VIII

Automata & Formal Languages

EIGHTH LECTURE

23 OCTOBER 2023

Recap from Lecture VII: the (regular) pumping lemma

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.

Theorem (Lecture VII, page 8)
Every regular language satisfies the (R)PL.

Applications

1. Prove that a language cannot be regular.

   EXAMPLE $\{0^n 1^n ; n > 0\}$

2. Prove that a regular language cannot have a small automaton.

   EXAMPLE $\{0^{100} w ; w \in \mathbb{W}\}$

Remark

The validity of (R)PL is **NOT** equivalent to being regular.

Let $\Sigma = \{0,1\}$ and $X \subseteq \mathbb{N}$.

$L_X := \{ w ;$ either $w = 1^k$ for some k or w contains a zero & the # of 1s after the last 0 is in $X \}$

Fact 1 if $X \neq X'$, then $L_X = L_{X'}$.

Fact 2 L_X satisfies the (R)PL for pumping # 2.

Prop. 2.15
Cor. 2.16
in the typed notes

THUS. There are languages that satisfy (R)PL, but are not regular.

[By Fact 1, there are uncountably many L_X, but only ctbly many reg. languages.]

→ EXAMPLE SHEET #2

CLOSURE PROPERTIES

Lecture V, page 4

CLOSURE PROPERTIES

Proposition on page 7

	Type 0	Type 1	Type 2	Type 3
Concatenation	✓	✓	✓	~~?~~ ✓
Union	✓	✓	✓	~~?~~ ✓
Intersection	?	?	?	~~?~~ ✓
Complement	?	?	?	~~?~~ ✓

Solving orange ? is the goal for this course!

We proved that the class of regular languages is closed under concatenation & union.

Will show: Reg. languages are closed under complement.

Therefore: Reg. languages <u>are</u> closed under intersection.

<u>THEOREM</u> If L is regular, then so is $\mathbb{W}^+ \setminus L$.

<u>Proof.</u> If $D = (\Sigma, Q, \delta, q_0, F)$ is any automaton and $L = \mathcal{L}(D)$,

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

Clearly, w is accepted by D

$$\Longleftrightarrow$$

w is rejected by $\overline{D}$

$$\mathcal{L}(\overline{D}) = \mathbb{W} \setminus \mathcal{L}(D).$$

Problem is that $\overline{D}$ is not quite an automaton: it doesn't satisfy the condition that q_0 is not an accept state.

First attempt at solution:

Define $\overline{D}^* := (\Sigma, Q, \delta, q_0, Q \setminus (F \cup \{q_0\}))$.

This does not accept ε anymore, but what if $\hat{\delta}(q_0, w) = q_0$.

Then $\overline{D}^*$ will also reject it, so $\mathcal{L}(\overline{D}^*) \neq \mathbb{W}^+ \setminus \mathcal{L}(D)$.

Clearly, if $\mathcal{D}$ has the property that for no

(*) $w \neq \varepsilon$, we have $\hat{\delta}(q_0, w) = q_0$,

then this problem doesn't occur.

If $\mathcal{D}$ has this property, then

$$\mathcal{L}(\mathcal{D}^*) = \mathbb{W}^+ \setminus \mathcal{L}(\mathcal{D}).$$

For (*), it's enough to have $q_0 \notin \mathrm{ran}(\delta)$.

<u>Claim</u> Any automaton $\mathcal{D}$ is equivalent [i.e., $\mathcal{L}(\mathcal{D}) = \mathcal{L}(\mathcal{D}')$] to an automaton $\mathcal{D}'$ with $q_0 \notin \mathrm{ran}(\delta)$.

[Let $q^+ \notin Q$ and replace all occurrences of q_0 in δ by q^+. Call this δ^+.

$$\mathcal{D}^+ := (\Sigma, Q \cup \{q^+\}, \delta^+, q_0, F)$$

$q^+ \notin F$

Clearly, $\mathcal{L}(\mathcal{D}^+) = \mathcal{L}(\mathcal{D})$.]

SUMMARY:

$$\mathcal{D} \rightsquigarrow \mathcal{D}^+ \rightsquigarrow \overline{\mathcal{D}^+} \rightsquigarrow \overline{\mathcal{D}^+}^{\,*}$$

$\mathcal{L}$ $\quad$ $\mathcal{L}$ $\quad$ $\mathbb{W} \setminus \mathcal{L}$ $\quad$ $\mathbb{W}^+ \setminus \mathcal{L}$.

q.e.d.

An alternative construction

PRODUCT AUTOMATA

Two automata $\mathcal{D} = (\Sigma, Q, \delta, q_0, F)$
$\mathcal{D}' = (\Sigma, Q', \delta', q_0', F')$

Product automata:

$$(\Sigma, Q \times Q', \delta \times \delta', (q_0, q_0'), G)$$

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

and G can be:

$$G = F \wedge F' := \{(q, q');\ q \in F \text{ and } q' \in F'\}$$

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

If $\mathcal{D} \wedge \mathcal{D}'$ is the automaton with $G = F \wedge F'$
$\mathcal{D} \vee \mathcal{D}'$ ——"—— $G = F \vee F'$

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

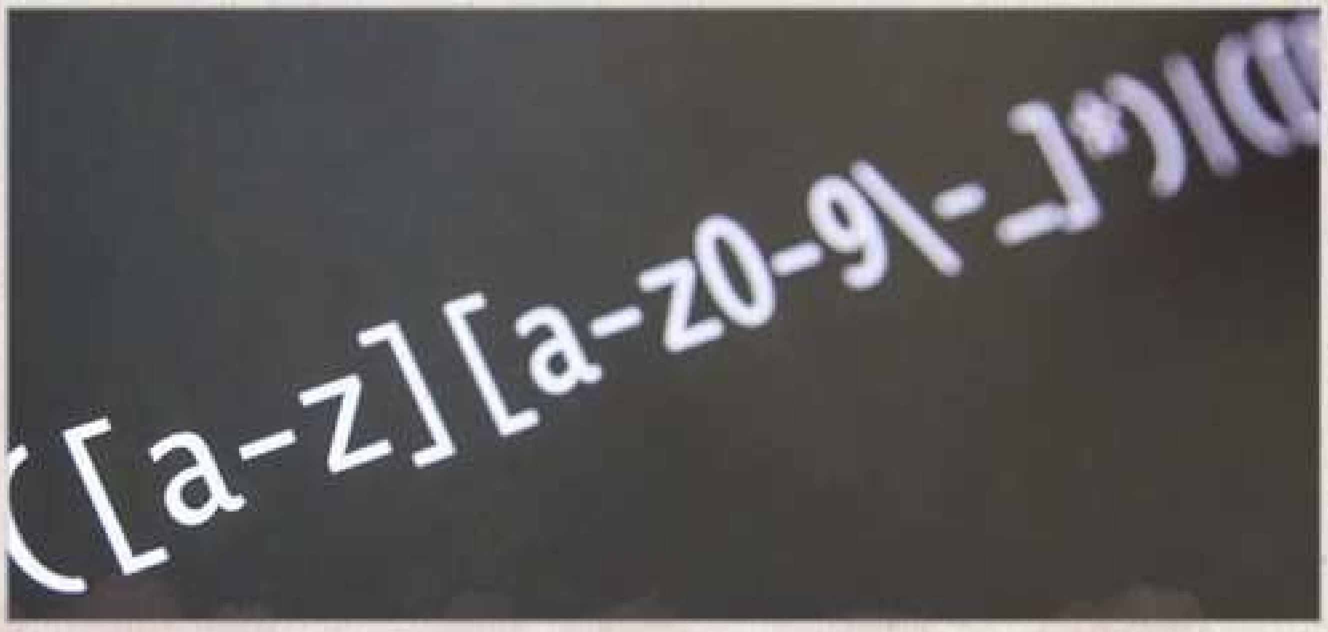
$$\mathcal{L}(\mathcal{D} \vee \mathcal{D}') = \mathcal{L}(\mathcal{D}) \cup \mathcal{L}(\mathcal{D}')$$

[Check the definitions: typed notes P 2.18]

⇝ See Example Sheet #1.

REGULAR EXPRESSIONS

Searching databases:



GOAL Find all lists in the database that belong to a given (regular) language.

NEED A concise way to enter the information describing that regular language.

SOLUTION REGULAR EXPRESSIONS

Fix alphabet Σ.

Symbols: Elements of Σ plus

Superscript + & *

Ω $\quad \emptyset \quad \varepsilon \quad (\quad) \quad + \quad {}^{+} \quad {}^{*}$

A <u>regular expression</u> is an element of Ω^* uniquely identifying a reg. language

Kleene star

$$L \subseteq \mathbb{W} \rightsquigarrow L^* \subseteq \mathbb{W}$$

Kleene plus

$$L \subseteq \mathbb{W} \rightsquigarrow L^+ \subseteq \mathbb{W}$$

$$L^+ := \{ w_0 \dots w_n \,;\, n \in \mathbb{N} \text{ and } w_i \in L \}$$

CONCATENATION of finitely many words in L (non-empty)

$$L^* := L^+ \cup \{ \varepsilon \}$$

Stephen Kleene

Born	January 5, 1909 Hartford, Connecticut, U.S.
Died	January 25, 1994 (aged 85) Madison, Wisconsin, U.S.

<u>Question</u>: Are the regular languages closed under Kleene star & plus?

Let Σ be an alphabet. Among the finite strings over the set $\Sigma \cup \{\varnothing, \varepsilon, (,), +, ^+, ^*\}$; we shall define the notion of *regular expressions over* Σ by recursion:[10]

(1) The symbol $\varnothing$ is a regular expression;

(2) the symbol ε is a regular expression;

(3) every $a \in \Sigma$ is a regular expression,

(4) if R and S are regular expressions, then $(R + S)$ is a regular expression;

(5) if R and S are regular expressions, then (RS) is a regular expression;

(6) if R is a regular expression, then R^+ is a regular expression;

(7) if R is a regular expression, then R^* is a regular expression;

(8) nothing else is a regular expression.

Remark on parentheses:

$(a(b(cd)))$ is a regex

$abcd$ is <u>not</u> a regex

$(a+b)$ is a regex

$a+b$ is <u>not</u> a regex

Informally, we often allow to drop them when there is no ambiguity <u>BUT</u> the true regex has them.

We now associate languages to regular expressions by recursion:

(1) If $E = \varnothing$, then $\mathcal{L}(E) = \varnothing$;

(2) if $E = \varepsilon$, then $\mathcal{L}(E) = \{\varepsilon\}$;

(3) if $E = a$ for $a \in \Sigma$, then $\mathcal{L}(E) = \{a\}$;

(4) if R and S are regular expressions, then $\mathcal{L}((R + S)) = \mathcal{L}(R) \cup \mathcal{L}(S)$;

(5) if R and S are regular expressions, then $\mathcal{L}((RS)) = \mathcal{L}(R)\mathcal{L}(S)$;

(6) if R is a regular expression, then $\mathcal{L}(R^*) = \mathcal{L}(R)^*$;

(7) if R is a regular expression, then $\mathcal{L}(R^+) = \mathcal{L}(R)^+$.[11]

THEOREM

Let $L \subseteq \mathbb{W}$ be a language. Then the following are eq.:

(i) L is essentially regular

(ii) there is a regex R s.t.
$$\mathcal{L}(R) = L.$$

We now associate languages to regular expressions by recursion:

(1) If $E = \varnothing$, then $\mathcal{L}(E) = \varnothing$;

(2) if $E = \varepsilon$, then $\mathcal{L}(E) = \{\varepsilon\}$;

(3) if $E = a$ for $a \in \Sigma$, then $\mathcal{L}(E) = \{a\}$;

(4) if R and S are regular expressions, then $\mathcal{L}((R+S)) = \mathcal{L}(R) \cup \mathcal{L}(S)$;

(5) if R and S are regular expressions, then $\mathcal{L}((RS)) = \mathcal{L}(R)\mathcal{L}(S)$;

(6) if R is a regular expression, then $\mathcal{L}(R^*) = \mathcal{L}(R)^*$;

(7) if R is a regular expression, then $\mathcal{L}(R^+) = \mathcal{L}(R)^+$.[11]

<u>Remark</u> In this course, we do not prove (i) ⇒ (ii), only (ii) ⇒ (i). On ES#2 there are some examples relevant for the (i) ⇒ (ii) direction.

Proof of (ii) ⇒ (i):
Clearly (1) – (3) are ess. reg. languages.
(4) – (7) are about closure properties.
We already proved (4) & (5): union & concatenation.

Clearly enough to prove:
If L is regular, then so is L^+.

Start of proof of that closure property:
[If L regular, then L^+ regular.]

Let G be a regular grammar,
$L = \mathcal{L}(G)$.
Need to find grammar G^+ s.t.
$\mathcal{L}(G^+) = L^+$.

$G = (\Sigma, V, P, S)$

By previous discussion, can assume w.l.o.g. G is ε-adequate.

$P^+ := P \cup \{A \to aS \,;\, A \to a \in P\}$

$G^+(\Sigma, V, P^+, S)$

Claim (proof in Lecture IX):

$$\mathcal{L}(G^+) = L^+$$