

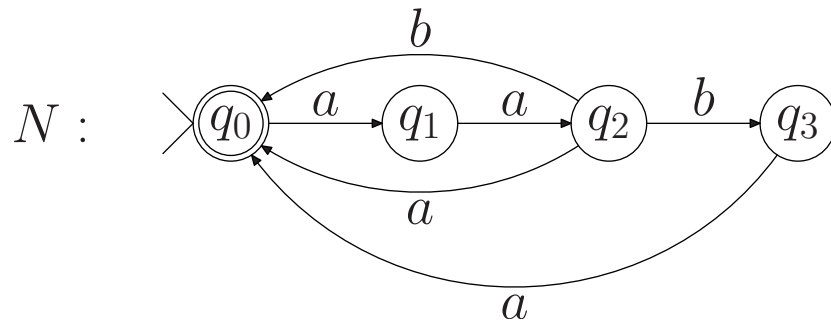
**Harvard CS 121 and CSCI E-121**  
**Lecture 4: Nondeterministic Finite Automata and Closure Properties**

Harry Lewis

September 12, 2013

**Reading:** Sipser, §1.1 and §1.2.

## Example of an NFA



$N = (\{q_0, q_1, q_2, q_3\}, \{a, b\}, \delta, q_0, \{q_0\})$ , where  $\delta$  is given by:

	$a$	$b$	$\varepsilon$
$q_0$	$\{q_1\}$	$\emptyset$	$\emptyset$
$q_1$	$\{q_2\}$	$\emptyset$	$\emptyset$
$q_2$	$\{q_0\}$	$\{q_0, q_3\}$	$\emptyset$
$q_3$	$\{q_0\}$	$\emptyset$	$\emptyset$

## Nondeterministic Finite Automata

An NFA is a 5-tuple  $(Q, \Sigma, \delta, q_0, F)$ , where

- $Q, \Sigma, q_0, F$  are as for DFAs
- $\delta : Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow P(Q)$ .

When in state  $p$  reading symbol  $\sigma$ , can go to any state  $q$  in the set  $\delta(p, \sigma)$ .

- there may be more than one such  $q$ , or
- there may be none (in case  $\delta(p, \sigma) = \emptyset$ ).

Can “jump” from  $p$  to any state in  $\delta(p, \varepsilon)$  without moving the input head.

## Computations by an NFA

$N = (Q, \Sigma, \delta, q_0, F)$  accepts  $w \in \Sigma^*$  if we can write  $w = y_1 y_2 \cdots y_m$  where each  $y_i \in \Sigma \cup \{\varepsilon\}$  and there exist  $r_0, \dots, r_m \in Q$  such that

1.  $r_0 = q_0$ ,
2.  $r_{i+1} \in \delta(r_i, y_{i+1})$  for each  $i = 0, \dots, m - 1$ , and
3.  $r_m \in F$ .

**Nondeterminism:** Given  $N$  and  $w$ , the states  $r_0, \dots, r_m$  are not necessarily determined.

## How to simulate NFAs?

- NFA accepts  $w$  if there is at least one accepting computational path on input  $w$
- But the number of paths may grow exponentially with the length of  $w$ !
- Can exponential search be avoided?

## NFAs vs. DFAs

NFAs seem more “powerful” than DFAs. Are they?

**Theorem:** For every NFA  $N$ , there exists a DFA  $M$  such that  $L(M) = L(N)$ .

**Proof Outline:** Given any NFA  $N$ , to construct a DFA  $M$  such that  $L(M) = L(N)$ :

## NFAs vs. DFAs

NFAs seem more “powerful” than DFAs. Are they?

**Theorem:** For every NFA  $N$ , there exists a DFA  $M$  such that  $L(M) = L(N)$ .

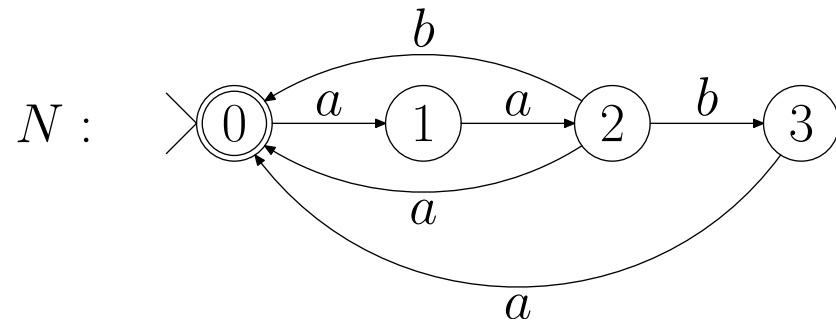
**Proof Outline:** Given any NFA  $N$ , to construct a DFA  $M$  such that  $L(M) = L(N)$ :

Have the DFA keep track, at all times, of all possible states the NFA could be in after reading the same initial part of the input string.

I.e., the states of  $M$  are sets of states of  $N$ , and  $\delta_M^*(R, w)$  is the set of all states  $N$  could reach after reading  $w$ , starting from a state in  $R$ .

## Example of the SUBSET CONSTRUCTION

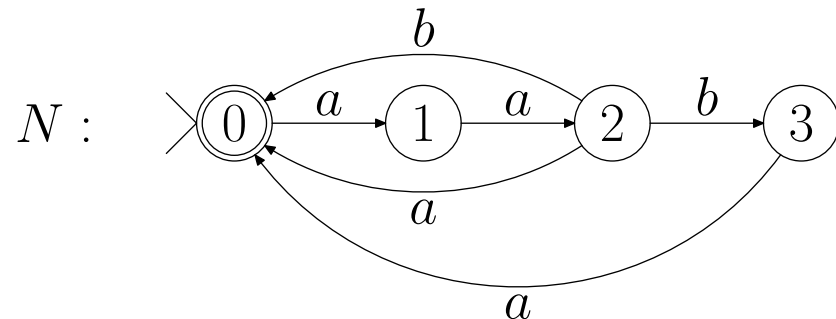
NFA  $N$  for  $\{x_1x_2\cdots x_k : k \geq 0 \text{ and each } x_i \in \{aab, aaba, aaa\}\}$ .



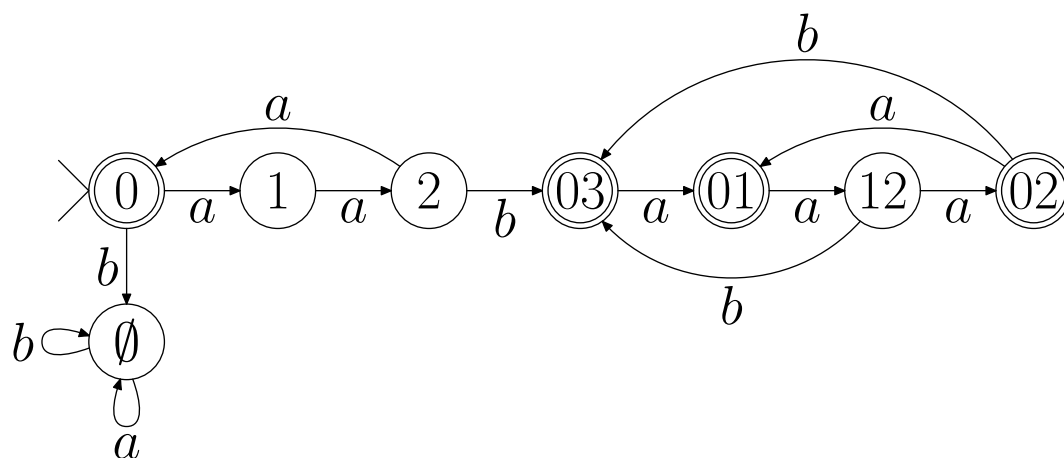
$N$  starts in state 0 so we will construct a DFA  $M$  starting in state  $\{0\}$ .

## Example of the SUBSET CONSTRUCTION

NFA  $N$  for  $\{x_1x_2\cdots x_k : k \geq 0 \text{ and each } x_i \in \{aab, aaba, aaa\}\}$ .



$N$  starts in state 0 so we will construct a DFA  $M$  starting in state  $\{0\}$ . Here it is:



All other transitions are to the “dead state”  $\emptyset$ . The other states are unreachable, though technically must be defined. Final states are all those containing 0, the final state of  $M$ .

## Formal Construction of DFA $M$ from NFA $N = (Q, \Sigma, \delta, q_0, F)$

On the assumption that  $\delta(p, \varepsilon) = \emptyset$  for all states  $p$ .

(i.e., we assume no  $\varepsilon$ -transitions, just to simplify things a bit)

$M = (Q', \Sigma, \delta', q'_0, F')$  where

$$Q' = P(Q)$$

$$q'_0 = \{q_0\}$$

$$F' = \{R \subseteq Q : R \cap F \neq \emptyset\} \text{ (that is, } R \in Q')$$

$$\delta'(R, \sigma) = \{q \in Q : q \in \delta(r, \sigma) \text{ for some } r \in R\}$$

$$= \bigcup_{r \in R} \delta(r, \sigma)$$

## Proving that the construction works

**Claim:** For every string  $w$ , running  $M$  on input  $w$  ends in the state

$\{q \in Q : \text{some computation of } N \text{ on input } w \text{ ends in state } q\}$ .

**Pf:** By induction on  $|w|$ .

Can be extended to work even for NFAs with  $\varepsilon$ -transitions.

“THE SUBSET CONSTRUCTION”

# Rabin & Scott, "Finite Automata and Their Decision Problems," 1959

1976 – Michael O. Rabin *See the ACM Author Profile in the Digital Library*

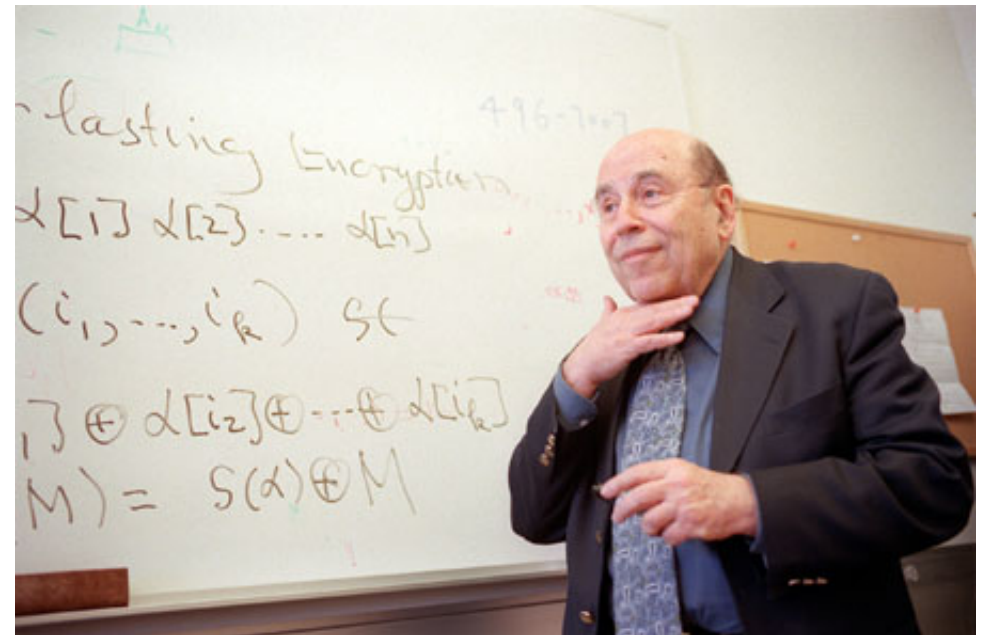
## Citation

For their joint paper "Finite Automata and Their Decision Problem," which introduced the idea of nondeterministic machines, which has proved to be an enormously valuable concept. Their (Scott & Rabin) classic paper has been a continuous source of inspiration for subsequent work in this field.

## Biographical Information



Michael O. Rabin (born 1931 in Breslau, Germany) is a noted computer scientist and a recipient of the Turing Award, the most prestigious award in the field.



## Nondeterminism gives us a new programming tool

- Strings that begin with *aaba*

## Nondeterminism gives us a new programming tool

- Strings that begin with *aaba*
- Strings that end with *aaba*

## Nondeterminism gives us a new programming tool

- Strings that begin with *aaba*
- Strings that end with *aaba*
- Strings that begin or end with *aaba*

## Nondeterminism gives us a new programming tool

- Strings that begin with *aba*
- Strings that end with *aba*
- Strings that begin or end with *aba*
- Strings that have *aba* as a substring anywhere

## Closure Properties

**Theorem:** The class of regular languages is closed under:

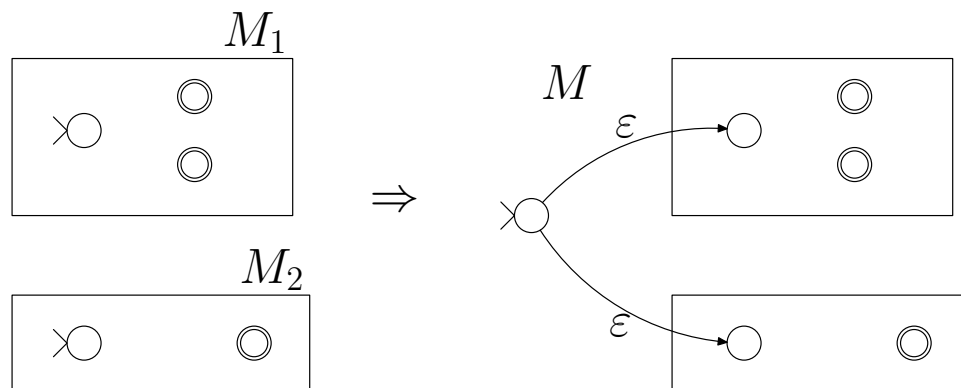
- Union:  $L_1 \cup L_2$
- Concatenation:  $L_1 \circ L_2 = \{xy : x \in L_1 \text{ and } y \in L_2\}$
- Kleene \*:  $L_1^* = \{x_1x_2 \cdots x_k : k \geq 0 \text{ and each } x_i \in L_1\}$
- Complement:  $\overline{L_1}$
- Intersection:  $L_1 \cap L_2$

## Closure Properties

**Theorem:** The class of regular languages is closed under:

- Union:  $L_1 \cup L_2$
- Concatenation:  $L_1 \circ L_2 = \{xy : x \in L_1 \text{ and } y \in L_2\}$
- Kleene \*:  $L_1^* = \{x_1x_2 \cdots x_k : k \geq 0 \text{ and each } x_i \in L_1\}$
- Complement:  $\overline{L_1}$
- Intersection:  $L_1 \cap L_2$

Union: If  $L_1$  and  $L_2$  are regular, then  $L_1 \cup L_2$  is regular.



$M$  has the states and transitions of  $M_1$  and  $M_2$  plus a new start state  $\epsilon$ -transitioning to the old start state

# Concatenation, Kleene \*, Complementation

Concatenation:

$$L(M) = L(M_1) \circ L(M_2)$$

Kleene \*:

$$L(M) = L(M_1)^*$$

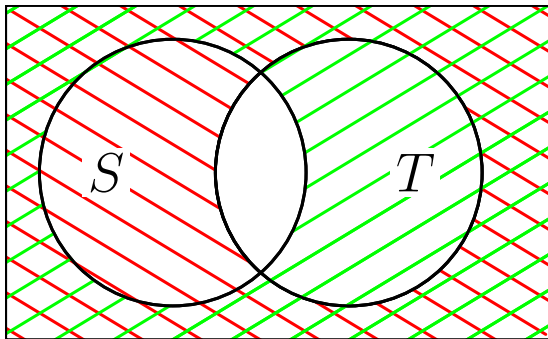
Complement:

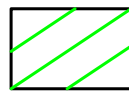
$$L(M) = \overline{L(M_1)}$$


## Closure under Intersection

### Intersection

$$S \cap T = \overline{\overline{S} \cup \overline{T}}$$



 =  $\overline{S}$

 =  $\overline{T}$

Hence closure under union and complement implies closure under intersection

## A more constructive and direct proof of closure under intersection

Better way (“Cross Product Construction”):

From DFAs  $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$  and  $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ , construct  $M = (Q, \Sigma, \delta, q_0, F)$ :

$$Q = Q_1 \times Q_2$$

$$F = F_1 \times F_2$$

$$\delta(\langle r_1, r_2 \rangle, \sigma) = \langle \delta_1(r_1, \sigma), \delta_2(r_2, \sigma) \rangle$$

$$q_0 = \langle q_1, q_2 \rangle$$

Then  $L(M_1) \cap L(M_2) = L(M)$

## Some Efficiency Considerations

The subset construction shows that any  $n$ -state NFA can be implemented as a  $2^n$ -state DFA.

NFA States	DFA States
4	16
10	1024
100	$2^{100}$
1000	$2^{1000} \gg$ the number of particles in the universe

How to implement this construction on ordinary digital computer?

NFA states

$1, \dots, n$

DFA state bit vector

0	1	1	0	...	1
1	2				$n$

## Is this construction the best we can do?

Could there be a construction that always produces an  $n^2$  state DFA for example?

**Theorem:** For every  $n \geq 1$ , there is a language  $L_n$  such that

1. There is an  $(n + 1)$ -state NFA recognizing  $L_n$ .
2. There is no DFA recognizing  $L_n$  with fewer than  $2^n$  states.

**Conclusion:** For finite automata, nondeterminism provides an *exponential savings* over determinism (in the worst case).

## Proving that exponential blowup is sometimes unavoidable

(Could there be a construction that always produces an  $n^2$  state DFA for example?)

Consider (for some fixed  $n=17$ , say)

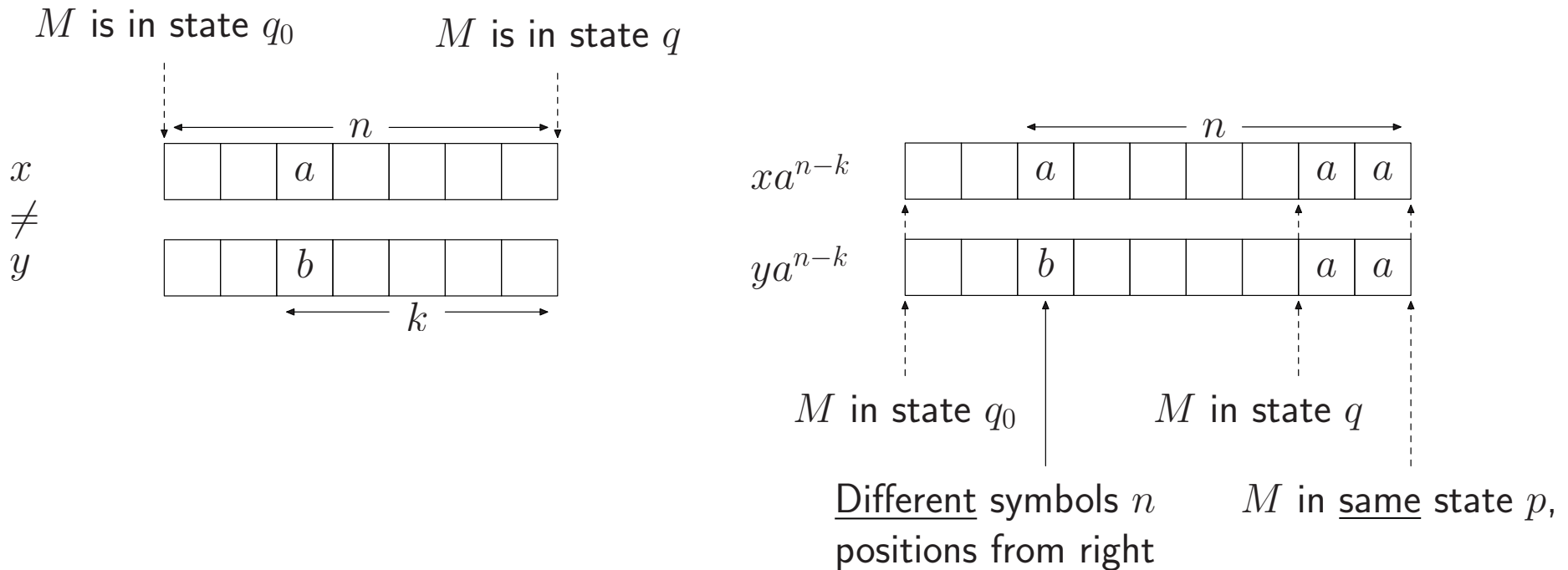
$$L_n = \{w \in \{a, b\}^* : \text{the } n\text{th symbol} \\ \text{from the right end of } w \text{ is an } a\}$$

- There is an  $(n + 1)$ -state NFA that accepts  $L_n$ .
- There is no DFA that accepts  $L_n$  and has  $< 2^n$  states

## A “Fooling Argument”

- Suppose a DFA  $M$  has  $< 2^n$  states, and  $L(M) = L_n$
- There are  $2^n$  strings of length  $n$ .
- By the pigeonhole principle, two such strings  $x \neq y$  must drive  $M$  to the same state  $q$ .
- Suppose  $x$  and  $y$  differ at the  $k^{\text{th}}$  position from the right end (one has  $a$ , the other has  $b$ )  
( $k = 1, 2, \dots, \text{or } n$ )
- $M$  must treat  $xa^{n-k}$  and  $ya^{n-k}$  identically (accept both or reject both). These strings differ at position  $n$  from the right end.
- So  $L(M) \neq L_n$ , contradiction. QED.

## Illustration of the fooling argument



- $x$  and  $y$  are different strings  
 (so there is a position  $k$  where one has  $a$  and the other has  $b$ )
- But both strings drive  $M$  from  $s$  to the same state  $q$

## What the argument proves

- This shows that the subset construction is within a factor of 2 of being optimal
- In fact it is optimal, i.e., as good as we can do in the *worst case*.
- Still, in many cases, the “generate-states-as-needed” method yields a DFA with  $\ll 2^n$  states  
(e.g. if the NFA was deterministic to begin with!)