Languages That Are and Are Not Context-Free

Read K & S 3.5, 3.6, 3.7.
Read Supplementary Materials: Context-Free Languages and Pushdown Automata: Closure Properties of Context-Free Languages
Read Supplementary Materials: Context-Free Languages and Pushdown Automata: The Context-Free Pumping Lemma.
Do Homework 16.

Deciding Whether a Language is Context-Free

Theorem: There exist languages that are not context-free.

Proof:
(1) There are a countably infinite number of context-free languages. This true because every description of a context-free language is of finite length, so there are a countably infinite number of such descriptions.

(2) There are an uncountable number of languages.

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

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

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

Showing that a Language 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, these are weaker than they are for regular languages.

The Context-Free Languages are Closed Under Union

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

Assume that G_1 and G_2 have disjoint sets of nonterminals, not including S.

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

We can show that L is context-free by exhibiting a CFG for it:

The Context-Free Languages are Closed Under Concatenation

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

Assume that G_1 and G_2 have disjoint sets of nonterminals, not including S.

Let $L = L(G_1)\, L(G_2)$

We can show that L is context-free by exhibiting a CFG for it:

The Context-Free Languages are Closed Under Kleene Star

Let $G_1 = (V_1, \Sigma_1, R_1, S_1)$

Assume that G_1 does not have the nonterminal S.

Let $L = L(G_1)^*$

We can show that L is context-free by exhibiting a CFG for it:

What About Intersection and Complement?

We know that they share a fate, since

$$L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$$

But what fate?

We proved closure for regular languages two different ways. Can we use either of them here:

1. Given a deterministic automaton for L, construct an automaton for its complement. Argue that, if closed under complement and union, must be closed under intersection.
2. Given automata for L_1 and L_2, construct a new automaton for $L_1 \cap L_2$ by simulating the parallel operation of the two original machines, using states that are the Cartesian product of the sets of states of the two original machines.

More on this later.

The Intersection of a Context-Free Language and a Regular Language is Context-Free

$L = L(M_1)$, a PDA $= (K_1, \Sigma, \Gamma_1, \Delta_1, s_1, F_1)$
$R = L(M_2)$, a deterministic FSA $= (K_2, \Sigma, \delta, s_2, F_2)$

We construct a new PDA, M_3, that accepts $L \cap R$ by simulating the parallel execution of M_1 and M_2.

$M = (K_1 \times K_2, \Sigma, \Gamma_1, \Delta, (s_1, s_2), F_1 \times F_2)$

Insert into Δ:

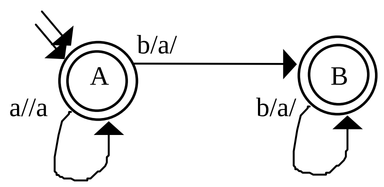
For each rule $((q_1, \quad a, \beta), \; (p_1, \quad \gamma))$ in Δ_1,
and each rule $(q_2, \quad a, \quad p_2)$ in δ,
$(((q_1, q_2), \; a, \beta), ((p_1, p_2), \gamma))$

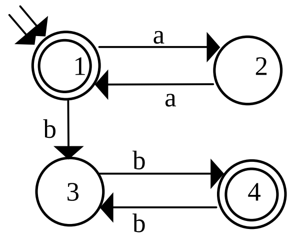
For each rule $((q_1, \quad \varepsilon, \beta), \; (p_1, \quad \gamma)$ in Δ_1,
and each state q_2 in K_2,
$(((q_1, q_2), \varepsilon, \beta), ((p_1, q_2), \gamma))$

This works because: we can get away with only one stack.

Example

L = a^nb^n $\cap$ (aa)*(bb)*





((A, a, ε), (A, a))	(1, a, 2)
((A, b, a), (B, ε))	(1, b, 3)
((B, b, a), (B, ε))	(2, a, 1)
	(3, b, 4)
	(4, b, 3)

A PDA for L:

Don't Try to Use Closure Backwards

One Closure Theorem:
If L_1 and L_2 are context free, then so is

$$\underset{\uparrow}{L_3} = \underline{L_1} \cup \underline{L_2}.$$

But what if L_3 and L_1 are context free? What can we say about L_2?

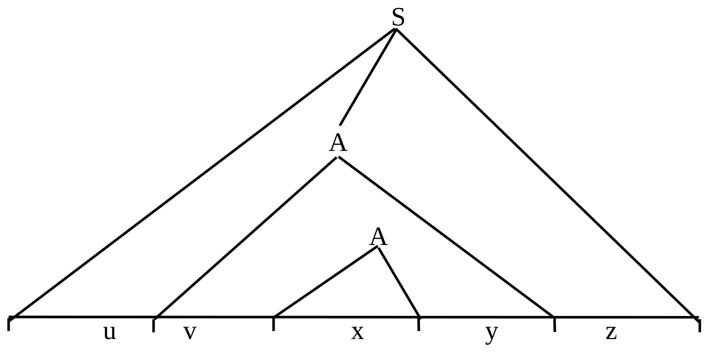
$$\underline{L_3} = \underline{L_1} \cup \underset{\uparrow}{L_2}.$$

Example:

$$a^nb^nc^* = a^nb^nc^* \cup a^nb^nc^n$$

The Context-Free Pumping Lemma

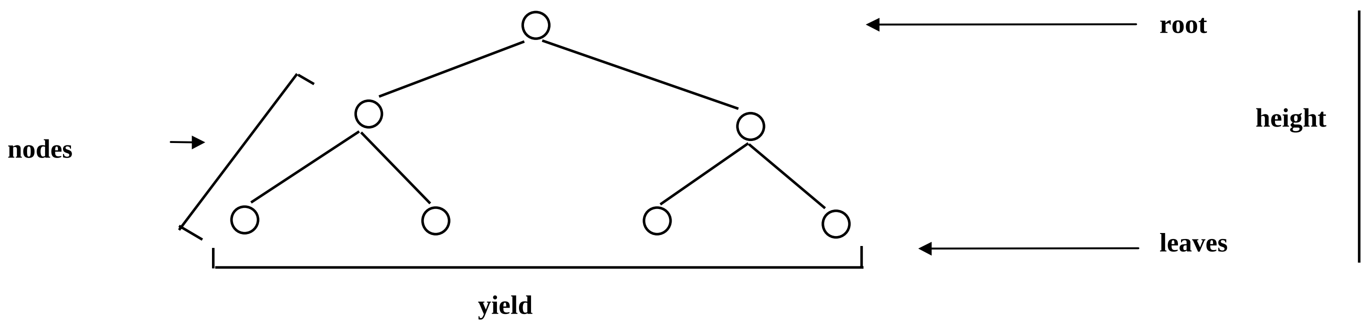
This time we use parse trees, not automata as the basis for our argument.



If L is a context-free language, and if w is a string in L where |w| > K, for some value of K, then w can be rewritten as uvxyz, where |vy| > 0 and |vxy| ≤ M, for some value of M.

uxz, uvxyz, uvvxyyz, uvvvxyyyz, etc. (i.e., uv^nxy^nz, for $n \geq 0$) are all in L.

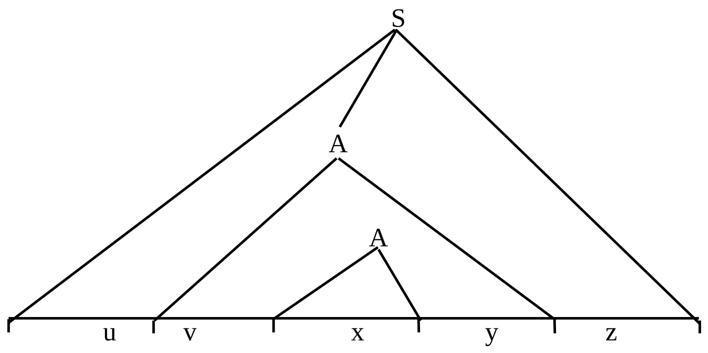
Some Tree Basics



Theorem: The length of the yield of any tree T with height H and branching factor (**fanout**) B is $\leq B^H$.

Proof: By induction on H. If H is 1, then just a single rule applies. By definition of fanout, the longest yield is B.
Assume true for H = n.
Consider a tree with H = n + 1. It consists of a root, and some number of subtrees, each of which is of height $\leq n$ (so induction hypothesis holds) and yield $\leq B^n$. The number of subtrees $\leq$ B. So the yield must be $\leq B(B^n)$ or B^{n+1}.

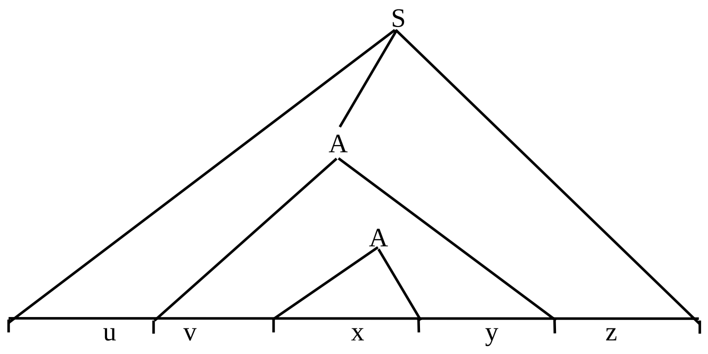
What Is K?



Let T be the number of nonterminals in G.
If there is a tree of height > T, then some nonterminal occurs more than once on some path. If it does, we can pump its yield.
Since a tree of height = T can produce only strings of length $\leq B^T$, any string of length $> B^T$ must have a repeated nonterminal and thus be pumpable.

So $K = B^T$, where T is the number of nonterminals in G and B is the branching factor (fanout).

What is M?



Assume that we are considering the bottom most two occurrences of some nonterminal. Then the yield of the upper one is at most B^{T+1} (since only one nonterminal repeats).

So $M = B^{T+1}$.

The Context-Free Pumping Lemma

Theorem: Let G = (V, Σ, R, S) be a context-free grammar with T nonterminal symbols and fanout B. Then any string $w \in L(G)$ where $|w| > K$ (B^T) can be rewritten as w = uvxyz in such a way that:

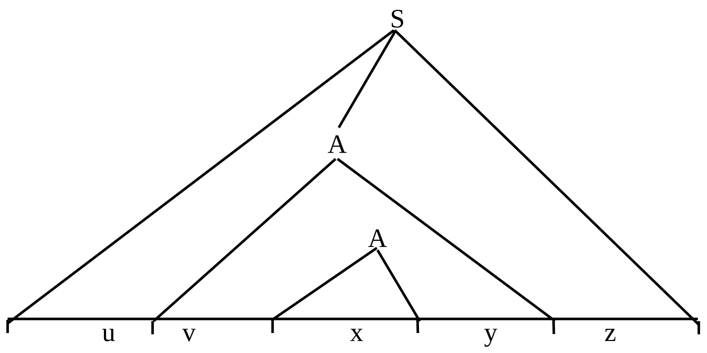
- $|vy| > 0$,
- $|vxy| \le M$ (B^{T+1}), (making this the "strong" form),
- for every $n \ge 0$, uv^nxy^nz is in L(G).

Proof:
Let w be such a string and let T be the parse tree with root labeled S and with yield w that has the smallest number of leaves among all parse trees with the same root and yield. T has a path of length at least T+1, with a bottommost repeated nonterminal, which we'll call A. Clearly v and y can be repeated any number of times (including 0). If $|vy| = 0$, then there would be a tree with root S and yield w with fewer leaves than T. Finally, $|vxy| \le B^{T+1}$.

An Example of Pumping

$L = \{a^nb^nc^n : n \ge 0\}$

Choose $w = a^ib^ic^i$ where $i > \lceil K/3 \rceil$ (making $|w| > K$)



Unfortunately, we don't know where v and y fall. But there are two possibilities:

1. If vy contains all three symbols, then at least one of v or y must contain two of them. But then uvvxyyz contains at least one out of order symbol.
2. If vy contains only one or two of the symbols, then uvvxyyz must contain unequal numbers of the symbols.

Using the Strong Pumping Lemma for Context Free Languages

If L is context free, then
  There exist K and M (with $M \ge K$) such that
    For all strings w, where $|w| > K$,
      (Since true for all such w, it must be true for any paricular one, so you pick w)
      (Hint: describe w in terms of K or M)

      there exist u, v, x, y, z such that w = uvxyz and $|vy| > 0$, and
        $|vxy| \le M$, and
        for all $n \ge 0$, uv^nxy^nz is in L.

We need to **pick w**, then show that there are no values for uvxyz that satisfy all the above criteria. To do that, we just need to focus on possible values for v and y, the pumpable parts. So we **show that all possible picks for v and y violate at least one of the criteria**.

Write out a single string, w (in terms of K or M) **Divide w into regions**.

For each possibility for v and y (described in terms of the regions defined above), find some value n such that uv^nxy^nz is not in L. Almost always, the easiest values are 0 (pumping out) or 2 (pumping in). Your value for n may differ for different cases.

	v	**y**	**n**	**why the resulting string is not in L**
[1]				
[2]				
[3]				
[4]				
[5]				
[6]				
[7]				
[8]				
[9]				
[10]				

Convince the reader that there are no other cases.

Q. E. D.

A Pumping Lemma Proof in Full Detail

Proof that $L = \{a^n b^n c^n : n \geq 0\}$ is not context free.

Suppose L is context free. The context free pumping lemma applies to L. Let M be the number from the pumping lemma. Choose w = $a^M b^M c^M$. Now $w \in L$ and $|w| > M \geq K$. From the pumping lemma, for all strings w, where $|w| > K$, there exist u, v, x, y, z such that w = uvxyz and $|vy| > 0$, and $|vxy| \leq M$, and for all $n \geq 0$, $uv^n xy^n z$ is in L. There are two main cases:

1. Either v or y contains two or more different types of symbols (“a”, “b” or “c”). In this case, uv^2xy^2z is not of the form a*b*c* and hence $uv^2xy^2z \notin L$.
2. Neither v nor y contains two or more different types of symbols. In this case, vy may contain at most two types of symbols. The string uv^0xy^0z will decrease the count of one or two types of symbols, but not the third, so $uv^0xy^0z \notin L$

Cases 1 and 2 cover all the possibilities. Therefore, regardless of how w is partitioned, there is some uv^nxy^nz that is not in L. Contradiction. Therefore L is not context free.

Note: the underlined parts of the above proof is “boilerplate” that can be reused. A complete proof should have this text or something equivalent.

Context-Free Languages Over a Single-Letter Alphabet

Theorem: Any context-free language over a single-letter alphabet is regular.
Examples:

$L = \{a^n b^n\}$
$L' = \{a^n a^n\}$
$= \{a^{2n}\}$
$= \{w \in \{a\}^* : |w| \text{ is even}\}$

$L = \{ww^R : w \in \{a, b\}^*\}$
$L' = \{ww^R : w \in \{a\}^*\}$
$= \{ww : w \in \{a\}^*\}$
$= \{w \in \{a\}^* : |w| \text{ is even}\}$

$L = \{a^n b^m : n, m \geq 0 \text{ and } n \neq m\}$
$L' = \{a^n a^m : n, m \geq 0 \text{ and } n \neq m\}$
$=$

Proof: See Parikh's Theorem

Another Language That Is Not Context Free

L = $\{a^n : n \geq 1 \text{ is prime}\}$

Two ways to prove that L is not context free:

1. Use the pumping lemma:

Choose a string $w = a^n$ such that n is prime and n > K.

$w = \text{aaaaaaaaaaaaaaaaaaaaaaaa}$

$u \quad \overline{v} \quad x \quad \overline{y} \quad z$

Let $vy = a^p$ and $uxz = a^r$. Then $r + kp$ must be prime for all values of k. This can't be true, as we argued to show that L was not regular.

2. $|\Sigma_L| = 1$. So if L were context free, it would also be regular. But we know that it is not. So it is not context free either.

Using Pumping and Closure

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

L is not context free.

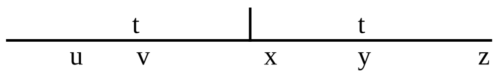
Try pumping: Let $w = a^K b^K c^K$

Now what?

Using Intersection with a Regular Language to Make Pumping Tractable

L = $\{tt : t \in \{a, b\}^* \}$

Let's try pumping: $|w| > K$



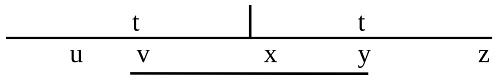
What if u is ε,
v is w,
x is ε,
y is w, and
z is ε

Then all pumping tells us is that $t^n t^n$ is in L.

$L = \{tt : t \in \{a, b\}^*\}$

What if we let $|w| > M$, i.e. choose to pump the string a^Mba^Mb:

Now v and y can't be t, since $|vxy| \leq M$:

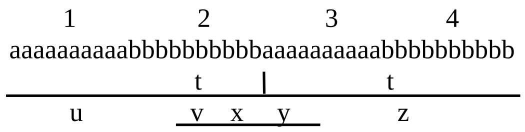


Suppose $|v| = |y|$. Now we have to show that repeating them makes the two copies of t different. But we can't.

$L = \{tt : t \in \{a, b\}^*\}$

But let's consider $L' = L \cap a^*b^*a^*b^*$

This time, we let $|w| > 2M$, and the number of both a's and b's in w >M:



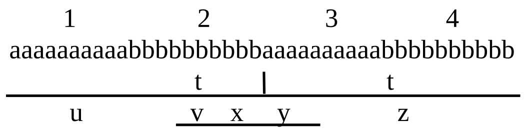
Now we use pumping to show that L' is not context free.

First, notice that if either v or y contains both a's and b's, then we immediately violate the rules for L' when we pump.

So now we know that v and y must each fall completely in one of the four marked regions.

$L' = \{tt : t \in \{a, b\}^*\} \cap a^*b^*a^*b^*$

$|w| > 2M$, and the number of both a's and b's in w >M:



Consider the combinations of (v, y):

(1,1)

(2,2)

(3,3)

(4,4)

(1,2)

(2,3)

(3,4)

(1,3)

(2,4)

(1,4)

The Context-Free Languages Are Not Closed Under Intersection

Proof: (by counterexample)

Consider $L = \{a^nb^nc^n: n \geq 0\}$

L is not context-free.

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

Both L_1 and L_2 are context-free.

But $L = L_1 \cap L_2$.

So, if the context-free languages were closed under intersection, L would have to be context-free. But it isn't.

The Context-Free Languages Are Not Closed Under Complementation

Proof: (by contradiction)

By definition:

$$L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$$

Since the context-free languages are closed under union, if they were also closed under complementation, they would necessarily be closed under intersection. But we just showed that they are not. Thus they are not closed under complementation.

The Deterministic Context-Free Languages Are Closed Under Complement

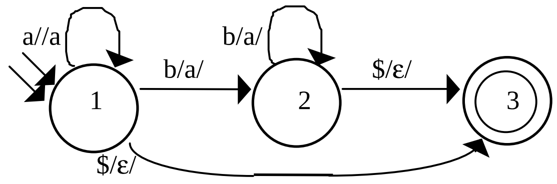
Proof:

Let L be a language such that L$ is accepted by the deterministic PDA M. We construct a deterministic PDA M' to accept (the complement of L)$, just as we did for FSMs:

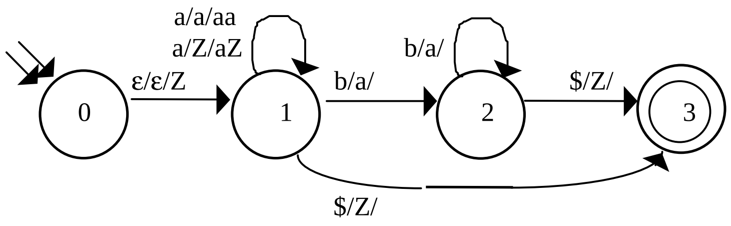
1. Initially, let M' = M.
2. M' is already deterministic.
3. Make M' simple. Why?
4. Complete M' by adding a dead state, if necessary, and adding all required transitions into it, including:
   - Transitions that are required to assure that for all input, stack combinations some transition can be followed.
   - If some state q has a transition on (ε, ε) and if it does not later lead to a state that does consume something then make a transiton on (ε, ε) to the dead state.
5. Swap final and nonfinal states.
6. Notice that M′ is still deterministic.

An Example of the Construction

$L = a^nb^n$ M accepts L$ (and is deterministic):

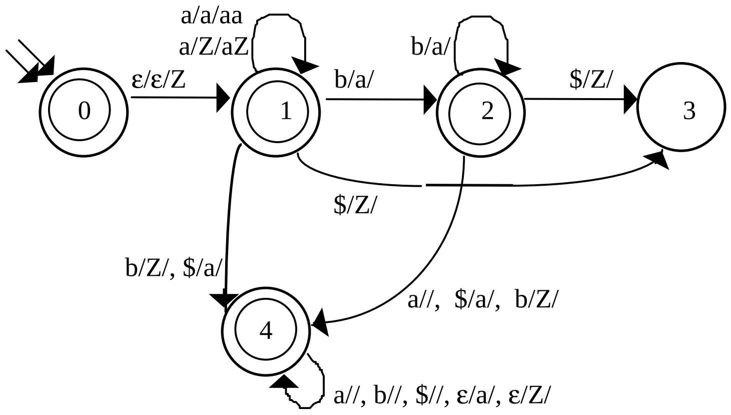


Set M = M'. Make M simple.



The Construction, Continued

Add dead state(s) and swap final and nonfinal states:



Issues: 1) Never having the machine die
2) $\neg$(L$) $\neq$ ($\neg$L)$
3) Keeping the machine deterministic

Deterministic vs. Nondeterministic Context-Free Languages

Theorem: The class of deterministic context-free languages is a ***proper*** subset of the class of context-free languages.

Proof: Consider $L = \{a^nb^mc^p : m \neq n \text{ or } m \neq p\}$ L is context free (we have shown a grammar for it).

But L is not deterministic. If it were, then its complement L_1 would be deterministic context free, and thus certainly context free. But then

$$L_2 = L_1 \cap a^*b^*c^* \text{ (a regular language)}$$

would be context free. But

$$L_2 = \{a^nb^nc^n : n \geq 0\}, \text{ which we know is not context free.}$$

Thus there exists at least one context-free language that is not deterministic context free.

Note that deterministic context-free languages are **not** closed under union, intersection, or difference.

Decision Procedures for CFLs & PDAs

Decision Procedures for CFLs

There are decision procedures for the following (G is a CFG):

- Deciding whether $w \in L(G)$.
- Deciding whether $L(G) = \varnothing$.
- Deciding whether L(G) is finite/infinite.

Such decision procedures usually involve conversions to Chomsky Normal Form or Greibach Normal Form. Why?

Theorem: For any context free grammar G, there exists a number n such that:

1. If $L(G) \neq \varnothing$, then there exists a $w \in L(G)$ such that $|w| < n$.
2. If L(G) is infinite, then there exists $w \in L(G)$ such that $n \leq |w| < 2n$.

There are **not** decision procedures for the following:

- Deciding whether $L(G) = \Sigma^*$.
- Deciding whether $L(G_1) = L(G_2)$.

If we could decide these problems, we could decide the halting problem. (More later.)

Decision Procedures for PDA's

There are decision procedures for the following (M is a PDA):

- Deciding whether $w \in L(M)$.
- Deciding whether $L(M) = \varnothing$.
- Deciding whether L(M) is finite/infinite.

Convert M to its equivalent PDA and use the corresponding CFG decision procedure. Why avoid using PDA's directly?

There are **not** decision procedures for the following:

- Deciding whether $L(M) = \Sigma^*$.
- Deciding whether $L(M_1) = L(M_2)$.

If we could decide these problems, we could decide the halting problem. (More later.)

Comparing Regular and Context-Free Languages

Regular Languages

- regular exprs.
    - or
- regular grammars
- recognize
- = DFSAs
- recognize
- minimize FSAs

- closed under:
    - * concatenation
    - * union
    - * Kleene star
    - * complement
    - * intersection
- pumping lemma
- deterministic = nondeterministic

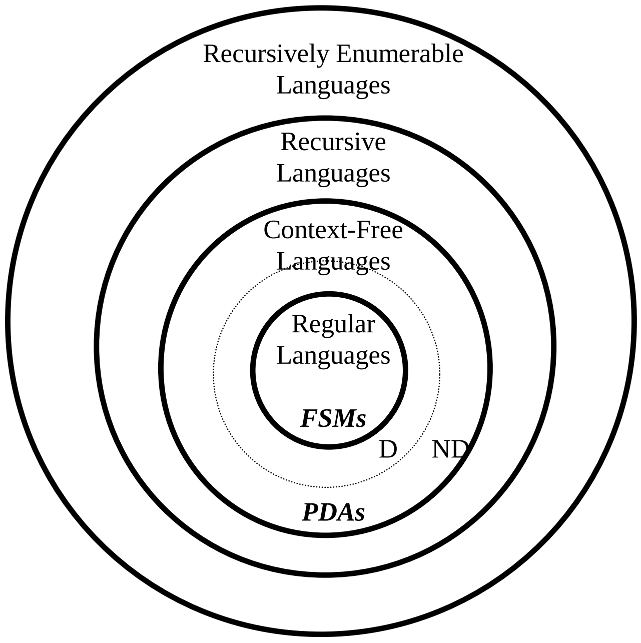
Context-Free Languages

- context-free grammars

- parse
- = NDPDAs
- parse
- find deterministic grammars
- find efficient parsers
- closed under:
    - * concatenation
    - * union
    - * Kleene star

- intersection w/ reg. langs
- pumping lemma
- deterministic ≠ nondeterministic

Languages and Machines


Recursively Enumerable Languages
Recursive Languages
Context-Free Languages
Regular Languages
FSMs
D
ND
PDAs