Nondeterministic FSMs

CS 536

Explore NFAs
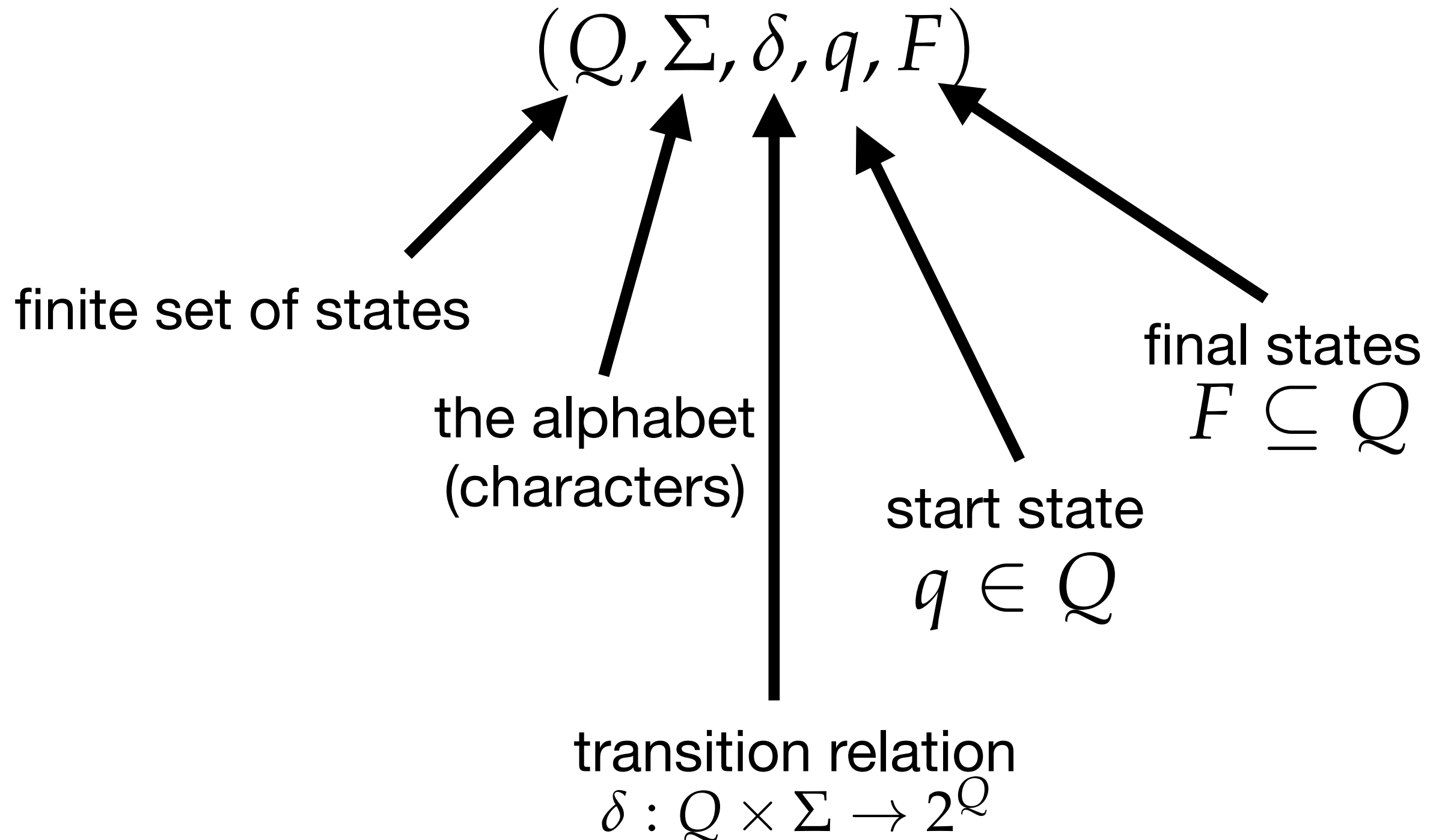
Claim: NFAs add no power to DFAs

Epsilon transitions

Claim: Epsilon transitions add no power
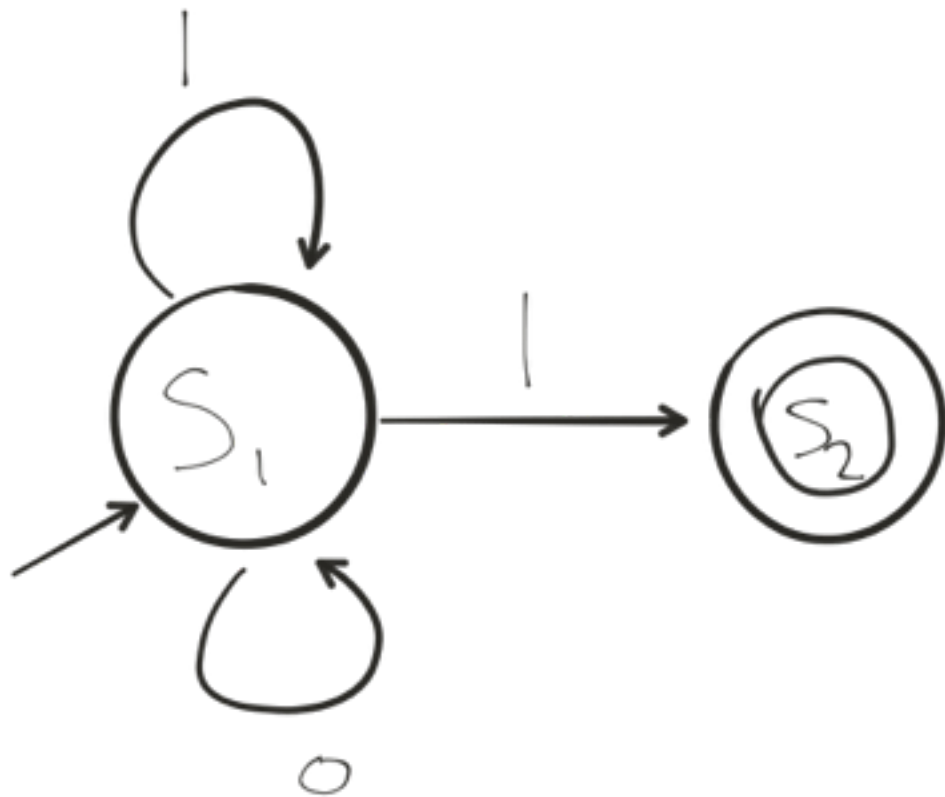
Regular expressions

NFMs, formally

$$(Q, \Sigma, \delta, q, F)$$

finite set of states

the alphabet
(characters)

transition relation
$$\delta : Q \times \Sigma \to 2^Q$$

start state
$$q \in Q$$

final states
$$F \subseteq Q$$

NFA

To check if string is in *L(M) of* NFA *M,* simulate **set** of **choices** it could make


```
     1     1     1
 s1   s1    s1   s1
 s1   s1    s1   s2
```

```
      1      1      1
 1 —> S1 —> S1 —> S1
      |         \——>S2
      \——>S2 —X
 \——>S2 —X
```
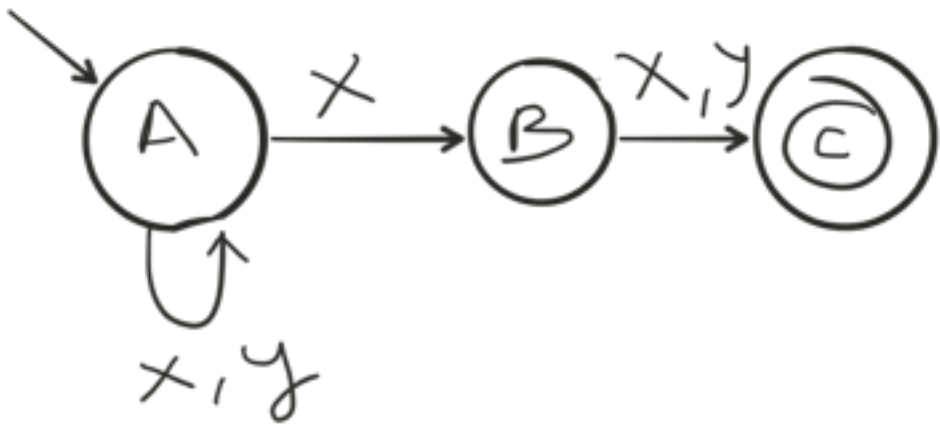
NFA == DFA

Claim: *L(NFA) = L(DFA)*

Idea: we can only be in finitely many subsets of states at any one time

$$2^{|Q|}$$ possible combinations of states

Why?

Why 2^|Q| states?

Build DFA that tracks set of states the NFA is in!
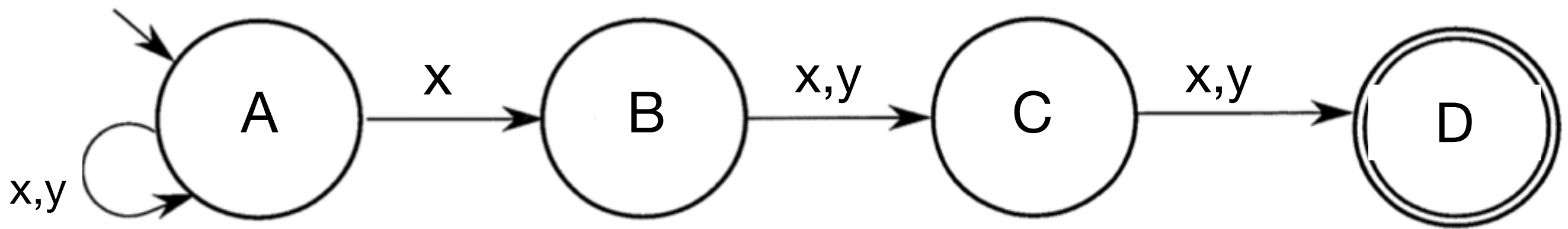
A B C

0 0 0 = {}

0 0 1 = {C}

0 1 0 = {B}

0 1 1 = {B,C}

1 0 0 = {A}

1 0 1 = {A,C}

1 1 0 = {A,B}

1 1 1 = {A,B,C}

Defn: let succ(s,c) be the set of choices the NFA could make in state s with character c

succ(A,x) = {A,B}
succ(A,y) = {A}
succ(B,x) = {C}
succ(B,y) = {C}
succ(C,x) = {D}
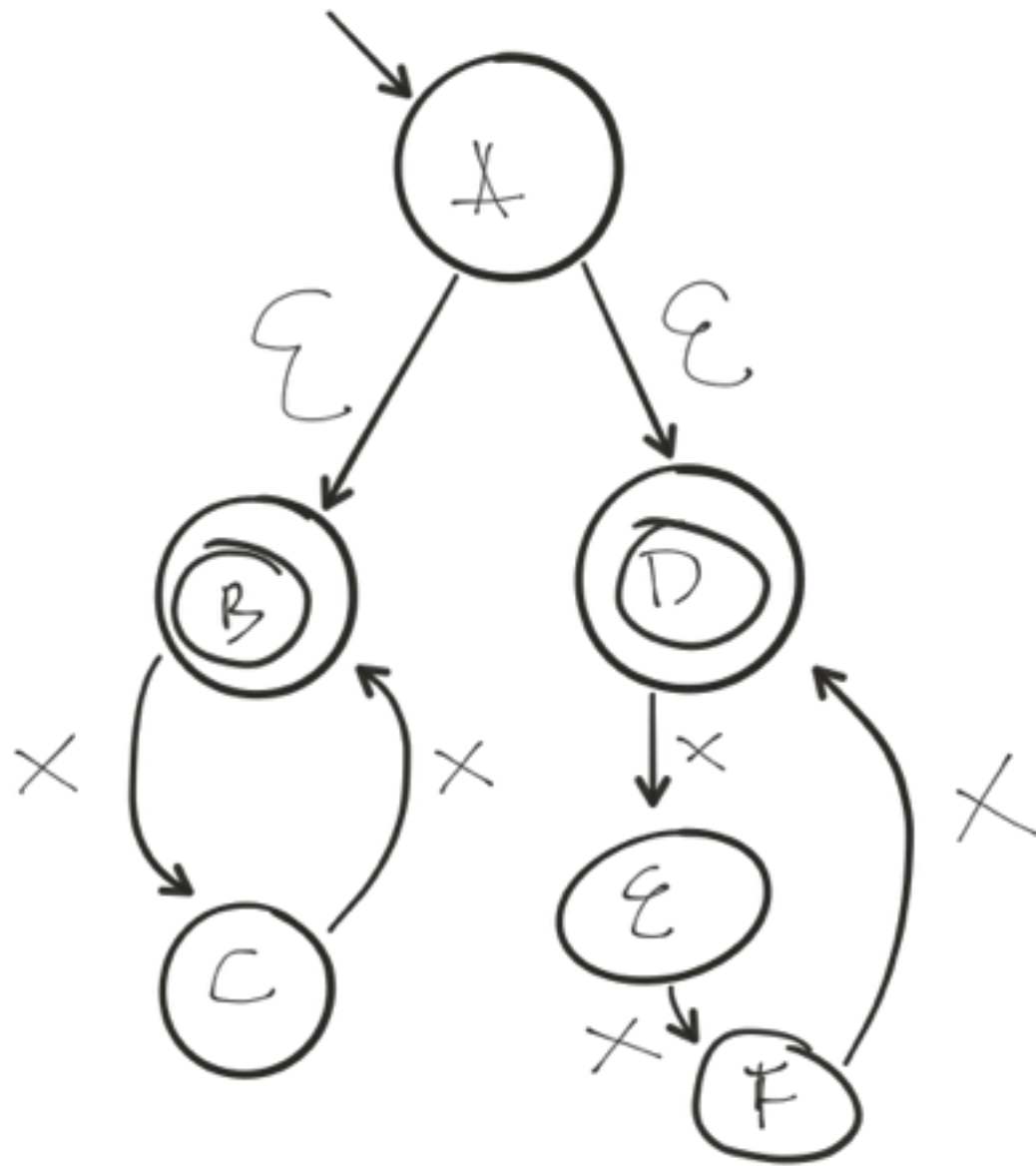succ(C,y) = {D}

Build new DFA M' where Q' = 2^Q

$succ(A,x) = \{A,B\}$
$succ(A,y) = \{A\}$
$succ(B,x) = \{C\}$
$succ(B,y) = \{C\}$
$succ(C,x) = \{D\}$
$succ(C,y) = \{D\}$

To build DFA: Add an edge from state S on character c to state S' if S' represents the union of states that all states in S could possibly transition to on input c

8

ε-transitions

Eg: x^n, where n is even **or** divisible by 3

Useful for taking union of two FSMs

In example, left side accepts even n; right side accepts n divisible by 3

	x	x
AB	C	B
AD	E	F
A		

Eliminating ε-transitions

We want to construct **ε**-free FSM M' that is equivalent to M

**Def:** eclose(s) = set of all states reachable from s in zero or more epsilon transitions

First, make s an accepting state of M' iff eclose(s) contains an accepting state

Second, put s,c—>t in transition relation of M' iff there is a q,c—>t for some q in eclose(s)

Def: eclose(s) = set of all states reachable from s in zero or more epsilon transitions

First, make s an accepting state of M' *iff* eclose(s) contains an accepting state

Second, put s,c—>t in transition relation of M' iff there is a q,c—>t for some q in eclose(s)
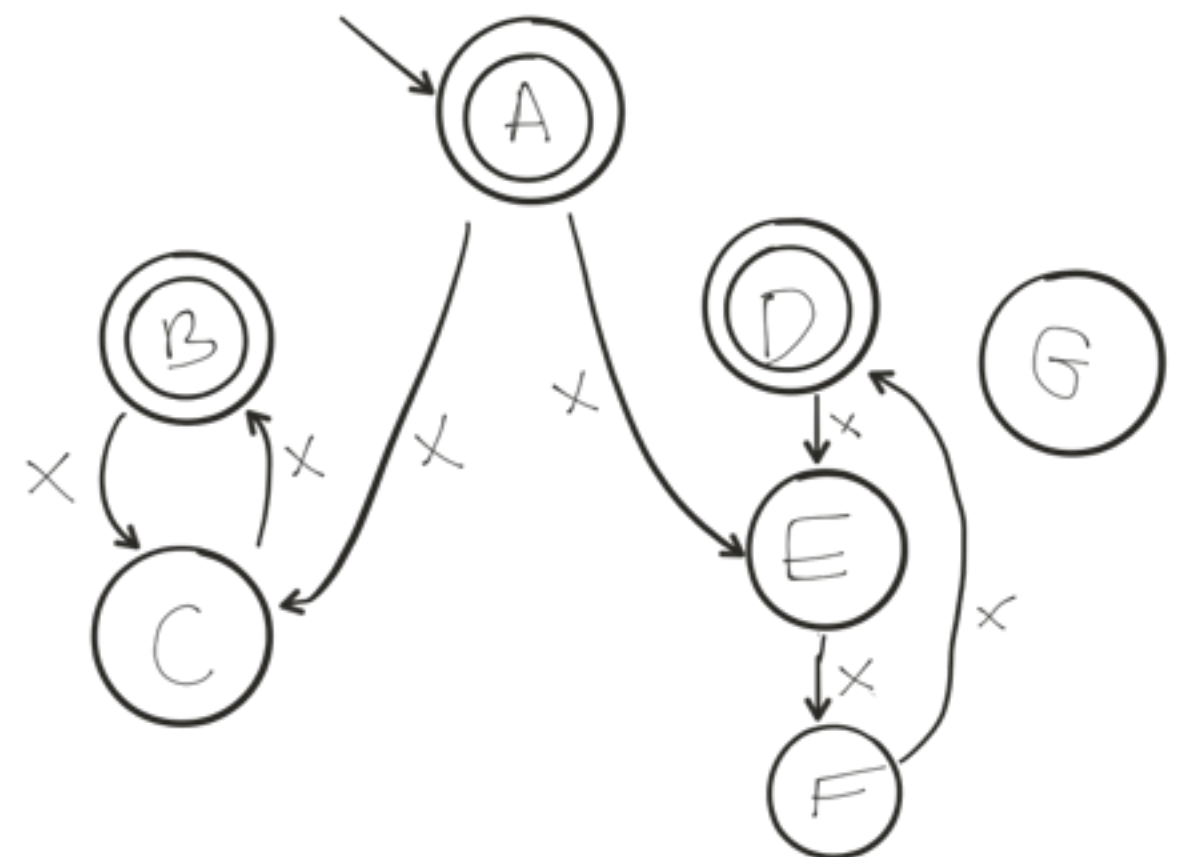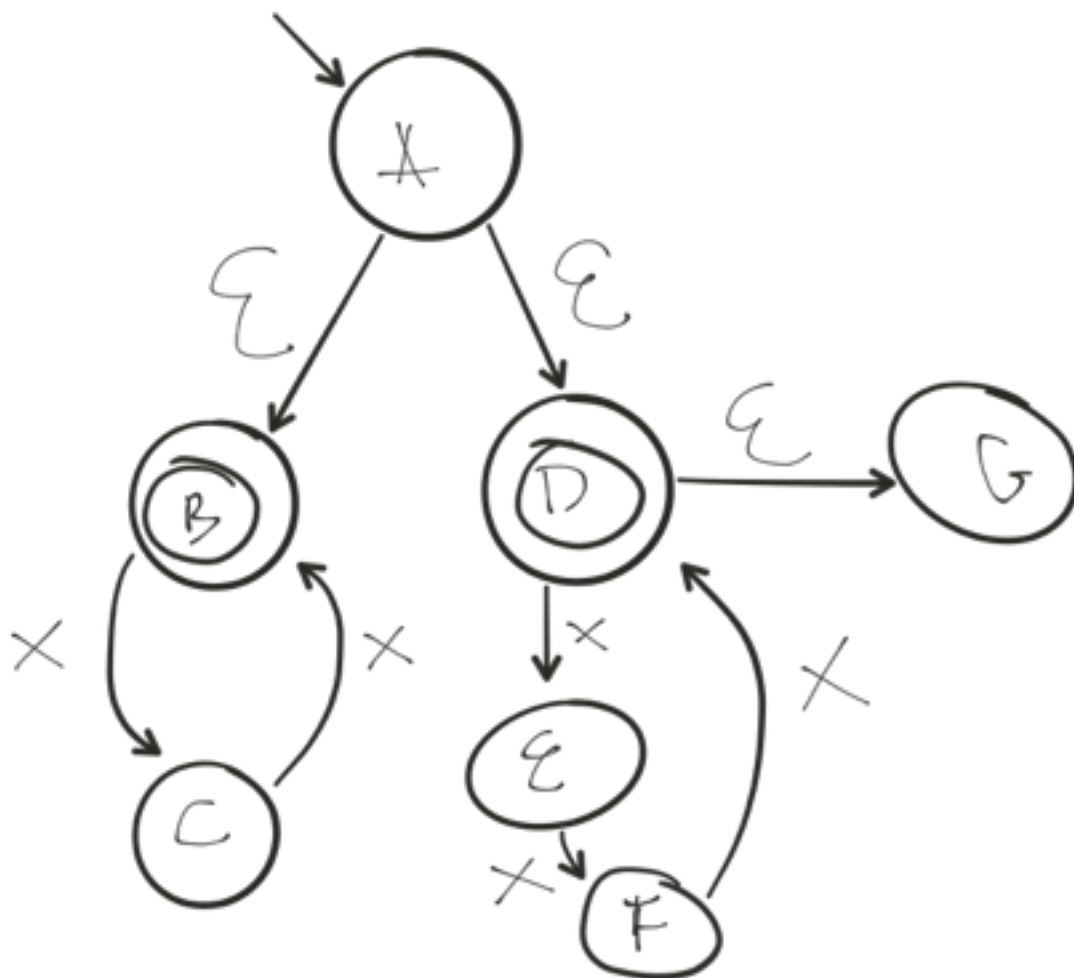
Recap

NFAs and DFAs are equally powerful

  any language definable as an NFA is definable as a DFA

ε-transitions do not add expressiveness to NFAs

  we showed a simple algorithm to remove epsilons

Regular expressions

Pattern describing a language

operands: single characters, epsilon

operators: from low to high precedence

alternation "or": a | b

catenation: a.b, ab, a^3 (which is aaa)

iteration: a* (0 or more a's) aka Kleene star

Regexp, cont'd

Conventions:

a+ is a.a*

letter is a|b|c|d|…|y|z|A|B|…|Z

digit is 0|1|2|…|9

not(x) all characters except x

. is any character

parentheses for grouping, e.g., (ab)*

ε, *ab, abab, ababab*

Regexp, example

Hex strings

    start with 0x or 0X

    followed by one or more hexadecimal digits

    optionally end with l or L

0(x|X)hexdigit+(L|l|**ε**)

    where hexdigit = digit|a|b|c|d|e|f|A|…|F

OR:

(0(x|X)hexdigit_lowercase+(L|l|**ε**)) | (0(x|X)hexdigit_uppercase+(L|l|**ε**))

Regexp, example

Single-line comments in Java/C/C++

// this is a comment

//(not('\n'))*'\n'