Automata Formal Languages & Logic

Preet Kanwal
Department of Computer Science & Engineering

Automata Formal Languages & Logic

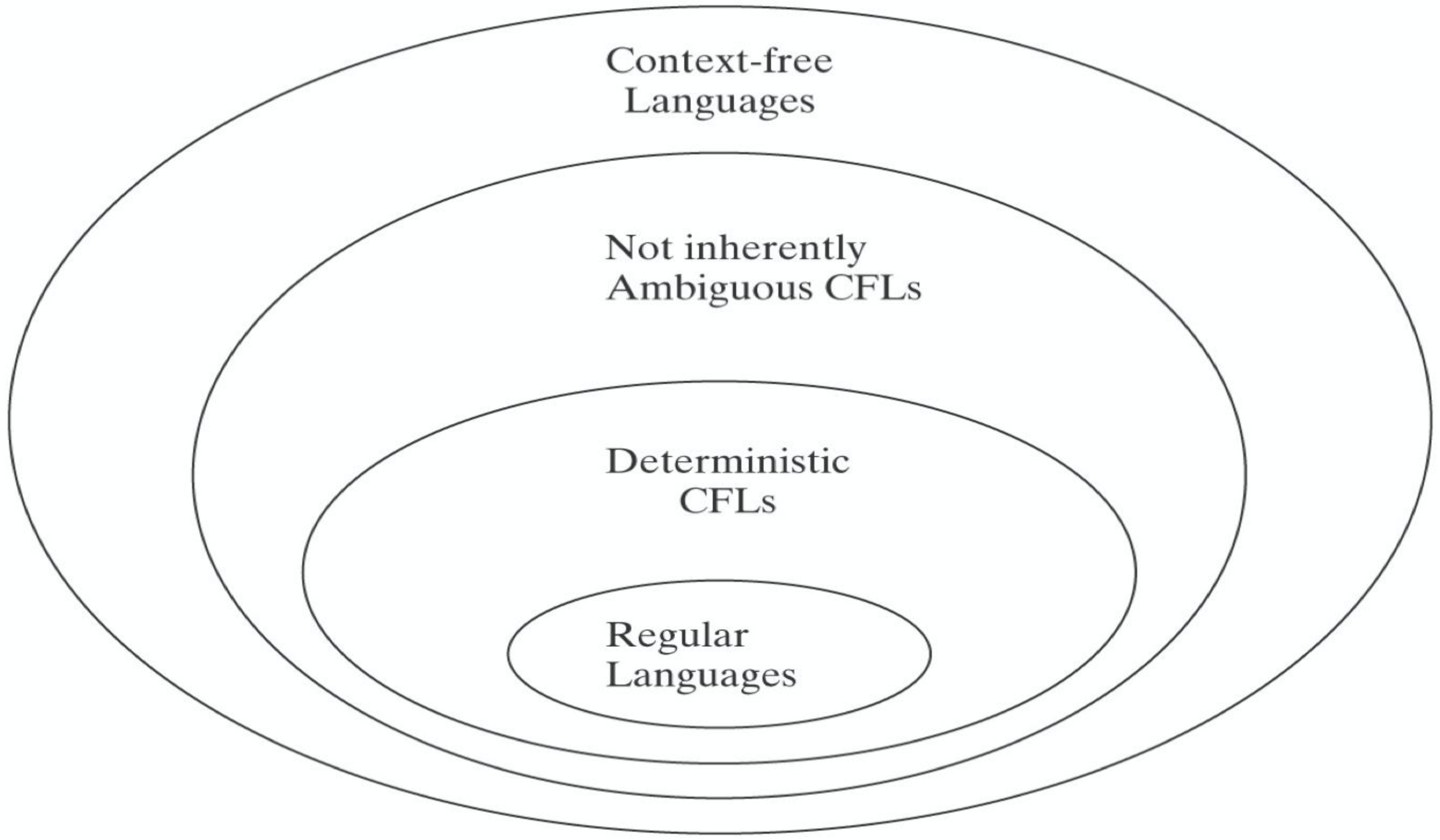
Unit 4 - Properties of Context Free Languages

Preet Kanwal

Department of Computer Science & Engineering

Unit 4 - Context Free Language Hierarchy

- CFLs are of two types:
  1. DCFLs
  2. CFLs*
- CFLs recognize a larger class of grammars.
- DCFLs are proper subsets of CFLs.
- DCFLs are always unambiguous.



*we don't explicitly call these languages as Non-deterministic CFLs(it is understood by default)

Closure properties						
	Union	Concatenation	Star Closure	Reversal	Complement	Intersection
Regular Languages	✓	✓	✓	✓	✓	✓
DCFL	✗	✗	✗	✗	✓	✗
CFL	✓	✓	✓	✓	✗	✗

Unit 4 - Regular Intersection

RL ∩ CFL = CFL

Example:

L_1 = {a*b*} is a Regular Language

L_2 = {a^nb^n, n>=0} is a Context Free Language

$L_1 \cap L_2$ = {a^nb^n, n>=0} which is a Context Free Language

Automata Formal Languages and Logic

Unit 4 - Closure properties of Context Free Languages

Assume	CFG	$G_1 = (V_1, \Sigma_1, P_1, S_1)$	$G_2 = (V_2, \Sigma_2, P_2, S_2)$
	CFL	$L(G_1) = \{a^n b^n : n \geq 0\}$	$L(G_2) = \{c^n d^n : n \geq 0\}$
	Productions	$S_1 \rightarrow aS_1b$ $S_1 \rightarrow \lambda$	$S_2 \rightarrow cS_2d$ $S_2 \rightarrow \lambda$
Closure under	Union $S \rightarrow S_1 \mid S_2$	$G = (V_1 \cup V_2 \cup \{S\}, \Sigma_1 \cup \Sigma_2, P, S)$ where $P = P_1 \cup P_2 \cup \{S \rightarrow S_1 \mid S_2\}$ & S is the new Start symbol. Then, $L(G) = L(G_1) \cup L(G_2) = \{a^n b^n : n \geq 0\} \cup \{c^n d^n : n \geq 0\}$	
	Concatenation $S \rightarrow S_1 S_2$	$G = (V_1 \cup V_2 \cup \{S\}, \Sigma_1 \cup \Sigma_2, P, S)$ where $P = P_1 \cup P_2 \cup \{S \rightarrow S_1 S_2\}$ & S is the new Start symbol. Then, $L(G) = L(G_1)L(G_2) = \{a^m b^m c^n d^n : m,n \geq 0\}$	
	Kleene Star $S \rightarrow S_1 S \mid \lambda$	$G = (V_1 \cup S, \Sigma_1 , P, S)$ where $P = P_1 \cup \{S \rightarrow S_1 S \mid \lambda)$ & S is the new Start symbol. Then, $L(G) = L(G_1)^* = \{a^n b^n : n \geq 0\}^*$	
	Reversal	$G = (V_1 , \Sigma_1 , P_1^R, S_1)$where $P_1^R = S_1 \rightarrow bS_1a \mid \lambda$ (all RHS of the production are reversed) Then, $L(G) = L(G_1)^R = \{b^n a^n : n \geq 0\}^*$	

Unit 4 - CFLs and DCFLs are not closed under Intersection

If L_1 and If L_2 are two context free languages, their intersection L1 ∩ L2 need not be context free. Consider the following CFLs:

$L_1 = \{ a^n b^n c^m : n >= 0, m >= 0 \}$

$L_2 = \{ a^n b^m c^m : n >= 0, m >= 0 \}$

Consider language L which represents the intersection of these two languages :

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

This language L is not a CFL.

L_1 says number of a's should be equal to number of b's and L_2 says number of b's should be equal to number of c's. Their intersection says both conditions need to be true, but push down automata can compare only two. So it cannot be accepted by pushdown automata, hence the language is not context free.

Unit 4 - CFLs are not closed under Complement

Using DeMorgan's Law:

- $L_1 \cap L_2 = \neg(\neg L_1 \cup \neg L_2)$
- Recall that the context-free languages are closed under union
- So if they were closed under complement, they would also be closed under intersection (which they are not)

Unit 4 - DCFLs are not closed under Union

Consider the following DCFLs for which we can easily construct a DPDA:

$L_1 = \{ a^n b^n c^m \ : n >= 0, m >= 0 \}$

$L_2 = \{ a^n b^m c^m : n >= 0, m >= 0 \}$

Consider language L which represents the union of these two languages :

$L = \{ a^n b^n c^m \ \cup \ a^n b^m c^m \mid n >= 0, m >= 0 \}$

This language L cannot be recognized by DPDA because either number of a's and b's can be equal or either number of b's and c's can be equal. So, it can only be implemented by NPDA.

Thus, it is CFL but not DCFL.

Hence DCFLs are not closed under Union.

Unit 4 - DCFLs are not closed under Concatenation

It is possible that the suffix of a former DCFL matches the prefix of a later DCFL. The DPDA would not know which part it is currently processing and would need non-determinism to correctly recognize the strings of the newly formed language. Hence, DCFLs are NOT closed under concatenation.

Consider the following DCFLs for which we can easily construct a DPDA:

$L_1 = \{a^m b^n \mid m < n\}$

$L_2 = \{wcw^R \mid w \in \{a, b\}^*$

Consider language L which represents the concatenation of these two languages :

$L=\{a^m b^n \mid m < n\} \cdot \{wcw^R \mid w \in \{a, b\}^*\}$

The string abbb·bbbacabbb is accepted, but how would you know on which b you should ”jump across” from one language to the other and stop consuming the b and start stacking it?

Therefore, $L=\{a^m b^n \mid m < n\} \cdot \{wcw^R \mid w \in \{a, b\}^*\}$ is NOT a DCFL but CFL.

Automata Formal Languages and Logic

Unit 4 - DCFLs are not closed under Star Closure

Let L=L$_1$ U L$_2$ = $\{ca^nb^n | n \in \mathbb{N} \cup \{0\}\} \cup \{a^mb^{2m} | m \in \mathbb{N}\}$, which is a deterministic context free language thanks to the letter c in front of every word in L$_1$. If L* is the Kleene closure of L, any string $x \in$ L* should be a concatenated permutation of words (let's say y$_i$) which belong to L , with the concatenation repeated numerably many times, zero included.

Now let y_1=c and y_2=ab^2, which are both clearly in L. This can be seen by letting n=0 and m=1 in the sets L$_1$ and L$_2$. Because of this, the word cab^2 is in L* , which can be obtained by taking the first iteration to generate c and second iteration to generate ab^2 from the set and forming a concatenation of these two words.

As both parts of the union that form L are DCFLs, they are both recognized by the deterministic pushdown automata (DPDA). In order to recognize the word y_1y_2=cab^2, you would have to form a new automaton, by connecting the individual automata appropriately. However, this connection cannot be made deterministically.

Unit 4 - DCFLs are not closed under Reversal

Example 1 :

Consider the language L = $WcW^R(a+b+c)*$, where $W \in (a+b)^+$; this is deterministic because you can write a deterministic PDA to accept simple palindromes of the form W^RcW and modify the accepting state to loop to itself on any of a, b and c.

The reverse of the language L, denoted by $L^R = (a+b+c)^*WcW^R$, where $W \in (a+b)^+$; this is non-deterministic because you don't know where the WcW bit starts.

Hence, DCFLs are not closed under reversal.

Unit 4 - DCFLs are not closed under Reversal

Example 2 :

L = {x0y | |x| = |y|, $x \in 1^*$, $y \in \{0, 1\}^*$ }

The language L is deterministic context-free.

A deterministic PDA makes use of the stack to check to see if the two sides have the same length with the first occurrence of a 0 considered as the middle symbol of the input.

L^R = {y0x | |x| = |y|, $x \in 1^*$, $y \in \{0,1\}^*$ }

Reversing the language will present difficulty for a deterministic PDA.

While a machine can easily identify the first occurrence of a 0, it cannot decide which occurrence of 0 is the last occurrence unless the machine has finished reading the whole input.

Unit 4 - DCFLs are closed under Complement

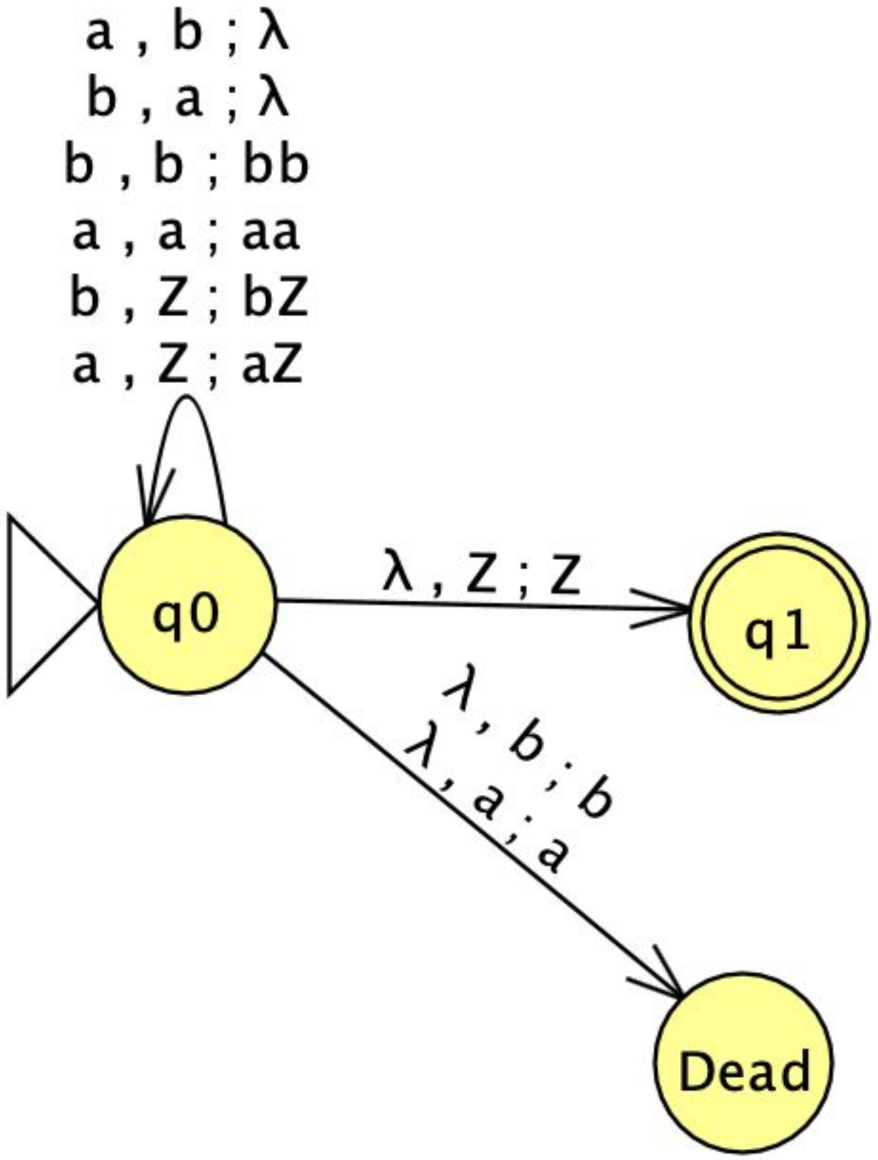
The class of DCFLs is closed under complementation.

The DPDA must be complete in order to take complement.

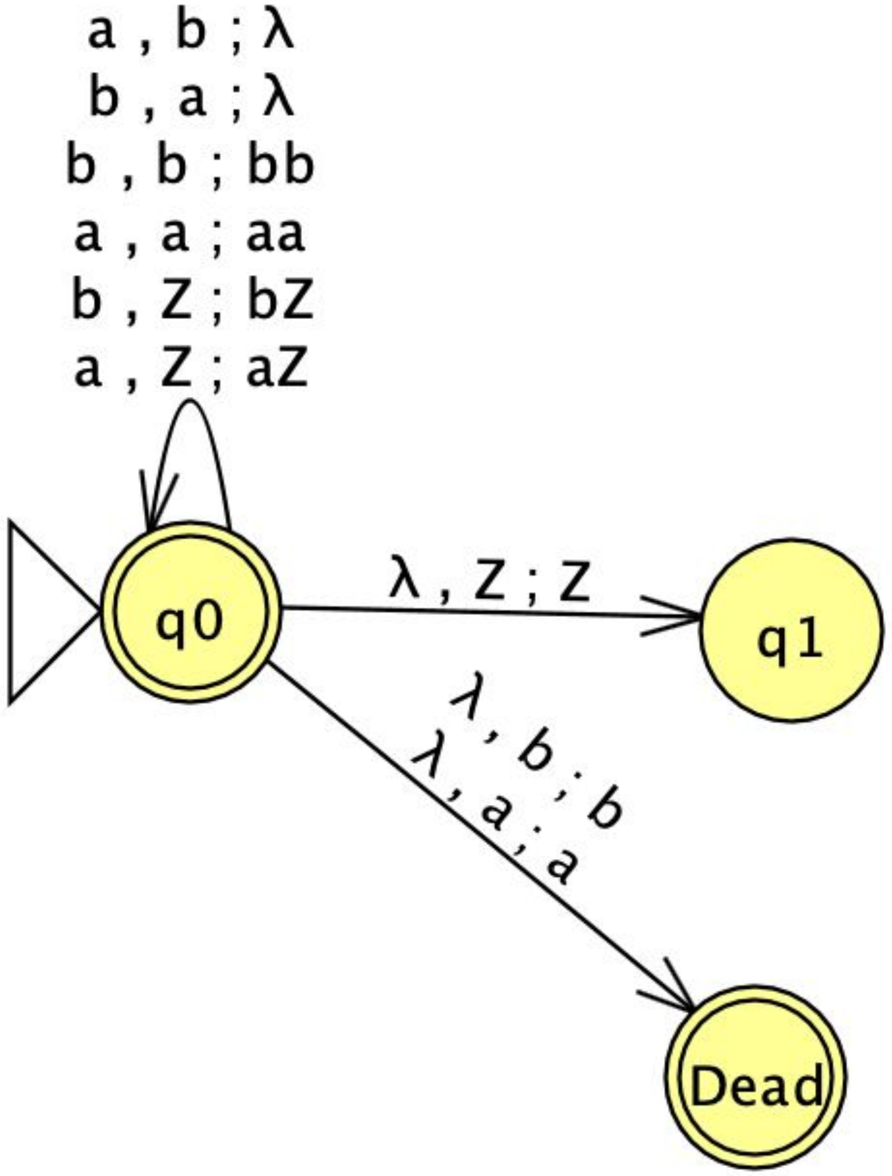
Introduce a dead state for invalid configurations and then toggle final and non-final states.

Unit 4 - DCFLs are closed under Complement

Example:
$L = \{n_a(w) = n_b(w), w \in \{a,b\}^*\}$
A complete DPDA for L:



Consider:
$L^c = \{n_a(w) \neq n_b(w), w \in \{a,b\}^*\}$
We complement DPDA for L:



Unit 4 - Interesting question on Closure properties

What is the complement of the Context free language L which represents an even length palindrome given as :
L = {WW^R, W {a,b}*}

Is the complement of L context free?

If L and its complement are context free, does that mean its a DCFL?

Automata Formal Languages and Logic

Unit 4 - Interesting questions on Closure properties

Solution:

WW^R means even length palindrome so its complement must accept

1) Odd length string

2) Even length string which are not palindrome

So here we can get help of non determinism, we perform push and at arbitrary point X assuming it to be half of string now we pop so there are few possibility

1) at least one symbol did not match (success)

2) Stack is empty and no input string is remaining to read (failure)

case 1 is definitely true for odd length string thus accepting but it also help in accepting even length string which are not palindrome.

case 2 is true when string is even length palindrome so thus at the end we will make a transition to non final state.

Unit 4 - Decidable Questions about CFLs

Is it possible to write an algorithm to answer the following about context free language(s)??

1. Given a CFL L, is it possible to determine if L is infinite?
   Sol :- if the dependency graph of Non-terminals have a cycle.

2. Given a CFL L, is it possible to determine if L is empty i.e. $L = \Phi$?
   Sol :- if start symbol of Grammar for L is useless.

3. Given a CFL L, is it possible to determine if L generates everything, i.e. $L = \Sigma^*$?
   Sol :- Decidable for DCFLs but not for CFLs as DCFLs are closed under complement.

4. Given a CFG G and a string w, is it possible to determine if $w \in L(G)$?
   Sol :- Using CYK

Unit 4 - Meaning of Undecidability

Undecidability means there is no algorithm that provides correct answer for all inputs.

The following properties of CFLs are undecidable:

1. **Given a CFG G, is it possible to determine if G is ambiguous?**
   Sol :- Not possible

2. **Given two CFLs L_1 and L_2, is it possible to determine if $L_1 = L_2$ i.e. if they generate the same language?**
   Sol :- If not, we will eventually get an answer. If yes, we will never get an answer as both the languages are infinite.

3. **Given two CFLs L1 and L2, is it possible to determine if $L_1 \cap L_2 = \Phi$?**
   Sol :- if the languages have string in common, we will eventually get an answer, if not we will wait forever as the languages are infinite.

THANK YOU

Preet Kanwal

Department of Computer Science & Engineering

preetkanwal@pes.edu

+91 80 6666 3333 Extn 724