Programming Languages: Lecture 7
Regex

Rishabh Dhiman

15 January 2022

1 Deterministic Finite Automaton

For a deterministic automaton,

1. No ε-transitions exist
2. Exactly one transition exists for every character and state.

We can convert NFA to DFA using the powerset construction.

2 Scanning with Output

Scanner is very similar to a scanner.

- DFA just accepts or rejects a token/lexeme. Scanner needs to classify them all.
- Algorithms minimize number of accepting states of DFA, this is not desirable in scanners where you need to classify the strings.

3 Production rules

Form production rules from the DFA.

$$
\begin{aligned}
S &\to aA \mid bB \\
A &\to aA \mid bC \\
B &\to aA \mid bB \\
C &\to aA \mid bD \\
D &\to aA \mid bB \mid \varepsilon
\end{aligned}
$$

You start with the start state, S here. Then apply the production rules to generate strings. End at a terminal variable, one with a transition to ε.

This is a regular or right-linear grammar as all the variables come at the end in the production rules.

We can have production rules which don't generate a regular language. Consider the language

$$P = \{a^n b^n \mid n \in \mathbb{N}_0\}.$$

P is not regular (there exists $i \neq j$ such that a^i and a^j end at same state in DFA, which, $a^i b^i$ being accepted implies $a^j b^i$ gets accepted) but can easily be represented by the production rule

$$X \to aXb \mid \varepsilon.$$

4 Grammar

Definition 1 (Grammar)**.** A grammar $G = \langle N, T, P, S \rangle$ consists of

- a set N of nonterminal symbols,
- a start symbol $S \in N$,
- a set T of terminal symbols or the alphabet,
- a set P of productions or rewrite rules where each rule is of the form $\alpha \to \beta$ for $\alpha, \beta \in (N \cup T)^*$.

Definition 2. Given a grammar G, any $\alpha \in (N \cup T)^*$ is called a sentential form. Any $x \in T^*$ is called a sentence.