

## Lecture 15.2. Automata and Grammars I, continued. Finite automata, regular languages, and Type 3 grammars,

Reading (for all of Lecture 15): 1

1. Languages, grammars, automata: basic concepts. (Chapter 16) 1
  2. Finite automata. 1
  3. Regular languages 1
  4. Pumping Theorem for  $\text{fal}'\text{s}$  1
  5. Type 3 grammars and their relation to fsa. 2
  6. Properties of regular languages. 3
  7. Inadequacy of Type 3 grammars for natural languages. 3
- Homework 15. [repeated from last time; nothing additional now] 3

### Reading (for all of Lecture 15):

Chapter 16, "Basic Concepts [of Languages, Grammars, and Automata]", of PtMW, pp. 431-452.  
Chapter 17, "Finite Automata, Regular Languages, and Type 3 Grammars" of PtMW, pp. 453-484.

Last time we did:

### 1. Languages, grammars, automata: basic concepts. (Chapter 16)

### 2. Finite automata.

### 3. Regular languages

(Definition of)

And we mentioned but did not prove the following theorem:

**Theorem.** (Kleene) A set of strings is a finite automaton language iff it is a regular language. (p.461).

We can sketch one half of the proof by showing how to construct a finite state automaton corresponding to any given regular expression. (See pp. 464-468)

Steps in the proof:

- i. The empty language is a  $\text{fal}$  (finite automaton language)
- ii. The unit language for every symbol in  $\Sigma$  is a  $\text{fal}$ .
- iii.  $\text{fal}'\text{s}$  are closed under union.
- iv.  $\text{fal}'\text{s}$  are closed under concatenation.
- v.  $\text{fal}'\text{s}$  are closed under the Kleene star operation.

Therefore the regular languages are included in the  $\text{fal}'\text{s}$ .

[The converse proof is more complicated.]

### 4. Pumping Theorem for $\text{fal}'\text{s}$

**Theorem 17.2** If  $L$  is an infinite  $\text{fal}$  over alphabet  $\Sigma$ , then there are strings  $x, y, z \in \Sigma^*$  such that  $x \neq \epsilon$  and  $xy^n z \in L$  for all  $n \geq 0$ .

Why: show that machines for infinite languages must have loops. The string  $y$  in the theorem corresponds to a string accepted during a traversal of a loop.

Note that theorem does not say 'iff'.

Main use of the theorem: in proving that some language is NOT regular.

Example:  $L = \{a^n b^n \mid n \geq 0\}$

But the pumping theorem is not always useful for showing a language to be non-regular.

## 5. Type 3 grammars and their relation to fsa.

Review the definition of formal grammars that we skipped before.

**Definition 16.2** (p.436) Let  $\Sigma = V_T \cup V_N$ . A *formal grammar*  $G$  is a quadruple  $\langle V_T, V_N, S, R \rangle$ , where  $V_T$  and  $V_N$  are finite disjoint sets,  $S$  is a distinguished member of  $V_N$ , and  $R$  is a finite set of ordered pairs in  $\Sigma^* V_N \Sigma^* \times \Sigma^*$ .

The ordered pairs in  $R$  are called *rules*; and  $\langle \alpha, \beta \rangle$  is usually written as  $\alpha \rightarrow \beta$  (read as " $\alpha$  rewrites as  $\beta$ "). The last condition above says that the string on the left must contain at least one non-terminal symbol. Other conditions are imposed for particular classes of grammars.

**Definition 16.3.** Given a grammar  $G = \langle V_T, V_N, S, R \rangle$ , a *derivation* is a sequence of strings  $x_1, x_2, \dots, x_n$  ( $n \geq 1$ ) such that  $x_1 = S$  and for each  $x_i$  ( $2 \leq i \leq n$ ),  $x_i$  is obtained from  $x_{i-1}$  by one application of some rule in  $R$ .

**Definition 16.4.** A grammar  $G$  *generates* a string  $x \in V_T^*$  if there is a derivation  $x_1, \dots, x_n$  by  $G$  such that  $x_n = x$ .

Note that by this definition only strings of terminal symbols are said to be generated.

**Definition 16.5.** The language generated by a grammar  $G$ , denoted by  $L(G)$ , is the set of all strings generated by  $G$ .

### The Chomsky hierarchy:

Types of grammars defined in terms of additional restrictions on the form of the rules:

**Type 0:** No restriction.

**Type 1:** Each rule is of the form  $\alpha A \beta \rightarrow \gamma \delta$ , where  $A \neq e$ .

**Type 2:** Each rule is of the form  $A \rightarrow \beta$ . ( $\beta$  may be  $e$ .)

**Type 3:** Each rule is of the form  $A \rightarrow xB$  or  $A \rightarrow x$ .

Common names:

**Type 0:** Unrestricted rewriting systems.

**Type 1:** Context-sensitive grammars.

**Type 2:** Context-free grammars.

**Type 3:** Right-linear, or regular, or finite state grammars.

**Correspondence of type 3 grammars and fsa's.** (Construction p.473) Every type 3 language is a fal. Can also show that every fal is a type 3 language (construction p.474).

Automata viewed as either generators or acceptors.  
Grammars viewed as either generators or acceptors.

## 6. Properties of regular languages.

Closure properties: We already know that the class of fal's is closed under union, concatenation, and Kleene star. What about intersection? Complementation?

Show complementation: if  $L$  is a fal, then  $\Sigma^* - L$  is a fal. Use fsa construction. Assume we have a deterministic fal  $M$  that accepts  $L$ . We can construct a deterministic fal  $M'$  which accepts the complement of  $L$  just by interchanging final and non-final states.

Therefore fal's are also closed under intersection (why?).

Therefore the class of regular languages over any fixed alphabet is a Boolean algebra.

Decidability properties: is there an algorithm for determining ... ?

- The membership question: yes.
- The emptiness question: yes.
- Does  $M$  accept all of  $\Sigma^*$ ?

Problem (opt. exercise): Is there an algorithm for determining, given two machines  $M_1, M_2$ , whether  $L(M_1) \cap L(M_2)$  ? (Yes. Show it.)

Is there an algorithmic solution to the question of whether two fsa's accept the same language?

## 7. Inadequacy of Type 3 grammars for natural languages.

Is English a regular language? No. Use closure under intersection plus the pumping theorem. (pp 477-479)

## Homework 15. [repeated from last time; nothing additional now]

**Note: for all of exercises 1-3 for Chapter 17, the alphabet must be specified as  $\{0,1\}$ .**

PtMW, Ch 17, pp 480-484. Exercises 1-11. You don't have to do all of them, but it would be good to try some of each 'kind', so that you have practice with the automata, the regular expressions, and the Type 3 grammars. Especially try exercises 7 and 8 so that you can get a feel for how finite state automata can form a Boolean algebra.