Automata & Formal Languages

Third Lecture

Saturday 19 October 2024

Recap

IN YOUR POST-PROCESSING OF LECTURE II, YOU HAVE DONE AT LEAST THREE EXAMPLES OF DERIVATIONS AND LEARNED HOW THEY WORK.

Ω finite set of symbols

$\Omega^+ \times \Omega^*$ production rules

(Ω, P) rewrite system

There are countably many RWS over Ω.

$\Omega = \Sigma \cup V$ LETTERS/VARIABLES

(Σ, V, S, P) grammar

There are countably many grammars over Σ, V.

Equivalent grammars: $\mathcal{L}(G) = \mathcal{L}(G')$

Isomorphic grammars.

There are countably many grammars over Σ up to isomorphism.

Question during Lecture II

Can you rewrite a letter?

The definition currently allows for that, so let us consider more reasonable grammars.

1.5 The Chomsky hierarchy

Fix Σ, V, and $S \in V$.

(1) A production rule $\alpha \to \beta$ is called *noncontracting* if $|\alpha| \leq |\beta|$.

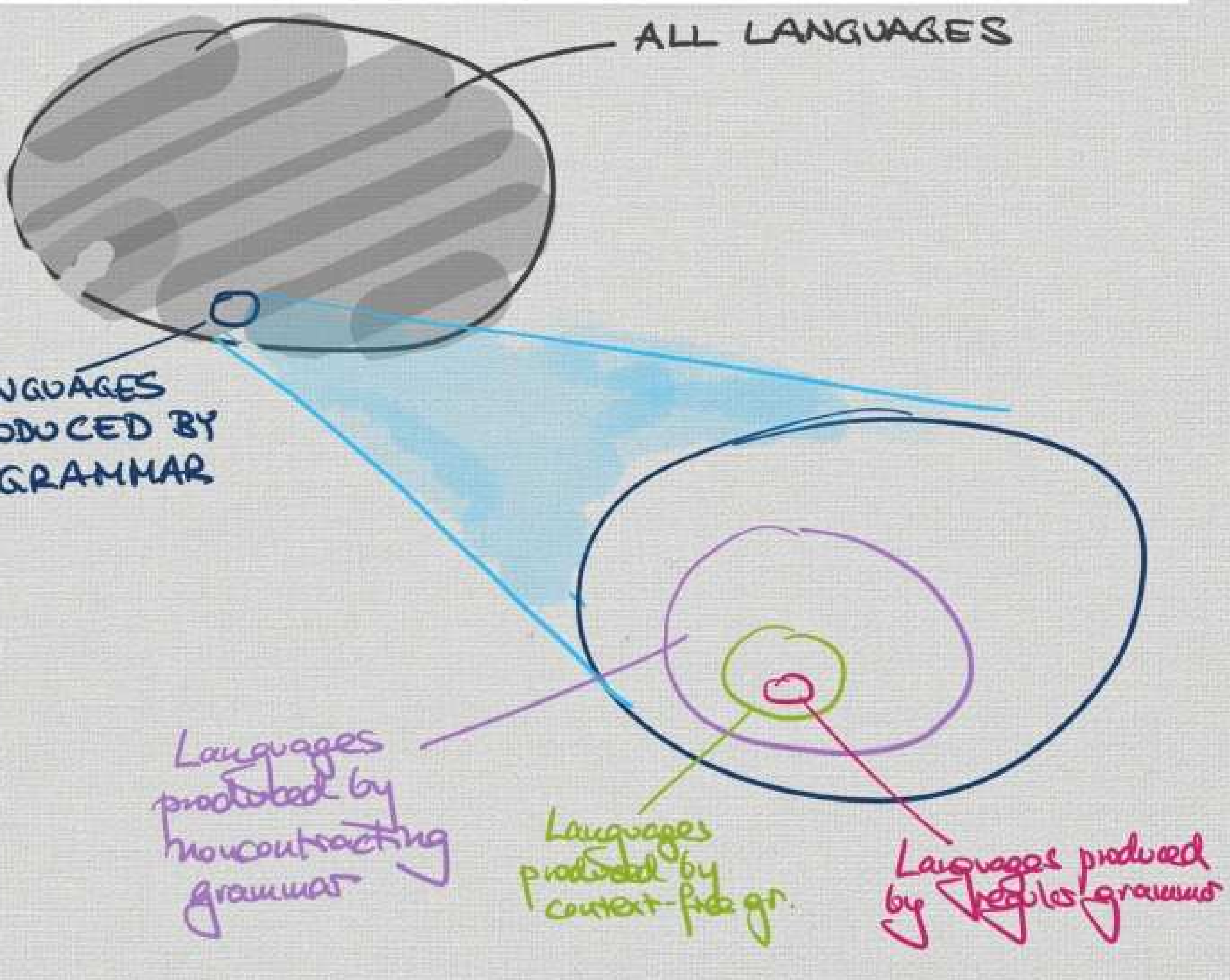
(2) A production rule $A \to \beta$ is called *context-free* if $A \in V$ and $|\beta| \geq 1$.

(3) Production rules $A \to a$ and $A \to aB$ are called *regular* if $A, B \in V$ and $a \in \Sigma$.

We observe that every regular rule is context-free and every context-free rule is noncontracting.

We call a grammar *noncontracting, context-free*, or *regular* if all of its production rules are noncontracting, context-free, or regular, respectively. If G is a noncontracting grammar, we know that any string in $\mathcal{D}(G, S)$ must have length at least $|S| = 1$ [proof by induction on the length of the derivation]. Thus, a noncontracting grammar can never derive the empty word ε.

We call a language *noncontracting, context-free*, or *regular* if it is produced by a noncontracting, context-free, or regular grammar, respectively.



TYPE 0: $\mathcal{L}(G)$ for any grammar G

TYPE 1: $\mathcal{L}(G)$ for noncontracting G

TYPE 2: $\mathcal{L}(G)$ for context-free G

TYPE 3: $\mathcal{L}(G)$ for regular G

Q : Are these inclusions strict?

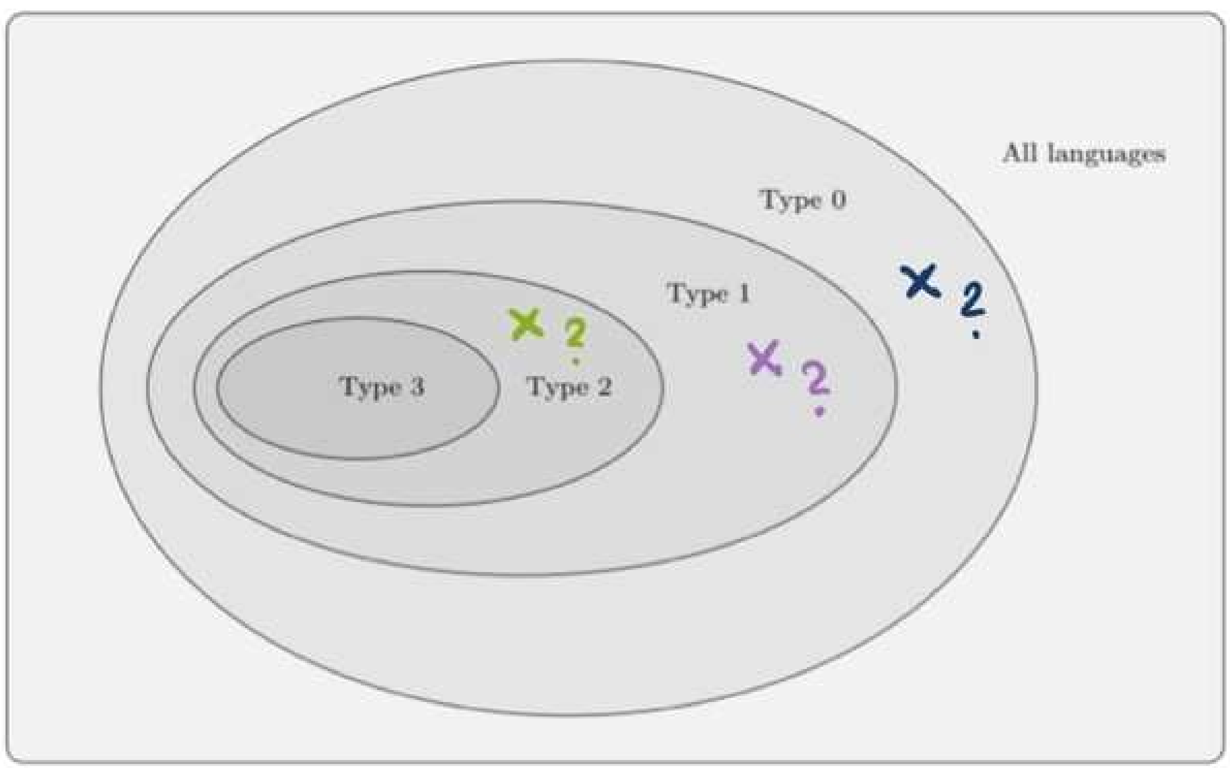


Figure 1: The Chomsky hierarchy

Example 1.10. Let $\Sigma = \{\mathbf{0}\}$, $V = \{S\}$, $P_0 := \{S \to \mathbf{00}S, S \to \mathbf{0}\}$, and $G_0 := (\Sigma, V, P_0, S)$. Then $\mathcal{L}(G_0)$ is the set of all odd-length words consisting of the letter $\mathbf{0}$.

$$P_1 := \{S \to \mathbf{0}S\mathbf{0}, S \to \mathbf{0}\},$$
$$P_2 := \{S \to S\mathbf{00}, S \to \mathbf{0}\},$$
$$P_3 := \{S \to \mathbf{00}S, S \to \mathbf{00}S\mathbf{00}, S \to \mathbf{0}\},$$
$$P_4 := \{S \to \mathbf{00}S, S \to S\mathbf{00}, S \to \mathbf{0}S\mathbf{0}, S \to \mathbf{0}\},$$
$$P_5 := \{S \to \mathbf{00}S, \mathbf{0}S\mathbf{0} \to \mathbf{000}, S \to \mathbf{0}\},$$
$$P_6 := \{S \to \mathbf{00}S, \mathbf{00}S \to \mathbf{0}S\mathbf{0}, S \to \mathbf{0}\}, \text{ or}$$
$$P_7 := \{S \to \mathbf{00}S, \mathbf{00}S \to \mathbf{0}, S \to \mathbf{0}\}, \text{ etc.}$$

CONTE-FREE

NON-CON-TRACTING not c-f

CONTRACTING

This means:
it's hard to prove that a LANGUAGE is not in some Chomsky class.

⇝ Chapters 2 & 3

§1.6 DECISION PROBLEMS

In this section, we'll formulate the typical decision problems for grammars. Let G and G' be formal grammars and $w \in \mathbb{W}$ be a word.

The word problem. Is there an algorithm to determine whether $w \in \mathcal{L}(G)$?

The emptiness problem. Is there an algorithm to determine whether $\mathcal{L}(G) = \varnothing$?

The equivalence problem. Is there an algorithm to determine whether $\mathcal{L}(G) = \mathcal{L}(G')$?

We say that a decision problem is *solvable* if there is such an algorithm and that it is *unsolvable* if there is not.

In general, all three problems are <u>unsolvable</u>.

However, restrictions to some Chomsky classes are solvable.

Chomsky Hierarchy

Context-free $A \longrightarrow \alpha$ $\quad (\alpha \in \Omega^+)$

Regular $A \longrightarrow a$ or $A \longrightarrow aB$

Theorem 1.21 The word problem for noncontracting grammars is solvable.

Proof

① There is a systematic way of listing all G-derivations of length $\leq n$.

Fun Q Find good upper bound to the number of derivations.

② Main Claim For each w, G there is an N s.t.

$w \in \mathcal{L}(G) \iff$ there is a G-derivation of w of length $\leq N$.

---

③ Proof of Thm from ②: Calculate N, list all G-derivations of length $\leq N$, check all of them. If one produces $w \rightsquigarrow$ "Yes"; o/w $\rightsquigarrow$ "No".

④ Proof of ② "$\Longleftarrow$" trivial.

"$\Longrightarrow$".

$$S \xrightarrow{\quad G \quad} w$$

$\uparrow$ length 1 $\qquad\qquad$ $\uparrow$ length $|w|$

$\underbrace{\quad}_{1}\ \underbrace{\quad}_{2}\ \underbrace{\quad}_{3}\ \cdots$

Suppose $w \in \mathcal{L}(G)$. Take a derivation of minimal length.

The derivation splits into $|w|$ blocks ... have same length.

How long is block n? There are $|\Omega|^n$ many different strings of length n.

By minimality there can be no repetitions.

$$N := \sum_{n=1}^{|w|} |\Omega|^n$$

1.7 Closure properties

There are a number of algebraic operations on languages that allow us to combine languages to new languages. Let $L, M \subseteq \mathbb{W}^+$ be any languages over an alphabet Σ.

(a) *Concatenation.* The language LM consists of words vw such that $v \in L$ and $w \in M$.

(b) *Union.* The language $L \cup M$ consists of words either in L or in M.

(c) *Intersection.* The language $L \cap M$ consists of words that are both in L and M.

(d) *Complement.* The language $\overline{L} := \mathbb{W}^+ \setminus L$ consists of nonempty words that are not in L.

(e) *Difference.* The language $L \setminus M$ consists of words in L that are not in M.

GOAL

Understand which Chomsky classes are closed under which operations.

If $L, M \subseteq W$

$LM := \{vw ; v \in L, w \in M\}$

<u>Concatenation</u> Given $G = (\Sigma, V, S, P)$ $\quad \Omega = \Sigma \cup V$

$G' = (\Sigma, V', S', P')$ $\quad \Omega' = \Sigma \cup V'$

Want H s.t. $\mathcal{L}(H) = \mathcal{L}(G)\mathcal{L}(G')$.

$V^* := V \cup V' \cup \{T\}$ where $T \notin V \cup V'$

$P^* := P \cup P' \cup \{T \to SS'\}$

$H := (\Sigma, V^*, T, P^*)$ CONCATENATION GRAMMAR

<u>Def</u> G is <u>variable-based</u> if only variables show up on LHS of rules.

<u>Prop 1.26</u> If G, G' are variable-based and $V \cap V' = \emptyset$, then $\mathcal{L}(H) = \mathcal{L}(G)\mathcal{L}(G')$.

<u>Remarks</u> (i) Assumptions are necessary $\rightsquigarrow$ ES#1.

(ii) The definition of H preserves being context-free and noncontracting (but not being regular).

<u>Proof of P 1.26</u> "$\supseteq$": If $S \xrightarrow{G} v$, $S' \xrightarrow{G'} w$, then $T \xrightarrow{H}_1 SS' \xrightarrow{H} vw$.

"$\subseteq$" If $u \in \mathcal{L}(H)$:

$$T \xrightarrow{H}_1 \alpha_0 \xrightarrow{H}_1 \alpha_1 \xrightarrow{H}_1 \alpha_2 \xrightarrow{H}_1 \cdots\cdots \xrightarrow{H}_1 w$$

SS' $\quad \beta S'$ $\quad S\gamma$ $\quad \in V$ $\quad \in V'$

① $\alpha_0 = SS'$

② T never shows up again, so all rules used after that are either in P or P'

③ Since $V \cap V' = \emptyset$, rules in P are applied to strings from V and rules in P' to strings in V'

④ Recursively, for each symbol that shows up anywhere in the α_i and w, we can identify whether it derived from S or from S'.

⑤ Thus, we can split w into the parts that come from S and S', respectively:

$w = vu$

⑥ Reordering the rules by first doing all of ~~the~~ P-rules and then all of the P'-rules, we get

$$\underbrace{T \xrightarrow{H}_1 SS' \xrightarrow{G} vS'}_{S \xrightarrow{G} v,\ \in \mathcal{L}(G)} \underbrace{\xrightarrow{G'} vu}_{S' \xrightarrow{G'} u,\ u \in \mathcal{L}(G')} = w$$

q.e.d.