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

Author
Christiaan Dirkx
Alex Dings

Date
June 9, 2015

Version
0.3

Automata answers explained

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
$\Sigma^n = \{a_1 a_2 ... a_n | \forall i, 1 \leq i \leq n : a_i \in \Sigma\}$

$$\Sigma^* = \bigcup_{n=0}^{\infty} \Sigma^n$$

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. $u_1 v = w$ and $u_2 v = w$ 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

Learning targets chapter 1

At the end of this chapter the student should be able to prove relations and properties on strings and languages using proofs of induction.

Definition 1.1 - Definition of an alphabet

Let Σ be an alphabet. The set Σ^* of strings or finite words of Σ is defined as follows:

- The empty string $\epsilon \in \Sigma^*$
- if $a \in \Sigma$ and $w \in \Sigma^*$ then $aw \in \Sigma^*$

Definition 1.2 - Definition of length of a string

Given an alphabet Σ, the length $|w|$ for a string $w \in \Sigma^*$ is given by: (i) $|\epsilon| = 0$, (ii) $|aw| = |w| + 1$.

Definition 1.4 - Definition of concatenations

Let Σ be an alphabet. The concatenation $wv \in \Sigma^*$ of strings $w, v \in \Sigma^*$ is given by:
(i) $\epsilon v = v$, and (ii) $(aw)v = a(wv)$

A string v is called a prefix of a string w if $vu = w$ for some string u, notation $v \preccurlyeq w$. In the situation that $v \preccurlyeq w$ we occasionaly write $u = w/v$, so if $v \preccurlyeq w$ then we have $v(w/v) = w$.

Definition 1.5

Let Σ be an alphabet and $c \in \Sigma$. The count $\#_c(w)$ of a symbol $c \in \Sigma$ in a string $w \in \Sigma^*$ is given by:

- $\#_c(\epsilon) = 0$;
- $\#_c(cw) = \#_c(w) + 1$;
- $\#_c(aw) = \#_c(w)$ if $a \neq c$

Definition 1.6 - Definition of a language

Let Σ be an alphabet. A subset $L \subseteq \Sigma^*$ is called a language over Σ.

1.8a - Definition of language concatenation

Let $L_1, L_2 \subseteq \Sigma^*$ be two languages over an alphabet Σ. The concatenation $L_1 \cdot L_2$ of L_1 and L_2 is given by:
$L_1 \cdot L_2 = \{w_1 w_2 | w_1 \in L_1, w_2 \in L_2\}$

1.8b - Definition of Kleene-closure

Let $L \subseteq \Sigma$ be a language over an alphabet Σ. The Kleene-closure L^* of L is given by:
$L^* = \{w_1 \cdots w_n | n \geq 0, w_1, \cdots, w_n \in L\}$

Exercise 1.1

> Let Σ be an alphabet.
>
> (a) Prove $|wv| = |w| + |v|$ for all strings w and v.
>
> (b) Prove $\#_a(wv) = \#_a(w) + \#_a(v)$ for every symbol $a \in \Sigma$ and every string $w, v \in \Sigma^*$.

(a) This can be proven using induction. Induction should be performed on one variable only, this case w.

To be proved: $|wv| = |w| + |v|$

Base case: $|w| = 0$, $w = \epsilon$.
For all v we have $|wv| = |\epsilon v| \xlongequal{Def\ 1.4\ i} |v| = 0 + |v| \xlongequal{Def\ 1.2\ i} |\epsilon| + |v| = |w| + |v|$

Def 1.4 i: $\epsilon v = v$

Def 1.2 i: $|\epsilon| = 0$

Inductive step: $|w| = n + 1$.
We have $w = aw'$ for some $a \in \Sigma$ and $w' \in \Sigma^*$.
Assume that for all v we have $|w'v| = |w'| + |v|$ *[IH]*.
For all v we have $|wv| = |(aw')v| \xlongequal{Def\ 1.4\ ii} a(w'v) \xlongequal{Def 1.2 ii} 1 + |w'v| \xlongequal{[IH]} 1 + |w'| + |v| \xlongequal{Def\ 1.2\ ii} |aw'| + |v| = |w| + |v|$.

Def 1.4 ii: $(aw)v = a(wv)$

Def 1.2 ii: $|aw| = |w| + 1$

(b) This can be proven using induction. Induction should be performed on one variable only, this case w.

To be proved: $\#_a(wv) = \#_a(w) + \#_a(v)$

Base case: $|w| = 0$, $w = \epsilon$.
For all v we have $\#_a(wv) = \#_a(\epsilon v) \xlongequal{Def\ 1.4\ i} \#_a(v) = 0 + \#_a(v) \xlongequal{Def\ 1.5\ i} \#_a(\epsilon) + \#_a(v) = \#_a(w) + \#_a(v)$.

Def 1.4 i: $\epsilon v = v$

Def 1.5 i: $\#_c(\epsilon) = 0$

Inductive step: $|w| = n + 1$
We thus have $w = aw'$ for some $a \in \Sigma$ and $w' \in \Sigma^*$.
Assume that for all v we have $\#_a(w'v) = \#_a(w') + \#_a(v)$. *[IH]*
For all v we have
$\#_a(wv) = \#_a((bw')v) \xlongequal{Def 1.2 ii} \#_a(b(w'v))$

if $a = b$
$\xlongequal{Def 1.5 ii} 1 + \#_a(w'v) \xlongequal{[IH]} 1 + \#_a(w') + \#_a(v)$
$\xlongequal{Def 1.5 ii} \#_a(bw') + \#_a(v) = \#_a(w) + \#_a(v)$

Def 1.5 ii: $\#_c(cw) = \#_c(w) + 1$

if $a \neq b$
$\xlongequal{Def 1.5 iii} \#_a(w'v) \xlongequal{[IH]} \#_a(w') + \#_a(v)$
$\xlongequal{Def 1.5 iii} \#_a(bw') + \#_a(v) = \#_a(w) + \#_a(v)$

Def 1.5 iii: $\#_c(aw) = \#_c(w)$, if $a \neq c$

Exercise 1.2

> Let $w_1, w_2, \ldots, w_k$ be k strings, for some $k \geq 0$ over the alphabet Σ, such that $w_1 \preccurlyeq w_2 \preccurlyeq \cdots \preccurlyeq w_k$. Put $w_0 = \epsilon$. Prove $(w_1/w_0)(w_2/w_1)\ldots(w_k/w_{k-1}) = w_k$.

(a) This can be proven using induction on k.

To be proved: $(w_1/w_0)(w_2/w_1)\ldots(w_k/w_{k-1}) = w_k$

Base case: $k = 0$.
Thus $(w_1/w_0)(w_2/w_1)\ldots(w_k/w_{k-1}) \overset{k=0}{=\!=} \epsilon = w_0 = w_k$ — empty statement

Inductive step: $k = l + 1$.
Assume $(w_1/w_0)(w_2/w_1)\ldots(w_l/w_{l-1}) = w_l$ *[IH]*.
We now have: — Intro 2, Uniqueness Properties

$$
\begin{aligned}
& (w_1/w_0)(w_2/w_1)\ldots(w_l/w_{k-1}) \\
\overset{k=l+1}{=\!=} & (w_1/w_0)(w_2/w_1)\ldots(w_l/w_{l-1})(w_l + 1/w_l) \\
\overset{[IH]}{=\!=} & w_l(w_l + 1/w_l) \\
\overset{Intro\ 2\ Uniqueness}{=\!=\!=\!=} & w_{l+1} \\
\overset{k=l+1}{=\!=} & w_k
\end{aligned}
$$

Exercise 1.3

> The reverse w^R of a string w is given by (i) $\epsilon^R = \epsilon$ and (ii) $(aw)^R = (w^R)a$.
> Prove that $(wv)^R = (v^R)(w^R)$ for every two strings w and v.

(a) This can be proven using induction on string w.

To be proved: $(wv)^R = (v^R)(w^R)$

Base case: $w = \epsilon$.
For all v we have
$(wv)^R = (\epsilon v)^R \overset{Def\ 1.4\ i}{=\!=\!=} v^R \overset{Def\ 1.4\ i}{=\!=\!=} \epsilon v^R$

Def 1.4 i: $\epsilon v = v$

$\overset{Def\ \ i}{=\!=\!=} \epsilon^R v^R = w^R v^R$

Inductive step: $|w| = n + 1$,
Thus we have $w = aw'$ for some $a \in \Sigma$ and $w' \in \Sigma^*$.
Assume that for all v we have $(w'v)^R = (v^R)(w'^R)$ *[IH]*.
For all v we have
$(wv)^R = ((aw')v)^R \overset{Def\ 1.4\ ii}{=\!=\!=} (a(w'v))^R \overset{Def\ \ ii}{=\!=\!=} (w'v)^R a$

Def 1.4 ii: $(aw)v = a(wv)$

$\overset{[IH]}{=\!=\!=} v^R (w')^R a \overset{Def\ \ ii}{=\!=\!=} v^R (aw')^R = v^R w^R$

Exercise 1.4

> A full binary tree is a tree where each node has either 0 or 2 children. Prove that a full binary tree with n leaves has at most $2n-1$ nodes (a leaf is a node with 0 children).

(a) This can be proven using structural induction on a tree.

To be proved: a full binary tree with n leaves has at most $2n-1$ nodes.

Base case: A tree with a root having 0 children.
The root is thus also the only leaf. Therefore $n = 1$ leaves and $2n-1 = 2\cdot 1-1 = 1 =$ number of nodes in the tree.

Inductive step: A tree with a root having 2 children.
The root is thus not a leaf of the tree. The two children may or may not have children of their own. We can thus say that the root node has a left subtree and a right subtree, which are full binary trees themselves. The left subtree then has a n_L leaves and the right subtree has n_R leaves.
Now assume that a binary tree with $l, l < n$ leaves has at most $2l-1$ nodes. *[IH]* Note that the left and right subtree both have at least 1 leaf. We can thus say that $n = n_L + n_R$ and $n_L, n_R < n$. According to the *[IH]* we thus have that the left subtree has at most $2n_L - 1$ nodes and the right subtree has at most $2n_R - 1$ nodes. The entire tree thus has at most $1 + 2n_L - 1 + 2n_R - 1 = 2(n_L + n_R) - 1 = 2n - 1$ nodes.

Exercise 1.5

(The Towers of Hanoi) See http://bit.ly/1c4QSUf. Suppose you have three posts and a stack of n different sized disks, initially placed on one post with the largest disk on the bottom and with each disk above it smaller than the disk below. You are to move the disks so they end up all on another post, again in decreasing order of size with the largest disk on the bottom. The only moves you are allowed involve taking the top disk from one post and moving it so that it becomes the top disk on another post, without being put on a smaller disk.

(a) Show that for any n there must be a sequence of moves that does indeed end with all the disks on a post different from the original one in the desired configuration.

(b) How many moves are at least required given an initial stack of n disks in the sequence of moves revealed by your answer to the previous question?

(a) This can be proven using induction on n.

To be proved: for any n there is a sequence of moves that moves all disks to a different post in the correct configuration.

Base case: $n = 1$.
Move the disk to a different post. Due to there being only one disk, the stack is automatically in the correct configuration.

Inductive step: $n = k + 1$.
Assume there is an algorithm A to move k disks from one pole to another pole in the correct configuration. *[IH]*.
To move n disk to another pole, you first use A to move the top k disks of the stack to the auxiliary pole. Then move the last and largest disk to the destination pole. Finally apply A again on the auxiliary pole to move every disk to the destination pole. Since the largest disks is at the bottom of the remaining stack and the top k disks are in the correct configuration due to *[IH]*, the resulting thus also in the correct configuration.

(b) For $n = 1$ the needed moves is 1. For $n > 1$ first all $n - 1$ disks first need to be moved to the auxiliary pole, this takes $moves(n - 1)$ moves, after which the last disk can be moved to the destination pole in 1 move. Then moving all disks from the auxiliary pole to the destination pole takes again $moves(n - 1)$ moves. In total we thus need $moves(n - 1) + 1 + moves(n - 1) = 2 \cdot moves(n - 1) + 1$.

$$moves(n) = \begin{cases} n = 1 & 1 \\ n > 1 & 2 \cdot moves(n - 1) + 1 \end{cases}$$

Writing this out gives $moves(1) = 1, moves(2) = 3, moves(3) = 7$ and $moves(4) = 15$. This gives the suspicion that the recursive definition of moves can be resolved to $moves(n) = 2^n - 1$, as $2^1 - 1 = 1, 2^2 - 1 = 3, 2^3 - 1 = 7$ and $2^n 4 - 1 = 15$.

We can now try to prove this via induction on n.

To be proved: $moves(n) = 2^n - 1$

Base case: $n = 1$.
$moves(1) = 1 = 2^2 - 1$.

Inductive step: $n = k + 1$.
Assume that $moves(k) = 2^k - 1$. *[IH]*.

Now $moves(n) = 2 \cdot moves(n-1) + 1 = 2 \cdot moves(k) + 1 \overset{[IH]}{=\!=} 2 \cdot 2^k - 1 = 2 \cdot 2^{n-1} - 1 = 2^n - 1$.

Exercise 1.6

Let Σ be an alphabet.

(a) Calculate the language concatenations $\{ab, bcd\} \cdot \{e, ef\}$ and $\{a\}^* \cdot \{bb\}^*$.

(b) Prove that $\{\epsilon\} \cdot L = L \cdot \{\epsilon\} = L$ for every language $L \subseteq \Sigma^*$.

(c) Prove that $\emptyset \cdot L = L \cdot \emptyset = \emptyset$ and $\emptyset^* = \{\epsilon\}$.

(d) Give a counterexample for $(L_1 \cdot L_2)^* = L_1^* \cdot L_2^*$ for two languages $L_1, L_2 \subseteq \Sigma^*$.

(a) $\{ab, bcd\} \cdot \{e, ef\} = \{abe, abef, bcde, bcdef\}$

$\{a\}^* \cdot \{bb\}^* \xlongequal{Def 1.8b} \{a^k | k \geq 0\} \cdot \{(bb)^l | l \geq 0\} = \{a^k(bb)^l | k \geq 0, l \geq 0\} = \{\epsilon, a, aa, aaa, \ldots, bb, abb, aabb, aaabb, \ldots, bbbb, abbbb, \ldots\}$

Def 1.8 b: $L^* = \{w_1, \ldots, w_n | n \geq 0, w_1, \ldots w_n \in L\}$

Def 1.8 a: $L_1 \cdot L_2 = \{w_1 w_2 | w_1 \in L_1, w_2 \in L_2\}$

Def 1.4 i: $\epsilon v = v$

(b)

$$\begin{aligned}
\{\epsilon\} \cdot L &\xlongequal{Def\ 1.8\ a} \{uv | u \in \{\epsilon\} \wedge v \in L\} \\
&= \{\epsilon v | v \in L\} \\
&\xlongequal{Def\ 1.4\ 1} \{v | v \in L\} \\
&= L \\
&= \{u | u \in L\} \\
&\xlongequal{Def\ 1.4\ i} \{u\epsilon | v \in L\} \\
&= \{uv | u \in L \wedge u \in \{\epsilon\}\} \\
&\xlongequal{Def\ 1.8\ a} L \cdot \{\epsilon\}
\end{aligned}$$

Def 1.8 a: $L_1 \cdot L_2 = \{w_1 w_2 | w_1 \in L_1, w_2 \in L_2\}$

(c)

$$\begin{aligned}
\{\emptyset\} \cdot L &\xlongequal{Def\ 1.8\ a} \{uv | u \in \{\emptyset\} \wedge v \in L\} \\
&= \{uv | False \wedge v \in L\} \\
&= \{uv | False\} \\
&= \emptyset \\
&= \{uv | False\} \\
&= \{uv | u \in L \wedge False\} \\
&= \{uv | u \in L \wedge v \in \{\emptyset\}\} \\
&\xlongequal{Def\ 1.8\ a} L \cdot \{\emptyset\}
\end{aligned}$$

[NOTE 1: $\emptyset^* = \{\epsilon\}$]

(d) Take $L_1 = \{a\}$ and $L_2 = \{b\}$.
$L_1 \cdot L_2 = \{ab\}$, and $(L_1 \cdot L_2)^* = \{(ab)^k | k \geq 0\}$.
$L_1^* = \{a^l | l \geq 0\}, L_2^* = \{b^m | m \geq 0\}$ and $L_1^* \cdot L_2^* = \{a^l b^m | l, m \geq 0\}$.

Now $abab \in \{(ab)^k | k \geq 0\}$, but $abab \notin \{a^l b^m | l, m \geq 0\}$

Exercise 1.7

The shuffle $w \parallel v$ of two strings $w, v \in \Sigma^*$ yealds a set of strings, and is given by:

(i) $\epsilon \parallel \epsilon = \{\epsilon\}$

(ii) $w \parallel \epsilon = \{w\}$

(iii) $\epsilon \parallel v = \{v\}$

(iv) $aw' \parallel bv' = \{a\} \cdot (w' \parallel bv') \cup \{b\} \cdot (aw' \parallel v')$

(a) Calculate $aa \parallel bb$.

(b) Prove $w \parallel v = v \parallel w$

(c) Does it always hold that $wv, vw \in w \parallel v$?

(a)

$$
\begin{aligned}
aa \parallel bb &\overset{Def\ iv}{=\!=\!=} \{a\} \cdot (a \parallel bb) \cup \{b\} \cdot (aa \parallel b) \\
&\overset{2xDef\ iv}{=\!=\!=} \{a\} \cdot \{a\} \cdot (\epsilon \parallel bb) \cup \{a\} \cdot \{b\} \cdot (a \parallel b) \cup \{b\} \cdot \{a\} \cdot (a \parallel b) \cup \{b\} \cdot \{b\} \cdot (aa \parallel \epsilon) \\
&\overset{Def\ ii,iii}{=\!=\!=} \{a\} \cdot \{a\} \cdot \{bb\} \cup \{a\} \cdot \{b\} \cdot (a \parallel b) \cup \{b\} \cdot \{a\} \cdot (a \parallel b) \cup \{b\} \cdot \{b\} \cdot \{aa\} \\
&\overset{2xDef\ iv}{=\!=\!=} \{a\} \cdot \{a\} \cdot \{bb\} \cup \{a\} \cdot \{b\} \cdot \{a\} \cdot (\epsilon \parallel b) \cup \{a\} \cdot \{b\} \cdot \{b\} \cdot (a \parallel \epsilon) \cup \\
&\qquad \{b\} \cdot \{a\} \cdot \{a\} \cdot (\epsilon \parallel b) \cup \{b\} \cdot \{a\} \cdot \{b\} \cdot (a \parallel \epsilon) \cup \{b\} \cdot \{b\} \cdot \{aa\} \\
&\overset{Def\ ii,iii}{=\!=\!=} \{a\} \cdot \{a\} \cdot \{bb\} \cup \{a\} \cdot \{b\} \cdot \{a\} \cdot \{b\} \cup \{a\} \cdot \{b\} \cdot \{b\} \cdot \{a\} \cup \\
&\qquad \{b\} \cdot \{a\} \cdot \{a\} \cdot \{b\} \cup \{b\} \cdot \{a\} \cdot \{b\} \cdot \{a\} \cup \{b\} \cdot \{b\} \cdot \{aa\} \\
&= \{aabb\} \cup \{abab\} \cup \{abba\} \cup \{baab\} \cup \{baba\} \cup \{bbaa\} \\
&= \{aabb, abab, abba, baab, baba, bbaa\}
\end{aligned}
$$

(b) This can be proven using mathematical induction on the combined length of the two strings, |w| + |v|.

To be proved: $w \parallel v = v \parallel w$

Base case: $|w| + |v| = 0, sow = \epsilon, v = \epsilon$.
$w \parallel v = \epsilon \parallel \epsilon = v \parallel w$

Inductive step: $|w| + |v| > 0$

case distinction:

- $w = \epsilon$: $w \parallel v = \epsilon \parallel v \overset{Def\ iii}{=\!=\!=} \{v\} \overset{Def\ ii}{=\!=\!=} v \parallel \epsilon = v \parallel w$
- $v = \epsilon$: $w \parallel v = w \parallel \epsilon \overset{Def\ ii}{=\!=\!=} \{w\} \overset{Def\ iii}{=\!=\!=} \epsilon \parallel w = v \parallel w$
- $w \neq \epsilon, v \neq \epsilon$, so $w = aw', v = bv'$

  Assume $s \parallel t = t \parallel s$, for all $s, t |s| + |t| < |w| + |v|$ *[IH]*.

Assuming $w' \parallel v' = v' \parallel w'$ is also correct, but forces you to extract two characters instead of one.

$w \parallel v = (aw') \parallel (bv') \xlongequal{Def\ iv} \{a\} \cdot (w' \parallel (bv')) \cup \{b\} \cdot ((aw') \parallel y)$
$\xlongequal{[IH]} \{a\} \cdot ((bv') \parallel w') \cup \{b\} \cdot (y \parallel (aw'))$
$= \{b\} \cdot (y \parallel (aw')) \cup \{a\} \cdot ((bv') \parallel w') \xlongequal{Def\ iv} (vv' \parallel aw') = v \parallel w$

|w'| + |bv'|, |aw'| + |v'|< |w| + |v|

(c) Since $wv \in w \parallel v = vw \in v \parallel w$, and we have just proven $w \parallel v = v \parallel w$. We thus only need to prove $wv \in w \parallel v$. This can be proven using mathematical induction on the combined length of the two strings, |w| + |v|.

To be proved: $wv \in w \parallel v$.

Base case: $|w| + |v| = 0$, so $w = \epsilon, v = \epsilon$.
$wv = \epsilon \cdot \epsilon \xlongequal{Def\ 1.4\ i} \epsilon \in \{\epsilon\} \xlongequal{Def\ i} \epsilon \parallel \epsilon = w \parallel v$

Def 1.4 i: $\epsilon v = v$

Inductive step: $|w| + |v| > 0$

case distinction:

- $w = \epsilon$: $wv = \epsilon v \xlongequal{Def1.4i} v \in \{v\} \xlongequal{Def\ iii} \epsilon \parallel v = w \parallel v$
- $v = \epsilon$: $wv = w\epsilon \xlongequal{Def1.4i} w \in \{w\} \xlongequal{Def\ ii} w \parallel \epsilon = w \parallel v$
- $w \neq \epsilon, v \neq \epsilon$, so $w = aw', v = bv'$

  Assume $st \in s \parallel t$, for all $s, t|s| + |t| < |w| + |v|$ *[IH]*.

  $wv = aw'bv' \in \{aw'bv'\} = \{a\} \cdot \{w'bv'\} \overset{[IH]}{\subseteq} \{a\} \cdot (w' \parallel (bv'))$
  $\overset{Def\ i}{\subseteq} (aw')||(bv') = w \parallel v$

Assuming $w'v' \in w' \parallel v'$ is also correct, but forces you to extract two characters instead of one.

|w'| + |bv'|< |w| + |v|

2 Finite Automata and Regular Languages

Learning targets chapter 2

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

- Construct a DFA from a language.
- Construct a DFA from the union of two other DFAs.
- Derive a DFA from an NFA.
- Prove with pathsets that a language is accepted by a DFA.
- Construct a NFA from a language language or regular expression.
- Construct a regular expression from a language or DFA.
- Prove that a language is regular.
- Prove that a language is not regular.
- Prove that the class of regular languages is closed under a property.

Definition 2.1 - Definition of a Deterministic Finite Automaton

A DFA is a tuple $D = (Q, \Sigma, \delta, q_0, , F)$ with Q a finite set of states, Σ a finite alphabet, $\delta : Q \times \Sigma \to Q$ the transition *function*, $q_0 \in Q$ the initial state, and $F \subseteq Q$ the set of final states.

Definition of $\vdash_D$:
$(q, w) \vdash_D (q', w') \Leftrightarrow w = aw'$ and $\delta(q, a) = q'$ for some $a \in \Sigma$

Lemma 2.3 - basic properties of a DFA

Let D be a DFA, then:

(i) For all states q, q', q'' and words w, w' it holds that:

$$\text{if } q, w \vdash_D^* (q', w') \text{ and } (q, w) \vdash_D^* (q'', w') \text{ then } q' = q''$$

(ii) For states q, q' and all words w, w', v it holds that:

$$(q, w) \vdash_D{}^* (q', w') \Leftrightarrow (q, wv) \vdash_D{}^* (q', w'v)$$

Definition 2.4 - Definition of the language defined by a DFA

Let $D = (Q, \Sigma, \delta, q_0, F)$ be a finite automaton. The language $\mathcal{L}(D) \subseteq \Sigma^*$ accepted by D is defined by:

$$\mathcal{L}(D) = \{w \in \Sigma^* | \exists q \in F : (q_0, w) \vdash_D^* (q, \epsilon)\}$$

Definition 2.7 - Definiton of an NFA

(Non deterministic finite automaton with silent steps). An NFA is a quintuple $N = (Q, \Sigma, \to_N, q_0, F)$ with Q a finite set of stats, Σ a finite alphabet, $\to_N \subseteq Q \times (\Sigma \cup \{\tau\}) \times Q$ the transion *relation*, $q_0 \in Q$ the initial state, and $F \subseteq Q$ the set of final states.
Important to note that $\to_N$ is a relation and not a function, meaning there may be several transitions possible from a certain state for a given letter, or none at all for a certain letter.

Basic definition of the $\vdash_N$ yield relation:

$$(q, w) \vdash_N (q', w') \Leftrightarrow \exists a \in \Sigma : q \xrightarrow{a}_N q' \wedge w = aw' \text{ or } q \xrightarrow{\tau}_N q' \wedge w = w'$$

Lemma 2.8 - Consistency property

Lemma stating a consistency property of NFA's for all words w, w', v and states q, q':

$$(q, w) \vdash_N^* (q', w') \Leftrightarrow (q, wv) \vdash_N^* (q', w'v)$$

Definition 2.9 - Definiton of the language defined by an NFA

Let $N = (Q, \Sigma, \rightarrow_N, q_0, F)$ be a finite automaton. The language $\mathcal{L}(N)$ accepted by N is defined by:

$$\mathcal{L}(N) = \{w \in \Sigma^* | \exists q \in F : (q_0, w) \vdash_N^* (q, \epsilon)\}$$

Theorem 2.12

Theorem: If a language $L \subseteq \Sigma^*$ is accepted by a DFA, then L is also accepted by some NFA.
The proof of this is trivial, since any state or function rule of a DFA is also valid in an NFA.

Theorem 2.13

Theorem: If a language $L \subseteq \Sigma^*$ is accepted by an NFA, then L is also accepted by a DFA.
[NOTE 2: Proof]

[NOTE 3: missing Theorem 2.25 and 2.26]

Theorem 2.27 - Pumping Lemma for regular languages

This theorem can be used to prove that a language is not regular.

Details:
Let L be a regular language over an alphabet $\sum$. There exists a constant $m > 0$ such that each $w \in L$ with $|w| > m$ can be written as $w = xyz$ where $x, y, z \in \sum^*, y \neq \epsilon, |xy| \leq m$, and for all $k > 0$: $xy^kz \in L$.

Theorem 2.30

Let L be a regular language over an alphabet $\sum$ represented by an NFA N accepting L. Then it can be decided if $L = \emptyset$ or not.

Theorem 2.31

With this theorem you can test whether a string is in a language or not.

Details:

Let $L \subseteq \sum^*$ be a regular language over the alphabet $\sum$, represented by an NFA N accepting L, and let $w \in \sum^*$ be a string over $\sum$. Then it can be decided if $w \in L$ or not.

How to prove this:

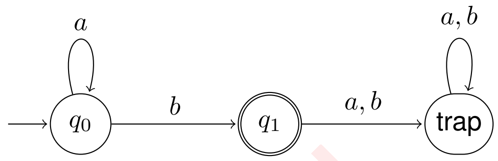
Construct, using the algorithm given in the proof of Theorem 2.13, a DFA D such that $\mathcal{L}(D) = \mathcal{L}(N)$. Simulate D starting from its initial state on input w, say

$(q_0, w) \vdash_D^* (q', \epsilon)$ for some state q' of D. $w \in L$ if q' is a final state of D, $w \notin L$ otherwise.

Exercise 2.1

> Construct a DFA D_1 with alphabet $\{a, b\}$ (with no more than three states) for the language $L_1 = \{a^n b | n \geq 0\}$ and prove with the help of pathsets that $\mathcal{L}(D_1) = L_1$.

(a) DFA D_1



state	**path set**
q_0	$\{a^n \| n \geq 0\}$
q_1	$\{a^n b \| n \geq 0\}$
trap	$(\{a^n \| n \geq 0\} \cup \{a^n b \| n \geq 0\})^C$

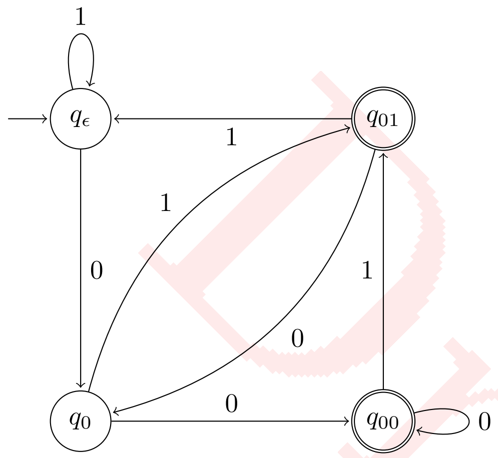
Only (q_2) is an accepting state, thus only the language $\{a^n b | n \geq 0\}$ is accepted. Thus $\mathcal{L}(D_1) = \{a^n b | n \geq 0\} = L_1$.

Exercise 2.2

> Construct a DFA D_1 with alphabet $\{0,1\}$ (with no more than four states) for the language $L_2 = \{w \in \{0,1\}^* |$ the second last element of w is $0\}$ and prove with the help of pathsets that $\mathcal{L}(D_2) = L_2$.

(a) Second last element, thus formally $L_2 = \{u0a | u \in \{0,1\}^* \wedge a \in \{0,1\}\}$, or $\{w \in \{0,1\}^* | w \succcurlyeq 00$ or $01\}$

DFA D_2



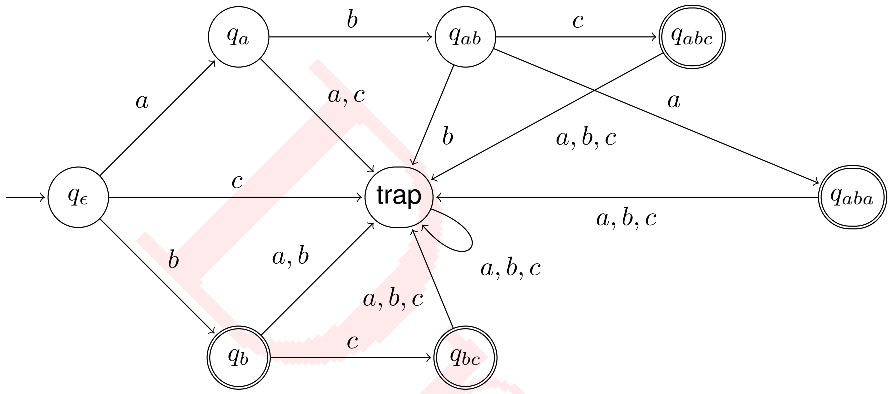
state	path set
q_ϵ	$\{w \in \{0,1\}^* \| w \succcurlyeq \epsilon$
q_0	$\{w \in \{0,1\}^* \| w \succcurlyeq 0$
q_{00}	$\{w \in \{0,1\}^* \| w \succcurlyeq 00$
q_{01}	$\{w \in \{0,1\}^* \| w \succcurlyeq 01$

Both (q_{00}) and (q_{01}) are accepting states, thus the only accepted languages are $\{w \in \{0,1\}^* | w \succcurlyeq 00\}$ and $\{w \in \{0,1\}^* | w \succcurlyeq 01\}$, thus $\{w \in \{0,1\}^* | w \succcurlyeq 00$ or $01\}$ is accepted. This means $\mathcal{L}(D_2) = \{w \in \{0,1\}^* | w \succcurlyeq 00$ or $01\} = L_2$.

Exercise 2.3

> (a) Construct a DFA for the language $\{aba, abc, bc, b\}$ over the alphabet $\{a, b, c\}$.
>
> (b) If $L \subseteq \{a, b, c\}^*$ is finite, does there exist a DFA D such that $\mathcal{L}(D) = L$?

(a) DFA D



state	path set
q_ϵ	$\{w \in \{a, b, c\}^* \mid w = \epsilon\}$
q_a	$\{w \in \{a, b, c\}^* \mid w = a\}$
q_b	$\{w \in \{a, b, c\}^* \mid w = b\}$
q_{bc}	$\{w \in \{a, b, c\}^* \mid w = bc\}$
q_{ab}	$\{w \in \{a, b, c\}^* \mid w = ab\}$
q_{abc}	$\{w \in \{a, b, c\}^* \mid w = abc\}$
q_{aba}	$\{w \in \{a, b, c\}^* \mid w = aba\}$
trap	$\{w \in \{a, b, c\}^* \mid w \neq \epsilon, a, b, bc, ab, abc, aba\}$

(b) **[NOTE 4:** TODO**]**

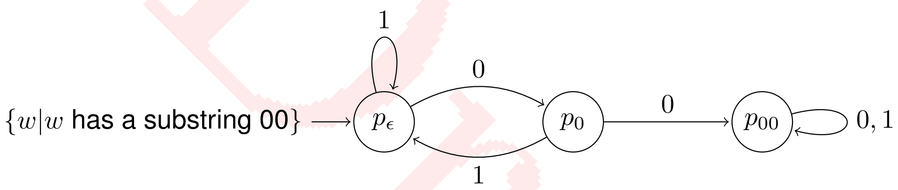
Exercise 2.4

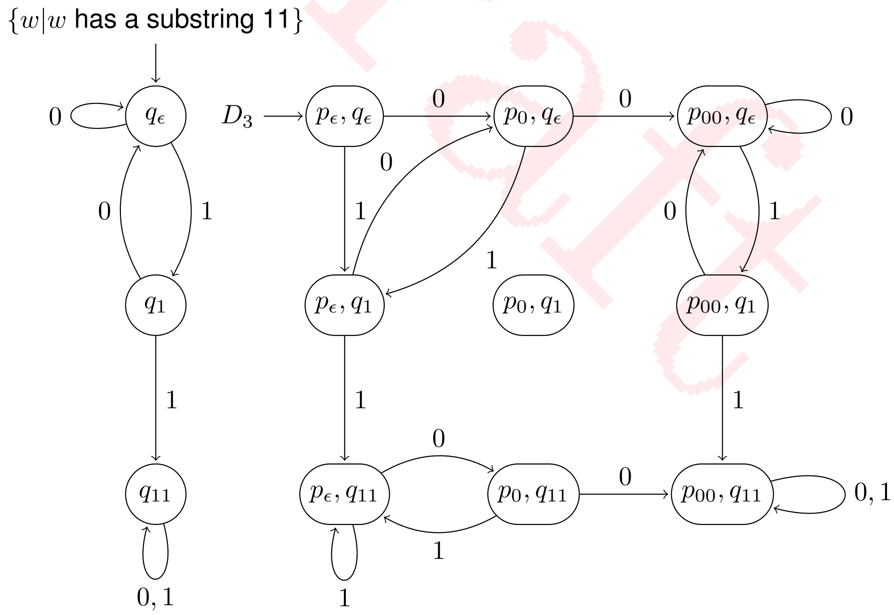
> (a) Construct a DFA D_3 with alphabet $\{0, 1\}$ (with no more than eight states) for the language $L_3 = \{w \in \{0, 1\}^* | w \text{ contains substring } 00 \text{ and } 11\}$ and prove that $\mathcal{L}(D_3) = L_3$.
>
> (b) Construct a DFA D_4 with alphabet $\{0, 1, 2\}$ (with no more than eight states) for the language $L_4 = \{w \in \{0, 1, 2\}^* | w \text{ contains substring } 00 \text{ and } 11\}$ and prove that $\mathcal{L}(D_4) = L_4$.

(a) The DFA should accept the langage $\{w \in \{0, 1\}^* | w*$ has a substring 00 and a substring 11$\}$. We can therefor say that it should accept $\{w | w$ has a substring 00$\} \cap \{w | w$ has a substring 11$\}$.

We can thus derive a DFA as follows:

DFA D

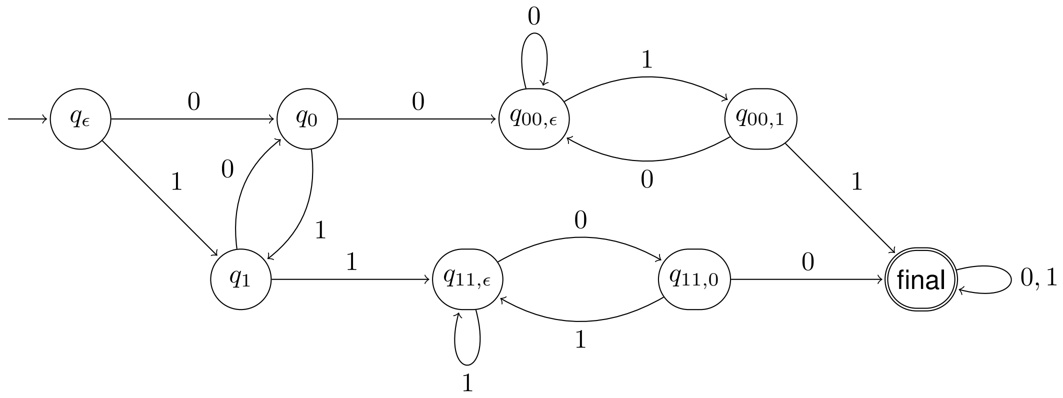




See exercise *(2.7)* for formal proof of the correctness of this construction technique.

State (p_0, q_1) can be removed as it is not reachable from the initial state. This results in the following DFA as a solution:

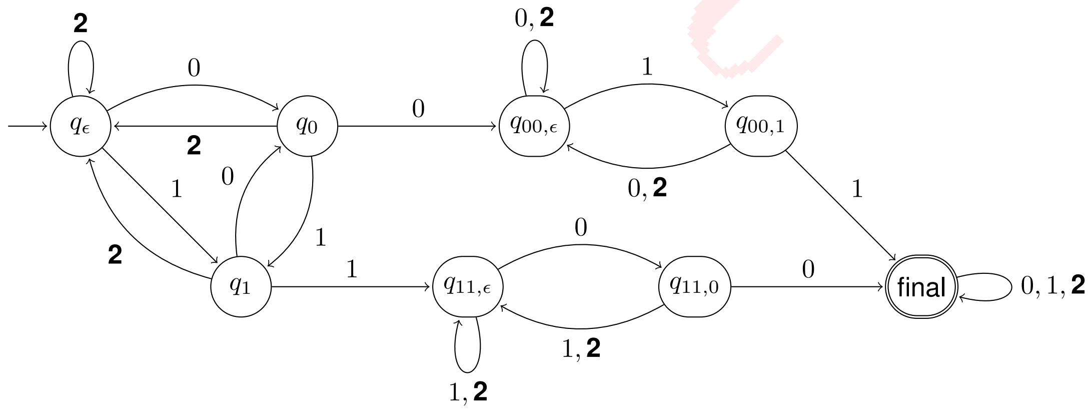
DFA D_3



state	path set
q_ϵ	$\{w \in \{0,1\}^* \| w = \epsilon\}$
q_0	$\{w \in \{0,1\}^* \| w$ does not contain substrings 00 and 11, has as suffix $0\}$
q_1	$\{w \in \{0,1\}^* \| w$ does not contain substrings 00 and 11, has as suffix $1\}$
q_{00}	$\{w \in \{0,1\}^* \| w$ contains substring 00, but not 11, has as suffix $0\}$
q_{11}	$\{w \in \{0,1\}^* \| w$ contains substring 11, but not 00, has as suffix $1\}$
$q_{00,1}$	$\{w \in \{0,1\}^* \| w$ contains substring 00, but not 11, has as suffix $1\}$
$q_{11,0}$	$\{w \in \{0,1\}^* \| w$ contains substring 11, but not 00, has as suffix $0\}$
final	$\{w \in \{0,1\}^* \| w$ contains substring 00 and $11\}$

Only (final) is an accepting state, thus only the language $\{w \in \{0,1\}^* | w$ contains substring 00 and $11\}$ is accepted. Thus $\mathcal{L}(D_3) = \{w \in \{0,1\}^* | w$ contains substring 00 and $11\} = L_3$.

(b) DFA D_4

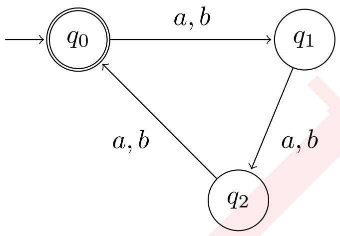


See exercise *(a)* for similar derivation and formal proof.

Exercise 2.5

> Construct DFAs D_5 and D_6 accepting the following languages:
>
> (a) $L_5 = \{w \in \{a,b\}^* | \ |w| \text{ mod } 3 = 0\}$
>
> (b) $L_6 = \{w \in \{0,1\}^* | w \text{ as binary number is divisible by } 3\}$

(a) DFA D_5



state	path set
q_0	$\{w \in \{a,b\}^* \| \|w\| \text{ mod } 3 = 0\}$
q_1	$\{w \in \{a,b\}^* \| \|w\| \text{ mod } 3 = 1\}$
q_2	$\{w \in \{a,b\}^* \| \|w\| \text{ mod } 3 = 2\}$

Only (q_0) is an accepting state, thus only the language $\{w \in \{a,b\}^* | \ |w| \text{ mod } 3 = 0\}$ is accepted. Thus $\mathcal{L}(D_5) = \{w \in \{a,b\}^* | \ |w| \text{ mod } 3 = 0\} = L_5$.

(b) First let us define the value of a string $w \in \{0,1\}^*, w$ as binary number.

$value : \{0,1\}^* \mapsto \mathbb{N}$

(i) $value(\epsilon) = 0$

(ii) $value(w \cdot 0) = 2 \cdot value(w)$

(iii) $value(w \cdot 1) = 2 \cdot value(w) + 1$

The language L_6 can now be rewritten to $\{w \in \{0,1\}^* | value(w) \text{ mod } 3 = 0\}$. From this we can conclude that D_6 has 3 states with the following pathset:

state	path set
q_0	$\{w \in \{a,b\}^* \| value(w) \text{ mod } 3 = 0\}$
q_1	$\{w \in \{a,b\}^* \| value(w) \text{ mod } 3 = 1\}$
q_2	$\{w \in \{a,b\}^* \| value(w) \text{ mod } 3 = 2\}$

Since $value(\epsilon) \text{ mod } 3 = 0 \text{ mod } 3 = 0$, ($q_0$) is both the initial and accepting state.

We can now formally caluclate all transitions:

(q_0) $value(w) \text{ mod } 3 = 0$, thus $value(w) = 3 \cdot k$, for some $k \in \mathbb{N}$

$value(w \cdot 0) \text{ mod } 3 = (2 \cdot value(w)) \text{ mod } 3 = (2 \cdot 3 \cdot k) \text{ mod } 3 == (6 \cdot k) \text{ mod } 3 = 0$.
Thus $\delta(q_0, 0) = q_0$.

$value(w \cdot 1)$ mod $3 = (2 \cdot value(w)+1)$ mod $3 = (2 \cdot 3 \cdot k+1)$ mod $3 = (6 \cdot k+1)$ mod $3 = 1$.
Thus $\delta(q_0, 1) = q_1$.

(q_1) $value(w)$ mod $3 = 1$, thus $value(w) = 3 \cdot k + 1$, for some $k \in \mathbb{N}$

$value(w \cdot 0)$ mod $3 = (2 \cdot value(w))$ mod $3 = (2 \cdot (3 \cdot k+1))$ mod $3 = (6 \cdot k+2)$ mod $3 = 2$.
Thus $\delta(q_1, 0) = q_2$.

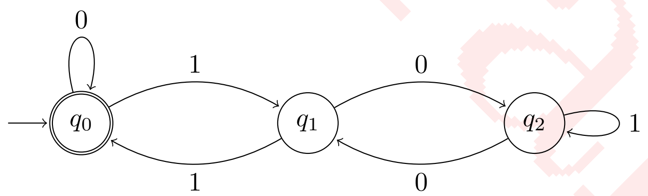
$value(w \cdot 1)$ mod $3 = (2 \cdot value(w) + 1)$ mod $3 = (2 \cdot (3 \cdot k + 1) + 1)$ mod $3 = (6 \cdot k + 3)$ mod $3 = 0$.
Thus $\delta(q_1, 1) = q_0$.

(q_2) $value(w)$ mod $3 = 2$, thus $value(w) = 3 \cdot k + 2$, for some $k \in \mathbb{N}$

$value(w \cdot 0)$ mod $3 = (2 \cdot value(w))$ mod $3 = (2 \cdot (3 \cdot k+2))$ mod $3 = (6 \cdot k+4)$ mod $3 = 1$.
Thus $\delta(q_2, 0) = q_1$.

$value(w \cdot 1)$ mod $3 = (2 \cdot value(w) + 1)$ mod $3 = (2 \cdot (3 \cdot k + 2) + 1)$ mod $3 = (6 \cdot k + 5)$ mod $3 = 2$.
Thus $\delta(q_2, 1) = q_2$.

Now that we have all states and transitions, we can construct the final DFA:
DFA D_6



Exercise 2.6

Suppose a language $L \subseteq \Sigma^*$ is accepted by a DFA D. Construct a DFA D^C that accepts the language $L^C = \{w \in \Sigma^* | w \notin L\}$.

(a) *Given :*
$L \subseteq \Sigma^*$
$D \overset{Def\ 2.1}{=\!=} (Q, \Sigma, \delta, q_0, F)$
$\mathcal{L}(D) = L$

Now we can define D^C as $(Q, \Sigma, \delta, q_0, Q \setminus F)$.

$$
\begin{aligned}
&\mathcal{L}(D^C) \\
\overset{Def\ 2.4}{=\!=}&\{w \in \Sigma^* | \exists_{q \in Q \setminus F}[(q_0, w) \vdash^*_{D^C} (q, \epsilon)]\} \\
\overset{\delta}{=}&\{w \in \Sigma^* | \exists_{q \in Q \setminus F}[(q_0, w) \vdash^*_{D} (q, \epsilon)]\} \\
=&\{w \in \Sigma^* | \neg\exists_{q \in F}[(q_0, w) \vdash^*_{D} (q, \epsilon)]\} \\
=&\{w \in \Sigma^* | \exists_{q \in F}[(q_0, w) \vdash^*_{D} (q, \epsilon)]\}^C \\
\overset{Def\ 2.4}{=\!=}&\mathcal{L}(D)^C \\
=&L^C
\end{aligned}
$$

Exercise 2.7

> Let D_1 and D_2 be two DFAs, say $D_i = (Q_i, \Sigma, \delta_i, q_0^i, F_i)$for $i \in \{1, 2\}$.
>
> (a) Give a DFA D with set of states $Q_1 \times Q_2$ and alphabet Σ such that $\mathcal{L}(D) = \mathcal{L}(D_1) \cap \mathcal{L}(D_2)$.
>
> (b) Prove by induction on the length of a string w that $((q_1, q_2), w) \vdash_D^n ((q_1', q_2'), w') \Leftrightarrow (q_1, w) \vdash_D^n (q_1', w') \wedge (q_2, w) \vdash_D^n (q_2', w')$
>
> (c) Conclude that indeed $\mathcal{L}(D) = \mathcal{L}(D_1) \cap \mathcal{L}(D_2)$.

(a) $D = (Q_1 \times Q_2, \Sigma, \delta_{1,2}, (q_0^1, q_0^2), F_1 \times F_2)$
where $\delta_{1,2}((q_1, q_2), a) = (\delta_1(q_1, a), \delta_2(q_2, a)), (q_1, q_2) \in Q_1 \times Q_2$, and $a \in \Sigma$.

(b) This can be proven using mathematical induction on the length of string w, n.
[NOTE 5: Definitions?**]**

To be proved: $((q_1, q_2), w) \vdash_D^n ((q_1', q_2'), w') \Leftrightarrow (q_1, w) \vdash_D^n (q_1', w') \wedge (q_2, w) \vdash_D^n (q_2', w')$.

Base case: $n = 0$, $w = \epsilon$.

$$
\begin{aligned}
&((q_1, q_2), w) \vdash_D^0 ((q_1', q_2'), w') \\
\overset{val}{=\!=}\, & q_1 = q_1' \wedge q_2 = q_2' \wedge w = w' \\
\overset{val}{=\!=}\, & (q_1, w) \vdash_{D_1}^0 (q_1', w') \wedge (q_2, w) \vdash_{D_2}^0 (q_2', w')
\end{aligned}
$$

Inductive step: $n = k + 1$.
Assume $((q_1, q_2), w) \vdash_D^k ((q_1', q_2'), w') \Leftrightarrow (q_1, w) \vdash_D^k (q_1', w') \wedge (q_2, w) \vdash_D^k (q_2', w')$. *[IH]*

$$
\begin{aligned}
&((q_1, q_2), w) \vdash_D^{k+1} ((q_1', q_2'), w') \\
=\, & ((q_1, q_2), w) \vdash_D^1 ((q_1'', q_2''), w'') \wedge ((q_1'', q_2''), w'') \vdash_D^k ((q_1', q_2'), w') \\
& \text{for some } q_1'', q_2'', w'' \\
\overset{\delta,[IH]}{=\!=}\, & \delta((q_1, q_2), a) = (q_1'', q_2'') \wedge (q_1'', w'') \vdash_{D_1}^k (q_1', w') \wedge (q_2'', w'') \vdash_{D_2}^k (q_2', w') \\
& \text{for some } q_1'', q_2'', w'', a \text{ with } w = aw'' \\
\overset{\delta}{=\!=}\, & \delta_1(q_1, a) = q_1'' \wedge \delta_2(q_2, a) = q_2'' \wedge (q_1'', w'') \vdash_{D_1}^k (q_1', w') \wedge (q_2'', w'') \vdash_{D_2}^k (q_2', w') \\
& \text{for some } q_1'', q_2'', w'', a \text{ with } w = aw'' \\
=\, & (q1, w) \vdash_{D_1}^1 (q_1'', w'') \wedge (q_1'', w'') \vdash_{D_1}^k (q_1', w') \wedge (q_2, w) \vdash_{D_2}^1 (q_2'', w'') \wedge (q_2'', w'') \vdash_{D_2}^k (q_2', w') \\
& \text{for some } q_1'', q_2'', w'' \\
=\, & (q_1, w) \vdash_{D_1}^{k+1} (q_1', w') \wedge (q_2, w) \vdash_{D_2}^{k+1} (q_2', w') \\
=\, & (q_1, w) \vdash_{D_1}^n (q_1', w') \wedge (q_2, w) \vdash_{D_2}^n (q_2', w')
\end{aligned}
$$

(c)

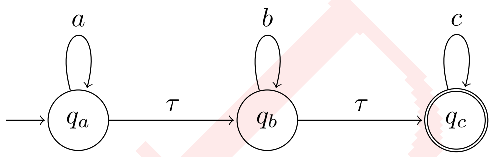
$$
\begin{aligned}
&\mathcal{L}(D)\\
\xlongequal{Def\ 2.4}&\{w \in \Sigma^* | \exists_{(f_1,f_2)\in F_1\times F_2}[((q_0^1, q_p^2), w) \vdash_D^* ((f_1, f_2), \epsilon)]\}\\
=&\{w \in \Sigma^* | \exists_{f_1 \in F_1}[(q_0^1, w) \vdash_D^* (f_1, \epsilon)] \wedge \exists_{f_2 \in F_2}[(q_0^2, w) \vdash_D^* (f_2, \epsilon)]\}\\
=&\{w \in \Sigma^* | \exists_{f_1 \in F_1}[(q_0^1, w) \vdash_D^* (f_1, \epsilon)]\} \cap \{w \in \Sigma^* | \exists_{f_2 \in F_2}[(q_0^2, w) \vdash_D^* (f_2, \epsilon)]\}\\
\xlongequal{Def\ 2.4}&\mathcal{L}(D_1) \cap \mathcal{L}(D_2)
\end{aligned}
$$

Exercise 2.8

> Consider the alphabet $\Sigma = \{a, b, c\}$
>
> (a) Construct an NFA that accepts the language $L = \{a^n b^m c^l | n, m, l \geq 0\}$ and has no more than three states.
>
> (b) Derive a DFA accepting L from the NFA constructed in part (a).

(a) $\mathcal{L}(D) = \{a^n b^m c^l | n, m, l \geq 0\} = \{a\}^* \cdot \{b\}^* \cdot \{c\}^* = \mathcal{L}(a^* b^* c^*)$ (regular expression)

NFA N

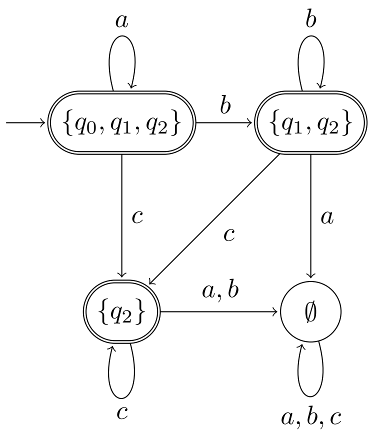


(b) A DFA D can be constructed from an NFA according to Theorem *2.13*.

DFA state	a		b		c	
	$\vdash^a_N$	E	$\vdash^b_N$	E	$\vdash^c_N$	E
$\{q_0, q_1, q_2\}$	$\{q_0\}$	$\{q_0, q_1, q_2\}$	$\{q_1\}$	$\{q_1, q_2\}$	$\{q_2\}$	$\{q_2\}$
$\{q_1, q_2\}$	$\emptyset$	$\emptyset$	$\{q_1\}$	$\{q_1, q_2\}$	$\{q_2\}$	$\{q_2\}$
$\{q_2\}$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\{q_2\}$	$\{q_2\}$
$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$

This leads to the following DFA:

DFA D



Exercise 2.9

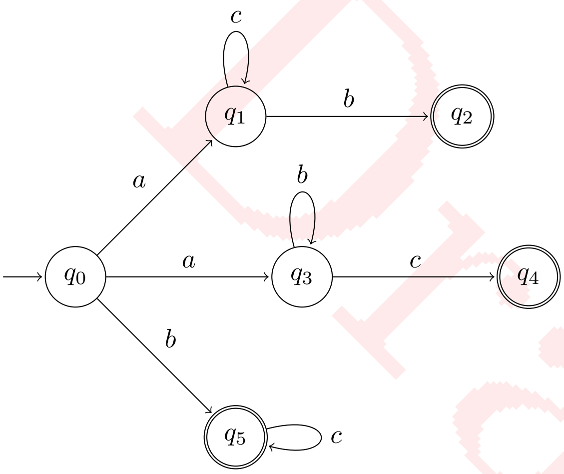
Consider the alphabet $\Sigma = \{a, b, c\}$

(a) Construct a single NFA that accepts a string $w \in \Sigma^*$ if

w is of the form $ac^n b$ for some $n \geq 0$, or
w is of the form $ab^m c$ for some $m \geq 0$, or
w is of the form bc^l for some $l \geq 0$

(b) Derive a DFA accepting L from the NFA constructed in part (a).

(a) NFA N

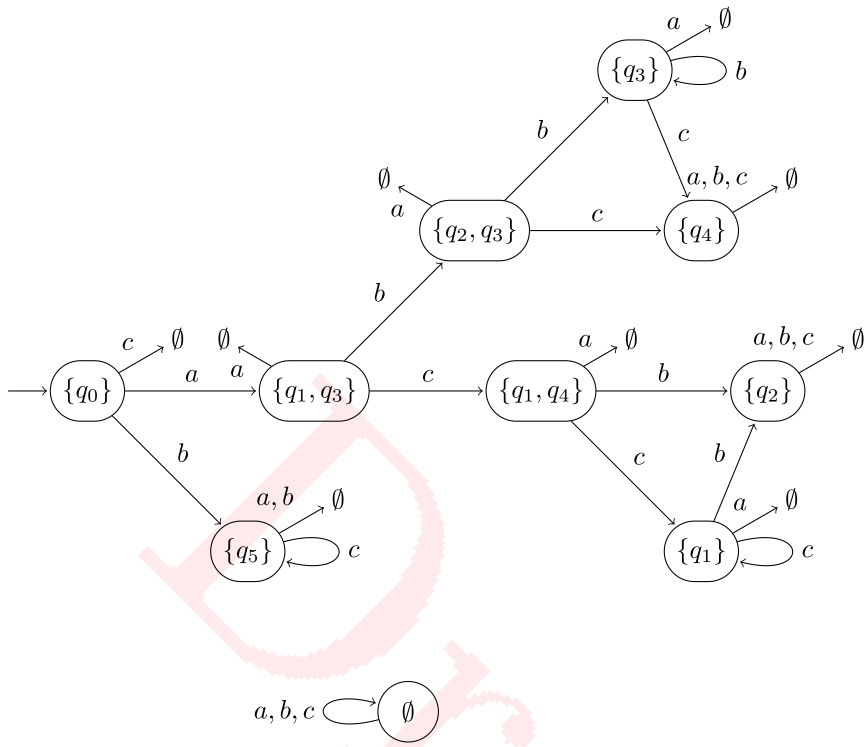


(b) A DFA D can be constructed from an NFA according to Theorem *2.13*. The NFA does not contain any τ-transitions, therefore the ϵ-closure is superfluous and yields no extra states. Unreachable states are omitted.

DFA state	$\vdash^a_N$	$\vdash^b_N$	$\vdash^c_N$
$\{q_0\}$	$\{q_1, q_3\}$	$\{q_5\}$	$\emptyset$
$\{q_1, q_3\}$	$\emptyset$	$\{q_2, q3\}$	$\{q_1, q_4\}$
$\{q_5\}$	$\emptyset$	$\emptyset$	$\{q_5\}$
$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$
$\{q_2, q_3\}$	$\emptyset$	$\{q_3\}$	$\{q_5\}$
$\{q_1, q_4\}$	$\emptyset$	$\{q_2\}$	$\{q_5\}$
$\{q_3\}$	$\emptyset$	$\{q_3\}$	$\{q_5\}$
$\{q_4\}$	$\emptyset$	$\emptyset$	$\emptyset$
$\{q_2\}$	$\emptyset$	$\emptyset$	$\emptyset$
$\{q_1\}$	$\emptyset$	$\{q_2\}$	$\{q_1\}$

This leads to the following DFA:

DFA D

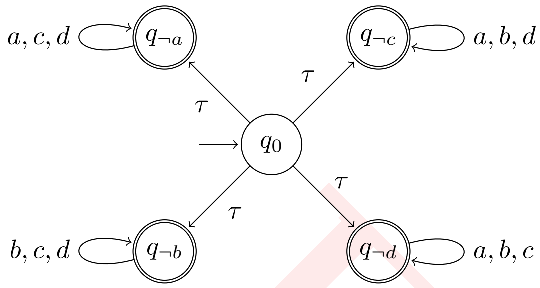


a $\emptyset$
$\{q_3\}$
b
b
c
$\emptyset$
a
$\{q_2, q_3\}$
c
a, b, c $\emptyset$
$\{q_4\}$
b
c $\emptyset$
$\emptyset$
a $\emptyset$
a, b, c $\emptyset$
$\{q_0\}$
a
a
$\{q_1, q_3\}$
c
$\{q_1, q_4\}$
b
$\{q_2\}$
b
c
b
a, b $\emptyset$
a $\emptyset$
$\{q_5\}$
c
$\{q_1\}$
c
a, b, c
$\emptyset$

Exercise 2.10

Give an automaton over the alphabet $\{a, b, c, d\}$ that accepts all strings in which at least one symbol of the alphabet does not occur.

(a) NFA N



Exercise 2.11

Suppose the language $L \subseteq \Sigma^*$ is regular.

(a) Show that $L \setminus \{\epsilon\}$ is regular.

(b) Let $w \in Sigma^*$ be an arbitrary string. Prove that $L \cap \{w\}$ is regular.

(a) **[NOTE 6:** TODO**]**

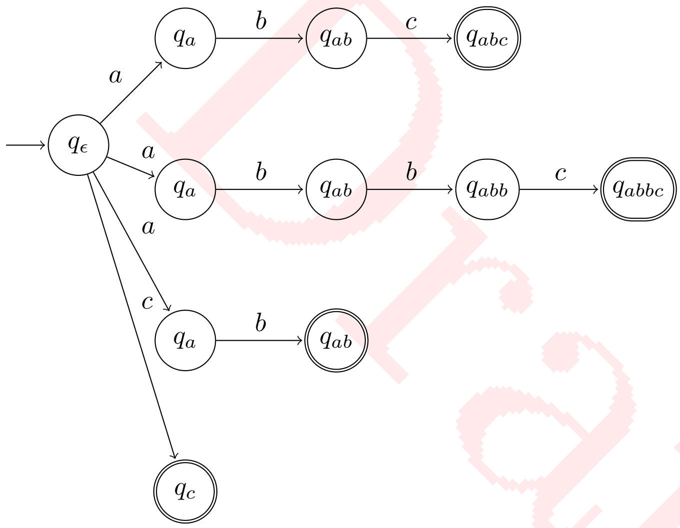
(b) **[NOTE 7:** TODO**]**

Exercise 2.12

> (a) Prove that the language $L = \{abc, abbc, ab, c\}$ is regular.
>
> (b) Construct a DFA accepting L.
>
> (c) Prove that every finite language over some alphabet Σ is regular.

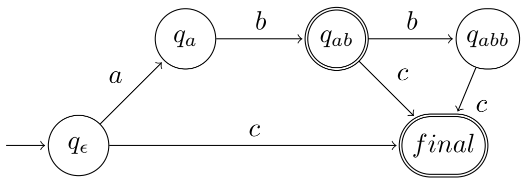
(a) A language is regular if there exists an DFA, NFA or regular expression (equivalents) which accepts it. We can thus prove L to be regular by constructing a NFA N which accepts it:

NFA N



Which can be reduced to:

NFA N



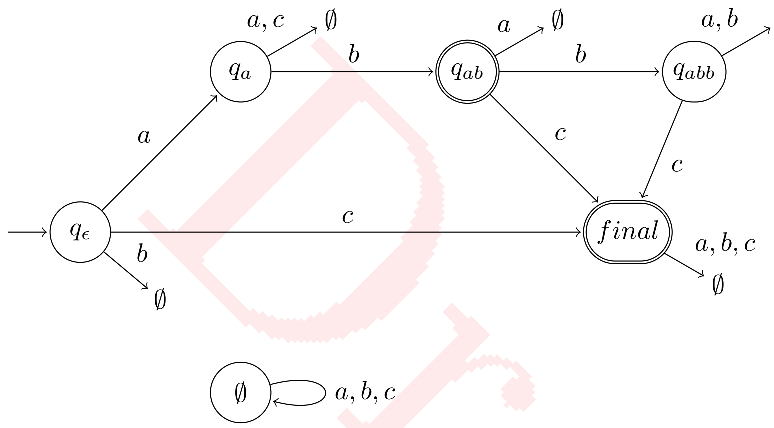
Proof of the NFA accepting L can now be done using pathsets, which is left as an exercise for the reader.

(b) A DFA D can be constructed from NFA N according to Theorem *2.13*. The NFA does not contain any τ-transitions, therefore the ϵ-closure is superfluous and yields no extra states. Unreachable states are omitted.

This leads to the following DFA:

DFA D

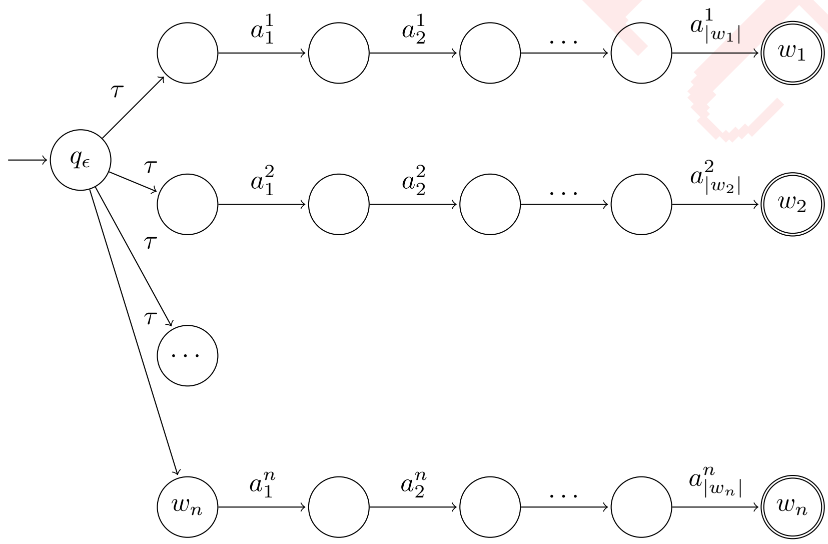
DFA state	$\vdash^a_N$	$\vdash^b_N$	$\vdash^c_N$
$\{q_e\}$	$\{q_a\}$	$\emptyset$	$final$
$\{q_a\}$	$\emptyset$	$\{q_{ab}\}$	$\emptyset$
$\{q_{ab}\}$	$\emptyset$	$\{q_{ab}\}$	$final$
$\{q_{abb}\}$	$\emptyset$	$\emptyset$	$final$
$final$	$\emptyset$	$\emptyset$	$\emptyset$
$\emptyset$	$\emptyset$	$\emptyset$	$\emptyset$



(c) L a finite language : $L = \{w_1, w_2, \ldots, w_n\}$ with $w_i = a^i_1 \cdot a^i_2 \ldots a^i_{|w_i|}, i = 1, 2, \ldots, n$

We can thus easily construct a NFA N that accept L, thereby proving L to be regular.

NFA N



Exercise 2.13

(Sipser 1997) For the language of each of the following regular expression over the alphabet $\{a, b\}$, give two strings in the language and two strings not in the language.

(a) a^*b^*

(b) $a(ba)^*b$

(c) $(a+b)^*a(a+b)^*b(a+b)^*a(a+b)^*$

(d) $(1+a)b$

(a) $L_a = \mathcal{L}(a^*b^*)$
included : $\epsilon, a, b, ab, aabb$

excluded : $abab, ba$

(b) $L_b = \mathcal{L}(a(ab)^*) = \mathcal{L}(ab(ab)^*) = \mathcal{L}(ab^+)$
included : $ab, abab, ababab$

excluded : ϵ, a, b, ba

(c) $L_c = \mathcal{L}(a+b)^*a(a+b)^*b(a+b)^*a(a+b)^*)$
included : $aba, abaabbabaab$

excluded : $\epsilon, a, ab, abb, bbbbbb$

(d) $L_c = \mathcal{L}((1+a)b) = \mathcal{L}(1+a) \cdot \mathcal{L}(b) = (\mathcal{L}(1) \cup \mathcal{L}(a)) \cdot \{b\} = (\{\epsilon\} \cup \{a\}) \cdot \{b\} = \{b, ab\}$
included : a, ab

excluded : ϵ, a, aa, bb

Exercise 2.14

Provide a regular expression for each of the following languages.

(a) $\{w \in \{a,b\}^* | w$ starts with a and ends in $b\}$

(b) $\{w \in \{a,b,c\}^* | w$ contains at most two a's and at least one $b\}$

(c) $\{w \in \{a,b,c\}^* | \; |w| \leq 3\}$

(a) $\{w \in \{a,b\}^* | w$ starts with a and ends in $b\} = \{aub | u \in \{a,b\}^*\} = \{a\} \cdot \{u | u \in \{a,b\}^*\} \cdot \{b\} = \mathcal{L}(a) \cdot \mathcal{L}((a+b)^*) \cdot \mathcal{L}(b) = a(a+b)^*b$.

(b) at most two a's : $(b+c)^* \cdot (1+a) \cdot (b+c)^* \cdot (1+a) \cdot (b+c)^*$.
at least one b : $(a+b+b)^* b(a+b+c)^*$

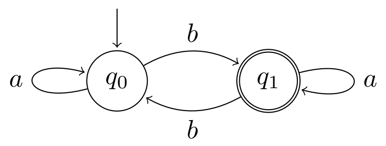
[NOTE 8: TODO**]**

(c) $(1+a+b)(1+a+b)(1+a+b) = (1+a+b)^3$

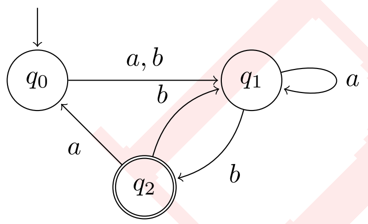
Exercise 2.15

(Sipser 1997) Construct regular expressions for the languages accepted by the following DFAs:

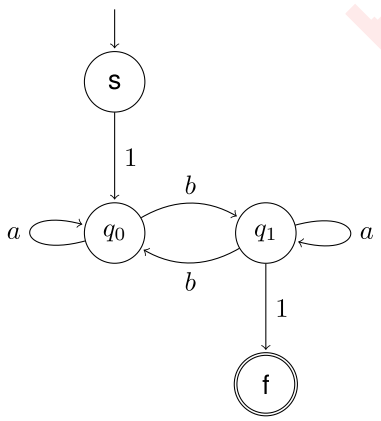
(a)



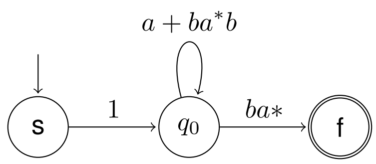
(b)

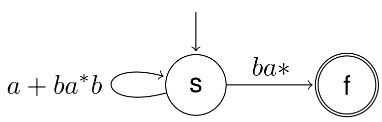


(a)



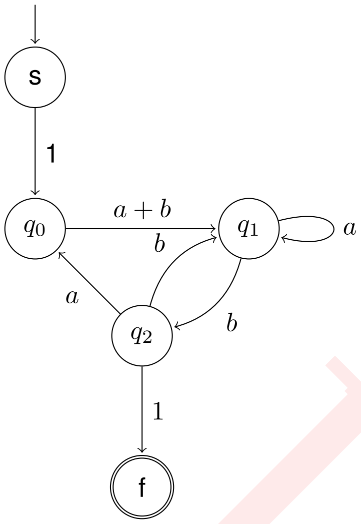
Can be transfomed into:



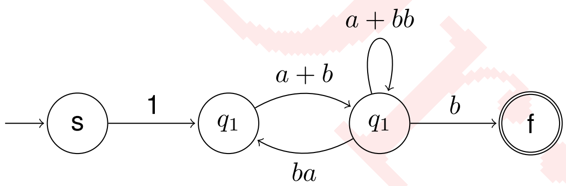


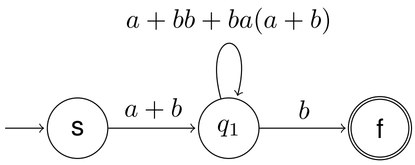
We can thus conclude that this automaton accepts language $(a + ba^*b)^*ba^*$.

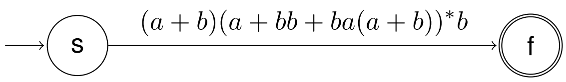
(b)



Can be transfomed into:







Exercise 2.16

Guess a regular expression for each of the following languages. Next provide a DFA for each language and construct a regular expression via elimination of states.

(a) $\{w \in \{a,b\}^* |$ in w, each maximal substring of a's of length 2 or more is followed by a symbol $b\}$

(b) $\{w \in \{a,b\}^* | w$ has no substring $bab\}$

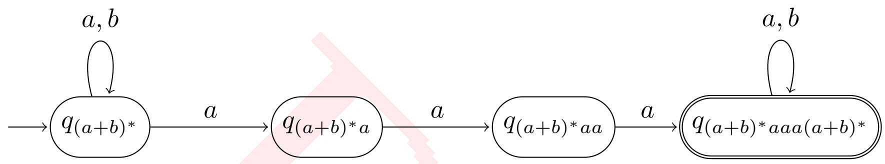
(c) $\{w \in \{a,b\}^* | \#_a(w) = \#_b(w) \wedge$ if $v \preccurlyeq w$ then $-2 \leq \#_a(v) - \#_b(v) \leq 2\}$

(a) **[NOTE 9:** TODO**]**

Exercise 2.17

> (Sipser 1997) Convert the following regular expression to an equivalent NFA:
>
> (a) $(a + b)^*aaa(a + b)^*$
>
> (b) $(((aa) * bb) + ab)^*$

(a) **[NOTE 10:** Official construction**]**



(b) **[NOTE 11:** TODO**]**

Exercise 2.18

> Prove that the following languages are not regular
>
> (a) $\{a^k b a^k | k \geq 0\}$
>
> (b) $\{a^k b^l | k > l > 0\}$
>
> (c) $\{a^k b^l c^{k+l} | k, l \geq 0\}$

(a) Assume L is a regular language. Let $m > 0$.
Choose $w = a^m b a^m$, then $w \in L$ and $|w| = 2m + 1 \geq m$.
Let xyz be strings with $w = xyz, |xy| \leq m, y \neq \epsilon$.

It follows that xy consist of a's only.
Thus $\#_a(y) = l, 1 \leq l \leq m$, since $|xy| \leq m, y \neq \epsilon$.
Choose $i = 0$.
Now we have $xy^0z = a^{m-l}ba^m \notin L$, since $m - 1 \neq m, l \geq 1$.

any $i \neq 1$ will do

Since the property for regular languages from the pumping lemma thus not holds ($xy^iz \notin L$), we can conclude that L is therefore not regular.

(b) Assume L is a regular language. Let $m > 0$.
Choose $w = a^{m+1} b^m$, then $w \in L$ and $|w| = 2m + 1 \geq m$.
Let xyz be strings with $w = xyz, |xy| \leq m, y \neq \epsilon$.

It follows that xy consist of a's only.
Thus $\#_a(y) = l, 1 \leq l \leq m$, since $|xy| \leq m, y \neq \epsilon$.
Choose $i = 0$.
Now we have $xy^0z = a^{m+1-l}ba^m \notin L$, since $m + 1 - l \not> m, l \geq 1$.

only $i = 0$ will work

We can thus conclude that L is not a regular language.

(c) Assume L is a regular language. Let $m > 0$.
Choose $w = a^m c^m$, then $w \in L(k = m, l = 0)$ and $|w| = 2m \geq m$.
Let xyz be strings with $w = xyz, |xy| \leq m, y \neq \epsilon$.

It follows that xy consist of a's only.
Thus $\#_a(y) = l, 1 \leq l \leq m$, since $|xy| \leq m, y \neq \epsilon$.
Choose $i = 0$.
Now we have $xy^0z = a^{m-l}c^m \notin L$, since $m - 1 \neq m, l \geq 1$.

any $i \neq 1$ will do

We can thus conclude that L is not a regular language.

Exercise 2.19

Prove that the language $\{uu^R | u \in \{0,1\}^*\}$ is not regular.

(a) Assume L is a regular language. Let $m > 0$.
Choose $w = 0^m110^m$, then $w \in L$ and $|w| = 4m \geq m$.
Let xyz be strings with $w = xyz, |xy| \leq m, y \neq \epsilon$.

It follows that xy consist of 0's only.
Thus $\#_0(y) = l, 1 \leq l \leq m$, since $|xy| \leq m, y \neq \epsilon$.
Choose $i = 0$.
Now we have $xy^0z = 0^{m-l}110^m \notin L$, since $(0^{m-l}1)^R \neq (10^m), l \geq 1$.

any $i \neq 1$ will do

We can thus conclude that L is not a regular language.

Exercise 2.20

Prove that the language $\{a^n | n \text{ is prime}\}$ is not regular.

(a) **[NOTE 12:** TODO**]**

Exercise 2.21

> (a) Prove by induction on m that $m < 2^m$ for $m \geq 0$.
>
> (b) Prove that the language $\{a^n | n = 2^k \text{ for some } k \geq 0\}$ is not regular.

(a) *To be proved:* $m < 2^m$

Base case: $m = 0$.
$m = 0 < 1 = 2^0 = 2^m$

Inductive step: $m = n + 1$
Assume that for all $n, n \geq 0$, we have $n < 2^n$. *[IH]*
We now have:
$m = n + 1 \overset{[IH]}{<} 2^n + 1 \overset{2^k \geq 1}{\leq} 2^k + 2^k = 2^{k+1} = 2^m$.

(b) Assume L is a regular language. Let $m > 0$.
Choose $w = a^{2^m}$, then $w \in L$ and $|w| = 2^m \overset{(a)}{\geq} m$.
Let xyz be strings with $w = xyz, |xy| \leq m, y \neq \epsilon$.

It follows that xy consist of 0's only.
Thus $\#_0(y) = l, 1 \leq l \leq m$, since $|xy| \leq m, y \neq \epsilon$.
Choose $i = 2$.
Now we have $xy^2z = a^{2^m+l} \notin L$, since $2^m < 2^m + l \leq 2^m + m < 2^m + 2^m = 2^{m+1}$.

We can thus conclude that L is not a regular language.

Exercise 2.22

> Prove that the following languages are not regular.
>
> (a) $\{u \in \{0,1\}^* | \#_0(u) = \#_1(u)\}$
>
> (b) $\{u \in \{0,1\}^* | \#_0(u) \neq \#_1(u)\}$

To be used: $\{0^n 1^n | n \geq 0\}$ is not regular, See exercise 2.

(a) Assume L is a regular language.
The regular expression $\mathcal{L}(0^*1^*)$ is regular according to Theorem 2.25. We thus have that $\{0^k 1^l | k, l \geq 0\}$ is regular. From Theorem 2.26 it then follows that $L \cap \mathcal{L}(0^*1^*)$ is regular.
But $L \cap \mathcal{L}(0^*1^*) = \{0^n 1^n | n \geq 0\}$ is not regular.
From this contradiction it follows that L is not regular.

(b) Assume L is a regular language.
From Theorem 2.26b it follows that L^C is also regular.
But $L^C = \{u \in \{0,1\}^* | \#_0(u) = \#_1(u)\}$, which we have proven to be not regular.
From this contradiction it follows that L is not regular.

Exercise 2.23

> Prove that the class of regular languages is closed under reversal, if the language L is regular, then so is $L^R = \{w^R | w \in L\}$.

(a) We can prove this using two approaches: showing that there exists a DFA accepting the reverse language, or showing that every regular expression has a reverse.

- *DFA* Let L be regular.
  Let $D = (Q, \Sigma, \delta, q_0, F)$ be a DFA with $\mathcal{L}(D) = L$.

  Define NFA $D^R = (Q \cup \{s\}, \Sigma, \vdash, s, \{q_0\}$ where
  $\vdash = \{(q, a, p) | p, q \in Q, a \in \Sigma, \delta(p, a) = q\} \cup \{(s, \tau, f) | f \in F\}$.

  For all $p, q \in Q, w \in \Sigma^*$ it holds that $(p, w) \vdash_D^* (q, \epsilon)$ iff $(q, w^R) \vdash_{D^R}^* (p, \epsilon)$.

  Furthermore for all $w \in \Sigma^*, q \in Q$ it holds that $(s, w) \vdash_D^* (q, \epsilon)$ iff $\exists_f [f \in F : (f, w) \vdash_{D^R}^* (q, \epsilon)]$

$$\begin{aligned}
&\mathcal{L}(D^R) \\
&\quad = \{w \in \Sigma^* | (s, w) \vdash_{D^R}^* (q_0, \epsilon)\} \\
&\quad = \{w \in \Sigma^* | \exists_f [f \ inF : (f, w) \vdash_{D^R}^* (q_0, \epsilon)]\} \\
&\quad = \{w \in \Sigma^* | w^R \in \mathcal{L}(D)\} \\
&\quad = \{w \in \Sigma^* | w^R \in L\} \\
&\quad = \{w \in \Sigma^* | w \in L^R\} \\
&= L^R
\end{aligned}$$

- *Regular expression* Every regular expression has a reverse, according to the following rules:
  (i) $\underline{0}^R = \underline{0}$
  (ii) $\underline{1}^R = \underline{1}$
  (iii) $a^R = a, a \in \Sigma^*$
  (iv) $(r_1 + r_2)^R = (r_1^R + r_2^R)$
  (v) $(r_1 \cdot r_2)^R = (r_2^R \cdot r_1^R)$
  (vi) $(r^*)^R = (r^R)^*$

  We can thus say $\mathcal{L}(r^R) = \mathcal{L}(r)^R$. The class of regular languages is thus closed under reversal.

Exercise 2.24

> The symmetric difference $X\Delta Y$ of two sets X and Y is given by $X\Delta Y = \{x \in X | x \notin Y\} \cup \{y \in Y | y \notin X\}$. Prove that the class of regular languages is closed under symmetric difference, *i.e.*, if the languages L_1 and L_2 are regular, then so is $L_1\Delta L_2$.

(a) Let L_1 and L_2 be regular languages.
Then according to the definition of symmetric difference $L_1\Delta L_2 = (L_1 \cap L_2^C) \cup (L_2 \cap L_1^C)$
Since L_1 and L_2 then according to Theorem 2.26b L_1^C and L_2^C are also regular.
According to Theorem 2.26c $L_1 \cap L_2^C$ and $L_2 \cap L_1^C$ are therefore also regular.
Finally Theorem 2.26a states $(L_1 \cap L_2^C) \cup (L_2 \cap L_1^C)$ must thus also be regular.

We can thus conclude that $L_1\Delta L_2$ is regular and therefore the class of regular languages is closed under symmetric difference.

[NOTE 13: Alternative proof**]**

3 Push-Down Automata and Context-Free Languages

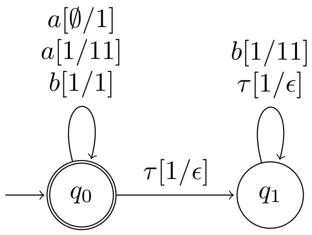
Learning targets chapter 3

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

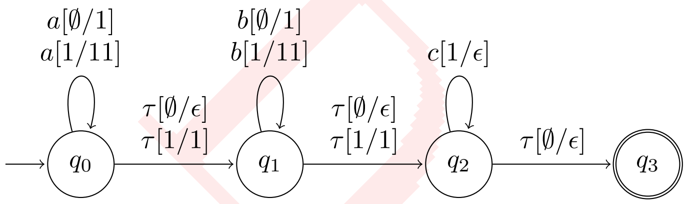
Construct a push-down automaton and give an invariant table for the following languages over the input alphabet $\Sigma = \{a, b, c\}$

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

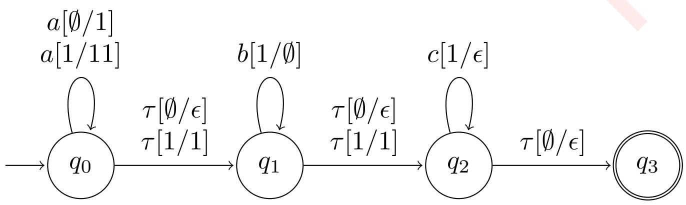
(a) $L_1 = \{a^n b^m c^{n+m} | n, m \geq 0\}$



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

state y	input w	stack 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) $L_2 = \{a^{n+m} b^n c^m | n, m \geq 0\}$



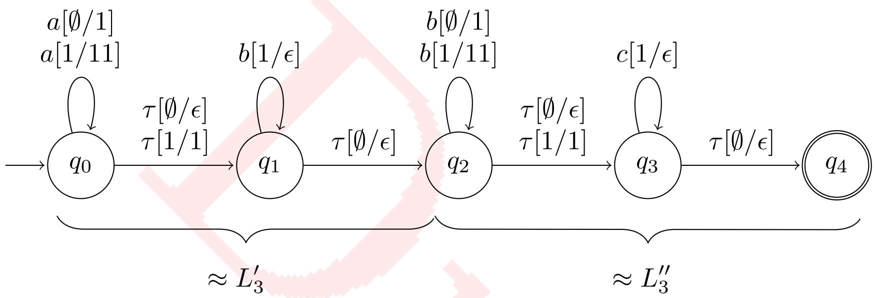
state y	input w	stack x	constraints
q_0	a^p	1^p	$p \geq 0$
q_1	$a^p b^q$	1^{p-q}	$0 \leq q \leq p$
q_2	$a^p b^q c^r$	1^{p-q-r}	$0 \leq q+r \leq p; q, r \geq 0$
q_3	$a^p b^q c^r$	ϵ	$q+r = p; q, r \geq 0$

(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:

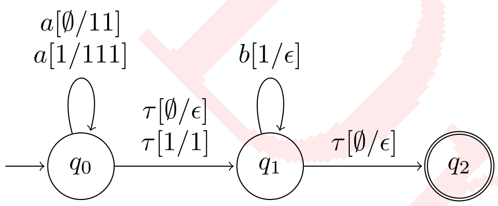
(a) $L_4 = \{a^n b^{2n} | n \geq 0\}$;

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

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

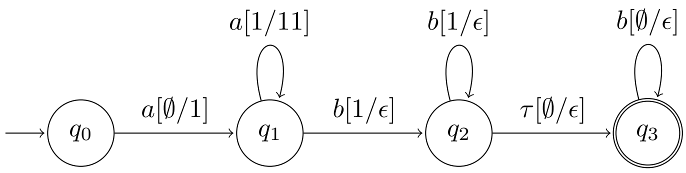
(d) $L_7 = \{a^n b^m | m, n \geq 0, m \neq n\}$.

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



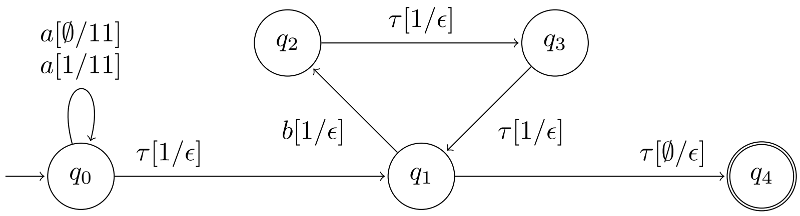
state	input	stack	constraints
q_0	a^n	1^{2n}	$n \geq 0$
q_2	$a^n b^m$	1^{2n-m}	$n \geq 0, 0 \leq m \leq 2n$
q_1	$a^n b^{2n}$	ϵ	$n \geq 0$

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



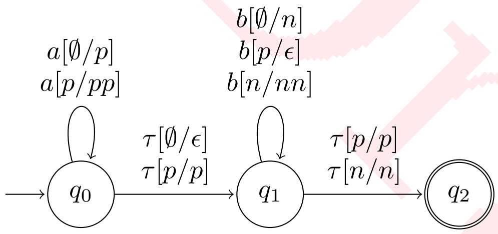
state	input	stack	constraints
q_0	ϵ	ϵ	
q_1	a^n	1^n	$n \geq 1$
q_2	$a^n b^m$	1^{n-m}	$1 \leq m \leq n$
q_3	$a^n b^m$	ϵ	$m \geq n \geq 1$

(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	$3m + 1 \leq 2n, m \geq 0, n > 0$

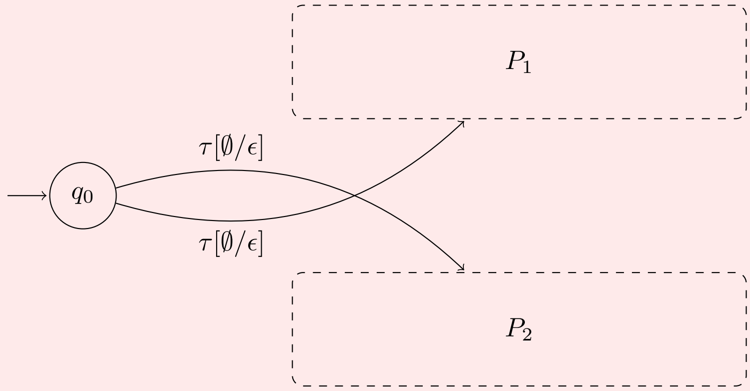
(d) $L_7 = \{a^n b^m | m, n \geq 0, m \neq n\}$



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

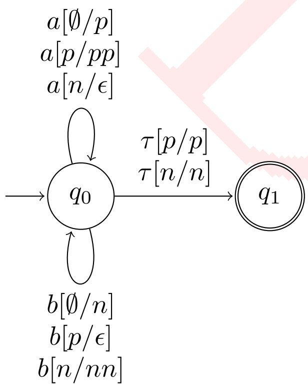
(a) Give a push-down automaton for the language

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

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

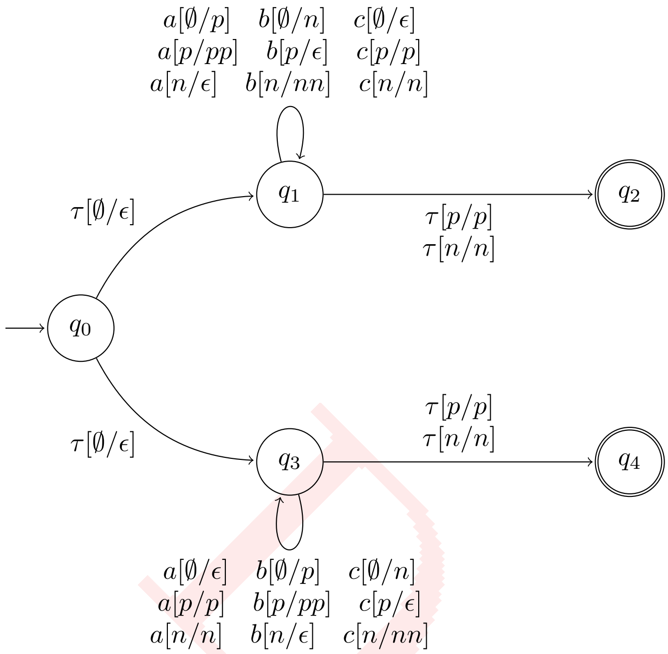
$$L_9 = \{w \in \{a,b,c\} | \#_a(w) \neq \#_b(w) \vee \#_b(w) \neq \#_c(w)\}$$

(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 14:** ref**]** we come up with the following automaton:



Definitions on.. [NOTE 15: whut]

A production rule:
$S \rightarrow XbY$

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

$$\begin{gathered} S \rightarrow XbY \\ X \rightarrow \epsilon | aX \\ Y \rightarrow \epsilon | aY | bY \end{gathered}$$

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 16: to be filled**]**

Exercise 3.6

Consider the context-free grammar G given by the production rules

$$S \to A|B$$
$$A \to \epsilon|aA$$
$$B \to \epsilon|bB$$

(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:

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

(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:

$$w \in \mathcal{L}(G)$$
$$\overset{val}{=}$$
$$w \in \mathcal{L}_G(S)$$
$$\overset{val}{=}$$
$$S \Rightarrow^* w \wedge w \in \{a, b\}^*$$
$$\overset{val}{=} \{casedistinctionfirststep : S \Rightarrow AorS \Rightarrow B\}$$
$$(A \Rightarrow^* w \vee B \Rightarrow^* w) \wedge w \in \{a, b\}^*$$
$$\overset{val}{=}$$
$$(A \Rightarrow^* \wedge w \in \{a, b\}^*) \vee (B \Rightarrow^* \wedge w \in \{a, b\}^*)$$
$$\overset{val}{=}$$
$$w \in \mathcal{L}_G(A) \vee w \in \mathcal{L}_G(B)$$
$$\overset{val}{=} \{Seeexercise(a)\}$$
$$w \in L_a \vee w \in L_b$$
$$\overset{val}{=}$$
$$w \in L_a \cup L_b$$

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:

$$\begin{gathered} S \to AT | TB \\ T \to \epsilon | aTb \\ A \to a | aA \\ B \to b | bB \end{gathered}$$

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 17:** 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 18:** work out**]**

Lemmas: **[NOTE 19:** existing?**]**

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

Proof 4:

$$\begin{aligned} \mathcal{L}(S) &= \mathcal{L}(AT) \cup \mathcal{L}(TB) \\ &= \mathcal{L}(A) \cdot \mathcal{L}(T) \cup \mathcal{L}(T) \cdot \mathcal{L}(B) \\ &= L_1 \textbf{[NOTE 20:} according to above lemmas\textbf{]} \end{aligned}$$

(b)

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

$$S \to S_1 C | A S_2$$

$$\begin{cases} S_1 \to A_1 T_1 | T_1 B_1 \\ T_1 \to \epsilon | a T_1 b \\ A_1 \to a | a A_1 \\ B_1 \to b | b B_1 \end{cases}$$

$$\begin{cases} C \to \epsilon c C \end{cases}$$

$$\begin{cases} A \to \epsilon a A \end{cases}$$

$$\begin{cases} S_1 \to B_2 T_2 | T_2 C_2 \\ T_1 \to \epsilon | b T_2 c \\ A_1 \to b | b B_2 \\ B_1 \to c | c C_2 \end{cases}$$

Exercise 3.8

Give a construction, based on the number of operators, that shows that every **[NOTE 21:** 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 22: 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 23:** 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 24: 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 25:**$nocluesss$**]**$]$

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

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)

Copyholder
Christiaan Dirkx & Alex Dings

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

    $S \xRightarrow{S \to aS}{}^{h} a^h S \xRightarrow{S \to Sb}{}^{\ell} a^h S b^{\ell} \xRightarrow{S \to a}{}^{1} a^{h+1} b^{\ell}$
    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:

    $S \xRightarrow{S \to aS}{}^{h} a^h S \xRightarrow{S \to Sb}{}^{\ell} a^h S b^{\ell} \xRightarrow{S \to b}{}^{1} a^h b^{\ell+1}$
    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 26: 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 27:** 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 28:** 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

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

  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.

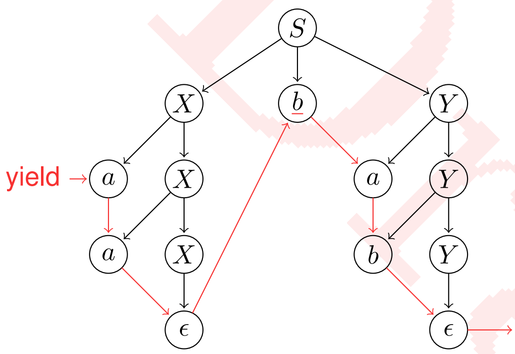
Exercise 3.12

Consider again the the grammar of Exercise 3.5 with production rules

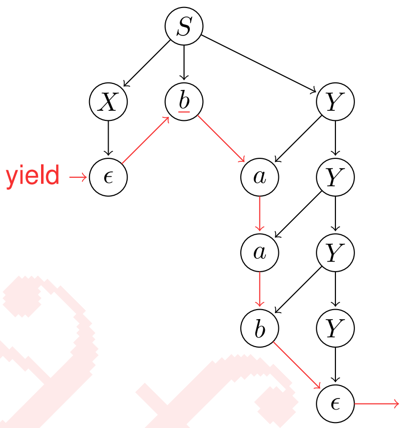
$$
\begin{aligned}
S &\rightarrow XbY \\
X &\rightarrow \epsilon | aX \\
Y &\rightarrow \epsilon | aY | bY
\end{aligned}
$$

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*.

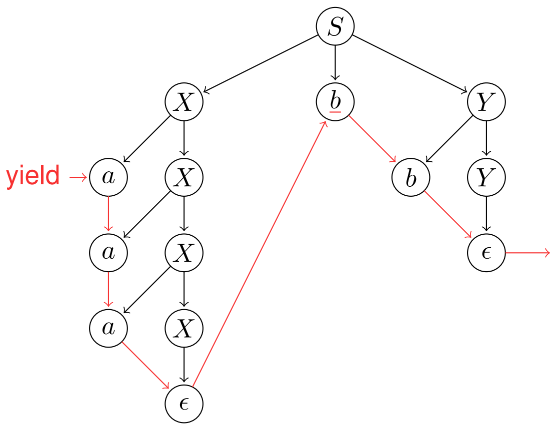
Parse tree for aabab (unique):



Parse tree for baab (unique):



Parse tree for aaabb (unique):



Exercise 3.13

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

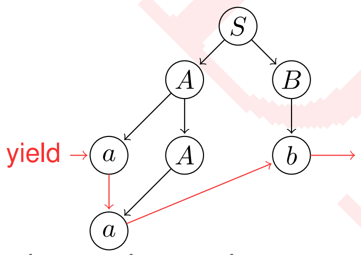
$$S \to AB|aaB \qquad A \to a|Aa \qquad B \to b$$

is ambiguous.

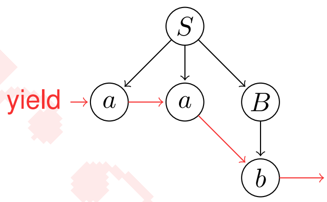
(b) Provide an unambiguous grammar G that generates the same language as G. Argue why G is unambiguous and why $\mathcal{L}(G') = \mathcal{L}(G)$.

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

String aab has two different complete parse trees:



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



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

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

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 29: could be added later**]**

Exercise 3.14

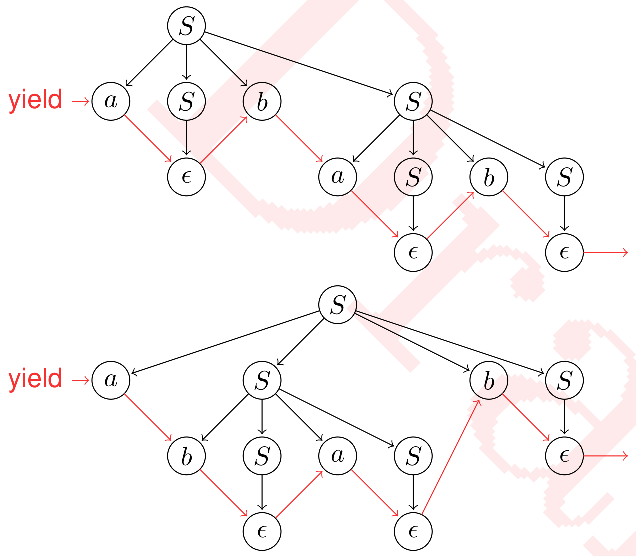
(a) Show that the grammar G given by the production rules

$$S \to \epsilon | aSbS | bSaS$$

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 30:** ermergewd, te lang**]**

Exercise 3.15

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

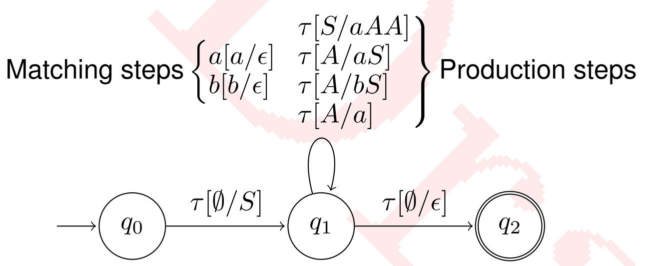
$$S \rightarrow aAAA \qquad \rightarrow aS|bS|a$$

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

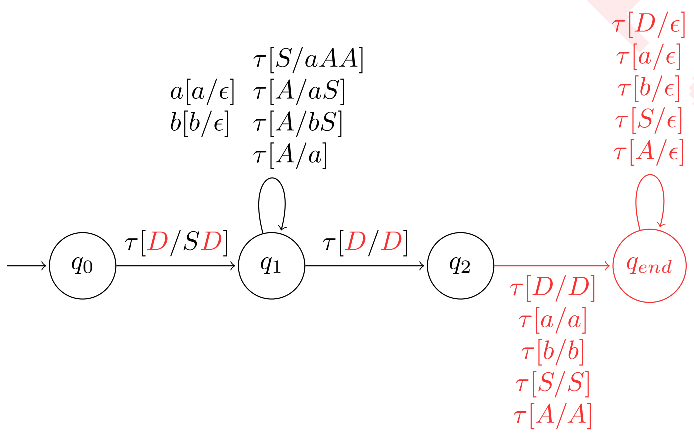
First method:

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

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



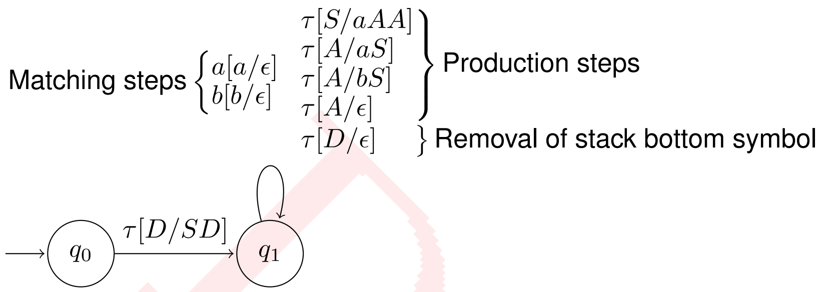
Transformation to a PDA accepting on empty stack from the proof of TH 3.29



Second method:

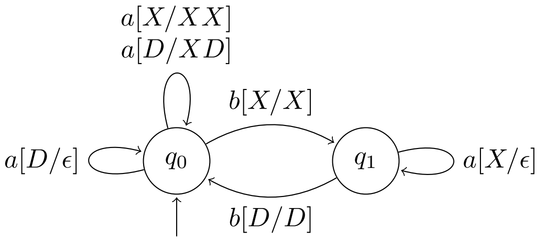
$G: \quad S \to aAA \quad A \to aS|bS|a$

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



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

$S \rightarrow [q_0 D q_0] | [q_0 D q_1]$

$q_0 \xrightarrow{a[D/XD]} q_0$	$[q_0 D q_0] \rightarrow a[q_0 X q_0][q_0 D q_0] \mid a[q_0 X q_1][q_1 D q_0]$ $[q_0 D q_1] \rightarrow a[q_0 X q_0][q_0 D q_1] \mid a[q_0 X q_1][q_1 D q_1]$
$q_0 \xrightarrow{a[X/XX]} q_0$	$[q_0 X q_0] \rightarrow a[q_0 X q_0][q_0 X q_0] \mid a[q_0 X q_1][q_1 X q_0]$ $[q_0 X q_1] \rightarrow a[q_0 X q_0][q_0 X q_1] \mid a[q_0 X q_1][q_1 X q_1]$
$q_0 \xrightarrow{b[X/X]} q_1$	$[q_0 X q_0] \rightarrow b[q_1 X q_0]$ $[q_0 X q_1] \rightarrow b[q_1 X q_1]$
$q_0 \xrightarrow{a[X/\epsilon]} q_1$	$[q_1 X q_1] \rightarrow a$
$q_1 \xrightarrow{b[D/D]} q_0$	$[q_1 D q_0] \rightarrow b[q_0 D q_0]$ $[q_1 D q_1] \rightarrow b[q_0 D q_1]$
$q_0 \xrightarrow{\tau[D/\epsilon]} q_0$	$[q_0 D q_0] \rightarrow \epsilon$

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

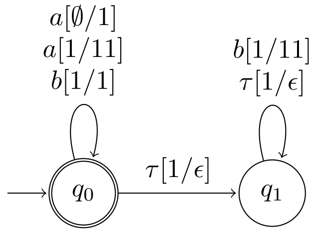
① $[q_0 D q_0]$ ② $[q_1 D q_0]$ ③ $[q_1 X q_1]$ ④ $[q_0 X q_1]$ ⑤ S

Reduced grammar (rules only):

$$
\begin{aligned}
S &\rightarrow [q_0 D q_0] \\
[q_0 D q_0] &\rightarrow a[q_0 X q_1][q_1 D q_0] | \epsilon \\
[q_0 X q_1] &\rightarrow a[q_0 X q_1][q_1 X q_1] | b[q_1 X q_1] \\
[q_1 X q_1] &\rightarrow a \\
[q_1 D q_0] &\rightarrow [q_0 D q_0]
\end{aligned}
$$

Exercise 3.17

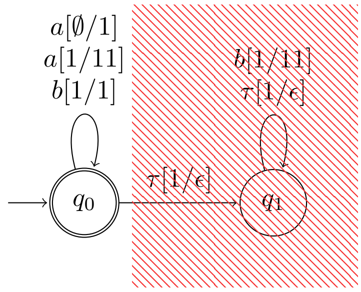
(Hopcroft, Motwani & Ullman, 2001) Consider again the PDA of Exercise 3.1 repeated below.



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

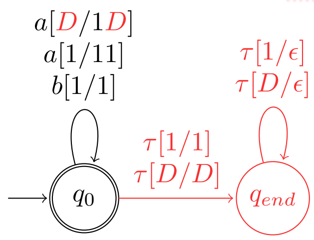
(b) Determine all productive symbols in the constructed grammar and give the reduced grammar (see (b) of the previous question for a definition of productive symbols).

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

$S \rightarrow [q_0 D q_0] | [q_0 D q_{and}]$

$q_0 \xrightarrow{a[D/1D]} q_0$	$[q_0 D q_0]$	$\rightarrow a[q_0 1 q_0][q_0 D q_0] \mid a\,[q_0 1 q_{end}][q_{end} D q_0]$
	$[q_0 D q_{end}]$	$\rightarrow a[q_0 1 q_0]\,[q_0 D q_{end}] \mid a\,[q_0 1 q_{and}]\,[q_{end} D q_{end}]$
$q_0 \xrightarrow{a[1/11]} q_0$	$[q_0 1 q_0]$	$\rightarrow a[q_0 1 q_0][q_0 1 q_0] \mid a\,[q_0 1 q_{end}][q_{end} 1 q_0]$
	$[q_0 1 q_{end}]$	$\rightarrow a[q_0 1 q_0]\,[q_0 1 q_{end}] \mid a\,[q_0 1 q_{and}]\,[q_{end} 1 q_{end}]$
$q_0 \xrightarrow{b[1/1]} q_0$	$[q_0 1 q_0]$	$\rightarrow b[q_0 1 q_0]$
	$[q_0 1 q_{end}]$	$\rightarrow b\,[q_0 1 q_{end}]$
$q_0 \xrightarrow{\tau[1/1]} q_{end}$	$[q_0 1 q_0]$	$\rightarrow [q_{end} 1 q_0]$
	$[q_0 1 q_{end}]$	$\rightarrow [q_{end} 1 q_{end}]$
$q_0 \xrightarrow{\tau[D/D]} q_{end}$	$[q_0 D q_0]$	$\rightarrow [q_{end} D q_0]$
	$[q_0 D q_{end}]$	$\rightarrow [q_{end} D q_{end}]$
$q_{end} \xrightarrow{\tau[1/\epsilon]} q_{end}$	$[q_{end} 1 q_{end}]$	$\rightarrow \epsilon$
$q_{end} \xrightarrow{\tau[D/D]} q_{end}$	$[q_{end} D q_{end}]$	$\rightarrow \epsilon$

(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):

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

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

Reduced grammar (rules only), after substitution:

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

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 $m^2 < m^2 + |vy| \leq m^2 + m = m(m+1) < (m+1)^2$.

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

Learning targets chapter 4

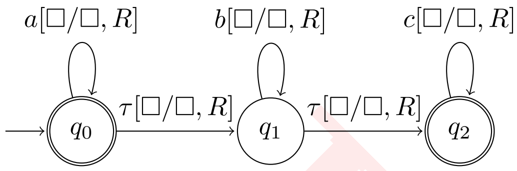
At the end of this chapter the student should be able to:

- Construct a reactive Turing machine from a language.
- Give an accepting sequence for an input of a reactive turing machine
- Argue why an input is not accepted. (No formal proof required)
- Construct a classical Turing machine that computes a function.
- Give a computing sequence for an input of a classical Turing machine.

Exercise 4.1

Construct a reactive Turing machine for the language $L = \{a^n b^m c^\ell | n, m, \ell \geq 0\}$. Give an accepting computation sequence for the string $abbccc$. Argue why the strings $aaccbb$ and bca are not accepted. A proof of correctness is not asked for.

$L = \{a^n b^m c^\ell | n, m, \ell \geq 0\}$ (regular language)



- $abbccc$ is accepted;
  accepting computation sequence:

$$\begin{aligned}(q_0, abbccc, < \Box \geq) &\vdash (q_0, bbccc, < \Box \geq)\\ &\vdash (q_1, bbccc, < \Box \geq)\\ &\vdash (q_1, bccc, < \Box \geq)\\ &\vdash (q_1, ccc, < \Box \geq)\\ &\vdash (q_2, ccc, < \Box \geq)\\ &\vdash (q_2, cc, < \Box \geq)\\ &\vdash (q_2, c, < \Box \geq)\\ &\vdash (q_2, \epsilon, < \Box \geq)\end{aligned}$$

- $aaccbb$ is not accepted:

$$(q_0, bca, < \Box \geq)$$

  Only possible transitions to process 2 a's:

$$\vdash^2 (q_0, ccbb, < \Box \geq)$$

  Only possible transitions that lead to the possibility of processing 2 c's:

$$\vdash^2 (2_2, ccbb, < \Box \geq)$$

  Only possible transitions to process 2 c's:

$$\vdash^2 (q_2, bb, < \Box \geq) \nvdash$$

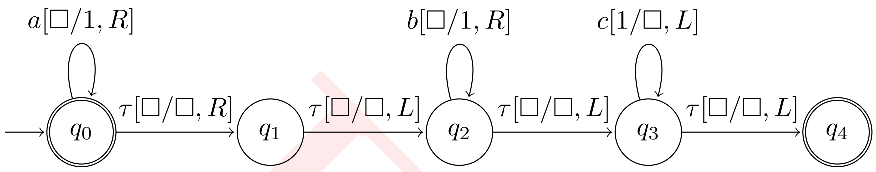
- bca is not accepted:
  Only way to process a b followed by a c:

$$\begin{aligned}(q_0, bca, < \Box \geq) &\vdash (q_1, bca, < \Box \geq)\\ &\vdash (q_1, ca, < \Box \geq)\\ &\vdash (q_2, ca, < \Box \geq)\\ &\vdash (q_2, a, < \Box \geq) \nvdash\end{aligned}$$

Exercise 4.2

Construct, for the language $L = \{a^n b^m c^{n+m} | n, m \geq 0\}$, a reactive Turing machine. Give an accepting computation sequence for the string aaabcccc. Argue why the strings $aabbcc$ and $abccc$ are not accepted. A proof of correctness is not asked for.

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



- $aaabcccc$ is accepted:

$$
\begin{aligned}
(q_0, aaabcccc, < \Box \geq) &\vdash^4 (q_0, bcccc, 111 < \Box \geq) \\
&\vdash (q_1, bcccc, 111\Box < \Box \geq) \\
&\vdash (q_2, bcccc, 111 < \Box \geq) \\
&\vdash (q_2, cccc, 1111 < \Box \geq) \\
&\vdash (q_3, cccc, 111 < \Box \geq) \\
&\vdash^4 (q_3, \epsilon, < \Box \geq) \\
&\vdash (q_4, \epsilon, < \Box \geq)
\end{aligned}
$$

- $aabbcc$ is not accepted:

$$
\begin{aligned}
(q_0, aabbcc, < \Box \geq) &\vdash^* (q_2, cc, 1111 < \Box \geq) \\
&\vdash (q_3, cc, 111 < 1 \geq) \\
&\vdash^2 (q_3, \epsilon, 1 < 1 \geq) \not\vdash
\end{aligned}
$$

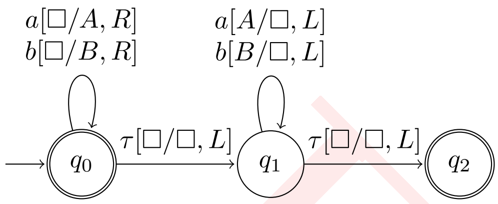
- $abccc$ is not accepted:

$$
\begin{aligned}
(q_0, abccc, < \Box \geq) \vdash^* &\vdash^* (q_2, ccc, 11 < \Box \geq) \\
&\vdash (q_3, ccc, 1 < \Box \geq) \\
&\vdash^2 (q_3, c, < \Box \geq) \\
&\vdash (q_4, c, < \Box \geq) \not\vdash
\end{aligned}
$$

Exercise 4.3

Construct a reactive Turing machine for the language $L = \{ww^R | w \in \{a, b\}\}$. Give an accepting computation sequence for the string $aabbaa$. Argue why the strings $aabb$ and $abbba$ are not accepted. A proof of correctness is not asked for.

$L = \{ww^R | w \in \{a, b\}\}$ (context-free language)



– General accepting sequence for ww^R with $w \neq \epsilon$ (say $w = ud$)

$$
\begin{aligned}
(q_0, ww^R, < \Box \geq) &\vdash^* (q_0, w^R, W < \Box \geq) \\
&\vdash (q_1, w^R, U < D \geq) \\
&\vdash^* (q_1, \epsilon, < \Box \geq) \\
&\vdash (q_2, \epsilon, < \Box \geq)
\end{aligned}
$$

– $aabbaa$ is accepted:

$$
\begin{aligned}
(q_0, aabbaa, < \Box \geq) &\vdash^3 (q_0, baa, AAB < \Box \geq) \\
&\vdash (q_1, baa, AA < B \geq) \\
&\vdash^3 (q_1, \epsilon, < \Box \geq) \\
&\vdash (q_2, \epsilon, < \Box \geq)
\end{aligned}
$$

- $aabb$ is not accepted:

$$(q_0, aabb, <\Box\geq)$$

All possible computations, none of which is accepting:

1. $$\vdash (q_1, aabb, <\Box\geq) \vdash (q_2, aabb, <\Box\geq) \nvdash$$

2. $$\vdash (q_0, abb, A<\Box\geq) \vdash (q_1, abb, <A\geq) \vdash (q_1, bb, <\Box\geq) \vdash (q_2, bb, <\Box\geq) \nvdash$$

3. $$\vdash^2 (q_0, bb, AA<\Box\geq) \vdash (q_1, bb, A<A\geq) \nvdash$$

4. $$\vdash^3 (q_0, b, AAB<\Box\geq) \vdash (q_1, b, AA<B\geq) \vdash (q_1, \epsilon, A<A\geq) \nvdash$$

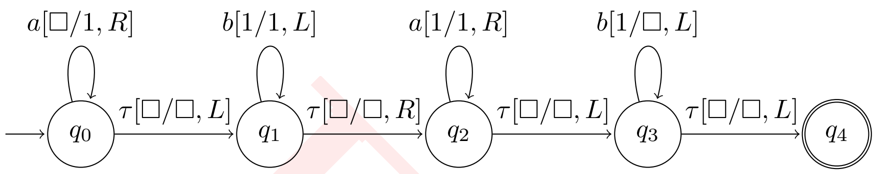
5. $$\vdash^4 (q_0, \epsilon, AABB<\Box\geq) \vdash (q_1, \epsilon, AAB<B\geq) \nvdash$$

- $abbba$ is not accepted:
By a case distinction as above it can be shown that there is no configuration (q_2, ϵ, z) with $(q_0, abbba, <\Box\geq) \vdash^* (q_2, \epsilon, z)$

Exercise 4.4

Construct a reactive Turing machine for the language $L = a^n b^n a^n b^n | n \geq 0$. Give an accepting computation sequence for the string $aabbaabb$. Argue why the strings $abbaabb$ and $abbbab$ are not accepted. A proof of correctness is not asked for.

$L = a^n b^n a^n b^n | n \geq 0$ (<u>not</u> context-free)



– General accepting sequence for $a^n b^n a^n b^n$ with $n \geq 0$

$$\begin{aligned}
(q_0, a^n b^n a^n b^n, < \Box >) &\vdash^n (q_0, b^n a^n b^n, 1^n < \Box >) \\
&\vdash (q_1, b^n a^n b^n, 1^{n-1} < \Box >) \\
&\vdash^n (q_1, a^n b^n, < \Box > 1^n) \\
&\vdash (q_2, a^n b^n, < 1 > 1^{n-1} < 1 >) \\
&\vdash^n (q_2, b^n, 1^n < \Box >) \\
&\vdash (q_3, b^n, 1^{n-1} < 1 >) \\
&\vdash^n (q_3, \epsilon, < \Box >) \\
&\vdash (q_4, \epsilon, < \Box >)
\end{aligned}$$

– $a^0 b^0 a^0 b^0 = \epsilon$ is accepted:

$$\begin{aligned}
(q_0, \epsilon, < \Box >) &\vdash (q_1, \epsilon, < \Box >) \\
&\vdash (q_2, \epsilon, < \Box >) \\
&\vdash (q_3, \epsilon, < \Box >) \\
&\vdash (q_4, \epsilon, < \Box >)
\end{aligned}$$

– $ab^2a^2b^2$ is not accepted:

$$(q_0, abbaabb, < \Box >)$$

All possible computations, none of which is accepting:

1.
$$\vdash^4 (q_4, abbaabb, < \Box >) \nvdash$$

2.
$$(q_0, bbaabb, 1 < \Box >) \vdash (q_1, bbaabb, < 1 >) \vdash (q_1, baabb, < \Box > 1) \vdash (q_2, baabb, < 1 >) \nvdash$$

– ab^3ab is not accepted:

$$(q_0, abbbab, <\Box>)$$

All possible computations, none of which is accepting:

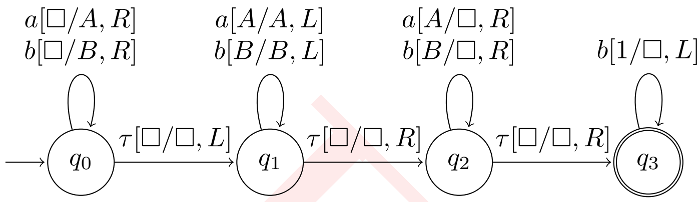
1.
$$\vdash^4 (q_4, abbbab, <\Box>) \nvdash$$

2.
$$(q_0, bbbab, 1<\Box>) \vdash (q_1, bbbab, <1>) \vdash (q_1, bbab, <\Box>1) \vdash (q_2, bbab, <1>) \nvdash$$

Exercise 4.5

Construct a reactive Turing machine for the language $L = \{ww^Rw | w \in \{a,b\}\}$. Give a computation sequence for the string $aabbaaaab$. Argue why the strings aab and $ababba$ are not accepted. A proof of correctness is not asked for.

$L = \{ww^Rw | w \in \{a,b\}\}$ (<u>not</u> context-free)



– $aabbaaaab$ is accepted:

$$\begin{aligned}(q_0, aabbaaaab, <\square>) &\vdash^3 (q_0, baaaab, AAB<\square>)\\ &\vdash (q_1, baaaab, AA<B>)\\ &\vdash^3 (q_1, aab, <\square>AAB)\\ &\vdash (q_2, aab, <A>AB)\\ &\vdash^3 (q_2, \epsilon, <\square>)\\ &\vdash (q_3, \epsilon, <\square>)\end{aligned}$$

– aab is not accepted:

aab can not be accepted, since the b must be either processed:

- in state q_0 resulting in writing a B on the tape. This B cannot be skipped in q_1 or removed in q_2 due to missing additional b's in the input
- in state q_1 or state q_2, which is impossible since no B has been written to the tape

– $ababba$ is not accepted:

$$ab|ab|ba$$

$$(q_0, ababba, < \Box >)$$

1 step in state q_0

$$\vdash^1 \ldots \nvdash$$

2 step in state q_0

$$\vdash^2 (q_0, abba, AB < \Box >) \vdash (q_1, abba, A < B >) \nvdash$$

3 step in state q_0

$$\vdash^3 \ldots \nvdash$$

4 step in state q_0

$$\vdash^4 (q_0, ba, ABAB < \Box >) \vdash (q_1, ba, ABA < B >) \vdash^2 (q_1, \epsilon, A < B >) \nvdash$$

5 step in state q_0

$$\vdash^5 \ldots \nvdash$$

6 step in state q_0

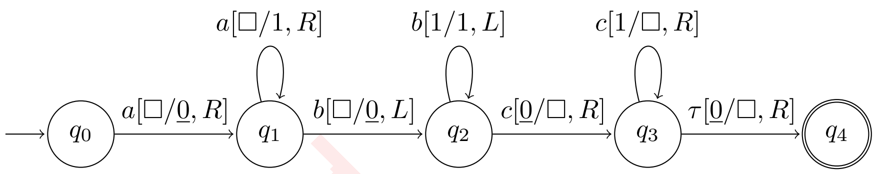
$$\vdash^6 (q_0, \epsilon, ABABBA < \Box >) \vdash (q_1, \epsilon, ABABB < A >) \nvdash$$

[NOTE 31: Why the dots? See above**]**

Exercise 4.6

Construct a reactive Turing machine for the language $L = \{a^n b^n c^n | n > 0\}$ that has at most one τ-move. A proof of correctness is not asked for.

$L = \{a^n b^n c^n | n > 0\}$ (<u>not</u> context-free)

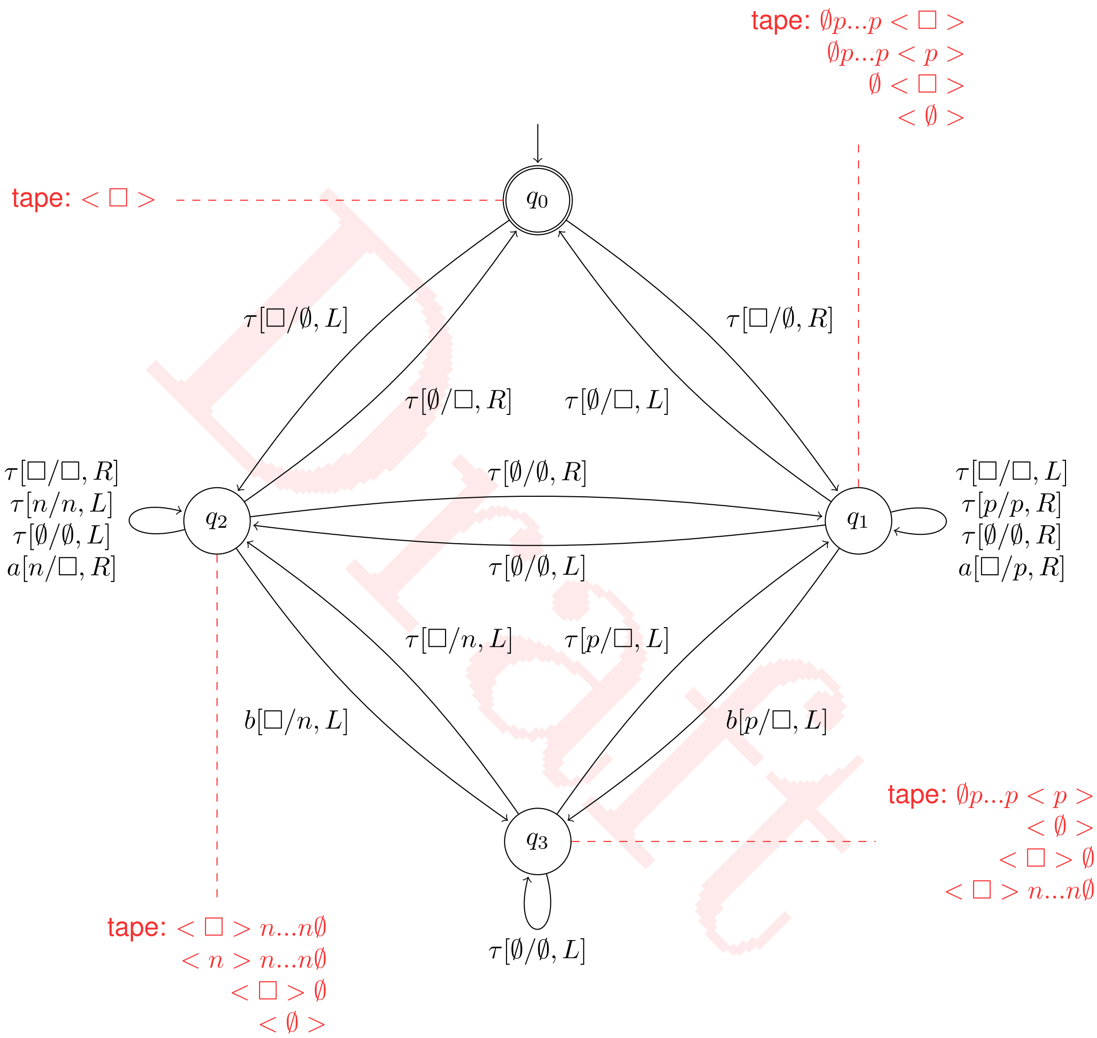


General accepting sequence for $a^n b^n c^n$ with $n > 0$

$$\begin{aligned}
(q_0, a^n b^n c^n, < \square >) &\vdash^n (q_1, b^n c^n, \underline{0}1^{n-1} < \square >)\\
&\vdash^n (q_2, c^n, < \underline{0} > 1^{n-1}\underline{0})\\
&\vdash^n (q_3, \epsilon, < \underline{0} >)\\
&\vdash (q_4, \epsilon, < \square >)
\end{aligned}$$

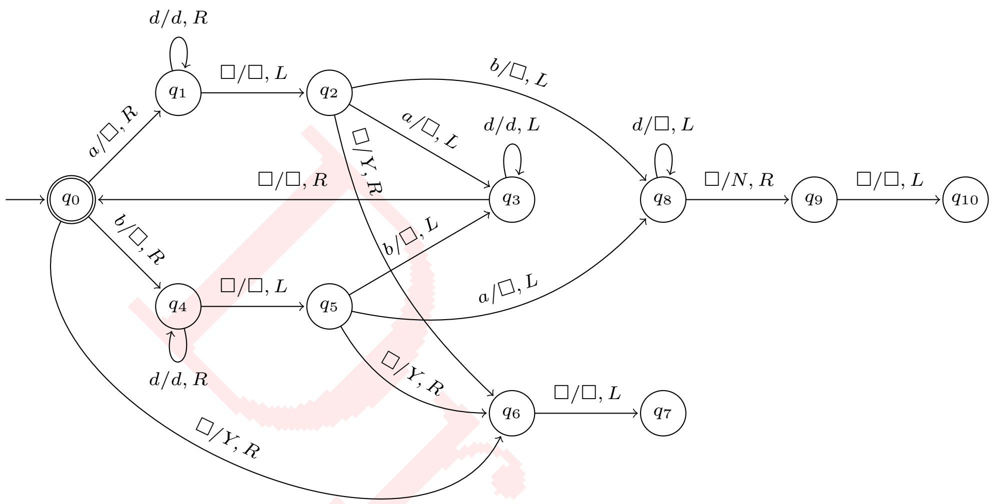
Exercise 4.7

(optional) Construct a reactive Turing machine for the language $L = \{w \in \{a, b\} | \#_a(w) = 2 \cdot \#_b(w)\}$ with at most 4 states. A proof of correctness is not asked for.



Exercise 4.8

Construct a classical Turing machine that computes a function $p : \{a, b\} \rightarrow \{Y, N\}$ such that $p(w) = Y$ if w is a palindrome, and $p(w) = N$ if w is not a palindrome. Give a computation sequence for the strings $ababa$ and $abba$ producing Y , and for the strings the strings $aaba$ and baa producing N. A proof of correctness is not asked for.



$d \in \{a, b\}$

Notation$< u >$:

$< \epsilon > = < \square >$
$< du' > = < d > u'$

$p(ababa) = Y$

$$
\begin{array}{ll}
 & (q_0, < a > baba) \\
\vdash & (q_1, < b > aba) \\
\vdash^* & (q_1, baba < \Box >) \\
\vdash & (q_2, bab < a >) \\
\vdash & (q_3, ba < b >) \\
\vdash^* & (q_3, < \Box > bab) \\
\vdash & (q_0, < b > ab) \\
\vdash & (q_4, < a > b) \\
\vdash^* & (q_4, ab < \Box >) \\
\vdash & (q_5, a < b >) \\
\vdash & (q_3, < a >) \\
\vdash & (q_3, < \Box > a) \\
\vdash & (q_0, < a >) \\
\vdash & (q_1, < \Box >) \\
\vdash & (q_2, < \Box >) \\
\vdash & (q_6, Y < \Box >) \\
\vdash & (q_7, < Y >)
\end{array}
$$

$p(abba) = Y$

$$
\begin{array}{ll}
 & (q_0, < a > bba) \\
\vdash & (q_1, < b > ba) \\
\vdash^* & (q_1, bba < \Box >) \\
\vdash & (q_2, bb < a >) \\
\vdash & (q_3, b < b >) \\
\vdash^* & (q_3, < \Box > bb) \\
\vdash & (q_0, b < b >) \\
\vdash & (q_4, < b >) \\
\vdash & (q_4, b < \Box >) \\
\vdash & (q_5, < b >) \\
\vdash & (q_3, < \Box >) \\
\vdash & (q_0, < \Box >) \\
\vdash & (q_6, Y < \Box >) \\
\vdash & (q_7, < Y >)
\end{array}
$$

$p(aaba) = N$

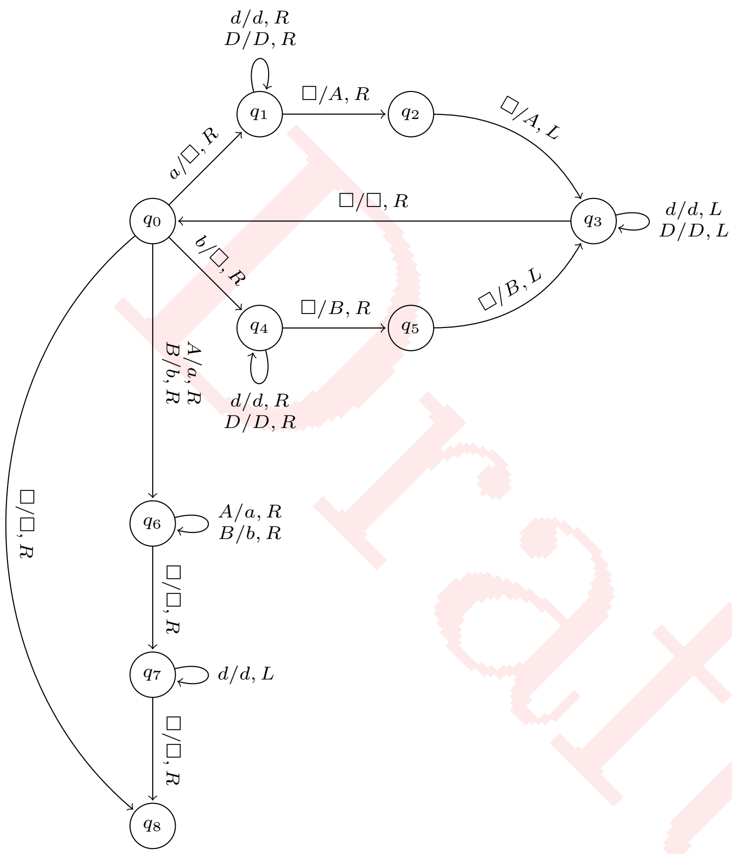
$$
\begin{array}{ll}
 & (q_0, < a > aba) \\
\vdash & (q_1, < a > ba) \\
\vdash^* & (q_1, aba < \Box >) \\
\vdash & (q_2, ab < a >) \\
\vdash & (q_3, a < b >) \\
\vdash^* & (q_3, < \Box > ab) \\
\vdash & (q_0, < a > b) \\
\vdash & (q_1, < b >) \\
\vdash & (q_1, b < \Box >) \\
\vdash & (q_2, < b >) \\
\vdash & (q_8, < \Box >) \\
\vdash & (q_9, N < \Box >) \\
\vdash & (q_{10}, < N >)
\end{array}
$$

$p(baa) = N$

$$
\begin{array}{ll}
 & (q_0, < b > aa) \\
\vdash & (q_4, < a > a) \\
\vdash^* & (q_4, aa < \Box >) \\
\vdash & (q_5, a < a >) \\
\vdash & (q_8, < a >) \\
\vdash & (q_8, < \Box >) \\
\vdash & (q_9, N < \Box >) \\
\vdash & (q_{10}, < N >)
\end{array}
$$

Exercise 4.9

Construct a classical Turing machine that computes the function $dbl : \{a, b\} \to \{a, b\}$ defined by $dbl(\epsilon) = \epsilon$ and $dbl(eu) = eedbl(u)$ for $e \in \{a, b\}$ and $u \in a, b$. Give a computation sequence for strings ϵ and aab. A proof of correctness is not asked for.



$d \in \{a, b\}$

$D \in \{A, B\}$

$dbl(\epsilon) = \epsilon$

$$\begin{array}{ll} & (q_0, <\Box>) \\ \vdash & (q_8, <\Box>) \end{array}$$

$dbl(aab) = aadbl(ab) = aaaadbl(b) = aaaabb$

$$\begin{array}{ll} & (q_0, <a> ab) \\ \vdash & (q_1, <a> b) \\ \vdash^2 & (q_1, ab <\Box>) \\ \vdash & (q_2, abA <\Box>) \\ \vdash & (q_3, ab <A> A) \\ \vdash^3 & (q_3, <\Box> abAA) \\ \vdash & (q_0, <a> bAA) \\ \vdash^* & (q_0, <b> AAAA) \\ \vdash^* & (q_0, <A> AAABB) \\ \vdash & (q_6, a <A> AABB) \\ \vdash^5 & (q_6, aaaabb <\Box>) \\ \vdash & (q_7, aaaab <b>) \\ \vdash^6 & (q_7, <\Box> aaaabb) \\ \vdash & (q_8, <a> aaabb) \end{array}$$

Exercise 4.10

Construct a classical Turing machine for the Dutch national flag problem, *i.e.*, a Turing machine that computes the function $dnf : \{R, W, B\} \to \{R, W, B\}$ such that, for any string $w \in \{R, W, B\}$, $dnf(w) = R^n W^m B^\ell$ where $n = \#_R(w), m = \#_W(w)$, and $\ell = \#_B(w)$. Give a computation sequence for the strings $RWBRW$, and BWB. A proof of correctness is not asked for.

In place sorting algorithm on array a (elements indexed: 0,1,...,N-1), with output

$$R^n W^m B^\ell$$

where $n = \#_R(w), m = \#_W(w)$, and $\ell = \#_B(w)$.

$r := 0, i := 0, b := N$

do if $i \neq b$

    if $a[i] = R$

        swap $a[r]$ and $a[i]$

        $r := r + 1, i := i + 1$

    else if $a[i] = W$

        $i := i + 1$

    else if $a[i] = B$

        swap $a[i]$ and $a[b - 1]$

        $b := b - 1$

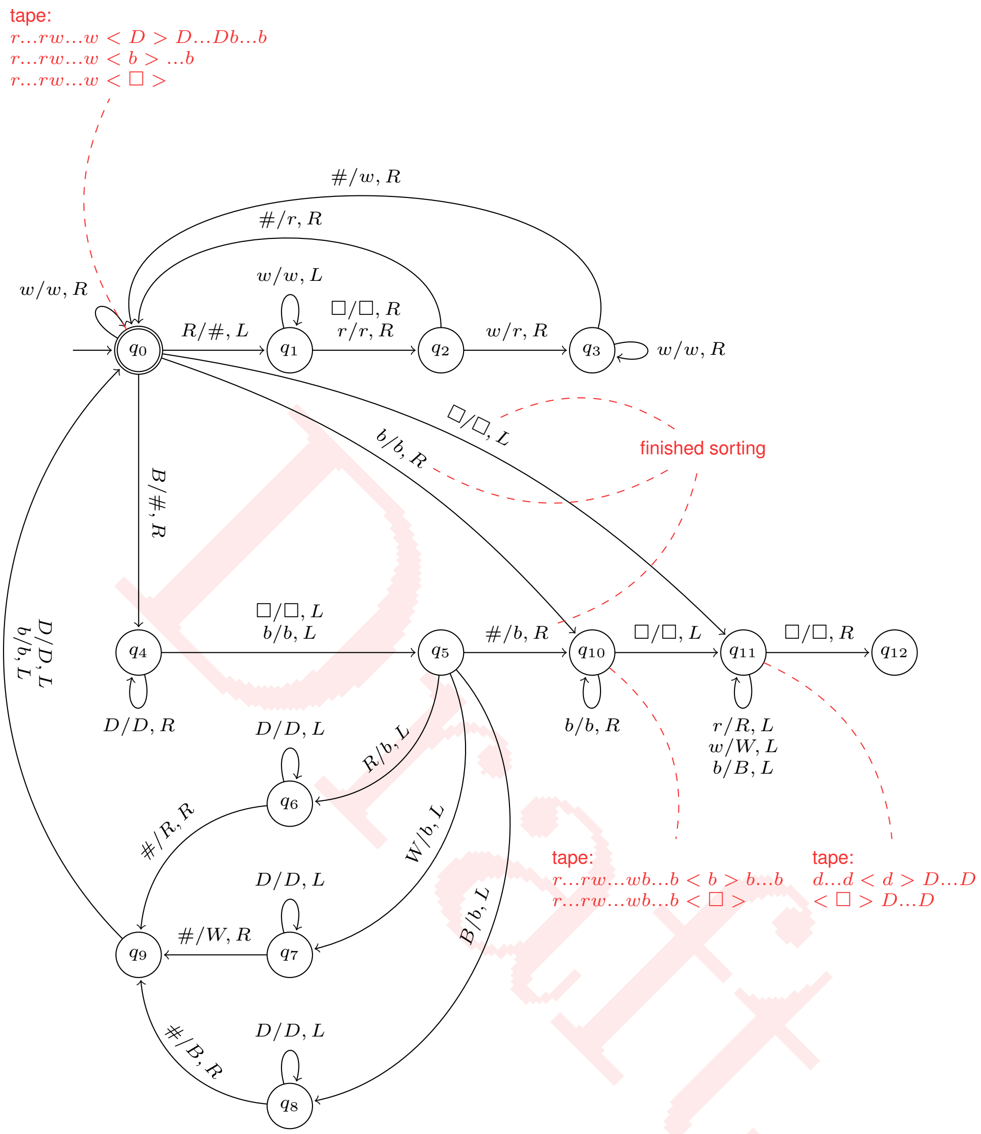
    fi

od

$dnf(RWBRW) = RRWWB$

$$\begin{array}{ll}
 & (q_0, < R > WBRW) \\
\vdash & (q_1, < \Box > \#WBRW) \\
\vdash & (q_2, < \# > WBRW) \\
\vdash & (q_0, r < W > BRW) \\
\vdash & (q_0, rw < B > RW) \\
\vdash & (q_4, rw\# < R > W) \\
\vdash^* & (q_4, rw\#RW < \Box >) \\
\vdash & (q_5, rw\#R < W >) \\
\vdash & (q_7, rw\# < R > b) \\
\vdash & (q_7, rw < \# > Rb) \\
\vdash & (q_9, rwW < R > b) \\
\vdash & (q_0, rw < W > Rb)
\end{array}
\qquad
\begin{array}{ll}
\vdash & (q_0, rww < R > b) \\
\vdash & (q_1, rw < w > \#b) \\
\vdash^* & (q_1, < r > ww\#b) \\
\vdash & (q_2, r < w > w\#b) \\
\vdash & (q_3, rr < w > \#b) \\
\vdash & (q_3, rrw < \# > b) \\
\vdash & (q_0, rrww < b >) \\
\vdash & (q_{10}, rrwwb < \Box >) \\
\vdash & (q_{11}, rrww < b >) \\
\vdash^* & (q_{11}, < \Box > RRWWB) \\
\vdash & (q_{12}, < R > RWWB)
\end{array}$$

[NOTE 32: Needs BWB**]**



$D \in \{R, W, B\}$

Additional tape symbols:

r, w, b "sorted" symbols on tape
$\#$ marker symbol used for swapping

Exercise 4.11

(a) Construct a classical Turing machine that computes a function ${}^2log : \mathcal{L}(1 \cdot (0+1)) \rightarrow \{0,1\}$ such that $2log(w) = n$ if $2^n \leq w < 2^{n+1}$ with the string w interpreted as a binary number. The result should be represented as a binary number.

(b) Construct a Turing machine that computes a function $\texttt{2to3} : \{0,1\} \rightarrow \{0,1,2\}$ such that if the string w represents in binary the number n, the string $\texttt{2to3}(w)$ represent in ternary the same number n.

[NOTE 33: ToDo**]**

Lijst van Notities - ToDo