Compiler Construction

Exercises

1 Review of some Topics in Formal Languages

1.

(a) Prove that two words x, y commute (i.e., satisfy $xy = yx$) if and only if there exists a word w such that $x = w^m, y = w^n$ for some non-negative integers m, n.

(b) Characterize all pairs of words x, y, z satisfying the equality $x^2y^2 = z^2$.

(c) Characterize all triples of words x, y, z satisfying the equality $xyz = yzx$.

2. Let $\Sigma = \{a, b, \ldots, z, 0, 1, \ldots, 9, \_\}$. Let L be the language consisting of all non-empty words over Σ which (i) do not start with a digit, (ii) do not contain two consecutive occurrences of '_', and (iii) do not end with '_'.

(a) Construct a regular expression r such that $L(r) = L$.

(b) Construct a DFA accepting L.

(c) How many words of each length n does L include?

3. Show that the following languages over Σ are regular:

(a) The collection of all words whose length is congruent to k modulo m (where $0 \leq k \leq m - 1$).

(b) The collection of all words having the property that, for some fixed positive integer k and all pairs of letters σ_1, σ_2, the difference between the number of occurrences of σ_1 and of σ_2 in every prefix does not exceed k.

4. How many words of length n do the languages, corresponding to the following regular expressions, contain?

(a) $\sigma_1^*\sigma_2^*\dots\sigma_k^*$ (where the σ_i's are all distinct).

(b) $(0 \cup 11 \cup 22 \cup 3333 \cup 4444 \cup 5555 \cup 6666)^*$.

5. Show that, if L is a regular language, then so are the following languages:

(a) The language obtained by replacing, in each word of L, each occurrence of aa by b. (The replacement is done consecutively; thus, the block a^{2k} is replaced by b^k and the block a^{2k+1} by $b^k a$.)

(b) The language obtained from L by deleting the second last letter in every word of length 2 or more:

6. Let L_1, L_2 be two languages over Σ. Show that the "equation"

$$L_1 L \cup L_2 = L$$

has a solution. Moreover, if L_1, L_2 are both regular (or both context-free), then there exists a solution L with the same property.

7. Let $\Sigma = \{a, b, c\}$. Construct DFA's accepting the following languages:

(a) All words containing neither aaa nor aca as a subword.

(b) All words containing either $ababa$ or $abcba$ as a subword.

8. Construct NFA's accepting the languages corresponding to the following regular expressions:

(a) $bab(bba \cup abb)^* bab$.

(b) $ab(ab \cup bba)^* \cup a(ba \cup \phi^*)bba$.

9. Present an algorithm which, given a DFA, returns all words of minimal length accepted by it (or an error if it accepts the empty language).

10.

(a) Show that an infinite regular language may be written as an infinite disjoint union of infinite regular languages.

(b) Does an infinite context-free language necessarily contain an infinite regular language?

11. Show that the following languages are not regular:

(a) $\{a^m b^n c^{m+n} : \ m, n \geq 0\}$.

(b) $\{a^k b^l c^m d^n : \ k, l, m, n \geq 0, |\{k, l, m, n\}| \geq 2\}$.

(c) $\{0^{m^3+n^3} : \ m, n \geq 0\}$.

12. Given a set of non-negative integers, the set of their expansions in base 10 forms a partial language of $\{0, 1, \ldots, 9\}^*$. For each of the following sets show that the corresponding language is regular or not (as indicated):

(a) All powers of 1000 (regular).

(b) All powers of 7 (not regular).

(c) All perfect cubes (not regular).

13. Find the language accepted by the grammar:

$S \to AB \mid BA\,,$

$A \to aAb \mid \varepsilon\,,$

$B \to bBa \mid \varepsilon\,.$

14. Let $a_1, a_2, \ldots, a_r$ be positive integers and $b_1, b_2, \ldots, b_r$ non-negative integers. Consider the language $\{\sigma_1^{a_1 n+b_1} \sigma_2^{a_2 n+b_2} \ldots \sigma_r^{a_r n+b_r} : n \geq 0\}$, where $\sigma_1, \sigma_2, \ldots, \sigma_r$ are any letters, not necessarily distinct. Specify the conditions under which this language is context-free.

15. Construct a DFA accepting the same language as the grammar:

$S \to abS \mid baS \mid bcA,$

$A \to babB,$

$B \to bcbaC,$

$C \to abaC \mid bC \mid \varepsilon.$

16. Construct a pushdown automaton accepting the same language as the grammar:

$S \to \varepsilon \mid SbS \mid AbS \mid SaB,$

$A \to Bb \mid a^2,$

$B \to b^2S \mid baA \mid b^2.$

17. Consider the grammar G defined by:

$$S \to S + S \mid S * S \mid a \mid b \mid c.$$

(a) Is the language $L(G)$ regular? If yes – find a DFA accepting the same language as G and a regular grammar G' equivalent to G. If not – prove it.

(b) How many words of each length does the language $L(G)$ include?

(c) How many derivation trees yield each word in $L(G)$? (Hint: Find a recurrence for the sequence which expresses the required number as a function of the word's length.)

2 Lexical Analysis

18.

(a) Write a regular expression r such that $L(r)$ consists of all identifiers (i.e., all strings of letters and digits, starting with a letter), with the exception of the three strings "if", "int" and "integer".

(b) Construct a DFA which recognizes the string "if" as a token of type IF, the strings "int" and "integer" as tokens of type NUM, and all other identifiers as tokens of type ID.

19.

(a) Given an arbitrary fixed integer $d \geq 2$, construct a regular expression r such that $L(r)$ consists of all base d expansions of even (positive) integers. (Thus, the alphabet consists of all digits in base d.)

(b) Construct a DFA which recognizes every non-negative number, expanded in base d, as EVEN or as ODD.

20.

(a) Given a regular expression r, define a regular expression $\vec{r}$ for vectors (of any non-negative length) of elements of type r. The entries of a vector are separated by commas and grouped by parentheses.

(b) Given a DFA recognizing elements of type r, construct a DFA which recognizes vectors of such elements.

(c) Can you design your DFA so as to recognize vectors whose length is (i) 3 modulo 10? (ii) a prime?

(d) The regular expression $\vec{\vec{r}}$ represents vectors of vectors of elements of type r. Can you construct a regular expression $[r]$ for matrices (i.e., vectors whose entries are vectors of the same length) of elements of type r?

21.

(a) For any $n \geq 0$, write a regular definition for the language of balanced parentheses of nesting level up to n.

(b) Write a computer program which, for given n, will output a DFA for this language. The DFA should inform of the nesting level. (Represent the DFA in any way you like – by a transition table, a graph, etc.)

22. Write a regular definition for the language of all strings over $\{a, b, \dots, z\}$, not containing "if" as a substring.

23. In Java, the command
`a = b+++--c;`
passes compilation, whereas the command
`a = b+++++c;`
does not. Why?

24. In a certain computer language, identifiers are strings of letters and digits, starting with a letter, with the additional constraint that a character should not appear more than once in the name. (A lower-case letter and the corresponding upper-case letter are considered as distinct.) Construct a DFA, with a minimal possible number of states, recognizing identifiers. How many states does this DFA consist of?

25. Consider the class of "special" NFAs discussed in class (with an initial state having no incoming transitions, a single accepting states with no outgoing transitions, and either a single σ-transition or up to two ε-transitions from any of the non-accepting states). Denoting the class by $\mathcal{S}$, consider the following problem:
Given M in $\mathcal{S}$, does there exist an equivalent M' in $\mathcal{S}$ having less states than M?
Is the problem decidable?

26. A "special" NFA (see Question 25), accepting the language $L(w_1|w_2|\dots|w_n)$, where $w_1, w_2, \dots, w_n$ are any words, is constructed according to the algorithm discussed in class. (Here, the construction is according to the order of the characters in the expression.)

(a) How many states does the NFA consist of?

(b) Suppose the words w_i are distinct from each other and none of them is empty. Does the NFA above consist of the minimal possible number of states (within the class of such NFAs)?

27. Consider the NFA

$$M = (\{q_{ij} : 1 \le i \le m, 1 \le j \le n\}, \{a, b\}, \Delta, q_{11}, q_{mn}) ,$$

where

$$\Delta(q_{ij}, \varepsilon) = \begin{cases} \{q_{i,j+1}, q_{i+1,j}\}, & 1 \le i \le m-1, \quad 1 \le j \le n-1, \\ \{q_{m,j+1}\}, & i = m, \quad 1 \le j \le n-1, \\ \{q_{i+1,n}\}, & 1 \le i \le m-1, \quad j = n, \\ \emptyset, & i = m, \quad j = n, \end{cases}$$

$$\Delta(q_{ij}, a) = \begin{cases} \{q_{i,j+1}\}, & 1 \le i \le m, \quad 1 \le j \le n-1, \\ \emptyset, & 1 \le i \le m, \quad j = n, \end{cases}$$

$$\Delta(q_{ij}, b) = \begin{cases} \{q_{i+1,j}\}, & 1 \le i \le m-1, \quad 1 \le j \le n, \\ \emptyset, & i = m, \quad 1 \le j \le n. \end{cases}$$

How many states are there in the equivalent DFA, constructed using the algorithm discussed in class?

3 Syntactic Analysis

28. Consider the grammar G_1 given by:

$S \longrightarrow iSeS \mid iS \mid \varepsilon\,,$

and the grammar G_2 given by:

$S \longrightarrow M \mid U\,,$

$M \longrightarrow iMeM \mid \varepsilon\,,$

$U \longrightarrow iMeU \mid iS\,.$

(Intuitively, you should think of these grammars as the two grammars presented in class for conditional statements. Here we deal only with occurrences of the words *if* and *else*, represented by i and e, respectively. M and U stand for *matched* and *unmatched*, respectively.)

(a) Prove that $L(G_1) = L(G_2)$.

(b) Which words are obtained by a unique derivation tree in G_1?

(c) Write a program that, given a word in $\{i, e\}^*$, finds the number of derivation trees (if any) over G_1 producing this word. Prove that your algorithm works in polynomial time in the length of the input.

(d) Prove that G_2 is unambiguous.

(e) Write a program that, given a word in $\{i, e\}^*$, finds the unique derivation sequence over G_2 producing this word (and gives an error message if the word does not belong to $L(G_2)$).

29. Consider the following grammar, designed to solve the ambiguity problem of the **if-then-else** grammar presented in class:

$$\textit{stmt} \to \textbf{if } \textit{cond} \textbf{ then } \textit{stmt} \mid \textit{matched}\,,$$

$$\textit{matched} \to \textbf{if } \textit{cond} \textbf{ then } \textit{matched} \textbf{ else } \textit{stmt} \mid \textit{unconditionalStmt}\,.$$

Show that the grammar is still ambiguous.

30. Consider the grammar G given by:

$$S \to SS\sigma_1 \mid SS\sigma_2 \mid \ldots \mid SS\sigma_r \mid \sigma_{r+1}\,,$$

where $\sigma_1, \sigma_2, \ldots, \sigma_{r+1}$ are distinct terminals. Is the grammar unambiguous? If yes – prove your claim, if not – produce a word in $L(G)$ with two distinct derivation trees.

31. Eliminate the left recursion (direct and indirect) in the following grammars:

(a)

$$\begin{aligned} S &\to SaS \mid Ab \mid ab\,, \\ A &\to SA \mid AB \mid ba\,, \\ B &\to ABa \mid Bb \mid bab\,. \end{aligned}$$

(b)

$$\begin{aligned} S &\to SAB \mid ba\,, \\ A &\to SBS \mid ABA \mid a\,, \\ B &\to Sa \mid AS \mid bab\,. \end{aligned}$$

(c)

$$\begin{aligned} S &\to Sb \mid a\,, \\ A &\to AbS \mid ba\,, \\ B &\to SABA \mid AA \mid b\,. \end{aligned}$$

32. Transfer the following grammars into Chomsky Normal Form (omitting the word ε from the language if it accepted by the grammar):

(a)

$$S \to aS \mid bScSd \mid e \mid f\,.$$

(b)

$$S \to aSb \mid A\,,$$
$$A \to SS \mid \varepsilon\,.$$

(c)

$$S \to BCA \mid c\,,$$
$$A \to bAc \mid b\,.$$
$$B \to cBcc \mid \varepsilon\,,$$
$$C \to aCaa \mid \varepsilon\,.$$

33. Suppose G is a grammar "almost" in Chomsky Normal Form, namely all rules are either of the form $A \to a$ or of the form $A \to BC$ or of the form $A \to BCD$. We use the idea of the CYK algorithm on this grammar directly. What will the runtime of the algorithm be?

34. Let G be a grammar in Chomsky Normal Form.

(a) The way the CYK algorithm was presented in class, it just finds, for any $w \in T^*$, whether $S \overset{*}{\Longrightarrow} w$ or not. Change the algorithm so that, in case $S \overset{*}{\Longrightarrow} w$, it will construct a corresponding parse tree. By how much does this addition change the runtime?

(b) Change the CYK algorithm so that it will return not a boolean value, indicating whether $S \overset{*}{\Longrightarrow} w$ or not, but an integer value, indicating the number of parse trees producing w. Make sure your algorithm still works in polynomial time.

(c) Consider again the previous part. Can you construct all these parse trees in polynomial time?

(d) Design an algorithm that, for any strings $\alpha, \beta \in (N \cup T)^*$, decides whether $\alpha \overset{*}{\Longrightarrow} \beta$. What is the runtime of your algorithm?

35. Find the FIRST sets of all non-terminals and right-hand sides of all rules for the following grammars:

(a)

$$\begin{aligned} S &\to aS \mid AS \mid BAb\,, \\ A &\to Aab \mid AB \mid \varepsilon\,, \\ B &\to Aa \mid BbB \mid \varepsilon\,. \end{aligned}$$

(b)

$$\begin{aligned} S &\to ABCS \mid SS \mid aba\,, \\ A &\to ACB \mid cb \mid \varepsilon\,, \\ B &\to BCB \mid A \mid bc\,, \\ C &\to AS \mid c\,. \end{aligned}$$

(c)

$$\begin{aligned} S &\to SS \mid AB \mid c\,, \\ A &\to Aa \mid Aab \mid a \mid \varepsilon\,, \\ B &\to bC \mid bB \mid Sb \mid b\,, \\ C &\to cA \mid SC \mid c\,. \end{aligned}$$

36. Find the FOLLOW sets of all non-terminals for the grammars in Question 35.