CSE443 Compilers

Dr. Carl Alphonce
alphonce@buffalo.edu
343 Davis Hall

Announcements

- Team info submitted!
- Starting today, sit with your teammates.
- Attendance sheets will be by team.
- Team meetings will start next week.

Phases of a compiler

Lexical structure

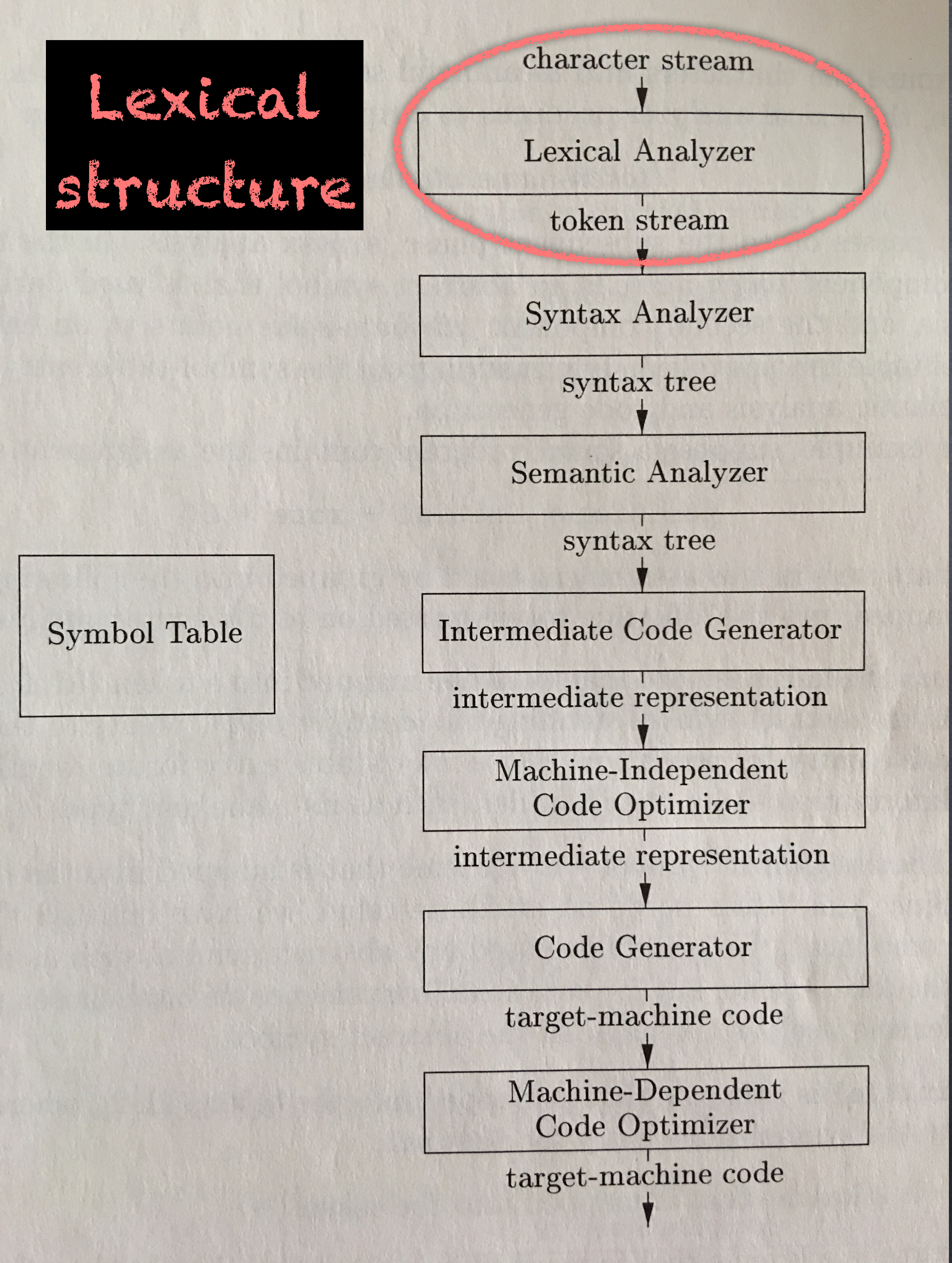
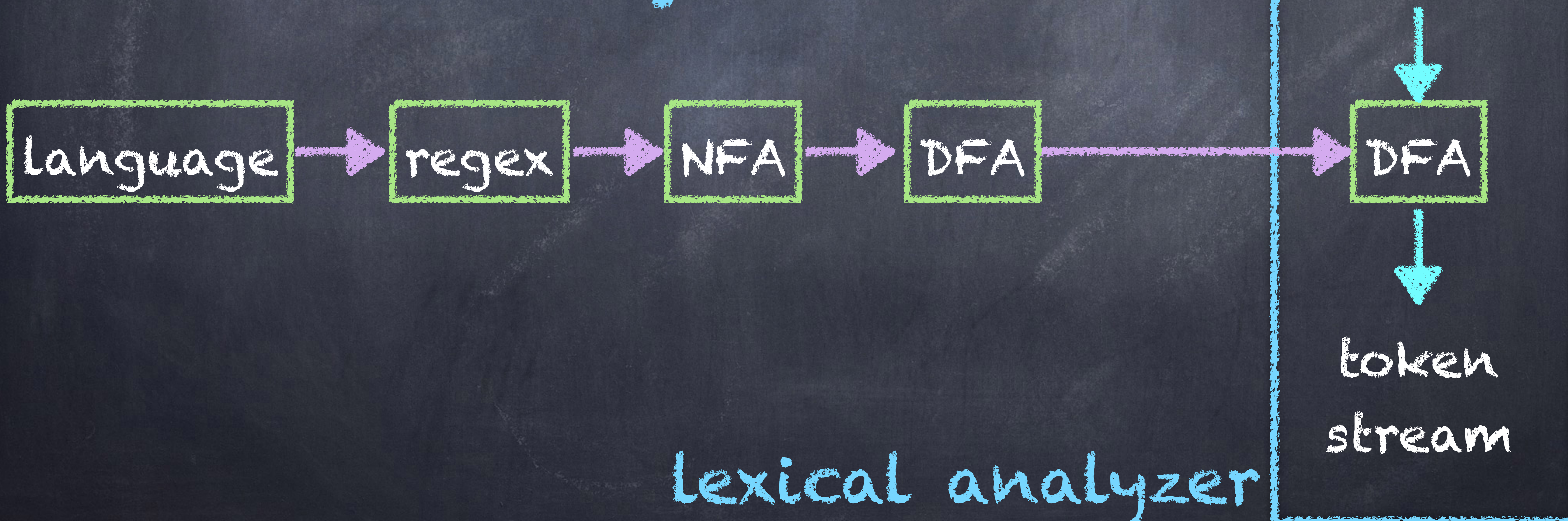


Figure 1.6, page 5 of text

Process of building lexical analyzer

5) The minimal DFA is our lexical analyzer



FLEX generates a C program which implements the minimized DFA

step 2

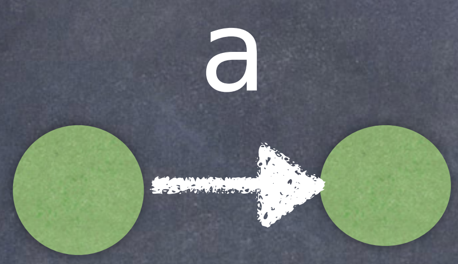
$\boxed{\text{NFA}} \rightarrow \boxed{\text{DFA}}$

(a|b)*abb

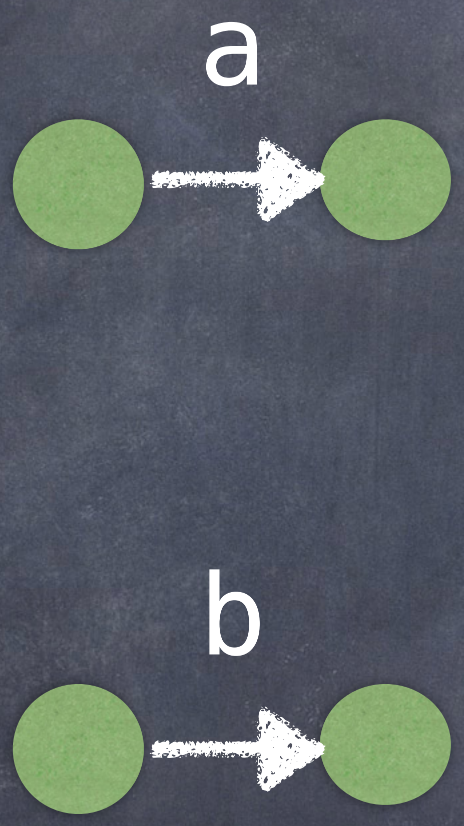
first we construct an NFA

from this regular expression

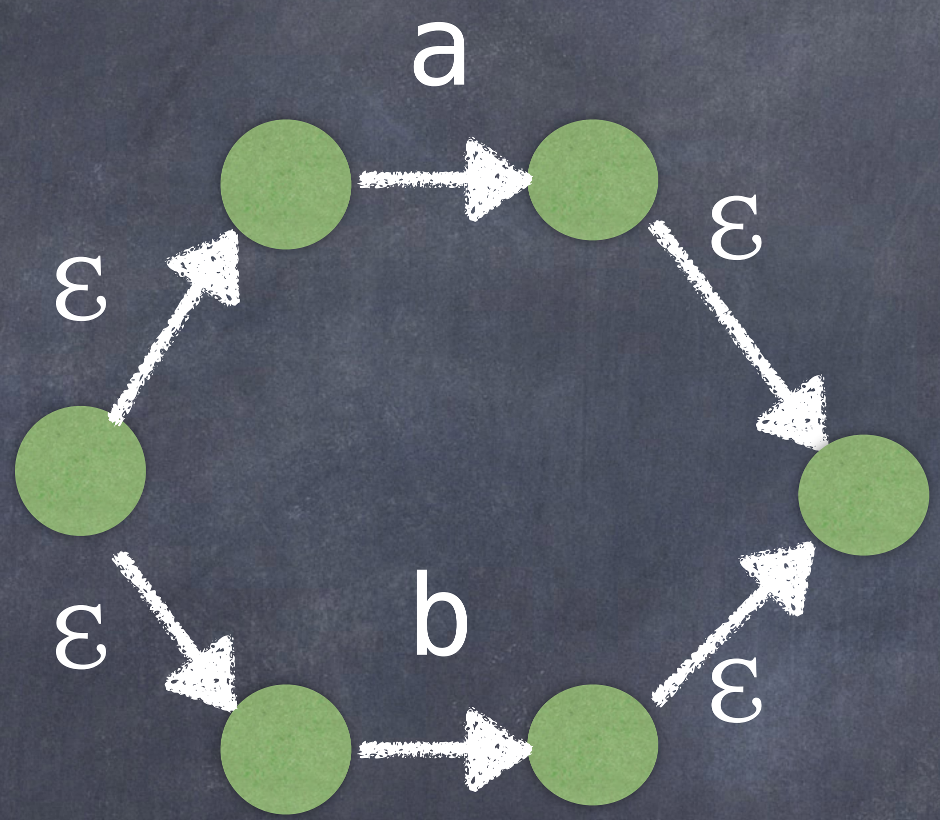
(a|b)*abb



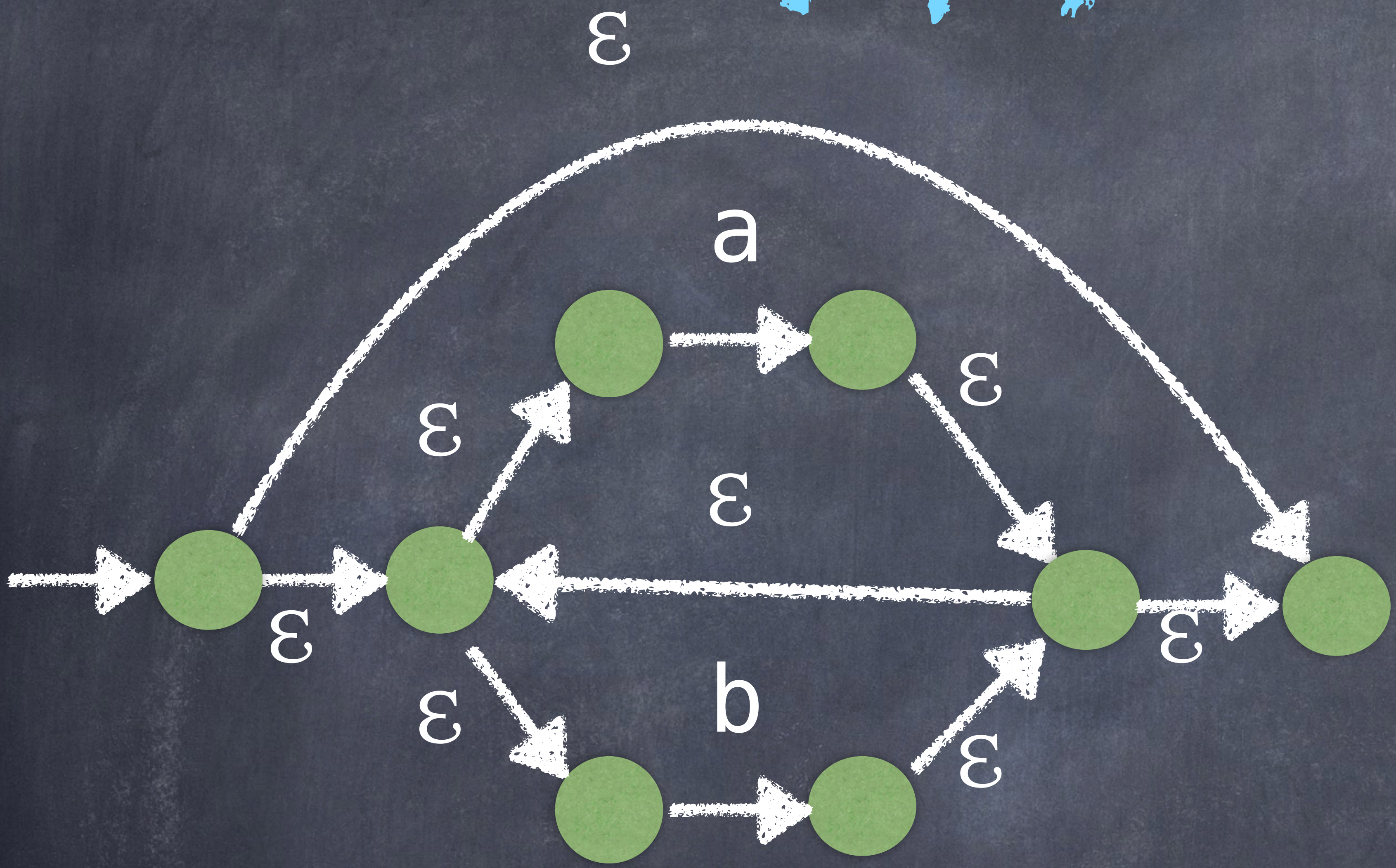
(a|b)*abb



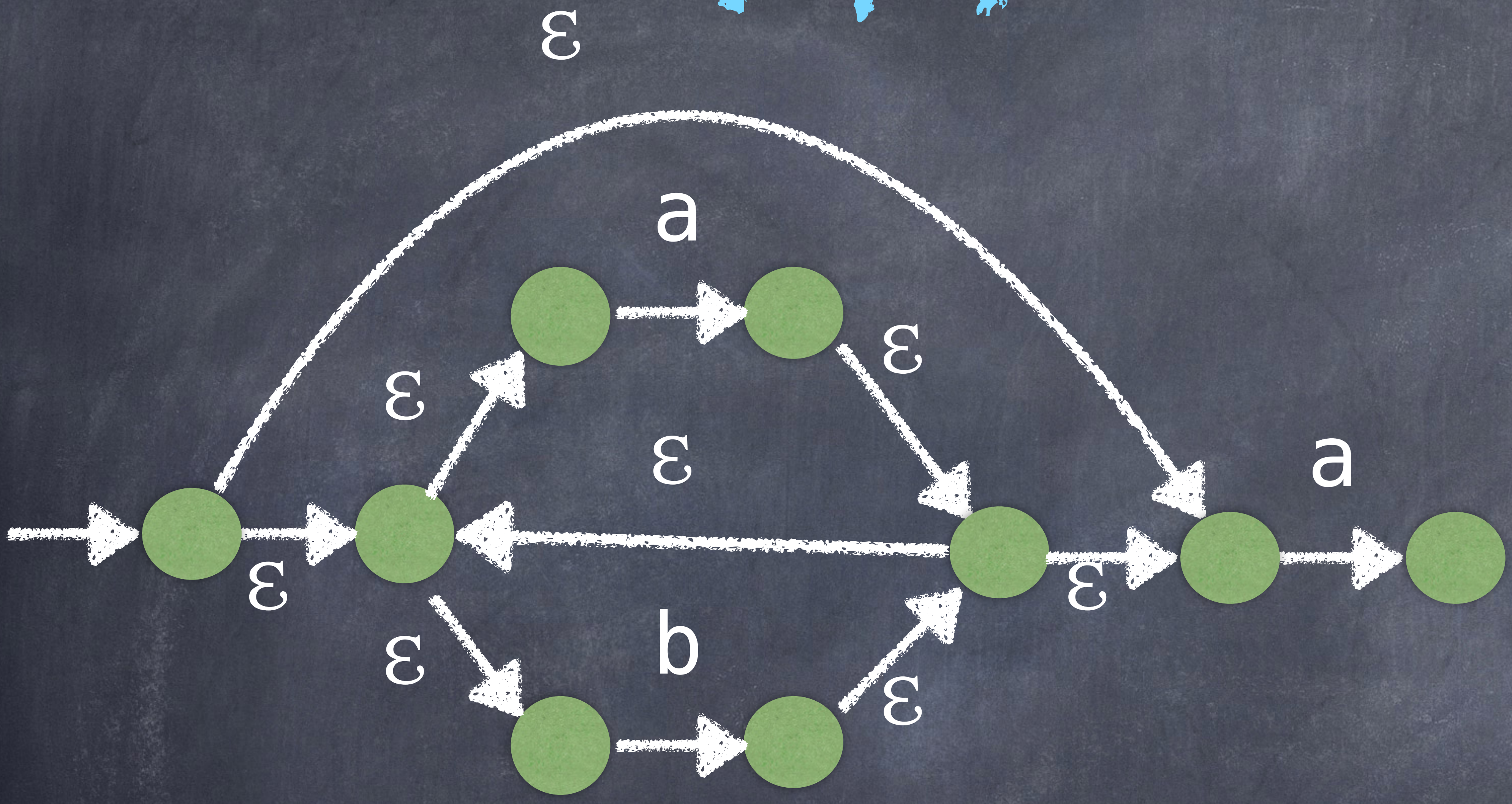
(a|b)*abb



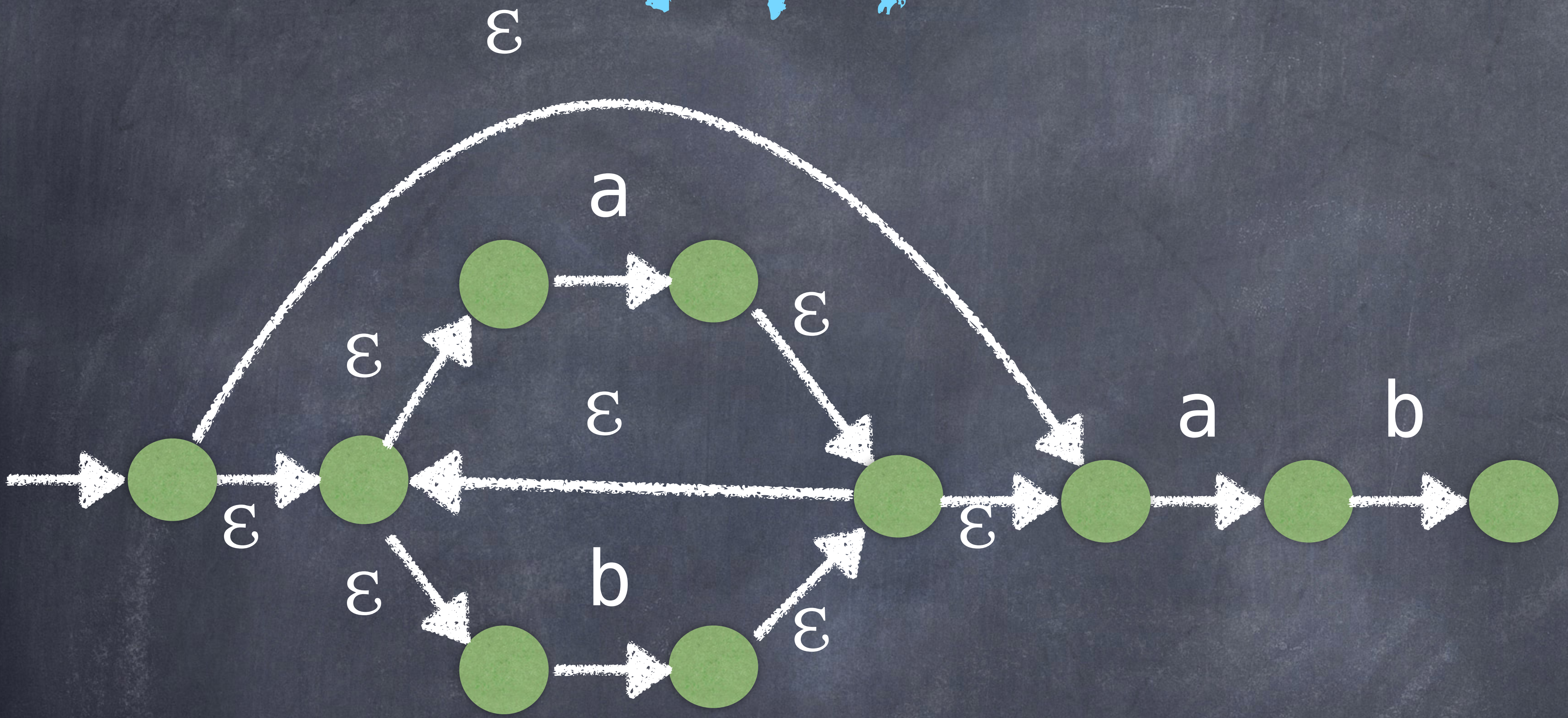
$(a|b)^*abb$



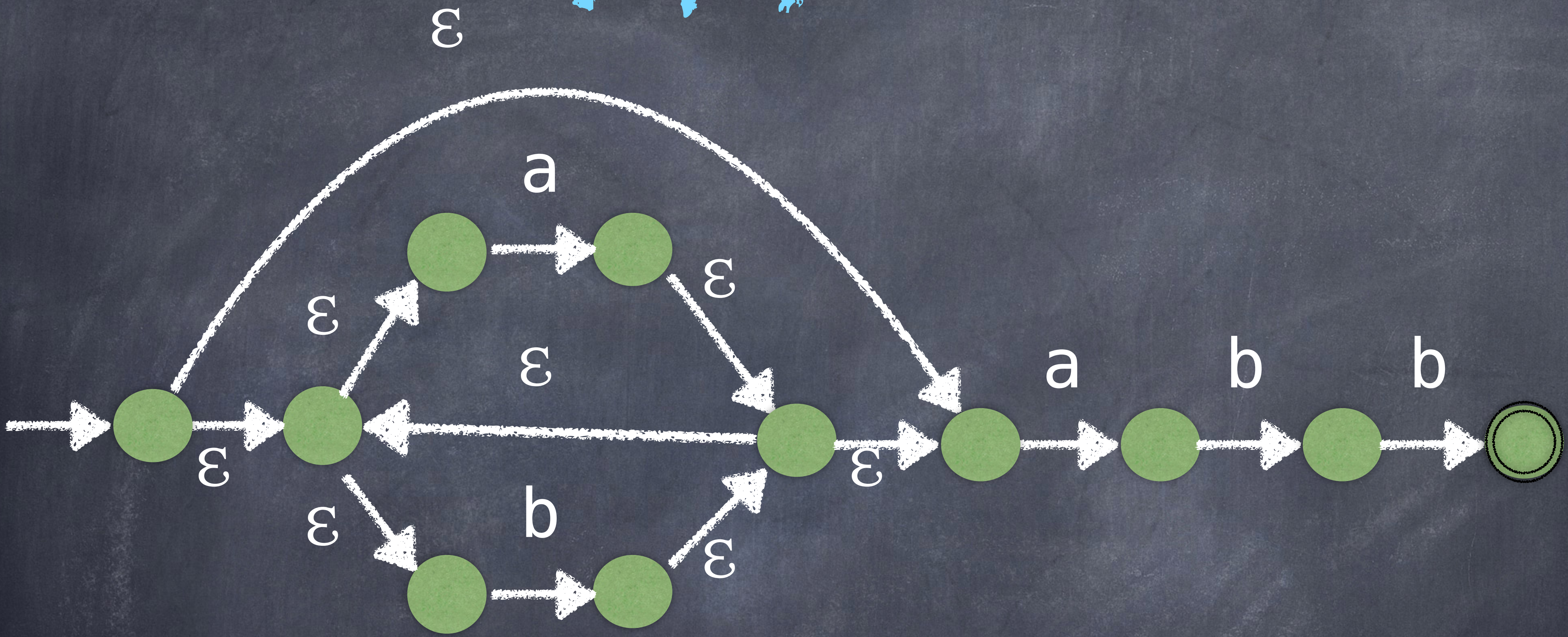
(a|b)*abb



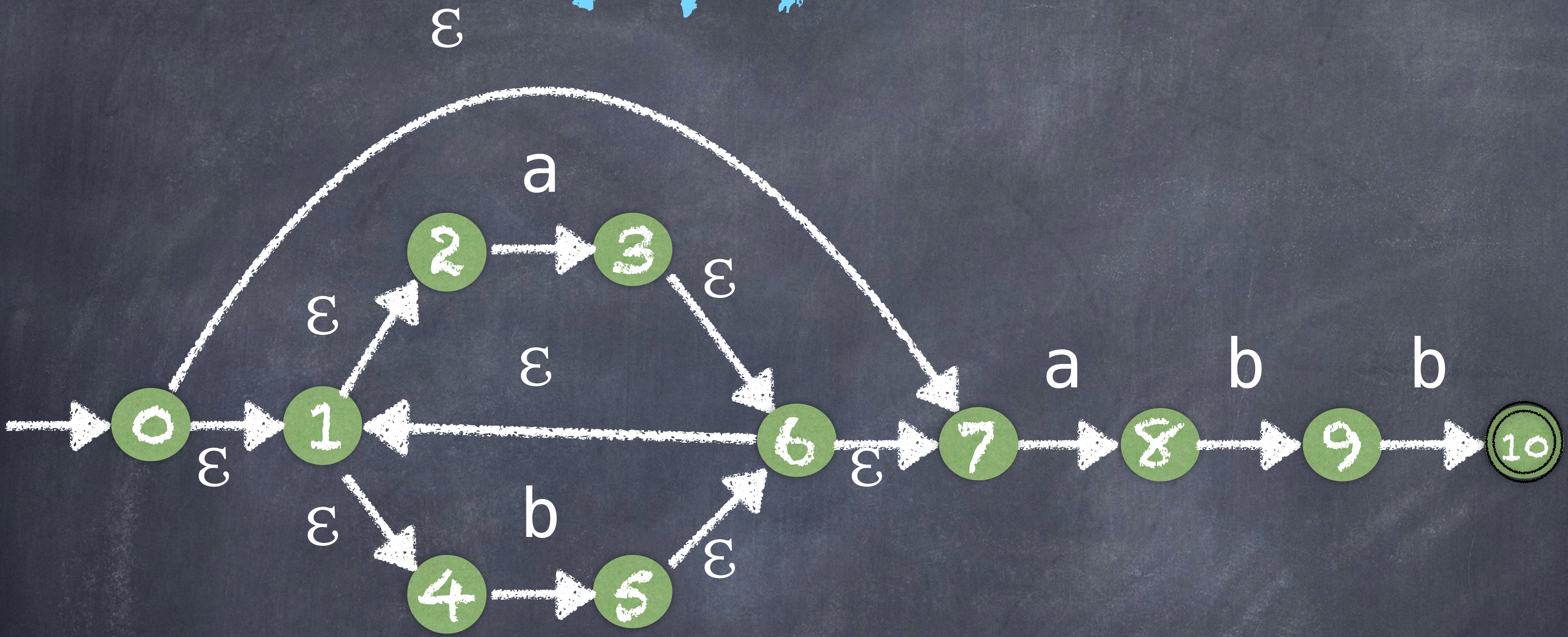
(a|b)*abb



(a|b)*abb



(a|b)*abb



Operations

- ε-closure(t) is the set of states reachable from state t using only ε-transitions.
- ε-closure(T) is the set of states reachable from any state $t \in T$ using only ε-transitions.
- move(T,a) is the set of states reachable from any state $t \in T$ following a transition on symbol $a \in \Sigma$.

NFA -> DFA algorithm

(set of states construction - page 153 of text)

- INPUT: An NFA $N = (S, \Sigma, \delta, s_0, F)$
- OUTPUT: A DFA $D = (S', \Sigma, \delta', s_0', F')$ such that $\mathcal{L}(D)=\mathcal{L}(N)$
- ALGORITHM:

Compute $s_0' = \varepsilon$-closure(s_0), an unmarked set of states

Set $S' = \{ s_0' \}$

while there is an unmarked $T \in S'$

  mark T

  for each symbol $a \in \Sigma$

    let $U = \varepsilon$-closure(move(T,a))

    if $U \notin S'$, add unmarked U to S'

    add transition: $\delta'(T,a) = U$

F' is the subset of S' all of whose members contain a state in F.

NFA -> DFA algorithm

(set of states construction - page 153 of text)

$S_0' = \{ A = \{0,1,2,4,7\} \}$

Pick an unmarked set from S_0', A, mark it, and $\forall x \in \Sigma$ let $U = \varepsilon\text{-closure}(move(A,x))$, if $U \notin S'$, add unmarked U to S' and add transition: $\delta'(A,x) = U$

$S_1' = \{ A✓ , B = \{1,2,3,4,6,7,8\} , C = \{1,2,4,5,6,7\}\}$

$\delta'(A,a) = B$

$\delta'(A,b) = C$

Pick an unmarked set from S_1', B, mark it, and $\forall x \in \Sigma$ let $U = \varepsilon\text{-closure}(move(B,x))$, if $U \notin S'$, add unmarked U to S' and add transition: $\delta'(B,x) = U$

$S_2' = \{ A✓ , B✓ , C , D = \{1,2,4,5,6,7,9\}\}$

$\delta'(B,a) = B$

$\delta'(B,b) = D$

Pick an unmarked set from S_2', C, mark it, and $\forall x \in \Sigma$ let $U = \varepsilon\text{-closure}(move(C,x))$, if $U \notin S'$, add unmarked U to S' and add transition: $\delta'(C,x) = U$

$S_3' = \{ A✓ , B✓ , C✓ , D \}$

$\delta'(C,a) = B$

$\delta'(C,b) = C$

NFA -> DFA algorithm

(set of states construction - page 153 of text)

Pick an unmarked set from S_3', D, mark it, and $\forall$ x $\in \Sigma$ let U = ε-closure(move(D,x)), if U $\notin$ S', add unmarked U to S' and add transition: $\delta'(D,x) = U$

$S_4' = \{ A✓, B✓, C✓, D✓, E = \{1,2,4,5,6,7,10\} \}$

$\delta'(D,a) = B$

$\delta'(D,b) = E$

Pick an unmarked set from S_4', E, mark it, and $\forall$ a $\in \Sigma$ let U = ε-closure(move(E,a)), if U $\notin$ S', add unmarked U to S' and add transition: $\delta'(E,a) = U$

$S_5' = \{ A✓, B✓, C✓, D✓, E✓ \}$

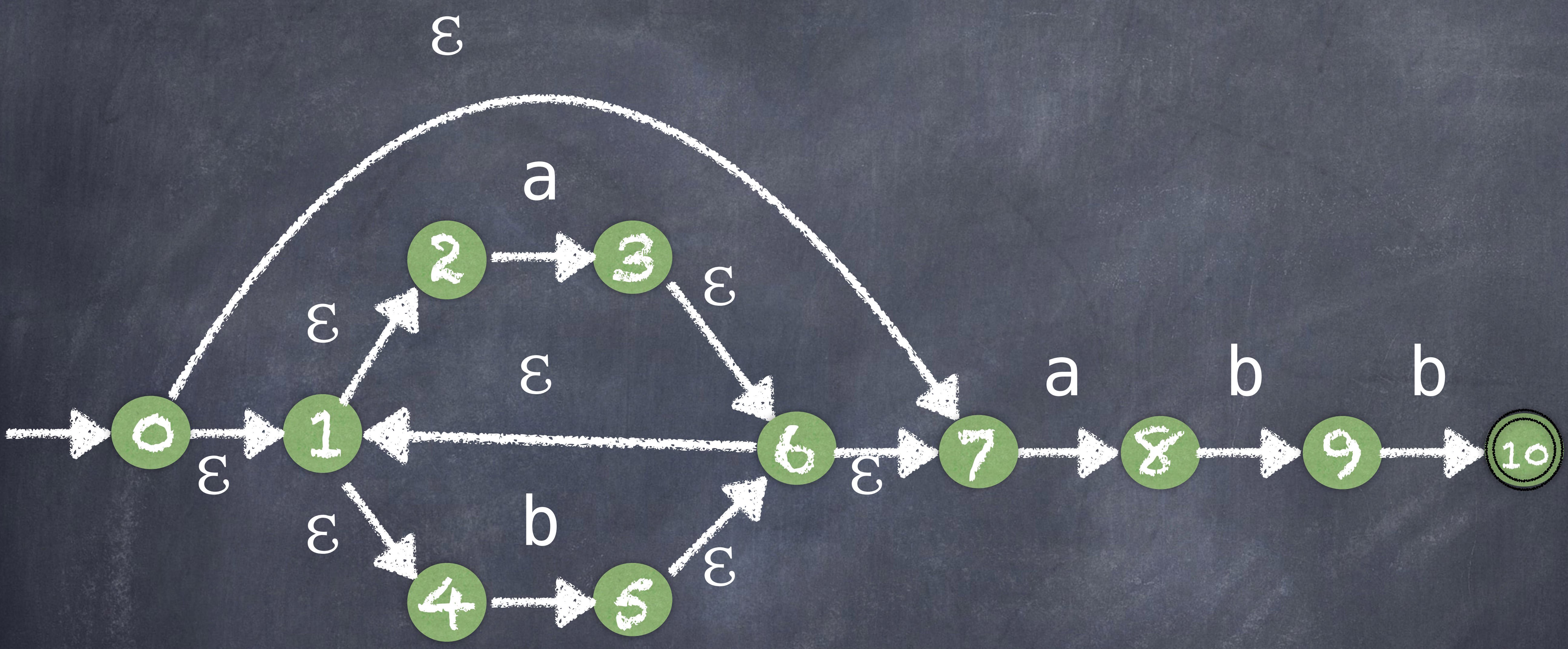
$\delta'(E,a) = B$

$\delta'(E,b) = C$

Since there are no unmarked sets in S_5' the algorithm has reached a fixed point.
STOP.

F' is the subset of S' all of whose members contain a state in F: {E}

The original NFA



The resulting DFA

DFA = ({A, B, C, D, E}, {a, b}, A, δ', {E}), where

δ'(A,a) = B

δ'(A,b) = C

δ'(B,a) = B

δ'(B,b) = D

δ'(C,a) = B

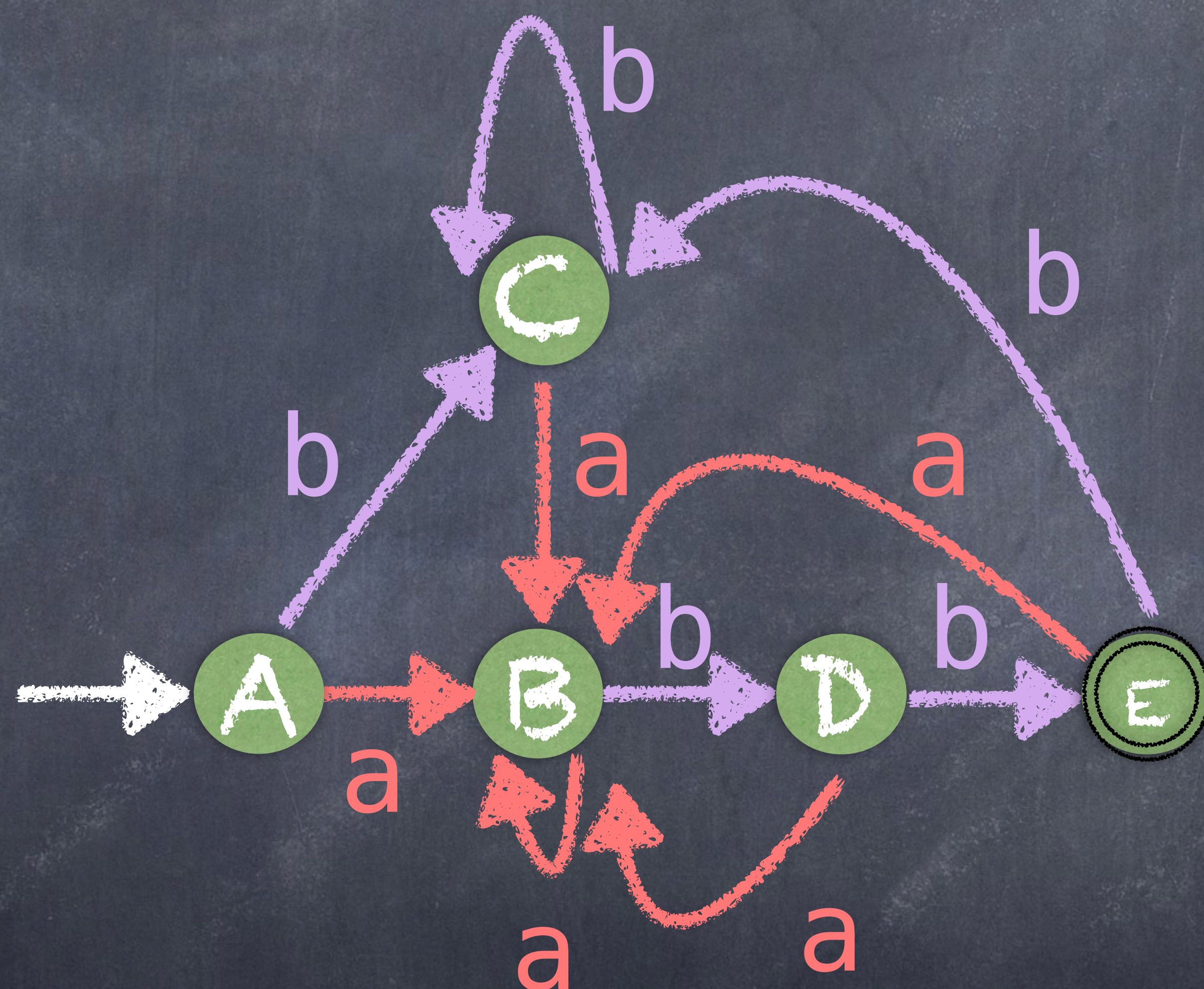
δ'(C,b) = C

δ'(D,a) = B

δ'(D,b) = E

δ'(E,a) = B

δ'(E,b) = C



step 3
DFA minimization

DFA -> minimal DFA algorithm

- INPUT: An DFA $D = (S, \Sigma, \delta, s_0, F)$
- OUTPUT: A DFA $D' = (S', \Sigma, \delta', s_0', F')$ such that
  - S' is as small as possible, and
  - $\mathcal{L}(D)=\mathcal{L}(D')$
- ALGORITHM:

1. Let $\pi = \{ F, S-F \}$
2. Let $\pi' = \pi$. For every group G of π:

   partition G into subgroups such that two states s and t are in the same subgroup iff for all input symbols a, states s and t have transitions on a to states in the same group of π

   Replace G in π' by the set of all subgrops formed
3. if $\pi'=\pi$ let $\pi''=\pi$, otherwise set $\pi=\pi'$ and repeat 2.
4. Choose one state in each group of π'' as a representative for that group.

   a) The start state of D' is the representative of the group containing the start state of D

   b) The accepting states of D' are the representatives of those groups that contain an accepting state of D

   c) Adjust transitions from representatives to representatives.

ORIGINAL DFA

$D = (S, \Sigma, s_0, \delta, F)$

$S = \{A, B, C, D, E\}$

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

$s_0 = A$

$\delta = \{(A,a) \to B, (A,b) \to C,$
$(B,a) \to B, (B,b) \to D,$
$(C,a) \to B, (C,b) \to C,$
$(D,a) \to B, (D,b) \to E,$
$(E,a) \to B, (E,b) \to C\}$

$F = \{E\}$

Finding the minimal set of distinct sets of states

π_0 = { F, S-F } = { {E}, {A,B,C,D} }

Pick a non-singleton set X = {A,B,C,D} from π_0 and

check behavior of states on all transitions on symbols in Σ (are they to states in X or to other groups in the partition?)

(A,a)->B, (B,a)->B, (C,a)->B, (D,a)->B
(A,b)->C, (B,b)->D, (C,b)->C, (D,b)->E

D behaves differently, so put it in its own partition.

Finding the minimal set of distinct sets of states

π_1 = { {E}, {A, B, C}, {D} }

Pick a non-singleton set X = {A,B,C} from π_1 and

check behavior of states on all transitions on symbols in Σ (are they to states in X or to other groups in the partition?)

(A,a)->B, (B,a)->B, (C,a)->B
(A,b)->C, (B,b)->D, (C,b)->C

B behaves differently, so put it in its own partition.

Finding the minimal set of distinct sets of states

π_2 = { {E}, {A, C}, {B}, {D} }

Pick a non-singleton set X = {A,C} from π_2 and check

behavior of states on all transitions on symbols in Σ (are they to states in X or to other groups in the partition?)

(A,a)->B, (C,a)->B
(A,b)->C, (C,b)->C

A and C both transition outside the group on symbol a, to the same group (the one containing B). Therefore A and C are indistinguishable in their behaviors, so do not split this group.

Finding the minimal set of distinct sets of states

$\pi_3 = \{ \{E\}, \{A, C\}, \{B\}, \{D\} \} = \pi_2$

We have reached a fixed point! STOP

Pick a representative from each group

$\pi_{\text{FINAL}} = \{\ \{E\}, \{A, C\}, \{B\}, \{D\}\ \}$

MINIMAL DFA

$D' = (S', \Sigma, s'_0, \delta', F')$

$S' = \{B, C, D, E\}$ -> the representatives

$\Sigma = \{a, b\}$ -> no change

$s'_0 = C$ -> the representative of the group that contained D's starting state, A

δ = (on next slide)

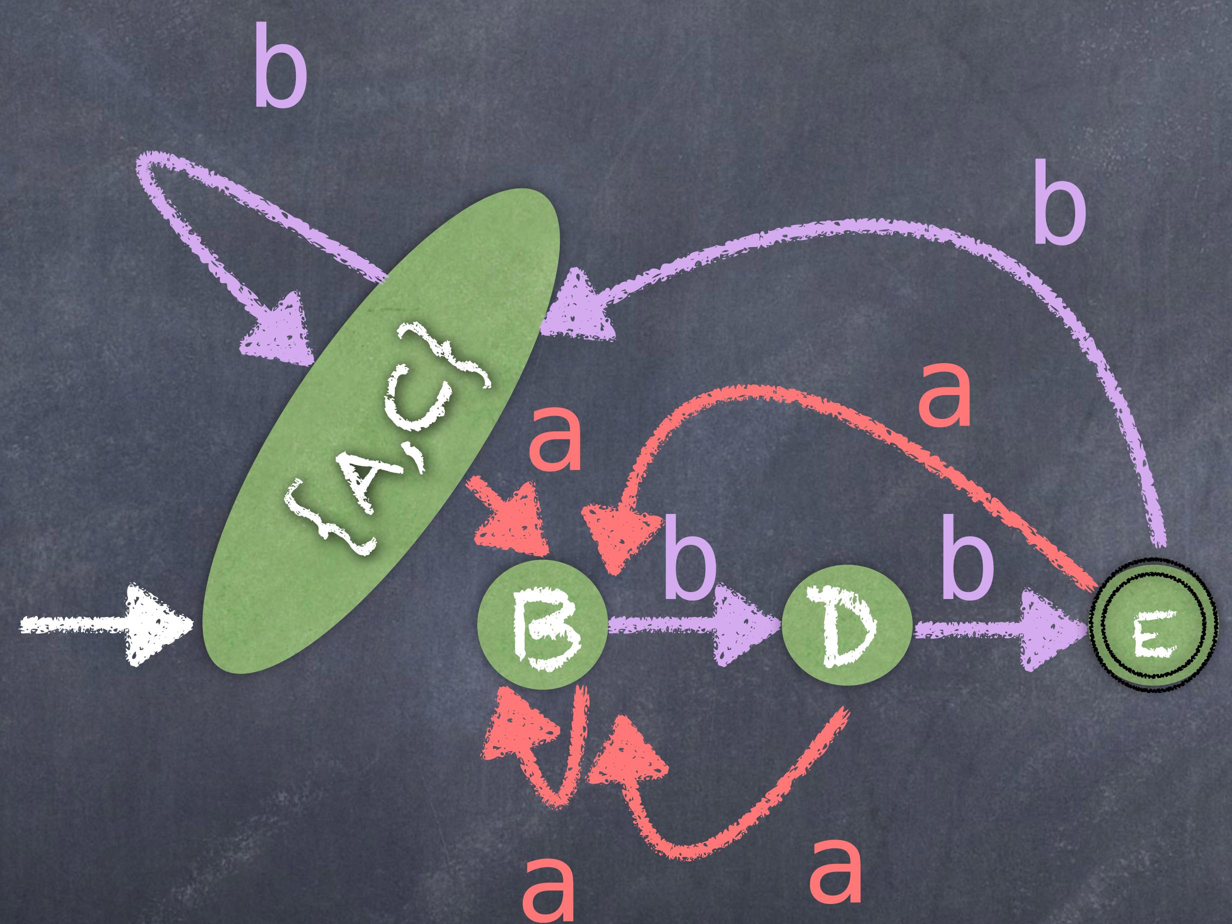
$F = \{E\}$ -> the representatives of all the groups that contained any of D's final states (which, in this case, was just $\{E\}$)

The new transition function δ'

- For each state $s \in S'$, consider its transitions in D, on each $a \in \Sigma$.
- if $\delta(s,a) = t$, then $\delta'(s,a) = r$, where r is the representative of the group containing t.

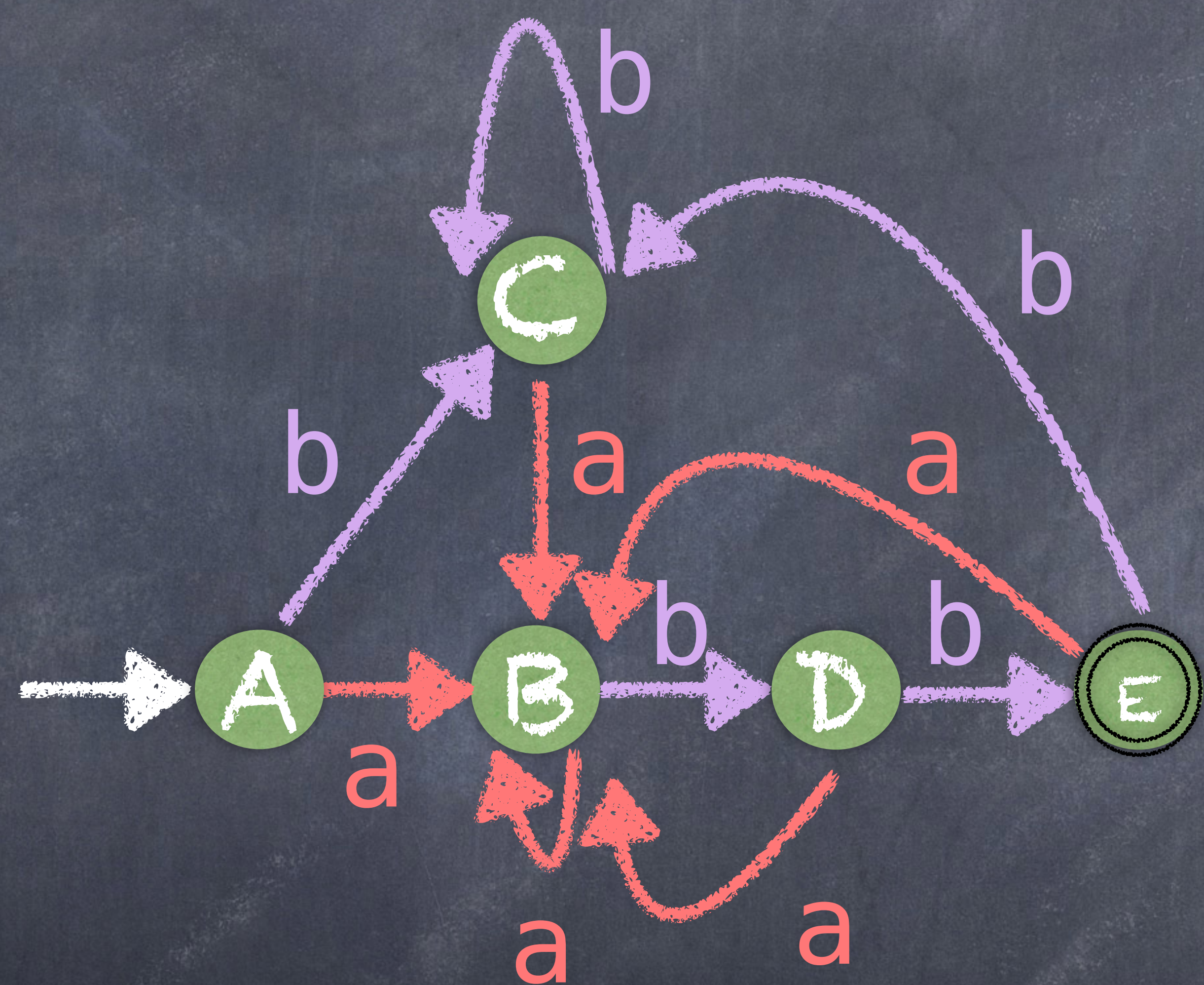
$$\delta = \{ (B,a) \rightarrow B,\ (B,b) \rightarrow D,$$
$$(C,a) \rightarrow B,\ (C,b) \rightarrow C,$$
$$(D,a) \rightarrow B,\ (D,b) \rightarrow E,$$
$$(E,a) \rightarrow B,\ (E,b) \rightarrow C \}$$

Minimal DFA for (a|b)*abb



Non-minimized

DFA for (a|b)*abb



Phases of a compiler

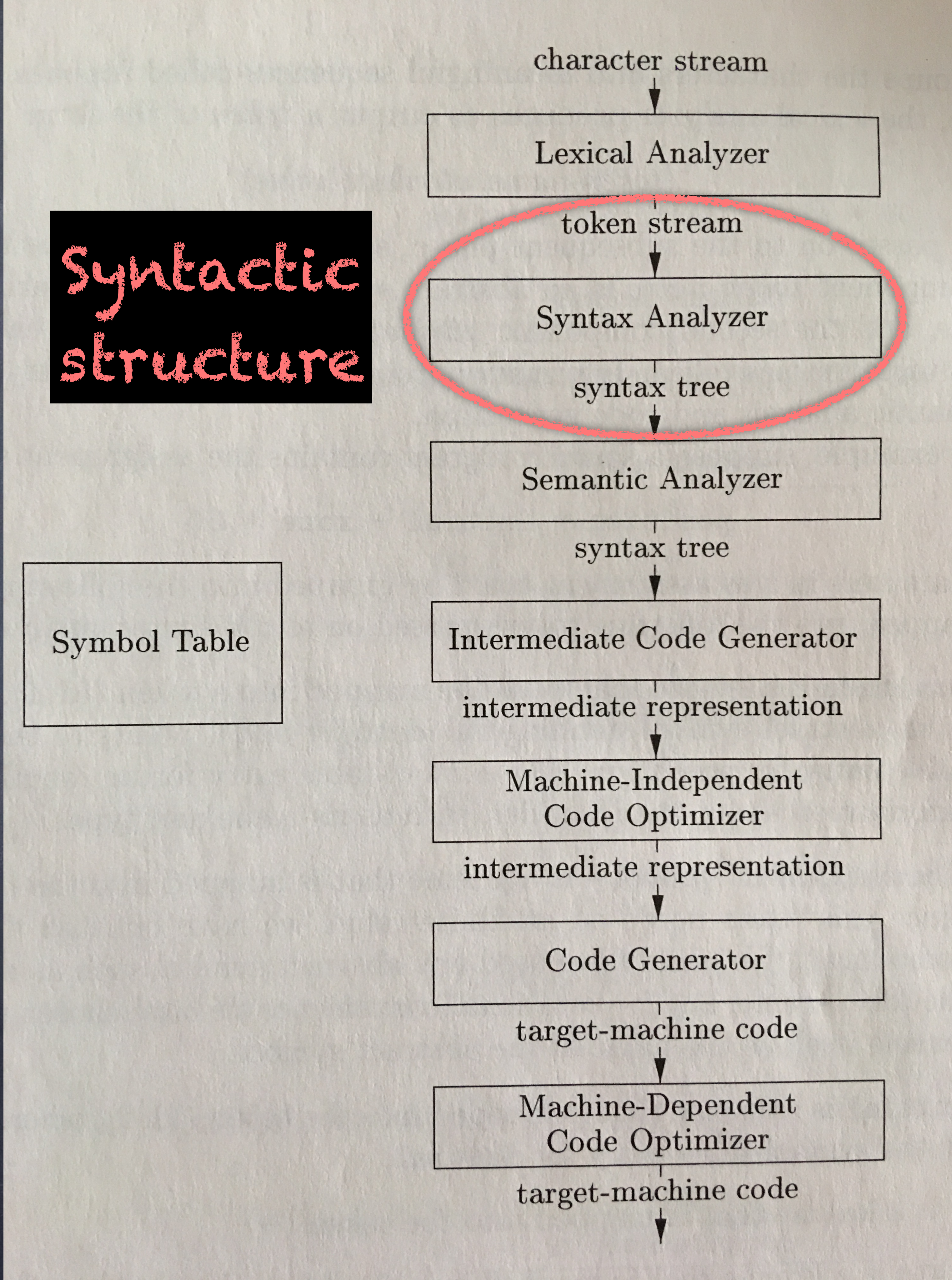
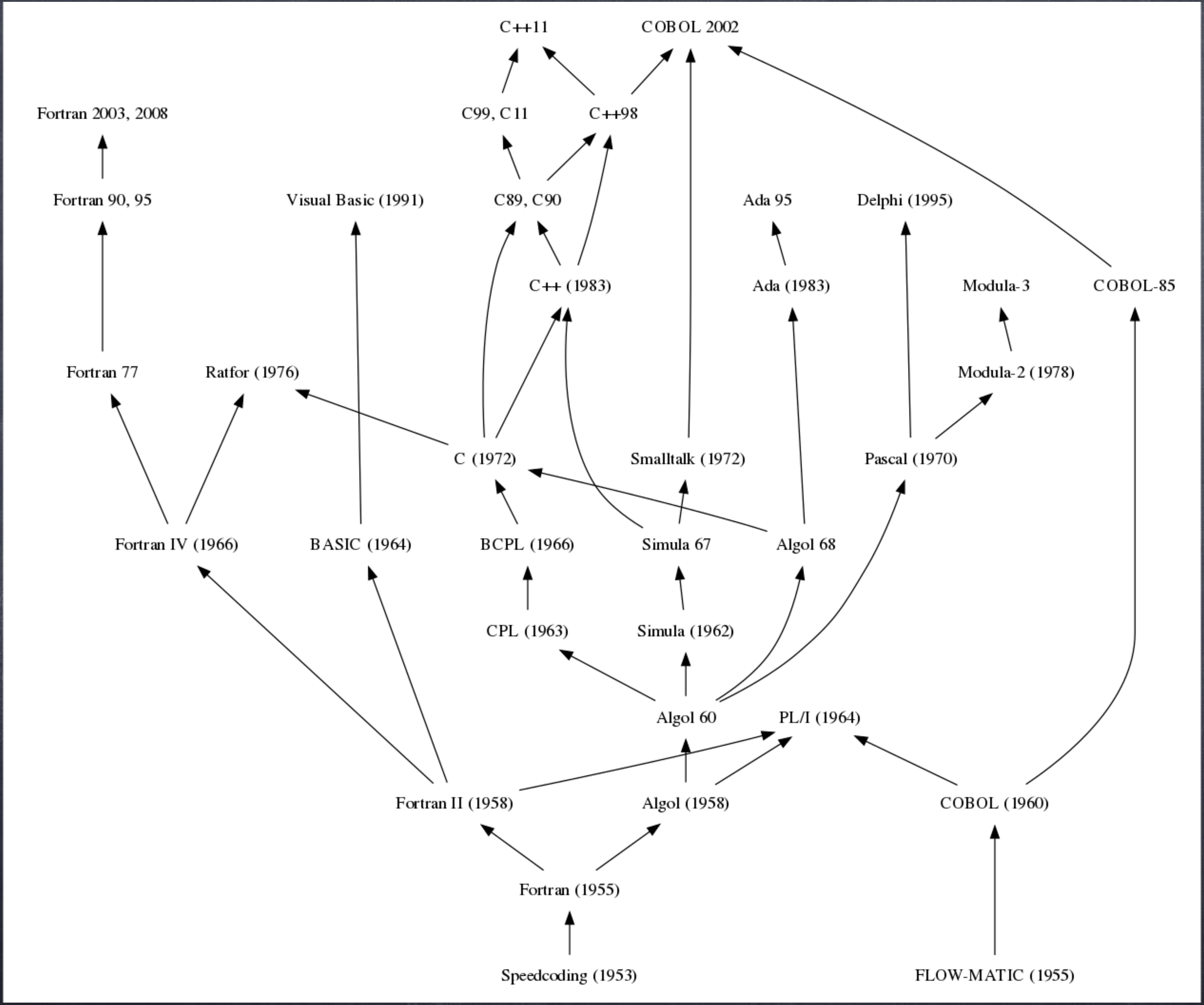


Figure 1.6, page 5 of text



http://www.softwarepreservation.org/projects/FORTRAN/paper/p4-backus.pdf

http://www.bitsavers.org/pdf/univac/flow-matic/U1518_FLOW-MATIC_Programming_System_1958.pdf

Rear Admiral Grace Murray Hopper
(1906 - 1992)

In 1952, Hopper completed her first compiler (for Sperry-Rand computer), known as the *A-0 System*. [...]

After the A-0, Grace Hopper and her group produced versions A-1 and A-2, improvements over the older version. The A-2 compiler was the first compiler to be used extensively, paving the way to the development of programming languages.

[...]

Hopper also originated the idea that computer programs could be written in English. She viewed letters as simply another kind of symbol that the computer could recognize and convert into machine code. Hopper's compiler later evolved to FLOW-MATIC compiler, which will be the base for the extremely important language—COBOL.

https://history-computer.com/ModernComputer/Software/FirstCompiler.html

Context Free Grammars

CFG $G = (N, T, P, S)$

N is a set of non-terminals

T is a set of terminals (= tokens from lexical analyzer)

$T \cap N = \varnothing$ (i.e. a symbol is either a terminal or a non-terminal, not both)

P is a set of productions/grammar rules

$P \subseteq N \times (N \cup T)^*$

$R \in P$ is written as $X \rightarrow \alpha$, where $X \in N$ and $\alpha \in (N \cup T)^*$

$S \in N$ is the start symbol

Derivations

$\Rightarrow_G$ "derives in one step (from G)"

If $A \rightarrow \beta \in P$, and $\alpha, \gamma \in (N \cup T)^*$ then $\alpha A \gamma \Rightarrow_G \alpha\beta\gamma$

$\Rightarrow_G{}^*$ "derives in many steps (from G)"

If $\alpha_i \in (N \cup T)^*$, $m \geq 1$ and $\alpha_1 \Rightarrow_G \alpha_2 \Rightarrow_G \alpha_3 \Rightarrow_G \alpha_4 \ \ldots \Rightarrow_G \alpha_m$

then $\alpha_1 \Rightarrow_G{}^* \alpha_m$

$\Rightarrow_G{}^*$ is the reflexive and transitive closure of $\Rightarrow_G$

Languages

$\mathcal{L}(G) = \{ w \mid w \in T^* \text{ and } S \Rightarrow_G{}^* w \}$

L is a CF language if it is $\mathcal{L}(G)$ for a CFG G.

G1 and G2 are equivalent iff $\mathcal{L}(G1)=\mathcal{L}(G2)$.

Example

L = { 0, 1, 00, 11, 000, 111, 0000, 1111, ... }

```
G = ( {0,1}, {S, ZeroList, OneList},
      {S -> ZeroList | OneList,
       ZeroList -> 0 | 0 ZeroList,
       OneList -> 1 | 1 OneList },
       S )
```

Derivations from G

Derivation of 0 0 0 0

S => ZeroList

=> 0 ZeroList

=> 0 0 ZeroList

=> 0 0 0 ZeroList

=> 0 0 0 0

Derivation of 1 1 1

S => OneList

=> 1 OneList

=> 1 1 OneList

=> 1 1 1