VIII

AUTOMATA & FORMAL LANGUAGES

Eighth Lecture

22 October 2022

REMINDER A CORRECTION TO EXAMPLE (6) ON EXAMPLE SHEET #1 IS ON MOODLE. THE VERSION ON THE DPMMS WEB-SITE IS NOT YET UPDATED!

(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 $\alpha = A_1 ... A_n$ and $\beta = B_1 ... B_m$ for $A_i \in V$, $B_i \in \Omega$, and $n \leq m$. Let $X_1, ..., X_n$ be n new variables that do not occur in V and consider the following list of $2n$ rules:

$$
\begin{array}{lcl}
A_1 A_2 A_3 \cdots A_{n-2} A_{n-1} A_n & \to & X_1 A_2 A_3 \cdots A_{n-2} A_{n-1} A_n \\
X_1 A_2 A_3 \cdots A_{n-2} A_{n-1} A_n & \to & X_1 X_2 A_3 \cdots A_{n-2} A_{n-1} A_n \\
X_1 X_2 A_3 \cdots A_{n-2} A_{n-1} A_n & \to & X_1 X_2 X_3 \cdots A_{n-2} A_{n-1} A_n \\
 & \vdots & \\
X_1 X_2 X_3 \cdots X_{n-2} A_{n-1} A_n & \to & X_1 X_2 X_3 \cdots X_{n-2} X_{n-1} A_n \\
X_1 X_2 X_3 \cdots X_{n-2} X_{n-1} A_n & \to & X_1 X_2 X_3 \cdots X_{n-2} X_{n-1} X_n B_{n+1} \cdots B_m \\
X_1 X_2 X_3 \cdots X_{n-2} X_{n-1} X_n B_{n+1} \cdots B_m & \to & B_1 X_2 X_3 \cdots X_{n-2} X_{n-1} X_n B_{n+1} \cdots B_m \\
B_1 X_2 X_3 \cdots X_{n-2} X_{n-1} X_n B_{n+1} \cdots B_m & \to & B_1 B_2 X_3 \cdots X_{n-2} X_{n-1} X_n B_{n+1} \cdots B_m \\
 & \vdots & \\
B_1 B_2 B_3 \cdots B_{n-2} X_{n-1} X_n B_{n+1} \cdots B_m & \to & B_1 B_2 B_3 \cdots B_{n-2} B_{n-1} X_n B_{n+1} \cdots B_m \\
B_1 B_2 B_3 \cdots B_{n-2} B_{n-1} X_n B_{n+1} \cdots B_m & \to & B_1 B_2 B_3 \cdots B_{n-2} B_{n-1} B_n B_{n+1} \cdots B_m
\end{array}
$$

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.

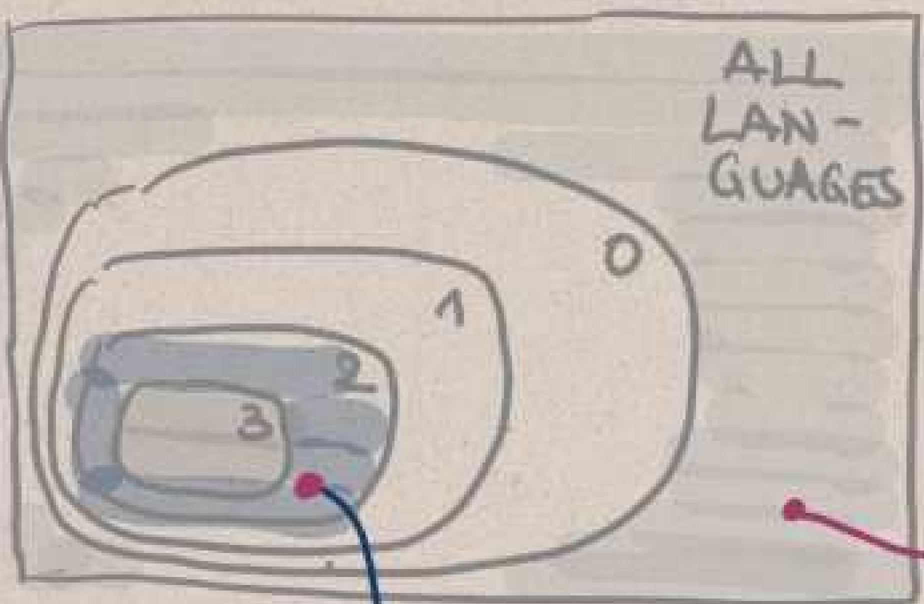
IF THE TRANSFORMATION RULES CONTAIN THE SYMBOL z, YOU HAVE THE OLD VERSION. THE CORRECTED VERSION HAS THE LETTER B.

RECAP:

$\boxed{\text{PUMPING LEMMA}}$: THE MOST IMPORTANT TOOL TO PROVE THAT LANGUAGES ARE NOT REGULAR.

$\boxed{\text{CLOSURE PROPERTIES}}$: THE CLASS OF REGULAR LANGUAGES IS CLOSED UNDER ALL FIVE OPERATIONS.

$L = \{0^k 1^k ; k \geq 1\}$ is not regular.



But L is context-free:

$$S \longrightarrow 0S1$$
$$S \longrightarrow 01$$

is a c-f grammar for L.

We already proved that not all languages are Type 0.

Application 1

There are c-f languages that are not regular.

Application 2

The EMPTINESS PROBLEM FOR REGULAR GRAMMARS:

Given G regular, is $\mathcal{L}(G) = \emptyset$?

Lemma If L is regular with pumping number n and $L \neq \emptyset$, then there is a word $w \in L$ s.t. $|w| < n$.

Proof. If $|v| \geq n$, then we can pump it down, so it cannot be the shortest word accepted.

So if $L \neq \emptyset$, there is a shortest word w and $|w| < n$. q.e.d.

Corollary The EMPTINESS PROBLEM for regular grammars is solvable

Pf. Step 1 Determine D s.t. $\mathcal{L}(D) = \mathcal{L}(G)$.
Step 2 Count $|Q|$ in D.
Step 3 Use solvability of WORD PROBLEM to check every single w with $|w| < n$.

Step 4

If all of these checks give "No",
then $\mathcal{L}(G) = \emptyset$.
Otherwise $\mathcal{L}(G) \neq \emptyset$.

q.e.d.

The remaining decision problem to solve:

EQUIVALENCE PROBLEM.

We'll do this in §2.8, but need the technique of MINIMISATION for this (§2.7).

§ 2.6 Regular Expressions

Two additional operations on languages.

KLEENE PLUS: $L^+ := \{ w \,;\, \exists w_0, \dots, w_n \in L \text{ s.t. } w = w_0 \dots w_n \}$ –

CONCATENATIONS OF FINITELY MANY ELEMENTS OF L [excluding zero!]

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

CONCATENATIONS OF FINITELY, POSSIBLY NONE, ELEMENTS OF L

Note: $\varepsilon \in L^*$, so L^* is never regular!

Let Σ be an alphabet. We define the *regular expressions over* Σ by recursion:

(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.

Reminder: L was essentially regular if there is L' regular s.t.

- $L = L'$ or
- $L = L' \cup \{\varepsilon\}$.

At best, L^* can be essentially regular!

(4) & (5) introduce lots of parentheses, sometimes unnecessary.

We often suppress superfluous parentheses, e.g.

$R+S$ instead of $(R+S)$

$RS+T$ instead of $((RS)+T)$

Let Σ be an alphabet. We define the *regular expressions over* Σ by recursion:

(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.

By recursion assign a language $\mathcal{L}(E)$ to every regular expression E:

REGULAR

ESSENTIALLY REGULAR

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

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

(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 $\mathcal{L}(R^+) = \mathcal{L}(R)^+$.

<u>Theorem</u> If E is a regular expression, then $\mathcal{L}(E)$ is essentially regular.

<u>REMARK</u>: The converse of this theorem holds, but will not be proved in this course.

[ES#2 will have some examples that deal with the converse.]

Proof of Theorem

$\boxed{G = (\Sigma, V, P, S) \text{ regular grammar}}$

Observe that the class of languages of the form $\mathcal{L}(E)$ is defined by recursion with basic languages $\emptyset, \{\varepsilon\}, \{a\}$ [which are all obviously essentially regular] and closure under the operations

- union
- concatenation
- $*$
- $+$

} done before [Note: only for regular; check that this implies the same for essentially regular]

We only need closure under $+$ and $*$; for these, it's enough to show that if L is regular, so is L^+.

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

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

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

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

"$\supseteq$": $w \in (\mathcal{L}(G))^+$

$\Longrightarrow w = w_0 \dots w_n$ $\quad w_i \in \mathcal{L}(G)$

Need to show that $w \in \mathcal{L}(G^+)$.

Proof by induction on n.

$n = 0$ $\quad w = w_0 \in \mathcal{L}(G) \subseteq \mathcal{L}(G^+)$

Since all rules in P are also in P^+, so every G-derivation is a G^+-derivation.

$n \mapsto n+1$ Assume true for n

$$w = w_0 \dots w_n w_{n+1}$$

By IH $w_0 \dots w_n \in \mathcal{L}(G^+)$

$w_{n+1} \in \mathcal{L}(G)$

$$S \xrightarrow{G^+} w_0 \dots w_n$$

By L2.1, the last rule used here is of the form $A \to a \in P$

So $A \to aS \in P^+$

Thus $\boxed{S \xrightarrow{G^+} w_0 \dots w_n S}$

$S \xrightarrow{G} w_{n+1}$

$\therefore S \xrightarrow{G^+} w_{n+1}$

$+$

$\boxed{w_0 \dots w_n S \xrightarrow{G^+} w_0 \dots w_n w_{n+1}}$

$S \xrightarrow{G^+} w_0 \dots w_{n+1}$.

$\Longrightarrow w_0 \dots w_{n+1} \in \mathcal{L}(G^+)$.

"$\subseteq$" $\mathcal{L}(G^+) \subseteq (\mathcal{L}(G))^+$.

w.l.o.g. assume that G was ε-adequate, so S never shows up on RHS of rules.

[We proved in Chapter 1 that each grammar G has an ε-adequate grammar G' s.t. $\mathcal{L}(G) = \mathcal{L}(G')$.]

Suppose we have

$$S \xrightarrow{G^+} w$$

Count the # of times that S occurs in this derivation.

Prove our claim by induction on that number, call it n.

If $n = 0$, then none of the extra rules of P^+ show up in $S \xrightarrow{G^+} w$,

so $S \xrightarrow{G} w$, so $w \in \mathcal{L}(G) \subseteq (\mathcal{L}(G))^+$.

$n \longmapsto n+1$

$$S \xrightarrow{G^+} w \quad ⊛$$

$$S \xrightarrow{G^+} \boxed{vS \xrightarrow{G^+} w}$$

where this is the last occurrence of S in the derivation

[This is possible by L 2.1]

$S \xrightarrow{G^+} vS$ means that last rule is one of the extra rules $A \to aS$ of P^+ [by ε-adequacy]

If $A \to aS \in P^+$, then $A \to a \in P$, so replacing $A \to aS$ by $A \to a$, we get $S \xrightarrow{G^+} v$.

$S \xrightarrow{G^+} v$ is a derivation with n occ. of S, so IH applies.

$\underset{IH}{\Longrightarrow}$ $\boxed{v = w_0 \dots w_k \text{ where } w_i \in \mathcal{L}(G).}$

[Note that we could strengthen our induction and show that $k=n$.]

By (*) from last page: $vS \xrightarrow{G^+} w$

By L 2.1, we know that $w = vu$ for some word u, so:

$$vS \xrightarrow{G^+} vu$$

Since no S shows up after the beginning, all rules are in P, so

$$vS \xrightarrow{G} vu$$

By Example (1): $S \xrightarrow{G} u$

$\Longrightarrow \boxed{u \in \mathcal{L}(G)}$

$w = vu = w_0 \dots w_k u$

$\in \mathcal{L}(G)^{k+1} \mathcal{L}(G) \subseteq \mathcal{L}(G)^+.$

q.e.d.