Grammars

See Chapter 5 of the text

Is the language of arithmetic expressions such as 22+33*44 regular?

Well, yes if you think that such an expression just has the form
number operator number operator ... number
Here is a regular expression for the language: $(\text{digit}^{+}\ \text{operator})^{*}\text{digit}^{+}$

This fails as soon as we add parentheses to the language to get (22+33)*45 or (((((3))))) It is easy to show that the parenthesized language fails the pumping lemma test.

We need a way to specify languages that are more complex than regular languages. So we turn to grammars.

Before we see definitions here is an example: a grammar that defines the language of parenthesized arithmetic expressions.

E => E+T

E => E-T

E => T

T => T*F

T => T/F

T => F

F => (E)

F => G

G => G digit

G =>digit

We *derive* a string in the language determined by this grammar by starting with E (the start symbol) and repeatedly replacing one of the symbols E,T,F,G with the right hand side of one of the grammar rules for that symbol. For example, we can replace E with E+T, E-T, or T. This substitution process continues until there are no remaining E,T,F or G symbols.

For example:

E => T
=> T*F
=> F*F
=> G*F
=> 3*F
=> 3*(E)
=> 3*(E+T)
=> 3*(T+T)
=> 3*(F+T)
=> 3*(G+T)
=> 3*(4+T)
=> 3*(4+F)
=> 3*(4+G)
=> 3*(4+5)

In each step I have underlined the grammar symbol that is expanded to generate the next step.

On the other hand, 3++4 is not a string that can be derived from this grammar. If we tried to derive it, the only rule with a + symbol is E => E+T. Nothing in T or below contains a +, so the E on the right hand side would need to generate 3+. That E could again go to E+T to match the 3 and +, but the resulting T can't derive ε.

In general, a *grammar* is a 4-tuple (Σ,N,S,P) where

- Σ is a finite alphabet of *terminal* symbols (like Σ)
- N is a finite alphabet of *non-terminal* or *grammar* symbols
- S $\in$ N is the *start* symbol
- P is a finite set of *production rules*. Each rule has the form α => β, where α and β are both strings in (Σ+N)*.

To save space, we often write all of the rules that have the same left side on one line, separating the right sides with |. The previous grammar would be written

E => E+T | E-T | T
T => T*F | T/F | F
F => (E) | G
G => G digit | digit

A *derivation* is a sequence of steps that replaces the left side of a production rule with the right side of this rule. We usually continue derivations until we have derived a string of terminal symbols.

Here is another grammar:

Terminal symbols: {a, b, c}

Nonterminal symbols: {S, T, U}

The start symbol is S

Rules:

S => aSTU

S => abU

bT => bb

bU => bc

UT => TU

cU => cc

Here is a quick derivation:

S => a<u>bU</u>

=> abc

Here is another derivation:

S => aSTU

=> aabUTU

=> aabTUU

=> aabbUU

=> aabbcU

=> aabbcc

It isn't terribly difficult to show that this grammar generates the language $\{a^n b^n c^n : n >= 1\}$

Grammars can be categorized by the types of rules they allow:

Regular Grammars: All production rules are either of the form A => a or A => aB, where A and B are nonterminal symbols and a is a terminal symbol.

Context Free: All production rules have the form A => α, where A is a single nonterminal symbol and α might have both terminals and nonterminals.

Context Sensitive: All production rules have the form α => β, where α and β are strings in $(\Sigma+N)^*$ with $|\alpha| <= |\beta|$

Arbitrary

Here is a look ahead:

The Chomsky Hierarchy

Grammar	**Machine that Recognizes**
Regular	DFA
Context Free	PDA (DFA+Stack)
Context Sensitive	Turing Machine with bounded memory
Arbitrary	Turing Machine

For any type of grammar, if w_1 and w_2 are strings in $(\Sigma+N)^*$, we say $w_1 \Rightarrow w_2$ if there is a grammar rule $\alpha \Rightarrow \beta$, where α is a substring of w_1 and w_2 can be produced from w_1 by replacing α with β.

We say $w_1 \overset{*}{\Rightarrow} w_2$ if there is a sequence of strings $v_1 \ldots v_n$ with

$$w_1 = v_1 \Rightarrow v_2 \Rightarrow \ldots \Rightarrow v_n = w_2$$

The language defined by the grammar is $\{w \in \Sigma^* \mid S \overset{*}{\Rightarrow} w\}$