen.