NFA-to-DFA conversion

Slides from

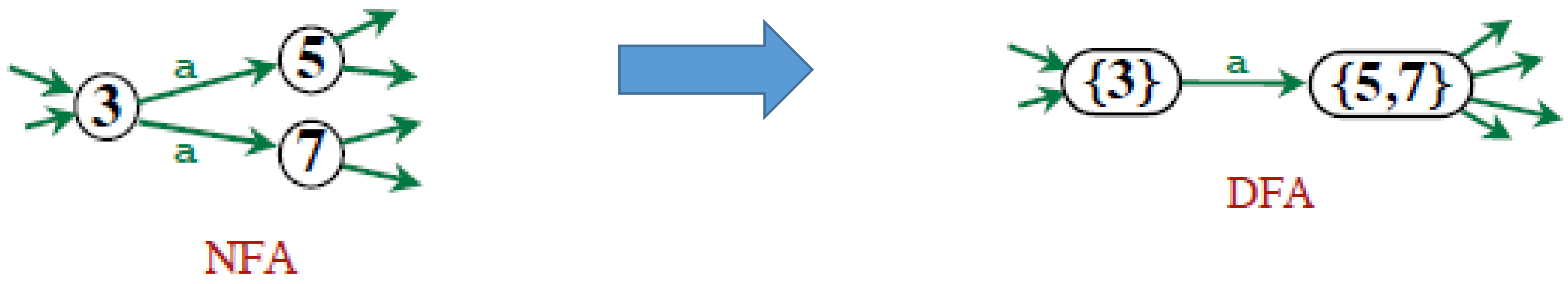
http://web.cecs.pdx.edu/~harry/compilers/slides/LexicalPart3.pdf

Two ways to deal with an NFA

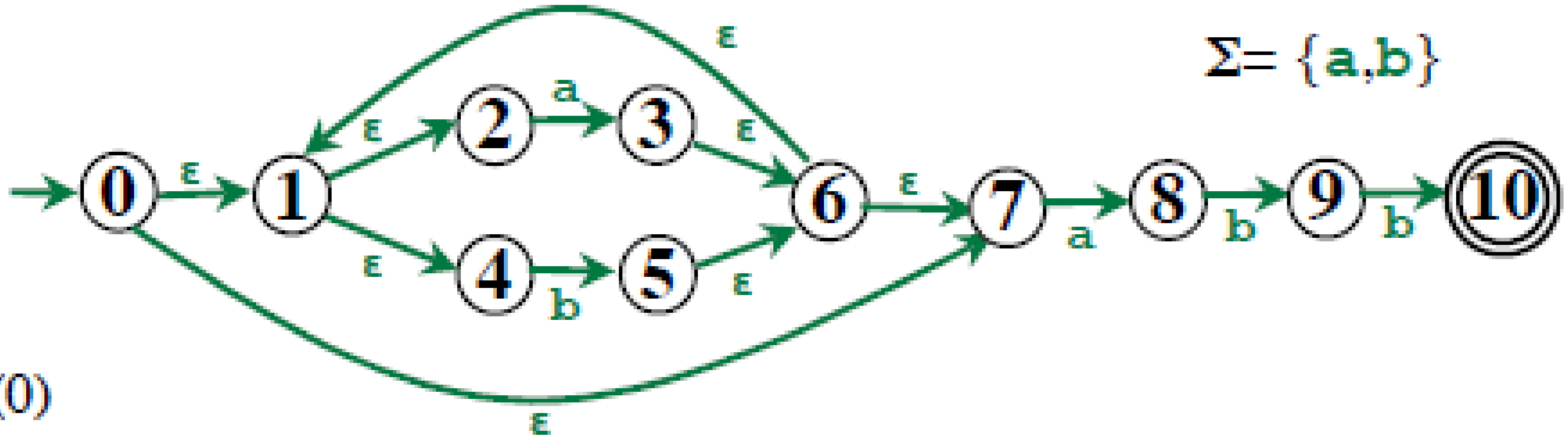
- Convert the NFA to an equivalent DFA first
- Use the NFA directly

Converting an NFA to a DFA

- Input: an NFA
- Output: a DFA, which is equivalent
- Idea: Each state in the DFA corresponds to a set of NFA states



Example



$\Sigma = \{a,b\}$

Start state:

ε-Closure (0)

= {0, 1, 2, 4, 7} = A

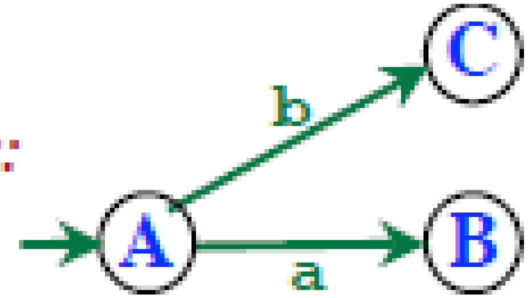
$\text{Move}_{\text{DFA}}(A,a)$

$= \varepsilon\text{-Closure}(\text{Move}_{\text{NFA}}(A,a))$

$= \varepsilon\text{-Closure}(\{3,8\})$

$= \{1,2,3,4,6,7,8\} = B$

So far:



$\text{Move}_{\text{DFA}}(A,b)$

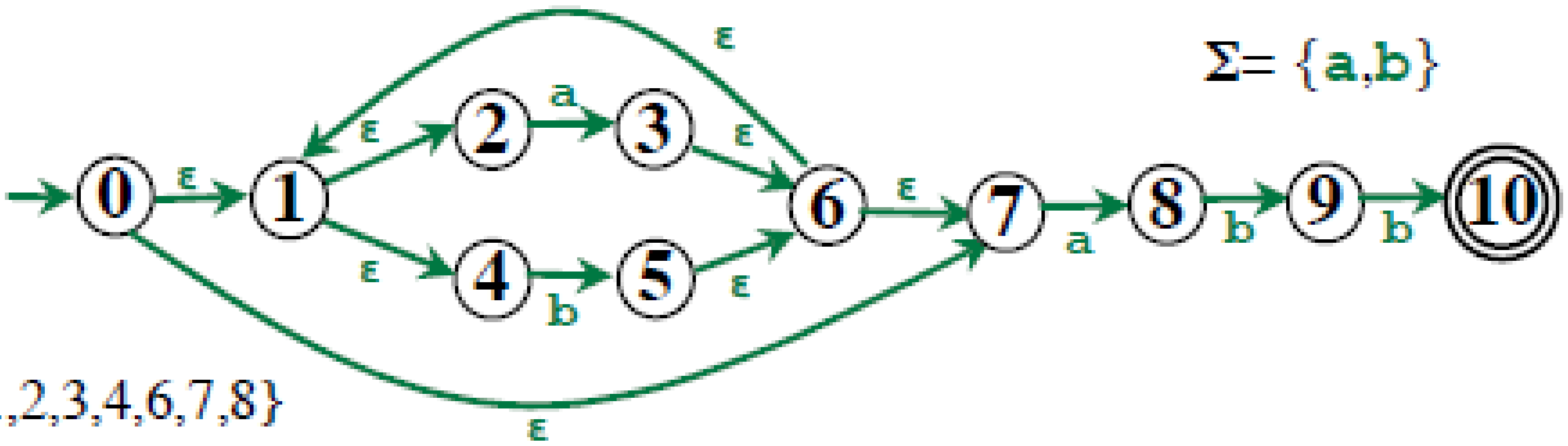
$= \varepsilon\text{-Closure}(\text{Move}_{\text{NFA}}(A,b))$

$= \varepsilon\text{-Closure}(\{5\})$

$= \{1,2,4,5,6,7\} = C$

A is now done; mark it!
B and C are unmarked.
Let's do B next...

Example

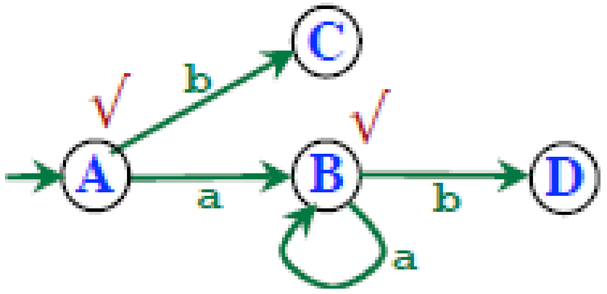


$\Sigma = \{a,b\}$

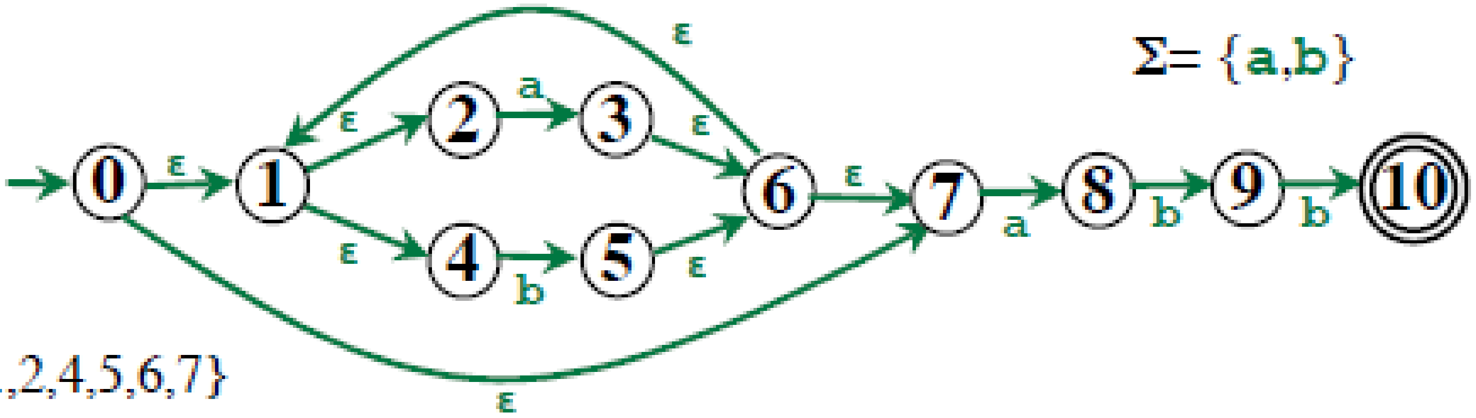
Process B = {1,2,3,4,6,7,8}

$Move_{DFA}(B,a)$
$= \varepsilon\text{-Closure}(Move_{NFA}(B,a))$
$= \varepsilon\text{-Closure}(\{3,8\})$
$= \{1,2,3,4,6,7,8\} = B$

$Move_{DFA}(B,b)$
$= \varepsilon\text{-Closure}(Move_{NFA}(B,b))$
$= \varepsilon\text{-Closure}(\{5,9\})$
$= \{1,2,4,5,6,7,9\} = D$



Example



$\Sigma = \{a,b\}$

Process C = {1,2,4,5,6,7}

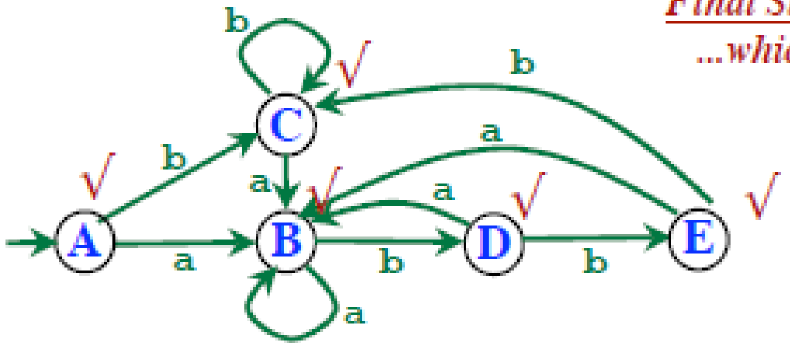
$Move_{DFA}(C,a) = \{1,2,3,4,6,7,8\} = B$
$Move_{DFA}(C,b) = \{1,2,4,5,6,7\} = C$

Process E = {1,2,4,5,6,7,10}

$Move_{DFA}(E,a) = \{1,2,3,4,6,7,8\} = B$
$Move_{DFA}(E,b) = \{1,2,4,5,6,7\} = C$

Process D = {1,2,4,5,6,7,9}

$Move_{DFA}(D,a) = \{1,2,3,4,6,7,8\} = B$
$Move_{DFA}(D,b) = \{1,2,4,5,6,7,10\} = E$



Final States in DFA?
...which state(s) contain 10?

Algorithm: Convert NFA to DFA

```
S_DFA = {}
Add ε-Closure(s_0) to S_DFA as the start state
Set the only state in S_DFA to "unmarked"
while S_DFA contains an unmarked state do
   Let T be that unmarked state
   Mark T
   for each a in Σ do
      S = ε-Closure(Move_NFA(T,a))
      if S is not in S_DFA already then
         Add S to S_DFA (as an "unmarked" state)
      endIf
      Set Move_DFA(T,a) to S


   endFor
endWhile
for each S in S_DFA do
   if any s∈S is a final state in the NFA then
      Mark S an a final state in the DFA
   endIf
endFor
```

A set of NFA states

Everywhere you could possibly get to on an a

i.e, add an edge to the DFA…

$$\underset{\mathbf{T}}{\{t_1,t_2,\ldots\}} \xrightarrow{a} \underset{\mathbf{S}}{\{s_1,s_2,\ldots\}}$$

Use NFA directly

- Your code will keep track of “current search states”. Once you reach the end of the input symbol sequence, check one of the current search states is a final state.
- Note that you do not need to enumerate all possible paths explicitly. You just need to keep track of multiple current search states.
- The method is explained in Section 2.2.5 in Jurasfsky & Martin (2^{nd} edition).