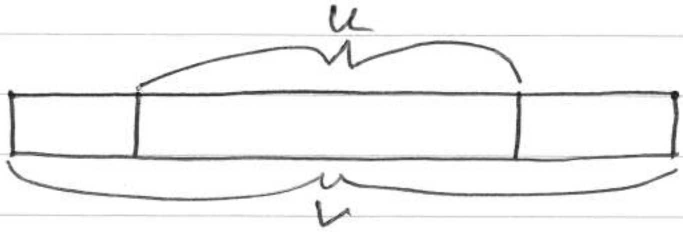
Definition 4: Example



$u \sqsubseteq v$

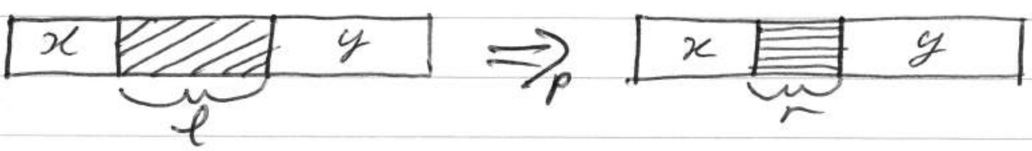
If $v = abc$, then all the substrings are:

$\epsilon, a, b, c, ab, bc, abc.$

Definition 5: Example

$(abc, aaaaa)$ is a rule over $\{a, b, c\}$, and we often write it as $abc \to aaaaa$.

Definition 6: Example



If $p = ab \to c$, and $w = abaab$, then there are exactly two ways to apply p:

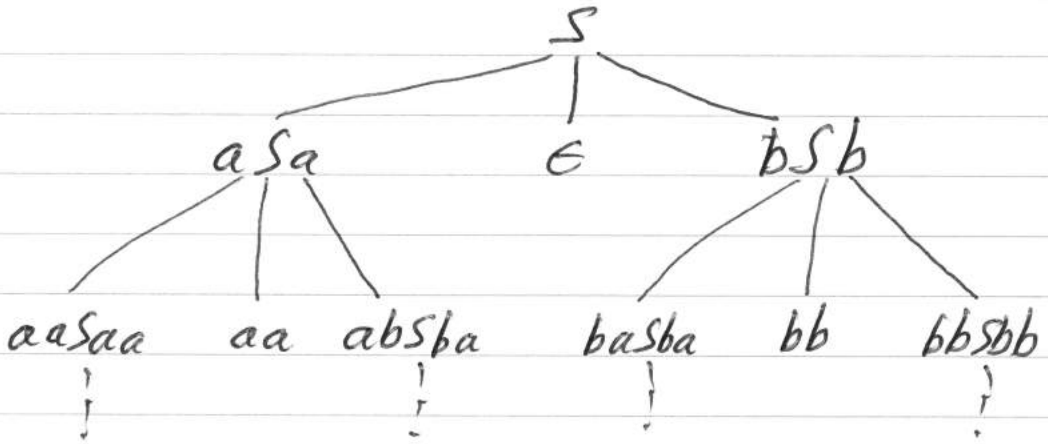
$w \Rightarrow_p caab, \qquad w \Rightarrow_p abac.$

Example 11: Extended

Consider the grammar:

$$G = (\{a, b\}, \{S\}, \{S \to aSa, S \to bSb, S \to \epsilon\}, S).$$

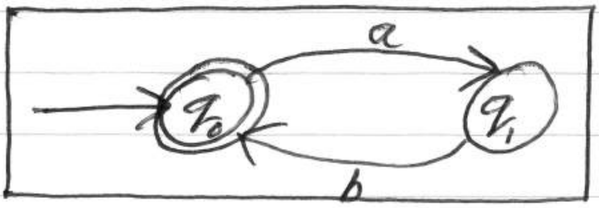
What can be derive from S?



$L(G)$ = the palindromes over $\{a, b\}$.

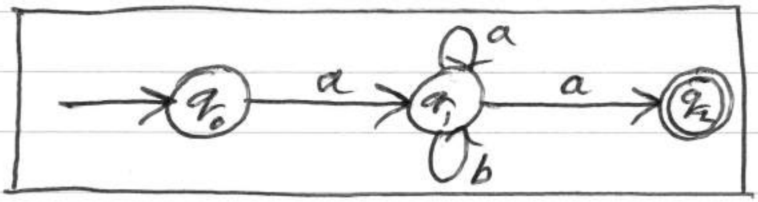
Non-terminals really do add generational power, since $L(G)$ can't be generated by a non-terminal free ($N = \emptyset$) grammar.

Definition 20–22: Example



deterministic

$L = \{(ab)^n \mid n \in \mathbb{N}\}$

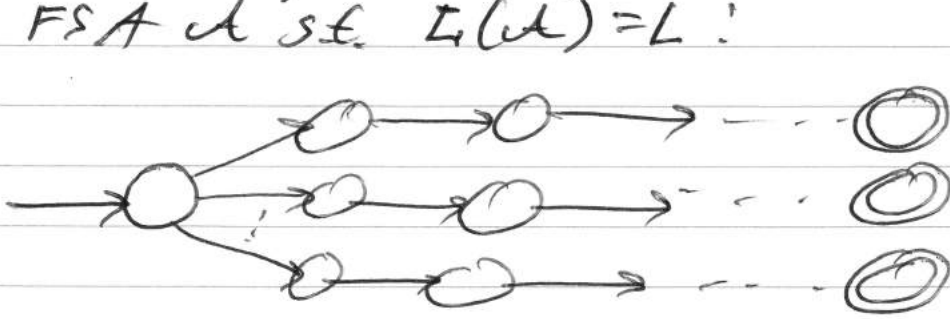


non-deterministic

$L = a\{a,b\}^* a$ ← all strings ending and starting with an a.

Proposition 24: Example

If L is a finite language, we can build an FSA $\mathcal{A}$ s.t. $L(\mathcal{A}) = L$:



Given $\mathcal{A}_1, \mathcal{A}_2$, we can build $\mathcal{A}$ s.t. $L(\mathcal{A}) = L(\mathcal{A}_1) \cup L(\mathcal{A}_2)$:

Proposition 27: Example

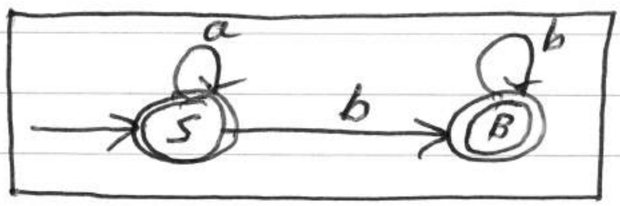
Consider the grammar from Example 26:

$$G = (\{a, b\}, \{S, B\}, \mathcal{R}, S)$$

where the rules in $\mathcal{R}$ are:

$$S \longrightarrow aS$$
$$S \longrightarrow bB$$
$$S \longrightarrow \epsilon$$
$$B \longrightarrow bB$$
$$B \longrightarrow \epsilon.$$

Then we have:



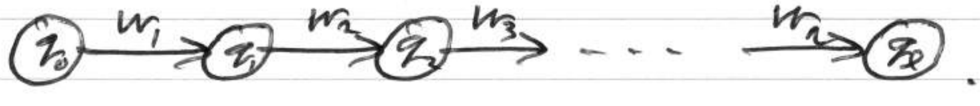
Theorem 28: Proof

Let $L \subseteq \Sigma_1^*$ be a regular language. Then there exists some $n \in \mathbb{N}^+$ such that for every $w \in L$ with length at least n, we can decompose w into three strings ($w = xyz$) such that:

(1) y has length at least 1;
(2) xy has length at most n;
(3) for all $k \in \mathbb{N}$, $xy^kz \in L$.

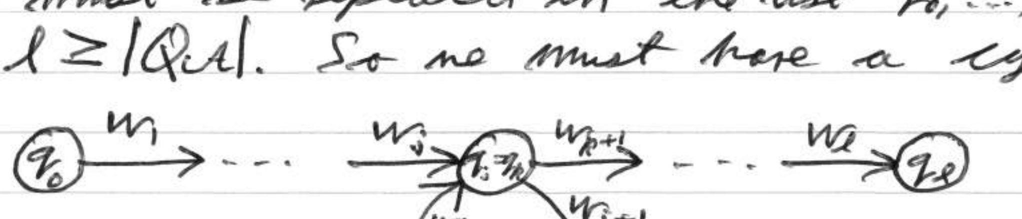
Proof: By Theorem 16, there must be a DFSA, $\mathcal{A}$, s.t. $L_1(\mathcal{A}) = L$. Set $n := |Q_{\mathcal{A}}|$ and choose some $w = w_1 \cdots w_l \in L$ s.t. $l \geq n$.

There must be a transition sequence from an initial state q_0 to an accepting state q_l: ~~such that~~



.

But then by the pigeonhole principle, some state must be repeated in the list $q_0, \ldots, q_l$ since $l \geq |Q_{\mathcal{A}}|$. So we must have a cycle:



Which means we can "pump" around this cycle to accept $w_1 w_2 \cdots w_j (w_{j+1} \cdots w_k)^i w_{k+1} \cdots w_l$ for all $i \in \mathbb{N}$. $\square$

Example 29: Proof

The language $L = \{a^n b^n \mid n \in \mathbb{N}\} \subseteq \{a,b\}^*$ is not regular.

Proof: By the pumping lemma.

Let m be the constant that exists by the pumping lemma and choose $w = a^m b^m$. Then we must be able to decomose w into xyz where $x = a^\alpha$, $y = a^\beta$, $z = a^\gamma b^m$ where $\beta \geq 1$ and $\alpha + \beta + \gamma = m$.

But then by the pumping lemma xz must be in L too, but $xz = a^{\alpha+\gamma} b^m$, and we know that $\alpha + \gamma \neq m$ ... a contradiction.

So, L is not a regular language. $\square$

Theorem 37: Proof

A group has a presentation with a regular word problem iff it is finite.

Proof: This was first shown by Anisimov (1971), but we follow the simpler proof of Muller and Schupp (1983).

$\Leftarrow$ Let $G = \{g_1, \ldots, g_n\}$ be a finite group with g_1 the identity. Then the multiplication table gives us the obvious presentation:

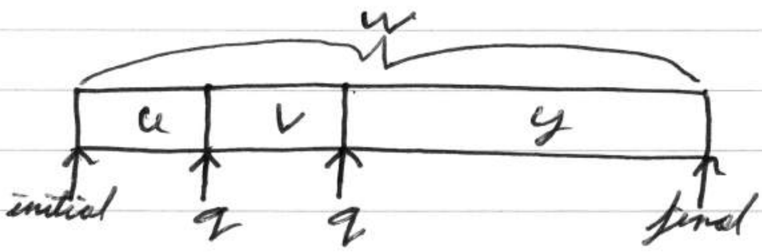
$$\langle g_1, \ldots, g_n \mid \ldots, g_i g_j = g_k, \ldots \rangle.$$

Define the FSA $\mathcal{A} = (G, G, \delta, g_1, \{g_1\})$ where $(g_i, g_j, g_k) \in \delta \iff g_i g_j = g_k$. Clearly $\mathcal{A}$ finishes in state g_1 iff the input string is equal to g_1 in the group.

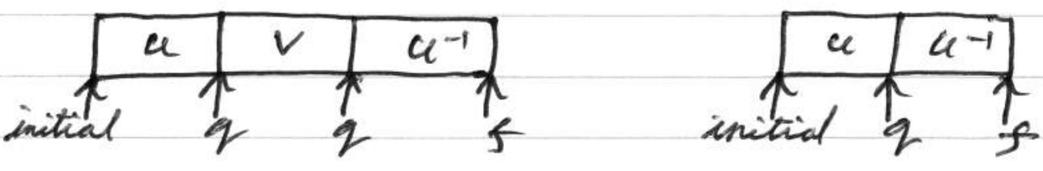
$\Rightarrow$ First, consider the finite presentation $\langle X \mid R \rangle$ of an infinite group. There must be arbitrarily long strings over X such that no non-empty substring is equal to the identity in the group. For if there were an upper bound B on the length of such ~~words~~ strings, every element of the group could be represented by a string of length not exceeding B, since X is finite. But then we have that the group is only finite: a contradiction.

Now, let $\mathcal{A}$ be an DFSA with input alphabet X, and let $w \in X^*$ be a string whose length is greater than the number of states in $\mathcal{A}$ and such that no non-empty substring of w is equal to the identity in the group.

If $\mathcal{A}$ begins reading w, then it must be in the same state, say q, after reading two distinct initial segments, due to the pigeonhole principle, say u and uv of w:



Now, uu^{-1} is certainly equal to the identity in the group, and uvu^{-1} is not, since it is a conjugate of v which is not equal to the identity in the group.



But $\mathcal{A}$ finishes in the same state, f, after reading uvu^{-1} or uu^{-1}, but only one of them is in the word problem. So no DFSA can solve the word problem for a presentation of an infinite group. $\square$

Definition 38: Examples

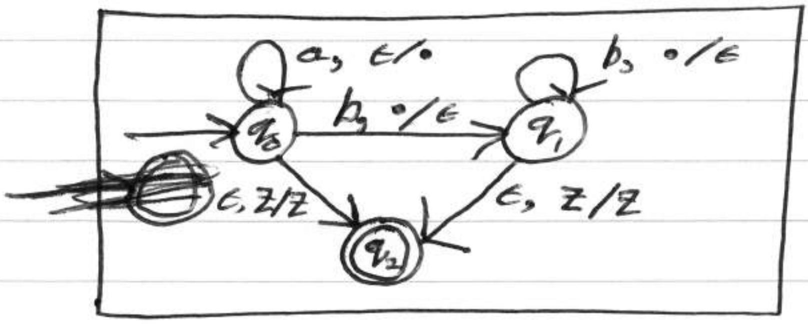
Let $\Sigma_1 = \{a, b\}$. In Example 11, we saw a grammar for the palindromes over Σ_1. It was context-free:

$$S \to aSa$$
$$S \to bSb$$
$$S \to \epsilon.$$

In Example 29, we saw that $\{a^n b^n \mid n \in \mathbb{N}\}$ was not regular. It is context-free:

$$S \to aSb$$
$$S \to \epsilon.$$

A pushdown automaton for this language is:



where the stack initially contains only the symbol Z. An example "run":

$$\begin{aligned}(aabb, q_0, Z) &\vdash (abb, q_0, Z\bullet)\\ &\vdash (bb, q_0, Z\bullet\bullet)\\ &\vdash (b, q_1, Z\bullet)\\ &\vdash (\epsilon, q_1, Z)\\ &\vdash (\epsilon, q_2, Z)\end{aligned}$$

✓

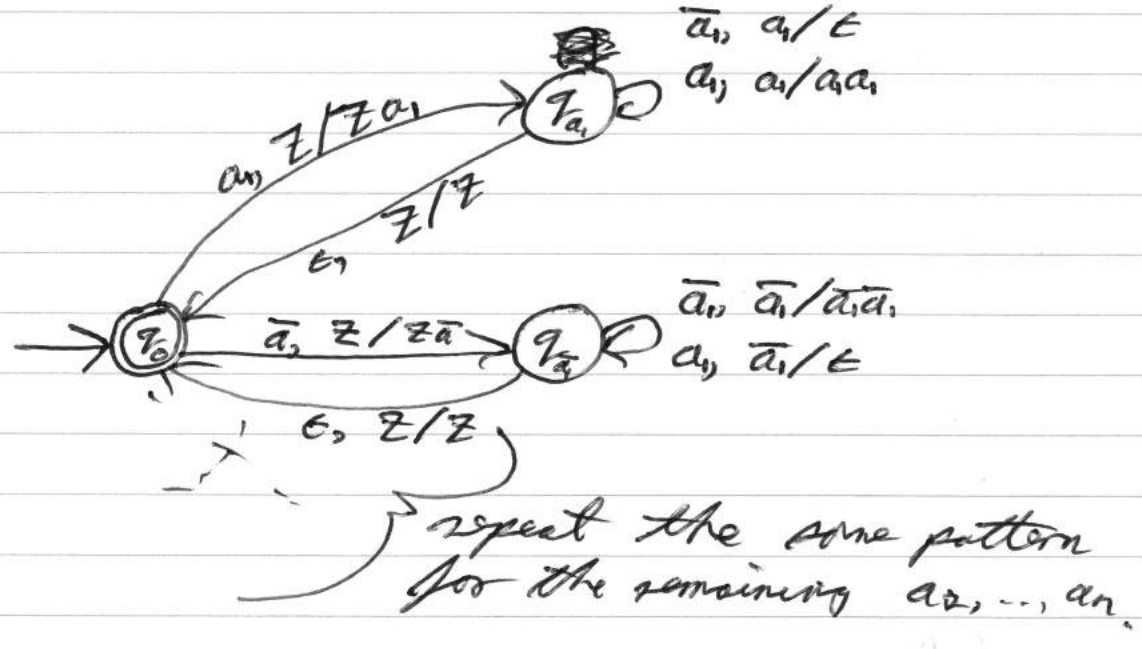
Example 41: Extended

Let $X_n = \{a_i, \bar{a}_i \mid i \in \{1,\dots,n\}\}$ and $\mathcal{R}_n = \{a_i\bar{a}_i \to \epsilon,\ \bar{a}_i a_i \to \epsilon \mid i \in \{1,\dots,n\}\}$. Then:

$$X_n^* / \mathcal{R}_n \cong F_n,$$

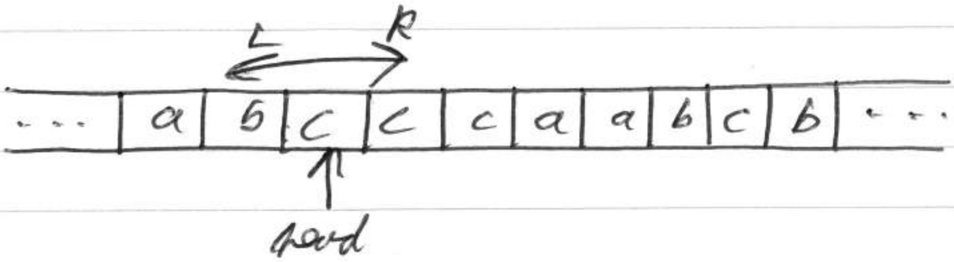
for all $n \geq 1$.

Then the following pushdown automaton recognises the word problem of F_n, for some fixed $n \geq 1$:



where Z is the special bottom of the stack symbol.

Definition 43: Picture



Theorem 44: Proof

Let SA be the language of encoded TMs that accept themselves. Then SA is r.e. but not recursive.

Proof: If SA were recursive, then $\overline{SA}$ must be too, but $\overline{SA}$ is not even r.e. To see this suppose $\mathcal{A}$ is a TM s.t. $L(\mathcal{A}) = \overline{SA}$, and consider the code $e(\mathcal{A})$. Either $e(\mathcal{A})$ is in $L(\mathcal{A})$ or not. But if $e(\mathcal{A}) \in L(\mathcal{A})$, then it accepts itself ※ and if $e(\mathcal{A}) \notin L(\mathcal{A})$ then it doesn't accept itself ※! So $\overline{SA}$ must not be r.e.

To see that SA is r.e. can be done by construction. Construct $\mathcal{A}$ s.t. first it checks if the input w is a valid encoding, and if it is execute the decoded machine on its encoding. Clearly $L(\mathcal{A}) = SA$. □