Simulating a DFA

function Match () returns boolean
  var s: State
  ch: char
  s = s₀
  ch = nextChar()
  while ch ≠ EOF do
    s = Move(s, ch)
    ch = NextChar()
  endwhile
  if s ∈ FinalStates then
    return true
  else
    return false
  endif
endFunction

The “Move” function
Perhaps an array $s = \text{Move}[s, ch]$
Perhaps a linked list representation, to save space
Is Move always defined?
Use “dead” state to deal with undefined edges.

Simulating a NFA
States: $s, t$
Sets of states: $S, T$
Deterministic Machine:
$\text{Move}(s, ch) \rightarrow t$
Non-deterministic Machine:
$\text{Move}_{\text{NFA}}(S, ch) \rightarrow T$
If $s \in S$ and there is an edge...
then $t \in T$
Example

Move_{NFA} (\{3,7\}, a) = \{ 4, 5, 8 \}

\begin{align*}
\text{Example} & \\
\text{Move}_{NFA} (\{3,7\}, a) & = \{ 4, 5, 8 \}
\end{align*}

Define $\varepsilon$-Closure (s):
The set of states reachable from s on $\varepsilon$-transitions.
$\varepsilon$-closure (4) = \{ 4, 5, 6, 8 \}

\begin{align*}
\text{\varepsilon-Closure} & \\
\text{Define $\varepsilon$-Closure (s):} & \\
& \text{The set of states reachable from s on $\varepsilon$-transitions.} \\
& \varepsilon\text{-closure (4) = \{ 4, 5, 6, 8 \}}
\end{align*}
**Define \( \mathcal{E}\)-Closure (s):**

The set of states reachable from \( s \) on \( \epsilon \)-transitions.

\[ \mathcal{E}\text{-closure}(4) = \{ 4, 5, 6, 8 \} \]

**Define \( \mathcal{E}\)-Closure(S):**

\[ \{ t | t \in \mathcal{E}\text{-closure}(s) \text{ for all } s \in S \} \]

\[ \mathcal{E}\text{-closure}(\{4, 7\}) = \{ 4, 5, 6, 7, 8, 9 \} \]

---

**Computation of \( \mathcal{E}\)-Closure**

*Given:* \( T (= \text{a set of states}) \)

*Goal:* Compute \( \mathcal{E}\)-Closure(T)

*Approach:* Use a stack of states (= the states that we still need to look at)

*Algorithm:*

```plaintext
var
  stack: stack of states
  result: set of states
push all states in T onto stack
result = T
while stack not empty do
  s = pop(stack)
  for each state u such that an edge \( s \to u \) exists do
    if u is not in result then
      add u to result
      push u onto stack
  endIf
endFor
endWhile
```

*The textbook presents a different algorithm!*
Example

Input String: abab

Let \( S \) be the state(s) we are in...

\[
S = \varepsilon\text{-Closure (\{0\})}
\]
**Example**

*Input String: abab*

Let S be the state(s) we are in...

\[
S = \varepsilon\text{-Closure } \{0\} = \{0,2\}
\]
Example

Input String: \texttt{abab}

Let $S$ be the state(s) we are in...

$S = \varepsilon$-Closure ($\{0\}$)
$= \{0,2\}$

Look at next character...

$ch = a$

Move to next state(s)...
Example

Input String: \textit{abab}

Let S be the state(s) we are in...
\[ S = \varepsilon\text{-Closure } \{0\} \]
\[ = \{0,2\} \]

Look at next character...
\[ \text{ch} = \text{a} \]
Move to next state(s)...
\[ S = \varepsilon\text{-Closure } (\text{Move}_{\text{NFA}}(\{0,2\}, \text{a})) \]
\[ = \varepsilon\text{-Closure } (\{1\}) \]
Example

Input String: abab

Let $S$ be the state(s) we are in...
$S = \varepsilon$-Closure ($\{0\}$)
= $\{0,2\}$

Look at next character...
ch = a

Move to next state(s)...  
$S = \varepsilon$-Closure ($\text{Move}_{\text{NFA}}(\{0,2\}, a)$)  
= $\varepsilon$-Closure ($\{1\}$)  
= $\{1\}$

Look at next character...
ch = b

Move to next state(s)...  
$S = \varepsilon$-Closure ($\text{Move}_{\text{NFA}}(\{1\}, b)$)
Example

Input String: abab

Let \( S \) be the state(s) we are in...
\[
S = \varepsilon\text{-Closure } \{(0)\} \\
= \{0,2\}
\]
Look at next character...
\( ch = a \)
Move to next state(s)...
\[
S = \varepsilon\text{-Closure } (\text{Move}_{\text{NFA}}(\{0,2\}, a)) \\
= \varepsilon\text{-Closure } (\{1\}) \\
= \{1\}
\]
Look at next character...
\( ch = b \)
Move to next state(s)...
\[
S = \varepsilon\text{-Closure } (\text{Move}_{\text{NFA}}(\{1\}, b)) \\
= \varepsilon\text{-Closure } (\{1,2\})
\]
**Example**

*Input String: abab*

Let $S$ be the state(s) we are in...

- $S = \varepsilon\text{-Closure } ((0))$
- $\{0, 2\}$

Look at next character...
- $ch = a$

Move to next state(s)... 
- $S = \varepsilon\text{-Closure } (\text{Move}_{\text{NFA}}((0, 2), a))$
- $\varepsilon\text{-Closure } ((1))$
- $\{1\}$

Look at next character...
- $ch = b$

Move to next state(s)... 
- $S = \varepsilon\text{-Closure } (\text{Move}_{\text{NFA}}((1), b))$
- $\varepsilon\text{-Closure } ((1, 2))$
- $\{1, 2\}$
Example

Input String: abab

Let $S$ be the state(s) we are in...
$S = \varepsilon$-Closure ($\{0\}$)  
  $= \{0,2\}$

Look at next character...
ch = a
Move to next state(s)...
$S = \varepsilon$-Closure ($\text{Move}_\text{NFA}($$\{0,2\}$, a)  
  $= \varepsilon$-Closure ($\{1\}$)  
  $= \{1\}$

Look at next character...
ch = b
Move to next state(s)...
$S = \varepsilon$-Closure ($\text{Move}_\text{NFA}($$\{1\}$, b)  
  $= \varepsilon$-Closure ($\{1,2\}$)  
  $= \{1,2\}$

Look at next character...
ch = a
Move to next state(s)...
$S = \varepsilon$-Closure ($\text{Move}_\text{NFA}($$\{1,2\}$, a)  
  $= \varepsilon$-Closure ($\{1\}$)  
  $= \{1\}$

Look at next character...
ch = b
Move to next state(s)...
$S = \varepsilon$-Closure ($\text{Move}_\text{NFA}($$\{1\}$, b)  
  $= \varepsilon$-Closure ($\{1,2\}$)  
  $= \{1,2\}$
Example

Input String: abab

Let $S$ be the state(s) we are in...

$S = \varepsilon$-Closure ($\{0\}$)
$\quad = \{0, 2\}$

Look at next character...

ch = a

Move to next state(s)...

$S = \varepsilon$-Closure ($\text{Move}_{\text{NFA}}(\{0, 2\}, a)$)
$\quad = \varepsilon$-Closure ($\{1\}$)
$\quad = \{1\}$

Look at next character...

ch = b

Move to next state(s)...

$S = \varepsilon$-Closure ($\text{Move}_{\text{NFA}}(\{1\}, b)$)
$\quad = \varepsilon$-Closure ($\{1, 2\}$)
$\quad = \{1, 2\}$
Example

Input String: abab

Let $S$ be the state(s) we are in...

$S = \varepsilon$-Closure (${0})
= \{0,2\}$

Look at next character...
$ch = a$

Move to next state(s)...
$S = \varepsilon$-Closure (Move$_{NFA}$(0,2), a)
  = \varepsilon$-Closure ({1})
  = {1}

Look at next character...
$ch = b$

Move to next state(s)...
$S = \varepsilon$-Closure (Move$_{NFA}$(1, b)
  = \varepsilon$-Closure (1,2)
  = {1,2}

Look at next character...
$ch = a$

Move to next state(s)...
$S = \varepsilon$-Closure (Move$_{NFA}$(1,2), a)
  = \varepsilon$-Closure (1)
  = {1}

Look at next character...
$ch = b$

Move to next state(s)...
$S = \varepsilon$-Closure (Move$_{NFA}$(1), b)
  = \varepsilon$-Closure (1,2)
  = {1,2}

Look at next character...
$ch = EOF$

Does $S$ contain a Final State?
Example

Input String: abab

Let S be the state(s) we are in...
S = $\varepsilon$-Closure ({0})
= {0,2}

Look at next character...
ch = a

Move to next state(s)...
S = $\varepsilon$-Closure (Move_{NFA}({0,2}, a)
= $\varepsilon$-Closure ({1})
= {1}

Look at next character...
ch = b

Move to next state(s)...
S = $\varepsilon$-Closure (Move_{NFA}({1}, b)
= $\varepsilon$-Closure ({1,2})
= {1,2}

Look at next character...
ch = a

Move to next state(s)...
S = $\varepsilon$-Closure (Move_{NFA}({1,2}, a)
= $\varepsilon$-Closure ({1})
= {1}

Look at next character...
ch = b

Move to next state(s)...
S = $\varepsilon$-Closure (Move_{NFA}({1}, b) = {1,2}

Look at next character...
ch = EOF

Does S contain a Final State?
This string is accepted!!!

Simulating a NFA

function Match () returns boolean
var S: set of states
ch: char
S = $\varepsilon$-Closure({s_0})
ch = nextChar()
while ch ≠ EOF do
    S = $\varepsilon$-Closure(Move_{NFA}(S, ch))
    ch = NextChar()
endWhile
if S ∩ FinalStates ≠ {} then
    return true
else
    return false
endif
endFunction
Thompson’s Construction

Build an NFA for: $ab^*c \mid d^*e^*$

Break the expression into sub-expressions

$$(ab^*) \mid (d^*e^*)$$

Build NFA for this

Build NFA for this
Thompson’s Construction

Build an NFA for: \( ab^*c | d^*e^* \)

Break the expression into sub-expressions

\[
\begin{align*}
(ab^*c) & \quad \mid \quad (d^*e^*) \\
\end{align*}
\]

Build NFA for this

Build NFA for this

Glue the two NFAs together
Lexical Analysis - Part 2

**Thompson’s Construction**

Build an NFA for: \(ab^*c|d*e^*\)

Break the expression into sub-expressions

\[
\begin{align*}
(ab^*c) & \quad | \\
(d*e^*) & 
\end{align*}
\]

Build NFA for this 
Build NFA for this

Glue the two NFAs together

---

© Harry H. Porter, 2005
**Thompson’s Construction**

**Given:**
Regular Expression, R

**Goal:**
Construct an NFA to recognize L(R)
Call the NFA which is constructed N(R)

**Approach:**
Look at the syntax of the expression R.
Top-most operator with sub-expressions:
\[ R = R_1 \oplus R_2 \]
For each sub-expression \( R_i \),...
Build an NFA called \( N(R_i) \)
For each larger expression
(...which is built from smaller expressions)
Build an NFA
using the NFA’s for component sub-expressions.
In other words, construct \( N(R) \) from \( N(R_1) \) and \( N(R_2) \)

---

What kinds of regular expressions are there?

- **case 1:** \( a \) where \( a \in \Sigma \)
- **case 2:** \( r_1 \mid r_2 \)
- **case 3:** \( r_1r_2 \)
- **case 4:** \( r_1^* \)
- **case 5:** \( \epsilon \)
- **case 6:** \( (r_1) \)

**Note:**
For every NFA we construct...
- 1 start state
- 1 accepting state
- No edge enters the start state
- No edge leaves the accepting state
**Case 1:** \( a \) where \( a \in \Sigma \)

For a regular expression consisting of only \( a \) (for any \( a \in \Sigma \))

Construct

\[
\begin{align*}
  & \quad a \\
\end{align*}
\]

...and call it \( N(a) \)

**Case 2:** \( r_1 | r_2 \)

For \( r_1 | r_2 \), construct \( N(r_1 | r_2) \)

Let \( N(r_i) \) be:

Then, \( N(r_1 | r_2) \) is

\[
\begin{align*}
  & N(r_1) \\
\end{align*}
\]

Note: These states are no longer final states
**Case 3: \( r_1 r_2 \)**

From \( N(\mathbf{r}_1) \) and \( N(\mathbf{r}_2) \)...

Construct \( N(\mathbf{r}_1 \mathbf{r}_2) \) as follows:

```
N(\mathbf{r}_1) \quad N(\mathbf{r}_2)
```

(Alternative: combine states)

From \( N(\mathbf{r}_1) \) and \( N(\mathbf{r}_2) \)...

Construct \( N(\mathbf{r}_1 \mathbf{r}_2) \) as follows:
**Case 4: r₁**

From $N(r₁)...$

Construct $N(r₁*)$ as follows:

**Case 5: ε**

Let $N(ε)$ be...

**Case 6: (r₁)**

Let $N((r₁))$ be $N(r₁)$ itself.
Example: \((a \mid b)^*abb\)
Example: \((a|b)^*abb\)

\[
\begin{align*}
N(a) & = N((a|b)*) \\
N(b) & \\
N(a|b) & = N((a|b))
\end{align*}
\]
Example: $(a|b)^*abb$

$N(a)$

$N(b)$

$N((a|b)^*)$

$N((a|b)) = N((a|b))$

$N((a|b)^*) = N((a|b)^*)$

$N(abb)$

$N((a|b)^*abb)$