cse303
ELEMENTS OF THE THEORY OF COMPUTATION

Professor Anita Wasilewska

LECTURE 10

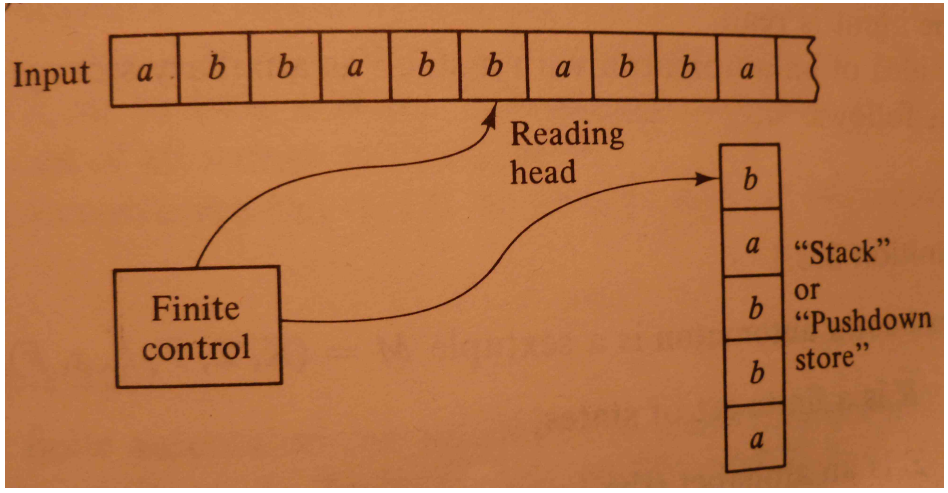
CHAPTER 3
CONTEXT-FREE LANGUAGES

1. Context-free Grammars
2. Parse Trees
3. Pushdown Automata
4. Pushdown automata and context -free grammars
5. Languages that are not context- free

CHAPTER 3
PART 3: Pushdown automata

Pushdown Automata PDA

Computational Model of Pushdown Automata PDA

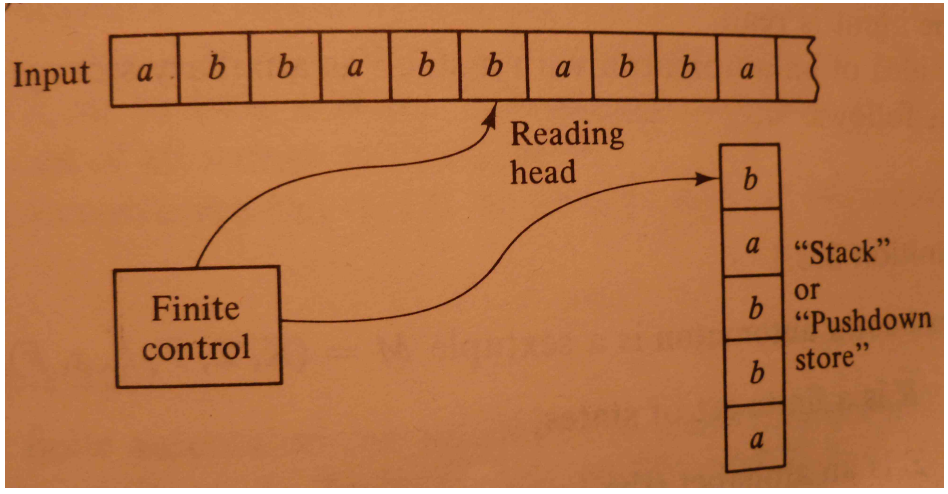


C1: Automata **"remembers"** what it has already read by putting it, one symbol at the time on **stack**, or on **pushdown store**

C2: It always puts symbols on the **top** of the stack

Pushdown Automata PDA

Computational Model of Pushdown Automata PDA



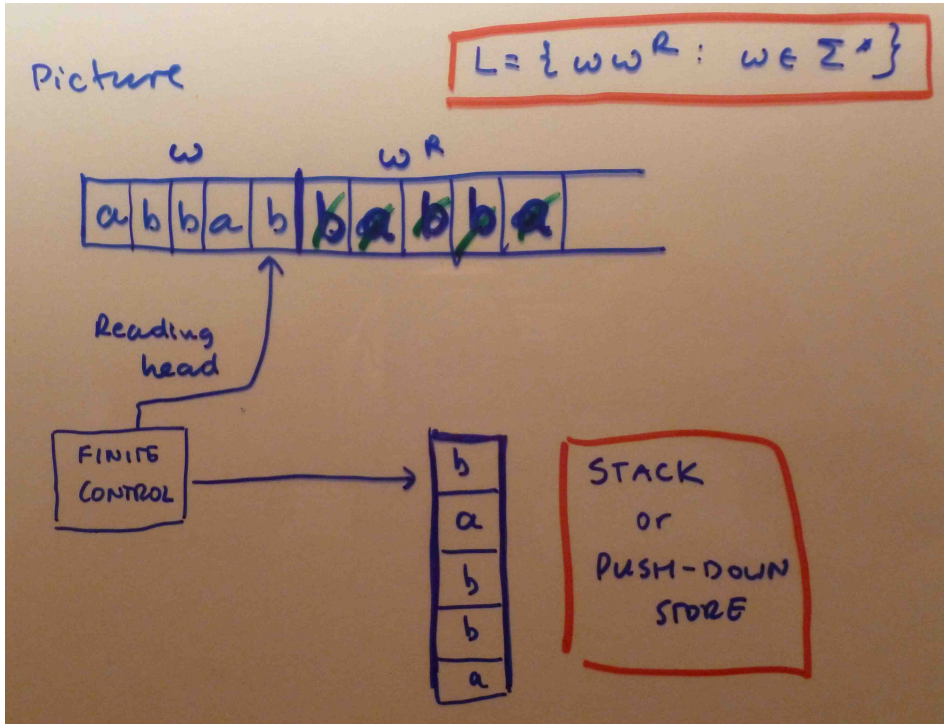
C3: symbols could be **removed** from the **top** of the stack and can be checked against the input

C4: Word is **accepted** when it has been read, **stack** is empty and automaton is in a final state

Pushdown Automata PDA

Pushdown Automata for the context-free language

$L == \{ww^R : \quad w \in \{a,b\}^*\}$



Idea: Automata will read abbab putting its reverse babba on the **stack** from down -to- up

It will **stop** nondeterministically and start to **compare** the stack content with the rest of the input removing content of the stack

PD Automata and CF Grammars

Goal

Our goal now is to **prove** a theorem similar to the theorem for finite automata establishing **equivalence** of regular languages

and finite automata, i.e. we want now to prove the following

Main Theorem

The class of languages **accepted** by pushdown automata **is exactly** the class of Context-free languages

It means that we want to find best way to **define** pushdown automaton order to achieve this goal

PD Automata and CF Grammars

Definition Idea

We have constructed, for any regular grammar G a **finite automaton** M such that

$$L(G) = L(M)$$

by **transforming** any rule $A \to wB$ into a corresponding transition $(A, w, B) \in \Delta$ of M that said:
" in state A **read** w and **move** to B "

We extend this idea to non-regular **rules** and pushdown automata as follows

Pushdown Automata PDA

Given a context-free grammar G and a rule

$$A \to aBb \quad \text{for} \quad a, b \in \Sigma, \quad A, B \in V - \Sigma$$

We now **translate** it to a corresponding transition
(to be defined formally) of a PD automata M that says:

M in state A **reads** a, **puts** b on **stack** and **goes** to state B
Later, the symbols on the stack can be **removed** and
checked agains the input when needed

Word is **accepted** when it has been read, **stack** is empty and
automaton is in a final state

PDA - Mathematical Model

Definition

A Pushdown Automata is a sextuple

$$M = (K,\ \Sigma,\ \Gamma,\ \ \Delta,\ s,\ F), \quad \text{where}$$

K is a finite set of **states**

Σ as an alphabet of **input symbols**

Γ as an alphabet of **stack symbols**

$s \in K$ is the **initial state**

$F \subseteq K$ is the set of **final states**

Δ is a **transition relation**

$$\Delta \subseteq (K \times \Sigma^* \times \Gamma^*) \times (K \times \Gamma^*)$$

Δ is a **finite** set

Transition Relation

Given a PDA

$$M = (K,\ \Sigma,\ \Gamma,\ \ \Delta,\ s,\ F)$$

We denote elements of **stack alphabet** by

$$\alpha,\ \beta,\ \gamma, \ldots$$

with indices if necessary

We usually use different symbols for $K,\ \Sigma$, i.e. we assume that $K \cap \Sigma = \emptyset$

Pushdown automata is **nondeterministic**,

Δ may be **not** a function

Transition Relation

Consider $M = (K,\ \Sigma,\ \Gamma,\ \Delta,\ s,\ F)$ with

$$\Delta \subseteq (K \times \Sigma^* \times \Gamma^*) \times (K \times \Gamma^*)$$

and let an element

$$((p,\ u,\ \beta),\ (q,\ \gamma)) \in \Delta$$

This means that the automaton M in the **state** p with β to the top of the stack,

reads u from the input,

replaces β by γ on the top of the stack, and

goes to state q

Special Transitions

Given a transition

$$((p,\ u,\ \beta),\ (q,\ \gamma)) \in \Delta$$

Here are some spacial cases, i.e some **special transitions** that operate on the stack

Push a - **adds** symbol a **to** the top of the stack

$$((p,\ u,\ e),\ (q,\ a)) \qquad \textbf{push}\ a$$

Pop a - **removes** symbol a **from** the top of the stack

$$((p,\ u,\ a),\ (q,\ e)) \qquad \textbf{pop}\ a$$

Configuration and Transition

In order to define a notion of —bf computation of M on an input string $w \in \Sigma^*$ we introduce, as always, a notion of a **configuration** and **transition** relation

A **configuration** is any tuple

$$(q, w, \gamma) \in K \times \Sigma^* \times \Gamma^*$$

where $q \in K$ represents a current state of M and $w \in \Sigma^*$ is unread part of the input, and γ is a content of the stack read top-down

Configuration and Transition

The **transition relation** acts between two **configurations** and hence $\vdash_M$ is a certain binary relation defined on $K \times \Sigma^* \times \Gamma^*$, i.e.

$$\vdash_M \subseteq (K \times \Sigma^* \times \Gamma^*)^2$$

Formal definition follows

Transition Relation Definotion

Definition

Given a push down automaton

$$M = (K,\ \Sigma,\ \Gamma,\ \ \Delta,\ s,\ F))$$

A binary relation $\vdash_M \subseteq (K \times \Sigma^* \times \Gamma^*)^2$ is a **transition relation** if and only if the following holds

For any $p,\ q\ \in K,\ \ u, x \in \Sigma^*,\ \ \alpha,\ \beta,\ \gamma$

$$(p,\ ux,\ \beta\alpha) \vdash_M (q,\ x,\ \gamma\alpha)$$

if and only if

$$((p,\ u,\ \beta),\ (q,\ \gamma)) \in \Delta$$

Language L(M)

We **denote** as usual, the reflexive, transitive closure of the **transition relation** $\vdash_M$ by $\vdash_M{}^*$ and define, as usual the language L(M) as follows

$$L(M) = \{w \in \Sigma^* : (s, w, e) \vdash_M{}^* (p, e, , e) \text{ for certain } p \in F\}$$

and we say that

$$M \textbf{ accepts } w \in \Sigma^* \text{ if and only if } w \in L(M)$$

Language L(M)

We say it In plain English:

M **accepts** $w \in \Sigma^*$ if and only if there is a **computation** in M such that **it starts** with w and with empty **stack**
(i.e. it starts with $(s,\ w,\ e)$)
and **it ends** in a final state after reading w and emptying all of the **stack**
(it ends with $(p,\ e,\ e)$ for certain $p \in F$)

Theorem

The class **FA** of finite automata is a proper subset of the class **PDA** of pushdown automata, i.e.

$$\textbf{FA} \subset \textbf{PDA}$$

Proof

We show that every FA automaton is a PDA automaton that operates on an empty stack

Given a FA automaton $M = (K, \Sigma, \delta, s, F)$

We construct PDA automaton

$$M' = (K, \Sigma, \Gamma, \Delta', s, F))$$

where $\Gamma = \emptyset$ and

$$\Delta' = \{((p, u, e), (q, e) : \quad (p, u, q) \in \Delta\}$$

Obviously, L(M) = L(M') and hence we proved that

$$M \approx M'$$

Useful Transitions)

Useful transitions

$((p, u, e), (q, a))$ **push** a

$((p, u, a), (q, e))$ **pop** a

In particular we have the following **compare** transitions:

$((p, a, a), (q, e))$ **compare** and **pop** a

$((p, a, b), (q, e))$ **compare** a with b and **pop** b

compare transition **compares** a on the input with a **or** b on the top of the stack and **pops** them from the stack

Examples and Exercises

Examples of PDA

Example 1

We construct M such that $L = \{wcw^R : \quad w \in \{a, b\}^*\}$

$$M = (K,\ \Sigma,\ \Gamma,\ \ \Delta,\ s,\ F))$$

for $K = \{s, f\},\ \Sigma = \{a, b, c\} = \Gamma,\ F = \{f\}$ and Δ has the following transitions

1. $((s, a, e), (s, a))$
2. $((s, b, e), (s, b))$
3. $((s, c, e), (f, e))$
4. $((f, a, a), (f, e))$
5. $((f, b, b), (f, e))$

Exercise 1 Trace a computation of of M accepting the word abbcbba

Example1

Let's analyze the transitions of Δ

1. ((s, a, e), (s, a)) - **pushes** a remaining in state s

2. ((s, b, e), (s, b)) - **pushes** b remaining in state s

3. ((s, c, e), (f, e)) - **switches** from s to f when sees c

4. ((f, a, a), (f, e)) - **compares** and **pops** a remaining in state f

5. ((f, b, b), (f, e)) - **compares** and **pops** b remaining in state f

Operation of M

1. + **2.** put what M reads from input on the **stack** bottom-up until it reaches c

Example 1

Operation of M

3. M **switches** to the final state leaving the stack untouched

The stack is being build bottom-up so what is on the stack is the **reverse** to the part read, it means to the word w

4. + **5.** **compare** the input located after c with what is located already on the stack and **remove** symbols when they match with the input

M is hence checking whether w from the input before c is equal to w^R

All the last actions are done with M remaining with the final state, so when the **stack is empty** it indicates that $wcw^R \in L(M)$ and that

$$L(M) = \{wcw^R : \quad w \in \{a, b\}^*\}$$

Exercise)

Exercise 1 Trace a computation of of M accepting the word abbcbba

Here it is (Book p.133)

State	Unread Input	Stack	Transition Used
s	$abbcbba$	e	–
s	$bbcbba$	a	1
s	$bcbba$	ba	2
s	$cbba$	bba	2
f	bba	bba	3
f	ba	ba	5
f	a	a	5
f	e	e	4

Observe that M is **deterministic**

Example 2

Example 2 Construct a PD M such that

$$L(M) = \{ww^R : \quad w \in \{a, b\}^*\}$$

M construction goes as in the previous **Example 1**, but now we don't have c as a marker when to end reading w and to **switch** to f and start to **compare** input with the content of the stack

We now **switch** to f **nondeterministically** at any stage of the computation, hoping to find the proper "middle" occupied previously by c

We replace the transition **3.** $((s,\ c,\ e),\ (f,\ e))$ by a new **non-deterministic** transition

3'. ((s, e, e), (f, e)) - **switches** from s to f at any time

Example 2

The new set of transitions Δ is now

1. ((s, a, e), (s, a)) - **pushes** a remaining in state s

2. ((s, b, e), (s, b)) - **pushes** b remaining in state s

3'. ((s, e, e), (f, e)) - **switches** from s to f at any time

4. ((f, a, a), (f, e)) - **compares** and **pops** a remaining in state f

5. ((f, b, b), (f, e)) - **compares** and **pops** b remaining in state f

By analysis as in the previous example we prove that

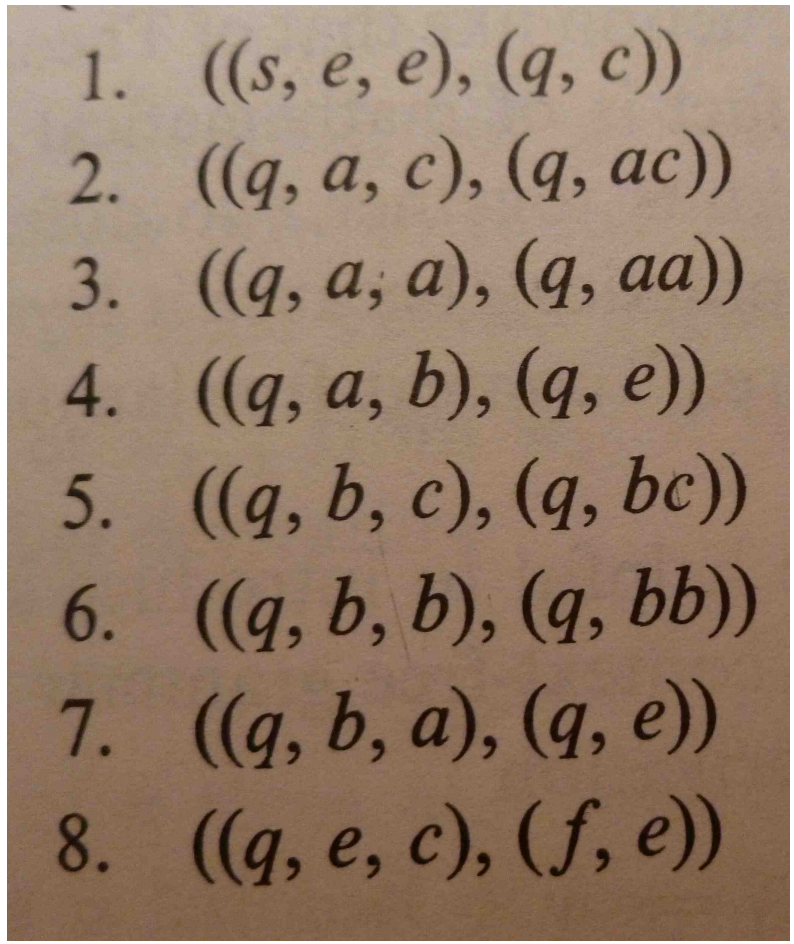
$$L(M) = \{ww^R : \quad w \in \{a, b\}^*\}$$

Exercise 2 Trace **all computation** of M accepting the word abbbba

Many of them will NOT accept $ww^R \in L(M)$ but one for sure will do

Example 3

Example 3 Let M be a PDA , such that its Δ has the following 8 transitions



$L(M) = \{w \in \{a, b\}^* : \quad w$ has the same number of a's and b's $\}$

Exercise 3

Given M

$$M = (K,\ \Sigma,\ \Gamma,\ \ \Delta,\ s,\ F))$$

for $K = \{s, q, f\},\ \Sigma = \{a, b\}, \Gamma = \{a, b, c\},\ F = \{f\}$ and Δ has the 8 transitions from the **Example 3**.

Write a short description of how M operates

Observe that now the input alphabet $\Sigma \neq \Gamma$

We use the symbol $c \in \Gamma$ as a **marker** which we put on the bottom of the stack

We keep either a or b on the stack

A stack of a 's indicate the **excess** of a 's over b 's and

a stack of b 's indicate the **excess** of b 's over a 's thus read

Exercise 3

M when it **reads** an a from input proceeds as follows:

either starts a stack of a 's from the bottom while keeping the bottom **marker** c (transition 2.),

or **pushes** a on the stack of a's (transition 3.),

or **pops** b from the stack of b's (transition 4.)

M when it **reads** an b from input proceeds analogously:

either starts a stack of b 's from the bottom while keeping the bottom **marker** c (transition 5.),

or **pushes** b on the stack of b's (transition 6.),

or **pops** a from the stack of a's (transition 7.)

Exercise 3

Finally, when bottom **marker** c is the only symbol left on the stack,

M may remove it and pass to the **final** state (transition 8.)

If at this point all the input has been **read** the input string is **accepted**

So we have proved that our construction is correct and

$L(M) = \{w \in \{a, b\}^* : \ w$ has the same number of a's and b's $\}$

Exercises

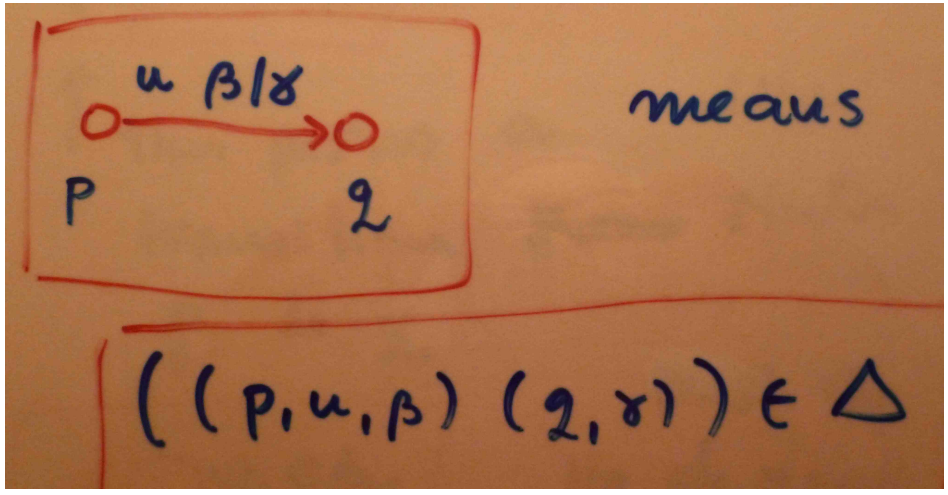
Example 4 Trace a **computation** of M accepting the word abbbabaaa

Here it is

State	Unread Input	Stack	Transition Used	Comments
s	*abbbabaa*	e	–	Initial configuration.
q	*abbbabaa*	c	1	
q	*bbbabaa*	ac	2	Start a stack of a's.
q	*bbabaa*	c	7	Remove one a.
q	*babaa*	bc	5	Start a stack of b's.
q	*abaa*	bbc	6	
q	*baa*	bc	4	
q	*aa*	bbc	6	
q	*a*	bc	4	
q	e	c	4	
f	e	e	8	Accepts.

State Diagrams for Pushdown Automata

Diagram

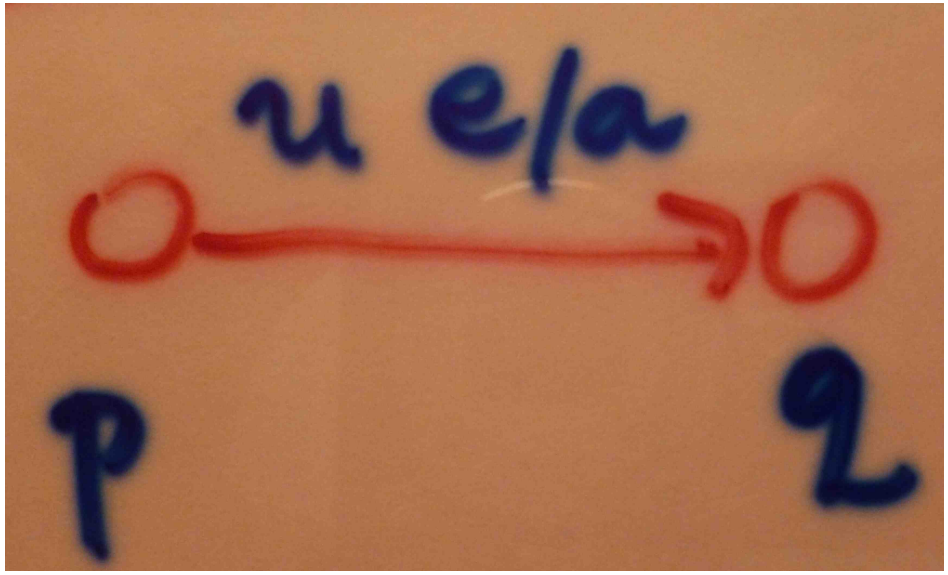


M in state p

1. reads w
2. replaces β by γ on the top of the stack
3. goes to the state q

State Diagrams for Pushdown Automata

Diagram



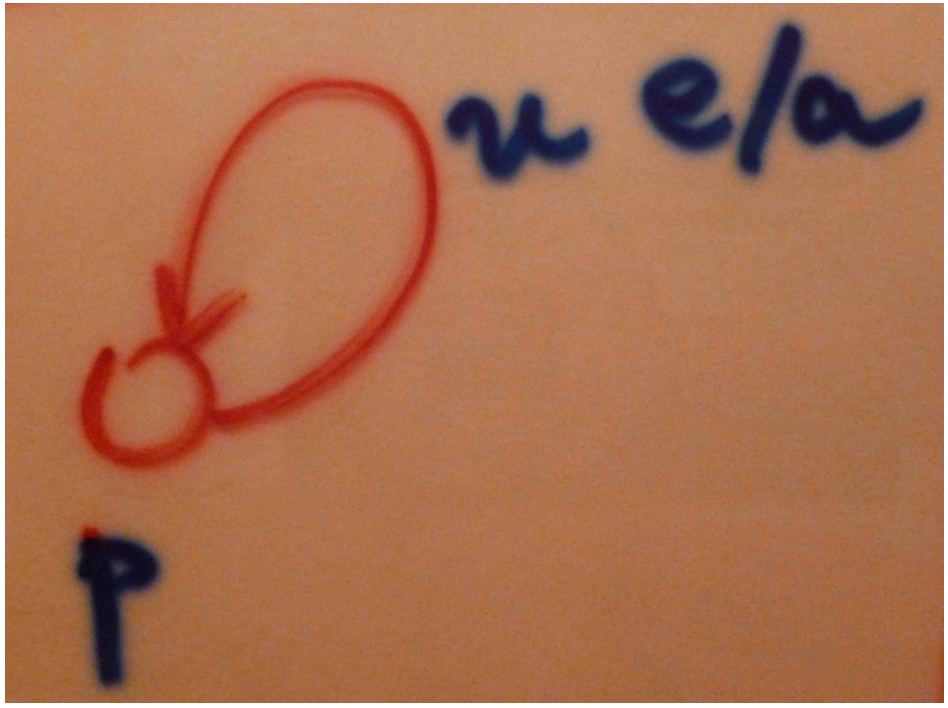
$((p,\ u,\ e),\ (q,\ a)) \in \Delta$ **push** a

M in state p

1. reads u
2. pushes a on the top of the stack
3. goes to the state q

State Diagrams for Pushdown Automata

Diagram



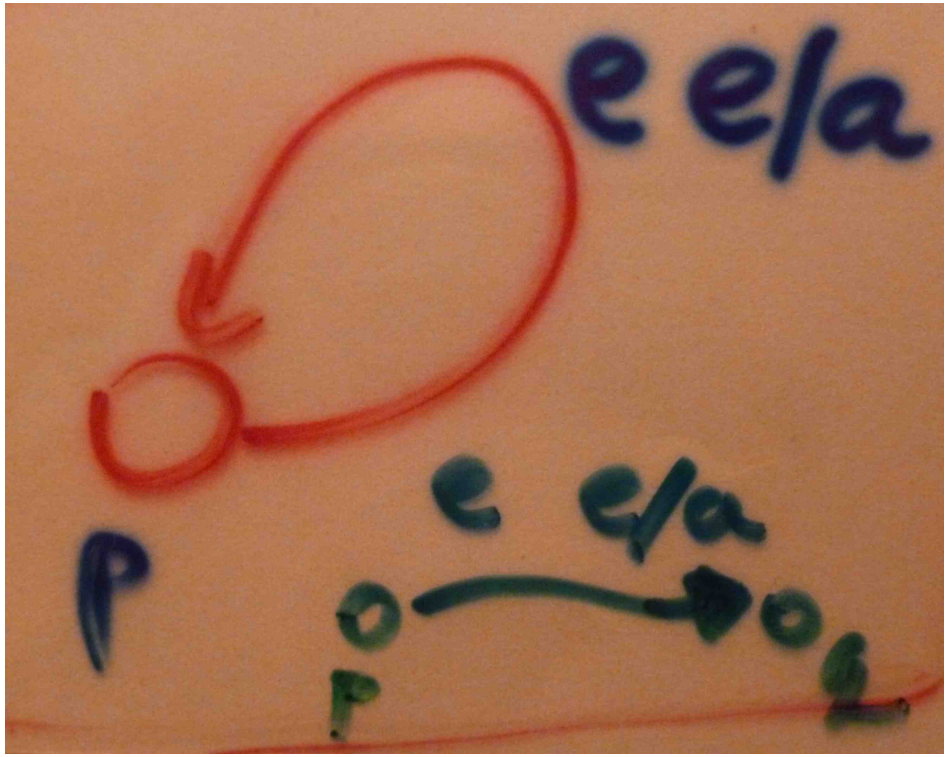
M pushes a with no change of state

In state p

1. reads u
2. pushes a on the top of the stack
3. goes to the state p

State Diagrams for Pushdown Automata

Diagram



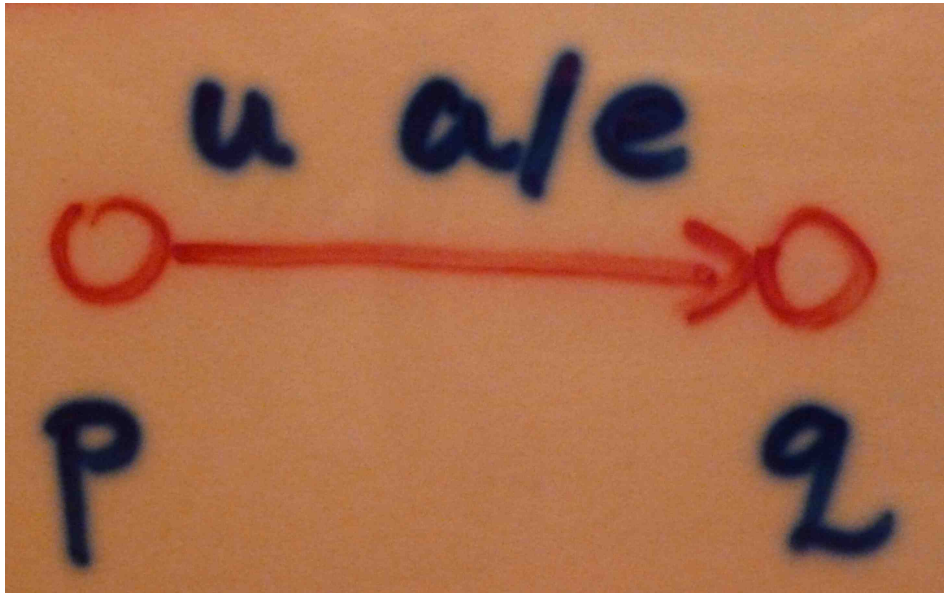
M pushes a with no change of state, reading nothing

In state p

1. reads e
2. pushes a on the top of the stack
3. goes to the state p OR goes to the state q

State Diagrams for Pushdown Automata

Diagram



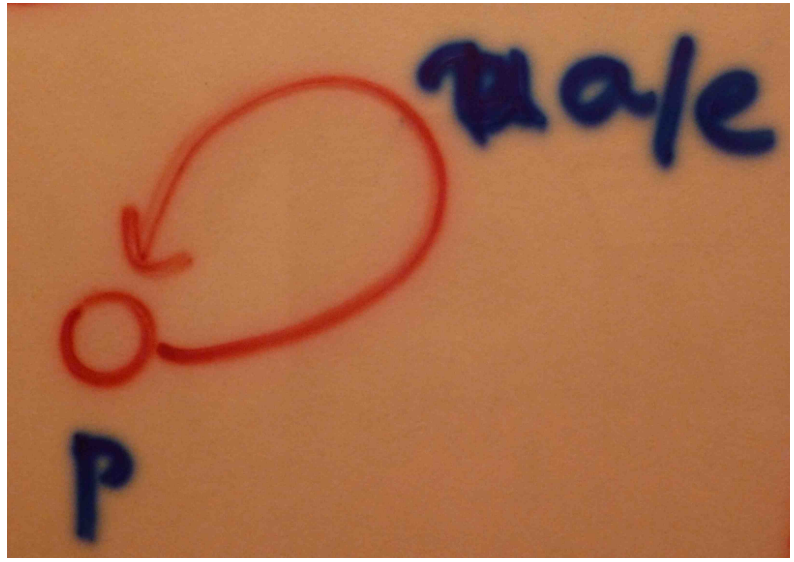
$((p,\ u,\ a),\ (q,\ e)) \in \Delta$ **pop** a

M in state p

1. reads u
2. pops a from the top of the stack
3. goes to the state q

State Diagrams for Pushdown Automata

Diagram



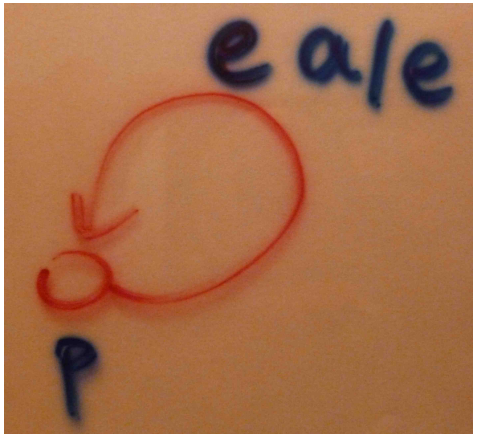
M pushes a with no change of state

In state p

1. reads u
2. pops a from the top of the stack
3. goes to the state p

State Diagrams for Pushdown Automata

Diagram



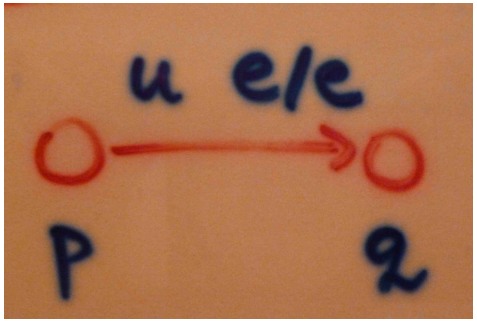
M pushes a with no change of state, reading nothing

In state p

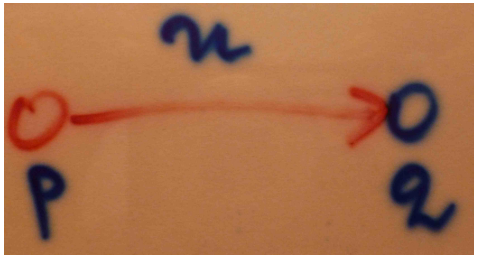
1. reads e
2. pushes a on the top of the stack
3. goes to the state p

State Diagrams for Pushdown Automata

Diagram of PD M'



that imitates the FA colorred M

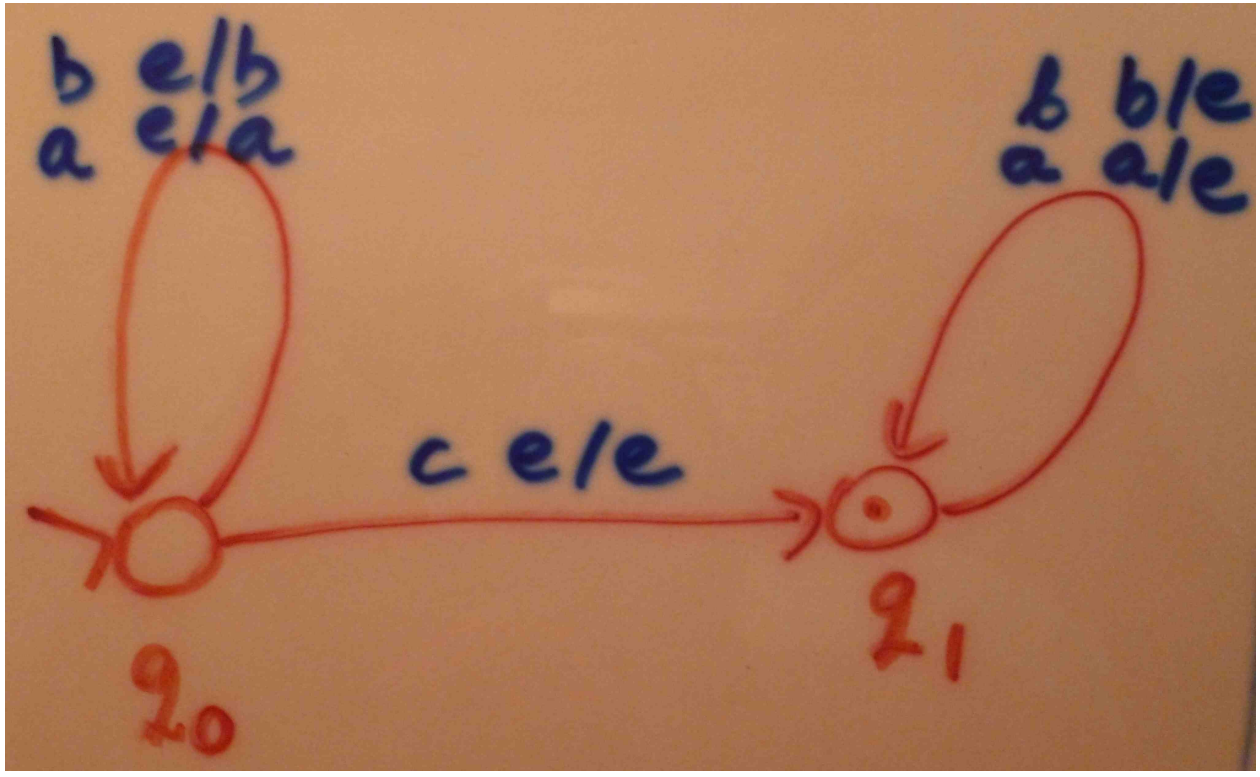


Theorem: Any FA automaton is a PD automaton

Exercises

Exercise 4

Diagram of M

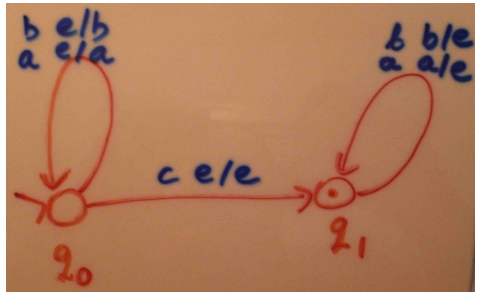


Write components of M and find its language L(M)

Exercises

Exercise 4 Solution

Diagram of M



Δ components are

$((q_0, a, e), (q_0, a))$ - **push** a

$((q_0, b, e), (q_0, b))$ - **push** b

$((q_0, c, e), (q_1, e))$ - **switches** to **final** q_1 when sees c

$((q_1, a, a), (q_1, e))$ - **compares** and **pops** a

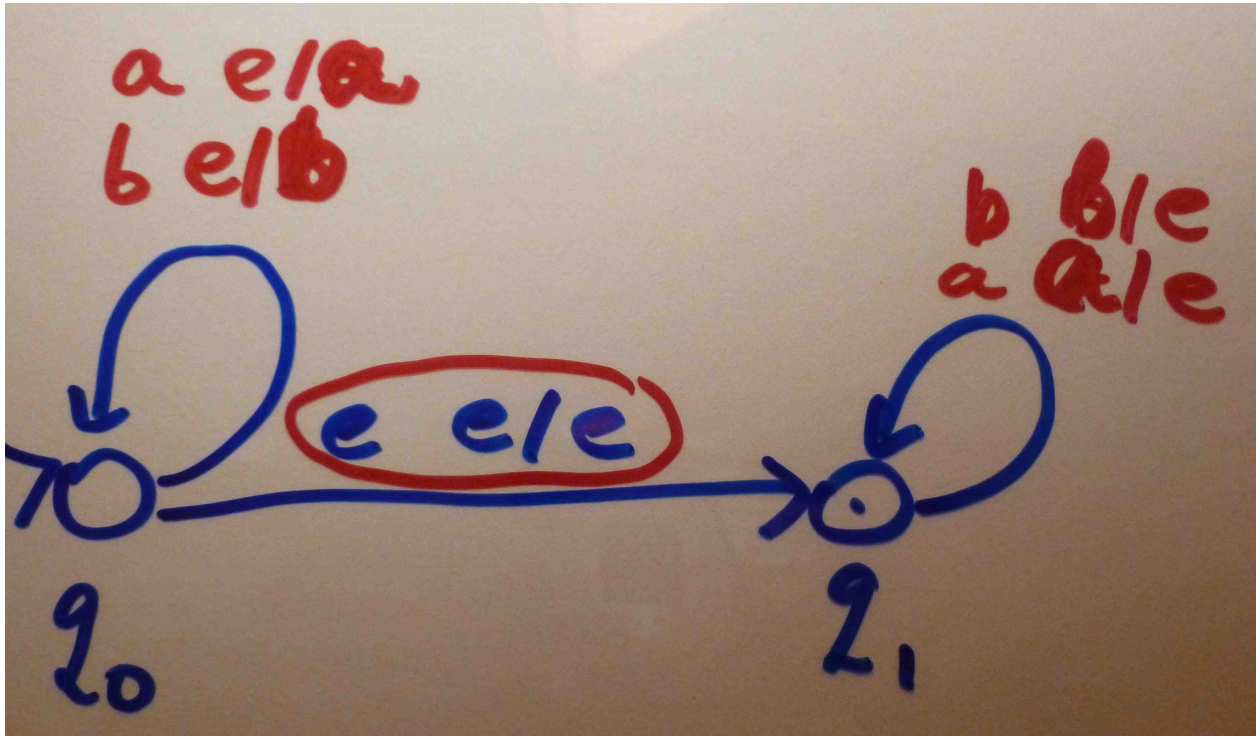
$((q_1, b, b), (q_1, e))$ - **compares** and **pops** b

$$L(M) = \{wcw^R : \quad w \in \{a, b\}^*\}$$

Exercises

Exercise 5

Diagram of M

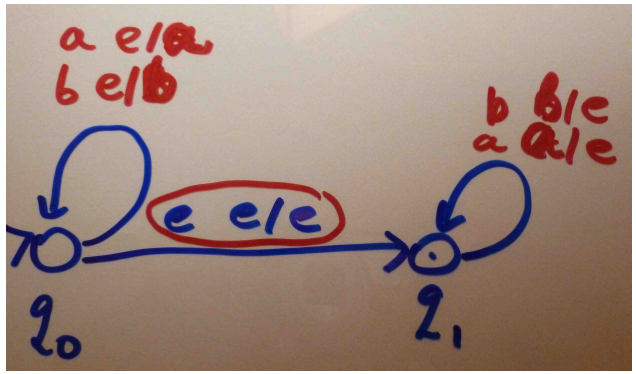


Write components of M and find its language L(M)

Exercises

Exercise 5 Solution

Diagram of M



Δ components are

$((q_0, a, e), (q_0, a))$ - **push** a

$((q_0, b, e), (q_0, b))$ - **push** b

$((q_0, e, e), (q_1, e))$ - **switch** from q_0 to q_1 **at any time**

$((q_1, a, a), (q_1, e))$ - **compare** and **pop** a

$((q_1, b, b), (q_1, e))$ - **compare** and **pop** b

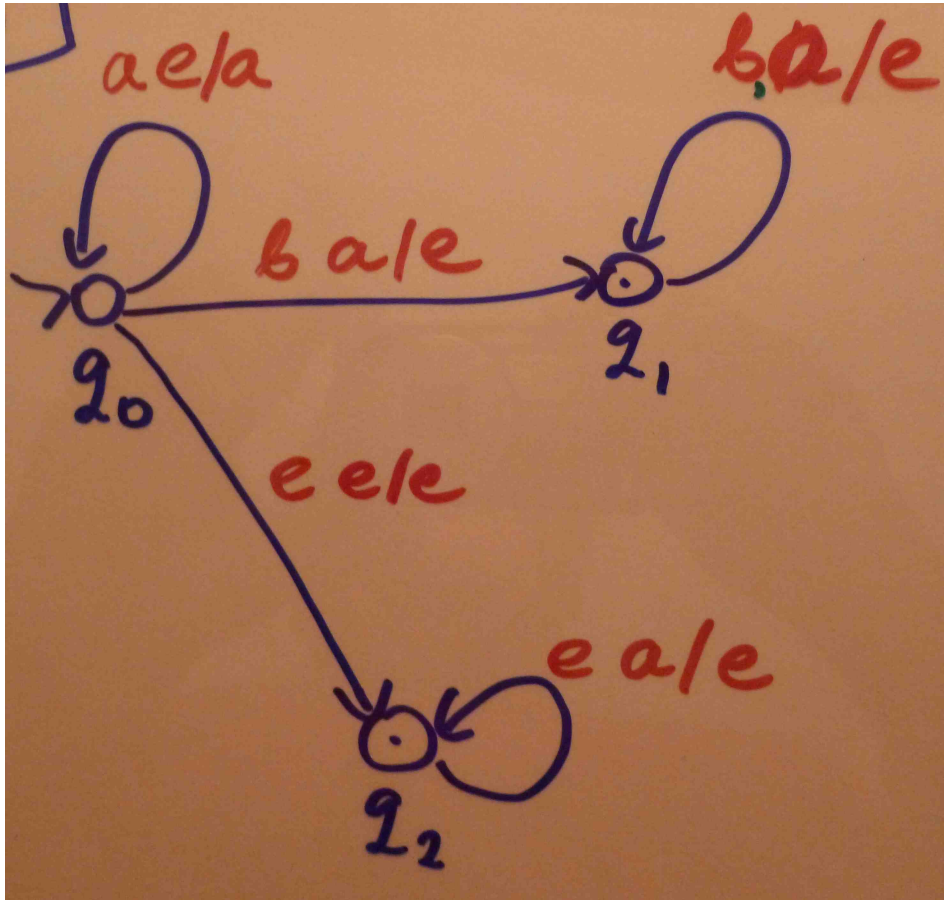
M is **nondeterministic**

$$L(M) = \{ww^R : \quad w \in \{a, b\}^*\}$$

Exercises

Exercise 6

Diagram of M is

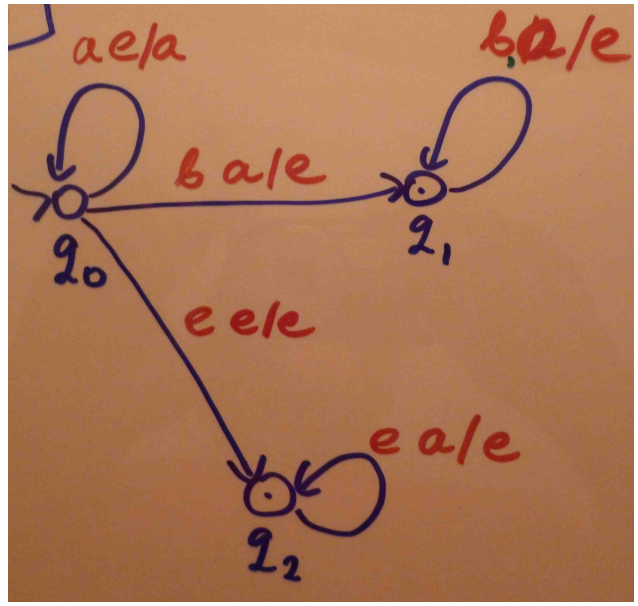


Write components of M and find its language L(M)

Exercises

Exercise 6 Solution

Diagram of M is



Δ components are

$((q_0, a, e), (q_0, a))$ - **push** a when reading a while in q_0

$((q_0, e, e), (q_2, e))$ - **switch** to q_2, write and read nothing

$((q_0, b, a), (q_1, e))$ - when reading b **switch** to q_1, **pop** a

$((q_1, b, a), (q_1, e))$ - when reading b **pop** a while in q_1

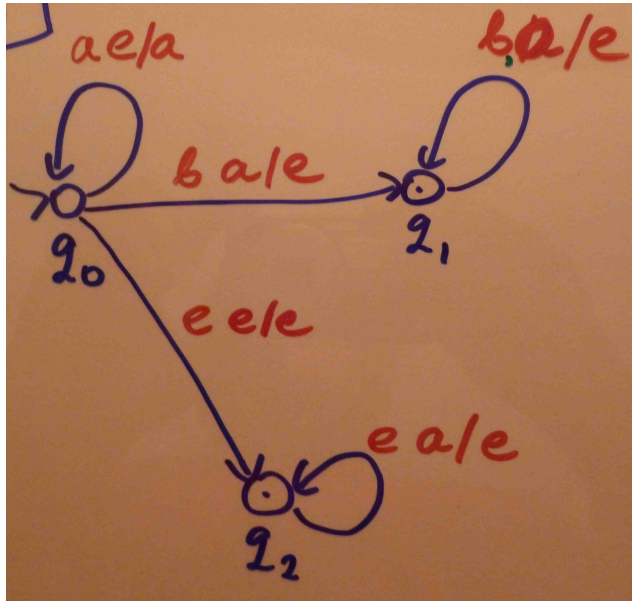
$((q_2, e, a), (q_2, e))$ - while in q_2 **pop** a

M is **nondeterministic**

Exercises

Exercise 6 Solution

Diagram of M is



The language is

$$L(M) = \{a^i : \quad i \geq 0\} \cup \{a^i b^i : \quad i \geq 1\}$$

CHAPTER 3
CONTEXT-FREE LANGUAGES

1. Context Free Grammars
2. Pushdown Automata
3. Pushdown automata and context -free grammars
4. Languages that are and are not context- free

CHAPTER 3
PART 3: Pushdown automata and context -free grammars

PDA Main Theorem

We are going to show now that the PD automaton is exactly what is needed to **accept** arbitrary context-free language, i.e. we are going to prove the following

PDF **Main Theorem**

The class of languages **accepted** by PD automata is exactly the class of context-free languages

PDA Main Theorem Proof

We break the proof of the PDF **Main Theorem** into two parts

Lemma 1

Each context free language is **accepted** by **some** PD automaton

Lemma 2

If a language is **accepted** by a PD automaton, it is a context free language

We prove here only the **Lemma 1**

The proof of and the **Lemma 2** is included in the Book on pages 139 - 142

Proof of Lemma 1

Lemma 1

Each context free language is **accepted** by **some**

PD automaton

Proof

Let $G = (V, \Sigma, R, S)$ be a context-free grammar; we must construct a PD automaton M, such that L(G) = L(M)

M we **construct** has only two states p and q and

M **remains** in state q after its first move

M **uses** V, the set of **grammar** terminals and nonterminals as its stack **alphabet**

Proof of Lemma 1

Given context-free grammar the $G = (V,\ \Sigma,\ R,\ S)$

We **define** corresponding PD automaton as

$$M = (K = \{p, q\},\ \Sigma,\ \Gamma = V,\ \ \Delta,\ p,\ \{q\})$$

where Δ contains the following transitions

1. ((p, e, e), (q, S))
2. ((q, e, A), (q, x)) - for each rule $A \rightarrow x$ in R
3. ((q, c, c), (q, e)) - for each $c \in \Sigma$

The PD automaton M **starts** operation by **pushing** grammar start symbol S on its initially **empty** pushdown store and entering state q (transition **1.**)

Proof of Lemma 1

Remaining Δ transitions are

2. ((q, e, A), (q, x)) - for each rule $A \rightarrow x$ in R

3. $((q, \sigma, \sigma), (q, e))$ - for each $\sigma \in \Sigma$

M on each **subsequent** step

either replaces the topmost **nonterminal** symbol A on the **stack** by the right side x of some rule $A \rightarrow x$ in R (transition of type **2.**)

or pops the topmost symbol of the **stack** provided it matches the input symbol (transition of type **3.**)

Proof of Lemma 1

The **transitions** of M are designed so that the stack during the **accepting** computation mimics a leftmost (regular) grammar **derivation** of the input string

M intermittently carries out a step of such derivation on the **stack** and between such steps it **pops** from the **stack** any terminal symbols that **match** the symbols read from the input

Proof of Lemma 1

Popping the terminals exposes in turn the leftmost **nonterminal**, so that the process can continue **until** the input is read and the stack is **empty**

All these **steps** are carried while M is in the **final** state, hence we get that the input word is **accepted**

We conduct the **formal proof** of the **Lemma 1** by induction on the length of derivation and computation

Example

Example of the construction of the proof of **Lemma 1**

Let G be such that

$$L(G) = \{wcw^R : \quad w \in \{a, b\}^*\}$$

i.e. $G = (V, \Sigma, R, S)$, for $V = \{a, b, c, S\}$, $\Sigma = \{a, b, c\}$, and $R = \{S \to aSa \,|bSb \mid c\}$

The **corresponding** PD automaton is

$$M = (K = \{p, q\}, \ \Sigma = \{a, b, c\}, \ \Gamma = \{a, b, c, S\}, \ \Delta, \ p, \ \{q\})$$

with Δ corresponding to rules of G ,defined as follows

Example

Δ transitions corresponding to rules of G are

$$
\begin{aligned}
\Delta = \{&((p, e, e), (q, S)), && (T1)\\
&((q, e, S), (q, aSa)), && (T2)\\
&((q, e, S), (q, bSb)), && (T3)\\
&((q, e, S), (q, c)), && (T4)\\
&((q, a, a), (q, e)), && (T5)\\
&((q, b, b), (q, e)), && (T6)\\
&((q, c, c), (q, e))\} && (T7).
\end{aligned}
$$

Example

The word $abbcbba \in L(M)$

Here is a computations accepting abbcbba

State	Unread Input	Stack	Transition
p	$abbcbba$	e	-
q	$abbcbba$	S	T1
q	$abbcbba$	aSa	T2
q	$bbcbba$	Sa	T5
q	$bbcbba$	$bSba$	T3
q	$bcbba$	Sba	T6
q	$bcbba$	$bSbba$	T3
q	$cbba$	$Sbba$	T6
q	$cbba$	$cbba$	T4
q	bba	bba	T7
q	ba	ba	T6
q	a	a	T6
q	e	e	T5

CHAPTER 3
CONTEXT-FREE LANGUAGES

1. Context Free Grammars
2. Pushdown Automata
3. Pushdown automata and context -free grammars
4. Languages that are and are not context- free

CHAPTER 3
PART 4: Languages that are and are not context- free

Establishing Context-freeness of Languages

The PDA **Main Theorem** proved the **equivalency** of the following two **views** of context -free languages

1. A language L is context-free if it is **generated** by a context-free grammar (definition)

2. A language L is context-free if it is **accepted** by a push-down automaton

These characterizations enrich our **understanding** of the context-free languages since they provide two different methods for **recognizing** when a language is context free

Establishing Context-freeness of Languages

We examine and provide now further **tools** for establishing context-freeness of languages

We will prove some important **closure properties** of the context free languages **under** language **operations**, as we have done in a case of the regular languages

Establishing Context-freeness of Languages

We also present a **Pumping Theorem** for the context free languages that is similar to the **Pumping Lemmas** we have proved for the regular languages

The **Pumping Theorem** enables us to **show** that certain languages **are not** context-free and we will also examine some of these languages

Closure Theorems

We are going the **prove** the following theorems

Closure Theorem 1

The context-free languages are **closed** under union, concatenation, and Kleene star

Closure Theorem 2

The **intersection** of a context-free language with a regular language is a context-free language

Closure Theorem 3

The context-free languages are **not closed** under intersection and complementation

Closure Theorem 1 Proof

Closure Theorem 1

The context-free languages are **closed** under union, concatenation, and Kleene star

Proof

Let $G_1 = (V_1\ \Sigma_1,\ R_1,\ S_1)$ and $G_2 = (V_2\ \Sigma_2,\ R_2,\ S_2)$

be two CF Grammars

We assume that they have two disjoint sets of nonterminals, i.e. that $(V_1 - \Sigma_1) \cap (V_2 - \Sigma_2) = \emptyset$

Union Closure $G = G_1 \cup G_2$

We construct a grammar $G = G_1 \cup G_2$ as follows

Let S be a new symbol and let

$$G = (V_1 \cup V_2 \cup \{S\},\ \Sigma_1, \cup \Sigma_2,\ R,\ S)$$

Closure Theorem 1 Proof

We define

$$R = R_1 \cup R_2 \cup \{S \rightarrow S_1, \ S \rightarrow S_2\}$$

For the only rules involving S are $S \rightarrow S_1$, $S \rightarrow S_2$ we have that

$$S \underset{G}{\overset{*}{\Rightarrow}} w \quad \text{if and only if} \quad S_1 \underset{G}{\overset{*}{\Rightarrow}} w \quad \text{or} \quad S_2 \underset{G}{\overset{*}{\Rightarrow}} w$$

Since G_1 and G_1 have two disjoint sets of nonterminals this is equivalent to saying that

$$w \in L(G) \ \text{ if and only if } \ w \in L(G_1) \ \text{ or } \ w \in L(G_1)$$

and it proves that

$$L(G) = L(G_1) \cup L(G_2)$$

Closure Theorem 1 Proof

Concatenation $G = G_1 \circ G_2$

We construct a grammar $G = G_1 \circ G_2$ as follows

$$G = (V_1 \cup V_2 \cup \{S\},\ \Sigma_1 \cup \Sigma_2,\ R,\ S)$$

where

$$R = R_1 \cup R_2 \cup \{S \rightarrow S_1 S_2\}$$

For the only rule involving S is $S \rightarrow S_1 S_2$ and G_1 and G_1 have two disjoint sets of nonterminals this is saying that

$w \in L(G)$ if and only if $w = w_1 w_2$ for $w_1 \in L(G_1), w_2 \in L(G_2)$

It proves that

$$L(G) = L(G_1) \circ L(G_2)$$

Closure Theorem 1 Proof

Kleene star $G = G_1{}^*$

We construct a grammar $G = G = G_1{}^*$ as follows

$$G = (V_1 \cup \{S\},\ \Sigma_1,\ R,\ S)$$

where

$$R = R_1 \cup R_2 \cup \{S \rightarrow e,\quad S \rightarrow SS_1\}$$

Observe that we need the rule $S \rightarrow e$ to make sure that $L(G) \neq set$

Obviouly,

$$L(G) = L(G_1)^*$$

Closure Theorems

We are going to **prove** now the following

Closure Theorem 2

The **intersection** of a context-free language with a regular language is a context-free language

Closure Theorem 2 Proof

Closure Theorem 2

The **intersection** of a context-free language with a regular language is a context-free language

Proof

Let R be a regular language and L a context-free language

By FA and PDF **Main Theorems** we have that

$L = L(M)_1$ for some **PD** automaton

$$M_1 = (K_1, \Sigma_1, \Gamma_1, \Delta_1, s_1, F_1))$$

and $R = L(M)_2$ for some **deterministic** automaton

$$M_2 = (K_2, \Sigma_2, \delta, s_2, F_2))$$

Closure Theorem 2 Proof

We construct a PD automaton M in a similar way to the **direct** construction in the proof of the **Theorem** of the **closure** of finite automata under intersection.

Namely, we define M as follows

We define $M = M_1 \cap M_2$ as

$$M = (K,\ \Sigma,\ \Gamma,\ \ \Delta,\ s,\ F))$$

where

$K = K_1 \times K_2, \qquad \Gamma = \Gamma_1, \quad s = (s_1,\ s_2), \quad F = F_1 \times F_2,$

and Δ is defined in such way that it allows the computations of M to be carried by the computations of $M_1,\ M_2$ in **parallel** and a **word** is accepted by M only if it would be **accepted** by both M_1 and M_2

Pumping Lemma
for Context Free Languages

Pumping Lemma

Pumping Lemma

Let G be a context-free grammar

Then there is a number K, depending on G, such that

any word $w \in L(G)$ of length greater than K

can be re-written as

$$w = uvxyz \quad \text{for} \quad v \neq e \quad \text{or} \quad y \neq e$$

and for any $n \geq 0$

$$uv^n xy^n z \in L(G)$$

Not Context-free Languages

We use the **Pumping Lemma** to prove the following

Theorem

The language

$$L = \{a^n b^n c^n : \quad n \geq 0\}$$

is **NOT** **context-free**

Proof

We carry the proof by contradiction

Assume that L is context-free, i.e. that $L = L(G)$ for some context-free grammar G

Let K be a constant for G as specified by the **Pumping Lemma**

and let $n > K/3$

Not Context-free Languages

Then $w = a^n b^n c^n \in L(G)$ has a representation $w = uvxyz$ such that $v \neq e$ or $y \neq e$ and $uv^n xy^n z \in L(G)$ for $i = 0, 1, 2, 3, \ldots$

But this is impossible

for $a^n b^n c^n = uvxyz$ and either v or y contains two symbols from $\{a, b, c\}$, then $uv^2 xy^2 z$ contains a b before an a or a c before a

and if v and y each contains only a's, only b's, or only c's, then $uv^2 xy^2 z$ cannot contain equal number of a's, b's, and c's

This contradiction **ends** the proof

Closure Theorems

Now we are ready to prove that the context-free languagaes are **not closed** under certain operations

Closure Theorem 3

The context-free languages are **not closed** under intersection and complementation

Proof

We divide the proof into proving the following two parts

Part 1

The context-free languages are **not closed** under intersection

Part 2

The context-free languages are **not closed** under complementation

Closure Theorem 3 Proof

Part 1

The context-free languages are **not closed** under intersection

Proof

Assume that the context-free languages are **are closed** under intersection

Observe that both languages

$$L_1 = \{a^n b^n c^m : \quad m, n \geq 0\} \quad \text{and} \quad L_2 = \{a^m b^n c^n : \quad m, n \geq 0\}$$

are **context-free**, so the language $L_1 \cap L_2$ must be **context-free**, but

$$L_1 \cap L_2 = \{a^n b^n c^n : \quad n \geq 0\}$$

and we have proved that $L = \{a^n b^n c^n : \quad n \geq 0\}$ is NOT context-free. **Contradiction**

Closure Properties

Part 2

The context-free languages are **not closed** under complementation

Proof

Assume that the context-free languages are **are closed** under complementation

Take any two context-free languages L_1, L_2

Then the language

$$L_1 \cap L_2 = \Sigma^* - ((\Sigma^* - L_1) \cup (\Sigma^* - L_2))$$

would be context-free, what **contradicts** just proved that fact

that the context-free languages are **not closed** under intersection

Not Context-free Languages

Theorem 4

The following languages are **NOT** **context-free**

$$L_1 = \{a^i b^j a^i b^j : \quad i, j \geq 0\}$$

$$L_2 = \{a^p : \quad \text{p is prime}\}$$

$$L_3 = \{a^{n^2} : \quad n \geq 0\}$$

$$L_4 = \{www : \quad w \in \{a, b\}^*\}$$

Proof

By the **Pumping Lemma**

Power of Pumping Lemma

We use the **Pumping Lemma** to prove that many languages **are not** context-free

Unfortunately, there are some very simple non-context-free languages which cannot be shown **not to be** context-free by a direct application of the **Pumping Lemma**

One such example is

$$L = \{a^m b^n : \text{ either } m > n, \text{ or } m \text{ is prime and } n \geq m\}$$

We **prove** L to be **not** context-free using the following **Parikh Theorem**

Parikh Theorem

Parikh Theorem

If L is context-free, then $\Psi(L)$ is semilinear,
where $\Psi(L)$ is a certain well defined set of of n-tuples of natural numbers associated with L

Hence to prove a language to be not context -free we use Parikh Theorem in a following equivalent form

Parikh Theorem

If $\Psi(L)$ **is not** semilinear, then L **is not** context-free

Parikh Theorem

We also use **Parikh Theorem** to show the following interesting property of contex-free languages

Theorem 6

Every contex-free language over a one symbol alphabet is **regular**

Context-free/ NOT Context-free

Exercise

Prove that the language

$$L = \{ww : \quad w \in \{a, b\}^*\}$$

is NOT context-free

Hint

We know that

$$L_1 = \{a^i b^j a^i b^j : \quad i, j \geq 0\}$$

is **NOT context-free**

Context-free/ NOT Context-free

Solution

Assume that $L = \{ww : \quad w \in \{a,b\}^*\}$ is context-free

Then the language

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

is context-free by **Closure Theorem 2** that says:

"The **intersection** of a context-free language with a regular language is a context-free language ". But the language

$$\{ww : \quad w \in \{a,b\}^*\} \cap a^*b^*a^*b^* = \{a^ib^ja^ib^j : \quad i,j \geq 0\}$$

is NOT context-free by **Theorem 5**

Contradiction

Context-free/ NOT Context-free

We have proved by constructing a PD automaton

and applying the **Main Theorem** that the language

$L = \{w \in \{a, b\}^* : w$ has the same number of a's and b's $\}$

is context- free

We use Pumping Lemma to prove that the languages

$L = \{w \in \{a, b, c\}^* : w$ has the same number of a's, b's and c's $\}$

$$L = \{a^p b^n : \ p \in Prime, \ n > p\}$$

are **NOT context- free**