cse581
Computer Science Fundamentals: Theory

Professor Anita Wasilewska

TCB - LECTURE 6

CONTEXT FREE, NOT CONTEXT FREE LANGUAGES
and
PUSHDOWN AUTOMATA

PDA Main Theorem

We are show that the Pushdown Automaton (PDA) is exactly what is needed to **accept** arbitrary context-free language, i.e. we prove the following

PDA Main Theorem

The class of languages **accepted** by PD Automata is exactly the class of Context-free Languages

PDA Main Theorem Proof

The PDA **Main Theorem** consists of two parts

PDA **Theorem 1**

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

PDA **Theorem 2**

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

We prove only the PDA **Theorem 1**. The proof of and the PDA **Theorem 2** is included in the Book B2 on pages 139 - 142

Establishing Context-freeness of Languages

The PDA **Main Theorem** establishes an **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 further **tools** for establishing context-freeness of languages

We prove some important **Closure Properties** of the context free languages **under** certain language **operations**, as we have done in a case of the regular languages.

Establishing Context-Freeness of Languages

We present a version of the **Pumping Lemma** for the Context Free Languages

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

Closure Theorems

We **prove** the following Closure Theorems by a direct construction of proper Context- Free Grammars

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

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

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 Theorem 2

We use FA **Main Theorem** and PDA **Main Theorem** to **prove** the following

Closure Theorem 2

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

Pumping Lemma
for Context Free Languages

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.

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

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

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 5

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 4**

Contradiction

Context-Free / NOT Context - Free

Main Equivalency Theorem

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

We have proved by constructing a PD automaton and applying the **Main Equivalency Theorem** that we get the language

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

is context- free

Context-free/ NOT Context-Free

We prove by **Pumping Lemma** that the languages

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

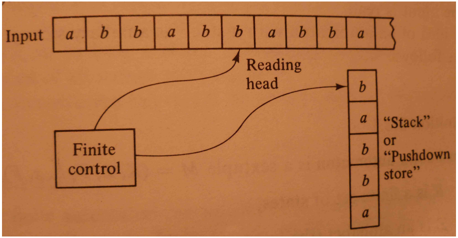
$$L = \{a^p b^n : \ p \in Prime, \ n > p\}$$

are **NOT Context- Free**

PUSH DOWN AUTOMATA
MAIN EQUIVALENCY THEOREMS

Pushdown Automata PDA

Computational Model of Pushdown Automata PDA

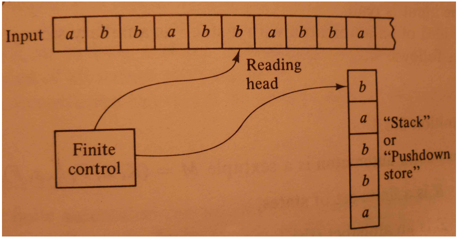


C1: Automata "**remembers**" what it has already read by putting it, one symbol at the time on **stack**, or on **pushdown store**

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

Pushdown Automata PDA

Computational Model of Pushdown Automata PDA

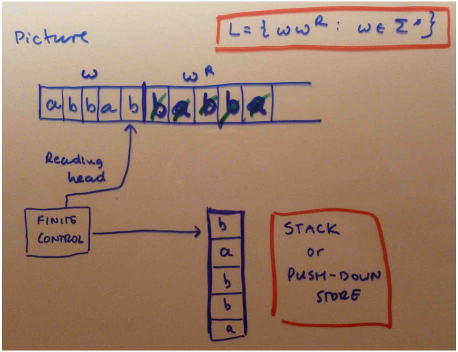


C3: symbols could be **removed** from the **top** of the stack and can be checked against the input

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 $A \rightarrow wB$ 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 is an alphabet of **input symbols**

Γ is 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

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

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

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

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

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 Definition

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

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

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 **accepts** $w \in \Sigma^*$ 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

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

Example of PDA

Example

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. $((f, b, b), (f, e))$

Example

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

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 Trace a computation of of M accepting the word abbcbba

Here it is (Book B2, 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**