5612 AZ Eindhoven
P.O. Box 513, 5600 MB Eindhoven
The Netherlands
www.tue.nl

Author
Christiaan Dirkx
Alex Dings

Date
June 8, 2015

Version
0.3

Automata answers explained

Where innovation starts

Table of contents

Title
Automata answers explained

Table of contents

Title
Automata answers explained

Table of contents

Title
Automata answers explained

Table of contents

Title
Automata answers explained

0 Introduction

Introduction 1: Special sets

Set			Element	
Name	**Notation**	**Definition**	**Name**	**Notation**
alphabet	Σ	Enumeration ($\neq \emptyset$)	symbol letter	$a, b,$ ... arbitrary symbols
n-symbol strings over an alphabet	$\Sigma^n, n \geq 0$	finite product (see below)	string word	$a_1, a_2, a_n, n \geq 0$ $a_1, a_2, a_n = \epsilon$
all finite strings over an alphabet	Σ^*	1. Set union 2. Set induction	string word	ϵ, empty word $w, v, u,$ arbitrary word
language	L	subset of Σ^*	word	

Definition 0.1
$\Sigma^n = \{a_1a_2...a_n | \forall i, 1 \leq i \leq n : a_i \in \Sigma\}$
$$\Sigma^* = \bigcup_{n=0}^{\infty} \Sigma^n$$

Definition 0.2
1. The empty word $\epsilon \in \Sigma^*$
2. If $a \in \Sigma, w \in \Sigma^*$, then $aw \in \Sigma^*$

Introduction 2: Relations on Σ^*

Name	Notation	Definition
(is a) prefix (of)	$v \preccurlyeq w$	$\exists u, u \in \Sigma^* : vu = w$
(is a) suffix (of)	$v \succcurlyeq w$	$\exists u, u \in \Sigma^* : uv = w$
(is a) substring (of)		$\exists x, x \in \Sigma^* : \exists y, y \in \Sigma^* xvy = w$

Uniqueness properties
1. $vu_1 = w$ and $vu_2 = w$ implies $u_1 = u_2$
unique suffix is denoted as a quotient $u = w/v$
2. $u_1v = w$ and $u_2v = w$ implies $u_1 = u_2$

Introduction 3: Operators and functions

Name	Notation	Definition
length	$\|v\|$	inductive 1. $\|\epsilon\| = 0$ 2. $\|aw\| = 1 + \|w\|$
count, for all $c \in \epsilon$	$\#_c(v)$	inductive 1. $\#_c(\epsilon) = 0$ 2. $\#_c(cw) = 1 + \#_c(w)$ 3. $\#_c(aw) = \#_c(w), a \neq c$
concatenation of strings	juxtaposition wv	inductive 1. $\epsilon v = v$ 2. $(aw)v = a(wv)$
concatenation of languages	$L_1 \cdot L_2$	set comprehension = $\{w_1w_2 \| w_1 \in L_1, w_2 \in L_2\}$
Kleene closure	L^*	set comprehension = $\{w_1w_2...w_n \| n \geq 0, w_1, w_2, ..., w_n \in L\}$

Introduction 4: (Algebraic) properties

Name	Notation (algebraic law)	Proof
unit of string concatenation	$\epsilon v = v = v\epsilon$	by induction
associativity of string concatenation	$(wv)u = w(vu)$	by induction
additivity of length operator	$\|wv\| = \|w\| + \|v\|$	by induction
additivity of count operator	$\#_c(wv) = \#_c(w) + \#_c(v)$	by induction
zero of language concatenation	$\emptyset \cdot L = \emptyset = L \cdot \emptyset$	element wise
unit of language concatenation	$\{\epsilon\} \cdot L = L = L \cdot \{\epsilon\}$	element wise

1 Preliminaries

2 Finite Automata and Regular Languages

3 Push-Down Automata and Context-Free Languages

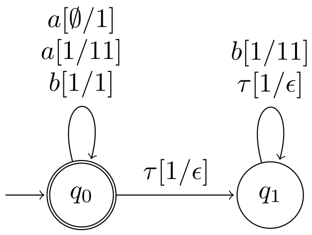
Learning targets chapter 3

At the end of this chapter the student should be able to:

- Construct a PDA from an language or CFG.
- Construct a CFG from an language or PDA.
- Give an invariant table of a PDA.
- Give derivations of a CFG.
- Prove that a CFG is equivalent to a language.
- Show a CFG to be ambiguous.
- Construct an unambiguous CFG from an ambiguous one.
- Prove that a language is context free.
- Prove that a language is not context free.

Exercise 3.1

(Hopcroft, Motwani & Ullman, 2001) Consider the following PDA.



Compute all maximal derivation sequences for the following inputs:

(a) ab;

(b) aabb;

(c) aba.

A maximal derivation sequence of a PDA P for a string w is a sequence

$$(q_0, w_0, x_0) \vdash P(q_1, w_1, x_1) \vdash P...(q_{n-1}, w_{n-1}, x_{n-1}) \vdash P(q_n, w_n, x_n) \nvdash P$$

where $q_0, q_1, ..., q_{n-1}$, q_n are states of P with q_0 its initial state, $w_0, w_1, ..., w_{n-1}, w_n$ strings over the input alphabet of P with w_0 equal to w, and $x_0, x_1, ..., x_{n-1}, x_n$ strings over the stack alphabet of P with x_0 equal to ϵ, the empty stack.

(a) $(q_0, ab, \epsilon) \vdash (q_0, b, 1) \vdash (q_0, \epsilon, 1) \vdash (q_1.\epsilon, \epsilon)$
$\vdash (q_1, b, \epsilon)$

(b) $(q_0, aabb, \epsilon) \vdash (q_0, abb, 1) \vdash (q_0, bb, 11) \vdash (q_0, b, 11) \vdash (q_0, \epsilon, 11) \vdash (q_1, \epsilon, 1) \vdash (q_1, \epsilon, \epsilon)$
$\vdash (q_1, abb, 1)$
$\vdash (q_1, bb, 1) \vdash (q_1, b, 11) \vdash (q_1, \epsilon, 111) \overset{3x}{\vdash} (q_1, \epsilon, \epsilon)$
$\vdash (q_1, b, 1) \vdash (q_1, \epsilon, 11) \overset{2x}{\vdash} (q_1, \epsilon, \epsilon)$
$\vdash (q_1, b, \epsilon)$
$\vdash (q_1, bb, \epsilon)$
$\vdash (q_1, b, 1) \vdash (q_1, \epsilon, 11) \vdash (q_1, \epsilon, 1) \vdash (q_1, \epsilon, \epsilon)$
$\vdash (q_1, b, \epsilon)$

(c) $(q_0, aba, \epsilon) \vdash (q_0, ba, 1) \vdash (q_0, a, 1) \vdash (q_o, \epsilon, 11) \vdash (q_1, \epsilon, 1) \vdash (q_1, \epsilon, \epsilon)$
$\vdash (q_1, a, \epsilon)$
$\vdash (q_1, ba, \epsilon)$

Configurations marked in red are accepting configurations.

Exercise 3.2

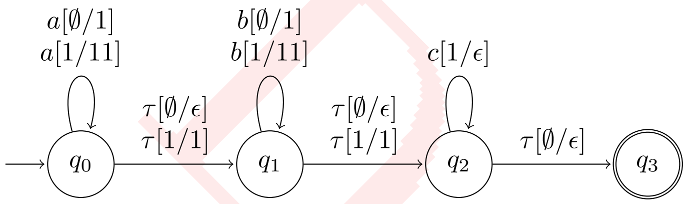
Construct a push-down automaton and give an invariant table for the following languages over the input alphabet $\Sigma = \{a, b, c\}$

(a) $L_1 = \{a^n b^m c^{n+m} | n, m \geq 0\}$;

(b) $L_2 = \{a^{n+m} b^n c^m | n, m \geq 0\}$;

(c) $L_3 = \{a^n b^{n+m} c^m | n, m \geq 0\}$;

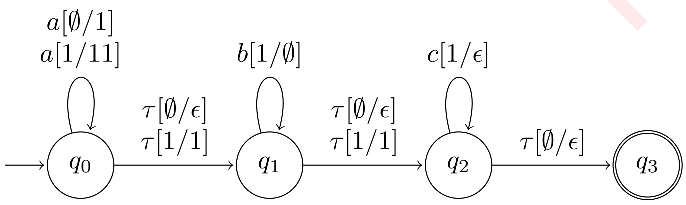
(a) $L_1 = \{a^n b^m c^{n+m} | n, m \geq 0\}$



interpretation:
$(q_o,\underline{w},\epsilon) \vdash^* (\underline{q},\epsilon,\underline{x})$

state $\underline{y}$	input $\underline{w}$	stack $\underline{x}$	constraints
q_0	a^n	1^n	$n \geq 0$
q_1	$a^n b^m$	1^{n+m}	$n, m \geq 0$
q_2	$a^n b^m c^p$	1^{n+m-p}	$0 \leq p \leq n+m; n, m \geq 0$
q_3	$a^n b^m c^{n+m}$	ϵ	$n, m \geq 0$

(b) $L_2 = \{a^{n+m} b^n c^m | n, m \geq 0\}$



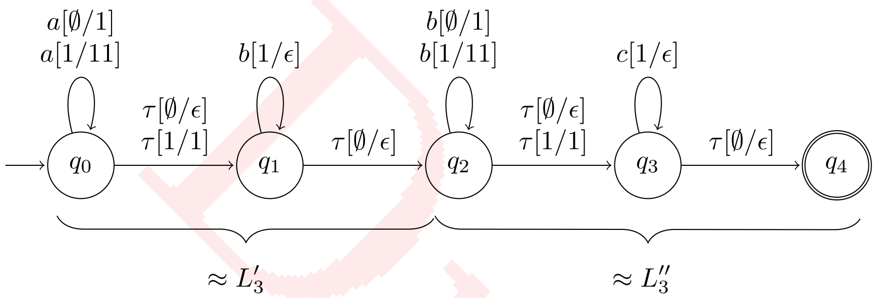
state y	input w	stack x	constraints
q_0	a^p	1^p	$p \geq 0$
q_1	$a^p b^q$	1^{p-q}	$0 \leq q \leq p$
q_2	$a^p b^q c^r$	1^{p-q-r}	$0 \leq q + r \leq p; q, r \geq 0$
q_3	$a^p b^q c^r$	ϵ	$q + r = p; q, r \geq 0$

(c) $L_3 = \{a^n b^{n+m} c^m | n, m \geq 0\}$

$$
\begin{aligned}
L_3 &= \{a^n b^{n+m} c^m | n, m \geq 0\} \\
&= \{a^n b^n b^m c^m | n, m \geq 0\} \\
&= \{a^n b^n | n \geq 0\} \cdot \{b^m c^m | m \geq 0\}
\end{aligned}
$$

Let be:

$$
\begin{aligned}
L_3' &= \{a^n b^n | n \geq 0\} \\
L_3'' &= \{b^m c^m | m \geq 0\}
\end{aligned}
$$



state y	input w	stack x	constraints
q_0	a^n	1^n	$n \geq 0$
q_1	$a^n b^p$	1^{1-p}	$0 \leq q \leq p$
q_2	$a^n b^{n+m}$	1^m	$n \geq 0, m \geq 0$
q_3	$a^n b^{n+m} c^q$	1^{m-q}	$0 \leq q \leq m, n \geq 0$
q_4	$a^n b^{n+m} c^m$	ϵ	$n, m \geq 0$

Exercise 3.3

Give a push-down automaton and invariant table for each of the following languages:

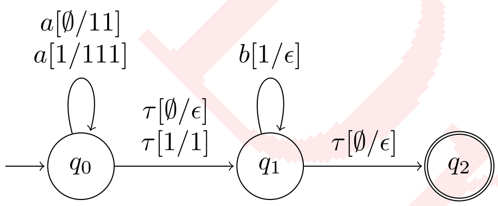
(a) $L_4 = \{a^n b^{2n} | n \geq 0\}$;

(b) $L_5 = \{a^n b^m | m \geq n \geq 1\}$;

(c) $L_6 = \{a^n b^m | 2n = 3m + 1\}$;

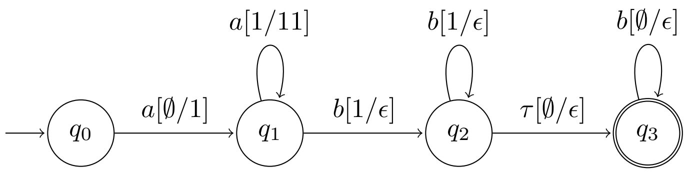
(d) $L_7 = \{a^n b^m | m, n \geq 0, m \neq n\}$.

(a) $L_4 = \{a^n b^{2n} | n \geq 0\}$



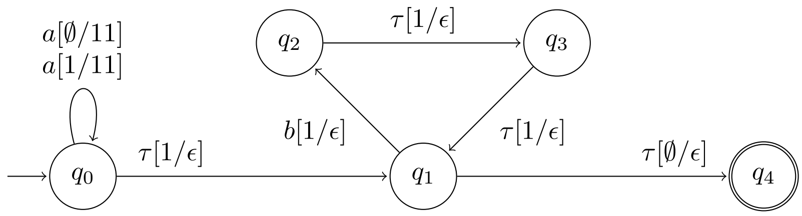
state	input	stack	constraints
q_0	a^n	1^{2n}	$n \geq 0$
q_2	$a^n b^m$	1^{2n-m}	$n \geq 0, 0 \leq m \leq 2n$
q_1	$a^n b^{2n}$	ϵ	$n \geq 0$

(b) $L_5 = \{a^n b^m | m \geq n \geq 1\}$



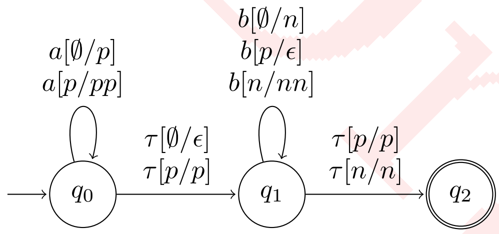
state	input	stack	constraints
q_0	ϵ	ϵ	
q_1	a^n	1^n	$n \geq 1$
q_2	$a^n b^m$	1^{n-m}	$1 \leq m \leq n$
q_3	$a^n b^m$	ϵ	$m \geq n \geq 1$

(c) $L_6 = \{a^n b^m | 2n = 3m + 1\}$



state	input	stack	constraints
q_0	a^n	1^{2n}	$n \geq 0$
q_1	$a^n b^m$	$1^{2n-3m-1}$	$3m + 1 \leq 2n, m \geq 0, n > 0$
q_2	$a^n b^m$	$1^{2n-3m+1}$	$3m - 1 \leq 2n, m \geq 0, n > 0$
q_3	$a^n b^m$	1^{2n-3m}	$3m \leq 2n, m \geq 0, n > 0$
q_4	$a^n b^m$	ϵ	$3m + 1 \leq 2n, m \geq 0, n > 0$

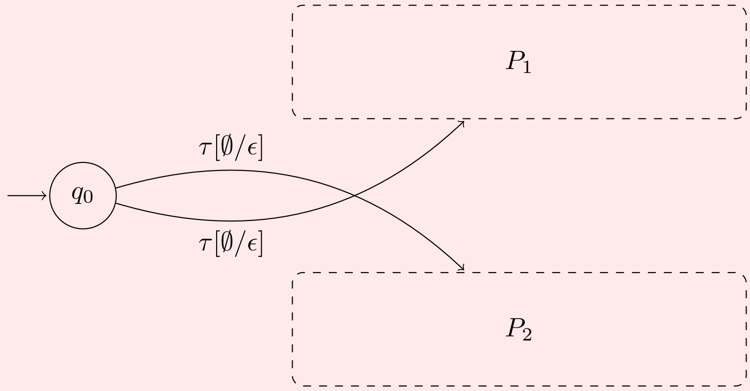
(d) $L_7 = \{a^n b^m | m, n \geq 0, m \neq n\}$



state	input	stack	constraints
q_0	a^n	p^n	$n \geq 0$
q_1	$a^n b^m$	p^{n-m}	$n \geq m \geq 0$
q_1	$a^n b^m$	n^{m-n}	$m \geq n \geq 0$
q_2	$a^n b^m$	p^{n-m}	$n \geq m \geq 0$
q_2	$a^n b^m$	n^{m-n}	$m \geq n \geq 0$

Closure property on union of two languages

if L_1 accepted by P_1
and L_2 accepted by P_2
then $L_1 \cup L_2$ accepted by a push-down automaton of the form:



Exercise 3.4

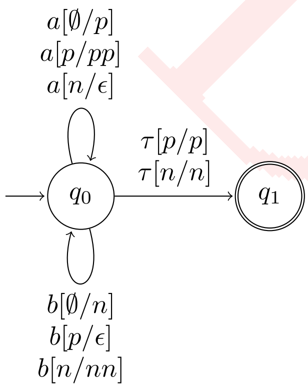
(a) Give a push-down automaton for the language

$$L_8 = \{w \in \{a,b\} | \#_a(w) \neq \#_b(w)\}$$

(b) Give a push-down automaton for the language

$$L_9 = \{w \in \{a,b,c\} | \#_a(w) \neq \#_b(w) \vee \#_b(w) \neq \#_c(w)\}$$

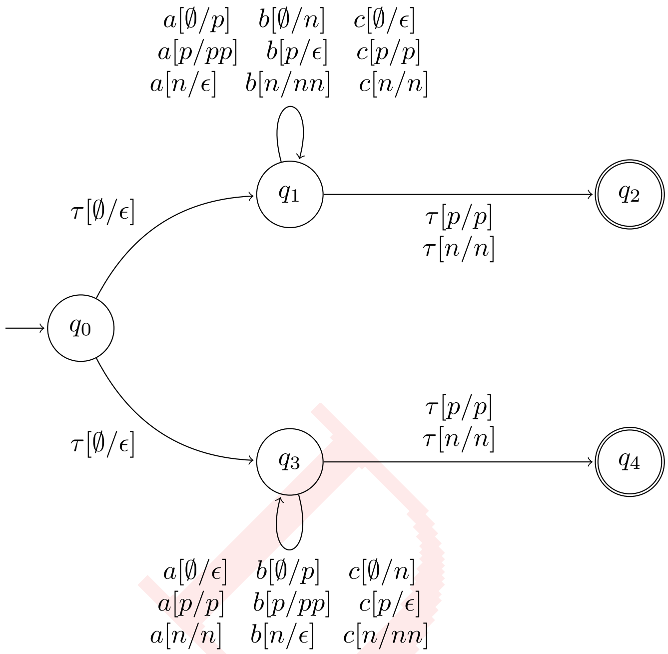
(a) $L_8 = \{w \in \{a,b\} | \#_a(w) \neq \#_b(w)\}$
This can be seen as a generalization of Exercise 3.4d



state	input	stack	constraints
q_0	w	p^h	$h = \#_a(w) - \#_b(w) \geq 0$
q_0	w	$n^{-\ell}$	$\ell = \#_a(w) - \#_b(w) \leq 0$
q_1	w	p^h	$h = \#_a(w) - \#_b(w) \geq 0$
q_1	w	$n^{-\ell}$	$\ell = \#_a(w) - \#_b(w) \leq 0$

(b) $L_9 = \{w \in \{a,b,c\} | \#_a(w) \neq \#_b(w) \vee \#_b(w) \neq \#_c(w)\}$
By following the idea from **[NOTE 1:** ref**]** we come up with the following automaton:

Copyholder
Christiaan Dirkx & Alex Dings



Definitions on.. [NOTE 2: whut]

A production rule:
$S \to XbY$

A production step:
$S \Rightarrow_G XbY$

A production sequence derivation:
$S \Rightarrow_G XbY \Rightarrow_G abY \Rightarrow_G abb$

Exercise 3.5

(Hopcroft, Motwani & Ullman 2001) Consider the context-free grammar G given by the production rules

$$\begin{aligned} S &\to XbY \\ X &\to \epsilon | aX \\ Y &\to \epsilon | aY | bY \end{aligned}$$

that generates the language of the regular expression $ab(a + b)$. Give leftmost and rightmost derivations for the following strings:

(a) $aabab$;

(b) $baab$;

(c) $aaabb$.

Normal production steps, following the production rule are denoted in the following manner: $S \overset{\ell}{\Rightarrow}_G XbY \overset{\ell}{\Rightarrow}_G aXbY$
As it is clear from the context that we are talking about the context-free grammar G, this is omitted in the following derivations.

Left most derivations:

(a) $S \overset{\ell}{\Rightarrow} XbY \overset{\ell}{\Rightarrow} aXbY \overset{\ell}{\Rightarrow} aaXbY \overset{\ell}{\Rightarrow} aabY \overset{\ell}{\Rightarrow} aabaY \overset{\ell}{\Rightarrow} aababY \overset{\ell}{\Rightarrow} aabab$

(b) $A \overset{\ell}{\Rightarrow} XbY \overset{\ell}{\Rightarrow} bY \overset{\ell}{\Rightarrow} baY \overset{\ell}{\Rightarrow} baaY \overset{\ell}{\Rightarrow} baabY \overset{\ell}{\Rightarrow} baab$

(c) $S \overset{\ell}{\Rightarrow} XbY \overset{\ell}{\Rightarrow} aXbY \overset{\ell}{\Rightarrow} aaXbY \overset{\ell}{\Rightarrow} aaXbY \overset{\ell}{\Rightarrow} aaabY \overset{\ell}{\Rightarrow} aaabbY \overset{\ell}{\Rightarrow} aaabb$

Right most derivations:

(a) $SvXbYvXbaY \overset{r}{\Rightarrow} XbabY \overset{r}{\Rightarrow} Xbab \overset{r}{\Rightarrow} aXbab \overset{r}{\Rightarrow} aaXbab \overset{r}{\Rightarrow} aabab$

(b) $S \overset{r}{\Rightarrow} XbY \overset{r}{\Rightarrow} XbaY \overset{r}{\Rightarrow} XbaaY \overset{r}{\Rightarrow} XbaabY \overset{r}{\Rightarrow} Xbaab \overset{r}{\Rightarrow} baab$

(c) $S \overset{r}{\Rightarrow} XbY \overset{r}{\Rightarrow} XbbY \overset{r}{\Rightarrow} Xbb \overset{r}{\Rightarrow} aXbb \overset{r}{\Rightarrow} aaXbb \overset{r}{\Rightarrow} aaaXbb \overset{r}{\Rightarrow} aaabb$

Lemma 3.15

[NOTE 3: to be filled**]**

Exercise 3.6

> Consider the context-free grammar G given by the production rules
>
> $$\begin{aligned} S &\to A|B \\ A &\to \epsilon|aA \\ B &\to \epsilon|bB \end{aligned}$$
>
> (a) Prove that $\mathcal{L}_G(A) = \{a^n|n \geq 0\}$.
> (b) Prove that $L(G) = \{a^n|n \geq 0\} \cup \{b^n|n \geq 0\}$.

Looking at the G, and the exercises (a & b) we can see that there are two languages. Let be:

$$L_a = \{a^n|n \geq 0\} \qquad\qquad L_b = \{b^n|n \geq 0\}$$

(a) *To be proven:* $\mathcal{L}_G(A) = L_a$

Firt part of the proof: $\mathcal{L}_G(A) \subseteq L_a$

Proof by induction on an n of:
If $A \Rightarrow_G^n w$ and $w \in \{a,b\}^*$, then $w \in L_a$, for all w

Base case: n=0. from $A \Rightarrow_G^0 w$ it follows that a=w.
Hence $\notin \{a,b\}^*$ so nothing needs to be proven.

Step: n=h+1, for some $h \geq 0$.

If $A \Rightarrow_G^h w$ and $w \in \{a,b\}*$, then $w \in L_a$, for all w *[IH]*

$A \Rightarrow_G^{h+1} w$ and $w \in \{a,b\}^*$

Case analysis on first step in derivation:

- $A \Rightarrow_G \epsilon \Rightarrow_G^h w$. It follows that $h = 0$ and $w = \epsilon$, so $w \in L_a$.
- $A \Rightarrow_G aA \Rightarrow_G^h w$. From Lemma 3.15 c it follows that $w = av$ and $A \Rightarrow_G^h v$, $v \in \{a,b\}^*$
  By the induction hypothesis $v \in L_a$, so $v = a^m$ for some $m \geq 0$..
  Thus $w = av = a^{m+1}$ and therefor $w \in L_a$.

Second part of the proof: $\mathcal{L}_G(A) \subseteq L_a$

Proof by induction on an n of:
If $A \Rightarrow_G^{n+1} a^n$

Base case: n=0. $A \to \epsilon$ is a production rule, so $a \Rightarrow_G^1 \epsilon = a^0$

Step: n=h+1, for some $h \geq 0$.

$A \Rightarrow^{h+1}$ *[IH]*

Due to $a \Rightarrow_G^0 a$ and Lemma 3.15 b: $aA \Rightarrow_G^{h+1} a^{h+1}$

Since $A \Rightarrow aA$ is a production rule, we have $A \Rightarrow_G aA \Rightarrow_G^{h+1} a^{h+1}$, so $A \Rightarrow_G^{h+2} a^{h+1}$

(b) *To be proven:* $\mathcal{L}(G) = L_a \cup L_b$

Proof:

$$
\begin{aligned}
& w \in \mathcal{L}(G) \\
\overset{val}{=} & \\
& w \in \mathcal{L}_G(S) \\
\overset{val}{=} & \\
& S \Rightarrow^* w \wedge w \in \{a,b\}^* \\
\overset{val}{=} & \{case distinction first step : S \Rightarrow A or S \Rightarrow B\} \\
& (A \Rightarrow^* w \vee B \Rightarrow^* w) \wedge w \in \{a,b\}^* \\
\overset{val}{=} & \\
& (A \Rightarrow^* \wedge w \in \{a,b\}^*) \vee (B \Rightarrow^* \wedge w \in \{a,b\}^*) \\
\overset{val}{=} & \\
& w \in \mathcal{L}_G(A) \vee w \in \mathcal{L}_G(B) \\
\overset{val}{=} & \{See exercise (a)\} \\
& w \in L_a \vee w \in L_b \\
\overset{val}{=} & \\
& w \in L_a \cup L_b
\end{aligned}
$$

Exercise 3.7

Give a context-free grammar for each of the following languages and prove them correct.

(a) $L_1 = \{a^n b^m | n, m \geq 0, n \neq m\}$;
(b) $L_2 = \{a^n b^m c^\ell | n, m, \ell \geq 0, n \neq m \vee m \neq \ell\}$;

(a)

$$\begin{aligned}
L_1 &= \{a^n b^m | n, m \geq 0; n \neq m\} \\
&= \{a^n b^m | n, m \geq 0; (n > m \vee n < m)\} \\
&= \{a^n b^m | n > m \geq 0\} \cup \{a^n b^m | 0 \leq n < m\} \\
&= \{a^{k+m} b^m | k > 0, m \geq 0\} \cup \{a^n b^{n+\ell} | n \geq 0, \ell > 0\} \\
&= \underbrace{\{a^h | h > 0\}}_{\mathcal{L}(A)} \cdot \underbrace{\{a^m b^m | m \geq 0\}}_{\mathcal{L}(T)} \cup \underbrace{\{a^n b^n | n \geq 0\}}_{\mathcal{L}(T)} \cdot \underbrace{\{b^\ell | \ell > 0\}}_{\mathcal{L}(B)}
\end{aligned}$$

CFG for L_1:

$$\begin{aligned}
S &\rightarrow AT | TB \\
T &\rightarrow \epsilon | aTb \\
A &\rightarrow a | aA \\
B &\rightarrow b | bB
\end{aligned}$$

Allowed arguments

$\mathcal{L}$ extended to strings of variables and terminals:
$\mathcal{L}(\epsilon) = \{\epsilon\}$ $\mathcal{L}(Xx) = \mathcal{L}(X) \cdot \mathcal{L}(x)$

Proof 1:
$\mathcal{L}(T) = \{a^m b^m | m \geq 0\}$
Proof analogous to Example 3.14 **[NOTE 4:** work out**]**

Proof 2:
$\mathcal{L}(A) = \{a^h | h > 0\}$

Proof 3:
$\mathcal{B}(A) = \{b^\ell | \ell > 0\}$ Proof analogous to Example 3.6a **[NOTE 5:** work out**]**

Lemmas: **[NOTE 6:** existing?**]**

- $\mathcal{L}(X_1 X_2 ... X_h) = \mathcal{L}(X_1) \cdot \mathcal{L}(X_2) ... \mathcal{L}(X_h)$
if $X \rightarrow x_1 | x_2 | ... | x_h$
then $\mathcal{L}(X) = \mathcal{L}(x_1) \cup \mathcal{L}(x_2) \cup ... \cup \mathcal{L}(X_h)$

Proof 4:

$$\begin{aligned}
\mathcal{L}(S) &= \mathcal{L}(AT) \cup \mathcal{L}(TB) \\
&= \mathcal{L}(A) \cdot \mathcal{L}(T) \cup \mathcal{L}(T) \cdot \mathcal{L}(B) \\
&= L_1 \textbf{[NOTE 7:} according to above lemmas\textbf{]}
\end{aligned}$$

(b)

$$
\begin{aligned}
L_2 &= \{a^n b^m c^\ell | n, m, \ell \geq 0, n \neq m \vee m \neq \ell\} \\
&= \{a^n b^m c^\ell | n, m, \ell \geq 0, n \neq m\} \cup \{a^n b^m c^\ell | n, m, \ell \geq 0, m \neq l\} \\
&= \underbrace{\{a^n b^m | n, m, n \neq m\}}_{\text{see (a)}} \cdot \{c^\ell | \ell \geq 0\} \cup \{a^n | n \geq 0\} \cdot \underbrace{\{a^n b^m c^\ell | n, m, \ell \geq 0, m \neq l\}}_{\text{see (a)}}
\end{aligned}
$$

$$S \to S_1C | AS_2$$

$$\begin{cases} S_1 \to A_1T_1 | T_1B_1 \\ T_1 \to \epsilon | aT_1b \\ A_1 \to a | aA_1 \\ B_1 \to b | bB_1 \end{cases}$$

$$\Big\{ C \to \epsilon c C$$

$$\Big\{ A \to \epsilon a A$$

$$\begin{cases} S_1 \to B_2T_2 | T_2C_2 \\ T_1 \to \epsilon | bT_2c \\ A_1 \to b | bB_2 \\ B_1 \to c | cC_2 \end{cases}$$

Exercise 3.8

Give a construction, based on the number of operators, that shows that every **[NOTE 8:** lol?**]** the language of every regular expression can be generated by a context-free grammar.

$G : RE_\Sigma \to CFG$ such that $\mathcal{L}(r) = \mathcal{L}(G(r))$

$G(\underline{0}) = (\{S_{\underline{0}}\}, \Sigma, \emptyset, S_{\underline{0}})$ $\quad \mathcal{L}(\underline{0}) = \emptyset = \mathcal{L}(G(\underline{0}))$

$G(\underline{1}) = (\{S_{\underline{1}}\}, \Sigma, S_{\underline{1}} \to \epsilon, S_{\underline{1}})$ $\quad \mathcal{L}(\underline{1}) = \{\epsilon\} = \mathcal{L}(G(\underline{1}))$

$G(a) = (\{S_a\}, \Sigma, S_a \to a, S_a)$ $\quad \mathcal{L}(a) = \{a\} = \mathcal{L}(G(a))$

for all $a \in \Sigma$

[NOTE 9: Constructions from the proof of TH 3.32**]**

- $G(r_1 + r_1) = (\{S_{r_1+r_2}\} \cup V_1 \cup V_2, \Sigma\{S_{r_1+r_2} \to S_{r_1}|S_{r_2}\} \cup R_1 \cup R_2, S_{r_1+r_2})$

  Where:
  $G(r_1) = (V_1, \Sigma, R_1, S_{r_1})$
  $G(r_2) = (V_2, \Sigma, R_2, S_{r_2})$

  Provided:
  $V_1 \cap V_2 = \emptyset$
  $S_{r_1+r_2} \notin V_1 \cup V_2$ — can be established by renaming variables

$$\begin{aligned}\mathcal{L}(r_1 + r_2) &= \mathcal{L}(r_1) \cup \mathcal{L}(r_2)\\ &= \mathcal{L}(G(r_1)) \cup \mathcal{L}(G(r_2))\\ &= \mathcal{L}_{G(r_1)}(S_{r_1}) \cup \mathcal{L}_{G(r_2)}(S_{r_2})\\ &= \mathcal{L}_{G(r_1+r_2)}(S_{r_1+r_2})\\ &= \mathcal{L}(G(r_1 + r_2))\end{aligned}$$

- $G(r_1 \cdot r_1) = (\{S_{r_1 \cdot r_2}\} \cup V_1 \cup V_2, \Sigma\{S_{r_1 \cdot r_2} \to S_{r_1}|S_{r_2}\} \cup R_1 \cup R_2, S_{r_1 \cdot r_2})$
  [NOTE 10: should be checked**]**

  Where:
  $G(r_1) = (V_1, \Sigma, R_1, S_{r_1})$
  $G(r_2) = (V_2, \Sigma, R_2, S_{r_2})$

  Provided:
  $V_1 \cap V_2 = \emptyset$
  $S_{r_1+r_2} \notin V_1 \cup V_2$ — can be established by renaming variables

$$\begin{aligned}\mathcal{L}(r_1 \cdot r_2) &= \mathcal{L}(r_1) \cdot \mathcal{L}(r_2)\\ &= \mathcal{L}(G(r_1)) \cdot \mathcal{L}(G(r_2))\\ &= \mathcal{L}_{G(r_1)}(S_{r_1}) \cdot \mathcal{L}_{G(r_2)}(S_{r_2})\\ &= \mathcal{L}_{G(r_1 \cdot r_2)}(S_{r_1 \cdot r_2})\\ &= \mathcal{L}(G(r_1 \cdot r_2))\end{aligned}$$

- $G(r*) = (\{S_{r^*}\} \cup V_1 \cup V_2, \Sigma\{S_{r^*} \to S_{r_1}|S_{r_2} \cup R_1 \cup R_2, S_{r^*})$
  $S_{r^*} \notin V$ — can be established by renaming variables

$\mathcal{L}(r^*) = \mathcal{L}(G(r^*))$
[NOTE 11: see proof of TH 3.32**]**

Exercise 3.9

(Hopcroft, Motwani & Ullman 2001) Consider the context-free grammar G given by the production rules $S \to aS|Sb|a|b$

(a) Prove that no string $w \in \mathcal{L}(G)$ has a substring ba.

(b) Give a description of $\mathcal{L}(G)$ that is independent of G.

(c) Prove that your answer for part (b) is correct.

(a) *To be proven:*
If $S \Rightarrow_G^* x$ Then $\exists_{h,\ell}[h, \ell \geq 0 : x = a^h S b^\ell \vee x = a^{h+1} S b^\ell] \vee x = a^h S b^{\ell+1}$
Proof by induction on the number of steps in the derivation:

Base case:
$S \Rightarrow_G^0 x$
It follows that $x = S = a^0 S b^0$

Induction step: $S \Rightarrow_G^{n+1} x$
If $S \Rightarrow_G^n y$, then $\exists_{h,\ell}[..y..$**[NOTE 12:**$nocluesss$**]**$]$

$S \Rightarrow_G^{n+1} x$ = $\underbrace{S \Rightarrow_G^n y} \Rightarrow_G^1 x$

Due to the induction hypothesis and the fact that from y a production step to x can be made:
$y = a^h S b^\ell$ for some $h, l \geq 0$

Case distinction on production rule applied in the last step:

- $S \to aS$ applied: $x = a^h a S b^\ell = a^{h+1} S b^\ell$
- $S \to bS$ applied: $x = a^h S b b^\ell = a^h S b^{\ell+1}$
- $S \to a$ applied: $x = a^h a b^\ell = a^{h+1} b^\ell$
- $S \to b$ applied: $x = a^h b b^\ell = a^h b^{\ell+1}$

if $w \in \mathcal{L}(G), then S \Rightarrow_G^* w$ and $w \in \{a, b\}^*$
So by the above property $w = a^{h+1} b^\ell$ or $w = a^h b^{\ell+1}$ for some $h, \ell \geq 0$
In neither form w contains the substring ba.

(b) $L = \{a^{h+1} b^\ell | h, \ell \geq 0\} \cup \{a^h b^{\ell+1} | h, l \geq 0\}$
Claim: $L = \mathcal{L}(G)$

(c) Proof of $L = \mathcal{L}(G)$

- $\mathcal{L}(G) \subseteq L$: see (a)

- $L \subseteq \mathcal{L}(G)$: let $w \in L$
  - Assume $w = a^{h+1}b^{\ell}$ for some $h, \ell \geq 0$.
    Then we have the following derivation:

    $S \overset{S\to aS}{\Longrightarrow}{}^{h} a^h S \overset{S\to Sb}{\Longrightarrow}{}^{\ell} a^h S b^{\ell} \overset{S\to a}{\Longrightarrow}{}^{1} a^{h+1}b^{\ell}$
    So $w \in \mathcal{L}(G)$
  - Assume $w = a^{h}b^{\ell+1}$ for some $h, \ell \geq 0$.
    Then we have the following derivation:

    $S \overset{S\to aS}{\Longrightarrow}{}^{h} a^h S \overset{S\to Sb}{\Longrightarrow}{}^{\ell} a^h S b^{\ell} \overset{S\to b}{\Longrightarrow}{}^{1} a^{h}b^{\ell+1}$
    So $w \in \mathcal{L}(G)$

Exercise 3.10

(Hopcroft, Motwani & Ullman 2001) Consider the context-free grammar G given by the production rules

$$S \rightarrow aSbS|bSaS|\epsilon$$

Prove that $\mathcal{L}(G) = \{w \in \{a, b\}|\#_a(w) = \#_b(w)\}$.

$CFGG:\qquad S \rightarrow aSbS|bSaS|\epsilon$

To be proven:
$\mathcal{L}(G) = L = \{w \in \{a, b\}|\#_a(w) = \#_b(w)\}$

Proof:

- $\mathcal{L}(G) \subseteq L$:
  Proof by induction on h
  If $S \Rightarrow^h x$, then $\#_a(x) = \#_b(x)$ for all x, for all $h \geq 0$.
  Base case: ($h = 0$) $S \Rightarrow^0 x$, so $x = S$
  $\#_a(x) = \#_a(S) = 0 = \#_b(S) = \#_b(x)$
  Step case: ($h = \ell + 1$) for some $\ell \geq 0$ if $S \Rightarrow^\ell y$, then $\#_a(y) = \#_b(y)$ for all y *[IH]*
  Case distinction on the last step of derrivation:
  - $S \Rightarrow^\ell uSv \Rightarrow uaSbSv = x$
    By the *[IH]* we have $\#_a(uSv) = \#_b(uSv)$
    ($\#_a(uSv) = \#_b(uSv)$) = ($\#_a(uv) = \#_b(uv)$)
    It follows that $\#_a(x) = \#_a(uaSbSv) = 1 + \#_a(uv) = 1 + \#_b(uv) = \#_b(uaSbSv) = \#_b(x)$
  - $S \Rightarrow^\ell uSv \Rightarrow ubSaSv = x$
    analogous reasoning **[NOTE 13:** okay.**]**
  - $S \Rightarrow^\ell uSv \Rightarrow uv = x$
    $\#_a(x) = \#_a(uv) = \#_b(uv) = \#_b(x)$
- $L \subseteq \mathcal{L}(G)$:
  Proof by structural induction on w (meaning: strong induction on $|w|$):
  if $w \in L$, then $w \in \mathcal{L}(G)$ for all w
  Base case: ($w = \epsilon$) $S \Rightarrow \epsilon = w$, so $w \in \mathcal{L}(G)$
  Step case: ($|w| \geq 2$)
  - $w = aua$: $\qquad w = \underbrace{a\overbrace{u_1}^{\in L}b}_{\in L}\underbrace{u_2a}_{\in L}$

    $|au_1b| < |w|$
    $|u_2a| < |w|$

    By the induction hypothesis the following derivation exists:

    i: $S \Rightarrow^* u_1$
    ii: $S \Rightarrow^* u_2a$

w can be derived as follows:
$S \Rightarrow aSbS \overset{i}{\Rightarrow}{}^* au_1bS \overset{ii}{\Rightarrow}{}^* au_1bu_2a = w$ **[NOTE 14:** using lemma 3.15**]**

- $w = bub$: analogous.
- $w = aub$:
  w can be derived as follows:
  $S \Rightarrow aSbS \overset{i}{\Rightarrow}{}^* aubS \Rightarrow aub = w$ **[NOTE 15:** using lemma 3.15**]**
- $w = bua$: analogous.

Exercise 3.11

A context-free grammar $G = (V, T, R, S,)$ is called *linear* if each production rule is of either of the following two forms: $A \to aB$ or $A \to \epsilon$ for $A, B \in V$, not necessarily different, and $a \in T$.

- Argue that every regular language is generated by a linear context-free grammar.
- Argue that every linear context-free grammar generates a regular language.

- See the proof of TH 3.18 for the DFA to linear CFG transformation and the argument of its correctness.
- Let $G = (V, T, R, S)$ be a linear context free grammar.

  Define $NFA\ N = (Q_n, \Sigma, \to_N, q_0, F_N)$ by

  $$\begin{aligned}&Q_N = V\\&\Sigma = T\\&q_0 = S\\&F_N = \{A \in V | A \to \epsilon \in R\}\\&\to_N = \{(A, a, B) | A \to aB \in R\}\end{aligned}$$

  For $u \in \Sigma^*$ $(= T^*)$ we have

  $S \Rightarrow_G^* uA$ iff $(S, u) \vdash_N^* (A, \epsilon)$

  and

  $S \Rightarrow_G^* u$ iff $(S, u) \vdash_N^* (B, \epsilon)$ for some $B \in F_N$

  (both can be proven by induction)

  It follows that $w \in \mathcal{L}(G)$ iff $w \in \mathcal{L}(N)$ for all $w \in \Sigma^*$

  So $\mathcal{L}(G) = \mathcal{L}(N)$
  and thus $\mathcal{L}(G)$ is a regular language.

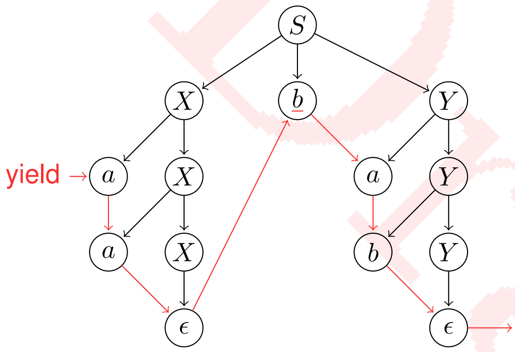
Exercise 3.12

Consider again the the grammar of Exercise 3.5 with production rules

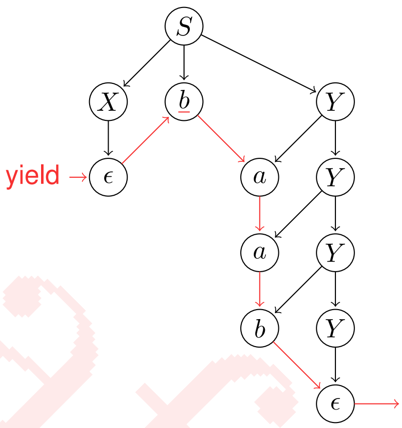
$$S \to XbY$$
$$X \to \epsilon | aX$$
$$Y \to \epsilon | aY | bY$$

Provide parse trees for this grammar with yield $aabab$, $baab$, and $aaabb$. A context-free grammar G is called *ambiguous* if there exist two different complete parse trees PT_1 and PT_2 of G such that yield(PT_1) = yield(PT_2). Otherwise G is called *unambiguous*.

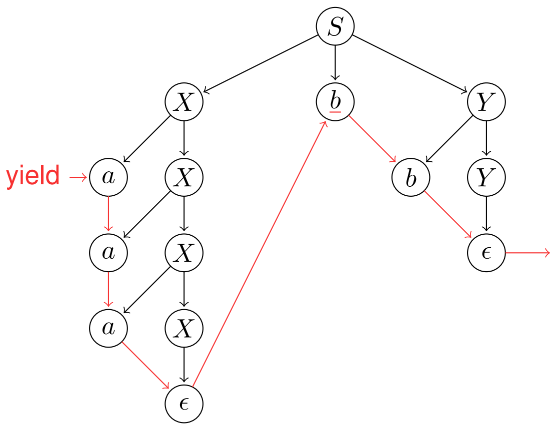
Parse tree for aabab (unique):



Parse tree for baab (unique):



Parse tree for aaabb (unique):



Exercise 3.13

(a) Show that the grammar G given by the production rules

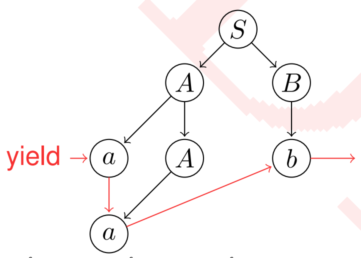
$$S \to AB|aaB \quad A \to a|Aa \quad B \to b$$

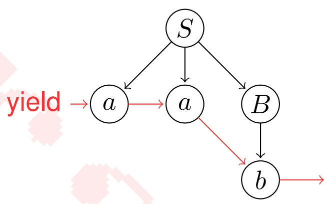
is ambiguous.

(b) Provide an unambiguous grammar G that generates the same language as G. Argue why G is unambiguous and why $\mathcal{L}(G') = \mathcal{L}(G)$.

(a) $S \to \underbrace{AB|aaB}_{\text{Source of ambiguity}} \quad A \to a|Aa \quad B \to b$

String aab has two different complete parse trees:





$S \overset{\ell}{\Rightarrow}_G AB \overset{\ell}{\Rightarrow}_G AB \overset{\ell}{\Rightarrow}_G aAB \overset{\ell}{\Rightarrow}_G aaB \overset{\ell}{\Rightarrow}_G aab$

$S \overset{\ell}{\Rightarrow}_G$

$aaB \overset{\ell}{\Rightarrow}_G aab$

So grammar G is ambiguous.

$\mathcal{L}(G) = \mathcal{L}(a^+b) = \{a^n b | n > 0\}$

(b) G': $\quad S \to AB \quad A \to a|Aa \quad B \to b$

(G' equals G with productionrule $S \to aaB$ removed)

To be proved: $\mathcal{L}(G') = \mathcal{L}(G)$

- $\mathcal{L}(G') \subseteq \mathcal{L}(G)$:
  Every derivation sequence in G' is a derivation sequence in G
- $\mathcal{L}(G) \subseteq \mathcal{L}(G')$:
  Let $w \in \mathcal{L}(G)$, so $S \Rightarrow_G^* w$
  Case distinction on the first step:
  - $S \Rightarrow_G AB \Rightarrow_G^* w$;
    this is a derivation in G' as well, so $w \in \mathcal{L}(G')$
  - $S \Rightarrow_G aaB \Rightarrow_G^* w$;
    it follows that $w = aab$ and $S \Rightarrow_{G'} AB \Rightarrow_{G'} Aab \Rightarrow_{G'} aab$, so $w \in \mathcal{L}(G')$

G' is unambiguous: $a^n b (n > 0)$ has only one parse tree in G'

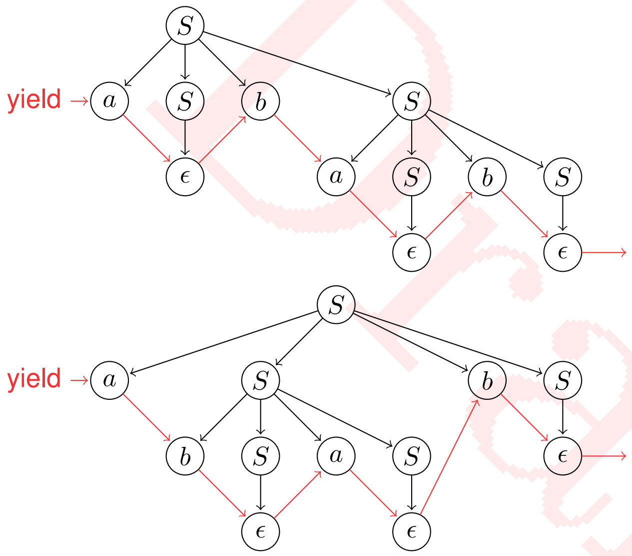
Intermezzo

[NOTE 16: could be added later**]**

Exercise 3.14

> (a) Show that the grammar G given by the production rules
>
> $$S \rightarrow \epsilon | aSbS | bSaS$$
>
> is ambiguous.
>
> (b) Provide an unambiguous grammar G that generates the same language as G. Argue why G is unambiguous and why $\mathcal{L}(G') = \mathcal{L}(G)$.

(a) G is ambiguous since $abab$ has two different (complete) parse trees.



(b) **[NOTE 17:** ermergewd, te lang**]**

Exercise 3.15

(Hopcroft, Motwani & Ullman 2001) Convert the context-free grammar G

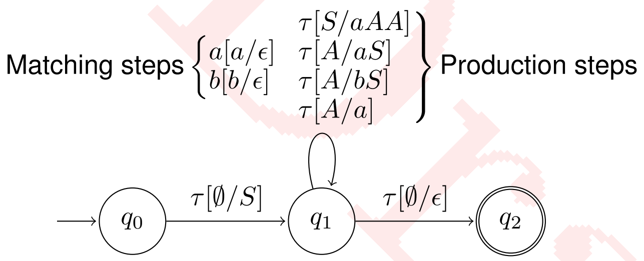
$$S \to aAAA \quad \to aS|bS|a$$

to a PDA P that accepts on empty stack with $\mathcal{N}(P) = \mathcal{L}(G)$.

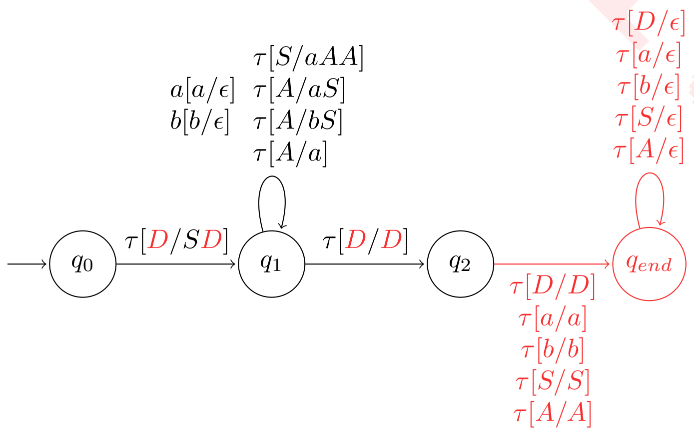
First method:

$G: \quad S \to aAA \quad A \to aS|bS|a$

Transformation to a PDA accepting on final state from the proof of TH 3.25



Transformation to a PDA accepting on empty stack from the proof of TH 3.29



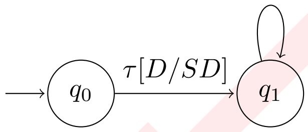
Second method:

$G: \quad S \to aAA \quad A \to aS|bS|a$

Direct ad hoc transformation to a PDA accepting on empty stack from the proof of TH 3.29

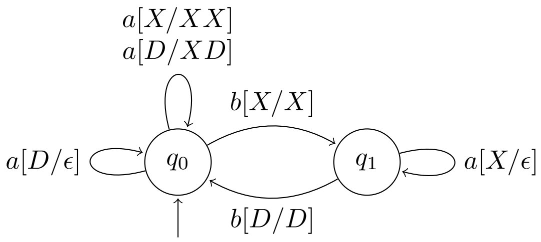
$$\text{Matching steps}\begin{cases} a[a/\epsilon] \\ b[b/\epsilon] \end{cases} \quad \left.\begin{matrix} \tau[S/aAA] \\ \tau[A/aS] \\ \tau[A/bS] \\ \tau[A/\epsilon] \end{matrix}\right\} \text{Production steps}$$

$$\tau[D/\epsilon] \quad \} \text{ Removal of stack bottom symbol}$$



Exercise 3.16

Consider the PDA P accepting on empty stack below.



(a) Construct a context-free grammar G such that $\mathcal{L}(G) = \mathcal{N}(P)$.

(b) Symbol $X \in V \cup T$ is called productive if $X \Rightarrow_{\ G} w$ for some $w \in T$. It follows that a terminal is always productive and that a variable A is productive if there exists a production rule $A \rightarrow_G X_1 X_2 ... X_k$ where all symbols $X_1, X_2, ..., and X_k$ are productive (note that in case k = 0 A is productive). Removing from G all non-productive symbols and all rules that contain non-productive symbols results in a reduced grammar G with $\mathcal{L}(G') = \mathcal{L}(G)$.

Determine all productive symbols in the constructed grammar and give the reduced grammar.

(a) $\mathcal{N}(P) = (\{a^n b a^n b | n \geq 1\})^*$

Transformation to a CFG from the proof of TH 3.30

$$\underbrace{S}_{5} \to \underbrace{[q_0Dq_0]}_{1} \mid [q_0Dq_1]$$

$q_0 \xrightarrow{a[D/XD]} q_0$	$\underbrace{[q_0Dq_0]}_{1} \to \underbrace{a}_{0}[q_0Xq_0]\underbrace{[q_0Dq_0]}_{1} \mid \underbrace{a}_{0}\underbrace{[q_0Xq_1]}_{4}\underbrace{[q_1Dq_0]}_{2}$ $[q_0Dq_1] \to \underbrace{a}_{0}[q_0Xq_0][q_0Dq_1] \mid \underbrace{a}_{0}\underbrace{[q_0Xq_1]}_{4}[q_1Dq_1]$
$q_0 \xrightarrow{a[X/XX]} q_0$	$[q_0Xq_0] \to \underbrace{a}_{0}[q_0Xq_0][q_0Xq_0] \mid \underbrace{a}_{0}\underbrace{[q_0Xq_1]}_{4}[q_1Xq_0]$ $\underbrace{[q_0Xq_1]}_{4} \to \underbrace{a}_{0}[q_0Xq_0]\underbrace{[q_0Xq_1]}_{4} \mid \underbrace{a}_{0}\underbrace{[q_0Xq_1]}_{4}\underbrace{[q_1Xq_1]}_{3}$
$q_0 \xrightarrow{b[X/X]} q_1$	$[q_0Xq_0] \to \underbrace{b}_{0}[q_1Xq_0]$ $\underbrace{[q_0Xq_1]}_{4} \to \underbrace{b}_{0}\underbrace{[q_1Xq_1]}_{3}$
$q_0 \xrightarrow{a[X/\epsilon]} q_1$	$\underbrace{[q_1Xq_1]}_{3} \to \underbrace{a}_{0}$
$q_1 \xrightarrow{b[D/D]} q_0$	$\underbrace{[q_1Dq_0]}_{2} \to \underbrace{b}_{0}\underbrace{[q_0Dq_0]}_{1}$ $[q_1Dq_1] \to \underbrace{b}_{0}[q_0Dq_1]$
$q_0 \xrightarrow{\tau[D/\epsilon]} q_0$	$\underbrace{[q_0Dq_0]}_{1} \to \underbrace{\epsilon}_{0}$

(b) Productive symbols (in order of discovery; see above)

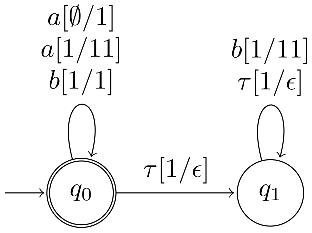
1 $[q_0Dq_0]$ 2 $[q_1Dq_0]$ 3 $[q_1Xq_1]$ 4 $[q_0Xq_1]$ 5 S

Reduced grammar (rules only):

$$
\begin{aligned}
S &\to [q_0Dq_0] \\
[q_0Dq_0] &\to a[q_0Xq_1][q_1Dq_0] \mid \epsilon \\
[q_0Xq_1] &\to a[q_0Xq_1][q_1Xq_1] \mid b[q_1Xq_1] \\
[q_1Xq_1] &\to a \\
[q_1Dq_0] &\to [q_0Dq_0]
\end{aligned}
$$

Exercise 3.17

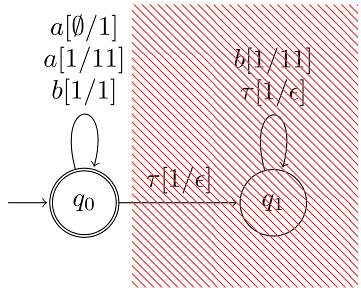
(Hopcroft, Motwani & Ullman, 2001) Consider again the PDA of Exercise 3.1 repeated below.



(a) Construct a context-free grammar that generates the same language as this PDA accepts.

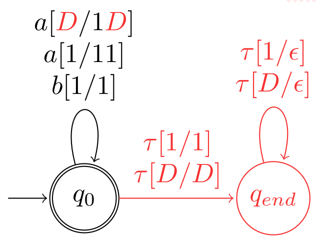
(b) Determine all productive symbols in the constructed grammar and give the reduced grammar (see (b) of the previous question for a definition of productive symbols).

(a) q_1 can be removed. The resulting PDA accepts the same language as P.
This is the case because the accepting state (q_0) is unreachable from q_1



PDA P accepting on final state.
$\mathcal{L}(P)\{aw|w \in \{a,b\}^*\}$

Construction from proof of Th 3.29



ePDA P' accepting on empty stack.
$\mathcal{N}(P') = \mathcal{L}(P)$

Construction from proof of Th 3.30

CFG G
$\mathcal{L}(G) = \mathcal{N}(P') = \mathcal{L}(P)$

$$\underbrace{S}_{5} \rightarrow [q_0 D q_0] \mid \underbrace{[q_0 D q_{and}]}_{3}$$

$q_0 \xrightarrow{a[D/1D]} q_0$	$[q_0 D q_0]$	$\rightarrow \underbrace{a}_{0} [q_0 1 q_0][q_0 D q_0] \mid \underbrace{a}_{0} \underbrace{[q_0 1 q_{end}]}_{4} [q_{end} D q_0]$
	$\underbrace{[q_0 D q_{end}]}_{3}$	$\rightarrow \underbrace{a}_{0} [q_0 1 q_0] \underbrace{[q_0 D q_{end}]}_{3} \mid \underbrace{a}_{0} \underbrace{[q_0 1 q_{and}]}_{4} \underbrace{[q_{end} D q_{end}]}_{1}$
$q_0 \xrightarrow{a[1/11]} q_0$	$[q_0 1 q_0]$	$\rightarrow \underbrace{a}_{0} [q_0 1 q_0][q_0 1 q_0] \mid \underbrace{a}_{0} \underbrace{[q_0 1 q_{end}]}_{4} [q_{end} 1 q_0]$
	$\underbrace{[q_0 1 q_{end}]}_{4}$	$\rightarrow \underbrace{a}_{0} [q_0 1 q_0] \underbrace{[q_0 1 q_{end}]}_{4} \mid \underbrace{a}_{0} \underbrace{[q_0 1 q_{and}]}_{4} \underbrace{[q_{end} 1 q_{end}]}_{2}$
$q_0 \xrightarrow{b[1/1]} q_0$	$[q_0 1 q_0]$	$\rightarrow \underbrace{b}_{0} [q_0 1 q_0]$
	$\underbrace{[q_0 1 q_{end}]}_{4}$	$\rightarrow \underbrace{b}_{0} \underbrace{[q_0 1 q_{end}]}_{4}$
$q_0 \xrightarrow{\tau[1/1]} q_{end}$	$[q_0 1 q_0]$	$\rightarrow [q_{end} 1 q_0]$
	$\underbrace{[q_0 1 q_{end}]}_{4}$	$\rightarrow \underbrace{[q_{end} 1 q_{end}]}_{2}$
$q_0 \xrightarrow{\tau[D/D]} q_{end}$	$[q_0 D q_0]$	$\rightarrow [q_{end} D q_0]$
	$\underbrace{[q_0 D q_{end}]}_{3}$	$\rightarrow \underbrace{[q_{end} D q_{end}]}_{1}$
$q_{end} \xrightarrow{\tau[1/\epsilon]} q_{end}$	$\underbrace{[q_{end} 1 q_{end}]}_{2}$	$\rightarrow \underbrace{\epsilon}_{0}$
$q_{end} \xrightarrow{\tau[D/D]} q_{end}$	$\underbrace{[q_{end} D q_{end}]}_{1}$	$\rightarrow \underbrace{\epsilon}_{0}$

(b) Productive symbols (in order of discovery; see above)

① $[q_{end}Dq_{end}]$ ② $[q_{end}1q_{end}]$ ③ $[q_1Dq_{end}]$ ④ $[q_01q_{end}]$ ⑤ S

Reduced grammar (rules only):

$$
\begin{aligned}
S &\to [q_0Dq_{end}] \\
[q_0Dq_{end}] &\to a[q_01q_{end}][q_{end}Dq_{end}]|[q_{end}Dq_{end}] \\
[q_01q_{end}] &\to a[q_01q_{end}][q_{end}1q_{end}]|b[q_01q_{end}]|[q_{end}1q_{end}]
\end{aligned}
$$

$$
\left.\begin{aligned}
[q_{end}1q_{end}] &\to \epsilon \\
[q_{end}Dq_{end}] &\to \epsilon
\end{aligned}\right\}\text{substitute in other production rules.}
$$

Reduced grammar (rules only), after substitution:

$$
\begin{aligned}
S &\to [q_0Dq_{end}] \\
[q_0Dq_{end}] &\to a[q_01q_{end}]|\epsilon \\
[q_01q_{end}] &\to a[q_01q_{end}]|b[q_01q_{end}]|\epsilon
\end{aligned}
$$

Exercise 3.18

- Show that the class of context-free languages is closed under reversal, i.e. if L is a context-free language then so is $L^R = \{w^R | w \in L\}$.
- Show that the class of context-free languages is not closed under set difference, i.e. if L_1 and L_2 are context-free languages, then $L_1 \backslash L_2 = \{w \in L_1 | w \notin L_2\}$ is not context-free in general.

(a) If L is a context-free language, then L^R is a context-free language

Proof: Let L be a context-free language.

Let $G = (V, \Sigma, R, S)$ be a context-free grammar with $\mathcal{L}(G) = L$

Define $G^R = (V, \Sigma, R^R, S)$,
where $R^R = \{A \to \alpha^R | A \to \alpha \in R\}$

Therefore:

$\beta A \gamma \Rightarrow_G \beta\alpha\gamma$ (rule:$A \to a$)
iff
$(\beta A \gamma)^R = \gamma^R A \beta^R \Rightarrow_{G^R} \gamma^R \alpha^R \beta^R = (\beta\alpha\gamma)^R$ (rule:$A \to \alpha^R$)

We have that:

$\gamma_0 \Rightarrow_G \gamma_1 \Rightarrow_G ... \Rightarrow_G \gamma_{n-1} \Rightarrow_G \gamma_n$
is a derivation (production sequence) for G

$\gamma_0^R \Rightarrow_{G^R} \gamma_1^R \Rightarrow_{G^R} ... \Rightarrow_{G^R} \gamma_{n-1}^R \Rightarrow_{G^R} \gamma_n^R$
is a derivation for G^R

It follows that:

$$\begin{aligned} & w \in L \\ \stackrel{val}{=} & w \in \mathcal{L}(G) \\ \stackrel{val}{=} & S \Rightarrow_G w \\ \stackrel{val}{=} & S^R \Rightarrow_{G^R} w^R \\ \stackrel{val}{=} & S \Rightarrow_{G^R} w^R \\ \stackrel{val}{=} & w^R \in \mathcal{L}(G^R) \end{aligned}$$

So $\mathcal{L}(G^R) = L^R$

(b) $L_1 = \{a^n b^n c^m | n, m \geq 0\}$ is a context-free language

$L_2 = \{w \in \{a, b, c\}^* | \#_b(w) \neq \#_c(w)\}$ is a context-free language
(accepted by a push down automaton)

$L_1 \backslash L_2 = \{a^n b^n c^m | \neg(n \neq m)\}$
$= \{a^n b^n c^n | n \geq 0\}$ is not context-free

Exercise 3.19

Show that the language $L_1 = \{a^{n^2}d|n \geq 0\}$ is not context-free.

(a) Assume L_1 is a context-free language. Let $m > 0$.
Choose $w = a^{m^2}$, then $w \in L_1$ and $|w| = m^2 \geq m$.
Let $uvxyz$ be strings with $w = uvxyz, |vxy| \leq m, vy \neq \epsilon$.

It follows that $v = a^{|v|}, y = a^{|y|}$, thus $1 \leq |vy| \leq m$.
Choose $i = 2$.
Then $uv^2xy^2z = a^{m^2+|vy|} \notin L_1$ since $m^2 < m^2 + |vy| \geq m^2 + m = m(m+1) < (m+1)^2$.

Since the property for context-free languages from the pumping lemma thus not holds ($uv^ixy^iz \notin L_1$), we can conclude that L_1 is therefore not context free.

Exercise 3.20

Show that the language $L_2 = \{ww^R w | w \in \{a, b\}\}$ is not context-free.

(a) Assume L_2 is a context-free language. Let $m > 0$.
Choose $w = a^m b^m b^m a^m a^m b^m = a^m b^{2m} a^{2m} b^m$, then $w \in L_2$ and $|w| = 6m \geq m$.
Let $uvxyz$ be strings with $w = uvxyz, |vxy| \leq m, vy \neq \epsilon$.

We can now apply case distinction, due to $|vxy| \leq m$:

- v and y contain only a's.
  Choose $i = 2$.
  Now either
  - $uv^2xy^2z = a^{m+|vy|}b^{2m}a^{2m}b^m \notin L_2$
  - $uv^2xy^2z = a^m b^{2m} a^{2m+|vy|} b^m \notin L_2$.
- v and y contain both a's and b's.
  Thus $\#_a(vy) = k > 0, \#_b(vy) = l > 0$.
  Choose $i = 0$.
  Now either
  - $uv^0xy^0z = a^{m-k}b^{2m-l}a^{2m}b^m \notin L_2$
  - $uv^0xy^0z = a^m b^{2m-l} a^{2m-k} b^m \notin L_2$
  - $uv^0xy^0z = a^m b^{2m} a^{2m-k} b^{m-l} \notin L_2$

We can thus conclude that L_2 is not a context-free language.

Exercise 3.21

Show that the language $L_3 = \{0^n 1 0^{2n} 1 0^{3n} | n \geq 0\}$ is not context-free.

(a) Assume L_3 is a context-free language. Let $m > 0$.
Choose $w = 0^m 1 0^{2m} 1 0^{3m}$, then $w \in L_3$ and $|w| = 6m + 2 \geq m$.
Let $uvxyz$ be strings with $w = uvxyz, |vxy| \leq m, vy \neq \epsilon$.

We can now apply case distinction, due to $|vxy| \leq m$ and thus vy cannot contain two 1's:

- vy contains one 1.
  Choose $i = 0$:
  uv^0xy^0z now only contains one 1, thus $\notin L_3$.
- vy contains only 0's.
  Choose $i = 0$:
  Now either
  - $uv^0xy^0z = 0^{m-|vy|} 1 0^{2m} 1 0^{3m} \notin L_3$
  - $uv^0xy^0z = 0^{m-|v|} 1 0^{2m-|v|} 1 0^{3m} \notin L_3$
  - $uv^0xy^0z = 0^{m} 1 0^{2m-|vy|} 1 0^{3m} \notin L_3$
  - $uv^0xy^0z = 0^{m} 1 0^{2m-|v|} 1 0^{3m-|y|} \notin L_3$
  - $uv^0xy^0z = 0^{m} 1 0^{2m} 1 0^{3m-|vy|} \notin L_3$

We can thus conclude that L_3 is not a context-free language.

Exercise 3.22

Show that the language $L_4 = \{a^n b^l c^m | n, l \geq m\}$ is not context-free.

(a) Assume L_4 is a context-free language. Let $m > 0$.
Choose $w = a^m b^m c^m$, then $w \in L_3$ and $|w| = 3m \geq m$.
Let $uvxyz$ be strings with $w = uvxyz, |vxy| \leq m, vy \neq \epsilon$.

We can now apply case distinction, due to $|vxy| \leq m$:

- vy contains a's and b's.
  Thus $\#_a(vy) = k, \#_b(vy) = l$, with $k + l \geq 1$.
  Choose $i = 0$:
  $uv^0xy^0z = a^{m-k}b^{m-l}c^m \notin L_4$.
- vy contains b's and c's.
  Thus $\#_b(vy) = k, \#_c(vy) = l$, with $k + l \geq 1$.
  Choose $i = 0$:
  $uv^0xy^0z = a^m b^{m-k}c^{m-l} \notin L_4$.

We can thus conclude that L_4 is not a context-free language.

4 Turing Machines and Computable Functions

Lijst van Notities - ToDo