VII

AUTOMATA & FORMAL LANGUAGES
SEVENTH LECTURE

20 October 2022

RECAP

The following are equivalent:

(i) $L = \mathcal{L}(G)$ for some regular grammar G

(ii) $L = \mathcal{L}(D)$ for some deterministic automaton D

(iii) $L = \mathcal{L}(N)$ for some nondeterministic automaton N

(ii) $\Rightarrow$ (i) direct.
(iii) $\Rightarrow$ (ii) subset construction
(i) $\Rightarrow$ (iii) direct

TODAY: Finally, a method to prove that a language is <u>not</u> regular:

THE PUMPING LEMMA

(6) Let $G = (\Sigma, V, P, S)$ be any grammar. As in the lectures, a production rule $\alpha \to \beta$ is called variable-based if $\alpha \in V^*$. Suppose that $\alpha \to \beta$ is a noncontracting variable-based rule, say with $\alpha = A_1 \dots A_n$ and $\beta = B_1 \dots B_m$ for $A_i \in V$, $B_i \in W$, and $n \leq m$. Let $X_1, \dots, X_n$ be n new variables that do not occur in V and consider the following list of $2n$ rules:

$$
\begin{aligned}
A_1A_2A_3 \dots A_{n-2}A_{n-1}A_n &\to X_1A_2A_3 \dots A_{n-2}A_{n-1}A_n\\
X_1A_2A_3 \dots A_{n-2}A_{n-1}A_n &\to X_1X_2A_3 \dots A_{n-2}A_{n-1}A_n\\
X_1X_2A_3 \dots A_{n-2}A_{n-1}A_n &\to X_1X_2X_3 \dots A_{n-2}A_{n-1}A_n\\
&\vdots\\
X_1X_2X_3 \dots X_{n-2}A_{n-1}A_n &\to X_1X_2X_3 \dots X_{n-2}X_{n-1}A_n\\
X_1X_2X_3 \dots X_{n-2}X_{n-1}A_n &\to X_1X_2X_3 \dots X_{n-2}X_{n-1}X_nB_{n+1} \dots B_m\\
X_1X_2X_3 \dots X_{n-2}X_{n-1}X_nB_{n+1} \dots B_m &\to B_1X_2X_3 \dots X_{n-2}X_{n-1}X_nB_{n+1} \dots B_m\\
B_1X_2X_3 \dots X_{n-2}X_{n-1}X_nB_{n+1} \dots B_m &\to B_1B_2X_3 \dots X_{n-2}X_{n-1}X_nB_{n+1} \dots B_m\\
&\vdots\\
B_1B_2B_3 \dots B_{n-2}X_{n-1}X_nB_{n+1} \dots B_m &\to B_1B_2B_3 \dots B_{n-2}B_{n-1}X_nB_{n+1} \dots B_m\\
B_1B_2B_3 \dots B_{n-2}B_{n-1}X_nB_{n+1} \dots B_m &\to B_1B_2B_3 \dots B_{n-2}B_{n-1}B_nB_{n+1} \dots B_m
\end{aligned}
$$

Show that each of these rules is context-sensitive and that replacing $\alpha \to \beta$ in P by this collection of $2n$ rules does not change the language produced by G. Use this to prove that a language is noncontracting if and only if it is context-sensitive.

IMPORTANT NOTE ON EXAMPLE SHEET #1.
The statement of example (6) has been updated.
The MOODLE has the updated version; the DPMMS page currently has the old version!

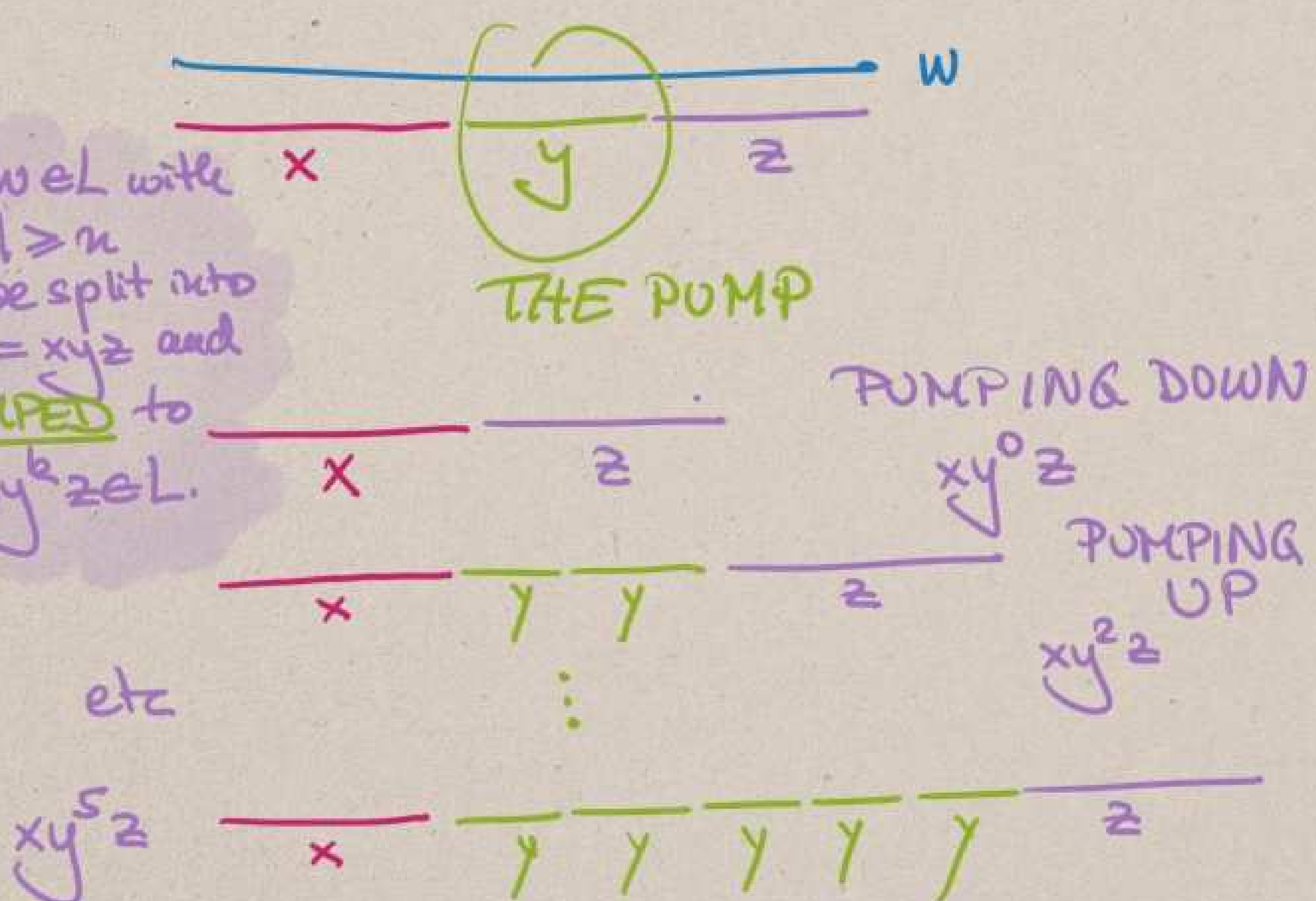
§ 2.4 The pumping lemma for regular languages

Definition 2.10. Let $L \subseteq \mathbb{W}$ be a language. We say that L *satisfies the (regular) pumping lemma with pumping number* n if for every word $w \in L$ such that $|w| \geq n$ there are words x, y, z such that $w = xyz$, $|y| > 0$, $|xy| \leq n$ and for all $k \in \mathbb{N}$, we have that $xy^kz \in L$. We say that L *satisfies the (regular) pumping lemma* if there is some n such that it satisfies the (regular) pumping lemma with pumping number n.

If a language L satisfies the pumping lemma and we have written $w = xyz$ as in the definition, then $xz = xy^0z$, xy^2z, xy^3z, etc. are all in L. We call the transition from $w = xyz$ to xz *pumping down* and the transition to xy^kz (for $k > 1$) *pumping up*.

Theorem 2.11 (The regular pumping lemma). For every regular language L, there is a number n such that L satisfies the regular pumping lemma with pumping number n.

All $w \in L$ with $|w| \geq n$ can be split into $w = xyz$ and PUMPED to $xy^kz \in L$.



Comment on bounds in the statement of PL:

Pumping # n means:

Every word w: if $|w| \geq n$, then it splits into $w = xyz$ with $|xy| \leq n$ & $|y| > 0$

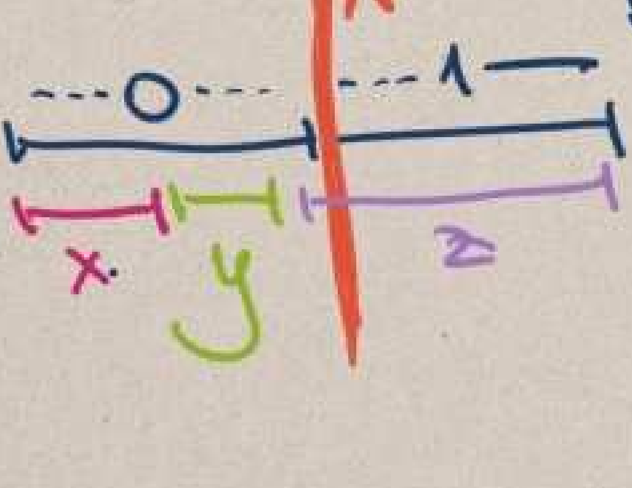
<u>Theorem 2.11</u> Every regular languages sat. the reg. PL.

<u>Observation</u> If I can ever pump something in a language, the language must be infinite.

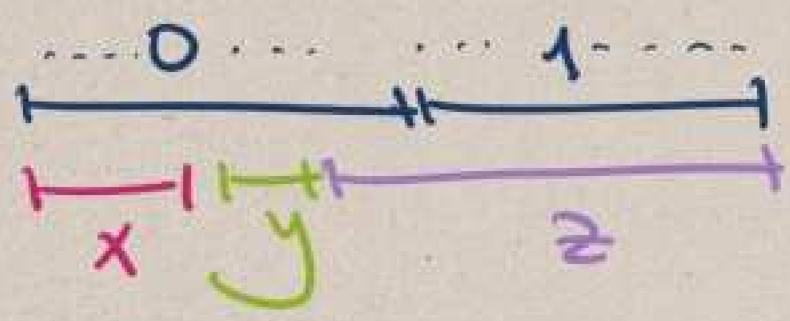
<u>Application 1</u> $L = \{0^k 1^k ; k > 0\}$ is not regular.

Proof using T2.11. Suppose it was, so it has a pumping #, say N.

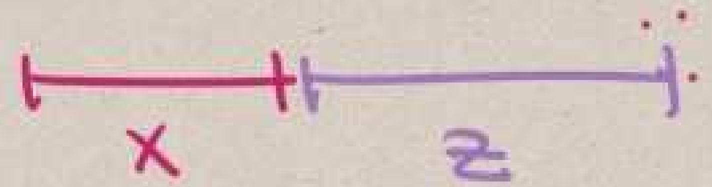
Pick $0^N 1^N \in L$. $|0^N 1^N| = 2N \geq N$, so it can be pumped.



The bound $|xy| \leq N$ means that the pump lies entirely in the first half.
So $y = 0^l$ some $l > 0$.



Pumping down :



$0^{N-\ell}1^{N}$ with $\ell > 0$

$\notin L$

Contradiction, so L is not regular!

Proof of the PL Let L be regular; by lecture VI, we know that $L = \mathcal{L}(D)$ for det. automaton $D = (\Sigma, Q, \delta, q_0, F)$

Define $n := |D|$ and claim n is the pumping number for L.

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

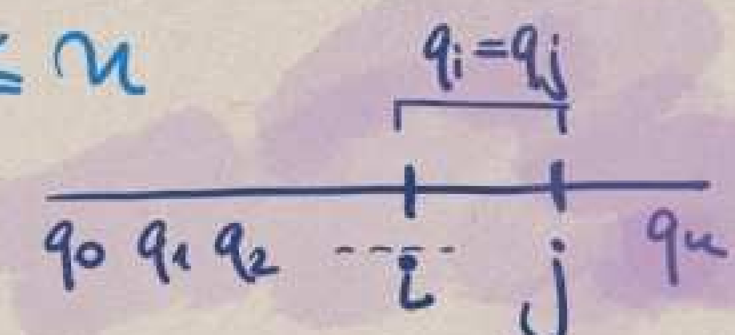
Write

$$w = a_0 a_1 \dots a_{n-1} v$$

where $v \in \mathbb{W}$.

The state seq. of D reading $a_0 \dots a_{n-1}$ is a sequence $(q_0, \dots, q_n)$ of length $n+1$.

So by pigeon hole, one of them must repeat, so there are $i < j \leq n$ s.t. $q_i = q_j$.



$x := a_0 \dots a_{i-1}$

$y := a_i \dots a_{j-1}$

$z := a_j \dots a_{n-1} v$

By construction

$w = xyz$

$|y| > 0$

$|xy| \leq n$.

Analyse the action of D:

$$\hat{\delta}(q_0, x) = q_i$$

$$\hat{\delta}(q_i, y) = q_j = q_i$$

$$\hat{\delta}(q_i, z) = \hat{\delta}(q_j, z) \in F$$

because $q_i = q_j$

Putting these together gives that

$$\hat{\delta}(q_0, xy^k z) \in F.$$

q.e.d.

Application 2

Fix $n > 0$.

$$L = \{ 0^n w ; w \in \mathbb{W} \}$$

is regular, but cannot have a det. aut. D s.t. $L = \mathcal{L}(D)$ and D has at most n states.

[For "regular" just write down a grammar:

$$\begin{aligned}
S &\longrightarrow 0X_0 \\
X_0 &\longrightarrow 0X_1 \\
X_1 &\longrightarrow 0X_2 \\
&\ \vdots \\
X_{n-2} &\longrightarrow 0X_{n-1} \\
X_{n-2} &\longrightarrow 0 \\
X_{n-1} &\longrightarrow 1 \\
X_{n-1} &\longrightarrow 0 \\
X_{n-1} &\longrightarrow 1X_{n-1} \\
X_{n-1} &\longrightarrow 0X_{n-1}
\end{aligned}$$

This grammar has $n+1$ variables!

Proof that no small automaton can do it:

If $\mathcal{L}(D) = L$ and D has $\leq n$ states, then L satisfies PL with Pumping # n.

So we can pump 0^n $[\,|0^n| = n\,]$ down and obtain a word with fewer zeros.]

?? Is the PL equivalent to "regular"?

[i.e., is it true that L is regular iff L satisfies the PL?]

ANSWER: NO!

$\Sigma = \{0,1\}$

If $w \in \mathbb{W}$ contains at least one zero, we say $\text{tail}(w)$ is the number of ones following the last zero

Ex. $\text{tail}(010\dot{1}111) = 4$.

Take $X \subseteq \mathbb{N}$ arbitrary and define

$L_X = \{w\,;$ either w contains no zero or it does & $\text{tail}(w) \in X\}$

If $X \neq Y$, then $L_X \neq L_Y$.

So, there are uncountably many languages of the type L_X.

CLAIM Each L_X satisfies PL.

Thus: some of them are non-regular languages satisfying PL.

[Since there are only ctbly many regular languages.]

Proof of Claim L_X has pumping number 2

$w \in L_X \quad |w| \geq 2$

Case 1. $w = 0z$ Let $x = \varepsilon$ $y = 0$
$[z \neq \varepsilon]$ Then $w = xyz$

$tail(w) = tail(z)$,
so if I pump up, $tail(0^k z)$
$= tail(z) = tail(w)$

$0^k z \in L_X$

if I pump down:
if z contains a zero, then
$tail(w) = tail(z)$
$xy^0z = \varepsilon z = z \in L_X$
if z contains no zero, $z \in L_X$ anyway.

Case 2 $w = 1z$ Let $x = \varepsilon$
$y = 1$
Then $w = xyz$.

if z contains no zeros, then $1^k z \in L_X$
if z contains zeros, then
$tail(1^k z) = tail(z) = tail(1z) = tail(w)$
$\Longrightarrow 1^k z \in L_X$. q.e.d.

§2.5 Closure properties

We already saw that regular languages are closed under concatenation & union.

They are closed under complementation, intersection & difference as well. For this, it's enough to show closure under complementation. [Intersection and difference can be expressed in terms of union & complementation.]

$$D = (\Sigma, Q, \delta, q_0, F)$$

Idea: Flip "accept" and "reject" states.

$$\overline{D} = (\Sigma, Q, \delta, q_0, Q \setminus (F \cup \{q_0\}))$$

Then $\mathcal{L}(\overline{D}) = W^+ \setminus \mathcal{L}(D)$.

Needed to guarantee that ε is not accepted.

Alternative construction to get union & intersection.

PRODUCT AUTOMATON

$D = (\Sigma, Q, \delta, q_0, F)$

$D' = (\Sigma, Q', \delta', q_0', F')$

Pointwise product:

$$\delta \times \delta'((q, q'), a) := \left(\delta(q, a), \delta'(q', a)\right)$$

$$F \wedge F' := \{(q, q');\ q \in F \ \& \ q' \in F'\}$$

$$F \vee F' := \{(q, q');\ q \in F \text{ or } q' \in F'\}$$

$$D \wedge D' := (\Sigma, Q \times Q', \delta \times \delta', (q_0, q_0'), F \wedge F')$$

$$D \vee D' := (\Sigma, Q \times Q', \delta \times \delta', (q_0, q_0'), F \vee F')$$

$$\mathcal{L}(D \wedge D') = \mathcal{L}(D) \cap \mathcal{L}(D')$$

$$\mathcal{L}(D \vee D') = \mathcal{L}(D) \cup \mathcal{L}(D')$$

Think of the state sequence of the product automaton as consisting of the two state sequences of D & D'.