INTRODUCTION TO THE THEORY OF COMPUTATION
LECTURE NOTES

Professor Anita Wasilewska
Stony Brook University (SUNY at Stony Brook)

Course Text Book

ELEMENTS OF THE THEORY OF COMPUTATION

Harry R. Lewis, and Christos H. Papadimitriou

Prentice Hall, 2nd Edition

Chapter 3
Context-free Languages

LECTURE SLIDES

Chapter 3
Context-free Languages

Slides Set 1

PART 1: Context-free Grammars

Slides Set 2

PART 2: Pushdown Automata

PART 3: Pushdown automata and context -free grammars

Slides Set 3

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

Chapter 3
Context-free Languages

Slides Set 1

PART 1: Context-free Grammars

Context-free Grammars

Finite Automata are formal language **recognizers**

- they are devises that **accept** valid strings

Context-free Grammars are a certain type of formal **language generators**

- they are devises that **produce** valid strings

Context-free Grammars

Such a language generator **devise** begins,
when given a start symbol, to **construct** a string

Its operation is not completely determined from he beginning
but is nevertheless limited by a **finite** set of rules

The process **stops**,
and the **devise** outputs a completed string

The language defined by the **devise** is the set of all strings
it can **produce**

Context-free Grammar Definition

Definition

A Context-Free Grammar is a quadruple

$$G = (V,\ \Sigma,\ R,\ S)$$

where

V is an **alphabet**

$\Sigma \subseteq V$ is a set of **terminals**

$V - \Sigma$ is the set of **nonterminals**

R is a finite set of **rules**

$$R \subseteq (V - \Sigma) \times V^*$$

$S \in V - \Sigma$ is the **start symbol**

Context-free Grammar Definitions

The alphabet V consists of two disjoint parts: **nonterminals** $V - \Sigma$ and **terminals** Σ, i.e.

$$V = (V - \Sigma) \cup \Sigma$$

Notations

We use symbols of capital letters, with indices if necessary for **nonterminals** $V - \Sigma$, i.e.

$$A,\ B, C, S, T, X, Y, \ldots A_i,\ \cdots \in V - \Sigma$$

The **terminal** alphabet Σ is as in case of the finite automata, the alphabet the words of the language are made from and we denote its elenets, as before by small letters, or symbol σ, with indices if necessary, i.e.

$$a,\ b, c,\ \sigma, \ldots a_i, \ldots \sigma_i,\ \cdots \in \Sigma$$

Context-free Grammar Definition

Notations

By definition, the set of **rules** R of a context-free grammar G is a **finite** set such that

$$R \subseteq (V - \Sigma) \times V^*$$

It means that $R = \{(A, u) : \quad A \in (V - \Sigma) \text{ and } u \in V^*\}$

where A is a **nonterminal** and $u \in V^*$ is a string that contains some **terminals** and **nonterminals**

We write

$$A \to_G u \quad \text{or} \quad A \to u \quad \text{for any} \quad (A, u) \in R$$

Context-free Grammar Definition

Given a **context-free grammar**

$$G = (V,\ \Sigma,\ R,\ S)$$

Definition

For any $u, v \in V^*$, we define a **one step derivation**

$$u \underset{G}{\Rightarrow} v$$

of v from u as follows

$\Rightarrow_G$ if and only if there are $A, x, y, v' \in V^*$ such that

1. $A \in V - \Sigma$
2. $u = xAy$ and $v = xv'y$
2. A $\rightarrow$ v' for certain $r \in R$

Context-free Grammar Definition

One step derivation in plain words:

$u \Rightarrow_G v$ if and only if we obtain v from u by a **direct** application of one rule $r \in R$

Definition of the language L(G) **generated** by G

$$L(G) = \{w \in \Sigma^* : S \overset{*}{\underset{G}{\Rightarrow}} w\}$$

where $\Rightarrow^*_G$ is a transitive, reflexive closure of $\Rightarrow_G$

Context-free and Regular Languages

Given a **derivation** of $w \in \Sigma$ in G

$$S \overset{*}{\underset{G}{\Rightarrow}} w$$

We write is in detail (by definition of $\Rightarrow^*_G$) as

$$S \underset{G}{\Rightarrow} w_1 \underset{G}{\Rightarrow} w_2 \underset{G}{\Rightarrow} \ldots \underset{G}{\Rightarrow} w \quad \text{for} \quad w_i \in V^*,\ w \in \Sigma$$

or in a simpler form when G is known as

$$S \Rightarrow w_1 \Rightarrow w_2 \Rightarrow \ldots \Rightarrow w$$

Context-free Grammar Example

Example

Consider a grammar

$$G = (V,\ \Sigma,\ R,\ S)$$

for $V = \{S, a, b\}$, $\Sigma = \{a, b\}$ and

$R = \{r1 :\ S \rightarrow aSb,\quad r2 :\ S \rightarrow e\}$

Here are some **derivations** in G

D1 $S \Rightarrow e$ so we have that

$$e \in L(G)$$

D2 $S \Rightarrow^{r1} aSb \Rightarrow^{r1} aaSbb \Rightarrow^{r2} aaebb$

and we have that

$$aabb \in L(G)$$

D3 $S \Rightarrow^{r1} aSb \Rightarrow^{r1} aaSbb \Rightarrow^{r1} aaaSbbb \Rightarrow^{r2} aaaebbb$

and we have that

$$a^3b^3 \in L(G)$$

We prove, by induction on the length of derivation that

$$L(G) = \{a^n b^n : \quad n \geq 0\}$$

Context-free and Regular Languages

Definition

A language L is a **context-free language** if and only if there is a context-free grammar G such that

$$L = L(G)$$

We have just proved

Fact 1

The language $L = \{a^n b^n : \ n \geq 0\}$ is **context-free**

Context-free and Regular Languages

Observe that we also proved that the language

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

is **not regular**

Denote by RL the class of all regular languages and by CFL the class of all contex-free languages

Context-free and Regular Languages

Hence we have **proved**

Fact 2 $RL \neq CFL$

Our next GOAL will be **to prove** the following

Theorem

The the class of all regular languages is a proper subset of the class of all contex-free languages, i.e.

$$RL \subset CFL$$

Exercises

Exercise 1

Show that the regular language $L = \{a^* : \ a \in \Sigma\}$

is **context-free**

Proof By definition of **context-free** language we have to construct a CF grammar G such that

$$L = L(G) \quad \text{i.e} \quad L(G) = \{a^* : \ a \in \Sigma\}$$

Here is the grammar $G = (V, \Sigma, R, S)$

for $V = \{S, a\}$, $\Sigma = \{a\}$ and

$R = \{ S \rightarrow aS, \quad S \rightarrow e\}$

We write rules of R in a shorter way as

$$R = \{ S \rightarrow aS \mid e\}$$

Exercises

Here is a **derivation** in G:

$$S \Rightarrow aS \Rightarrow aaS \Rightarrow aaaS \Rightarrow aaae$$

and we have that

$$aaaa \in L(G)$$

We prove, by induction on the length of derivation that

$$L(G) = \{a^* : \quad a \in \Sigma\}$$

Exercises

Exercise 2

Show that the NOT regular language

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

is **context-free**

Exercises

We construct a context-free grammar G such that

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

as follows

$$G = (V, \Sigma, R, S)$$

where $V = \{a, b, S\}, \ \Sigma = \{a, b\}$

$$R = \{S \to aSa \,|bSb \mid e\}$$

Derivation example:

$S \Rightarrow aSa \Rightarrow abSba \Rightarrow abbSbba \Rightarrow abbbba$

We prove, by induction on the length of derivation that

$ww^R \in L(G)$ for any $w \in \Sigma^*$

Exercises

Remark

The set of rules

$$R = \{S \rightarrow aSa \mid aSb \mid c\}$$

defines a grammar G with the language

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

Exercise 3

Show that the NOT regular language

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

is **context-free**

Exercises

We construct a context-free grammar G such that

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

as follows

$$G = (V, \Sigma, R, S)$$

where $V = \{a, b, S\}, \ \Sigma = \{a, b\}$

$$R = \{S \to aSa \,|bSb \mid a \mid b \mid e\}$$

Derivation example:

$$S \Rightarrow aSa \Rightarrow abSba \Rightarrow ababa$$

We check:

$$(ababa)^R = ((ab)a(ba))^R = (ba)^R a^R (ab)^R = ababa$$

Exercises

Grammar correctness justification

Observe that the rules

$$S \to aSa \mid bSb \mid e$$

generate the language (as was proved in Example 2)

$$L_1 = \{ww^R : \ w \in \Sigma^*\}$$

Adding additional rules $S \to a \mid b$ we get that

$$L(G) = L_1 \cup \{waw^R : \ \ w \in \Sigma^*\} \cup \{wbw^R : \ w \in \Sigma^*\}$$

Hence

$w \in L(G)$ iff $w = xx^R$ or $w = xax^R$ or $w = xbx^R$

Exercises

Hence

$w \in L(G)$ iff $w = xx^R$ or $w = xax^R$ or $w = xbx^R$

We show now that in each case $w = w^R$, i.e.

we **prove** that

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

as follows

Case 1: $w = xx^R$

We evaluate

$w^R = (xx^R)^R = (x^R)^R x^R = xx^R = w$

We used property: $(x^R)^R = x$

Exercises

Case 2: $w = xax^R$

We evaluate

$w^R = (xax^R)^R = (x^R)^R a^R x^R = xax^R = w$

We used properties $(x^R)^R = x$ and $a^R = a$

Case 3: $w = xbx^R$

We evaluate

$w^R = (xbx^R)^R = (x^R)^R b^R x^R = xbx^R = w$

This **ends the proof**

Regular Grammars

Definition

A context-free grammar

$$G = (V,\ \Sigma,\ R,\ S)$$

is called **regular**, or **right-linear** if and only if

$$R \subseteq (V - \Sigma) \times \Sigma^*((V - \Sigma) \cup \{e\})$$

Regular Grammars

That is, a regular (right-linear) grammar is a context-free grammar such that the right-hand side of every **rule** contains at most one **nonterminal**, which if present, must be the last symbol in the string

The rules must have a form

$$A \to wB, \quad A \to w \quad \text{for any} \quad A, B \in V - \Sigma, \quad w \in \Sigma^*$$

Remark

We didn't say $A \neq B$!

Exercise 4

Given a **regular grammar** $G = (V, \Sigma, R, S)$, where

$V = \{a, b, S, A\}$, $\Sigma = \{a, b\}$

$$R = \{S \to aS \mid A \mid e,\ \ A \to abA \mid a \mid b\}$$

1. Construct a finite automaton M, such that $L(G) = L(M)$

Solution

We construct a non-deterministic finite automaton

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

$$K = (V - \Sigma) \cup \{f\},\ \Sigma = \Sigma, s = S,\ \ F = \{f\}$$

$$\Delta = \{(S, a, S), (S, e, A), (S, e, f), (A, ab, A), (A, a, f), (A, b, f)\}$$

Exercise 4

2. Write a computation of M that leads to the **acceptance** of the string *aaaababa*

Compare it with a derivation of the same string in *G*

Solution

The accepting computation of M is:

$$(S, aaaababa) \vdash_M (S, aaababa) \vdash_M (S, aababa) \vdash_M (S, ababa)$$

$$\vdash_M (A, ababa) \vdash_M (A, aba) \vdash_M (A, a) \vdash_M (f, e)$$

Corresponding **derivation** in *G* is:

$$S \Rightarrow aS \Rightarrow aaS \Rightarrow aaaS \Rightarrow aaaA \Rightarrow aaaabA$$

$$\Rightarrow aaaababA \Rightarrow aaaababa$$

Regular and Context-free Languages

We are going to prove the following **theorem** that establishes the **relationship** between the Regular Languages and Regular Grammars

L-G Theorem

Language L is **regular** if and only if there exists a **regular grammar** G such that

$$L = L(G)$$

Regular and Context-free Languages

By definition, any regular grammar is context free and hence generates a context-free language and we get that

R - CF Theorem

The the class RL of all regular languages is a proper subset of the class CFL of all context-free languages, i.e.

$$RL \subset CFL$$

Proof of L-G Theorem

L-G Theorem

Language L is **regular** if and only if there exists a **regular** grammar G such that

$$L = L(G)$$

Proof part 1

Suppose that L is **regular**; then L is accepted by a **deterministic** finite automaton

$$M = (K,\ \Sigma,\ \delta,\ s,\ F)$$

We **construct** a regular grammar G as follows

$$G = (V,\ \Sigma,\ R,\ S)$$

for $V = \Sigma \cup K, \quad S = s$

$$R = \{q \rightarrow ap : \quad \delta(q, a) = p\} \cup \{q \rightarrow e : \quad q \in F\}$$

Proof of L-G Theorem

We need now to show that $L(M) = L(G)$

Observe that the rules of G are designed to mimic exactly the moves of M

For any $\sigma_1, \ldots, \sigma_n \in \Sigma$ and $p_0, \ldots, p_n \in K$

$$(p_0, \sigma_1, \ldots, \sigma_n) \vdash_M (p_1, \sigma_2, \ldots, \sigma_n) \vdash_M \ldots \vdash_M (p_n, e)$$

if and only if

$$p_0 \overset{*}{\underset{G}{\Rightarrow}} \sigma_1 p_1 \overset{*}{\underset{G}{\Rightarrow}} \sigma_1\sigma_2 p_2 \ \ldots \ \overset{*}{\underset{G}{\Rightarrow}} \sigma_1\sigma_2 \ \ldots \ \sigma_n p_n$$

This is because

$$\delta(q, a) = p \quad \text{if and only if} \quad q \ \rightarrow \ ap$$

Proof of L-G Theorem

We **prove** now that $L(M) \subseteq L(G)$

Suppose that $w \in L(M)$

Then for some $p \in F$,

$$(s,\ w) \vdash_M{}^* (p, e)$$

but

$$\delta(q, a) = p \quad \text{if and only if} \quad q \ \rightarrow \ ap$$

is

$$S \overset{*}{\underset{G}{\Rightarrow}} w$$

so $w \in L(G)$

Proof of L-G Theorem

We **prove** now that $L(G) \subseteq L(M)$

Suppose that $w \in L(G)$

Then

$$S \overset{*}{\underset{G}{\Rightarrow}} w \quad \text{that is} \quad s \overset{*}{\underset{G}{\Rightarrow}} w$$

The rule **used** at the last step of the derivation must have been of the form

$$p \rightarrow e \quad \text{for some} \quad p \in F$$

and so

$$s \overset{*}{\underset{G}{\Rightarrow}} wp \underset{G}{\Rightarrow} w$$

But then

$$(s,\ w) \vdash_M{}^* (p, e)$$

and so $w \in L(M)$ and

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

Proof of L-G Theorem

Proof part 2

Let now G be any **regular** grammar

$$G = (V,\ \Sigma,\ R,\ S)$$

We define a **nondeterministic** automaton M such that

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

as follows

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

$K = (V - \Sigma) \cup \{f\}$ where f is a new element

$$s = S, \quad F = \{f\}$$

Proof of L-G Theorem

The set Δ of transitions is

$$\Delta = \{(A, w, B) : \quad A \to wB \in R;\ A, B \in V - \Sigma, \quad w \in \Sigma^*\}$$

$$\cup\{(A, w, f) : \quad A \to w \in R;\ A, B \in V - \Sigma, \quad w \in \Sigma^*\}$$

Once again, derivations are mimicked by the moves, i.e, for any

$A_1, \ldots, A_n \in V - \Sigma, \quad w_1, \ldots w_n \in \Sigma^*$

$$A_1 \Rightarrow_G \ w_1 A_2 \ \Rightarrow_G \cdots \Rightarrow_G \ w_1 \ldots w_{n-1} A_n \Rightarrow_G \ w_1 \ldots w_n$$

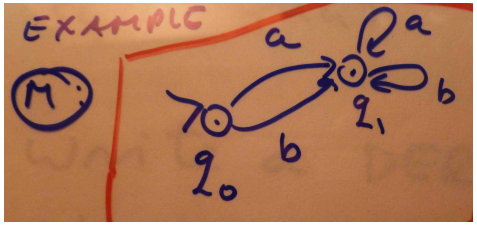
if and only if

$$(A_1, w_1 \ldots w_n) \vdash_M (A_2, w_2 \ldots w_n) \vdash_M \ \ldots \ \vdash_M (A_n, w_n) \vdash_M (f, e)$$

Exercises

Exercise 1

Given M defined by the **diagram** below, **construct** a regular grammar G, such that L(M) = L(G)



We follow the **proof** of L-G Theorem and we "read" the rules of G as follows

$$R = \{q_0 \rightarrow aq_1 \,|bq_1, \quad q_1 \rightarrow aq_1 \,|bq_1, \quad q_0 \rightarrow e, \quad q_1 \rightarrow e\}$$

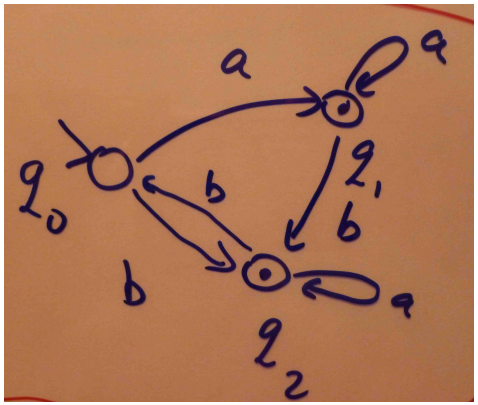
We re-write the rules using a standard notation for nonterminals as

$$R = \{S \rightarrow aA \mid bA, \quad A \rightarrow aA \mid bA, \quad S \rightarrow e, \quad A \rightarrow e\}$$

Exercises

Exercise 2

Given M defined by the **diagram** below, **construct** a regular grammar G, such that L(M) = L(G)



We "read" the rules of G as follows

$$R = \{q_0 \rightarrow aq_1 \,|bq_2, \quad q_1 \rightarrow aq_1 \,|bq_2 \,| \, e, \quad q_2 \rightarrow aq_2 \,|bq_0 \,| \, e\}$$

We re-write the rules using a standard notation for nonterminals as

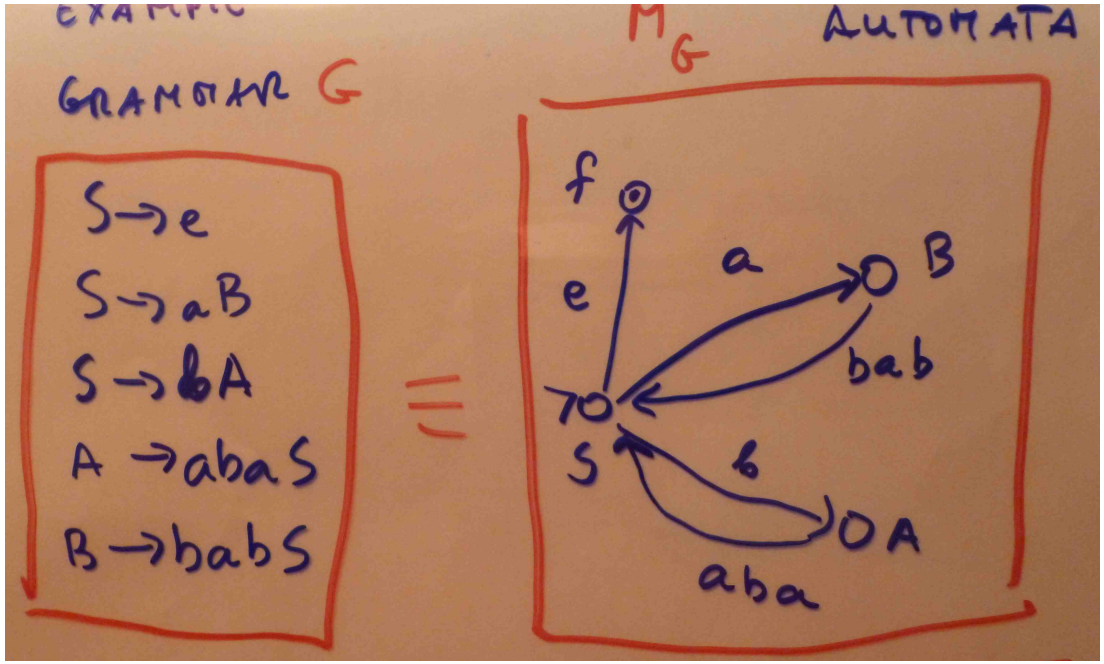
$$R = \{S \rightarrow aA \,|bB, \quad A \rightarrow aa \,|bB \,| \, e, \quad B \rightarrow aB \,|bS \,| \, e\}$$

Exercises

Exercise 3

Given a grammar G defined by the set of rules, **construct** a finite automata G, such that L(M) = L(G)

Here is a **picture** depicting the pattern of such constractions

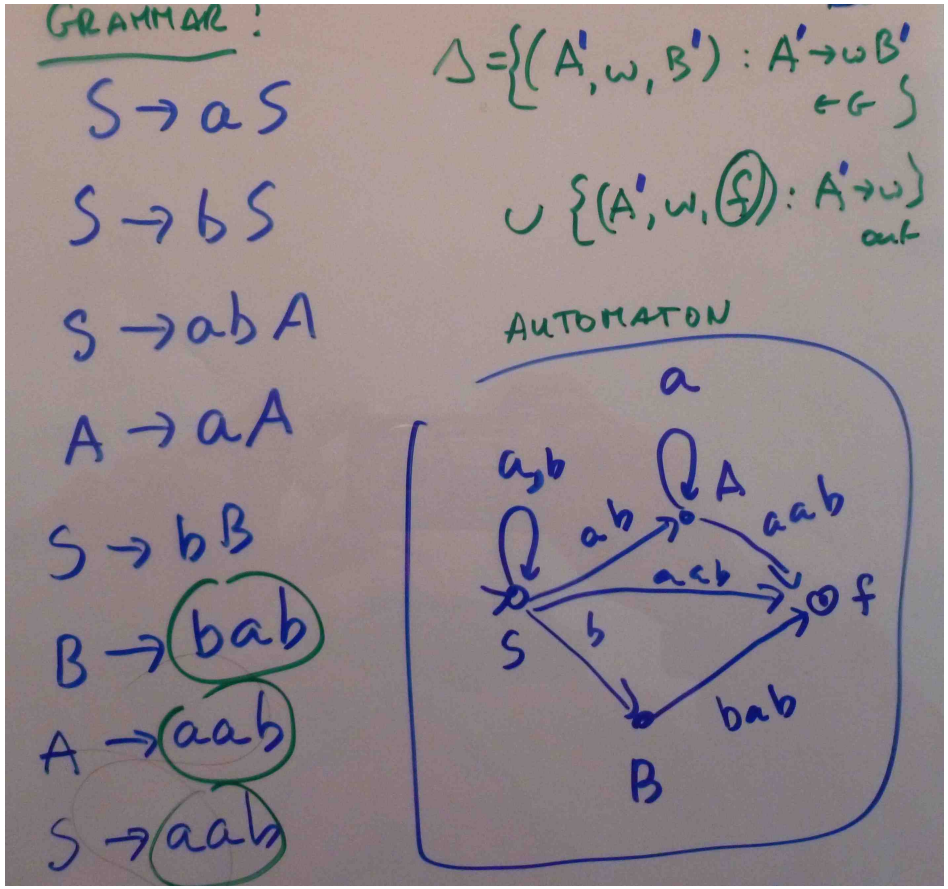


Exercises

Exercise 4

Given a grammar G defined by the set of rules, **construct** a finite automata G, such that L(M) = L(G)

Here is a **picture** depicting the pattern of such contractions



Exercise 5 PROVE the second part of **L-G Theorem**

Simple Questions

Justify if **True** or **False**

Q1 The set of terminals in a context free grammar G is a subset of the alphabet of G

Q2 The set of terminals and non- terminals in a context free grammar G form the alphabet of G

Q3 The set of non-terminals is always non- empty

Q4 The set of terminals is always non- empty

Simple Questions

Justify if **True** or **False**

Q1 The set of terminals in a context free grammar G is a subset of the alphabet of G

True By definition: $\Sigma \subseteq V$

Q2 The set of terminals and non- terminals in a context free grammar G form the alphabet of G

True By definition: $V = \Sigma \cup (V - \Sigma)$

Simple Questions

Justify if **True** or **False**

Q3 The set of non-terminals is always non- empty

True By definition: $S \in V$

Q4 The set of terminals is always non- empty

False $\Sigma = \emptyset$ is a finite set

Simple Questions

Justify if **True** or **False**

Q5 Let G be a context-free grammar

$$L(G) = \{w \in V : \quad S \overset{*}{\underset{G}{\Rightarrow}} w\}$$

Q6 The language $L \subseteq \Sigma^*$ is context-free if and only if

$$L = L(G)$$

Q7 A language is context-free if and only if it is accepted by a context-free grammar

Simple Questions

Justify if **True** or **False**

Q5 Let G be a context-free grammar

$$L(G) = \{w \in V : \quad S \overset{*}{\underset{G}{\Rightarrow}} w\}$$

False Should be $w \in \Sigma^*$

Q6 The language $L \subseteq \Sigma^*$ is context-free if and only if L = L(G)

False Holds only when G is a context -free grammar

Simple Questions

Justify if **True** or **False**

Q7 A language is context-free if and only if it is accepted by a context-free grammar

False Language is generated, not accepted by a grammar

Q8 Any regular language is context-free

True: Regular languages are generated by regular grammars, that are special case of CF grammars

Q9 Language is regular if and only if is generated by a regular grammar (right- linear)

True : Theorem proved in class

Simple Questions

Justify if **True** or **False**

Q10 $L = \{w \in \{a, b\}^* : \quad w = w^R\}$ is context-free

True: G with the rules:

$S \to aSa|bSb|a|b||e$ is the required grammar

Q11 A regular language is a CF language

True: Regular grammar is a special case of a context-free grammar

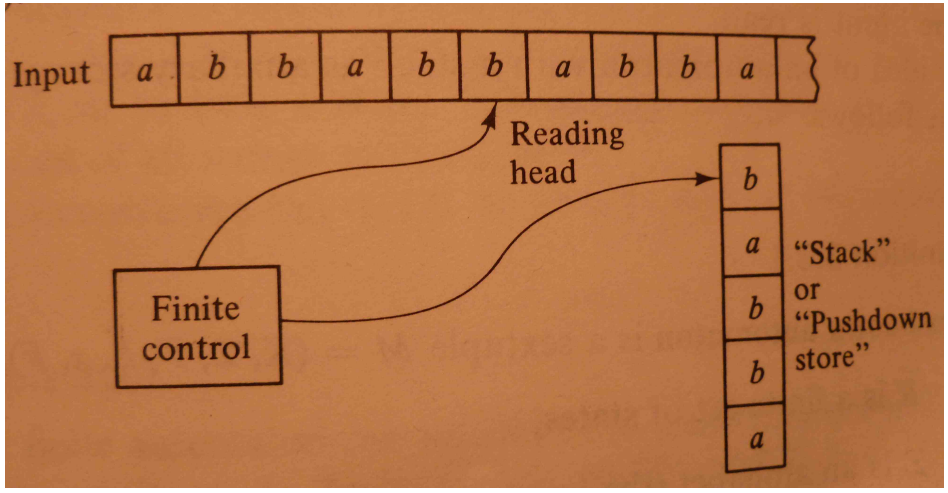
Chapter 3
Context-free Languages

Slides Set 2

PART 2: Pushdown Automata

Pushdown Automata PDA

Computational Model of Pushdown Automata PDA

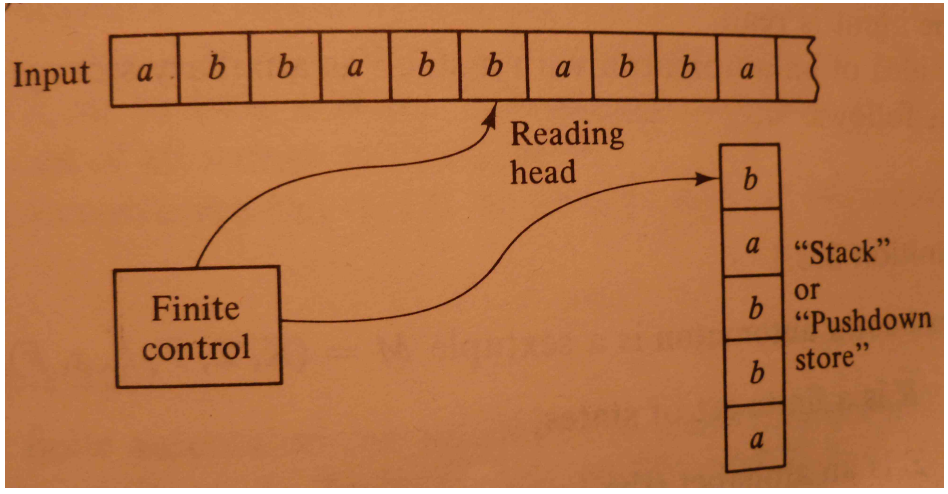


C1: Automata **"remembers"** what it has already read by putting it, one symbol at the time on **stack**, or on **pushdown store**

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

Pushdown Automata PDA

Computational Model of Pushdown Automata PDA



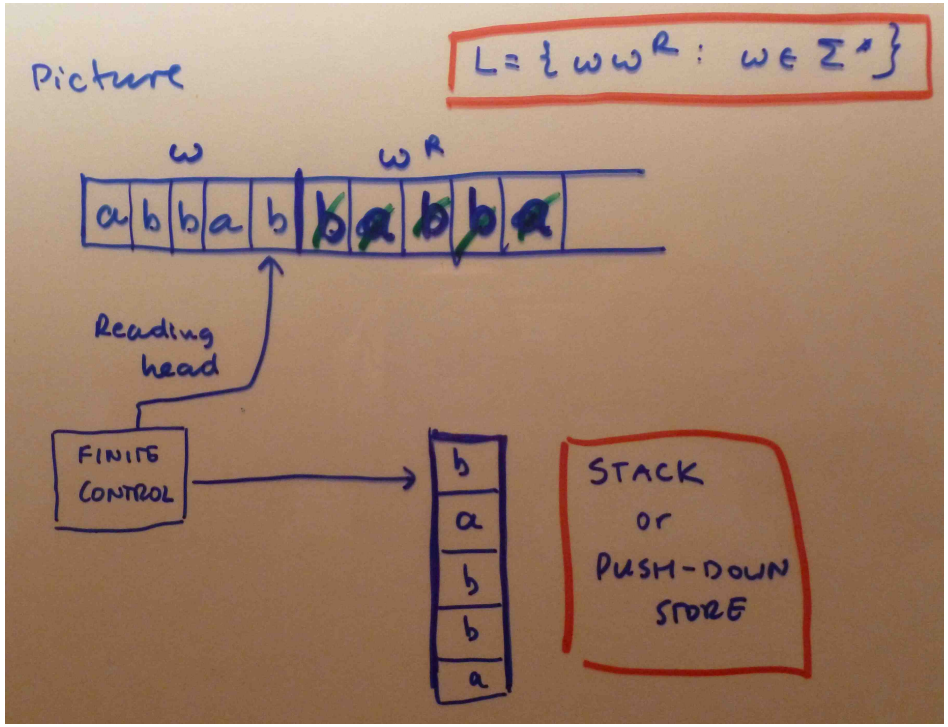
C3: symbols could be **removed** from the **top** of the stack and can be checked against the input

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

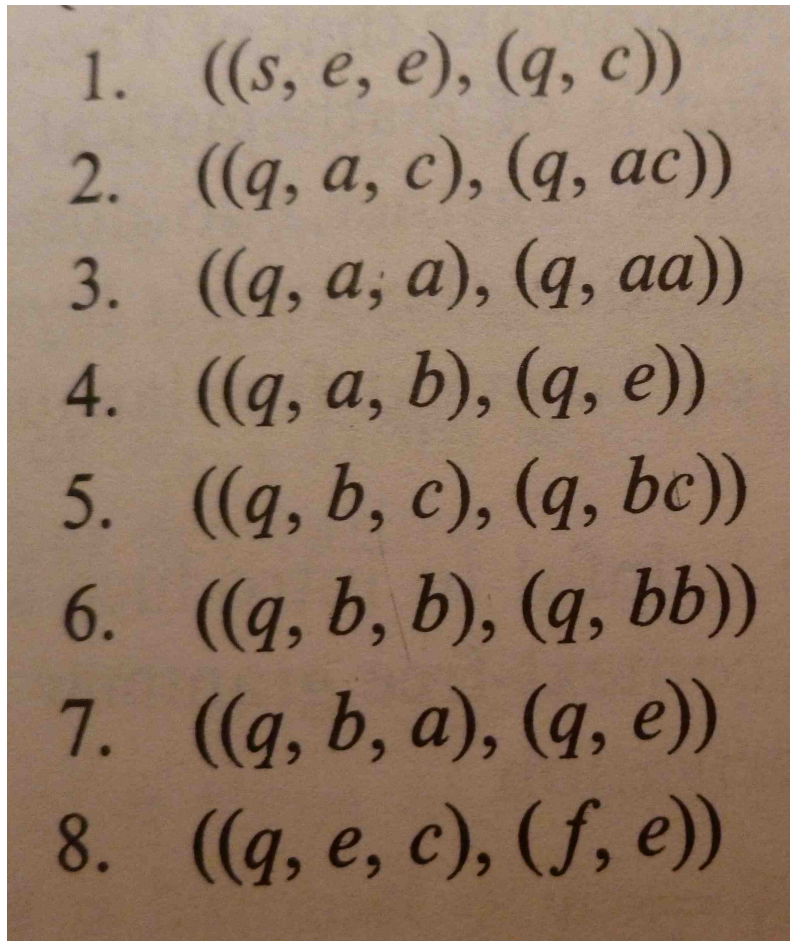
$$L(M) = \{ww^R : \quad w \in \{a, b\}^*\}$$

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

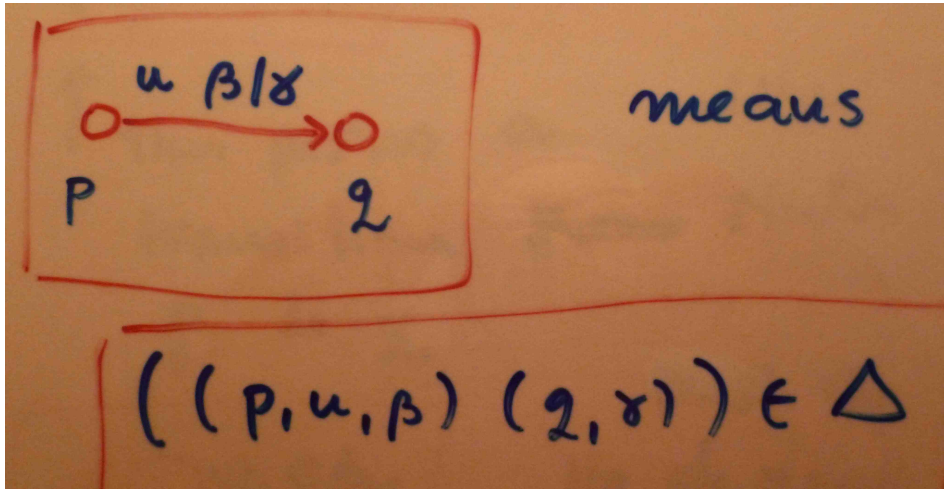
Example 4 Trace a **computation** of M accepting the word abbbabaaa

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

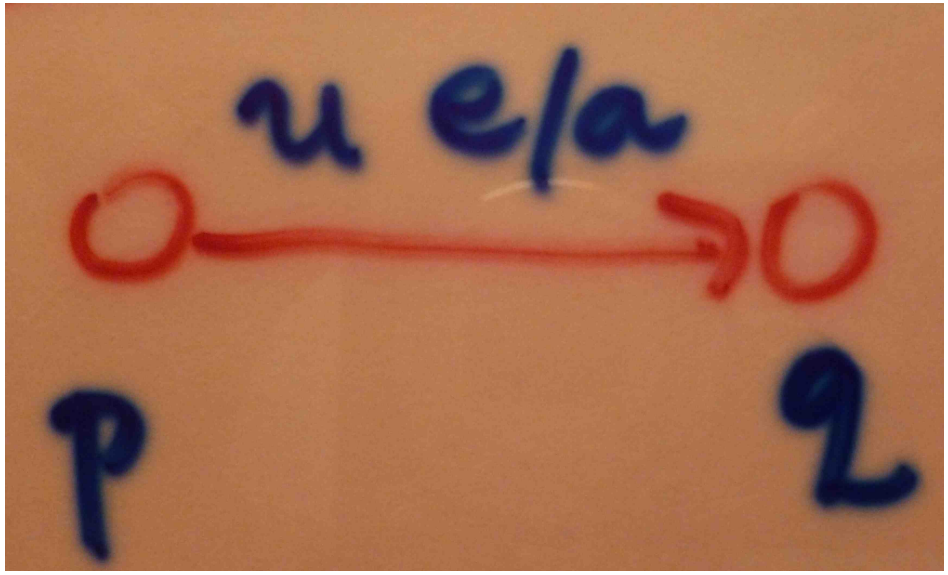


M in state p

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



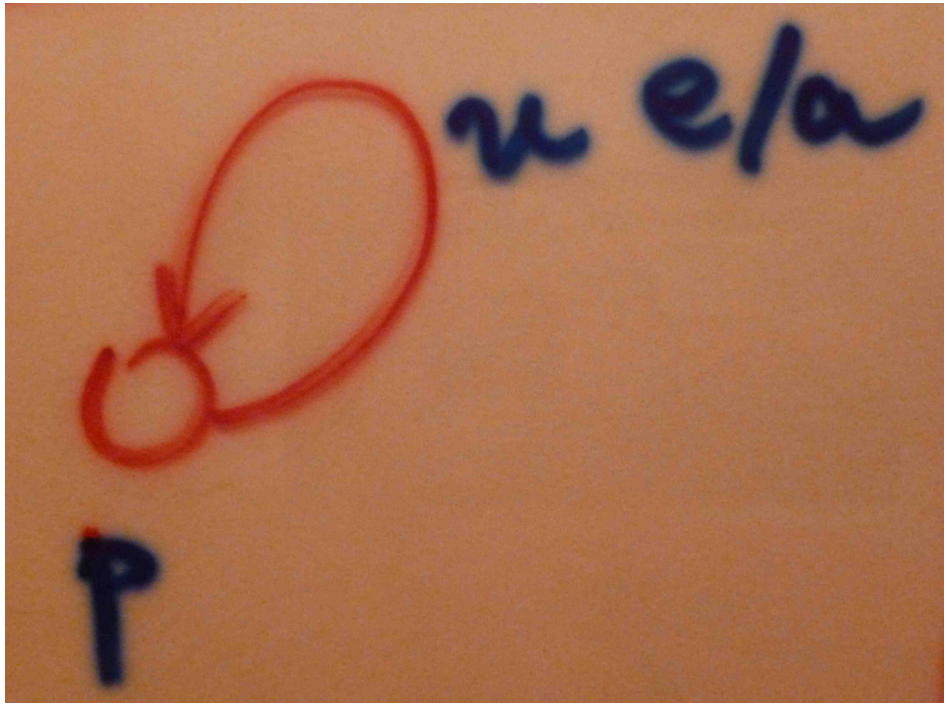
$((p,\ u,\ e),\ (q,\ a)) \in \Delta$ **push** a

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



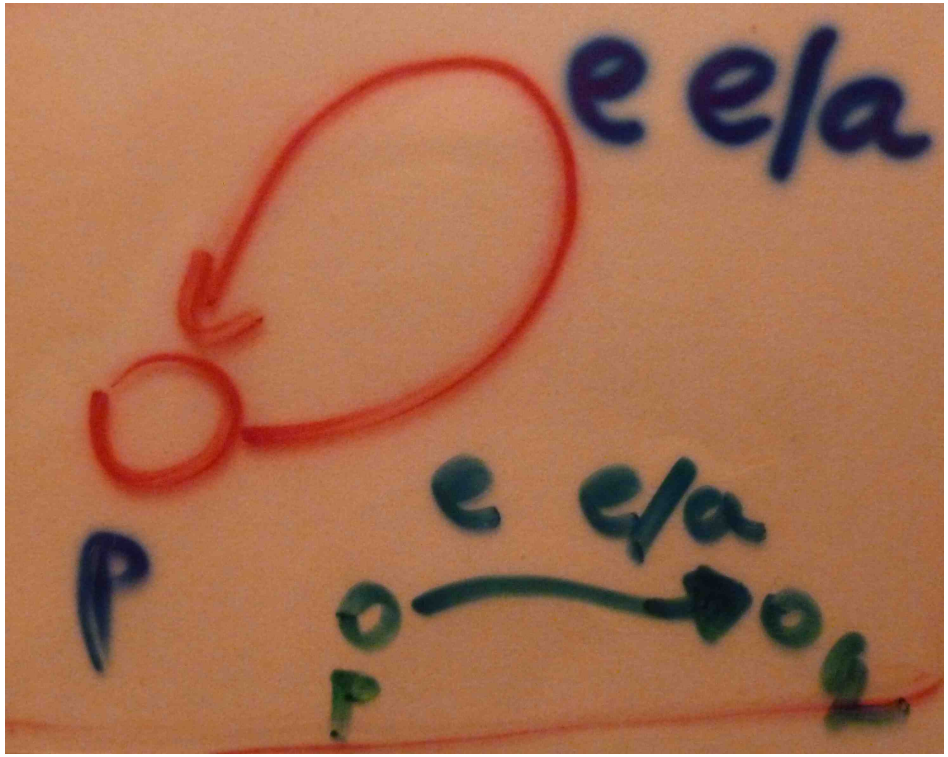
M pushes a with no change of state

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



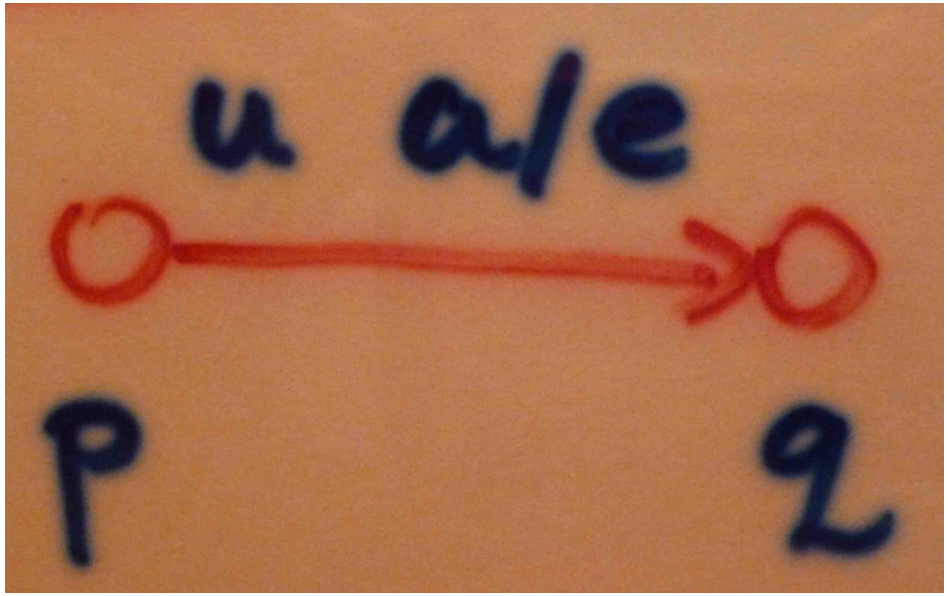
M pushes a with no change of state, reading nothing

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



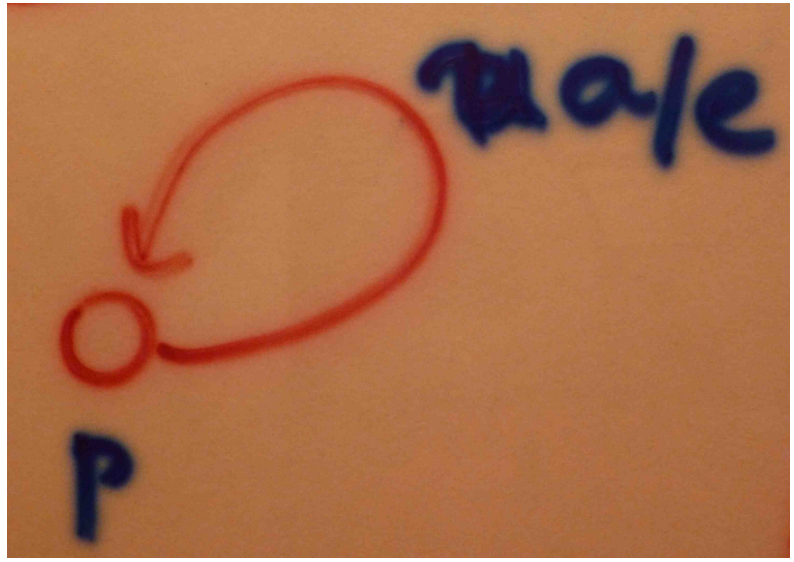
$((p,\ u,\ a),\ (q,\ e)) \in \Delta$ **pop** a

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



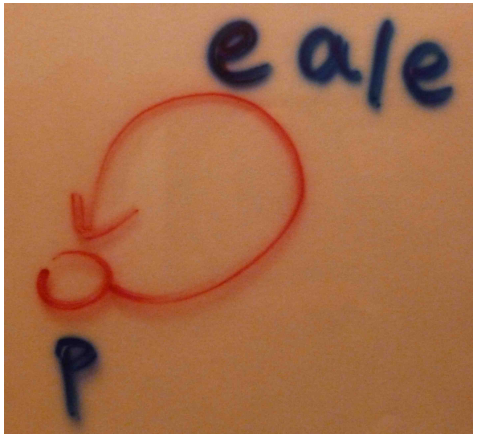
M pushes a with no change of state

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



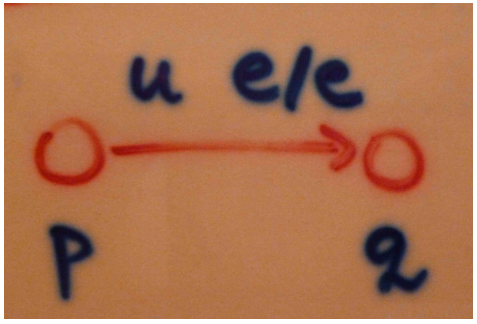
M pushes a with no change of state, reading nothing

In state p

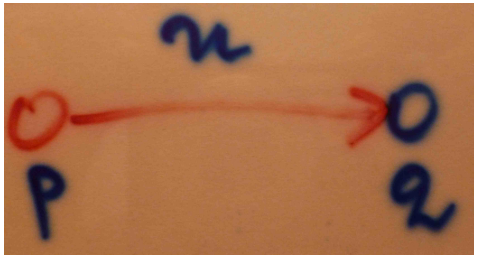
1. reads e
2. pushes a on the top of the stack
3. goes to the state p

State Diagrams for Pushdown Automata

Diagram of PD M'



that imitates the FA colorred M

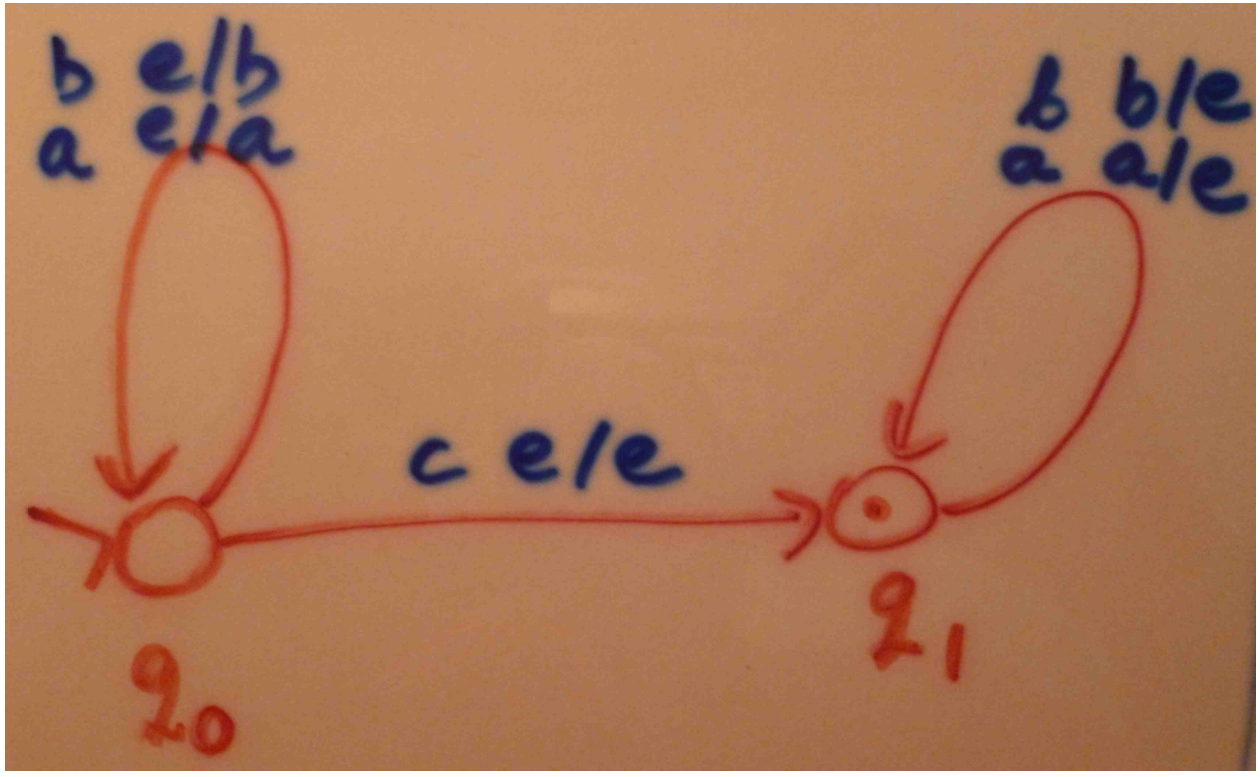


Theorem: Any FA automaton is a PD automaton

Exercises

Exercise 4

Diagram of M

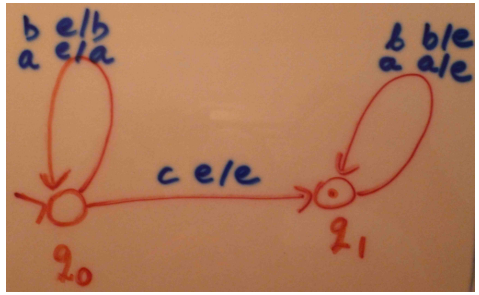


Write components of M and find its language L(M)

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

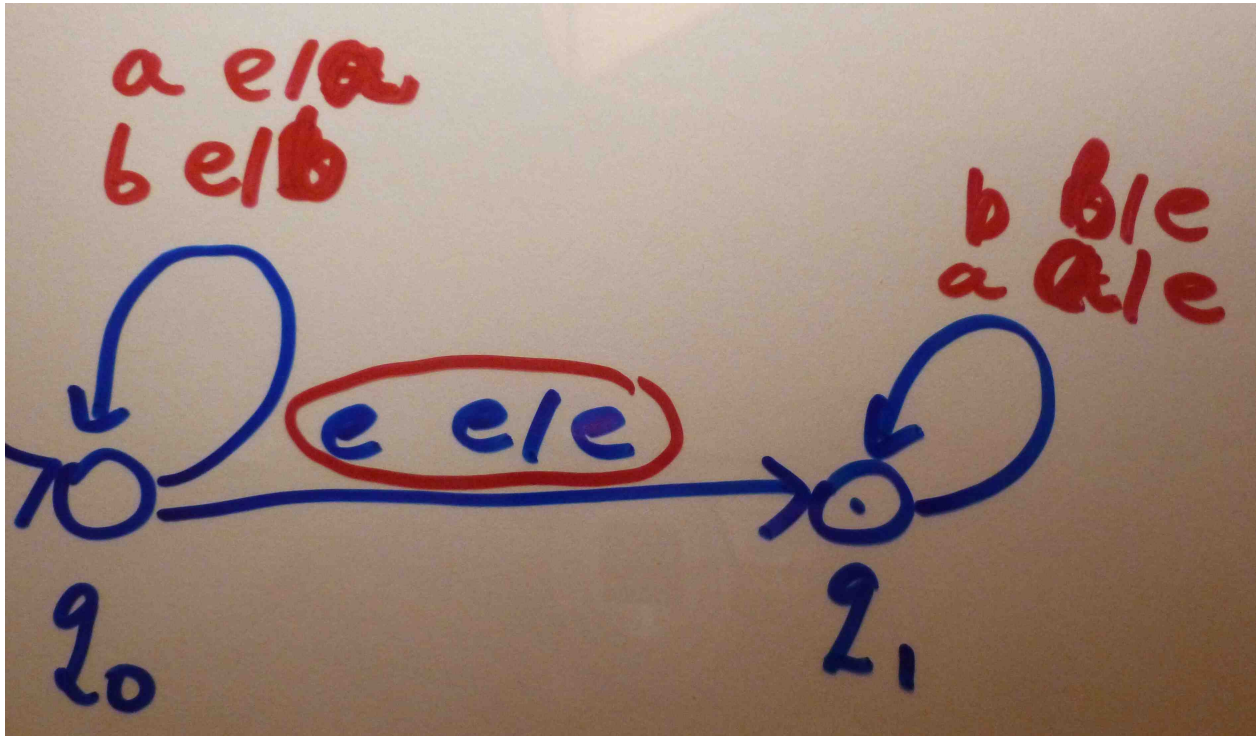
$((q_1, b, b), (q_1, e))$ - **compares** and **pops** b

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

Exercises

Exercise 5

Diagram of M

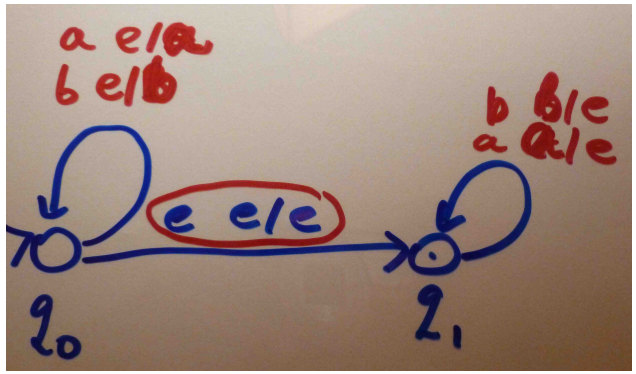


Write components of M and find its language L(M)

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

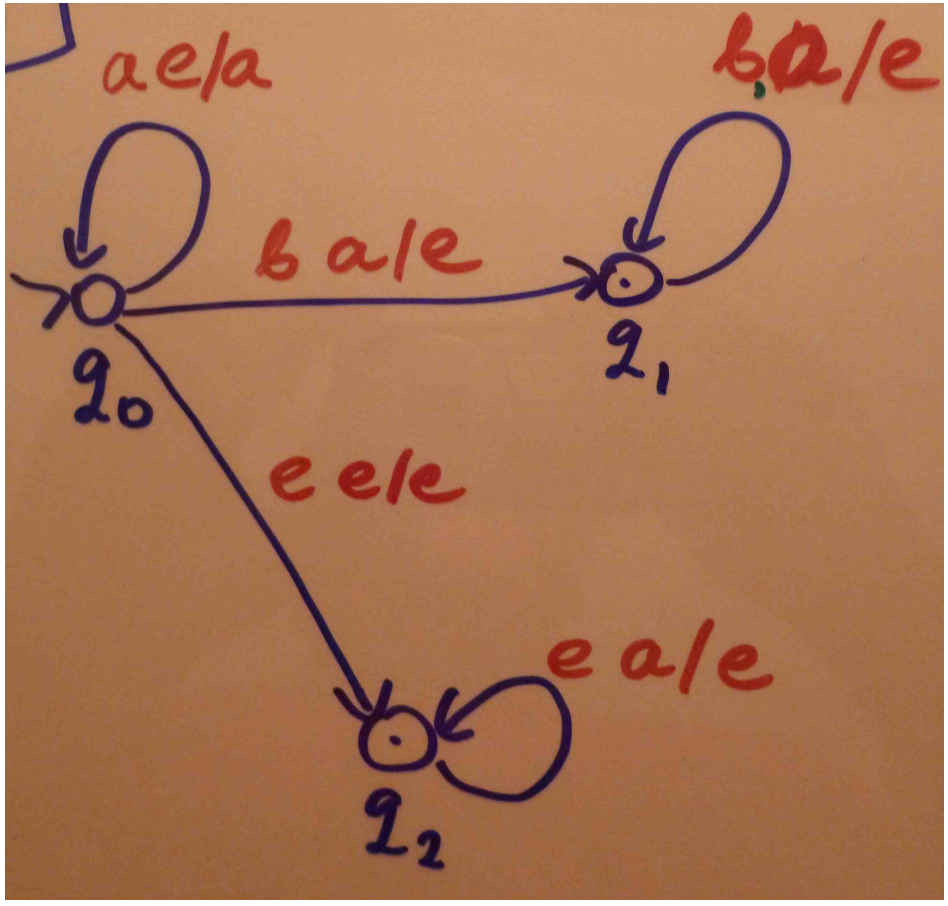
M is **nondeterministic**

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

Exercises

Exercise 6

Diagram of M is

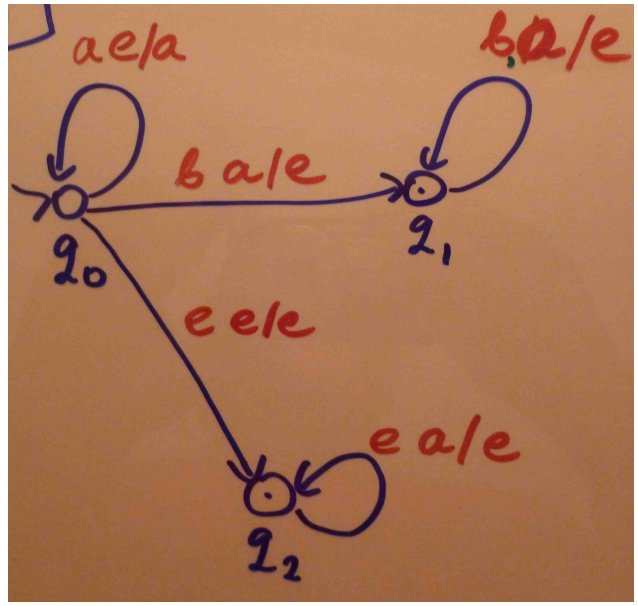


Write components of M and find its language L(M)

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

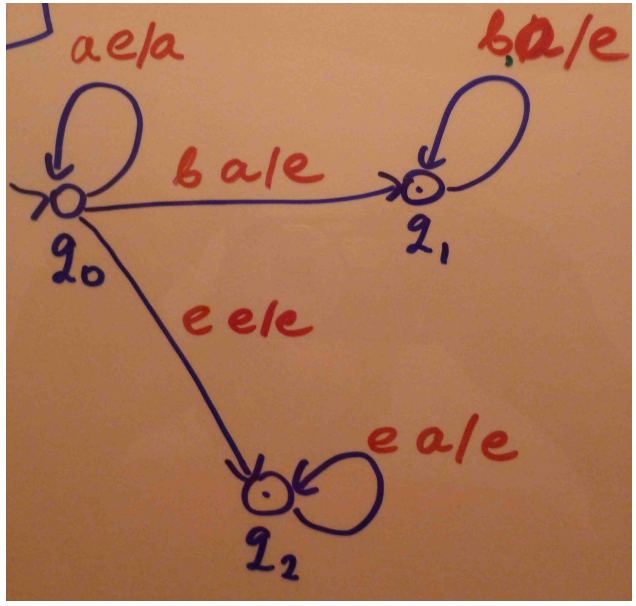
$((q_2, e, a), (q_2, e))$ - while in q_2 **pop** a

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

Slides Set 2

PART 3: Pushdown automata and context -free grammars

Main Theorems

We established in Chapter 2 a **relationship** between finite automata and regular languages by proving the following

FA Main Theorem

A language L is regular if and only if it is accepted by a finite automaton

Our **goal** now is to **prove** a similar **relationship** between push -down automata and context -free languages, i.e. to prove the following

PDA Main Theorem

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

Main Theorem Proof

We break the proof of the PDA **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

Slides Set 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

EXERCISES and QUESTIONS

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**

Some YES/NO questions

Q1

The set of **terminals** in a context free grammar G is a subset of the **alphabet** of G

Q2

$L(G) = \{w \in V : \ S \Rightarrow^*_G w\}$

Q3

A language $L \subseteq \Sigma^*$ is **context-free** if and only if there is a grammar G, such that $L = L(G)$

Q4

Any regular language is context-free

ANSWERS to YES/NO questions

Q1

The set of **terminals** in a context free grammar G is a subset of the **alphabet** of G

YES

By definition : $\Sigma \subseteq V$ for Σ terminals and V alphabet of any context free grammar G

Q2

$L(G) = \{w \in V : \ S \Rightarrow^*_G w\}$

NO Must be that $w \in \Sigma^*$

Q3

A language $L \subseteq \Sigma^*$ is **context-free** if and only if there is a grammar G, such that $L = L(G)$

NO It os true only when G is a context -free grammar

ANSWERS to YES/NO questions

Q3

A language $L \subseteq \Sigma^*$ is **context-free** if and only if there is a grammar G, such that $L = L(G)$

NO

It is true only when G is a context -free grammar

Q4

Any regular language is context-free

YES

Regular languages are generated by regular grammars, that are also context-free

Some YES/NO questions

Q5

The language $L = \{w \in \{a, b\}^* : w = w^R\}$ is **context-free**

Q6

Any regular language is accepted by a pushdown automaton

Q7

Context-free languages are **closed** under intersection

Q8

The **union** of a context-free language and regular language is a context-free language

Q9

Every **subset** of a regular language is a language

Q10

Any regular language is accepted by some PD automata

ANSWERS to YES/NO questions

Q5

The language $L = \{w \in \{a, b\}^* : w = w^R\}$ is **context-free**

YES

G with the rules:

$$S \to aSa|bSb|a|b|e$$

is such that $L = L(G)$

Q6

Any regular language is accepted by a pushdown automaton

YES

By FA Main Theorem, any regular language is accepted by a certain FA, and any finite automata is a pushdown automata operating on an empty stock

ANSWERS to YES/NO questions

Q7

Context-free languages are **closed** under intersection

NO

Take $L_1,\ L_2$ such that

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

Both $L_1,\ L_2$ are CF languages and we get

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

that is **not** CF

ANSWERS to YES/NO questions

Q8

The **union** of a context-free language and regular language is a context-free language

YES Regular language is also a context free language and context free languages are **closed** under union

Q9

Every subset of a regular language is a language

YES A subset of a set is a set

Q10

Any regular language is accepted by some PD automata

YES

Any regular language is accepted by a FA and a finite automaton is a PD automaton (that never operates on the stock)

Exercises

Exercise 1

Prove that the language

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

is context- free

We construct a grammar G such that

$$L(G) = \{a^m b^n : \quad m \geq n\}$$

as follows

$$G = (V, \Sigma, R, S)$$

where $V = \{a, b, S\}, \quad \Sigma = \{a, b\}$

$$R = \{S \rightarrow aS \,|aSb \mid e\}$$

Derivation examples

$$S \Rightarrow aS \Rightarrow aaS \Rightarrow aaaSb \Rightarrow aaab$$

$$S \Rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaabbb$$

Exercises

Exercise 2

Construct a grammar G such that

$$L(G) = \{a^m b^n : \quad m > n\}$$

We adopt rules

$$R = \{S \rightarrow aS \,|aSb \mid a\}$$

Exercise 3

Prove that

$$L = \{a^m b^n : \quad m < n\}$$

is context- free

Observe that

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

and hence L is context- free as regular language is context-free and context-free languages are **closed** under concatenation

Exercises

Exercise 4

Prove that

$$L = \{a^m b^n : \quad m \neq n\}$$

is context- free

Observe that

$$\{a^m b^n : \quad m \neq n\} = \{a^m b^n : \quad m > n\} \cup \{a^m b^n : \quad m < n\}$$

and hence L is context- free as context-free languages are **closed** under union and we have just proved that both union components are context-free

Exercises

Exercise 5

We proved that

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

We know that

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

is context- free

Prove that the complement of L is also **context- free**

Observe that

$$\Sigma^* - L = \{a^m b^n : \ m \neq n\} \cup \Sigma^* a \Sigma^* b \Sigma^* a \Sigma^* \cup \Sigma^* b \Sigma^* b \Sigma^* a \Sigma^*$$

and hence $\Sigma^* - L$ is context- free as **union** of context-free languages

Exercises

Exercise 6

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

Prove that $L - R$ is context-free

What about $R - L$?

Observe that

$$L - R = L \cap (\Sigma^* - R)$$

and regular languages are **closed** under intersection and

Theorem 3

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

$R - L$ does not need to be CF

Take $R = \Sigma^*$

$$R - L = \Sigma^* - L$$

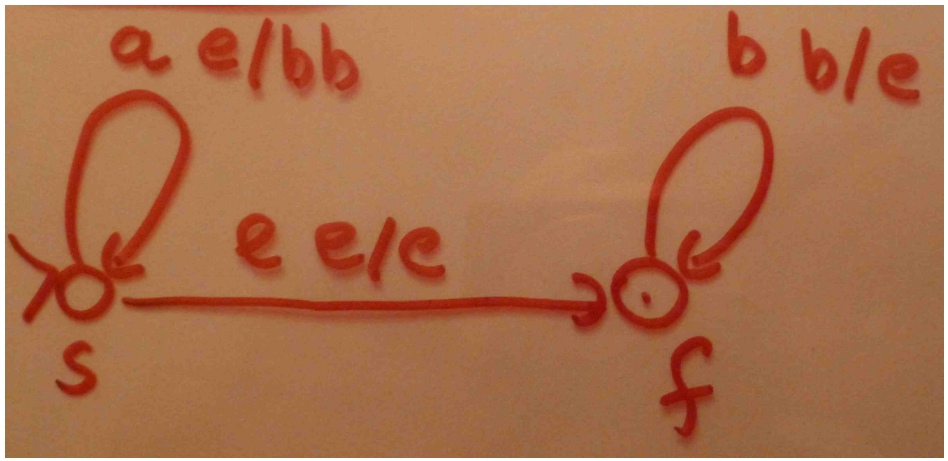
and CF languages are **not closed** under complementation

Exercise 7

Construct a PD automaton M such that

$$L(M) = \{a^n b^{2n} : \quad n \geq 0\}$$

The **diagram** of M is



The components of M are

$$K = \{s, f\}, \quad \Sigma = \{a, b\}, \quad \Gamma = \{a\}, \; s, \; F = \{f\}$$

Exercises

The set Δ of transitions is

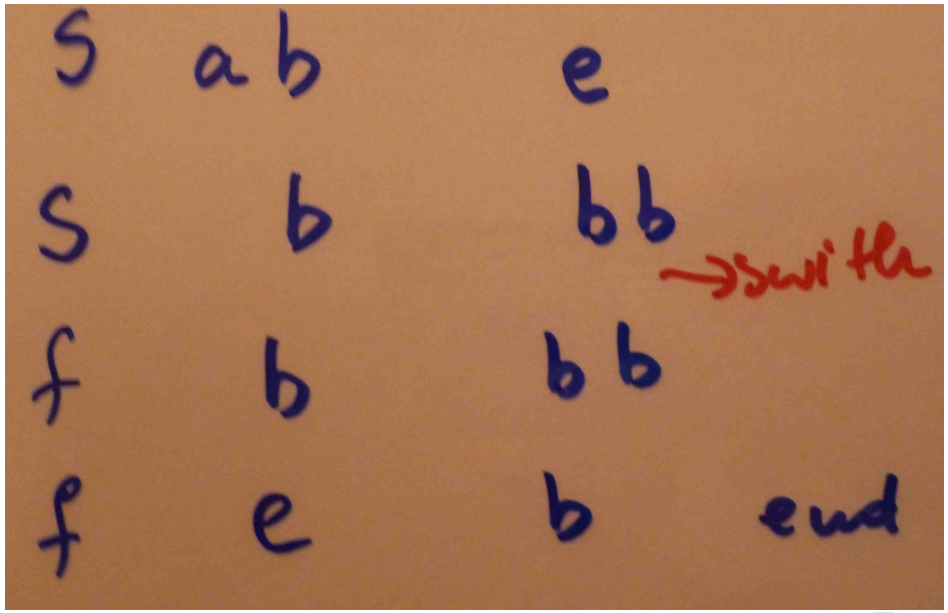
$$\Delta = \{((s,a,e),(s,bb)),\ ((s,e,e),(f,e)),\ ((f,b,b),(f,e))\}$$

Show that $ab \notin L(M)$

M is **non-deterministic**, so we have to consider **all possible** computations M starting at the word ab

Here they are:

1.



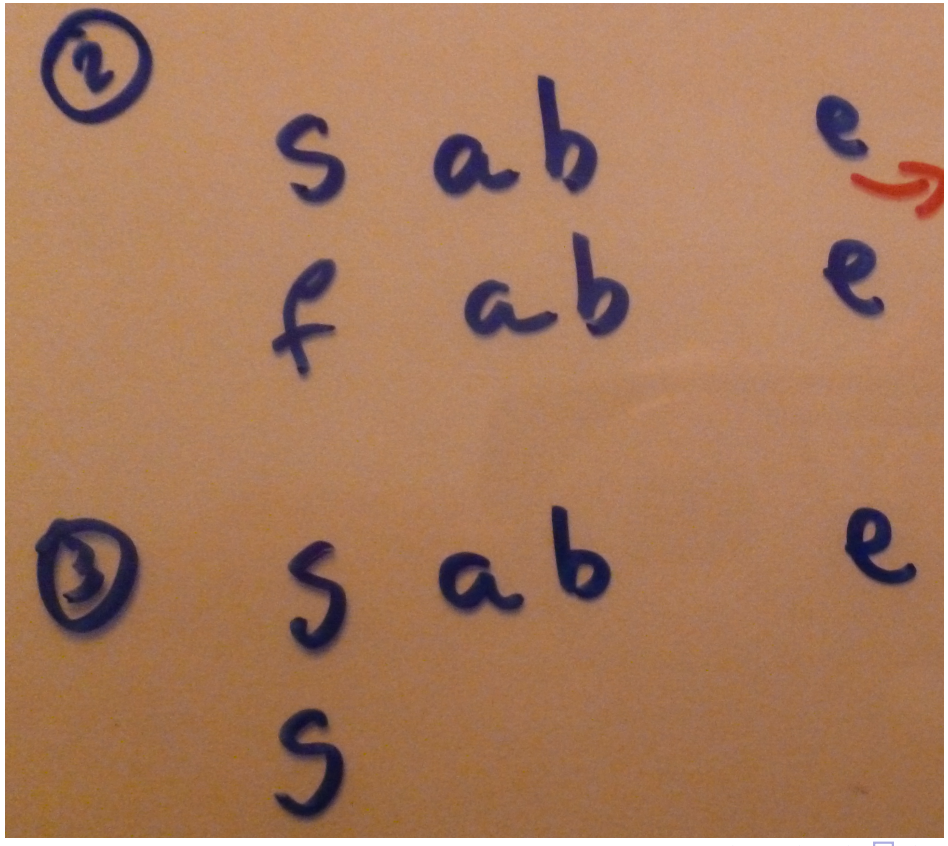
Exercises

The set Δ of transitions is

$$\Delta = \{((s, a, e), (s, bb)), \quad ((s, e, e), (f, e)), \quad ((f, b, b), (f, e))\}$$

There are only two more "switches"

2. and **3.**



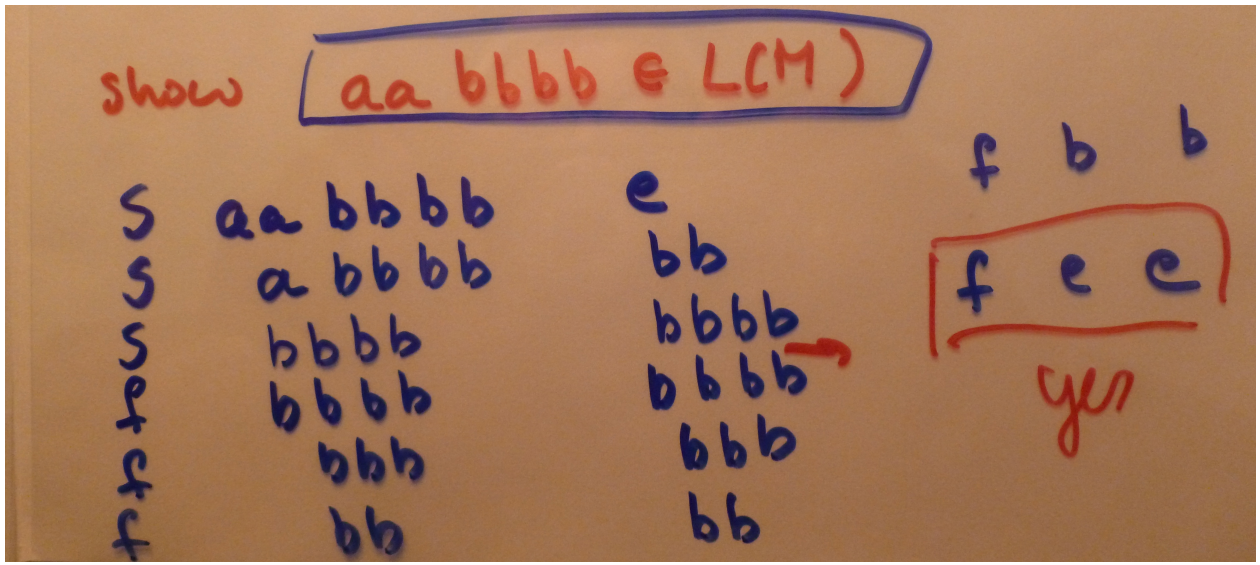
Exercises

The set Δ of transitions is

$$\Delta = \{((s, a, e), (s, bb)), \quad ((s, e, e), (f, e)), \quad ((f, b, b), (f, e))\}$$

Show that $aabbbb \in L(M)$

Here is a computation of M accepting *aabbbb*



Configuration and Transition

In order to define a notion of 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

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

Given

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

Transition relation

Definition

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 $\vdash_M$ denoted by $\vdash_M{}^*$ and define

Definition

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

M **accepts** $w \in \Sigma^*$ if and only if $w \in L(M)$

In plain English:

(s, w, e) means:

start with w and empty **stack**

(p, e, e) for certain $p \in F$ means:

finish in a final state after reading w and emptying all of the **stack**

Exercises

Exercise

Here is M

M operates as follows

Δ **pushes** aa on the top of the stock while M is reading b,

switches to f (final state) **non-deterministically**;

and **pops** a while **reading** a - all in final state

M **puts** on the stock **two** a's for each b,

and then **removes** all a's from the stock **comparing** them with a's included in the word

all theses actions while M is in the final state

Write L(M)

$$L(M) = \{b^n a^{2n} : \quad n \geq 0\}$$

Exercises

Trace formally a computation of M that leads to the **acceptance** of the string bbaaaa

The **accepting** computation is:

$$(s, bbaaaa, e) \vdash_M (s, baaaa, aa) \vdash_M (s, aaaa, aaaa)$$

$$\vdash_M (f, aaaa, aaaa) \vdash_M (f, aaa, aaa)$$

$$\vdash_M (f, aa, aa) \vdash_M (f, a, a) \vdash_M (f, e, e)$$

Alternative definition of $M = (K, \Sigma, \Gamma, \Delta, s, F)$ is

$$\Delta = \{((s, b, e), (s, b)), ((s, e, e), (f, e)), ((f, aa, b), (f, e))\}$$

Exercises

Exercise 8

Prove that L(G) is NOT regular, for

$$G = (V,\ \Sigma,\ S,\ R)$$

where $V = \{S,\ (,\ )\},\ \ \Sigma = \{(,\ )\}$

$$R = \{S \rightarrow\ SS \mid (S) \mid e\}$$

Proof by contradiction

Assume that L(G) is **regular**. The language $L_1 = (^*)^*$ is regular and regular languages are closed under $\cap$, so

$$L(G) \cap L_1 = (^n)^n$$

is regular. **Contradiction**

Exercises

Exercise 9

Given a context-free grammar

$$G = (V,\ \Sigma,\ S,\ R)$$

where $V = \{S,\ (,)\},\ \ \Sigma = \{(,)\}$

$$R = \{S \to\ SS \mid (S) \mid e\}$$

1. **Construct** a PD automaton M, such that

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

2. **Show** that $()() \in L(M)$

Exercises

We use construction described in the proof of our **Main Theorem**, in particular, in the proof of the its first part, i.e.

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 construct a PD automaton M, such that L(G) = L(M) as follows

M has only two states: initial s and a final f

M remains in state f after its first move

Δ contains the following transitions

1. ((s, e, e), (f, S))
2. ((f, e, A), (f, x)) - for each rule $A \to x$
3. ((f, c, c), (f, e)) - for each $c \in \Sigma$

Exercises

The rules of G are:

$$R = \{S \rightarrow \ SS \mid (S) \mid e\}$$

General case: Δ contains the following transitions

1. ((s, e, e), (f, S))

2. ((f, e, A), (f, x)) - for each rule $A \rightarrow x$

3. ((f, c, c), (f, e)) - for each $c \in \Sigma$

The transitions of M are

$$\Delta = \{ ((s, e, e), (f, S)), \quad ((f, e, S), (f, SS)),$$

$$((f, e, S), (f, (S))), \quad ((f, e, S), (f, e)),$$

$$((f, (, ( ), (f, e)), \quad ((f,),)), (f, e)) \}$$

Exercises

We **trace formally** a computation of M that leads to the **acceptance** of the string $()()$

The **accepting** computation is:

$$(s, ()(), e) \vdash_M (f, ()(), S)) \vdash_M (f, ()(), SS))$$

$$\vdash_M (f, ()(), (S)S) \vdash_M (f,)(), S)S) \vdash_M (f,)(),)S))$$

$$\vdash_M (f, (), S) \vdash_M (f, (), (S)) \vdash_M (f,), S)) \vdash_M (f,),)) \vdash_M (f, e, e)$$

We proved that

$$()() \in L(M)$$

Exercises

Exercise 9

Construct a regular grammar G such that

$$L(G) = b^* \cup a$$

We use the **closure** of CF languages over over UNION construction

Let G_1, G_2 be two regular grammars with has sets of rules

$$R_1 : \quad S_1 \to bS_1 \mid e$$

$$R_2 : \quad S_2 \to a$$

Obviously

$$L(G_1) = b^* \quad \text{and} \quad L(G_2) = a$$

Exercises

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

We **add** new initial state S such that $S \neq S_1, S_2$, and make S_1, S_2 the internal states

The rules for $G = G_1 \cup G_2$ are

$$S \to S_1 \,|S_2, \quad S_1 \to bS_1 \mid e, \quad S_2 \to a$$

We re-write them as

$$S \to A \,|B, \quad B \to bB \mid e, \quad A \to a$$

Here is another, direct grammar G with rules

$$S \to B \,|a, \quad B \to bB \mid e$$

Exercises

Exercise 10

Let G be a CF grammar with $\Sigma = \{a, b\}$ the following rules

$$S \to aSb \mid A \mid B$$

$$A \to abS \mid aSBb$$

$$B \to AB \mid Ba \mid bSaB$$

Describe L(G)

By definition

$$L(G) = \{w \in \Sigma^* : S \overset{*}{\underset{G}{\Rightarrow}} w\}$$

and hence we have that

$$L(G) = \emptyset$$

Exercises

Exercise 11

Let G be a CF grammar with $\Sigma = \{a, b\}$ the following rules

$$S \to aaA \mid B \mid abB \mid e$$

$$A \to bS \mid a$$

$$B \to bS$$

Construct a **nondeterministic** finite automaton M such that

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

We follow the poof of the **L-G Theorem**

L-G Theorem

Language L is **regular** if and only if there exists a **regular** grammar G such that

$$L = L(G)$$

Proof part 1

Suppose that L is **regular**; then L is accepted by a **deterministic** finite automaton

$$M = (K,\ \Sigma,\ \delta,\ s,\ F)$$

We **construct** a regular grammar G as follows

$$G = (V,\ \Sigma,\ R,\ S)$$

for $V = \Sigma \cup K, \quad S = s$

$$R = \{q \rightarrow ap : \quad \delta(q, a) = p\} \cup \{q \rightarrow e : \quad q \in F\}$$

L-G Theorem Part 2

Proof part 2

Let now G be any **regular** grammar

$$G = (V,\ \Sigma,\ R,\ S)$$

We define a **nondeterministic** automaton M such that

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

as follows

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

$K = (V - \Sigma) \cup \{f\}$ where f is a new element

$$s = S, \quad F = \{f\}$$

Proof of L-G Theorem Part 2

The set Δ of transitions is

$$\Delta = \{(A, w, B) : \quad A \to wB \in R;\ A, B \in V - \Sigma, \quad w \in \Sigma^*\}$$

$$\cup\{(A, w, f) : \quad A \to w \in R;\ A, B \in V - \Sigma, \quad w \in \Sigma^*\}$$

Once again, derivations are mimicked by the moves, i.e, for any

$A_1, \ldots, A_n \in V - \Sigma, \quad w_1, \ldots w_n \in \Sigma^*$

$$A_1 \Rightarrow_G \ w_1 A_2 \ \Rightarrow_G \cdots \Rightarrow_G \ w_1 \ldots w_{n-1} A_n \Rightarrow_G \ w_1 \ldots w_n$$

if and only if

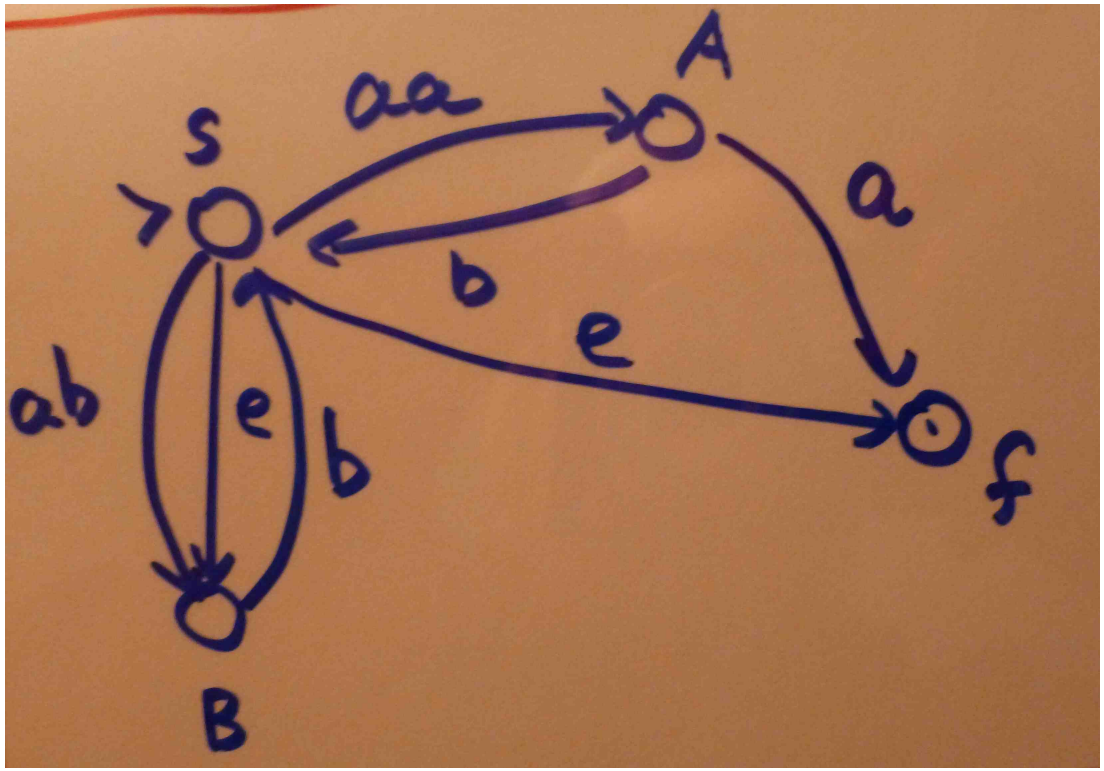
$$(A_1, w_1 \ldots w_n) \vdash_M (A_2, w_2 \ldots w_n) \vdash_M \ \ldots \ \vdash_M (A_n, w_n) \vdash_M (f, e)$$

Exercise 11 Solution

Rules of G

$$S \to aaA \mid B \mid abB \mid e,\ A \to bS \mid a,\ B \to bS$$

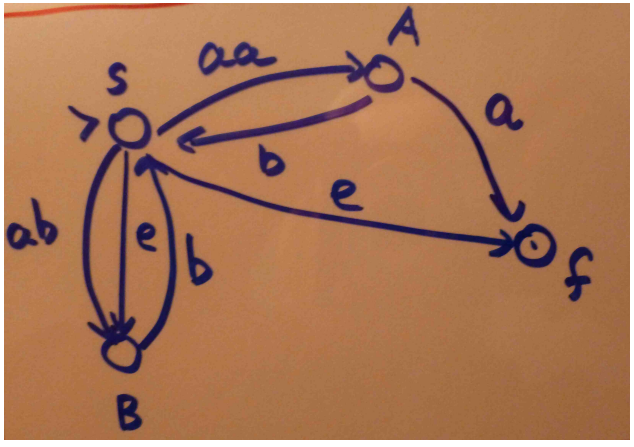
The **diagram** of M is



Trace a computation of M that leads to the acceptance of abbaaa and **compare** it with derivation of abbaaa in G

Exercises

The **diagram** of M is



Here is the **computation** of M and **derivation** of in G

