CS241 – Week 5 Tutorial Solutions

Regular Languages Part 2 & Context-Free Grammars

Spring 2018

1 Scanning Exercises

1. 0xa0xb0xcd

   ```
   HEXINT 0xa0
   ID xb0xcd
   ```

2. 0xend---

   ```
   HEXINT 0xe
   ID nd
   DECR --
   MINUS -
   ```

3. 1234-120xb

   ```
   INT 1234
   INT -120
   ID xb
   ```

4. abcend--en-3

   ```
   ID abcend
   DECR --
   ERROR en-
   ```

5. 01end-end10

   ```
   ZERO 0
   INT 1
   END end
   MINUS -
   ID end10
   ```

2 Context-Free Grammar Exercises

2.1 Derivations

Let G be the following grammar:

$$S \to aAb$$
$$S \to bAa$$
$$A \to aSa$$
$$A \to bSb$$
$$A \to c$$

1. Show that the word $abacbbb$ is in $L(G)$ by giving a derivation.

   Solution:

   $$S \Rightarrow aAb \Rightarrow abSbb \Rightarrow abaAbbb \Rightarrow abacbbb$$

2. Which of the following are valid derivations of words in $L(G)$?

   (a) $S \Rightarrow aAb \Rightarrow aaAab \Rightarrow aacab$

   (b) $S \Rightarrow aAb \Rightarrow aaAbb \Rightarrow aacbb$

   (c) $S \Rightarrow bAa \Rightarrow bca$

   (d) $A \Rightarrow aSa \Rightarrow aaAba \Rightarrow aacba$

   (e) $S \Rightarrow aaSab \Rightarrow aaaAbab \Rightarrow aaacbab$

   Solution:

   (a) is invalid since there is no $A \to aAa$ rule in G.

   (b) is invalid since there is no $A \to aSb$ rule in G.

   (c) is valid.

   (d) is invalid since A is not the start symbol of the grammar.

   (e) is invalid since there is no $S \to aaSab$ rule. However, it would be valid if we said $S \overset{*}{\Rightarrow} aaSab$, since $S \Rightarrow aAb \Rightarrow aaSab$.

2.2 Designing Grammars

1. Write context-free grammars for the following languages:

   (a) The language $L = \{\text{my}, \text{name}, \text{is}, \text{inigo}, \text{montoya}\}$.

   (b) The language of all words over $\Sigma = \{a, b, c\}$ not beginning with b.

   (c) The language defined by the regular expression $(0|1)^*((00)^*1)^*010$.

   (d) The language $L = \{a^n b^n c^m d^m : n, m \in \mathbb{N}\}$, where a^n means "n copies of a in a row."

   (e) The language of palindromes over $\Sigma = \{a, b\}$. A *palindrome* is a word or phrase which is spelled the same forwards and backwards, for example "ABBA," "racecar," or "Able was I ere I saw Elba."

   Solution:

   (a) This is just a union, which is very easy to enforce in a grammar.

   $$S \to my$$
   $$S \to name$$
   $$S \to is$$
   $$S \to inigo$$
   $$S \to montoya$$

   (b) We need to be careful here: ϵ does not begin with a b! Otherwise, we simply allow S to have an a or c followed by A, where A represents anything.

   $$S \to aA$$
   $$S \to cA$$
   $$S \to \epsilon$$
   $$A \to aA$$
   $$A \to bA$$
   $$A \to cA$$
   $$A \to \epsilon$$

   (c) It turns out every regular expression can be converted via a simple algorithm to a regular expression. We can do that to get a grammar that looks fairly similar to this one, but it may help to try to encode $a^*, aa, a|b$, and a as CFGs and then think about how you might combine them.

   $$S \to AB010$$
   $$A \to 0A$$
   $$A \to 1A$$
   $$A \to \epsilon$$
   $$B \to C1B$$
   $$B \to \epsilon$$
   $$C \to 00C$$
   $$C \to \epsilon$$

(d) This is simply two copies of the "matching parentheses" grammar, where ab and cd are our parentheses in the two parts of the problem. We can simply write down those two grammars then combine them with $S \to AB$.

$$S \to AB$$
$$A \to aAb$$
$$A \to \epsilon$$
$$B \to cBd$$
$$B \to \epsilon$$

(e) Even-length palindromes aren't too bad: we just need to enforce that we add an a or b to both the start and end at the same time. However, we need to be careful: ϵ is a palindrome, and some (odd-length) palindromes have a single copy of a or b in the exact centre. Combining these ideas we get:

$$S \to aSa$$
$$S \to bSb$$
$$S \to a$$
$$S \to b$$
$$S \to \epsilon$$

2. Argue that every regular language is context-free by describing how to convert any regular expression into a context-free grammar.

Solution:
Recall that a regular expression takes one of five forms. We will present a grammar which handles each case: then by induction on the complexity of the regular expression we see that any regular expression can be encoded in a CFG, and thus every regular language is context-free.

- $R = \epsilon$, the empty word. The corresponding production is simply $A \to \epsilon$.
- $R = a$, a single symbol. The corresponding production is simply $A \to a$.
- $R = R_1 R_2$, the concatenation of two regular expressions. The corresponding production is $A \to A_1 A_2$, where A_1, A_2 are the rules corresponding to R_1, R_2.
- $R = R_1 | R_2$, the union of two regular expressions. This has two corresponding productions: $A \to A_1$ and $A \to A_2$.
- $R = R_1^*$, the Kleene closure of a regular expression. Once again, we need two productions: $A \to A_1 A$ and $A \to \epsilon$.

Since all regular expressions take one of these four forms and every regular language can be encoded by a regular expression, every regular language is context-free.