California State University, Fullerton
Dept of Computer Science

# CPSC 323 Compilers & Languages (Spring 2024)

## ASSIGNMENT 1

Please answer the questions appropriately, and if you believe a diagram is required, draw one.

If you want to solve the problems, do so while documenting your steps.

NOTE: Please avoid writing single-word responses or just the single solution to a problem. Please submit a PDF document.

---

1. Explain the basic diference between Deterministic Finite Automata (DFA) and Non-Deterministic Finite Automata (NFA)?

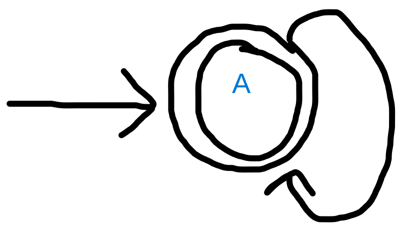
A DFA is when no state has more the one outgoing edge with the same label, and all FA were DFA, for each symbolic representation of the alphabet, there is only one state transition, DFA can't use empty string transition, can only be understood as one machine, the next possible state is distinctly set, more difficult to construct , it rejects the string in case it terminates in a state that's different from the accepting state, it takes less time for executing an input string ,it requires more space, dead configuration is not allowed, backtracking is allowed in DFA, conversion of regular expression to DFA is hard, epsilon move is not allowed in DFA , only allows only one move for single input alphabet whereas an NFA is when states may have more than one outgoing edge with the same label, edges may be labeled with a epsilon, the empty string, the automation can make an epsilon transition without consuming the current input character. There is no need to specify how the NFA react to some symbol, can use empty string transition, can be understood as multiple little machines computing something at the same time, each pair of state and input symbol can have many possible next states, easier to construct, rejects the string of all branches dying or refusing the string, it takes more time for executing an input string, not all NFA are DFA, dead configuration is allowed, backtracking is not always possible, epsilon move is allowed, you can move more than once for a single input alphabet

2. What is a Regular Expression? Construct an FA for the given RE's. (a+b)*

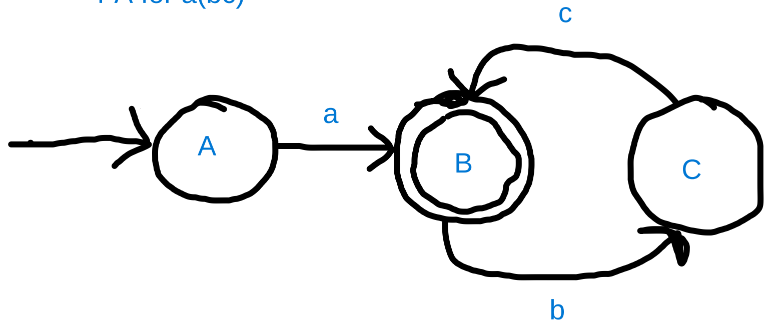
   a(bc)*

Regular expressions are a means to descirbe langauges, the class of languages that can be described by regular expressions coincides with the class of regular languages. REs are a more compact way to define a language that can be accepted by an FA. RE allows to express the token. Regular expressions are asuitable specification and a compact way to define a languagee

FA for (a+b)*

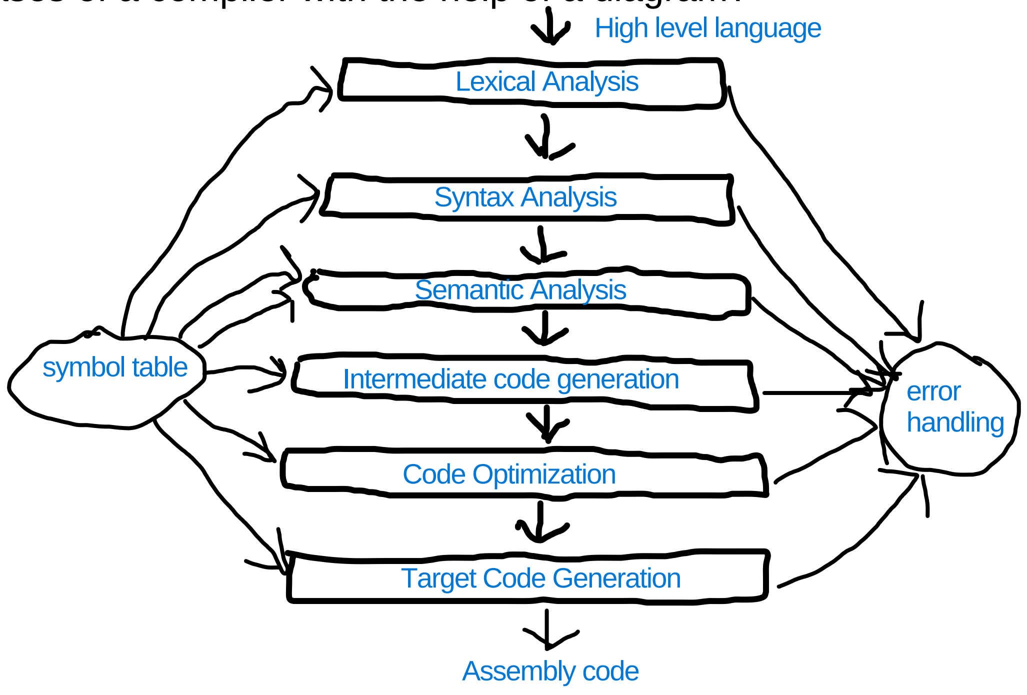


FA for a(bc)*



3. Describe the phases of a compiler with the help of a diagram?

The phases of a compiler in order are Lexical analysis, Syntax analysis, Semantic analysis, intermediate code generation, code optimization, and target code generation . The lexical analysis scans the source code as a stream of characters and converts it into lexemes, it also breaks up the source code into tokens, the syntax analysis checks and sees if the lexemes are tokens are syntactically correct, semantic analysis checks to see whether the code is semantically correct, intermediate code generation generates a intermediate representation of the source code so it can be translated into machine code, optimization applies to various optimization techniques to the intermediate code to improve the performance of code that is generated from the machine, code generation is the final phase in which it takes the optimized intermediate code, and generates the actual machine code that can be executed by the target hardware



4. What is the output of a lexical analyzer? List the tokens and lexemes in the following C statement –
printf("j = %m, &j = %k", j, &j);

The output of a lexical analyzer are tokens or the stream of tokens, which includes keywords, identifiers, literals, operators, and symbols

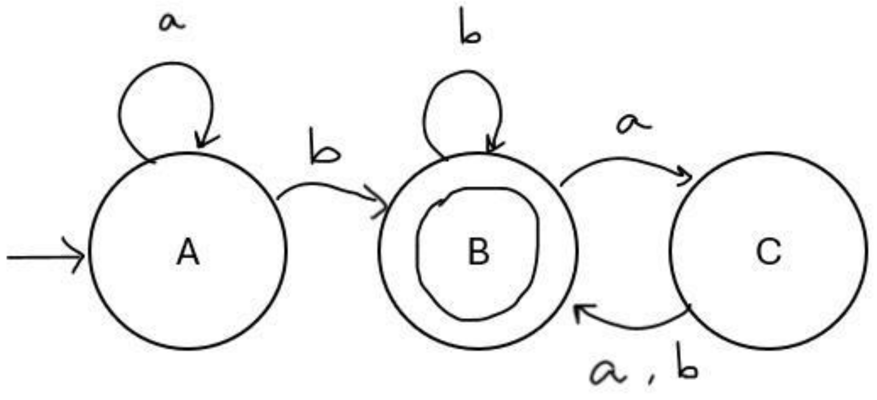
| Tokens: | Lexemes: |
|---|---|
| 1. Keyword | 1. printf |
| 2. Symbol | 2. ( |
| 3. Literal | 3."j = %m, &j = %k" |
| 4. Symbol | 4. , |
| 5. Identifier | 5. j |
| 6. Symbol | 6. , |
| 7. Operator | 7. & |
| 8. Identifier | 8. j |
| 9. Symbol | 9. ) |
| 10. Symbol | 10. ; |

5. Does the DFA given below accept the string bbab? Explain why? Does it also accept the string aaba?

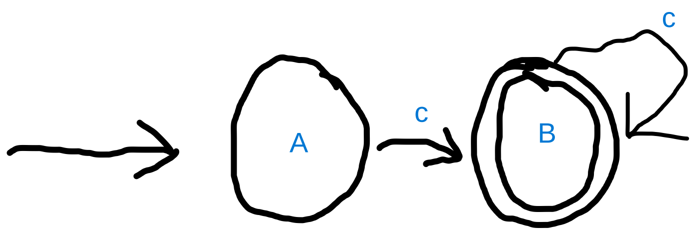
The DFA bellow accepts the string "bbab", because the first input is 'b' and since it is 'b' it goes to state B , the second input is 'b' and since it is 'b', it goes back to state B, the third input is 'a' and since it is 'a', it goes to state C, the final input is 'b' and since it is 'b', it goes back to state B, and since state B is a accepting and final state, thus it accepts the string "bbab"

---

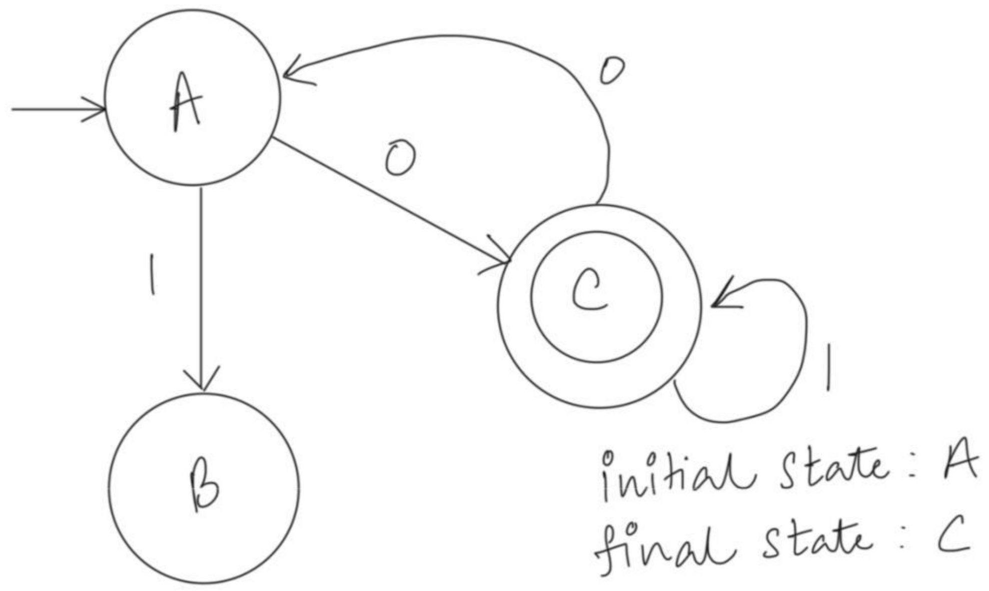
The DFA doesn't accept "abba", because the first input is 'a' and since it is 'a' it goes back to state A, the second input is 'a' and since it is 'a' it goes back to state A again, the third input is 'b' and since it is 'b' it goes to state B,  the final input is 'a' and since it is 'a' it goes to state C, and since state C is not the final or accepting state, thus it doesn't accept "abba"



6. Construct a DFA that accepts all strings from the language given below - L = {c, cc, ccc, cccc, . . .}



7. How many tuples are there in Finite State Automata? What are they? What input strings will be accepted by the below given state transition diagram?

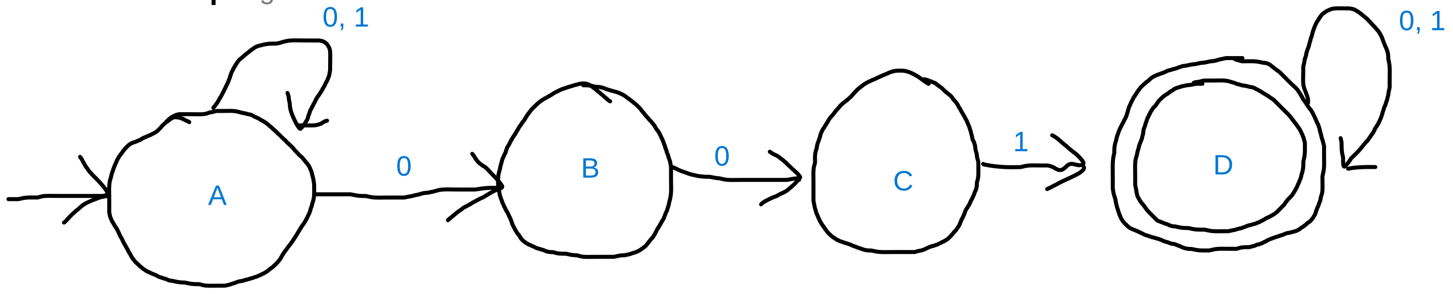


FSA = ({1,0}, {A,B,C}, A, {C}, N) sigma = {1,0} q0 = A

There are a total of 5 tuples, and the five tuples are the finite set of input symbols which includes {1,0}, the finite set of states which is {A,B,C}, and q0, which is the starting state, which is A, the accepting state is {C}, and the final tuple is a state transition function

The input strings that are accepted are "0", "01","000", "0001"

8. Design an NFA with ∑ = {0, 1} which accepts strings containing double '0' followed by single '1'. Also draw the State Transition Table.



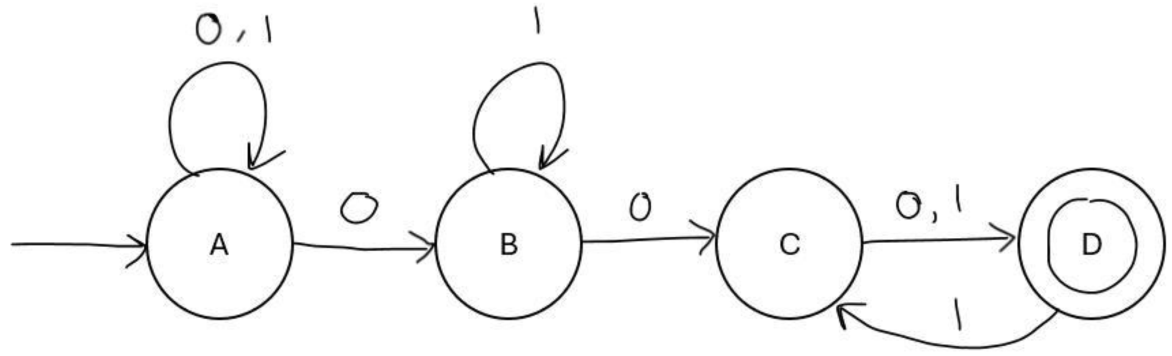
State transition table

| | 0 | 1 |
|---|---|---|
| A | {A,B} | A |
| B | C | ϕ |
| C | ϕ | D |
| D | D | D |

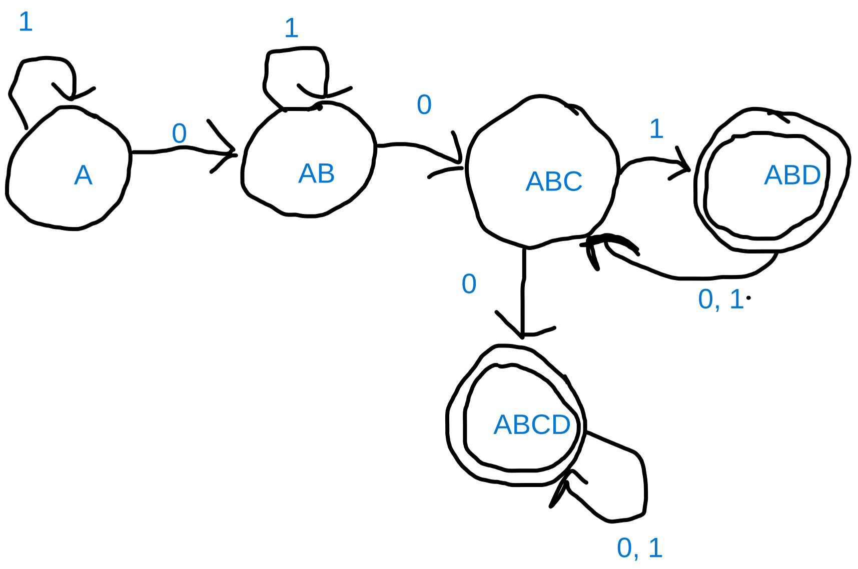
NFA state transition table

| | 0 | 1 |
|---|---|---|
| A | {A,B} | A |
| B | C | B |
| C | D | D |
| D | ∅ | C |

9. Convert the following NFA to a DFA –

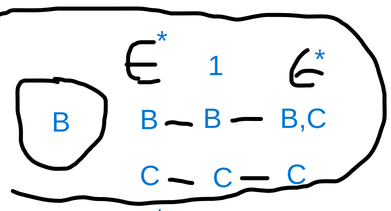


DFA state transition table



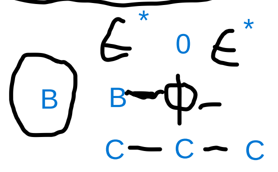
| | 0 | 1 |
|---|---|---|
| A | AB | A |
| AB | ABC | AB |
| ABC | ABCD | ABD |
| ABD | ABC | ABC |
| ABCD | ABCD | ABCD |

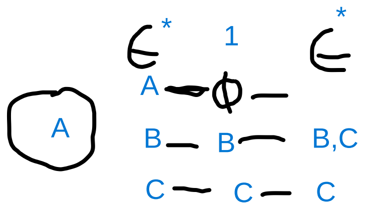
Question 10 work



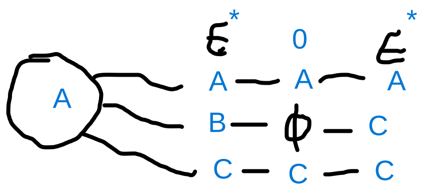
e-NFA transition table

|   | 0 | 1 | $\epsilon$ |
|---|---|---|---|
| A | A | $\phi$ | B |
| B | $\phi$ | B | C |
| C | C | C | $\phi$ |

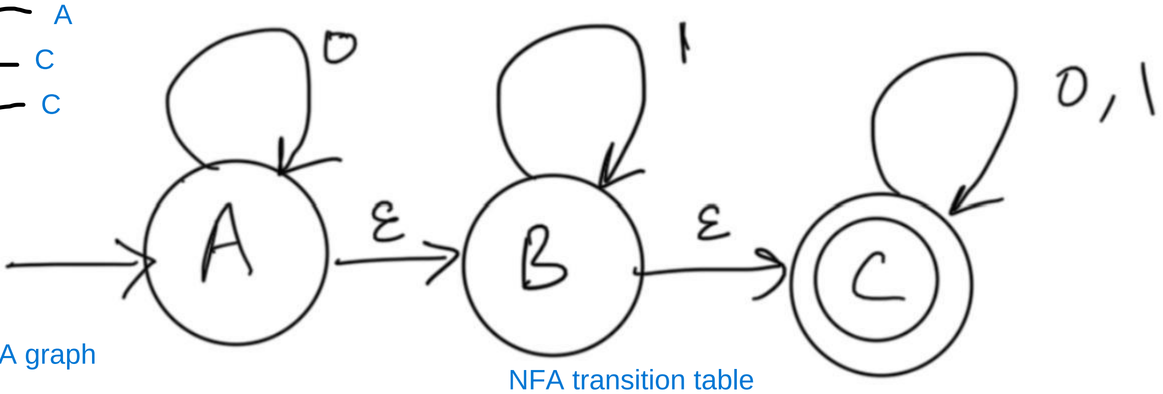




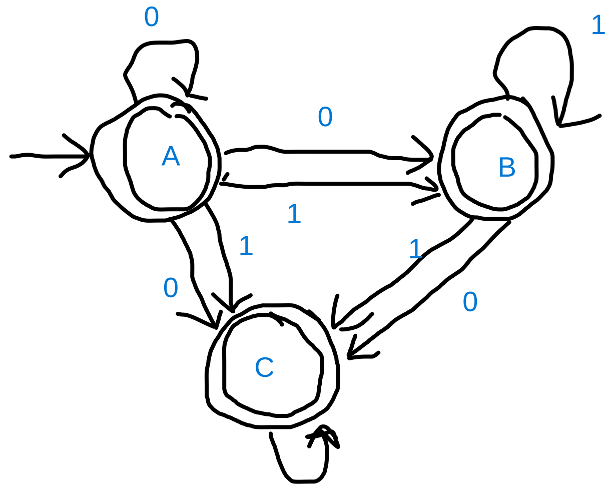
10. Convert the following e-NFA to NFA -



e-NDFA graph



NFA graph



NFA transition table

|   | 0 | 1 |
|---|---|---|
| A | {A,B,C} | {B,C} |
| B | {C} | {B,C} |
| C | {C} | {C} |