

# Stringology and Finite Automata

**Jan Holub**

Prague Stringology Club

Department of Computer Science and Engineering  
Czech Technical University in Prague, Czech Republic

<http://www.stringology.org>

# Motivation

1. Most of pattern matching problems are regular languages.
2. Absence of finite automata approach for these problems (with few exceptions like KMP, AC, BM, CW).
3. Many pattern matching algorithms were developed without of finite automata approach but actually they simulate run of a finite automaton.

# Contents

1. Finite Automata
2. Basic Simulation Method
3. Dynamic Programming
4. Bit Parallelism
5. Work in Progress in Prague Stringology Club

# Finite Automata

Deterministic Finite Automaton (DFA)  $M_{DFA} = (Q, \Sigma, \delta, q_0, F)$

- $Q$  is a finite set of states,
- $\Sigma$  is a finite input alphabet,
- $\delta$  is a mapping  $Q \times \Sigma \rightarrow Q$ ,
- $q_0 \in Q$  is an initial state,
- $F$  is a set of final states.

# Finite Automata

Deterministic Finite Automaton (DFA)  $M_{DFA} = (Q, \Sigma, \delta, q_0, F)$

- $Q$  is a finite set of states,
- $\Sigma$  is a finite input alphabet,
- $\delta$  is a mapping  $Q \times \Sigma \rightarrow Q$ ,
- $q_0 \in Q$  is an initial state,
- $F$  is a set of final states.

Nondeterm. Finite Automaton (NFA)  $M_{NFA} = (Q, \Sigma, \delta, q_0, F)$

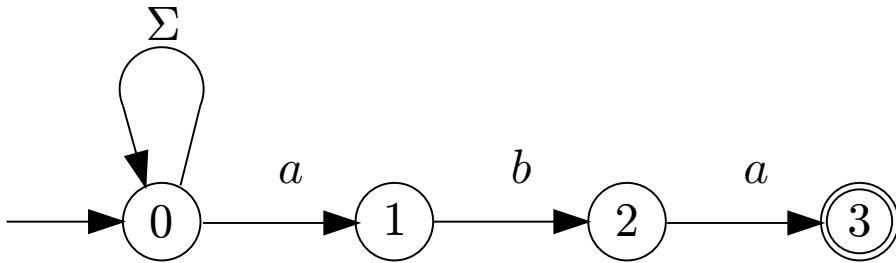
- $Q$  is a finite set of states,
- $\Sigma$  is a finite input alphabet,
- $\delta$  is a mapping  $Q \times (\Sigma \cup \{\varepsilon\}) \rightarrow \mathcal{P}(Q)$ ,
- $q_0 \in Q$  is an initial state,
- $F$  is a set of final states.

# Usage of *NFA*

Two possibilities of using *NFA*:

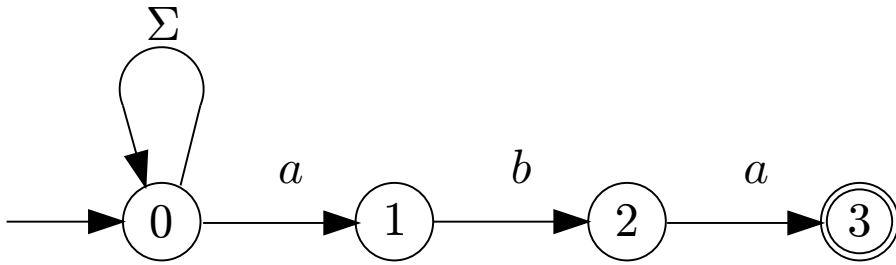
1. to transform *NFA* to an equivalent *DFA*:  
running time  $\mathcal{O}(n)$  (not considering the transformation)  
but memory  $\mathcal{O}(2^{|Q_{NFA}|})$ ,
2. to simulate the run of *NFA* in deterministic way:  
slower but less memory.

# Transforming *NFA* to *DFA*



$\delta_{NFA}$	<i>a</i>	<i>b</i>	<i>c</i>
0	{0, 1}	{0}	{0}
1		{2}	
2	{3}		
3			

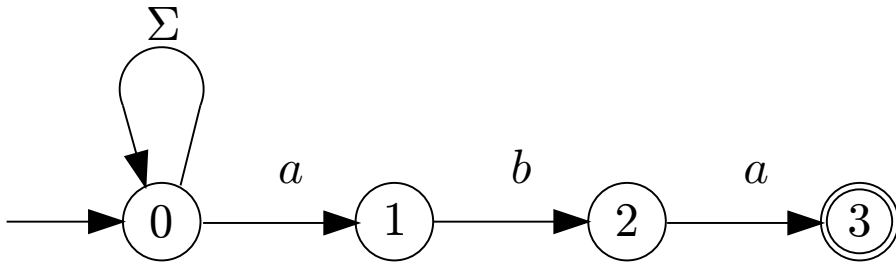
# Transforming *NFA* to *DFA*



$\delta_{NFA}$	$a$	$b$	$c$
0	{0, 1}	{0}	{0}
1		{2}	
2	{3}		
3			

$\delta_{DFA}$	$a$	$b$	$c$
$0 = [0]$			

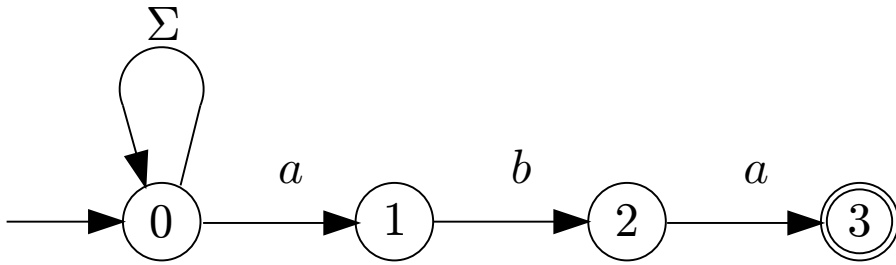
# Transforming *NFA* to *DFA*



$\delta_{NFA}$	<i>a</i>	<i>b</i>	<i>c</i>
0	{0, 1}	{0}	{0}
1		{2}	
2	{3}		
3			

$\delta_{DFA}$	<i>a</i>	<i>b</i>	<i>c</i>
$0 = [0]$	[0, 1]	[0]	[0]
$1 = [0, 1]$			

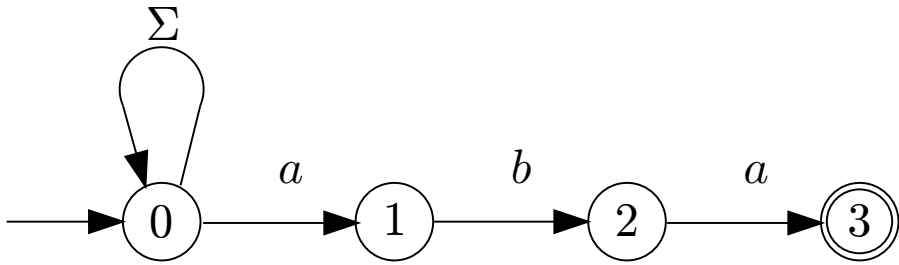
# Transforming *NFA* to *DFA*



$\delta_{NFA}$	<i>a</i>	<i>b</i>	<i>c</i>
0	{0, 1}	{0}	{0}
1		{2}	
2	{3}		
3			

$\delta_{DFA}$	<i>a</i>	<i>b</i>	<i>c</i>
$0 = [0]$	[0, 1]	[0]	[0]
$1 = [0, 1]$	[0, 1]	[0, 2]	[0]
$2 = [0, 2]$			

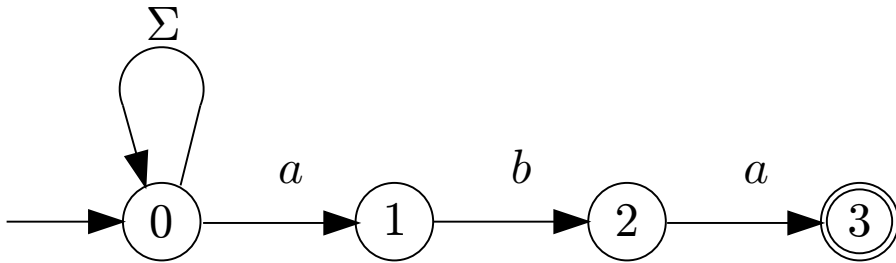
# Transforming *NFA* to *DFA*



$\delta_{NFA}$	<i>a</i>	<i>b</i>	<i>c</i>
0	{0, 1}	{0}	{0}
1		{2}	
2	{3}		
3			

$\delta_{DFA}$	<i>a</i>	<i>b</i>	<i>c</i>
$0 = [0]$	[0, 1]	[0]	[0]
$1 = [0, 1]$	[0, 1]	[0, 2]	[0]
$2 = [0, 2]$	[0, 1, 3]	[0]	[0]
$3 = [0, 1, 3]$			

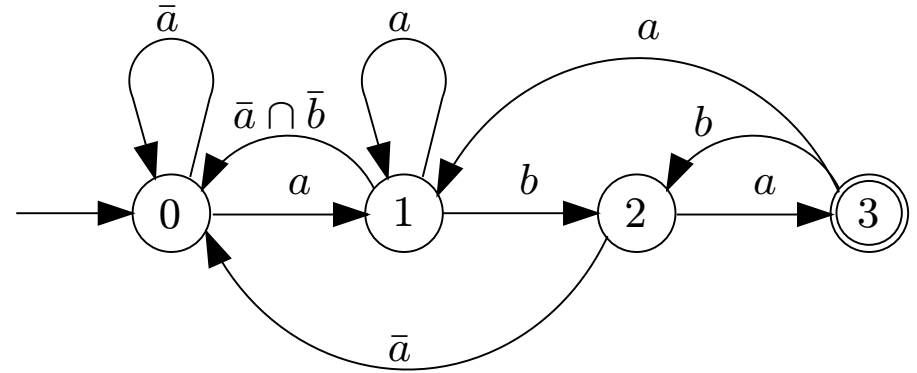
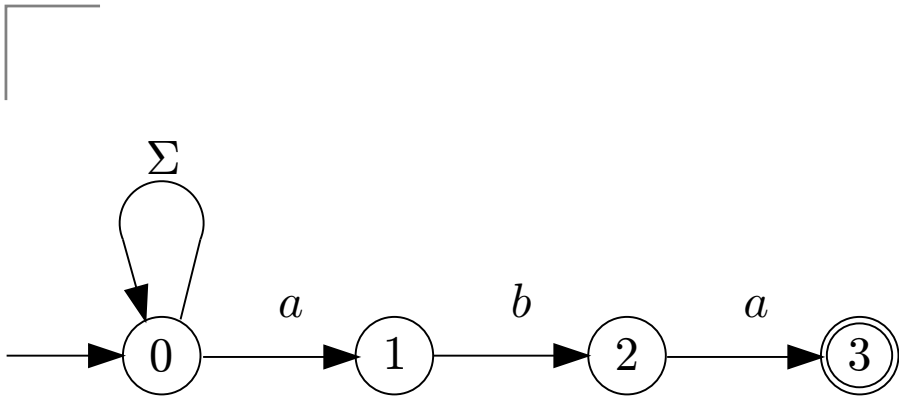
# Transforming *NFA* to *DFA*



$\delta_{NFA}$	<i>a</i>	<i>b</i>	<i>c</i>
0	{0, 1}	{0}	{0}
1		{2}	
2	{3}		
3			

$\delta_{DFA}$	<i>a</i>	<i>b</i>	<i>c</i>
$0 = [0]$	[0, 1]	[0]	[0]
$1 = [0, 1]$	[0, 1]	[0, 2]	[0]
$2 = [0, 2]$	[0, 1, 3]	[0]	[0]
$3 = [0, 1, 3]$	[0, 1]	[0, 2]	[0]

# Transforming *NFA* to *DFA*



$\delta_{NFA}$	$a$	$b$	$c$
0	{0, 1}	{0}	{0}
1		{2}	
2	{3}		
3			

$\delta_{DFA}$	$a$	$b$	$c$
$0 = [0]$	[0, 1]	[0]	[0]
$1 = [0, 1]$	[0, 1]	[0, 2]	[0]
$2 = [0, 2]$	[0, 1, 3]	[0]	[0]
$3 = [0, 1, 3]$	[0, 1]	[0, 2]	[0]

# Basic Simulation Method (BSM)

## Algorithm

**Input:** *NFA*  $M = (Q, \Sigma, \delta, q_0, F)$ , input text  $T = t_1 t_2 \dots t_n$ .

**Output:** Output of run of *NFA*.

**Method:** Set  $S$  of active states is used.

$S \leftarrow \varepsilon\text{CLOSURE}(\{q_0\}), i \leftarrow 1$

**while**  $i \leq n$  **and**  $S \neq \emptyset$  **do**

$S \leftarrow \bigcup_{q \in S} \varepsilon\text{CLOSURE}(\delta(q, t_i))$

**if**  $S \cap F \neq \emptyset$  **then**

**write**(information associated with each final state in  $S$ )

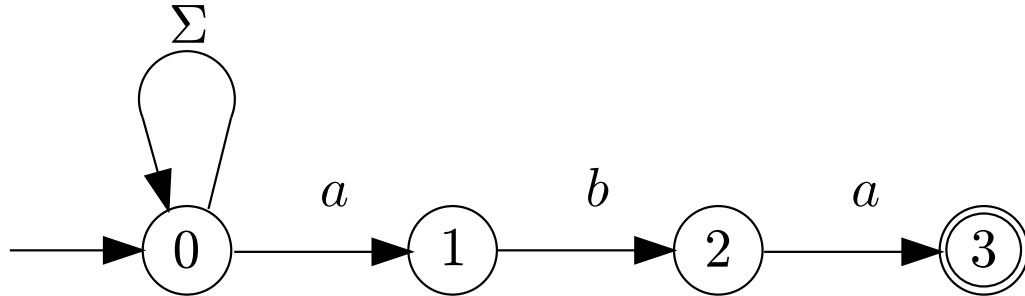
**endif**

$i \leftarrow i + 1$

**endwhile**

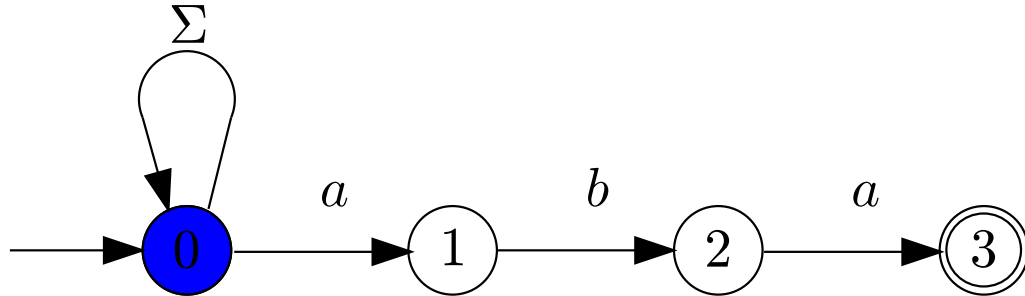
# Basic Simulation Method (BSM)

$T = aabaababa$ ,  $P = aba$



# Basic Simulation Method (BSM)

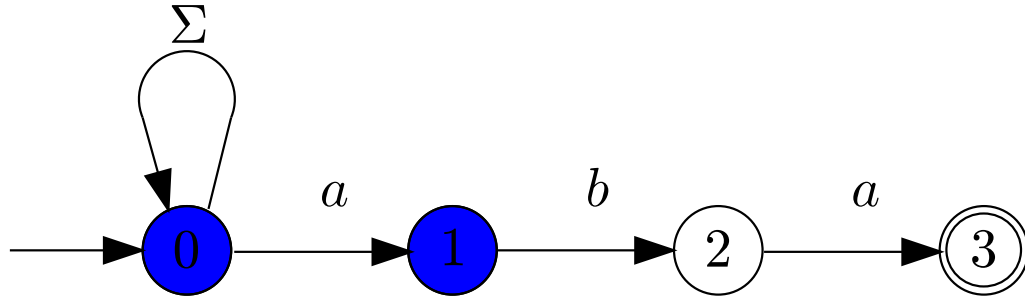
$T = aabaababa$ ,  $P = aba$



$T$	-	$a$
$S$	0	
output		

# Basic Simulation Method (BSM)

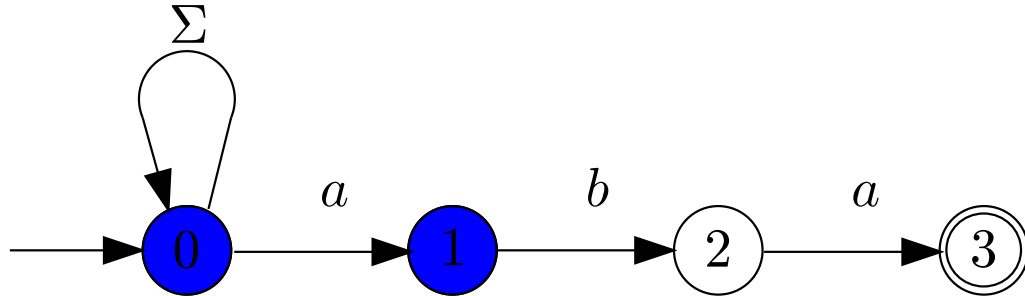
$T = aabaababa$ ,  $P = aba$



$T$	-	$a$	$a$
$S$	0	0 1	
output			

# Basic Simulation Method (BSM)

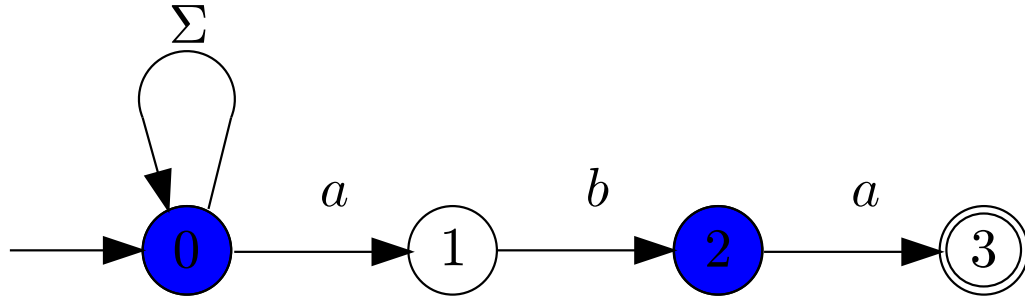
$T = aabaababa$ ,  $P = aba$



$T$	-	$a$	$a$	$b$
$S$	0	0 1	0 1	
output				

# Basic Simulation Method (BSM)

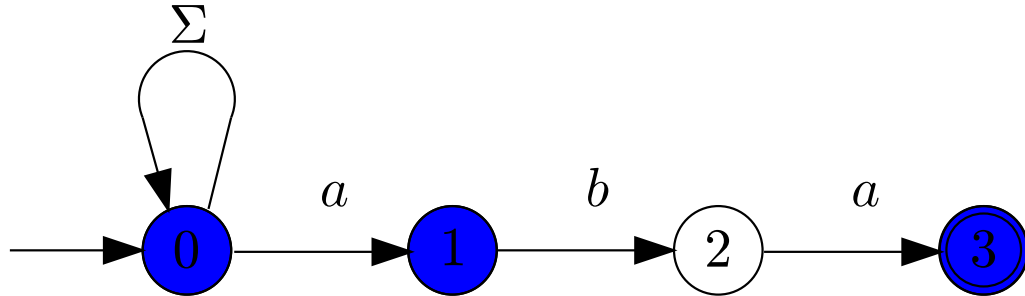
$T = aabaababa$ ,  $P = aba$



$T$	-	$a$	$a$	$b$	$a$
$S$	0	0	0	0	
		1	1	2	
output					

# Basic Simulation Method (BSM)

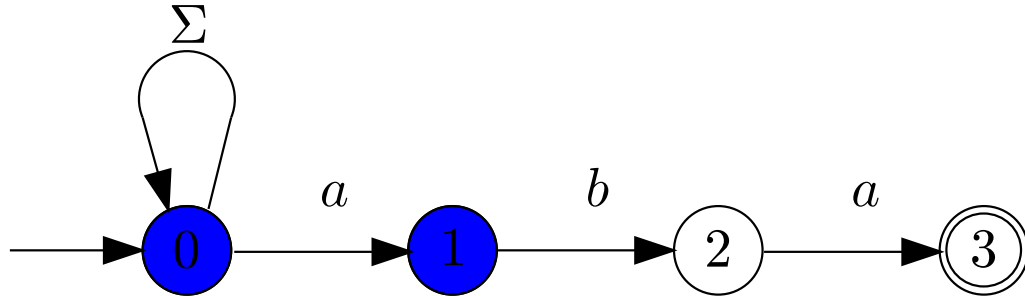
$T = aabaababa$ ,  $P = aba$



$T$	-	$a$	$a$	$b$	$a$	$a$
$S$	0	0	0	0	0	
		1	1	2	1	
					3	
output					4	

# Basic Simulation Method (BSM)

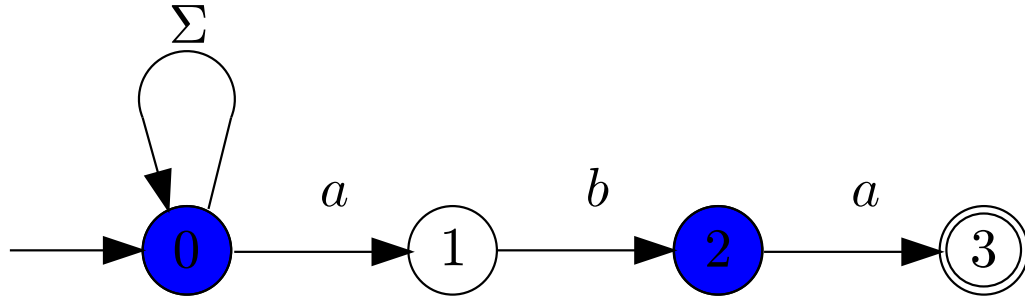
$T = aabaababa$ ,  $P = aba$



$T$	-	$a$	$a$	$b$	$a$	$a$	$b$
$S$	0	0	0	0	0	0	
		1	1	2	1	1	
					3		
output					4		

# Basic Simulation Method (BSM)

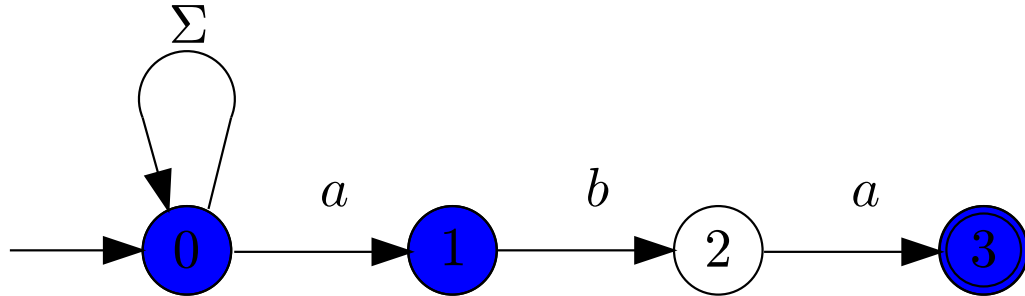
$T = aabaababa, P = aba$



$T$	-	$a$	$a$	$b$	$a$	$a$	$b$	$a$
$S$	0	0	0	0	0	0	0	
		1	1	2	1	1	2	
					3			
output					4			

# Basic Simulation Method (BSM)

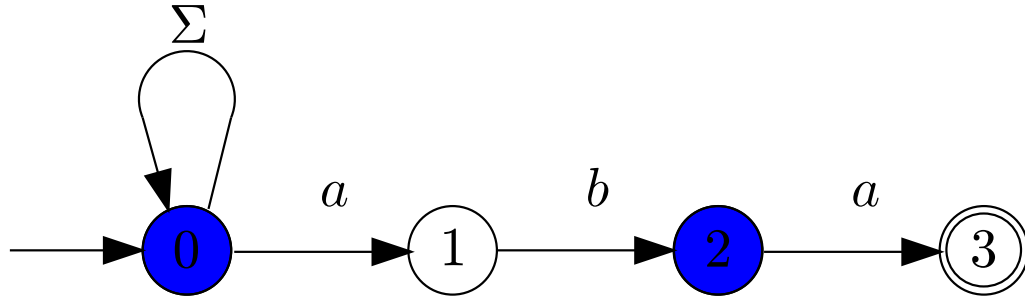
$T = aabaababa$ ,  $P = aba$



$T$	-	$a$	$a$	$b$	$a$	$a$	$b$	$a$	$b$
$S$	0	0	0	0	0	0	0	0	
		1	1	2	1	1	2	1	
					3			3	
output					4			7	

# Basic Simulation Method (BSM)

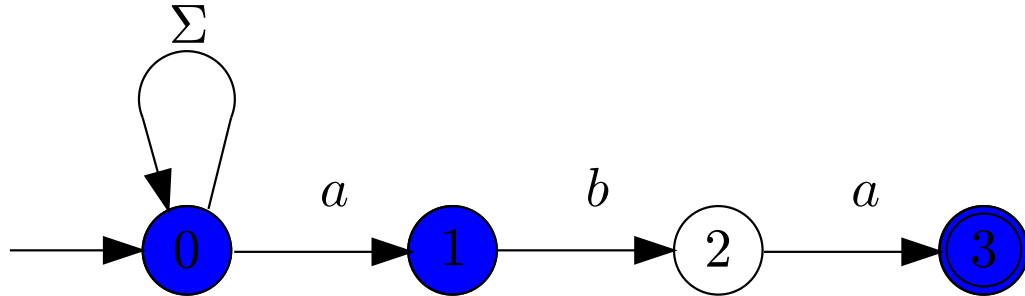
$T = aabaababa$ ,  $P = aba$



$T$	-	$a$	$a$	$b$	$a$	$a$	$b$	$a$	$b$	$a$
$S$	0	0	0	0	0	0	0	0	0	
		1	1	2	1	1	2	1	2	
					3			3		
output					4			7		

# Basic Simulation Method (BSM)

$T = aabaababa$ ,  $P = aba$



$T$	-	$a$	$a$	$b$	$a$	$a$	$b$	$a$	$b$	$a$
$S$	0	0	0	0	0	0	0	0	0	0
		1	1	2	1	1	2	1	2	1
					3			3		3
output					4			7		9

# Bit Implementation of BSM

$S_i$ 

0	1	0	1
---	---	---	---



$S_{i+1}$ 

1	0	1	1
---	---	---	---

$\delta$	$a$				$b$				$\dots$	$\varepsilon$			
0	1	0	1	0	1	1	0	1	$\dots$	1	1	0	0
1	0	1	1	0	1	0	1	0	$\dots$	0	1	0	1
$\vdots$	$\vdots$				$\vdots$				$\ddots$	$\vdots$			

# Bit Implementation of BSM

## Algorithm

**Input:**  $|Q|$ , transition table  $\delta$ , bit vector  $\mathcal{F}$ ,  $T = t_1 t_2 \dots t_n$ .

**Output:** Output of the run of *NFA*.

**Method:** Vector  $S_i$  of active states is used.

$S_0 \leftarrow [100 \dots 0]$ ,  $i \leftarrow 1$

**while**  $i \leq n$  **and**  $S_{i-1} \neq [00 \dots 0]$  **do**

$S_i \leftarrow [00 \dots 0]$

**for**  $j \leftarrow 0, 1, \dots, |Q| - 1$  **do**

**if**  $S_{i-1,j} = 1$  **then**  $S_i \leftarrow S_i \text{ OR } \delta[j, t_i]$

**endfor**

**if**  $(S_i \text{ AND } \mathcal{F}) \neq [00 \dots 0]$  **then write**(‘*NFA* has reached a final state’)

**endif**

$i \leftarrow i + 1$

**endwhile**

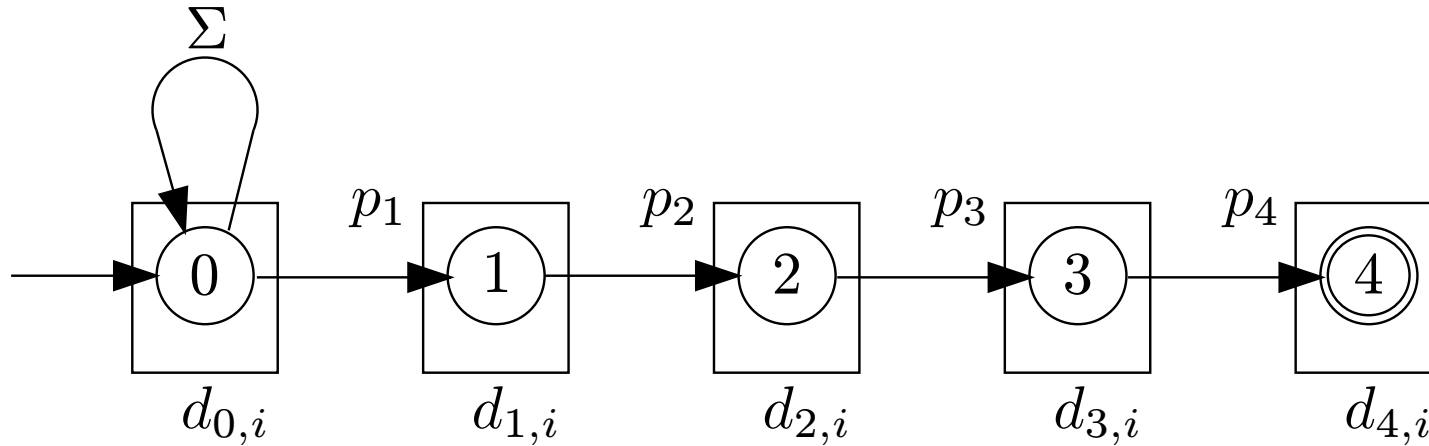
# Bit Implementation of BSM

Complexities:

- time  $\mathcal{O}(|Q|n \lceil \frac{|Q|}{w} \rceil)$ ,
- space  $\mathcal{O}(|Q||\Sigma| \lceil \frac{|Q|}{w} \rceil)$ ,

where  $w$  is bit-length of computer word.

# Dyn. Programming—Exact String M.



Vector of integers—one integer for each depth of *NFA*.

# Dyn. Programming—Exact String M.

$$d_{j,0} = 1, \quad 0 < j \leq m,$$

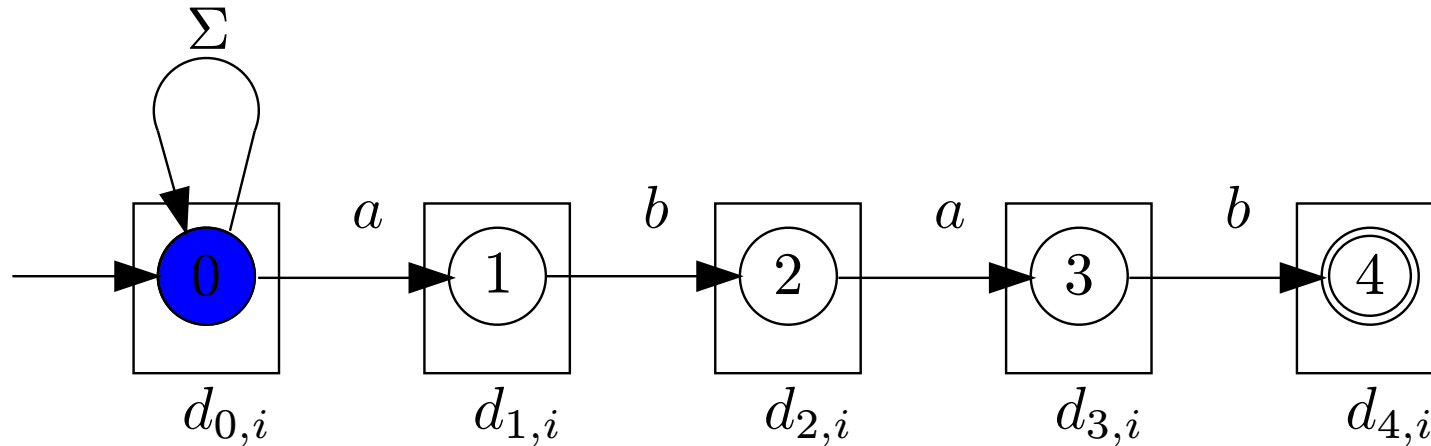
$$d_{0,i} = 0, \quad 0 < i \leq n,$$

$$d_{j,i} = \begin{cases} \text{if } t_i = p_j \text{ then } d_{j-1,i-1} \\ \text{else } 1 \end{cases} \quad 0 < i \leq n, 0 < j \leq m.$$

<i>D</i>		<i>a</i>	<i>a</i>	<i>b</i>	<i>a</i>	<i>b</i>	<i>a</i>	<i>a</i>	<i>b</i>	<i>a</i>	<i>b</i>
	0	0	0	0	0	0	0	0	0	0	0
<i>a</i>	1	0	0	1	0	1	0	0	1	0	1
<i>b</i>	1	1	1	0	1	0	1	1	0	1	0
<i>a</i>	1	1	1	1	0	1	0	1	1	0	1
<i>b</i>	1	1	1	1	1	<b>0</b>	1	1	1	1	<b>0</b>

# Dyn. Programming—Exact String M.

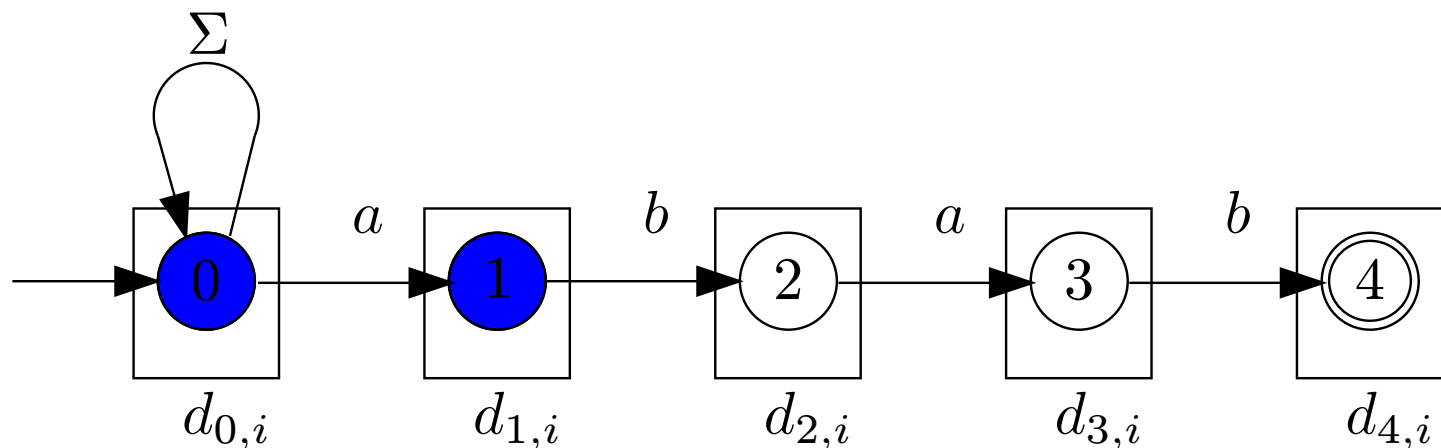
$P = abab$ ,  $T = aababaabab$



	$D$		$a$
	$d_{0,i}$	0	
$a$	$d_{1,i}$	1	
$b$	$d_{2,i}$	1	
$a$	$d_{3,i}$	1	
$b$	$d_{4,i}$	1	

# Dyn. Programming—Exact String M.

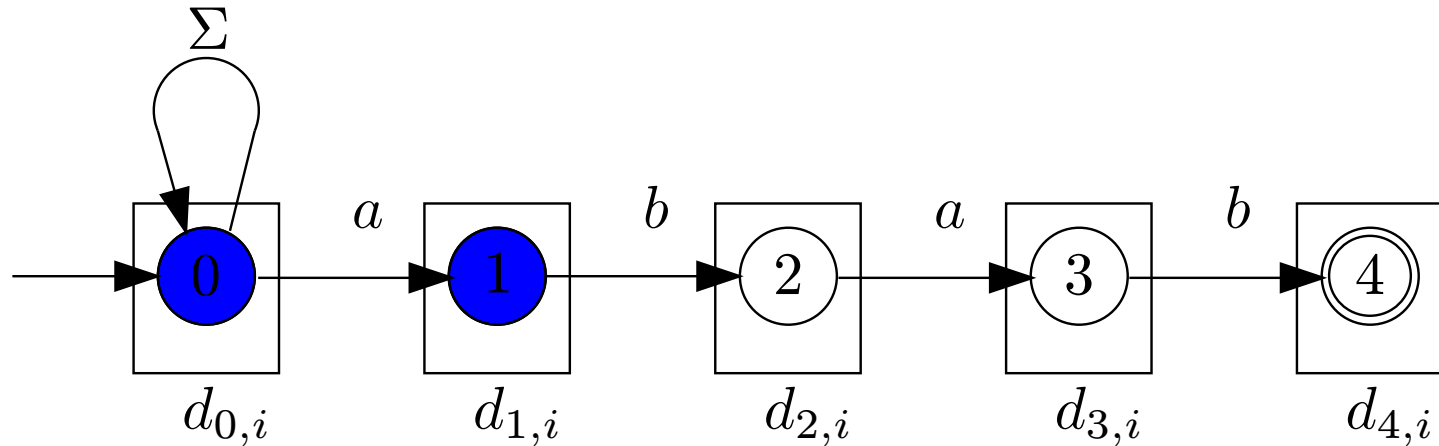
$P = abab$ ,  $T = aababaabab$



	$D$		$a$	$a$
	$d_{0,i}$	0	0	
$a$	$d_{1,i}$	1	0	
$b$	$d_{2,i}$	1	1	
$a$	$d_{3,i}$	1	1	
$b$	$d_{4,i}$	1	1	

# Dyn. Programming—Exact String M.

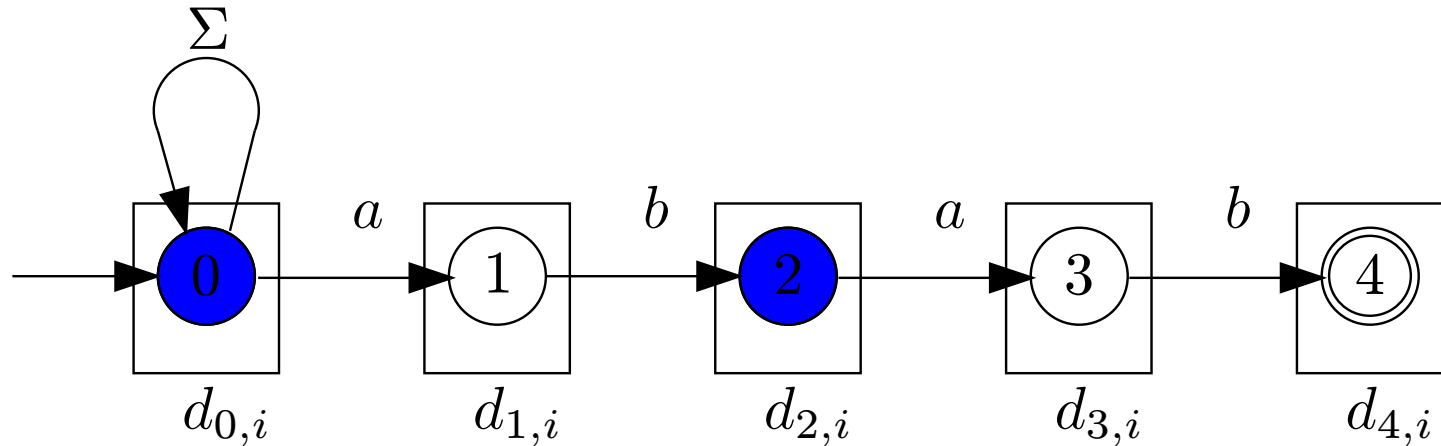
$P = abab$ ,  $T = aababaabab$



	$D$		$a$	$a$	$b$
	$d_{0,i}$	0	0	0	
$a$	$d_{1,i}$	1	0	0	
$b$	$d_{2,i}$	1	1	1	
$a$	$d_{3,i}$	1	1	1	
$b$	$d_{4,i}$	1	1	1	

# Dyn. Programming—Exact String M.

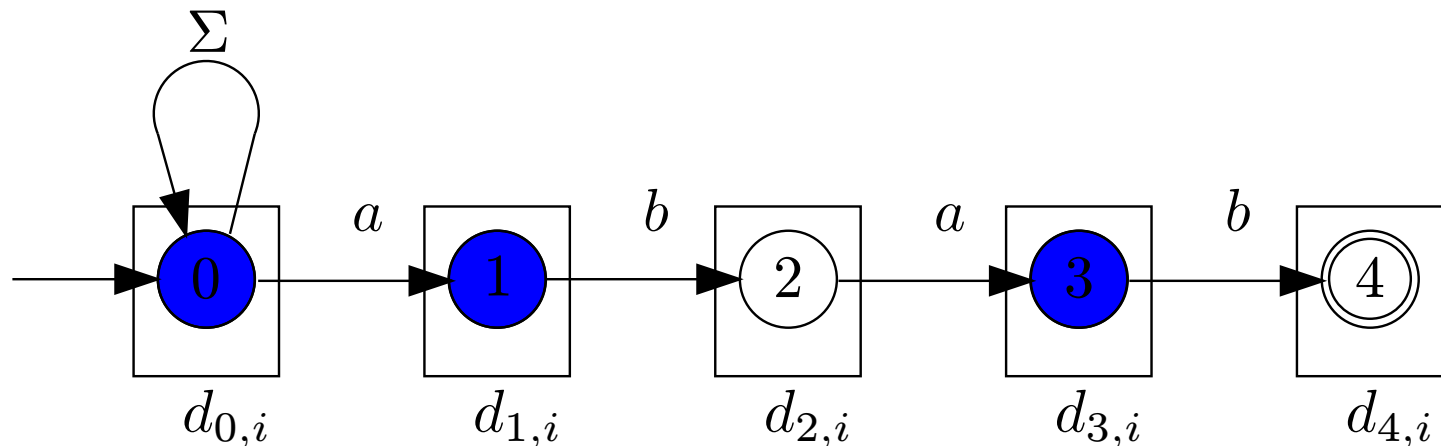
$P = abab$ ,  $T = aababaabab$



	$D$		$a$	$a$	$b$	$a$
	$d_{0,i}$	0	0	0	0	
$a$	$d_{1,i}$	1	0	0	1	
$b$	$d_{2,i}$	1	1	1	0	
$a$	$d_{3,i}$	1	1	1	1	
$b$	$d_{4,i}$	1	1	1	1	

# Dyn. Programming—Exact String M.

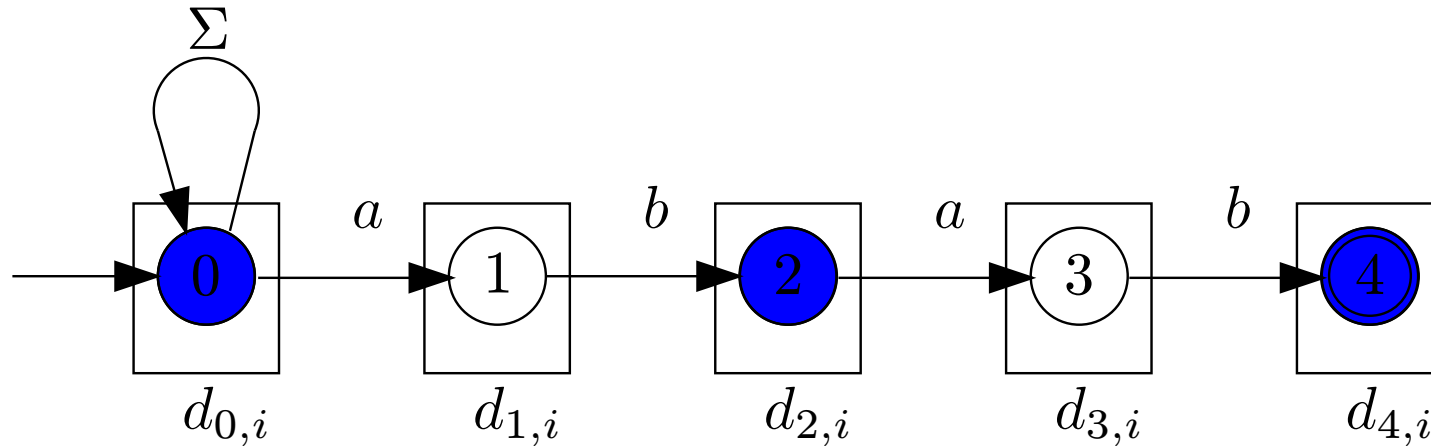
$P = abab$ ,  $T = aababaabab$



	$D$		$a$	$a$	$b$	$a$	$b$
	$d_{0,i}$	0	0	0	0	0	
$a$	$d_{1,i}$	1	0	0	1	0	
$b$	$d_{2,i}$	1	1	1	0	1	
$a$	$d_{3,i}$	1	1	1	1	0	
$b$	$d_{4,i}$	1	1	1	1	1	

# Dyn. Programming—Exact String M.

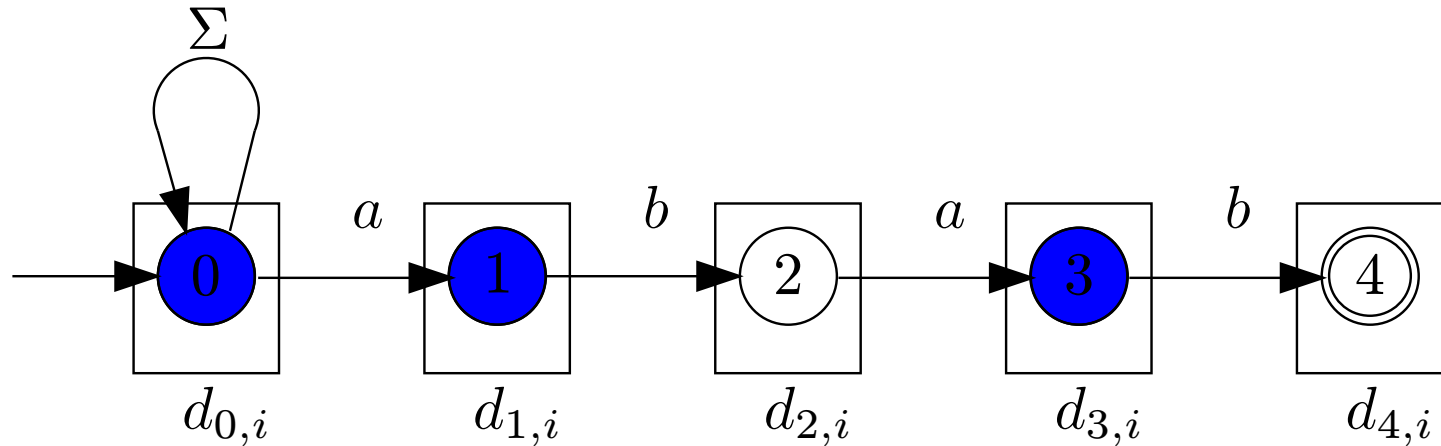
$P = abab$ ,  $T = aababaabab$



	$D$		$a$	$a$	$b$	$a$	$b$	$a$
	$d_{0,i}$	0	0	0	0	0	0	
$a$	$d_{1,i}$	1	0	0	1	0	1	
$b$	$d_{2,i}$	1	1	1	0	1	0	
$a$	$d_{3,i}$	1	1	1	1	0	1	
$b$	$d_{4,i}$	1	1	1	1	1	0	

# Dyn. Programming—Exact String M.

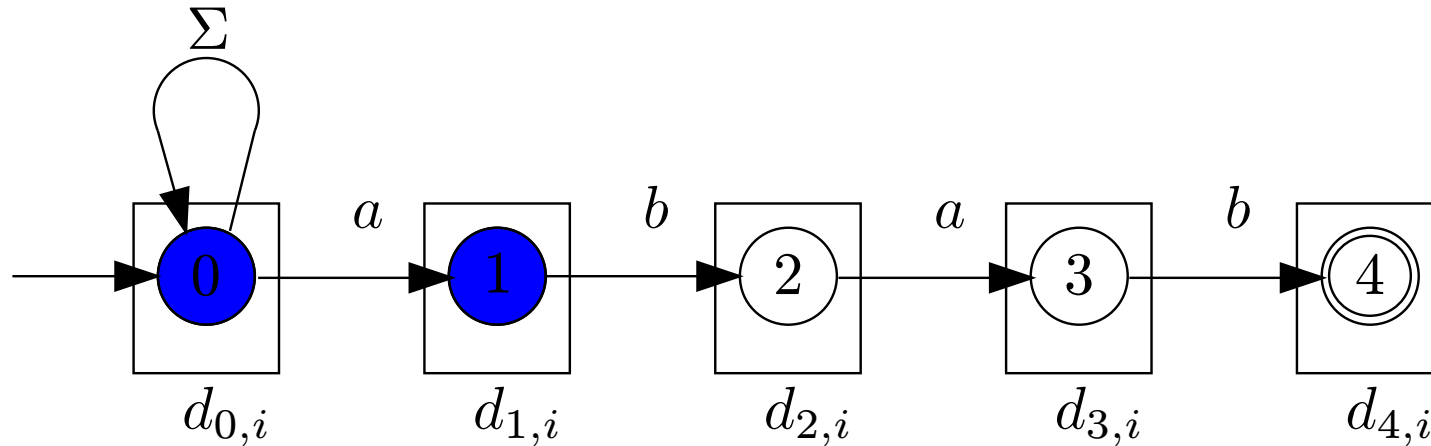
$P = abab$ ,  $T = aababaabab$



	$D$		$a$	$a$	$b$	$a$	$b$	$a$	$a$
	$d_{0,i}$	0	0	0	0	0	0	0	
$a$	$d_{1,i}$	1	0	0	1	0	1	0	
$b$	$d_{2,i}$	1	1	1	0	1	0	1	
$a$	$d_{3,i}$	1	1	1	1	0	1	0	
$b$	$d_{4,i}$	1	1	1	1	1	0	1	

# Dyn. Programming—Exact String M.

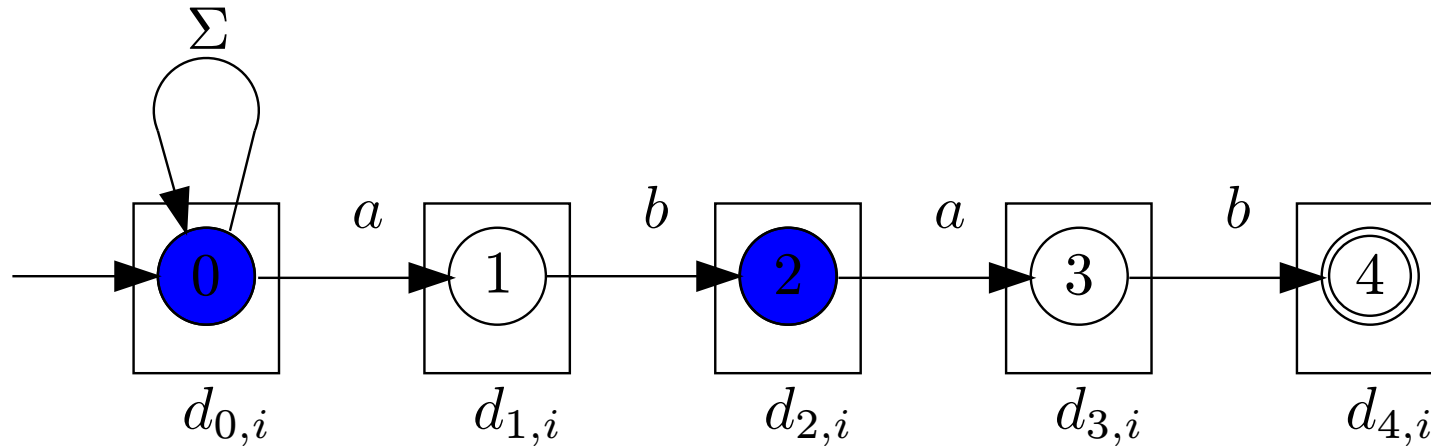
$P = abab$ ,  $T = aababaabab$



	$D$		$a$	$a$	$b$	$a$	$b$	$a$	$a$	$b$
	$d_{0,i}$	0	0	0	0	0	0	0	0	
$a$	$d_{1,i}$	1	0	0	1	0	1	0	0	
$b$	$d_{2,i}$	1	1	1	0	1	0	1	1	
$a$	$d_{3,i}$	1	1	1	1	0	1	0	1	
$b$	$d_{4,i}$	1	1	1	1	1	0	1	1	

# Dyn. Programming—Exact String M.

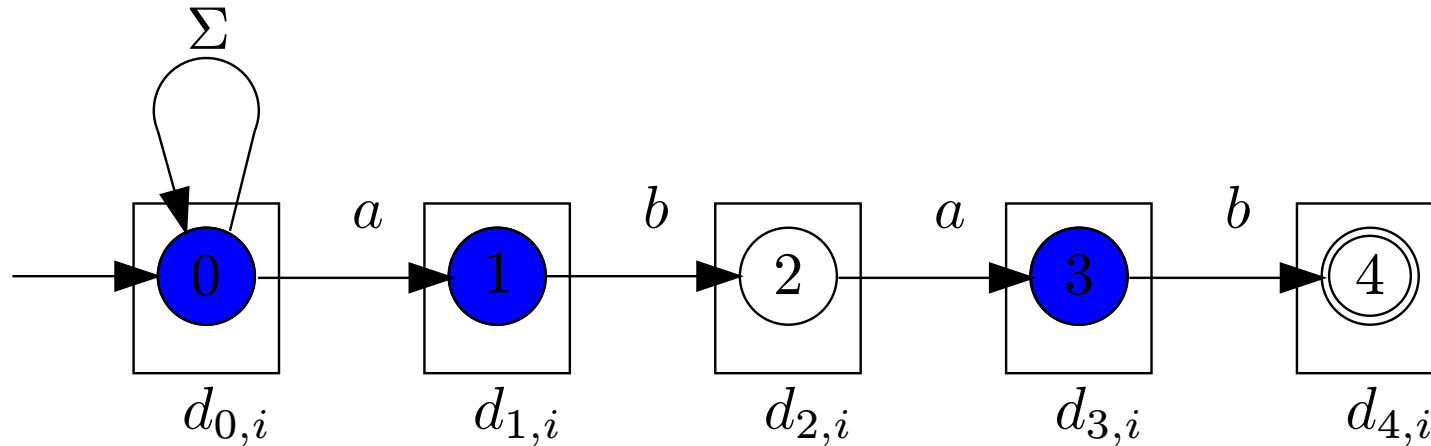
$P = abab$ ,  $T = aababaabab$



	$D$		$a$	$a$	$b$	$a$	$b$	$a$	$a$	$b$	$a$
	$d_{0,i}$	0	0	0	0	0	0	0	0	0	
$a$	$d_{1,i}$	1	0	0	1	0	1	0	0	1	
$b$	$d_{2,i}$	1	1	1	0	1	0	1	1	0	
$a$	$d_{3,i}$	1	1	1	1	0	1	0	1	1	
$b$	$d_{4,i}$	1	1	1	1	1	0	1	1	1	

# Dyn. Programming—Exact String M.

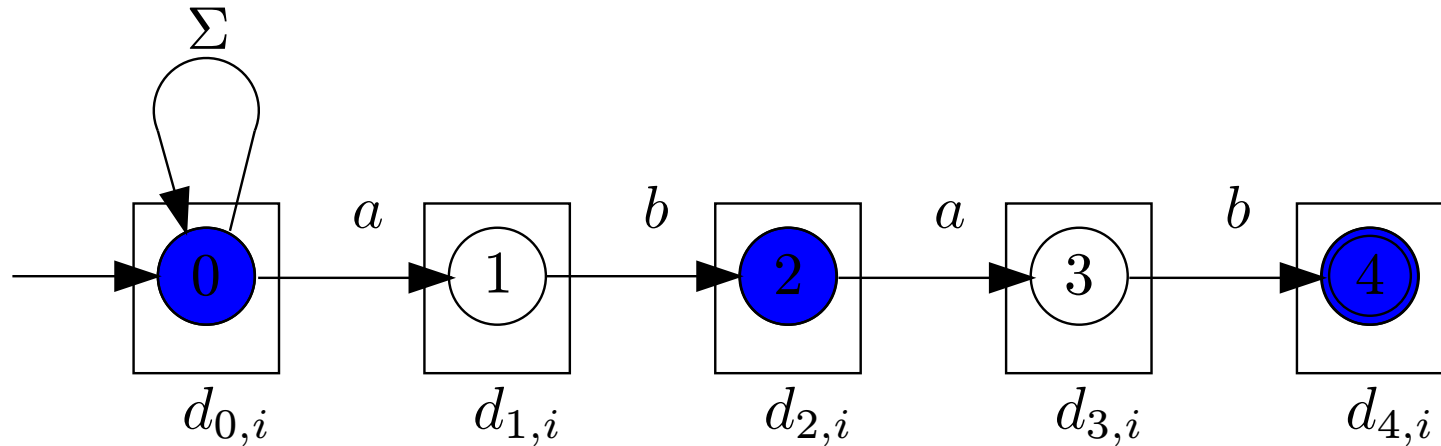
$P = abab$ ,  $T = aababaabab$



	$D$		$a$	$a$	$b$	$a$	$b$	$a$	$a$	$b$	$a$	$b$
	$d_{0,i}$	0	0	0	0	0	0	0	0	0	0	
$a$	$d_{1,i}$	1	0	0	1	0	1	0	0	1	0	
$b$	$d_{2,i}$	1	1	1	0	1	0	1	1	0	1	
$a$	$d_{3,i}$	1	1	1	1	0	1	0	1	1	0	
$b$	$d_{4,i}$	1	1	1	1	1	0	1	1	1	1	

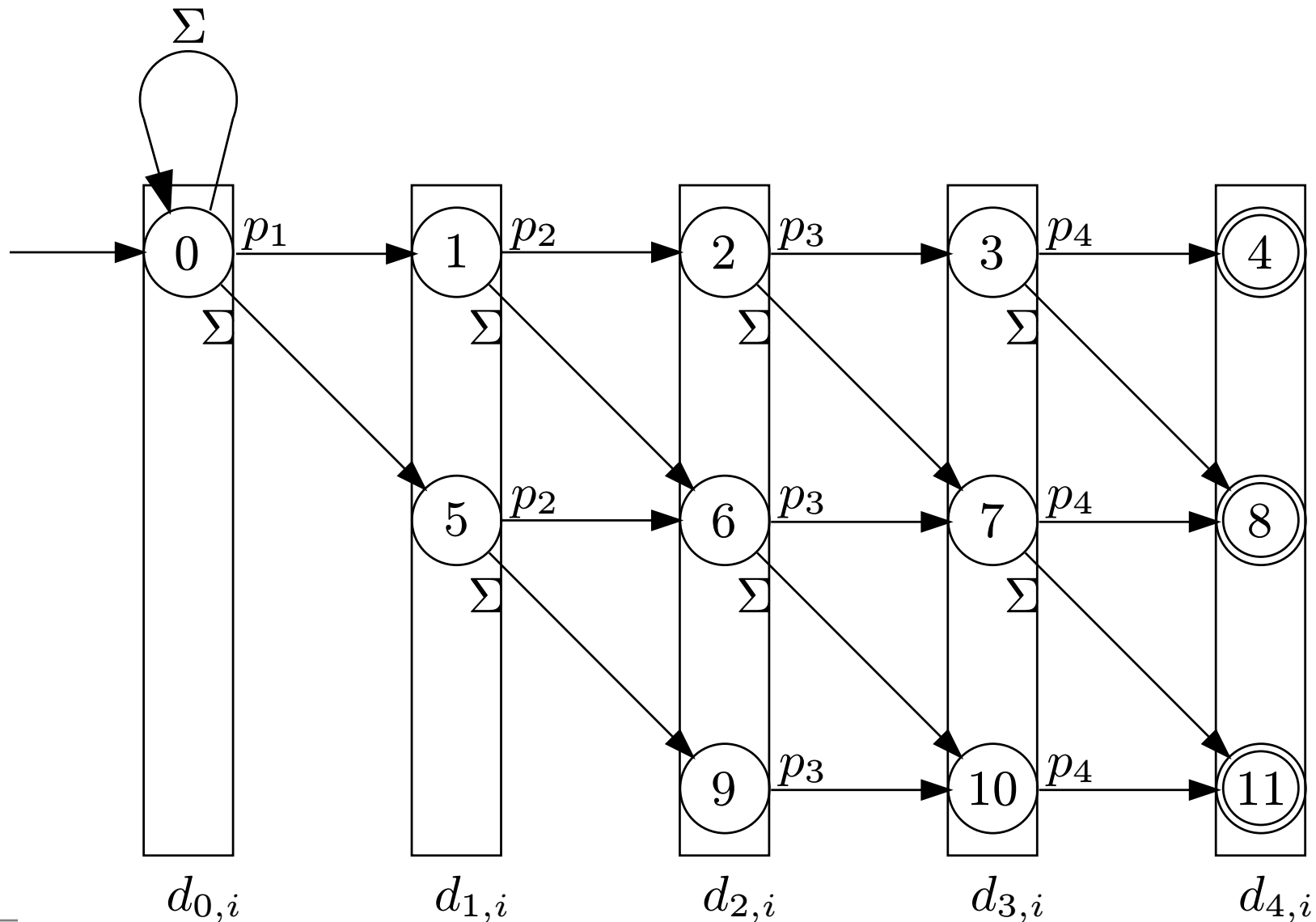
# Dyn. Programming—Exact String M.

$P = abab$ ,  $T = aababaabab$



	$D$		$a$	$a$	$b$	$a$	$b$	$a$	$a$	$b$	$a$	$b$
	$d_{0,i}$	0	0	0	0	0	0	0	0	0	0	0
$a$	$d_{1,i}$	1	0	0	1	0	1	0	0	1	0	1
$b$	$d_{2,i}$	1	1	1	0	1	0	1	1	0	1	0
$a$	$d_{3,i}$	1	1	1	1	0	1	0	1	1	0	1
$b$	$d_{4,i}$	1	1	1	1	1	0	1	1	1	1	0

# DP—Hamming distance



# DP—Hamming distance

$$d_{j,0} \leftarrow k + 1, \quad 0 < j \leq m$$

$$d_{0,i} \leftarrow 0, \quad 0 \leq i \leq n$$

$$d_{j,i} \leftarrow \begin{array}{l} \text{if } t_i = p_j \text{ then } d_{j-1,i-1} \\ \text{else } d_{j-1,i-1} + 1, \end{array} \quad 0 < i \leq n, 0 < j \leq m$$

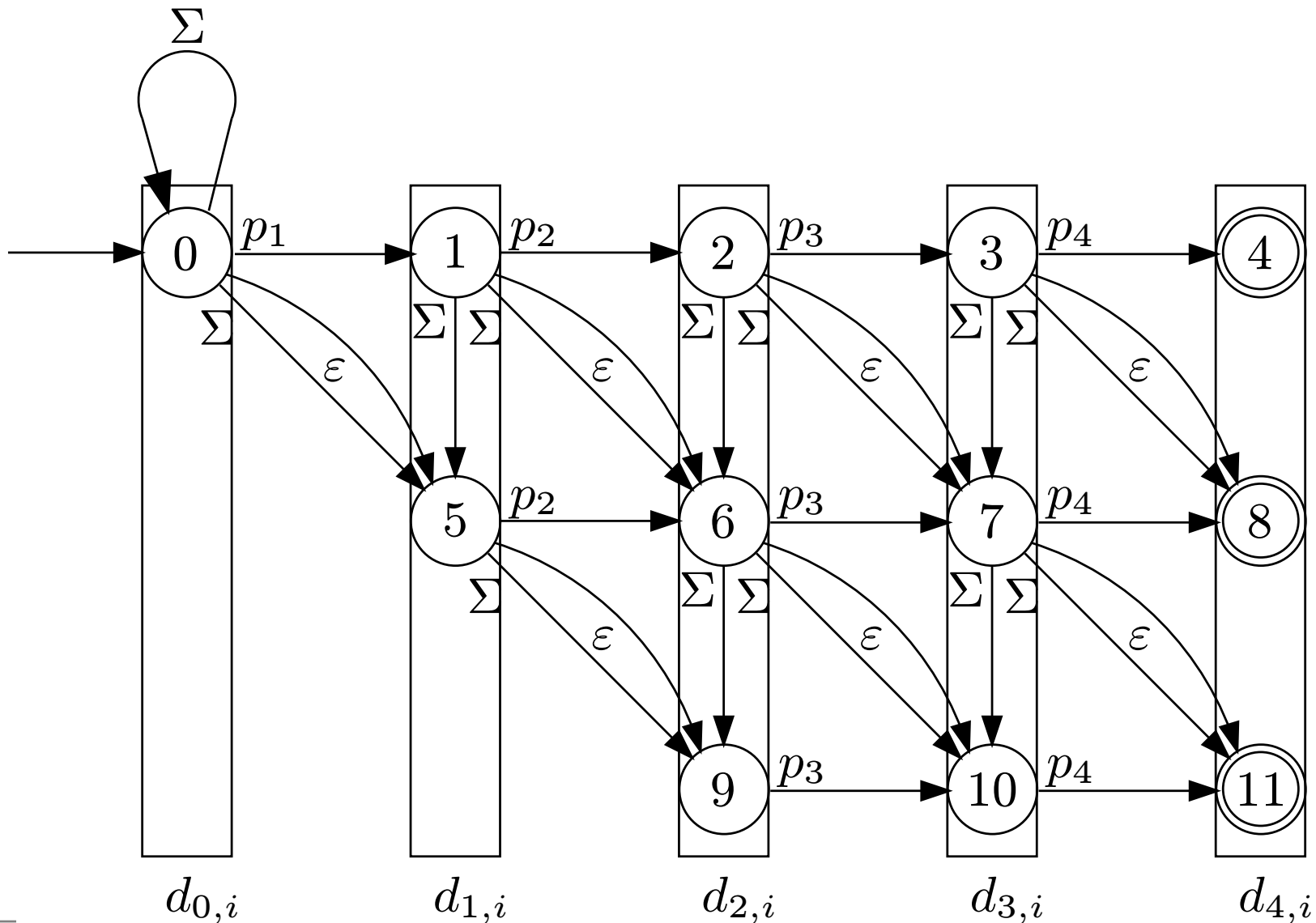
replace

# DP—Hamming distance

<i>D</i>	-	<i>a</i>	<i>d</i>	<i>c</i>	<i>a</i>	<i>b</i>	<i>c</i>	<i>a</i>	<i>a</i>	<i>b</i>	<i>a</i>	<i>d</i>	<i>b</i>	<i>b</i>	<i>c</i>	<i>a</i>
-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
<i>a</i>	4	0	1	1	0	1	1	0	0	1	0	1	1	1	1	0
<i>d</i>	4	5	0	2	2	1	2	2	1	1	2	0	2	2	2	2
<i>b</i>	4	5	6	1	3	2	2	3	3	1	2	3	0	2	3	3
<i>b</i>	4	5	6	7	2	3	3	3	4	3	2	3	3	0	3	4
<i>c</i>	4	5	6	6	8	3	3	4	4	5	4	3	4	4	0	4
<i>a</i>	4	4	6	7	6	9	4	3	4	5	5	5	4	5	5	0

Simulation of *NFA* for approximate string matching using Hamming distance with at most  $k = 3$  errors for pattern  $P = adbbca$  and text  $T = adcabcaabadbbca$ .

# DP—Levenshtein distance



# DP—Levenshtein distance

$$\begin{aligned}d_{j,0} &\leftarrow j, & 0 \leq j \leq m \\d_{0,i} &\leftarrow 0, & 0 \leq i \leq n \\d_{j,i} &\leftarrow \min(\text{if } t_i = p_j \text{ then } d_{j-1,i-1} \\ &\quad \text{else } d_{j-1,i-1} + 1, \\ &\quad \text{if } j < m \text{ then } d_{j,i-1} + 1, \\ &\quad d_{j-1,i} + 1), & 0 < i \leq n, \\ & & 0 < j \leq m\end{aligned}$$

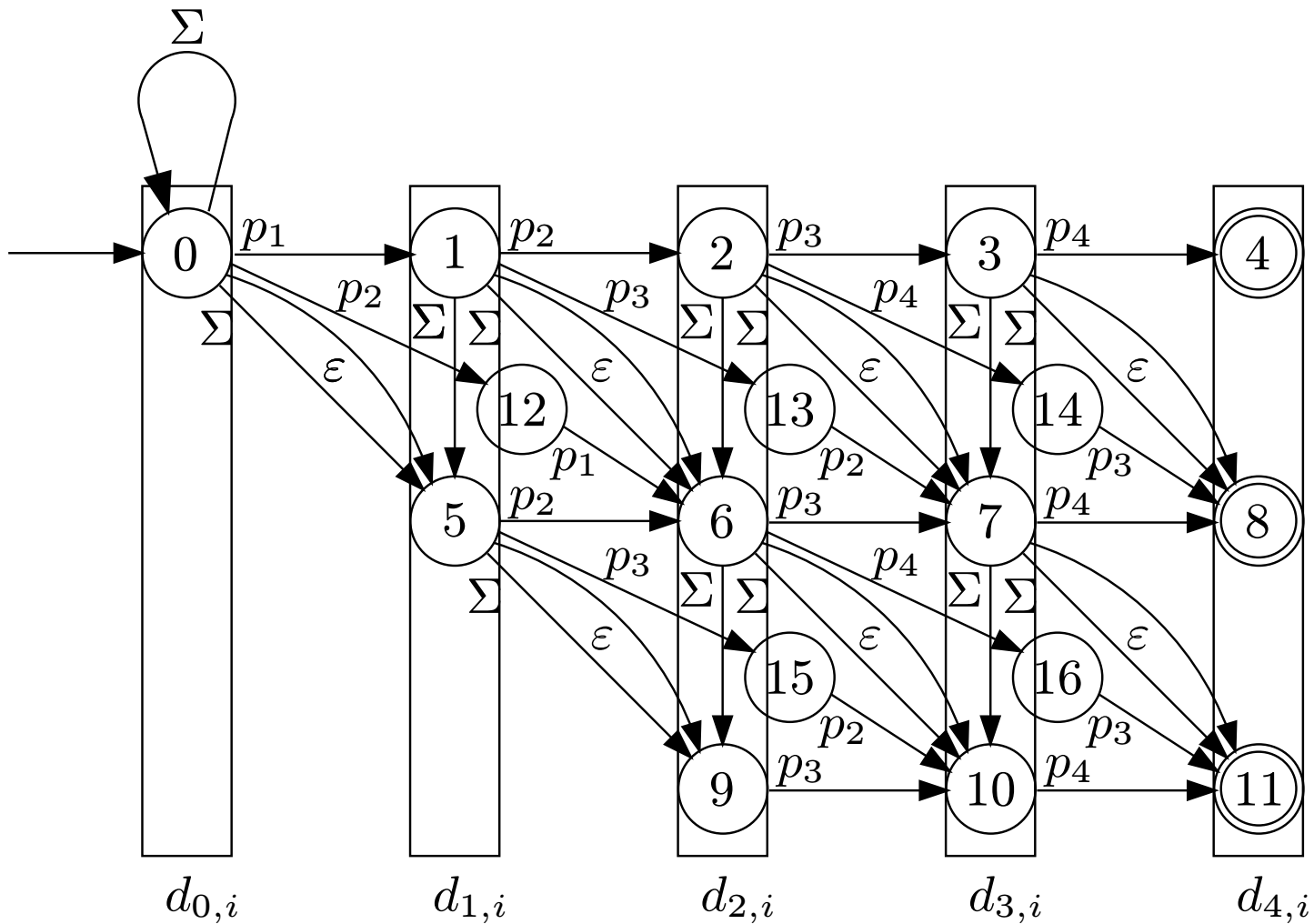
replace  
insert  
delete

# DP—Levenshtein distance

<i>D</i>	-	<i>a</i>	<i>d</i>	<i>c</i>	<i>a</i>	<i>b</i>	<i>c</i>	<i>a</i>	<i>a</i>	<i>b</i>	<i>a</i>	<i>d</i>	<i>b</i>	<i>b</i>	<i>c</i>	<i>a</i>
-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
<i>a</i>	1	0	1	1	0	1	1	0	0	1	0	1	1	1	1	0
<i>d</i>	2	1	0	1	1	1	2	1	1	1	1	0	1	2	2	1
<i>b</i>	3	2	1	1	2	1	2	2	2	1	2	1	0	1	2	2
<i>b</i>	4	3	2	2	2	2	2	3	3	2	2	2	1	0	1	2
<i>c</i>	5	4	3	2	3	3	2	3	4	3	3	3	2	1	0	1
<i>a</i>	6	5	4	3	2	4	3	2	3	4	3	4	3	2	1	0

Simulation of *NFA* for approximate string matching using Levenshtein distance with at most  $k = 3$  errors for pattern  $P = adbbca$  and text  $T = adcabcaabadbcca$ .

# DP—Damerau distance



# DP—Damerau distance

$$\begin{aligned}d_{j,0} &\leftarrow j, & 0 \leq j \leq m \\d_{0,i} &\leftarrow 0, & 0 \leq i \leq n \\d_{j,i} &\leftarrow \min(\text{if } t_i = p_j \text{ then } d_{j-1,i-1} \\ &\quad \text{else } d_{j-1,i-1} + 1, \\ &\quad \text{if } j < m \text{ then } d_{j,i-1} + 1, \\ &\quad d_{j-1,i} + 1, \\ &\quad \text{if } i > 1 \text{ and } j > 1 \\ &\quad \quad \text{and } t_{i-1} = p_j \text{ and } t_i = p_{j-1} \\ &\quad \text{then } d_{j-2,i-2} + 1), & 0 < i \leq n, \\ & & 0 < j \leq m\end{aligned}$$

replace, insert, delete, transpose

# DP—Damerau distance

<i>D</i>	-	<i>a</i>	<i>d</i>	<i>b</i>	<i>c</i>	<i>b</i>	<i>a</i>	<i>a</i>	<i>b</i>	<i>a</i>	<i>d</i>	<i>b</i>	<i>b</i>	<i>c</i>	<i>a</i>
-	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
<i>a</i>	1	0	1	1	1	1	0	0	1	0	1	1	1	1	0
<i>d</i>	2	1	0	1	2	2	1	1	1	1	0	1	2	2	1
<i>b</i>	3	2	1	0	1	2	2	2	1	2	1	0	1	2	2
<i>b</i>	4	3	2	1	1	1	2	3	2	2	2	1	0	1	2
<i>c</i>	5	4	3	2	1	1	2	3	3	3	3	2	1	0	1
<i>a</i>	6	5	4	3	2	2	1	2	4	3	4	3	2	1	0

Simulation of *NFA* for approximate string matching using Damerau distance with at most  $k = 3$  errors for pattern  $P = adbbca$  and text  $T = adcabcaabadbcca$ .

# Dynamic Programming

Complexities:

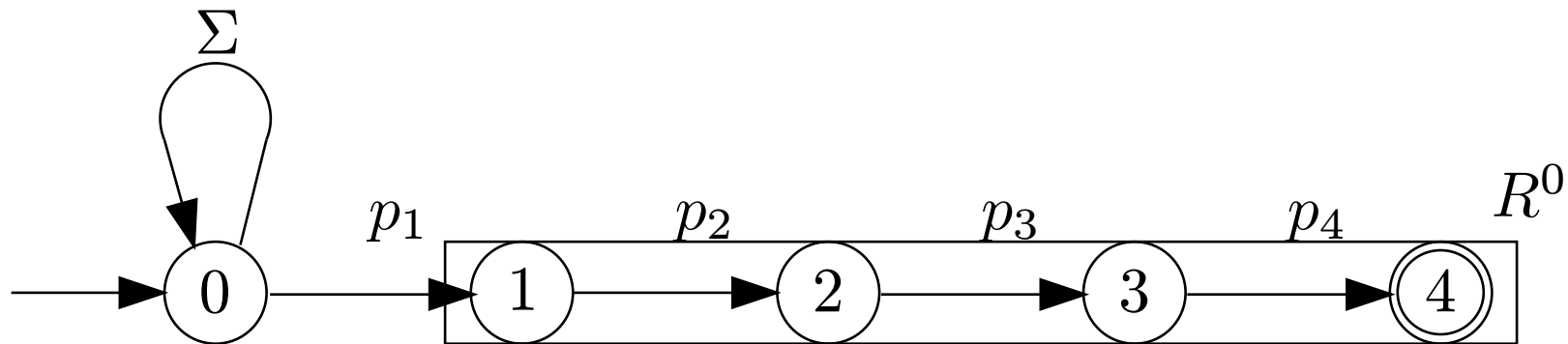
	time	space
Dynamic Programming	$\mathcal{O}(mn)$	$\mathcal{O}(m)$
[Galil & Park 1989]	$\mathcal{O}(kn)$	$\mathcal{O}(k)$

# Dynamic Programming

## References

- [1] R. A. Wagner and M. Fischer. The string-to-string correction problem. *J. Assoc. Comput. Mach.*, 21(1):168–173, 1974.
- [2] P. H. Sellers. The theory and computation of evolutionary distances: Pattern recognition. *J. Algorithms*, 1(4):359–373, 1980.
- [3] E. Ukkonen. Finding approximate patterns in strings. *J. Algorithms*, 6(1–3):132–137, 1985.

# Bit Parallelism—Simulation



$R^0$  is a bit vector (for 0 errors = exact string matching).

# Bit Parallelism—Exact String Matching

$$R_i^l = \begin{bmatrix} r_{1,i}^l \\ r_{2,i}^l \\ \vdots \\ r_{m,i}^l \end{bmatrix}, \quad \begin{array}{l} 0 \leq i \leq n, \\ 0 \leq l \leq k, \end{array} \quad \text{and} \quad D[x] = \begin{bmatrix} d_{1,x} \\ d_{2,x} \\ \vdots \\ d_{m,x} \end{bmatrix}, \quad x \in \Sigma.$$

$$\begin{array}{ll} r_{j,0}^0 & \leftarrow 1, & 0 < j \leq m \\ R_i^0 & \leftarrow \text{shr}(R_{i-1}^0) \text{ OR } D[t_i], & 0 < i \leq n \end{array}$$

# Bit Parallelism—Exact String Matching

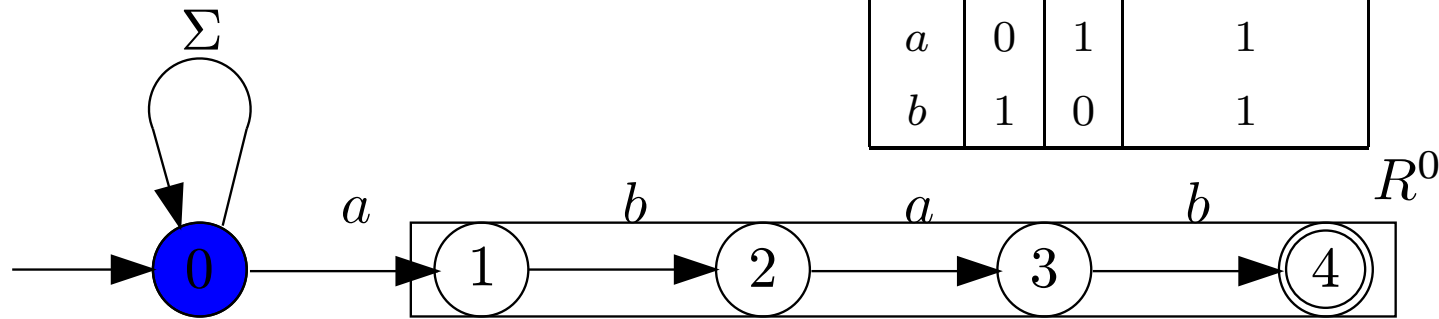
$D$	$a$	$b$	$\Sigma \setminus \{a, b\}$
$a$	0	1	1
$b$	1	0	1
$a$	0	1	1
$b$	1	0	1

Table 1: Matrix  $D$  for pattern  $P = abab$ .

# Bit Parallelism—Exact String Matching

$P = abab, T = aababaabab$

$D$	$a$	$b$	$\Sigma \setminus \{a, b\}$
$a$	0	1	1
$b$	1	0	1
$a$	0	1	1
$b$	1	0	1

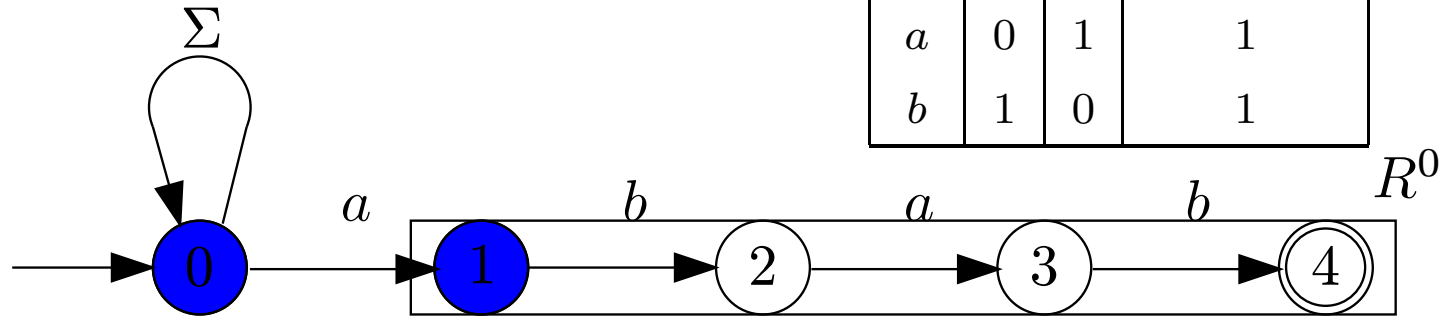


$R^0$		$a$
$a$	1	
$b$	1	
$a$	1	
$b$	1	

# Bit Parallelism—Exact String Matching

$P = abab, T = aababaabab$

$D$	$a$	$b$	$\Sigma \setminus \{a, b\}$
$a$	0	1	1
$b$	1	0	1
$a$	0	1	1
$b$	1	0	1

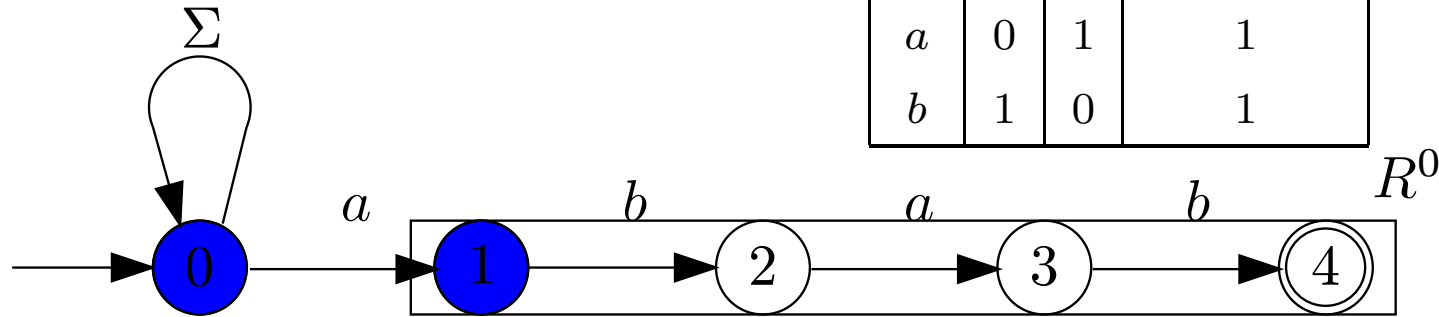


$R^0$		$a$	$a$
$a$	1	0	
$b$	1	1	
$a$	1	1	
$b$	1	1	

# Bit Parallelism—Exact String Matching

$P = abab, T = aababaabab$

$D$	$a$	$b$	$\Sigma \setminus \{a, b\}$
$a$	0	1	1
$b$	1	0	1
$a$	0	1	1
$b$	1	0	1

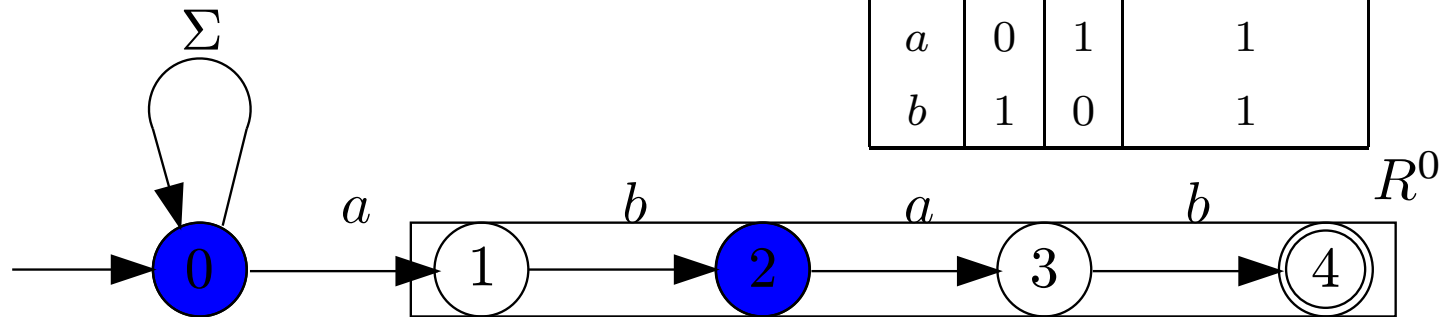


$R^0$		$a$	$a$	$b$
$a$	1	0	0	
$b$	1	1	1	
$a$	1	1	1	
$b$	1	1	1	

# Bit Parallelism—Exact String Matching

$P = abab, T = aababaabab$

$D$	$a$	$b$	$\Sigma \setminus \{a, b\}$
$a$	0	1	1
$b$	1	0	1
$a$	0	1	1
$b$	1	0	1

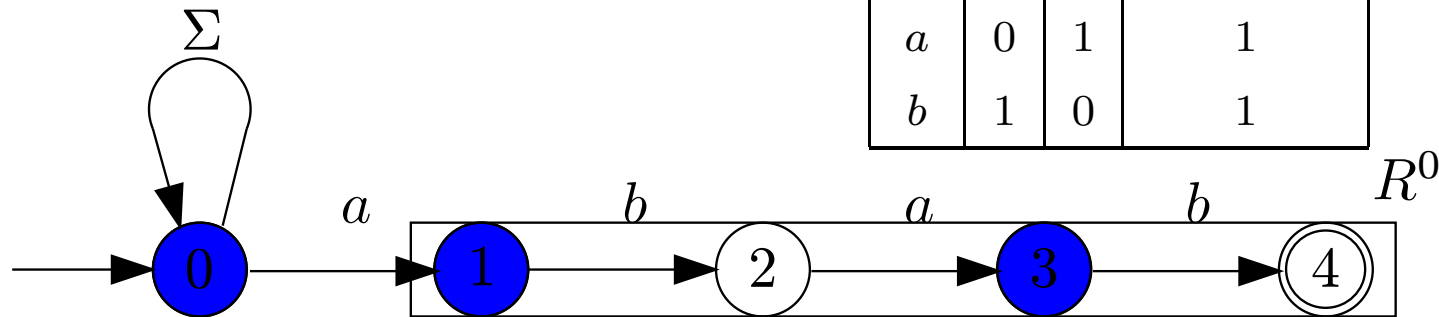


$R^0$		$a$	$a$	$b$	$a$
$a$	1	0	0	1	
$b$	1	1	1	0	
$a$	1	1	1	1	
$b$	1	1	1	1	

# Bit Parallelism—Exact String Matching

$P = abab, T = aababaabab$

$D$	$a$	$b$	$\Sigma \setminus \{a, b\}$
$a$	0	1	1
$b$	1	0	1
$a$	0	1	1
$b$	1	0	1

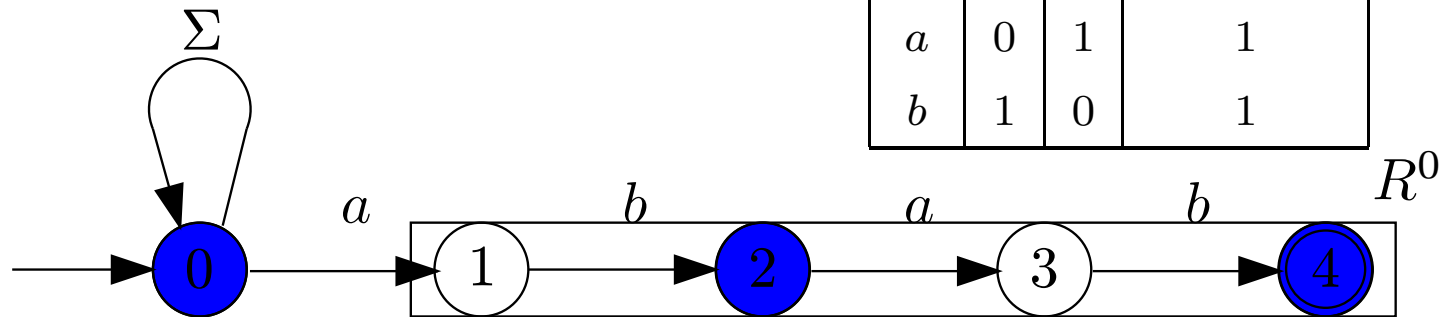


$R^0$		$a$	$a$	$b$	$a$	$b$
$a$	1	0	0	1	0	
$b$	1	1	1	0	1	
$a$	1	1	1	1	0	
$b$	1	1	1	1	1	

# Bit Parallelism—Exact String Matching

$P = abab, T = aababaabab$

$D$	$a$	$b$	$\Sigma \setminus \{a, b\}$
$a$	0	1	1
$b$	1	0	1
$a$	0	1	1
$b$	1	0	1

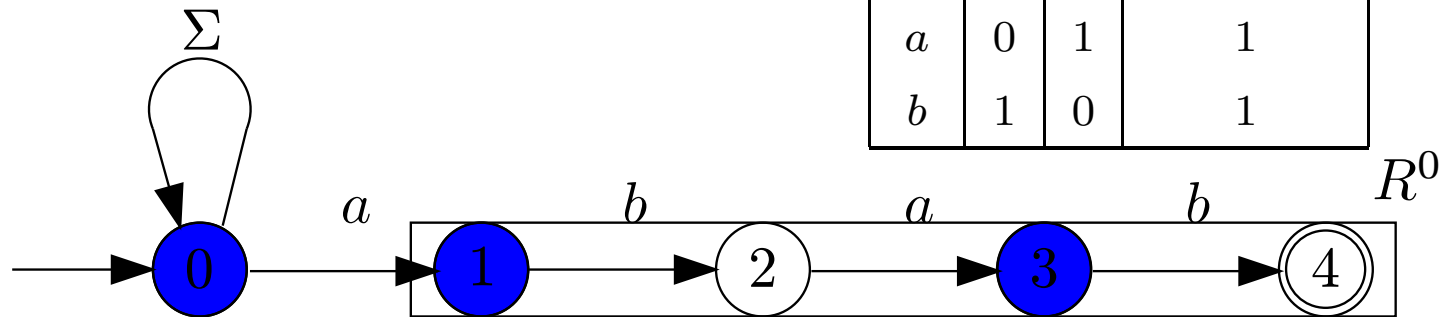


$R^0$		$a$	$a$	$b$	$a$	$b$	$a$
$a$	1	0	0	1	0	1	
$b$	1	1	1	0	1	0	
$a$	1	1	1	1	0	1	
$b$	1	1	1	1	1	0	

# Bit Parallelism—Exact String Matching

$P = abab, T = aababaabab$

$D$	$a$	$b$	$\Sigma \setminus \{a, b\}$
$a$	0	1	1
$b$	1	0	1
$a$	0	1	1
$b$	1	0	1

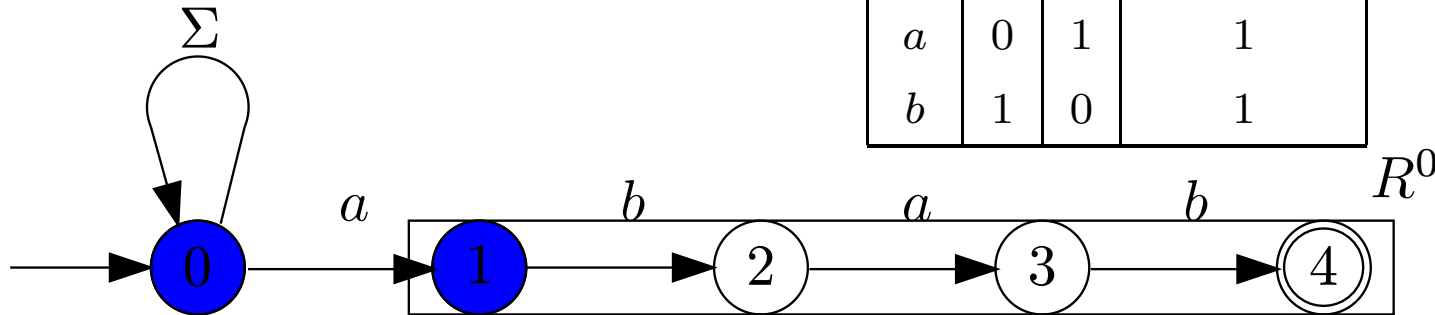


$R^0$		$a$	$a$	$b$	$a$	$b$	$a$	$a$
$a$	1	0	0	1	0	1	0	
$b$	1	1	1	0	1	0	1	
$a$	1	1	1	1	0	1	0	
$b$	1	1	1	1	1	0	1	

# Bit Parallelism—Exact String Matching

$P = abab, T = aababaabab$

$D$	$a$	$b$	$\Sigma \setminus \{a, b\}$
$a$	0	1	1
$b$	1	0	1
$a$	0	1	1
$b$	1	0	1

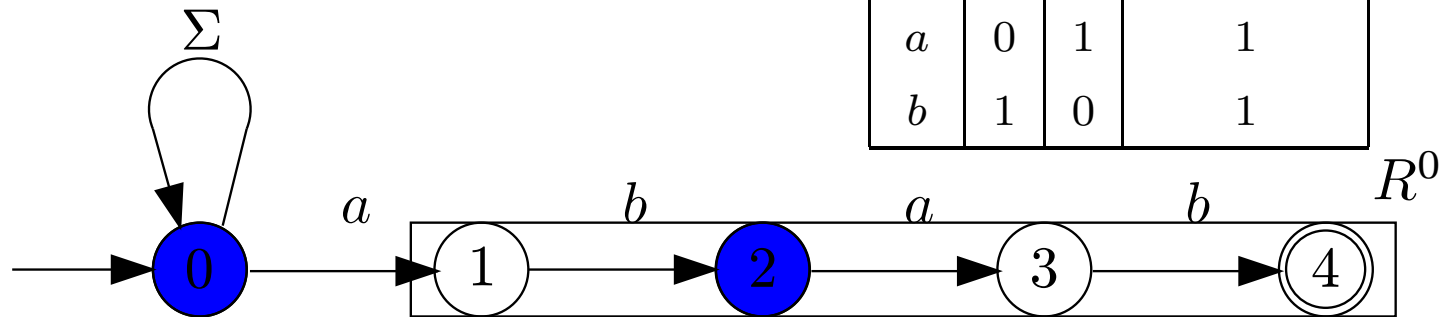


$R^0$		$a$	$a$	$b$	$a$	$b$	$a$	$a$	$b$
$a$	1	0	0	1	0	1	0	0	
$b$	1	1	1	0	1	0	1	1	
$a$	1	1	1	1	0	1	0	1	
$b$	1	1	1	1	1	0	1	1	

# Bit Parallelism—Exact String Matching

$P = abab, T = aababaabab$

$D$	$a$	$b$	$\Sigma \setminus \{a, b\}$
$a$	0	1	1
$b$	1	0	1
$a$	0	1	1
$b$	1	0	1

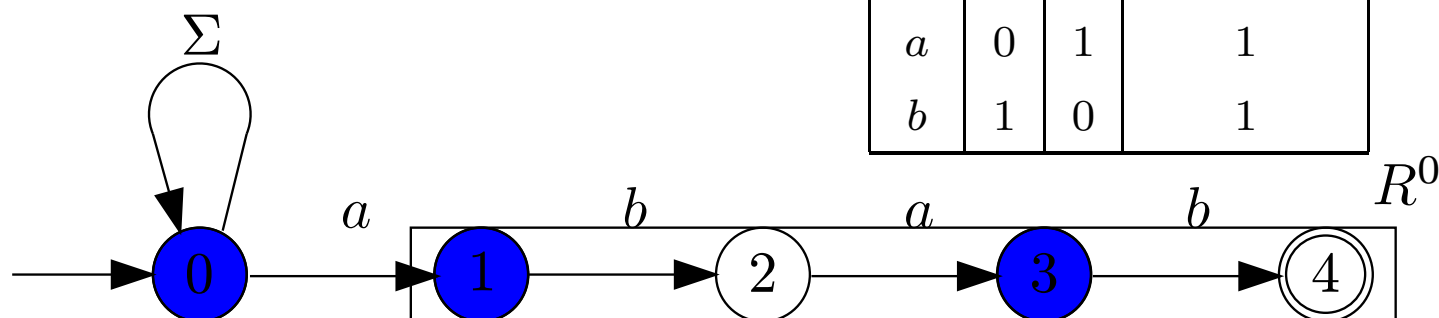


$R^0$		$a$	$a$	$b$	$a$	$b$	$a$	$a$	$b$	$a$
$a$	1	0	0	1	0	1	0	0	1	
$b$	1	1	1	0	1	0	1	1	0	
$a$	1	1	1	1	0	1	0	1	1	
$b$	1	1	1	1	1	0	1	1	1	

# Bit Parallelism—Exact String Matching

$P = abab, T = aababaabab$

$D$	$a$	$b$	$\Sigma \setminus \{a, b\}$
$a$	0	1	1
$b$	1	0	1
$a$	0	1	1
$b$	1	0	1

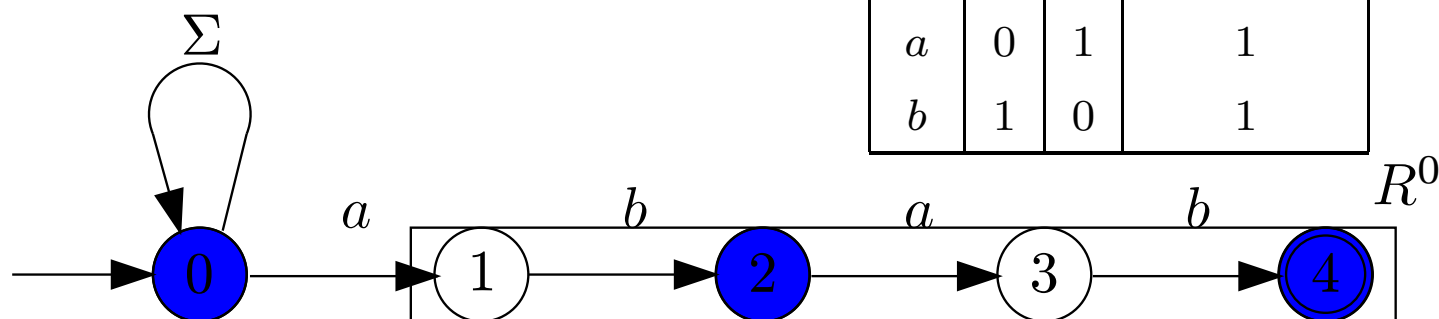


$R^0$		$a$	$a$	$b$	$a$	$b$	$a$	$a$	$b$	$a$	$b$
$a$	1	0	0	1	0	1	0	0	1	0	
$b$	1	1	1	0	1	0	1	1	0	1	
$a$	1	1	1	1	0	1	0	1	1	0	
$b$	1	1	1	1	1	0	1	1	1	1	

# Bit Parallelism—Exact String Matching

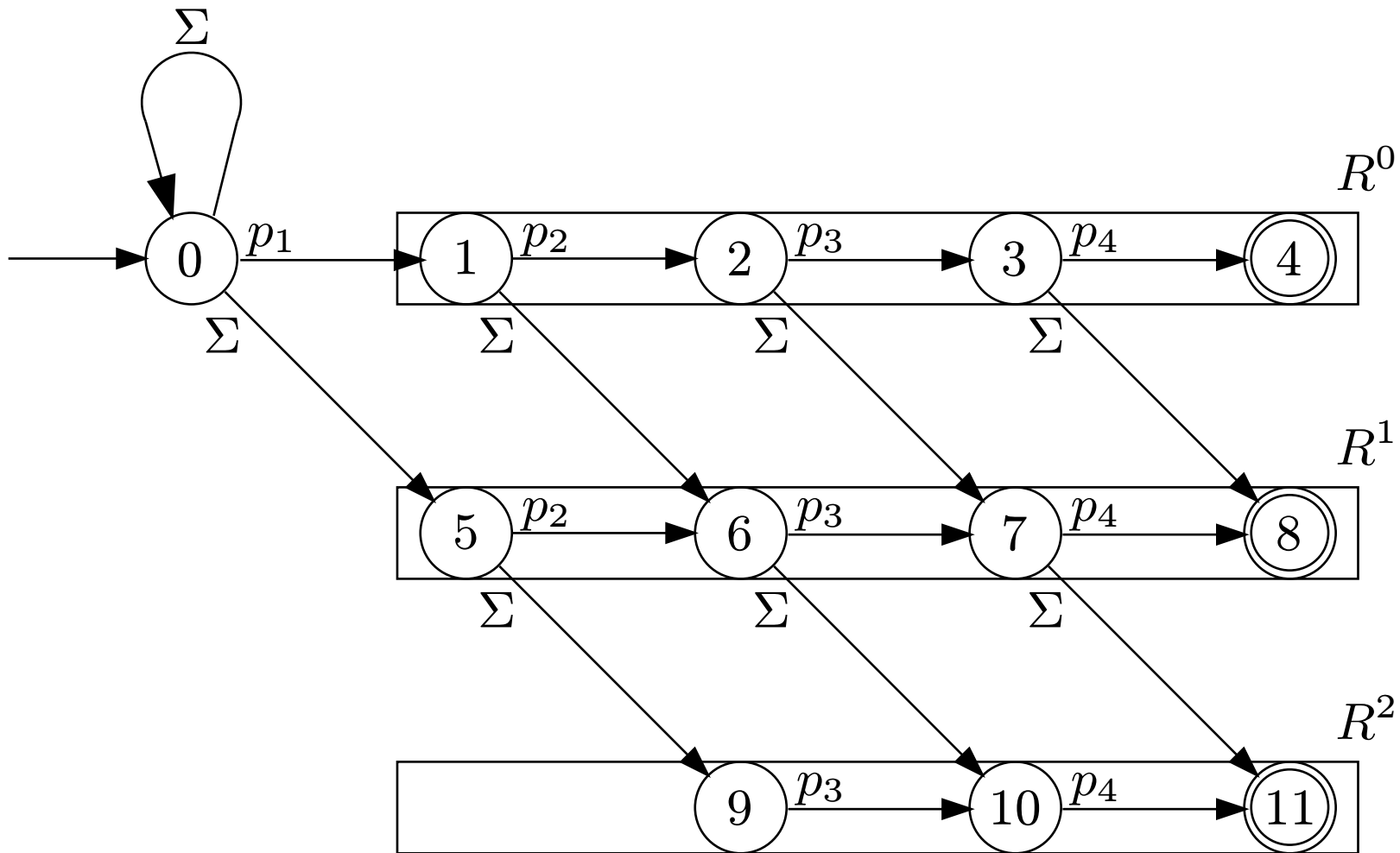
$P = abab, T = aababaabab$

$D$	$a$	$b$	$\Sigma \setminus \{a, b\}$
$a$	0	1	1
$b$	1	0	1
$a$	0	1	1
$b$	1	0	1



$R^0$		$a$	$a$	$b$	$a$	$b$	$a$	$a$	$b$	$a$	$b$
$a$	1	0	0	1	0	1	0	0	1	0	1
$b$	1	1	1	0	1	0	1	1	0	1	0
$a$	1	1	1	1	0	1	0	1	1	0	1
$b$	1	1	1	1	1	0	1	1	1	1	0

# BP—Hamming distance



# BP—Hamming distance

$$\begin{aligned}r_{j,0}^l &\leftarrow 1, & 0 < j \leq m, 0 \leq l \leq k \\R_i^0 &\leftarrow \text{shr}(R_{i-1}^0) \text{ OR } D[t_i], & 0 < i \leq n \\R_i^l &\leftarrow (\text{shr}(R_{i-1}^l) \text{ OR } D[t_i]) \\&\quad \text{AND } \text{shr}(R_{i-1}^{l-1}), & 0 < i \leq n, 0 < l \leq k\end{aligned}$$

replace

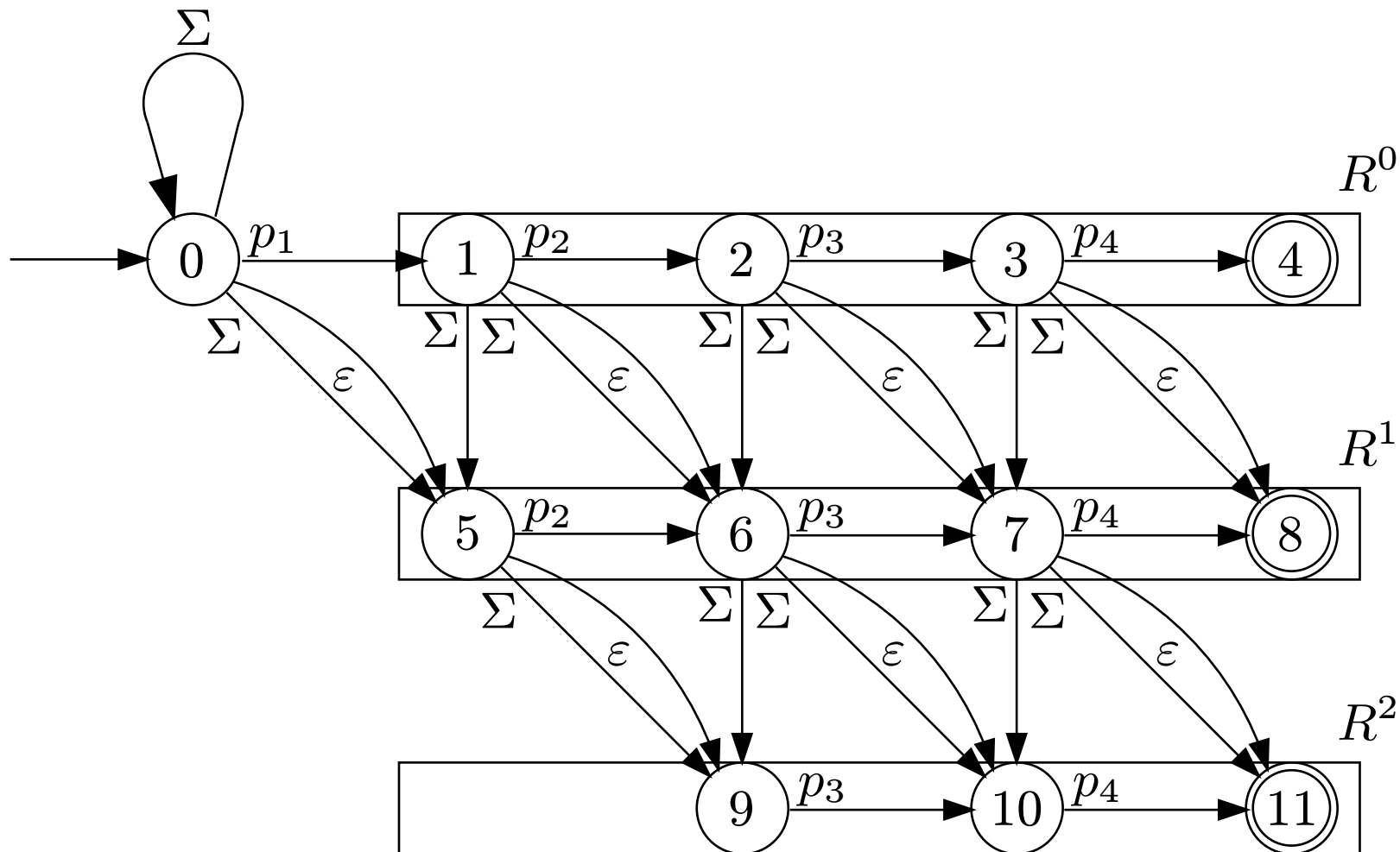
# BP—Hamming distance

$R^0$	-	$a$	$d$	$c$	$a$	$b$	$c$	$a$	$a$	$b$	$a$	$d$	$b$	$b$	$c$	$a$
$a$	1	<b>0</b>	1	1	<b>0</b>	1	1	<b>0</b>	<b>0</b>	1	<b>0</b>	1	1	1	1	<b>0</b>
$d$	1	1	<b>0</b>	1	1	1	1	1	1	1	1	<b>0</b>	1	1	1	1
$b$	1	1	1	1	1	1	1	1	1	1	1	1	<b>0</b>	1	1	1
$b$	1	1	1	1	1	1	1	1	1	1	1	1	1	<b>0</b>	1	1
$c$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	<b>0</b>	1
$a$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	<b>0</b>

$R^1$	-	$a$	$d$	$c$	$a$	$b$	$c$	$a$	$a$	$b$	$a$	$d$	$b$	$b$	$c$	$a$
$a$	1	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>	<b>0</b>
$d$	1	1	<b>0</b>	1	1	<b>0</b>	1	1	<b>0</b>	<b>0</b>	1	<b>0</b>	1	1	1	1
$b$	1	1	1	<b>0</b>	1	1	1	1	1	<b>0</b>	1	1	<b>0</b>	1	1	1
$b$	1	1	1	1	1	1	1	1	1	1	1	1	1	<b>0</b>	1	1
$c$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	<b>0</b>	1
$a$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	<b>0</b>

# BP—Levenshtein distance



# BP—Levenshtein distance

$$\begin{aligned}r_{j,0}^l &\leftarrow 0, & 0 < j \leq l, 0 < l \leq k \\r_{j,0}^l &\leftarrow 1, & l < j \leq m, 0 \leq l \leq k \\R_i^0 &\leftarrow \text{shr}(R_{i-1}^0) \text{ OR } D[t_i], & 0 < i \leq n \\R_i^l &\leftarrow (\text{shr}(R_{i-1}^l) \text{ OR } D[t_i]) \\&\quad \text{AND shr}(R_{i-1}^{l-1} \text{ AND } R_i^{l-1}) \\&\quad \text{AND } (R_{i-1}^{l-1} \text{ OR } V), & 0 < i \leq n, 0 < l \leq k\end{aligned}$$

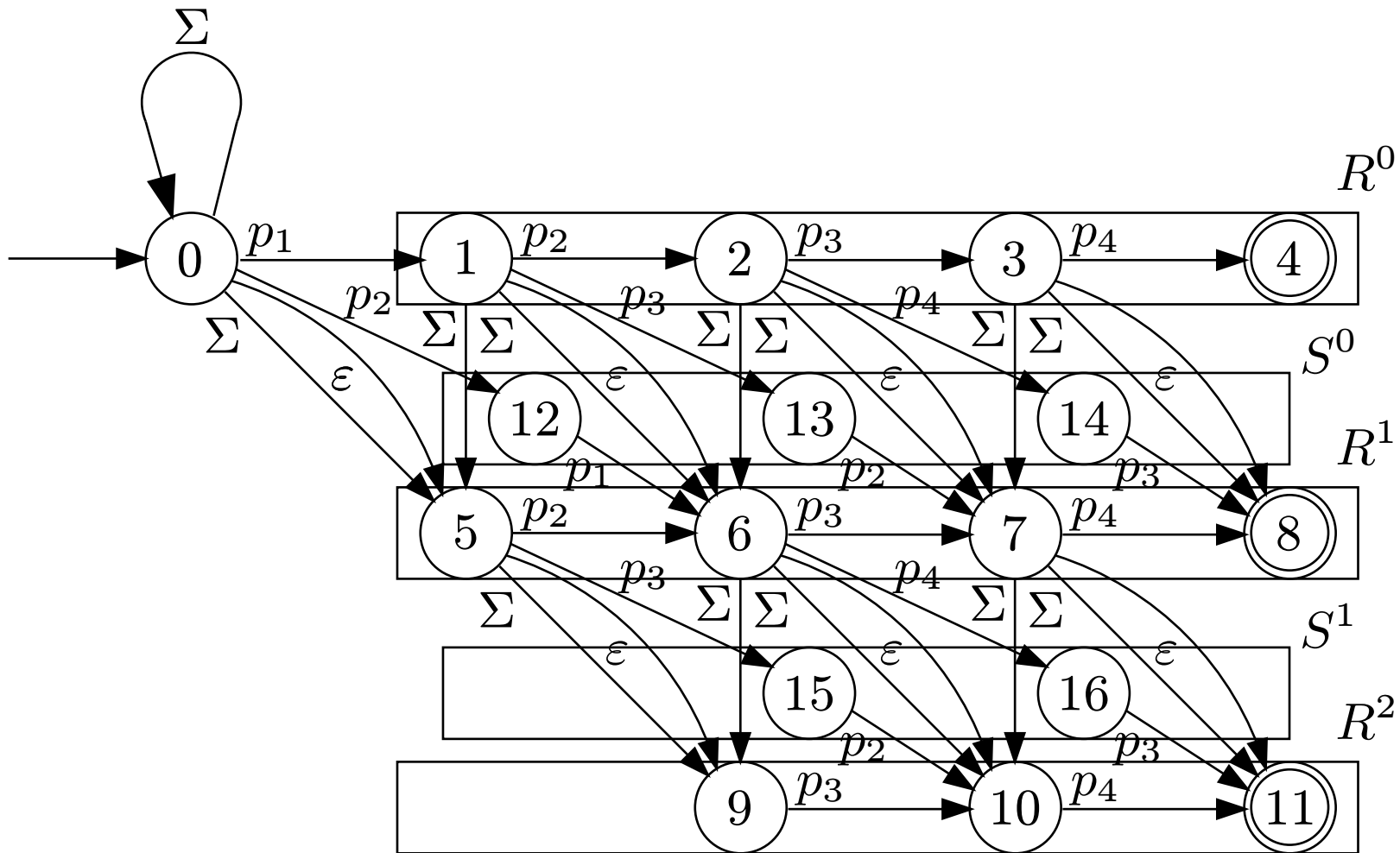
replace, insert, delete

$$V = \begin{bmatrix} v_1 \\ v_2 \\ \vdots \\ v_m \end{bmatrix}, \text{ where } v_m = 1 \text{ and } v_j = 0, \forall j, 1 \leq j < m.$$

# BP—Levenshtein distance

$R^0$	-	$a$	$d$	$c$	$a$	$b$	$c$	$a$	$a$	$b$	$a$	$d$	$b$	$b$	$c$	$a$
$a$	1	0	1	1	0	1	1	0	0	1	0	1	1	1	1	0
$d$	1	1	0	1	1	1	1	1	1	1	1	0	1	1	1	1
$b$	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
$b$	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1
$c$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
$a$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
$R^1$	-	$a$	$d$	$c$	$a$	$b$	$c$	$a$	$a$	$b$	$a$	$d$	$b$	$b$	$c$	$a$
$a$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
$d$	1	0	0	0	0	0	1	0	0	0	0	0	0	1	1	0
$b$	1	1	0	0	1	0	1	1	1	0	1	0	0	0	1	1
$b$	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0	1
$c$	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0	0
$a$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	0

# BP—Damerau distance



# BP—Damerau distance

$$S_i^l = \begin{bmatrix} s_{1,i}^l \\ s_{2,i}^l \\ \vdots \\ s_{m,i}^l \end{bmatrix}, 0 \leq l < k, 0 \leq i < n.$$

# BP—Damerau distance

$$\begin{aligned}r_{j,0}^l &\leftarrow 0, & 0 < j \leq l, 0 < l \leq k \\r_{j,0}^l &\leftarrow 1, & l < j \leq m, 0 \leq l \leq k \\R_i^0 &\leftarrow \text{shr}(R_{i-1}^0) \text{ OR } D[t_i], & 0 < i \leq n \\R_i^l &\leftarrow (\text{shr}(R_{i-1}^l) \text{ OR } D[t_i]) \\&\quad \text{AND shr}(R_{i-1}^{l-1} \text{ AND } R_i^{l-1}) \\&\quad \quad \text{AND } (S_{i-1}^{l-1} \text{ OR } D[t_i]) \\&\quad \text{AND } (R_{i-1}^{l-1} \text{ OR } V), & 0 < i \leq n, 0 < l \leq k \\s_{j,0}^l &\leftarrow 1, & 0 < j \leq m, 0 \leq l < k \\S_i^l &\leftarrow \text{shr}(R_{i-1}^l) \text{ OR shl}(D[t_i]), & 0 < i < n, 0 \leq l < k\end{aligned}$$

replace, insert, delete, transpose

# BP—Damerau distance

$R^0$	-	-	$a$	$d$	$b$	$c$	$b$	$a$	$a$	$b$	$a$	$d$	$b$	$b$	$c$	$a$	
	$a$	1	0	1	1	1	1	0	0	1	0	1	1	1	1	1	0
	$d$	1	1	0	1	1	1	1	1	1	1	0	1	1	1	1	1
	$b$	1	1	1	0	1	1	1	1	1	1	1	0	1	1	1	1
	$b$	1	1	1	1	1	1	1	1	1	1	1	1	0	1	1	1
	$c$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0	1
	$a$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	0
$S^0$	$a$	1	1	0	1	1	1	1	1	1	1	0	1	1	1	1	1
	$d$	1	1	1	1	1	1	1	1	0	1	1	1	1	1	1	1
	$b$	1	1	1	0	1	1	1	1	1	1	1	0	1	1	1	1
	$b$	1	1	1	1	0	1	1	1	1	1	1	1	1	1	1	1
	$c$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	$a$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	$a$	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
$R^1$	$a$	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
	$d$	1	0	0	0	1	1	0	0	0	0	0	0	1	1	0	
	$b$	1	1	0	0	0	1	1	1	0	1	0	0	0	1	1	
	$b$	1	1	1	0	0	0	1	1	1	1	1	0	0	0	1	
	$c$	1	1	1	1	0	0	1	1	1	1	1	1	0	0	0	
	$a$	1	1	1	1	1	1	0	1	1	1	1	1	1	0	0	
	$a$	1	1	1	1	1	1	0	1	1	1	1	1	1	0	0	

# Bit Parallelism

Complexities:

	time	space
Bit Parallelism	$\mathcal{O}(k \lceil \frac{m}{w} \rceil n)$	$\mathcal{O}(k \lceil \frac{m}{w} \rceil)$

where  $w$  is bit-length of computer word

# Bit Parallelism

## References

- [1] B. Dömölki. An algorithm for syntactical analysis. *Computational Linguistics*, (3):29–46, 1964.
- [2] R. K. Shyamasundar. A simple string matching algorithm. Technical report, Tata Institute of Fundamental Research, 1976.
- [3] K. Abrahamson. Generalized string matching. *SIAM J. Comput.*, 16(6):1039–1051, 1987.
- [4] R. A. Baeza-Yates and G. H. Gonnet. A new approach to text searching. *Commun. ACM*, 35(10):74–82, 1992.
- [5] S. Wu and U. Manber. Fast text searching allowing errors. *Commun. ACM*, 35(10):83–91, 1992.

# Work in Progress in PSC

- Simulation of NFA
- Implementation of Indexing Automata
  - suffix (factor) automaton,
  - compact suffix (factor) automaton
- Tessellation Automata
  - pattern matching in 2D text
- 2D Pattern Matching
  - pattern matching in 2D text using 1D searching automata

# Work in Progress in PSC (cont.)

- Regularities in Text
  - border,
  - cover,
  - seed,
  - evolutive motif,
  - tandem repeat
- Pattern Matching in Indeterminate Strings
  - string matching
- XML Format for Automata
- Data Compression
- Pattern Matching in Compressed Text