CST8152 – Compilers
Article #9
From Regular Expression to NFA to DFA

It is proven (Kleene's Theorem) that RE and FA are equivalent language definition methods. Based on this theoretical result practical algorithms have been developed enabling us actually to construct FA's from RE's and simulate the FA with a computer program using Transition Tables. In following this progression an NFA is constructed first from a **regular expression,** then the NFA is reconstructed to a DFA, and finally a Transition Table is built. The **Thompson's Construction Algorithm** is one of the algorithms that can be used to build a Nondeterministic Finite Automaton (**NFA**) from RE, and **Subset construction Algorithm** can be applied to convert the NFA into a Deterministic Finite Automaton (**DFA**). The last step is to generate a transition table.
(Text § 3.6, 3.7)

We need a finite state machine that is a deterministic finite automaton (DFA) so that each state has one unique edge for an input alphabet element. So that for code generation there is no ambiguity. But a **nondeterministic** finite automaton (NFA) with more than one edge for an input alphabet element is easier to construct using a general algorithm - Thompson's construction. Then following a standard procedure, we convert the NFA to a DFA for coding.

1. Regular expression

Consider the regular expression r = (a|b)*abb, that matches
{abb, aabb, babb, aaabb, bbabb, ababb, aababb,……}

To construct a NFA from this, use **Thompson's construction**.
This method constructs a regular expression from its components using ε-transitions. The ε transitions act as "glue or mortar" for the subcomponent NFA's. An ε-transition adds nothing since concatenation with the empty string leaves a regular expression unchanged (concatenation with ε is the identity operation).

Step 1.
Parse the regular expression into its subexpressions involving alphabet symbols `a` and `b` and ε:
`ε, a, b, a|b, ()*, ab, abb`

These describe

a. a regular expression for single characters `ε, a, b`
b. alternation between `a` and `b` representing the union of the sets: `L(a) U L(b)`

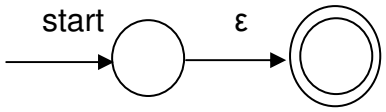
c. Kleene star ()*
d. concatenation of `a` and `b`: `ab`, and also `abb`

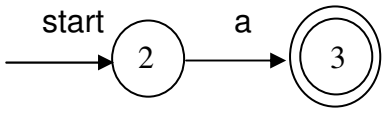
Subexpressions of these kinds have their own **Nondeterministic Finite Automata** from which the overall NFA is constructed. Each component NFA has its own start and end accepting states.

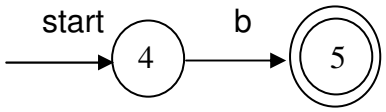
A **Nondeterministic Finite Automata** (NFA) has a transition diagram with possibly more than one edge for a symbol (character of the alphabet) that has a start state and an accepting state. The NFA definitely provides an accepting state for the symbol.

Take these NFA's in turn:

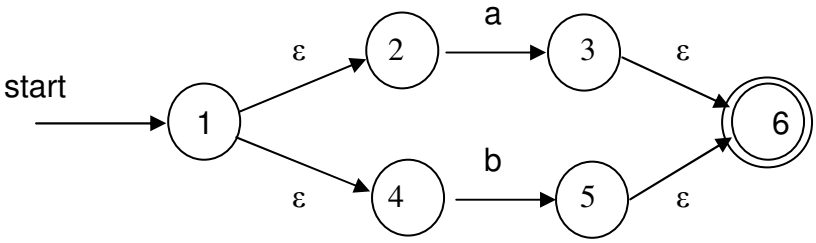
a. the NFA's for single character regular expressions `ε, a, b`



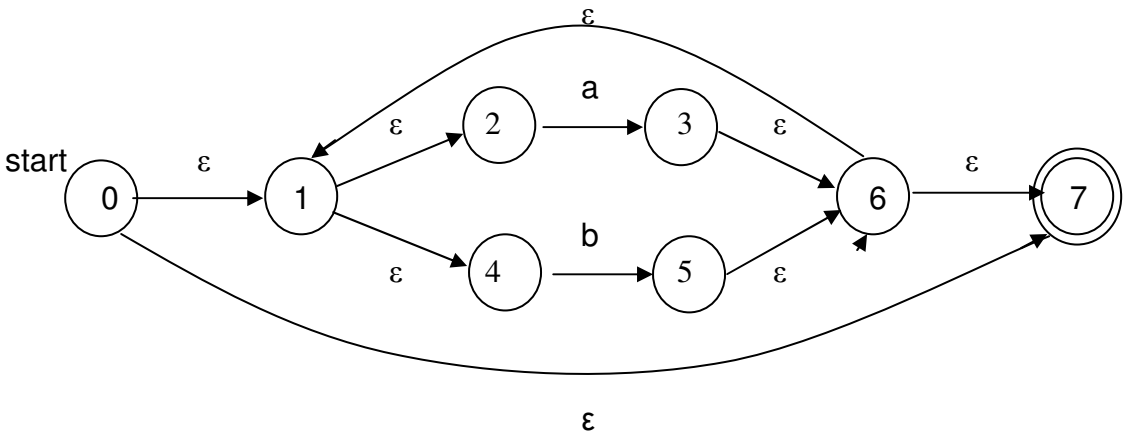




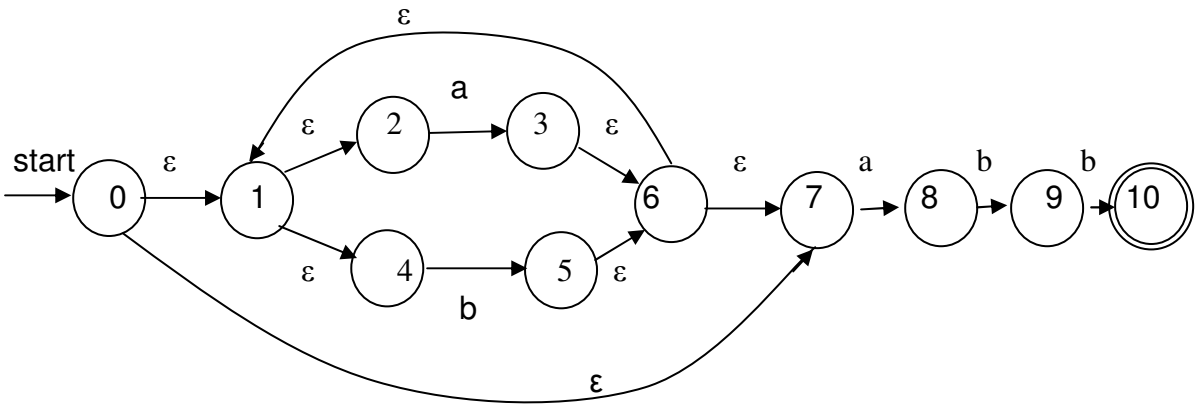
b. the NFA for the union of a and b: a|b is constructed from the individual NFA's using the ε NFA as "glue". Remove the individual accepting states and replace with the overall accepting state



c. Kleene star on (a|b)*. The NFA accepts ε in addition to (a|b)*



d. concatenate with abb



This is the complete NFA. It describes the regular expression (a|b)*abb.
The problem is that it is not suitable as the basis of a DFA transition table since there are multiple ε edges leaving many states (0, 1, 6).

Converting the NFA into a DFA

A Deterministic Finite Automaton (DFA) has at most one edge from each state for a given symbol and is a suitable basis for a transition table. We need to eliminate the ε-transitions by subset construction.

Definitions
Consider a single state s. Consider a set of states T

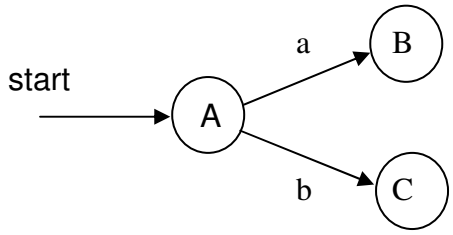
Operation	Description
ε-closure(s)	Set of NFA states reachable from NFA state **s** on ε-transitions alone
ε-closure(T)	Set of NFA states reachable from set of states **T** on ε-transitions alone
move(T,a)	Set of states to which there is a transition on input symbol **a** from some NFA state in **T**

We have as input the set of N states. We generate as output a set of D states in a DFA.
Theoretically an NFA with **n** states can generate a DFA with $\mathbf{2^n}$ states.

Start the Conversion

1. Begin with the start state 0 and calculate ε-closure(0).
    a. the set of states reachable by ε-transitions which includes 0 itself is { 0,1,2,4,7}. This defines a new state A in the DFA
    A = {0,1,2,4,7}

2. We must now find the states that A connects to. There are two symbols in the language (**a, b**) so in the DFA we expect only two edges: from A on **a** and from A on **b**. Call these states B and C:



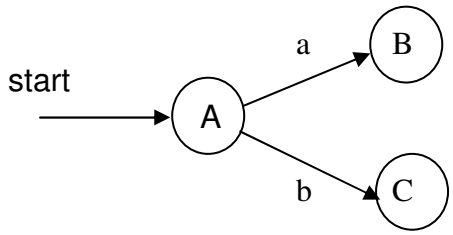
We find B and C in the following way:

Find the state B that has an edge on a from A

a. start with A{0,1,2,4,7}. Find which states in A have states reachable by **a** transitions. This set is called move(A,a) The set is {3,8}:
   move(A,a) = {3,8}
b. now do an ε-closure on **move(A,a)**. Find all the states in **move(A,a)** which are reachable with ε-transitions. We have 3 and 8 to consider. Starting with 3 we can get to 3 and 6 and from 6 to 1 and 7, and from 1 to 2 and 4. Starting with 8 we can get to 8 only. So the complete set is {1,2,3,4,6,7,8}. So
   ε-closure(move(A,a)) = **B = {1,2,3,4,6,7,8}**
   This defines the new state B that has an edge on **a** from A

Find the state C that has an edge on b from A

c. start with A{0,1,2,4,7}. Find which states in A have states reachable by **b** transitions. This set is called move(A,b) The set is {5}:
   move(A,b) = {5}
d. now do an ε-closure on **move(A,b)**. Find all the states in **move(A,b)** which are reachable with ε-transitions. We have only state 5 to consider. From 5 we can get to 5, 6, 7, 1, 2, 4. So the complete set is {1,2,4,5,6,7}. So
   ε-closure(move(A,a)) = **C = {1,2,4,5,6,7}**
   This defines the new state C that has an edge on **b** from A



A={0,1,2,4,7}
B={1,2,3,4,6,7,8}
C={1,2,4,5,6,7}

Now that we have B and C we can move on to find the states that have **a** and **b** transitions from B and C.

Find the state that has an edge on a from B

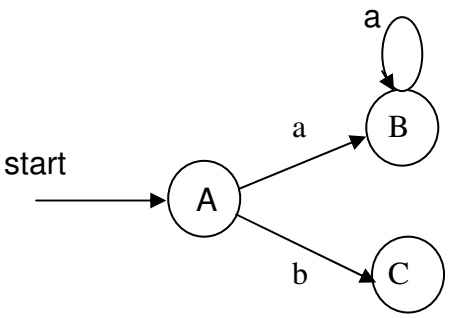
e. start with B{1,2,3,4,6,7,8}. Find which states in B have states reachable by **a** transitions. This set is called move(B,a) The set is {3,8}:

move(B,a) = {3,8}

f. now do an ε-closure on **move(B,a)**. Find all the states in **move(B,a)** which are reachable with ε-transitions. We have 3 and 8 to consider. Starting with 3 we can get to 3 and 6 and from 6 to 1 and 7, and from 1 to 2 and 4. Starting with 8 we can get to 8 only. So the complete set is {1,2,3,4,6,7,8}. So

ε-closure(move(A,a)) = **{1,2,3,4,6,7,8}**

which is the same as the state B itself. In other words, we have a repeating edge to B:



A={0,1,2,4,7}
B={1,2,3,4,6,7,8}
C={1,2,4,5,6,7}

Find the state D that has an edge on b from B

g. start with B{1,2,3,4,6,7,8}. Find which states in B have states reachable by **b** transitions. This set is called move(B,b) The set is {5,9}:

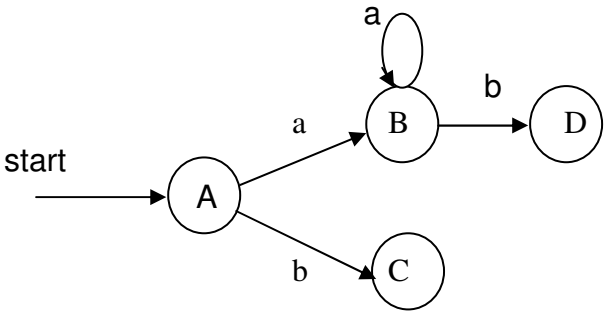
move(B,b) = {5,9}

h. now do an ε-closure on **move(B,b)**. Find all the states in **move(B,b)** which are reachable with ε-transitions. From 5 we can get to 5, 6, 7, 1, 2, 4. From 9 we get to 9 itself. So the complete set is {1,2,4,5,6,7,9}}. So

ε-closure(move(B,a)) = **D = {1,2,4,5,6,7,9}**

This defines the new state D that has an edge on **b** from B

A={0,1,2,4,7},B={1,2,3,4,6,7,8}, C={1,2,4,5,6,7}, D{1,2,4,5,6,7,9}

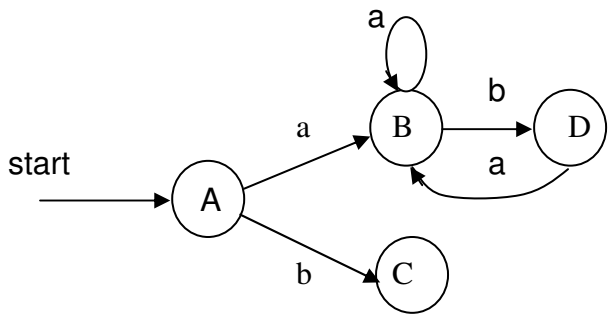


Find the state that has an edge on a from D

i. start with D{1,2,4,5,6,7,9}. Find which states in D have states reachable by **a** transitions. This set is called move(D,a) The set is {3,8}:
move(D,a) = {3,8}

j. now do an ε-closure on **move(D,a)**. Find all the states in **move(B,a)** which are reachable with ε-transitions. We have 3 and 8 to consider. Starting with 3 we can get to 3 and 6 and from 6 to 1 and 7, and from 1 to 2 and 4. Starting with 8 we can get to 8 only. So the complete set is {1,2,3,4,6,7,8}. So
ε-closure(move(D,a)) = **{1,2,3,4,6,7,8} =B**
This is a return edge to B:

A={0,1,2,4,7},B={1,2,3,4,6,7,8}, C={1,2,4,5,6,7}, D{1,2,4,5,6,7,9}



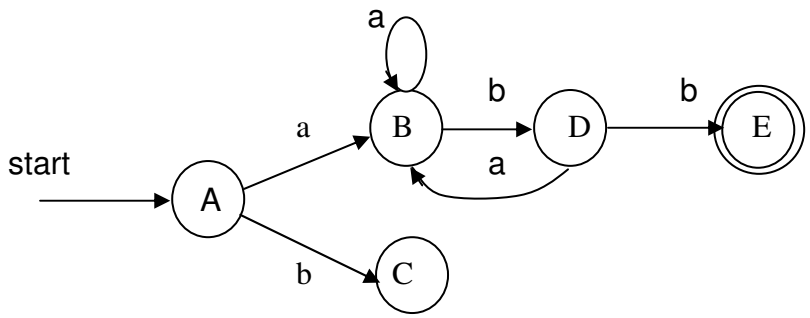
Find the state E that has an edge on b from D

k. start with D{1,2,4,5,6,7,9}. Find which states in D have states reachable by **b** transitions. This set is called move(B,b) The set is {5,10}:
move(D,b) = {5,10}

l. now do an ε-closure on **move(D,b)**. Find all the states in **move(D,b)** which are reachable with ε-transitions. From 5 we can get to 5, 6, 7, 1, 2, 4. From 10 we get to 10 itself. So the complete set is {1,2,4,5,6,7,10}}. So
ε-closure(move(D,b) = **E = {1,2,4,5,6,7,10}**
This defines the new state E that has an edge on **b** from D.

Since it contains an accepting state, it is also an accepting state.

A={0,1,2,4,7},B={1,2,3,4,6,7,8}, C={1,2,4,5,6,7}, D={1,2,4,5,6,7,9}, E={1,2,4,5,6,7,10}



We should now examine state C

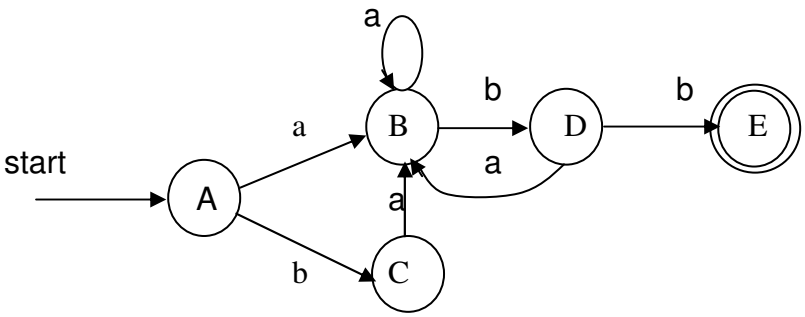
Find the state that has an edge on a from C

m. start with C{1,2,4,5,6,7}. Find which states in C have states reachable by **a** transitions. This set is called move(C,a) The set is {3,8}:

move(C,a) = {3,8}

we have seen this before. It's the state B

A={0,1,2,4,7},B={1,2,3,4,6,7,8}, C={1,2,4,5,6,7}, D={1,2,4,5,6,7,9}, E={1,2,4,5,6,7,10}



Find the state that has an edge on b from C

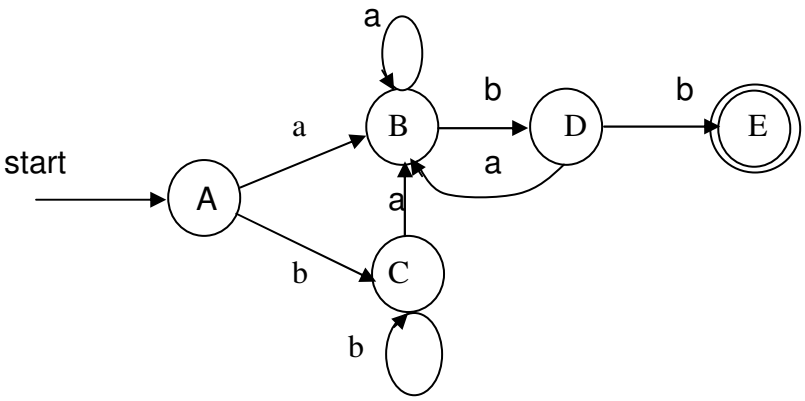
n. start with C{1,2,4,5,6,7}. Find which states in C have states reachable by **b** transitions. This set is called move(C,b) The set is {5}:

move(C,b) = {5}

o. now do an ε-closure on **move(C,b)**. Find all the states in **move(C,b)** which are reachable with ε-transitions. From 5 we can get to 5,6,7,1,2,4. which is C itself So

ε-closure(move(C,b)) = **C**

This defines a loop on C



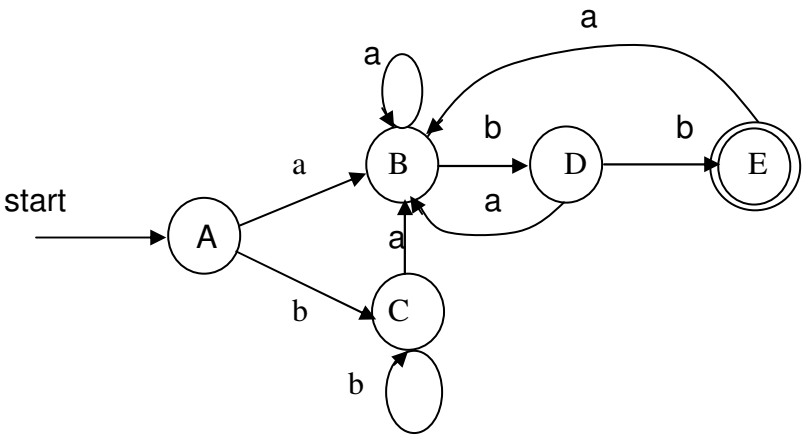
Finally we need to look at E. Although this is an accepting state, the regular expression allows us to repeat adding in more a's and b's as long as we return to the accepting E state finally. So

Find the state that has an edge on a from E

p. start with E{1,2,4,5,6,7,10}. Find which states in E have states reachable by **a** transitions. This set is called move(E,a) The set is {3,8}:

move(E,a) = {3,8}

We saw this before, it's B. So

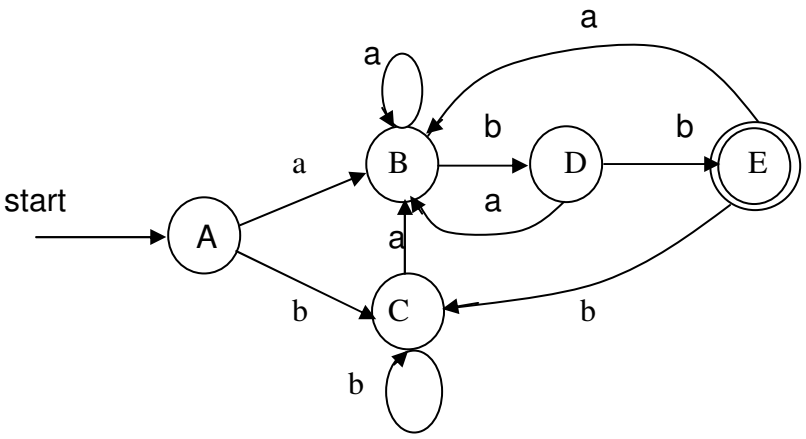


Find the state that has an edge on b from E

q. start with E{1,2,4,5,6,7,10}. Find which states in E have states reachable by **b** transitions. This set is called move(E,b) The set is {5}:

move(A,b) = {5}

We've seen this before. It's C. Finally



That's it ! There is only one edge from each state for a given input character. It's a DFA. Disregard the fact that each of these states is actually a group of NFA states. We can regard them as single states in the DFA. In fact it also requires **other** as an edge beyond E leading to the ultimate accepting state. Also the DFA is not yet optimized (there can be less states).

However, we can make the transition table so far. Here it is:

State	Input **a**	Input **b**
A	B	C
B	B	D
C	B	C
D	B	E
E	B	C

NFA vs. DFA (Time-Space Tradeoffs

The table below summarizes the worst-case for determining whether an input string ***x*** belongs to the language denoted by a regular expression ***r*** using recognizers constructed from NFA and DFA.

Automaton	**Space**	**Time**
NFA	$\boldsymbol{O}(\|r\|)$	$O(\|r\| \times \|x\|)$
DFA	$\boldsymbol{O}(2^{\|r\|})$	$O(\|x\|)$

| ***r*** | is the length of ***r***, and | ***x*** | is the length of ***x.***

Example:
For the regular expression

(a | b)*a(a | b)…(a | b) , where there are n-1 (a | b)'s at the end

There is no DFA with fewer than 2^n states.