BIBLIOGRAPHIC NOTES

The undecidability of L_u is the basic result of Turing [1936]. Theorems 8.6 and 8.7, characterizing recursive and r.e. index sets, are from Rice [1953, 1956]. Post's correspondence problem was shown undecidable in Post [1946], and the proof of undecidability used here is patterned after Floyd [1964]. Lemmas 8.6 and 8.7, relating Turing machine computations to CFG's, are from Hartmanis [1967].

The fundamental papers on undecidable properties of CFL's are Bar Hillel, Perles, and Shamir [1961] and Ginsburg and Rose [1963a]. However, Theorem 8.9, on ambiguity, was proved independently by Cantor [1962], Floyd [1962a], and Chomsky and Schutzenberger [1963]. Theorem 8.16, undecidability of inherent ambiguity, is taken from Ginsburg and Ullian [1966]. Linear grammars and their decision properties have been studied by Greibach [1963, 1966] and Gross [1964]. The approach used in the solution to Exercise 8.9 is from Baker and Book [1974].

Greibach's theorem is from Greibach [1968]. A generalization appears in Hunt and Rosenkrantz [1974], which includes a solution to Exercise 8.11. Hopcroft and Ullman [1968a] shows that for certain classes of languages defined by automata, the decidability of membership and emptiness are related. The S_{mn}- and recursion theorems are from Kleene [1952]. Example 8.10, on the nonexistence of proofs of halting or nonhalting for all TM's, is from Hartmanis and Hopcroft [1976].

Hartmanis and Hopcroft [1968] are the authors of the basic paper relating problems about CFL's to the hierarchy of undecidable problems. Theorems 8.19 and 8.20, as well as Exercise 8.6, are from there. Additional results of this nature have been obtained by Cudia and Singletary [1968], Cudia [1970], Hartmanis [1969], and Reedy and Savitch [1975]. Exercise 8.4, on tag systems, is from Minsky [1961].

CHAPTER

9

THE CHOMSKY HIERARCHY

Of the three major classes of languages we have studied—the regular sets, the context-free languages, and the recursively enumerable languages—we have grammatically characterized only the CFL's. In this chapter we shall give grammatical definitions of the regular sets and the r.e. languages. We shall also introduce a new class of languages, lying between the CFL's and the r.e. languages, giving both machine and grammatical characterizations for this new class. The four classes of languages are often called the *Chomsky hierarchy*, after Noam Chomsky, who defined these classes as potential models of natural languages.

9.1 REGULAR GRAMMARS

If all productions of a CFG are of the form $A \to wB$ or $A \to w$, where A and B are variables and w is a (possibly empty) string of terminals, then we say the grammar is *right-linear*. If all productions are of the form $A \to Bw$ or $A \to w$, we call it *left-linear*. A right- or left-linear grammar is called a *regular grammar*.

Example 9.1 The language $\mathbf{0(10)^*}$ is generated by the right-linear grammar

$$\begin{aligned} S &\to 0A \\ A &\to 10A \mid \epsilon \end{aligned} \tag{9.1}$$

and by the left-linear grammar

$$S \to S10 \mid 0 \tag{9.2}$$

Equivalence of regular grammars and finite automata

The regular grammars characterize the regular sets, in the sense that a language is regular if and only if it has a left-linear grammar and if and only if it has a right-linear grammar. These results are proved in the next two theorems.

Theorem 9.1 If L has a regular grammar, then L is a regular set.

Proof First, suppose $L = L(G)$ for some right-linear grammar $G = (V, T, P, S)$. We construct an NFA with ϵ-moves, $M = (Q, T, \delta, [S], \{[\epsilon]\})$ that simulates derivations in G.

Q consists of the symbols $[\alpha]$ such that α is S or a (not necessarily proper) suffix of some right-hand side of a production in P.

We define δ by:

1) If A is a variable, then $\delta([A], \epsilon) = \{[\alpha] \mid A \to \alpha \text{ is a production}\}$.
2) If a is in T and α in $T^* \cup T^*V$, then $\delta([a\alpha], a) = \{[\alpha]\}$.

Then an easy induction on the length of a derivation or move sequence shows that $\delta([S], w)$ contains $[\alpha]$ if and only if $S \overset{*}{\Rightarrow} xA \Rightarrow xy\alpha$, where $A \to y\alpha$ is a production and $xy = w$, or if $\alpha = S$ and $w = \epsilon$. As $[\epsilon]$ is the unique final state, M accepts w if and only if $S \overset{*}{\Rightarrow} xA \Rightarrow w$. But since every derivation of a terminal string has at least one step, we see that M accepts w if and only if G generates w. Hence every right-linear grammar generates a regular set.

Now let $G = (V, T, P, S)$ be a left-linear grammar. Let $G' = (V, T, P', S)$, where P' consists of the productions of G with right sides reversed, that is,

$$P' = \{A \to \alpha \mid A \to \alpha^R \text{ is in } P\}.$$

If we reverse the productions of a left-linear grammar we get a right-linear grammar, and vice versa. Thus G' is a right-linear grammar, and it is easy to show that $L(G') = L(G)^R$. By the preceding paragraph, $L(G')$ is a regular set. But the regular sets are closed under reversal (Exercise 3.4g), so $L(G')^R = L(G)$ is also a regular set. Thus every right- or left-linear grammar defines a regular set. □

Example 9.2 The NFA constructed by Theorem 9.1 from grammar (9.1) is shown in Fig. 9.1.

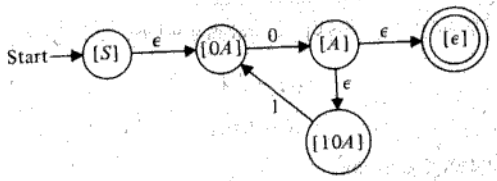

Fig. 9.1 NFA accepting **0(10)***.

Now consider grammar (9.2). If we reverse its productions we get

$$S \to 01S \mid 0$$

The construction of Theorem 9.1 for this grammar yields the NFA of Fig. 9.2(a). If we reverse the edges of that NFA and exchange initial and final states, we get another NFA for **0(10)***.

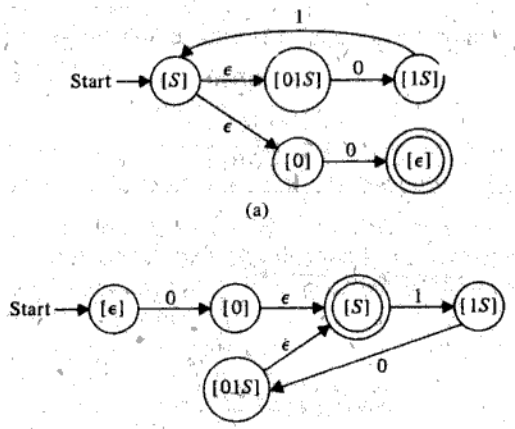

Fig. 9.2 Construction of an NFA for **0(10)*** from a left-linear grammar.

Theorem 9.2 If L is a regular set, then L is generated by some left-linear grammar and by some right-linear grammar.

Proof Let $L = L(M)$ for DFA $M = (Q, \Sigma, \delta, q_0, F)$. First suppose that q_0 is not a final state. Then $L = L(G)$ for right-linear grammar $G = (Q, \Sigma, P, q_0)$, where P consists of production $p \to aq$ whenever $\delta(p, a) = q$ and also $p \to a$ whenever $\delta(p, a)$ is a final state. Then clearly $\delta(p, w) = q$ if and only if $p \overset{*}{\Rightarrow} wq$. If wa is accepted by M, let $\delta(q_0, w) = p$, implying $q_0 \overset{*}{\Rightarrow} wp$. Also, $\delta(p, a)$ is final, so $p \to a$ is a production. Thus $q_0 \overset{*}{\Rightarrow} wa$. Conversely, let $q_0 \overset{*}{\Rightarrow} x$. Then $x = wa$, and $q_0 \overset{*}{\Rightarrow} wp \Rightarrow wa$ for some state (variable) p. Then $\delta(q_0, w) = p$, and $\delta(p, a)$ is final. Thus x is in $L(M)$. Hence $L(M) = L(G) = L$.

Now let q_0 be in F, so ϵ is in L. We note that the grammar G defined above generates $L - \{\epsilon\}$. We may modify G by adding a new start symbol S with productions $S \to q_0 \mid \epsilon$. The resulting grammar is still right-linear and generates L.

To produce a left-linear grammar for L, start with an NFA for L^R and then reverse the right sides of all productions of the resulting right-linear grammar. □

Example 9.3 In Fig. 9.3 we see a DFA for $\mathbf{0(10)^*}$.

The right-linear grammar from this DFA is

$$A \to 0B \mid 1D \mid 0$$
$$B \to 0D \mid 1C$$
$$C \to 0B \mid 1D \mid 0$$
$$D \to 0D \mid 1D$$

As D is useless we may eliminate it, obtaining grammar

$$A \to 0B \mid 0$$
$$B \to 1C$$
$$C \to 0B \mid 0$$

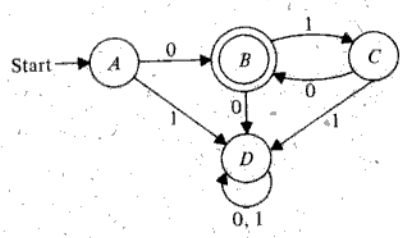

Fig. 9.3 DFA for $\mathbf{0(10)^*}$.

9.2 UNRESTRICTED GRAMMARS

The largest family of grammars in the Chomsky hierarchy permits productions of the form $\alpha \to \beta$, where α and β are arbitrary strings of grammar symbols, with $\alpha \neq \epsilon$. These grammars are known as *semi-Thue*, *type 0*, *phrase structure* or *unrestricted grammars*. We shall continue to use the 4-tuple notation $G = (V, T, P, S)$ for unrestricted grammars. We say $\gamma\alpha\delta \Rightarrow \gamma\beta\delta$ whenever $\alpha \to \beta$ is a production. As before, $\overset{*}{\Rightarrow}$ stands for the reflexive and transitive closure of the relation $\Rightarrow$:

$$L(G) = \{w \mid w \text{ is in } T^* \text{ and } S \overset{*}{\Rightarrow} w\},$$

exactly as for context-free grammars.

Example 9.4 A grammar generating $\{a^i \mid i \text{ is a positive power of } 2\}$ is given below.

1) $S \to ACaB$
2) $Ca \to aaC$
3) $CB \to DB$
4) $CB \to E$
5) $aD \to Da$
6) $AD \to AC$
7) $aE \to Ea$
8) $AE \to \epsilon$

A and B serve as left and right endmarkers for sentential forms; C is a marker that moves through the string of a's between A and B, doubling their number by production (2). When C hits the right endmarker B, it becomes a D or E by production (3) or (4). If a D is chosen, that D migrates left by production (5) until the left endmarker A is reached. At that point the D becomes a C again by production (6), and the process starts over. If an E is chosen, the right endmarker is consumed. The E migrates left by production (7) and consumes the left endmarker, leaving a string of 2^i a's for some $i > 0$. We can prove by induction on the number of steps in the derivation that if production (4) is never used, then any sentential form is either

i) S,

ii) of the form Aa^iCa^jB, where $i + 2j$ is a positive power of 2, or

iii) of the form Aa^iDa^jB, where $i + j$ is a positive power of 2.

When we use production (4) we are left with a sentential form Aa^iE, where i is a positive power of 2. Then the only possible steps in a derivation are i applications of (7) to yield AEa^i followed by one application of (8), producing sentence a^i, where i is a positive power of 2.

Equivalence of type 0 grammars and Turing machines

We shall prove in the next two theorems that unrestricted grammars characterize the r.e. languages. The first theorem states that every type-0 language generates an r.e. set. An easy proof would be to give an algorithm for enumerating all strings generated by a type-0 grammar. Instead we construct a Turing machine recognizer for sentences generated by a type-0 grammar, since this construction will be useful later for a similar proof about context-sensitive grammars (the remaining class in the Chomsky hierarchy).

Theorem 9.3 If L is $L(G)$ for unrestricted grammar $G = (V, T, P, S)$, then L is an r.e. language.

Proof Let us construct a nondeterministic two-tape Turing machine M to recognize L. M's first tape is the input, on which a string w will be placed. The second tape is used to hold a sentential form α of G. M initializes α to S. Then M repeatedly does the following:

1) Nondeterministically select a position i in α, so that any i between 1 and $|\alpha|$ can be chosen. That is, start at the left, and repeatedly choose to move right or select the present position.
2) Nondeterministically select a production $\beta \to \gamma$ of G.
3) If β appears beginning in position i of α, replace β by γ there, using the "shifting-over" technique of Section 7.4, perhaps shifting left if $|\gamma| < |\beta|$.