Automata & Formal Languages

FIFTH LECTURE
16 October 2023

Lecture IV pages 8 & 9
CLOSURE PROPERTIES

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.

Let $G = (\Sigma, V, P, S)$ and $G' = (\Sigma, V', P', S')$ be two grammars over the same alphabet Σ.

(a) *Concatenation.* The *concatenation grammar of G and G'* is $(\Sigma, V \cup V' \cup \{T\}, P^*, T)$ with a new variable T and $P^* := \{T \to SS'\} \cup P \cup P'$.

(b) *Union.* The *union grammar of G and G'* is $(\Sigma, V \cup V' \cup \{T\}, P^*, T)$ with a new variable T and $P^* := \{T \to S, T \to S'\} \cup P \cup P'$.

Goal: if $\mathcal{L}(G) = L$ & $\mathcal{L}(G') = M$ and H is the UNION GRAMMAR, then $\mathcal{L}(H) = LM$.

Unfortunately.: That's not true in general. (Example Sheet #1 : (7))

Def. G is variable-based if all rules only have variables on the left hand side.

Lecture IV page 10:

Theorem: if G, G' are variable-based and $V \cap V' = \emptyset$, then

(a) if H is the union grammar of G, G', then $\mathcal{L}(H) = \mathcal{L}(G) \cup \mathcal{L}(G')$

(b) if H is the concatenation gr. of G, G', then $\mathcal{L}(H) = \mathcal{L}(G)\mathcal{L}(G')$

Lecture IV, page 10.

Theorem If G, G' are variable-based and $V \cap V' = \emptyset$, then

(a) if H is the union grammar of G, G', then $\mathcal{L}(H) = \mathcal{L}(G) \cup \mathcal{L}(G')$

(b) if H is the concatenation gr. of G, G', then $\mathcal{L}(H) = \mathcal{L}(G)\mathcal{L}(G')$

UNION GRAMMAR

$T \to S$, $T \to S'$ + all rules in P and P'.

Proof. By construction:

$\mathcal{L}(H) \supseteq \mathcal{L}(G) \cup \mathcal{L}(G')$.

NTS: $\mathcal{L}(H) \subseteq \mathcal{L}(G) \cup \mathcal{L}(G')$.

Let $T = \sigma_0 \xrightarrow{H}_1 \sigma_1 \xrightarrow{H}_1 \cdots \xrightarrow{H}_1 \sigma_n = w$

be an H-derivation.

There are only two possibilities for the rule used in the step $\sigma_0 \xrightarrow{H}_1 \sigma_1$.

Case 1 This rule is $T \to S$. Then $\sigma_1 = S$.

Then $S \notin V'$, $S \neq T$. So the second rule applied must be from P.

In general (proof by induction). If $i > 0$, σ_i contains only variables in V and the rules applied to σ_i are rules in P.

[uses $V \cap V' = \emptyset$ & grammar v-b.]

Thus $S = \sigma_1 \xrightarrow{G} w$, so $w \in \mathcal{L}(G)$.

Case 2 The rule is $T \to S'$. Same argument w/ roles of V & V' interchanged gives $w \in \mathcal{L}(G')$. q.e.d. (a)

Sketch of (b).

Details in typed notes. P 1.24

CONCATENATION GRAMMAR

$T \to SS'$ and all rules from P and P'.

$$T \xrightarrow[1]{\#} \sigma_1 \xrightarrow[1]{\#} \cdots \xrightarrow[1]{\#} \sigma_n = w$$
$$\sigma_1 = SS'$$

Define by recursion for each $i > 0$ the first half and second half of σ_i, depending on whether it came from S or S':

$$\sigma_i = \underbrace{\alpha_0 \cdots \alpha_l}_{\substack{\text{first half} \\ \text{from } S}} \underbrace{\alpha_{l+1} \cdots \alpha_k}_{\substack{\text{second half} \\ \text{from } S'}}$$

Using $V \cap V' = \emptyset$ & v-b, we get:

first half only uses variables in V
second ______ V'

and first half only uses rules in P
second ______ P'.

$\sigma_n = w = vu$ where v is the first half
u is the second half

$$S \xrightarrow{G} v \qquad S' \xrightarrow{G'} u$$

so $w = vu \in \mathcal{L}(G)\mathcal{L}(G')$.

q.e.d.

DECISION PROBLEMS

	Type 0	Type 1	Type 2	Type 3
WORD P.	?	✓	✓	✓
EMPTINESS P.	?	?	?	?
EQUIVALENCE P.	?	?	?	?

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!

A COMMENT ON LANGUAGES WITH THE EMPTY WORD

(§ 1.8 in the typed notes)

REMINDER

If G is noncontracting, then $\varepsilon \notin \mathcal{L}(G)$.

Basic ε-rule $S \longrightarrow \varepsilon$.

If G is a grammar and G' is constructed by adding the basic ε-rule:
$$\mathcal{L}(G') \supseteq \mathcal{L}(G) \cup \{\varepsilon\}.$$

Unfortunately! in general $\neq$.

In order to avoid this, we call a production rule *ε-adequate* if the symbol S only appears on the left-hand side. A grammar (G, V, P, S) is called *ε-adequate* if all of its production rules are. In order to avoid very lengthy theorem statements, we use the letter Q to stand for one of the four properties of being regular, context-free, context-sensitive, or noncontracting.

Proposition 1.27. Any grammar is equivalent to an ε-adequate grammar. Moreover, any grammar with property Q is equivalent to an ε-adequate grammar with property Q.

A grammar is called *essentially* Q if it is ε-adequate and all of its production rules are either the basic ε-rule or have property Q. A language is called called *essentially* Q if it is produced by an essentially Q grammar.

Proposition 1.28. A language L is essentially Q if and only if $L \backslash \{\varepsilon\}$ is Q.

Chapter 2 REGULAR LANGUAGES

Reminder A grammar is **REGULAR** if all of the production rules are of the form

TERMINAL RULE → $A \longrightarrow a \quad (a \in \Sigma, A \in V)$

NONTERMINAL RULE → or $A \longrightarrow aB \quad (a \in \Sigma, A, B \in V)$

Understanding regular derivations:

Let G be a regular grammar and

$S = \sigma_0 \xrightarrow{G}_1 \cdots \xrightarrow{G}_1 \sigma_n = w$ a G-derivation.

① # of variables in each σ_i is either 0 or 1.
[Induction]

② The only variable occurring in σ_i is always in last position.
[Induction]

③ Exactly one of the σ_i, viz. σ_n, has 0 variables.

④ $|\sigma_i| = i+1$ if $i < n$
$|\sigma_n| = n$.

(a) The *regular concatenation grammar of* G *and* G' is $(\Sigma, V \cup V', P^*, S)$ where $P^* := P' \cup (P \setminus \{A \to a;\ A \to a \in P\}) \cup \{A \to aS';\ A \to a \in P\}$.

(b) The *regular union grammar of* G *and* G' is $(\Sigma, V \cup V' \cup \{T\}, P^*, T)$ with a new variable T and $P^* := P \cup P' \cup \{T \to \alpha;\ S \to \alpha \in P\} \cup \{T \to \alpha;\ S' \to \alpha \in P'\}$.

Proposition (typed notes P 2.2)

If G, G' are regular grammars and $V \cap V' = \emptyset$, then:

(a) If H is the regular concatenation grammar, then $\mathcal{L}(H) = \mathcal{L}(G)\mathcal{L}(G')$

(b) If H is the regular union grammar, then $\mathcal{L}(H) \supseteq \mathcal{L}(G) \cup \mathcal{L}(G')$.

Proof. Find the proof of (b) in the typed notes.

(a). $\boxed{\supseteq}$ Suppose

$S \xrightarrow{G} u$ Show: $uv \in \mathcal{L}(H)$.

$S' \xrightarrow{G'} v \Rightarrow S' \xrightarrow{H} v$

If the last rule in $S \xrightarrow{G} u$ is of the form $A \to a$, then $A \to aS'$ is in the regular concatenation gr.

$S \xrightarrow{H} uS'$

$S \xrightarrow{H} uv$, so $uv \in \mathcal{L}(H)$.

"$\subseteq$". Suppose

$$S = \sigma_0 \xrightarrow{H}_1 \cdots \xrightarrow{H}_1 \sigma_n = w.$$

Show that $w = uv$ with $u \in \mathcal{L}(G)$ & $v \in \mathcal{L}(G')$.

We by analysis of derivations that

(1) the final rule is terminal
$A \to a \in P'$, so $A \in V'$.

(2) All nonterminal rules are of one of three types:

(a) $A \to aB$ with $A, B \in V$

(b) $A \to aB$ with $A, B \in V'$

(c) $A \to aB$ with $A \in V$, $B = S'$ and $A \to a \in P$.

Since I start with $S \in V$ and end with a variable in V', we know that there is $k < n$ s.t.

$$\underbrace{S = \sigma_0 \xrightarrow{H}_1 \cdots \xrightarrow{H}_1 \underset{\overset{\parallel}{\sigma_k}}{uS'}}_{\text{uses } G\text{-rules}} \underbrace{\xrightarrow{H}_1 \cdots \xrightarrow{H}_1 w = uv}_{\text{uses } G'\text{-rules}}$$

$$S \xrightarrow{G} u$$
$$S' \xrightarrow{G'} v.$$

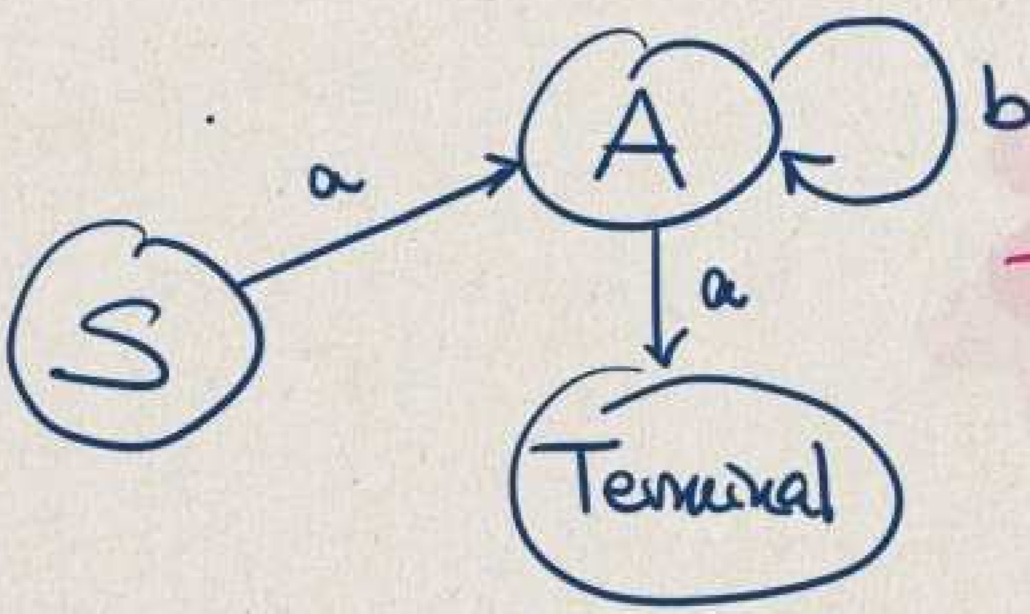
q.e.d.

Corollary The type 3 languages are closed under concatenation & union.

→ Remove two ? and replace these with ✓

Graphical representation of regular derivations:

Ex. $S \longrightarrow aA \longrightarrow abA \longrightarrow abbA \longrightarrow abba$



This is just an informal picture. Formal definitions will follow in Lecture VI.

This graphical representation is the motivation for automata.

Def. A tuple $(\Sigma, Q, \delta, q_0, F)$ is called a (deterministic) automaton

if Σ is an alphabet

Q is a finite set of states

$q_0 \in Q$ start state

$q_0 \notin F \subseteq Q$ set of accepting states

$\delta : Q \times \Sigma \longrightarrow Q$ transition function

INTUITION : Automaton starts in state q_0 and reads a word $w \in W$ letter-by-letter. For each letter a it reads (when it is in state $q \in Q$), it moves into state $\delta(q, a)$.

At the end, we reached final state $\bar{q}$. if $\bar{q} \in F$, we say ACCEPT!

$\bar{q} \notin F$, we say REJECT!

Next time: • graphical representation of automata (linking to ⊛ on page 9)

• connection between automata & grammars