Kleene's Theorem: Let $A \subseteq \Sigma^\star$ be any language. Then the following are equivalent:

1. $A = \mathcal{L}(D)$, for some DFA D.
2. $A = \mathcal{L}(N)$, for some NFA N without ϵ transitions
3. $A = \mathcal{L}(N)$, for some NFA N.
4. $A = \mathcal{L}(e)$, for some regular expression e.
5. A is regular.

Myhill-Nerode Theorem: The language A is regular iff $\sim_A$ has a finite number of equivalence classes. Furthermore, this number of equivalence classes is equal to the number of states in the minimum-state DFA that accepts A.

Closure Theorem for Regular Sets: Let $A, B \subseteq \Sigma^\star$ be regular languages and let $h : \Sigma^\star \to \Gamma^\star$ and $g : \Gamma^\star \to \Sigma^\star$ be homomorphisms. Then the following languages are regular:

1. $A \cup B$
2. AB
3. $\overline{A} = (\Sigma^\star - A)$
4. $A \cap B$
5. $h(A)$
6. $g^{-1}(A)$

A *homomorphism* of strings is a function g such that for any strings u and v, $g(uv) = g(u)g(v)$. The set $g^{-1}(A)$ is defined as $\{u : g(u) \in A\}$.

Proofs of Closure Properties:

Because we have so many equivalent models for the class of regular languages, we can pick the one that makes each proof easiest:

1. Regular Expressions: union, concatenation, star
2. DFA: complement, hence intersection
3. (product of DFA's gives intersection directly)
4. Forward homomorphism: substitute into regexp or general NFA
5. Inverse homomorphism: simulate on DFA
6. Reversal: easy by regular expressions, but also doable with NFA's (exercise)

Let $h : \Sigma^* \to \Delta^*$ be a homomorphism.

If R is a regular expression over Σ, we can compute a regular expression $h(R)$ by induction on the definition of regular expressions. Then we can prove by induction that $h(\mathcal{L}(R)) = \mathcal{L}(h(R))$.

If M is a DFA with alphabet Δ and $\mathcal{L}(M) = A$, we can make a DFA for $h^{-1}(A)$ as follows. States, start, and final states are the same as M. For every letter a in Σ and every state q, define $\delta(q, a)$ to be $\delta^*_M(q, h(a))$.

Then for any $w \in \Sigma^*$, $\delta^*(q, w) = \delta^*_M(q, h(w))$, and

$$\begin{aligned} \delta^*(q_0, w) \in F \ &\leftrightarrow \\ \delta^*_M(q_0, h(w)) \in F \ &\leftrightarrow \\ h(w) \in A \ &\leftrightarrow \\ w \in h^{-1}(A) \end{aligned}$$

Definition: A **context-free grammar** (CFG) is a 4-tuple $G = (V, \Sigma, R, S)$,

- V = variables = nonterminals,
- Σ = terminals,
- R = rules = productions, $R \subseteq V \times (V \cup \Sigma)^\star$,
- $S \in V$,
- V, Σ, R are all finite.

$$
\begin{aligned}
G_1 &= (\{S\}, \{a,b\}, R_1, S) \\
R_1 &= \{\langle S, aSb\rangle, \langle S, \epsilon\rangle\} \quad = \quad \{S \rightarrow aSb|\epsilon\}
\end{aligned}
$$

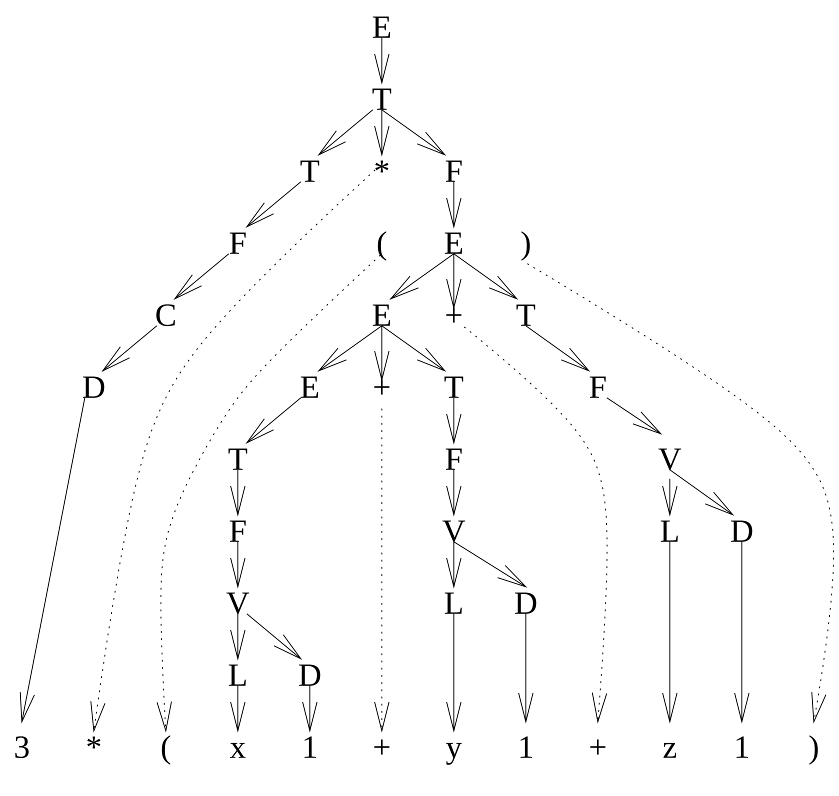
$$
\begin{aligned}
S &\rightarrow \epsilon \\
S &\rightarrow aSb \Rightarrow ab \\
S &\rightarrow aSb \Rightarrow aaSbb \Rightarrow aabb \\
S &\rightarrow aSb \Rightarrow aaSbb \Rightarrow aaaSbbb \Rightarrow aaabbb
\end{aligned}
$$

$$
\begin{aligned}
\mathcal{L}(G_1) &= \{w \in \{a,b\}^\star \mid S \underset{G_1}{\overset{\star}{\rightarrow}} w\} \\
&= \{a^n b^n \mid n \in \mathbf{N}\} \\
\mathcal{L}(G) &= \{w \in \Sigma^\star \mid S \underset{G}{\overset{\star}{\rightarrow}} w\}
\end{aligned}
$$

$$G_2 \; = \; (\{E,T,F,V,L,D,C\},\{(,),+,\star,x,y,z,0,1,\ldots,9\},R_2,E)$$

$$R_2 \quad = \quad \begin{array}{rcl} E & \rightarrow & E+T|T \\ T & \rightarrow & T\star F|F \\ F & \rightarrow & (E)|V|C \\ V & \rightarrow & LD \end{array} \qquad \begin{array}{rcl} L & \rightarrow & x|y|z \\ D & \rightarrow & 0|1|2|\cdots|9 \\ C & \rightarrow & D|CD \end{array}$$

Parse Tree:



Pumping Lemma for Regular Sets: Let $D = (Q, \Sigma, \delta, q_0, F)$ be a DFA. Let $n = |Q|$. Let $w \in \mathcal{L}(D)$ s.t. $|w| \geq n$. Then $\exists x, y, z \in \Sigma^\star$ s.t. the following all hold:

- $xyz = w$
- $|xy| \leq n$
- $|y| > 0$, and
- $(\forall k \geq 0) xy^k z \in \mathcal{L}(D)$

Proof: Let $w \in \mathcal{L}(D)$ s.t. $|w| \geq n$.

$$
\begin{array}{cccccccccccccc}
w & = & & & w_1 & & w_2 & & w_3 & & \cdots & & w_n & & u \\
& & & q_0 & & q_1 & & q_2 & & q_3 & \cdots & q_{n-1} & & q_n &
\end{array}
$$

By the Pigeonhole Principle, $(\exists i < j) q_i = q_j$

$$
\begin{array}{cccccccccc}
w & = & & \overbrace{w_1 \ldots w_i}^{x} & & \overbrace{w_{i+1} \ldots w_j}^{y} & & \overbrace{w_{j+1} \ldots w_n u}^{z} & \\
& & q_0 & & q_i & & q_i & & q_f
\end{array}
$$

$\delta^\star(q_i, y) = q_i$. Thus, $xy^k z \in \mathcal{L}(D)$ for all $k \in \mathbf{N}$. ♠

We showed: $E = \{a^r b^r \mid r \in \mathbf{N}\}$ is not regular.

Proof: Suppose that E were regular, accepted by a DFA with n states. Let $w = a^n b^n$.

By the pumping lemma, $w = a^n b^n = xyz$ where

- $|xy| \leq n$
- $|y| > 0$, and
- $(\forall k \in \mathbf{N}) xy^k z \in E$

Since $0 < |xy| \leq n, \quad y = a^i, 0 < i \leq n$.

Thus $xy^0 z = a^{n-i} b^n \in E$.

But, $a^{n-i} b^n \notin E$.

$\Rightarrow\Leftarrow$

Therefore E is not regular. $\spadesuit$

CFL Pumping Lemma: Let A be a CFL. Then there is a constant n, depending only on A, such that if $z \in A$ and $|z| \geq n$, then there exist strings u, v, w, x, y such that:

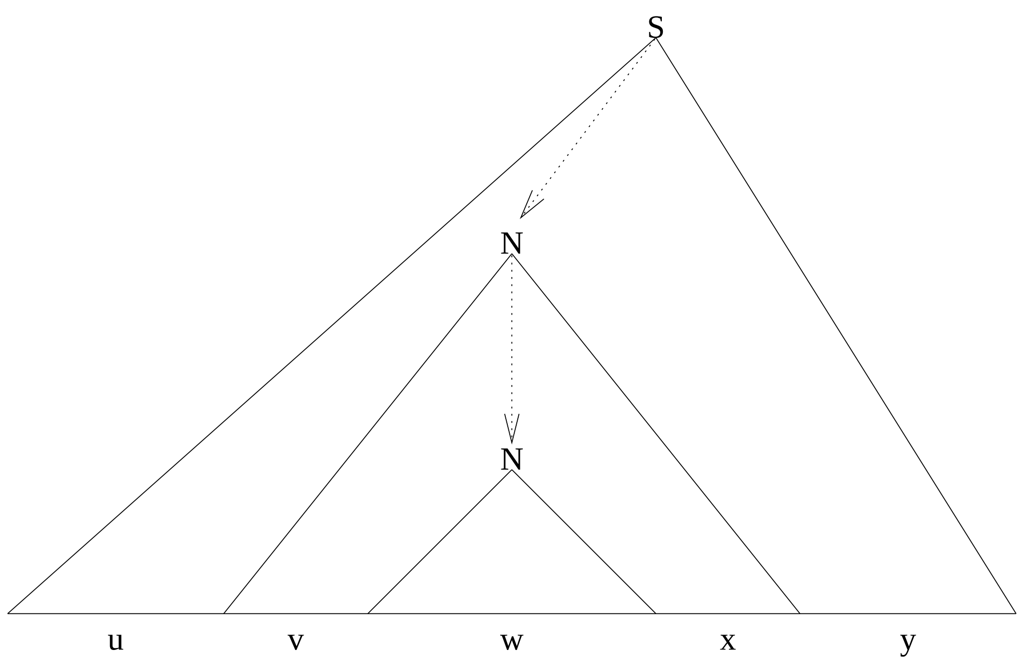
- $z = uvwxy$, and
- $|vx| \geq 1$, and
- $|vwx| \leq n$, and
- for all $k \in$ **N**, $uv^k wx^k y \in A$

Proof: Let $G = (V, \Sigma, R, S)$ be a CFG with $\mathcal{L}(G) = A$. Let n be so large that for $|z| \geq n$ s.t. $N \overset{\star}{\underset{G}{\rightarrow}} z$ for some $N \in V$, the parse tree for z has height $> |V| + 2$.

Let $z \in A$, $|z| \geq n$.

The parse tree for z has height greater than $|V| + 2$.

Thus, some path repeats a nonterminal, N.



$$z \;=\; uvwxy; \qquad (\forall k \in \mathbf{N})(uv^k wx^k y \in A)$$

Prop: $P = \{a^n b^m a^n b^m \mid n, m \in \mathbf{N}\}$ is not a CFL.

Proof: Suppose P were a CFL.

Let n be the constant of the CFL pumping lemma.

Let $z = a^n b^n a^n b^n$.

By the CFL pumping lemma, $z = uvwxy$, and

1. $|vx| \geq 1$,
2. $|vwx| \leq n$, and
3. for all $k \in \mathbf{N}$, $uv^k wx^k y \in P$

Since $|vwx| \leq n$, $\quad vwx \in a^\star b^\star$ or $vwx \in b^\star a^\star$.

If either v or x contains both a's, and b's, then uv^2wx^2y is not in P.

Suppose that vx contains at least one a. Then, uv^2wx^2y is not in P, because it has more a's in one group than the other.

Suppose that vx contains at least one b. Then, uv^2wx^2y is not in P, because it has more b's in one group than the other.

Thus, uv^2wx^2y is not in P.

$\Rightarrow\Leftarrow$ Thus P is not a CFL. ♠

Prop: $NONCFL = \{a^n b^n c^n : n \in \mathbf{N}\}$ is not a CFL.

Proof:

The argument is almost identical. We let $z = a^n b^n c^n$ where n is larger than the constant given by the CFL Pumping Lemma. So $z = uvwxy$ with $|vwx| > 0$, $|vwx| \leq n$, and $uv^i wx^i y$ in $NONCFL$ for all i. Again, neither v nor x can contain letters of two different types, or uv^2wx^2y is not in $a^*b^*c^*$. But then uv^2wx^2y cannot contain equal numbers of a's, b's, and c's, as only one or two types of letter have been added.

♠

Any CFL satisfies the conclusion of the CFL Pumping Lemma, but it is *not* true that any non-CFL must fail to satisfy it. There are other tools that can show a language to be a non-CFL. These include stronger forms of the Pumping Lemma and more *closure properties*.

Let $EQUAL$ be the set of strings in $(a \cup b \cup c)^*$ that have an equal number of a's, b's, and c's. You *can* use the CFL Pumping Lemma on this with the right choice of z, but far easier is using the fact that the intersection of $EQUAL$ with $a^*b^*c^*$ is the language $NONCFL$.

If A is a CFL and R a regular language, then $A \cup R$ must be regular. Proving this, however, requires a different characterization of the CFL's.

Definition: A **pushdown automaton** (PDA) is a 7-tuple, $P = (Q, \Sigma, \Gamma, \Delta, q_0, Z_0, F)$

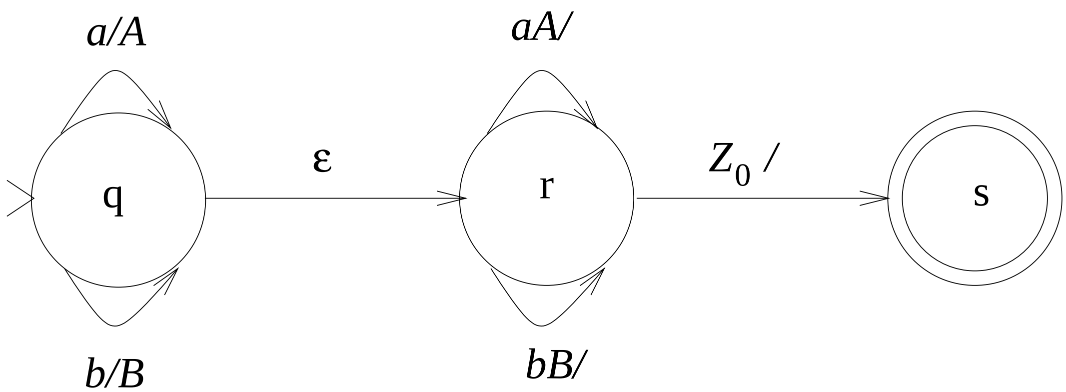
- $Q =$ finite set of states,
- $\Sigma =$ input alphabet,
- $\Gamma =$ stack alphabet,
- $\Delta \subseteq (Q \times \Sigma^\star \times \Gamma^\star) \times (Q \times \Gamma^\star)$ finite set of transitions,
- $q_0 \in Q$ start state,
- $Z_0 \in \Gamma$ initial stack symbol,
- $F \subseteq Q$ final states.

$$\text{PDA} \quad = \quad \text{NFA} + \text{stack}$$

$$\mathcal{L}(P) \quad = \quad \{w \in \Sigma^\star \mid (q_0, Z_0) \xrightarrow[P]{w} (q, X),\ q \in F, X \in \Gamma^\star\}$$

$$P_1 \;=\; (\{q, r, s\}, \{a, b\}, \{A, B, Z_0\}, \Delta_1, q, Z_0, \{s\})$$

$$\begin{aligned}\Delta_1 \;=\; & \{\langle(q, a, \epsilon), (q, A)\rangle, \langle(q, b, \epsilon), (q, B)\rangle, \langle(q, \epsilon, \epsilon), (r, \epsilon)\rangle \\ & \langle(r, a, A), (r, \epsilon)\rangle, \langle(r, b, B), (r, \epsilon)\rangle, \langle(r, \epsilon, Z_0), (s, \epsilon)\rangle\}\end{aligned}$$



$$\mathcal{L}(P_1) \;=\; \{ww^R \mid w \in \{a, b\}^\star\}$$

Theorem 4.1 *Let $A \subseteq \Sigma^\star$ be any language. Then the following are equivalent:*

1. *$A = \mathcal{L}(G)$, for some CFG G.*
2. *$A = \mathcal{L}(P)$, for some PDA P.*
3. *A is a context-free language.*

Proof: We give only a sketch here – there are detailed proofs in [HMU], [LP], and [S].

To prove (1) implies (2), we can build a "bottom-up parser" or "top-down" parser, similar to those used in real-world compilers except that the latter are *deterministic*.

The top-down parser is a PDA that:

- begins by pushing "$S\$$" onto its stack
- may pop a terminal from the stack if can at the same time read a matching input letter,
- may execute a rule $A \to w$ by popping A and pushing w^R,
- ends by popping the $\$$ when done with the input

The bottom-up parser, somewhat similarly:

- pushes "$" onto its stack,
- may transfer a terminal from the input to the stack,
- may execute $A \to w$ by popping w and pushing A,
- ends by popping $S\$$ when done

The proof that the language of any PDA is a CFL (that (2) implies (1)) is of less practical interest.

Given states i and j, let A_{ij} be the set of strings that *could* take the PDA from state i with empty stack to state j with empty stack.

If we can define rules making each A_{ij} a CFL we win, because the language of the PDA is the union of A_{sf} for all final states f, where s is the start state. (So our grammar has a rule $S \to A_{sf}$ for each f.)

We have all rules of the form $A_{pq} \to A_{pr}A_{rq}$, and a rule $A_{pq} \to aA_{rs}b$ whenever moves of the PDA warrant it.

Here I am skipping some assumptions on the PDA, and the (nontrivial) proof that any accepting run of the PDA corresponds to a valid derivation in our grammar. ♠