shwetatiwari08@recabn.ac.in
shwetatiwari08aug@gmail.com

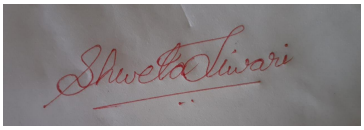
CD: UNIT-1 INTRODUCTION TO COMPILER DESIGN

FALL SEMESTER, YEAR (V/VI, 3rd)

Published: August, 2022

FALL SESSION: 2022-23

—



UNIT-1 CD
COMPILER DESIGN (KIT-052)

UNIT I – INTRODUCTION TO COMPILER DESIGN (PART-2)

PREPARED FOR
Engineering Students
All Engineering College

PREPARED BY
SHWETA TIWARI

Guest Faculty

Rajkiya Engineering College | Ambedkar Nagar, UP, India

Faculty Name: Miss. Shweta Tiwari, Subject: CD-COMPILER DESIGN (KIT-052), Year- 3rd Year, Semester- 5th/6th Sem, Branch- CS and IT, Session- Odd/Even Semester (2022-23)

CD-COMPILER DESIGN (KIT-052)

This **(Question Bank Part-1 Easy to Advanced Level (Belong your Syllabus))** only attempts to discover some questions with proper explanation that can be generated in "CD-COMPILER DESIGN" with the answer to all these questions. There can be some errors to these answers. If you find any errors then please do write to us.

CD-COMPILER DESIGN (KIT-052)

UNIT I INTRODUCTION TO COMPILER (PART-2)

LEXICAL ANALYSIS

2 MARKS

1. What is Lexical Analysis?

The first phase of the compiler is Lexical Analysis. This is also known as linear analysis in which the stream of characters making up the source program is read from left-to-right and grouped into tokens that are sequences of characters having a collective meaning.

2. What is a lexeme? Define a regular set.

- A Lexeme is a sequence of characters in the source program that is matched by the pattern for a token.
- A language denoted by a regular expression is said to be a regular set

3. What is a sentinel? What is its usage?

A Sentinel is a special character that cannot be part of the source program. Normally we use 'eof' as the sentinel. This is used for speeding-up the lexical analyzer.

4. What is a regular expression? State the rules, which define regular expression?

Regular expression is a method to describe regular language Rules:

1) ε-is a regular expression that denotes {ε} that is the set containing the empty string

2) If a is a symbol in ∑,then a is a regular expression that denotes {a}

3) Suppose r and s are regular expressions denoting the languages L(r) and L(s) Then,

a) (r)/(s) is a regular expression denoting L(r) U L(s).
b) (r)(s) is a regular expression denoting L(r)L(s)
c) (r)* is a regular expression denoting L(r)*.
d) (r) is a regular expression denoting L(r).

5. What are the Error-recovery actions in a lexical analyzer?

1. Deleting an extraneous character
2. Inserting a missing character
3. Replacing an incorrect character by a correct character

4. Transposing two adjacent characters

6. Construct Regular expression for the language

L= {w ε{a,b}/w ends in abb} Ans: {a/b}*abb.

7. What is a recognizer?

Recognizers are machines. These are the machines which accept the strings belonging to certain language. If the valid strings of such language are accepted by the machine then it is said that the corresponding language is accepted by that machine, otherwise it is rejected.

8. Differentiate compiler and interpreter.

Compiler produces a target program whereas an interpreter performs the operations implied by the source program.

9. Write short notes on buffer pairs.

Concerns with efficiency issues Used with a lookahead on the input. It is a specialized buffering technique used to reduce the overhead required to process an input character. Buffer is divided into two N-character halves. Use two pointers. Used at times when the lexical analyzer needs to look ahead several characters beyond the lexeme for a pattern before a match is announced.

10. Differentiate tokens, patterns, lexeme.

- Tokens- Sequence of characters that have a collective meaning.
- Patterns- There is a set of strings in the input for which the same token is produced as output. This set of strings is described by a rule called a pattern associated with the token
- Lexeme- A sequence of characters in the source program that is matched by the pattern for a token.

11. List the operations on languages.

- **Union -** L U M ={s | s is in L or s is in M}
- **Concatenation** – LM ={st | s is in L and t is in M}
- **Kleene Closure –** L* (zero or more concatenations of L)
- **Positive Closure –** L+ (one or more concatenations of L)

12. Write a regular expression for an identifier.

An identifier is defined as a letter followed by zero or more letters or digits.The regular expression for an identifier is given as **letter (letter | digit)***

13. Mention the various notational shorthands for representing regular expressions.

- One or more instances
- (+) Zero or one
- instance (?)
  Character classes ([abc] where a,b,c are alphabet symbols denotes the regular expressions a | b | c.)
- Non regular sets

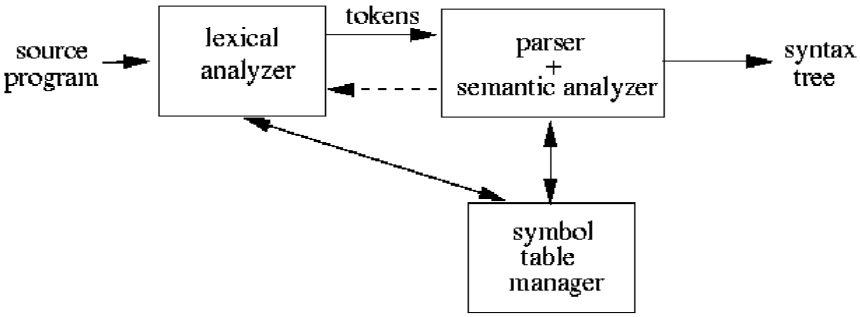
14. What is the function of a hierarchical analysis?

Hierarchical analysis is one in which the tokens are grouped hierarchically into nested collections with collective meaning. Also termed as Parsing.

15. What does semantic analysis do?

Semantic analysis is one in which certain checks are performed to ensure that components of a program fit together meaningfully. Mainly performs type checking.

16
MARKS



hat it is a syntactically

ı accordingly, in

ode generation.

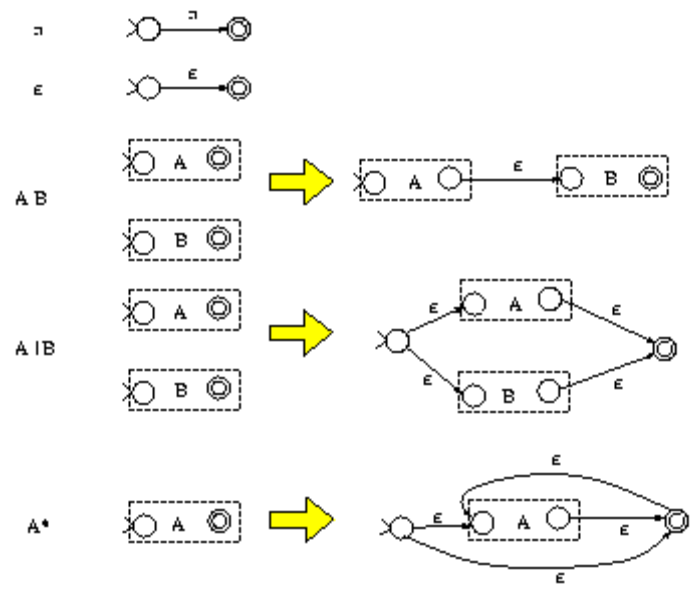
ministic Finite Automaton

The task of a scanner generator, such as JLex, is to generate the transition tables or to synthesize the scanner program given a scanner specification (in the form of a set of REs). So it needs to convert REs into a single DFA. This is accomplished in two steps: first it converts REs into a non-deterministic finite automaton (NFA) and then it converts the NFA into a DFA.

An NFA is similar to a DFA but it also permits multiple transitions over the same character and transitions over ε. In the case of multiple transitions from a state over the same character, when we are at this state and we read this character, we have more than one choice; the NFA succeeds if at least one of these choices succeeds. The transition doesn't consume any input characters, so you may jump to another state for free.

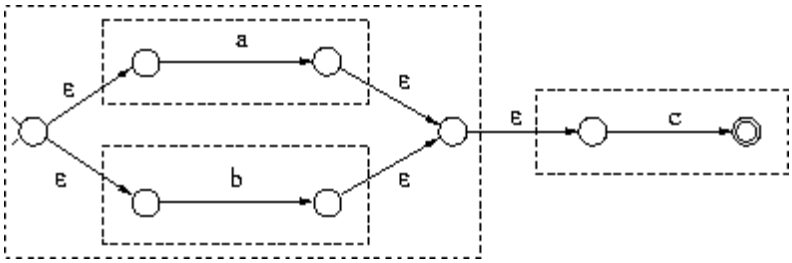
Clearly DFAs are a subset of NFAs. But it turns out that DFAs and NFAs have the same expressive power. The problem is that when converting a NFA to a DFA we may get an exponential blowup in the number of states.

We will first learn how to convert a RE into a NFA. This is the easy part. There are only 5 rules, one for each type of RE:



As it can be shown inductively, the above rules construct NFAs with only one final state. For example, the third rule indicates that, to construct the NFA for the RE *AB*, we construct the NFAs for *A* and *B*, which are represented as two boxes with one start state and one final state for each box. Then the NFA for *AB* is constructed by connecting the final state of *A* to the start state of *B* using an empty transition.

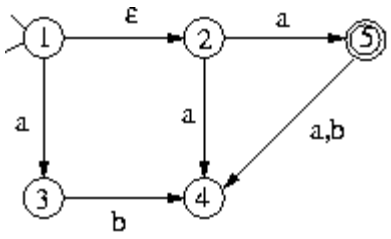
For example, the RE (*a*| *b*)*c* is mapped to the following NFA:

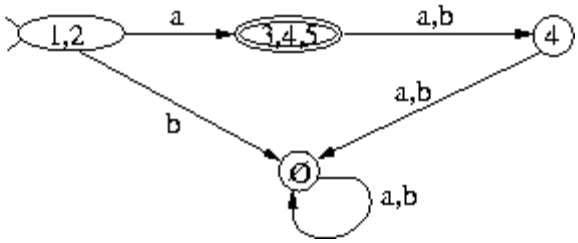


The next step is to convert a NFA to a DFA (called *subset construction*). Suppose that you assign a number to each NFA state. The DFA states generated by subset construction have sets of numbers, instead of just one number. For example, a DFA state may have been assigned the set
{5, 6, 8}. This indicates that arriving at the state labeled {5, 6, 8} in the DFA is the same as arriving at the state 5, the state 6, or the state 8 in the NFA when parsing the same input. (Recall that a particular input sequence when parsed by a DFA, leads to a unique state, while when parsed by a NFA it may lead to multiple states.)

First we need to handle transitions that lead to other states for free (without consuming any input). These are the ε transitions. We define the *closure* of a NFA node as the set

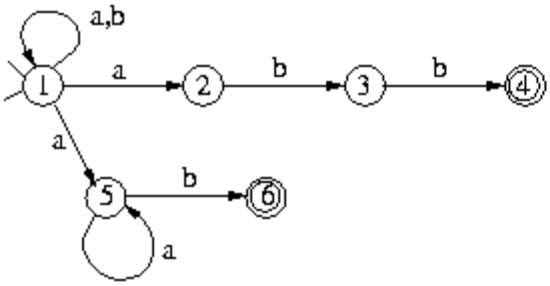
of all the nodes reachable by this node using ε zero, one, or more transitions. For example, The closure of node 1 in the left figure below

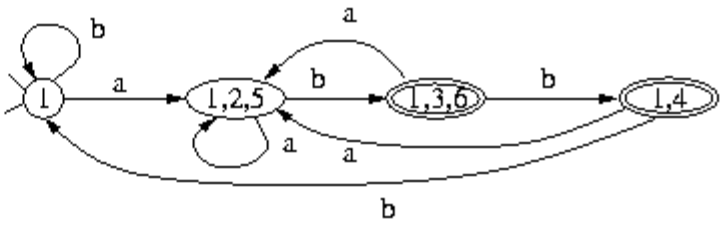




is the set {1, 2}. The start state of the constructed DFA is labeled by the closure of the NFA start state. For every DFA state labeled by some set $\{s_1,..., s_n\}$ and for every character *c* in the language alphabet, you find all the states reachable by s_1, s_2, ..., or s_n using *c* arrows and you union together the closures of these nodes. If this set is not the label of any other node in the DFA constructed so far, you create a new DFA node with this label. For example, node {1, 2} in the DFA above has an arrow to a {3, 4, 5} for the character *a* since the NFA node 3 can be reached by 1 on *a* and nodes 4 and 5 can be reached by 2. The *b* arrow for node {1, 2} goes to the error node which is associated with an empty set of NFA nodes.

The following NFA recognizes $(a|\ b)^*(abb\ |\ a^+b)$, even though it wasn't constructed with the above RE-to-NFA rules. It has the following DFA:





**********THE END**********