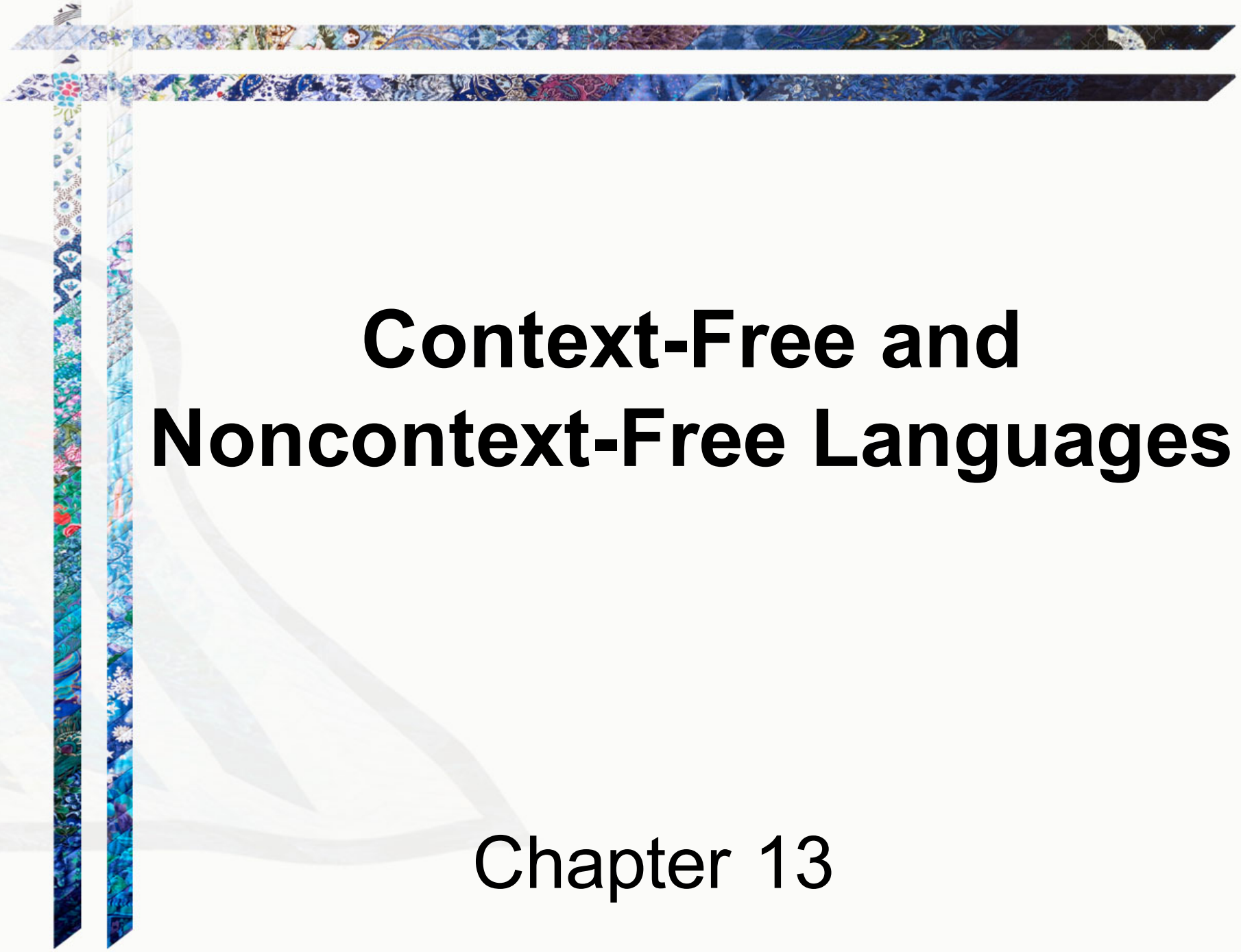


Context-Free and Noncontext-Free Languages

Chapter 13

Languages That Are and Are Not Context-Free

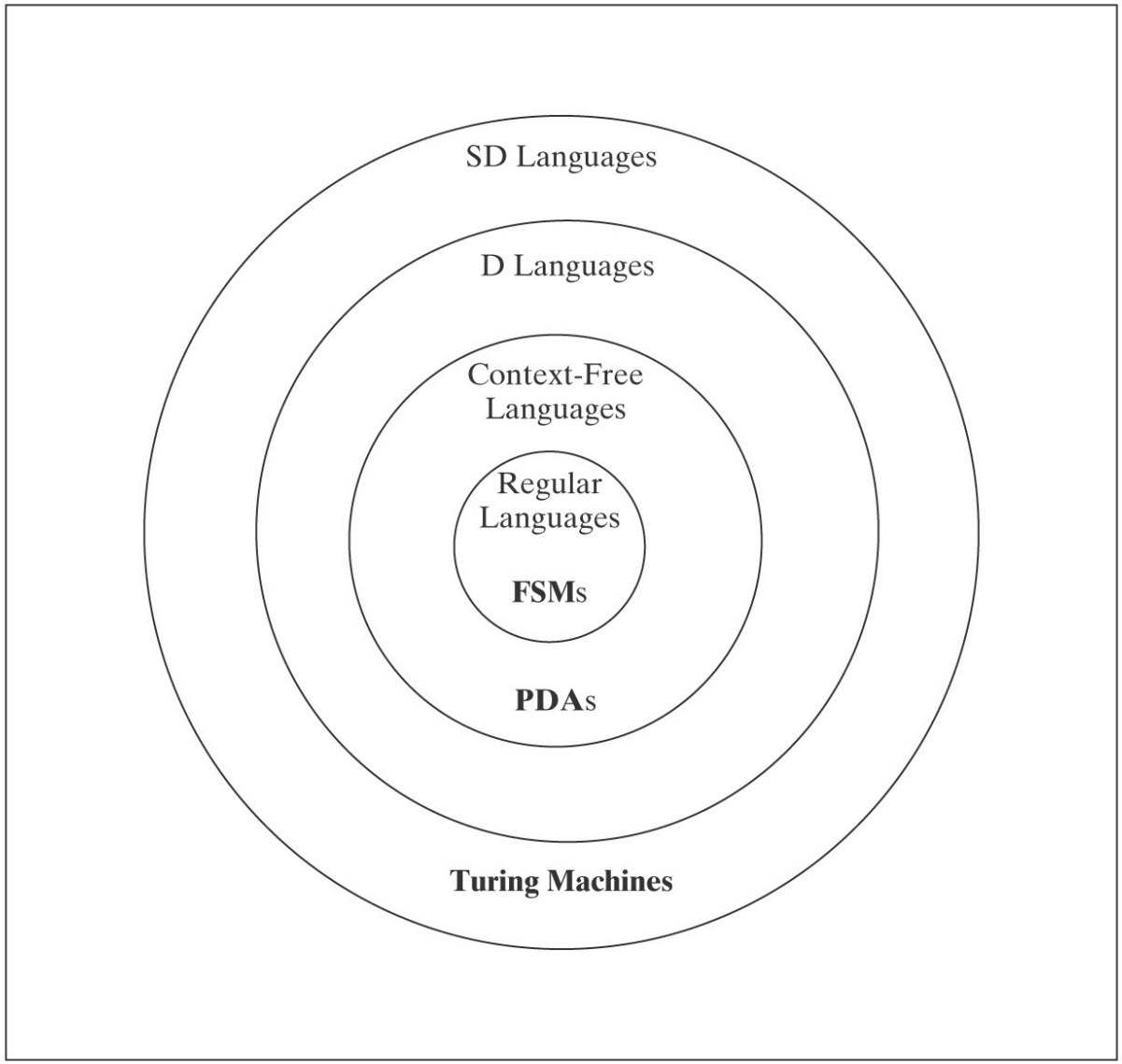
a^*b^* is regular.

$A^nB^n = \{a^n b^n : n \geq 0\}$ is context-free but not regular.

$A^nB^nC^n = \{a^n b^n c^n : n \geq 0\}$ is not context-free.

- Intuitively because a PDA's stack cannot count all three of the letter regions and compare them

Languages and Machines



The Regular and the CF Languages

Theorem: The regular languages are a proper subset of the context-free languages.

Proof: In two parts:

- Every regular language is CF.
- There exists at least one language that is CF but not regular.

The Regular and the CF Languages

Lemma: Every regular language is CF.

Proof: Every FSM is (trivially) a PDA:

Given an FSM $M = (K, \Sigma, \Delta, s, A)$ and elements of δ of the form: (p, c, q)

p: old state, c: input, q: new state

Construct a PDA $M' = (K, \Sigma, \{\varnothing\}, \Delta, s, A)$. Each (p, c, q) becomes:

$((p, c, \varepsilon), (q, \varepsilon))$

p: old state, c: input, ε: don't look at stack; q: new state, ε: don't push on stack

In other words, **we just don't use the stack**.

There Exists at Least One Language that is CF but Not Regular

Lemma: There exists at least one language that is CF but not regular

Proof: $\{a^n b^n, n \geq 0\}$ is context-free but not regular.

So the regular languages are a proper subset of the context-free languages.

How Many Context-Free Languages Are There?

Theorem: There is a countably infinite number of CFLs.

Proof:

- Upper bound: we can lexicographically enumerate all the CFGs.

- Lower bound: every regular language is context-free and there is a countably infinite number of regular languages

• So, there is at most and at least a countably infinite number of context-free langauges.

How Many Context-Free Languages Are There?

There is an uncountable number of languages (theorem 2.3).

Thus there are more languages than there are context-free languages.

So there must exist some languages that are not context-free.

Example: $\{a^n b^n c^n : n \geq 0\}$

Showing that *L* is Context-Free

Techniques for showing that a language *L* is context-free:

1. Exhibit a context-free grammar for *L*.

2. Exhibit a PDA for *L*.

3. Use the closure properties of context-free languages.
   - Unfortunately, there are fewer closure theorems than regular languages.

Closure Theorems for Context-Free Languages

The context-free languages are closed under:

- Union (natural, combination of CF grammar rules, combination of PDAs by episilon-transitions …)
- Concatenation
- Kleene star
- Reverse
- Letter substitution

What About Intersection?

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

The proof is by counterexample. Let:

$L_1 = \{a^n b^n c^m : n, m \geq 0\}$ /* equal a's and b's.
$L_2 = \{a^m b^n c^n : n, m \geq 0\}$ /* equal b's and c's.

Both L_1 and L_2 are context-free, since there exist straightforward context-free grammars for them.

But now consider:

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

What About Complement?

The context-free languages are **not** closed under complement:

Closure under complement implies closure under intersection, since:

$$L_1 \cap L_2 = \neg(\neg L_1 \cup \neg L_2)$$

The context-free languages are closed under union, so if they were closed under complement, they would be closed under intersection (which they are not).

Why are the Context-Free Languages Not Closed under Complement, Intersection and Subtraction But the Regular Languages Are?

Given an NDFSM M_1, build an FSM M_2 such that $L(M_2) = \neg L(M_1)$:

1. From M_1, construct an equivalent deterministic FSM M', using *ndfsmtodfsm*.
2. If M' is described with an implied dead state, add the dead state and all required transitions to it.
3. Begin building M_2 by setting it equal to M'. Then swap the accepting and the nonaccepting states. So:

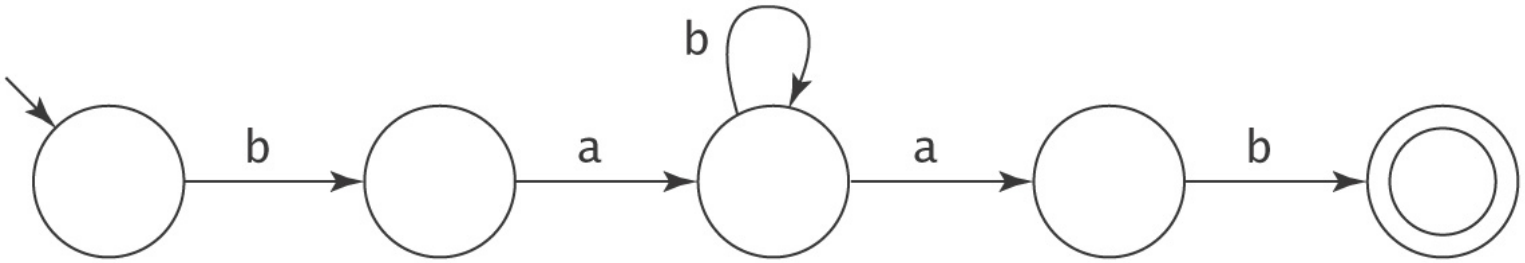
$$M_2 = (K_{M'}, \Sigma, \delta_{M'}, s_{M'}, K_{M'} - A_{M'}).$$

We could do the same thing for CF languages if we could do step 1, but we can't.

The need for nondeterminism is the key.

Showing that *L* is Not Context-Free

Remember the pumping argument for regular languages:

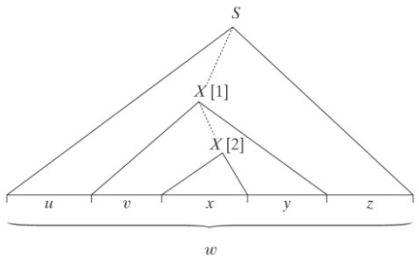


Proof by contradiction: If *L* were context-free, then it would possess certain properties. But it does not posses those properties. Therefore, *L* is not context-free.

Context-free pumping theorem is based on the structure of parse trees.

The Context-Free Pumping Theorem

If L is a context-free language, then $\exists k \geq 1$, such that
$\forall$ strings $w \in L$, where $|w| \geq k$,
$\exists$ u, v, x, y, z, such that: $w = uvxyz$, and
$vy \neq \varepsilon$, and
$|vxy| \leq k$, and
$\forall q \geq 0$, $uv^q xy^q z$ is in L.



Proof: L is generated by some CFG $G = (V, \Sigma, R, S)$ with n nonterminal symbols and branching factor b. Let k be b^{n+1}. The longest string that can be generated by G with no repeated nonterminals in the resulting parse tree has length b^n. Assuming that $b \geq 2$, it must be the case that $b^{n+1} > b^n$. So let w be any string in $L(G)$ where $|w| \geq k$. Let T be any smallest parse tree for w. T must have height at least $n + 1$. Choose some path in T of length at least $n + 1$. Let X be the bottom-most repeated nonterminal along that path. Then w can be rewritten as $uvxyz$. The tree rooted at [1] has height at most $n + 1$. Thus its yield, vxy, has length less than or equal to b^{n+1}, which is k. $vy \neq \varepsilon$ since if vy were ε then there would be a smaller parse tree for w and we chose T so that that wasn't so. uxz must be in L because $rule_2$ could have been used immediately at [1]. For any $q \geq 1$, $uv^q xy^q z$ must be in L because $rule_1$ could have been used q times before finally using $rule_2$.

The Pumping Theorem for Regular Languages

If L is regular, then every long string in L is pumpable.

So, $\exists k \geq 1$

($\forall$ strings $w \in L$, where $|w| \geq k$

($\exists\, x, y, z$ ($w = xyz$,
$|xy| \leq k$,
$y \neq \varepsilon$, and
$\forall q \geq 0$ ($xy^q z$ is in L)))).

- context free pumping lemma generalizes that regular pumping lemma?

Regular vs CF Pumping Theorems

Similarities:

- We choose *w*, the string to be pumped.
- We choose a value for *q* that shows that *w* isn't pumpable.
- We may apply closure theorems before we start.

Differences:

- Two regions, *v* and *y*, must be pumped in tandem.
- We don't know anything about where in the strings *v* and *y* will fall. All we know is that they are reasonably "close together", i.e., $|vxy| \leq k$.
- Either *v* or *y* could be empty, although not both.

An Example of Pumping: $A^nB^nC^n$

$A^nB^nC^n = \{a^n b^n c^n, n \geq 0\}$

Choose $w = a^k b^k c^k$

1 | 2 | 3

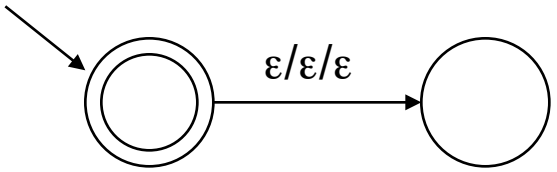
If either *v* or *y* spans regions, then let *q* = 2 (i.e., pump in once). The resulting string will have letters out of order and thus not be in $A^nB^nC^n$.

If both *v* and *y* each contain only one distinct character then set *q* to 2. Additional copies of at most two different characters are added, leaving the third unchanged. There are no longer equal numbers of the three letters, so the resulting string is not in $A^nB^nC^n$.

Deterministic PDAs

A PDA *M* is ***deterministic*** iff:

- Δ_M contains no pairs of transitions that compete with each other, and
- Whenever *M* is in an accepting configuration it has no available moves.



M can choose between accepting and taking the ε-transition, so it is not deterministic.

Deterministic CFLs

A language L is ***deterministic context-free*** iff $L\$$ can be accepted by some deterministic PDA.

$: end of string marker Why $?

Let $L = \mathtt{a}^* \cup \{\mathtt{a}^n\mathtt{b}^n : n > 0\}$.

• When it begins reading a's, M must push them onto the stack in case there are going to be b's it runs out of input without seeing b's, it needs a way to pop out the a's from the stack before it accepts.

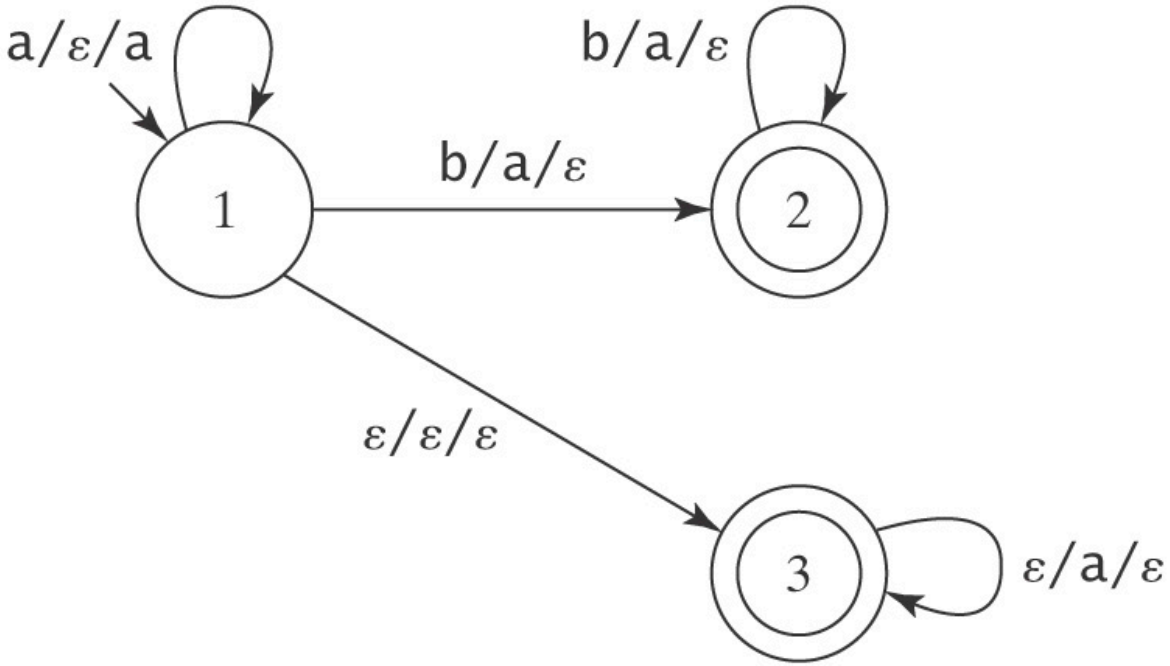
  • recall how "accept" was defined

• Add $ to make it easier to build DPDAs, it does not add power (to allow building a PDA for L that was not context-free)

• There exist CLFs that are not deterministic, e.g., $L = \{\mathtt{a}^i\mathtt{b}^j\mathtt{c}^k, i \neq j \text{ or } j \neq k\}$.

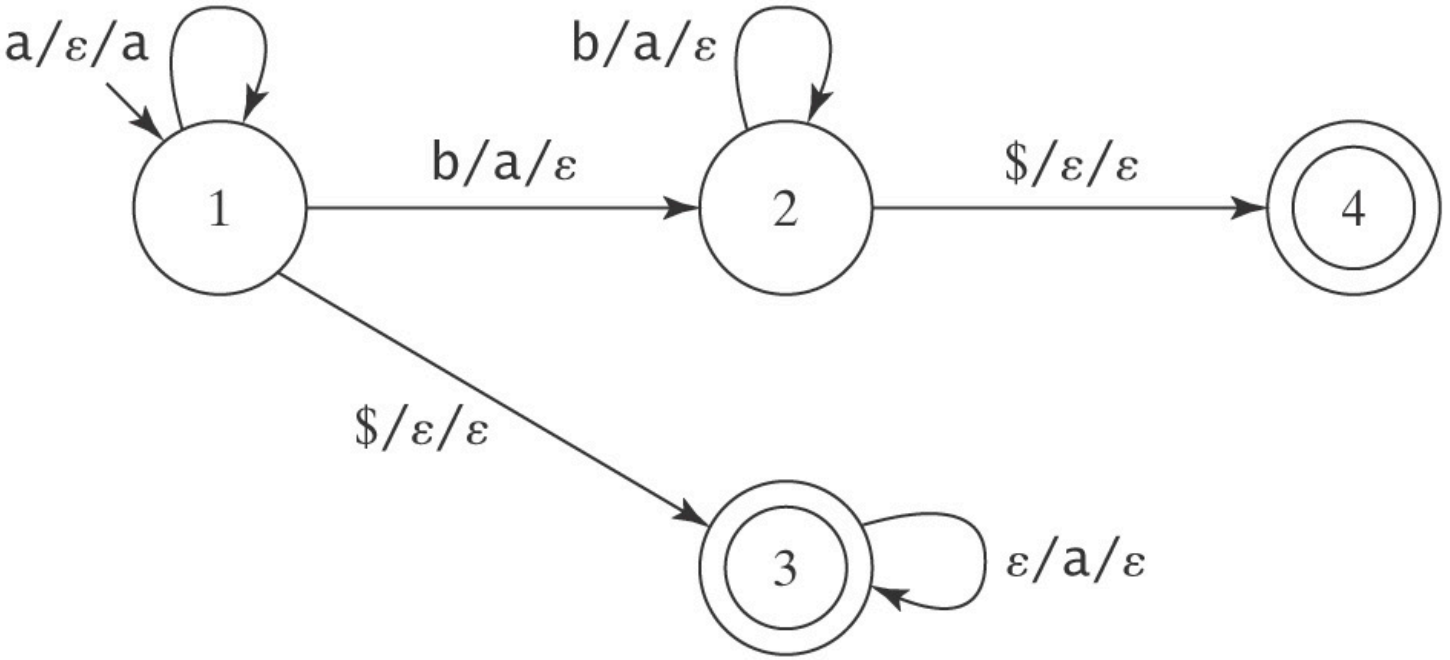
An NDPDA for *L*

$L = a^* \cup \{a^n b^n : n > 0\}$.



A DPDA for *L*$

$L = a^* \cup \{a^n b^n : n > 0\}$.



DCFLs are Closed Under Complement

Proof by construction.

$L_1 \cap L_2 = \neg(\neg L_1 \cup \neg L_2)$.

What about intersection and union?

DCFLs are Not Closed Under Union

$L_1 = \{a^i b^j c^k, i, j, k \geq 0 \text{ and } i \neq j\}$. (a DCFL)
$L_2 = \{a^i b^j c^k, i, j, k \geq 0 \text{ and } j \neq k\}$. (a DCFL)

$L' = L_1 \cup L_2$.
$= \{a^i b^j c^k, i, j, k \geq 0 \text{ and } (i \neq j) \text{ or } (j \neq k)\}$.

$L'' = \neg L'$.
$= \{a^i b^j c^k, i, j, k \geq 0 \text{ and } i = j = k\} \cup$
$\{w \in \{a, b, c\}^* : \text{the letters are out of order}\}$.

$L''' = L'' \cap a^*b^*c^*$.
$= \{a^n b^n c^n, n \geq 0\}$.

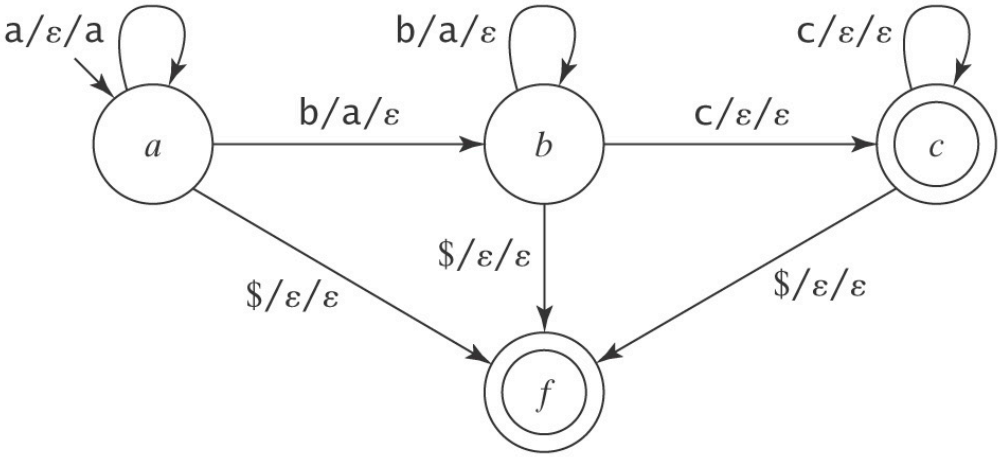
L''' is not even CF, much less DCF.

DCFLs are Not Closed Under Intersection

$L_1 = \{a^i b^j c^k, i, j, k \geq 0 \text{ and } i = j\}$.
$L_2 = \{a^i b^j c^k, i, j, k \geq 0 \text{ and } j = k\}$.

$L' = L_1 \cap L_2$
$=$

L_1 and L_2 are deterministic context-free:



Nondeterministic CFLs

Theorem: There exist CLFs that are not deterministic.

Proof: By example. Let $L = \{a^i b^j c^k, i \neq j \text{ or } j \neq k\}$. L is CF. If L is DCF then so is:

$$L' = \neg L$$
$$= \{a^i b^j c^k, i, j, k \geq 0 \text{ and } i = j = k\} \cup$$
$$\{w \in \{a, b, c\}^* : \text{the letters are out of order}\}.$$

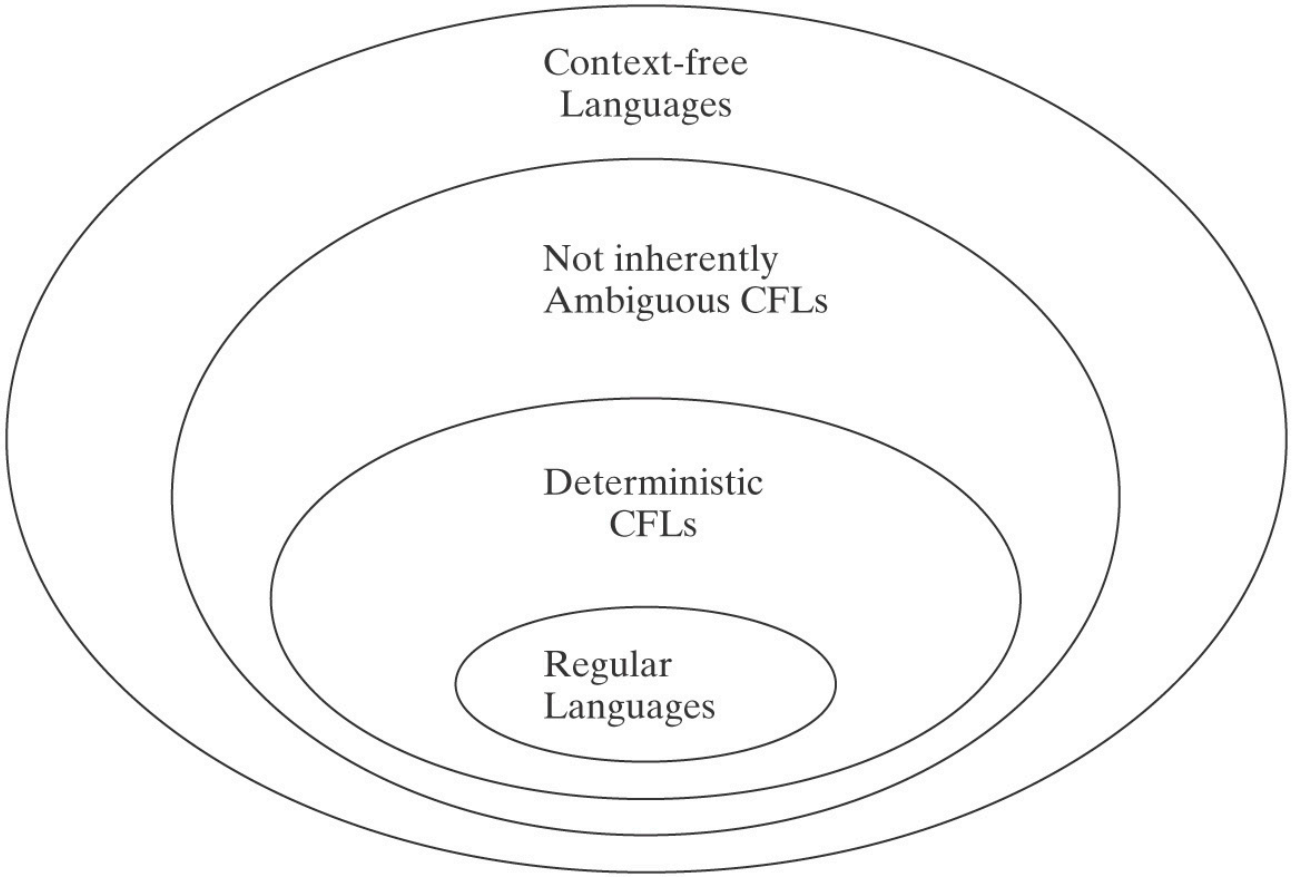
But then so is:

$L'' = L' \cap a^*b^*c^*$.
$= \{a^n b^n c^n, n \geq 0\}$.

But it isn't. So L is context-free but not deterministic context-free.

This simple fact poses a real problem for the designers of efficient context-free parsers.

The CFL Hierarchy



Algorithms and Decision Procedures for Context-Free Languages

Chapter 14

Decidability of CFLs

Theorem: Given a context free language *L* and a string *w*, there exists a decision procedure that answers the questions, "is *w* in *L*?"

Two approaches:

- Find a context-free grammar to generate it
- Find a PDA to accept it

Using a Grammar

decideCFLusingGrammar(*L*: CFL, *w*: string) =

1. If given a PDA, build *G* so that $L(G) = L(M)$.

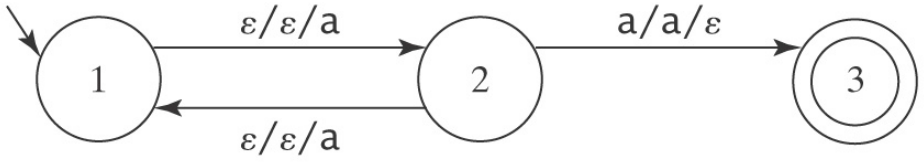
2. If $w = \varepsilon$ then if S_G is nullable then accept, else reject.

3. If $w \neq \varepsilon$ then:
   3.1 Construct G' in Chomsky normal form such that $L(G') = L(G) - \{\varepsilon\}$.

   3.2 If *G* derives *w*, it does so in $2 \cdot |w| - 1$ steps. Try all derivations in *G* of $2 \cdot |w| - 1$ steps. If one of them derives *w*, accept. Otherwise reject.

Using a PDA

Problem:



Theorem: Given any context-free language grammar $G = (V, \Sigma, R, S)$, there exists a PDA M such that $L(M) = L(G) - \{\varepsilon\}$ and M contains no transitions of the form $((q_1, \varepsilon, \alpha), (q_2, \beta))$. In other words, every transition reads exactly one input character.

Greibach Normal Form

All rules are of the following form:

- $X \rightarrow a\,A$, where $a \in \Sigma$ and $A \in (V - \Sigma)^*$.

No need to push the *a* and then immediately pop it.

So $M = (\{p, q\}, \Sigma, V, \Delta, p, \{q\})$, where Δ contains:

1. The start-up transitions:
   For each rule $S \rightarrow cs_2 \ldots s_n$, the transition:
   $((p, c, \varepsilon), (q, s_2 \ldots s_n))$.

2. For each rule $X \rightarrow cs_2 \ldots s_n$ (where $c \in \Sigma$ and s_2 through s_n are elements of $V - \Sigma$), the transition:

   $((q, c, X), (q, s_2 \ldots s_n))$

A PDA Without ε-transitions Must Halt

Consider the execution of M on input w:

- Each individual path of M must halt within $|w|$ steps.

- The total number of paths pursued by M must be less than or equal to $P = B^{|w|}$, where B is the maximum number of competing transitions from any state in M.

- The total number of steps that will be executed by all paths of M is bounded by $P * |w|$.

So all paths must eventually halt.

- **The same for NDFSM**

An Algorithm to Decide Whether *M* Accepts *w*

decideCFLusingPDA(*L*: CFL, *w*: string) =

1. If *L* is specified as a PDA, use *PDAtoCFG* to construct a grammar *G* such that *L*(*G*) = *L*(*M*).
2. If *L* is specified as a grammar *G*, simply use *G*.
3. If $w = \varepsilon$ then if S_G is nullable then accept, otherwise reject.
4. If $w \neq \varepsilon$ then:
    - 4.1 From *G*, construct *G′* such that *L*(*G′*) = *L*(*G*) – {ε} and *G′* is in Greibach normal form.
    - 4.2 From *G′* construct a PDA *M* such that *L*(*M*) = *L*(*G′*) and *M′* has no ε-transitions.
    - 4.3 All paths of *M* are guaranteed to halt within a finite number of steps. So run *M* on *w*. Accept if it accepts and reject otherwise.

What can you say about PDA's halting behavior?

- A PDA *M* must halt ?
- For an arbitrary PDA *M*, there exists *M'* that halts and *L*(*M'*) = *L*(*M*) ?

Decidability of Emptiness and Finiteness

Theorem: Given a context free language L, there exists a decision procedure that answers the following questions:

1. Given a context-free language L, is $L = \emptyset$?
2. Given a context-free language L, is L infinite?

Equivalence of DCFLs

Theorem: Given two *deterministic* context-free languages L_1 and L_2, there exists a decision procedure to determine whether $L_1 = L_2$.

The Undecidable Questions about CFLs

- Is $L = \Sigma^*$?
- Is the complement of L context-free?
- Is L regular?
- Is $L_1 = L_2$?
- Is $L_1 \subseteq L_2$?
- Is $L_1 \cap L_2 = \varnothing$?
- Is L inherently ambiguous?
- Is G ambiguous?

CFL Summary

- The theory of CFL is not as tidy as the theory of RL

- Interesting subsets, DCFL, and not inherently ambiguous CFL, are only proper subset of CFL
- Not closed under many common operations
- No algorithm for minimizing PDAs
- No fast recognition algorithm that works on arbitrary CFL
- No decision procedure for many important questions

- Yet substantial effort has been invested in CFLs as they are useful

RLs vs. CFLs

Regular Languages

- regular exprs.
  - or
- regular grammars
- = DFSMs
- recognize
- minimize FSMs

- closed under:
  - concatenation
  - union
  - Kleene star
  - complement
  - intersection
- pumping theorem
- D = ND

Context-Free Languages

- context-free grammars

- = NDPDAs
- parse
- find unambiguous grammars
- reduce nondeterminism in PDAs
- find efficient parsers
- closed under:
  - concatenation
  - union
  - Kleene star

  - intersection w/ reg. langs
- pumping theorem
- D ≠ ND