Hledání řetězců

Karel Richta a kol.

Katedra počítačů
Fakulta elektrotechnická
České vysoké učení technické v Praze

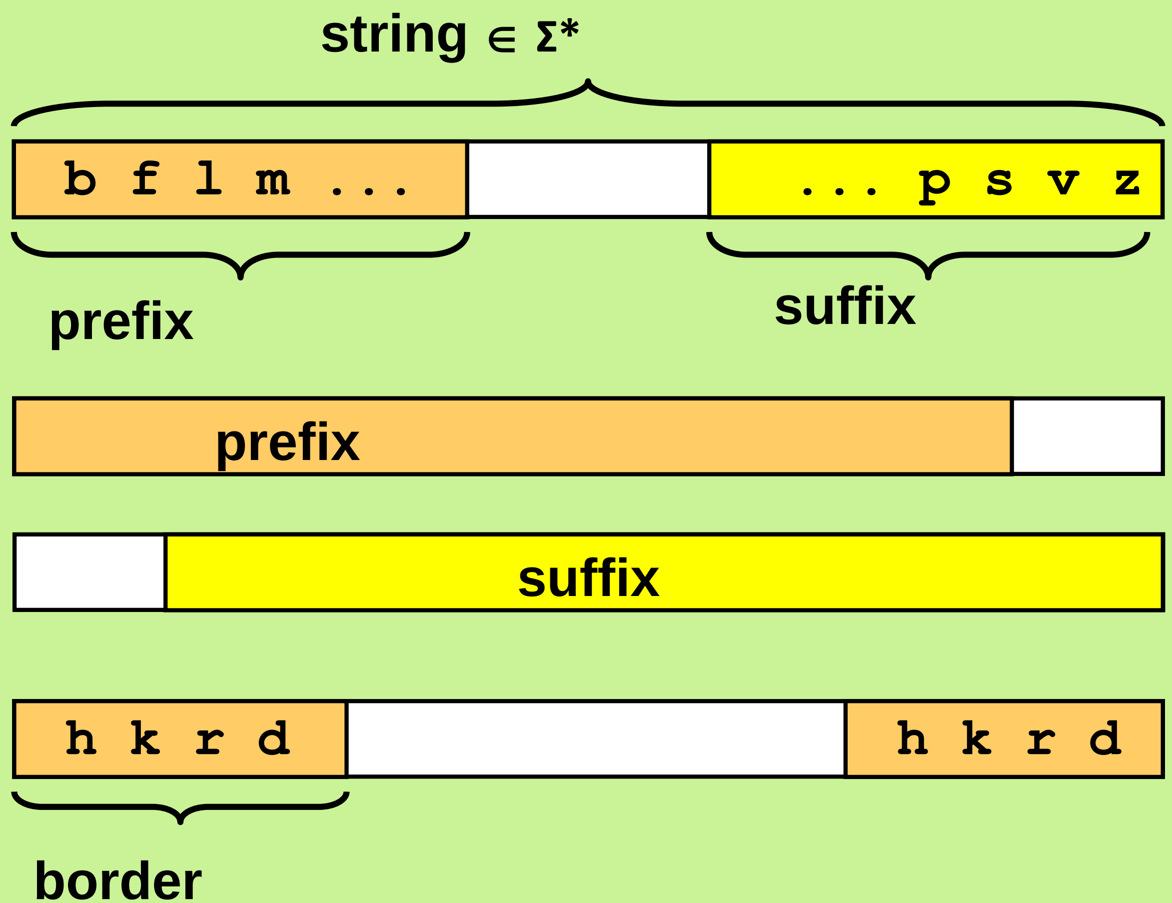
Datové struktury a algoritmy, B6B36DSA
01/2018, Lekce 14

https://cw.fel.cvut.cz/wiki/courses/b6b36dsa/start

Evropský sociální fond
Praha & EU: Investujeme do vaší budoucnosti

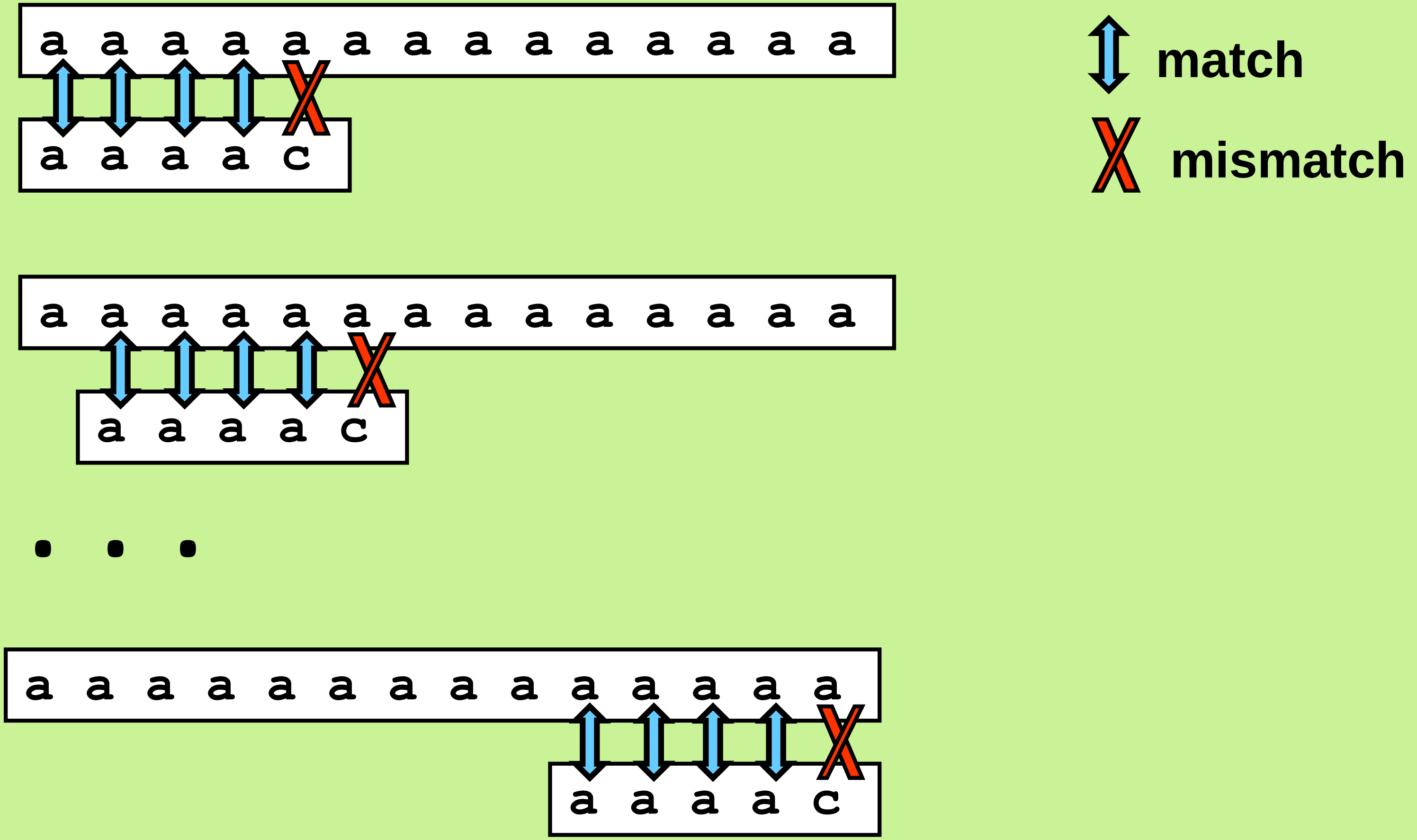
String matching (hledání řetězce v textu)

Σ – alphabet , e.g. Σ = {a, b, c, d .... }



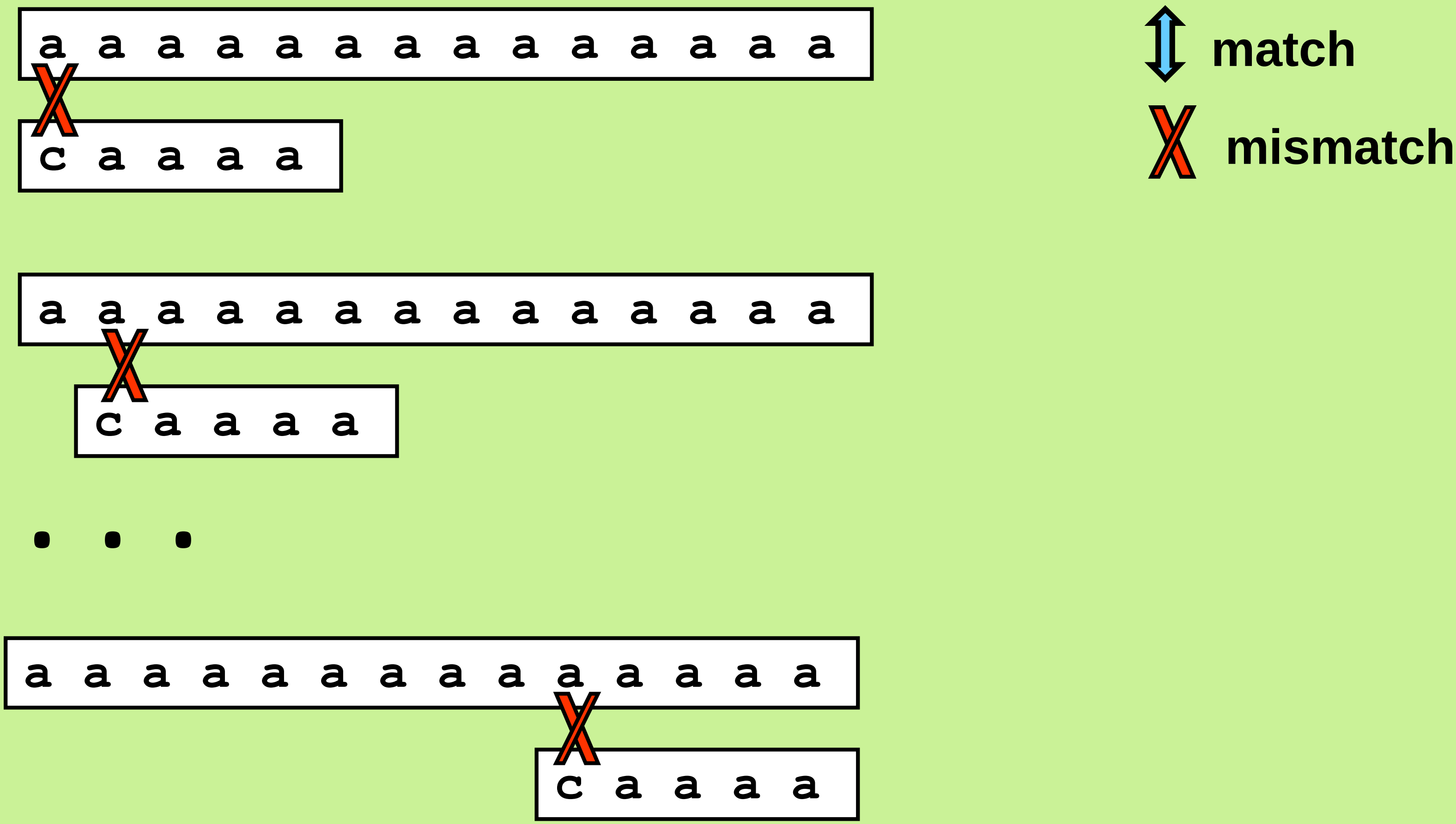
Naive algorithm

worst case



Naive algorithm

best case



Karp-Rabin Algorithm

t_5: 5 3 2 — $\phi(t_5) = 532$

t_6: 3 2 3 — $\phi(t_6) = 323$

T:

1	2	3	4	5	6	7	8	9	10	11	12
1	2	3	4	5	3	2	3	5	6	3	1

$\phi(t_5) = t_7+10(t_6+10(t_5)) = 2 + 10(3+10(5)) = 2 + 10(3+50) =$

$= 2+10\cdot 53 = 532$

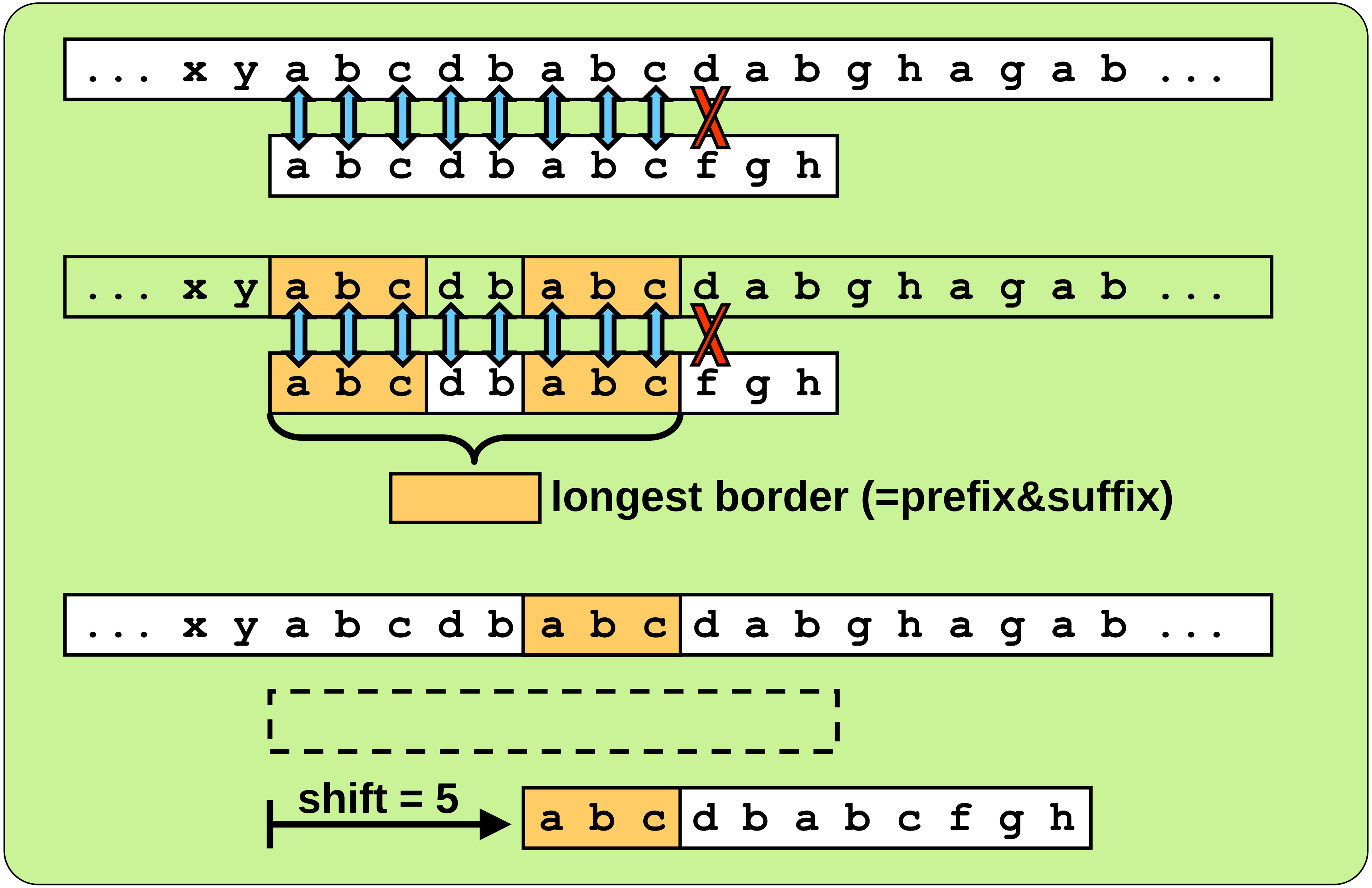
$\phi(t_6) = 10(\phi(t_5) - 10^2 \cdot t_5) + t_{5+3} = 10(532 - 100\cdot 5) + 3 =$

$= 10(532 - 500) + 3 = 10\cdot 32 + 3 = 323$

$$\phi(t_i) = t_{i+m-1} + 10(t_{i+m-2} + 10(\cdots + 10t_i)\cdots)$$

$$\phi(t_{i+1}) = 10(\phi(t_i) - 10^{m-1}t_i) + t_{i+m}$$

Knuth–Morris–Pratt algorithm

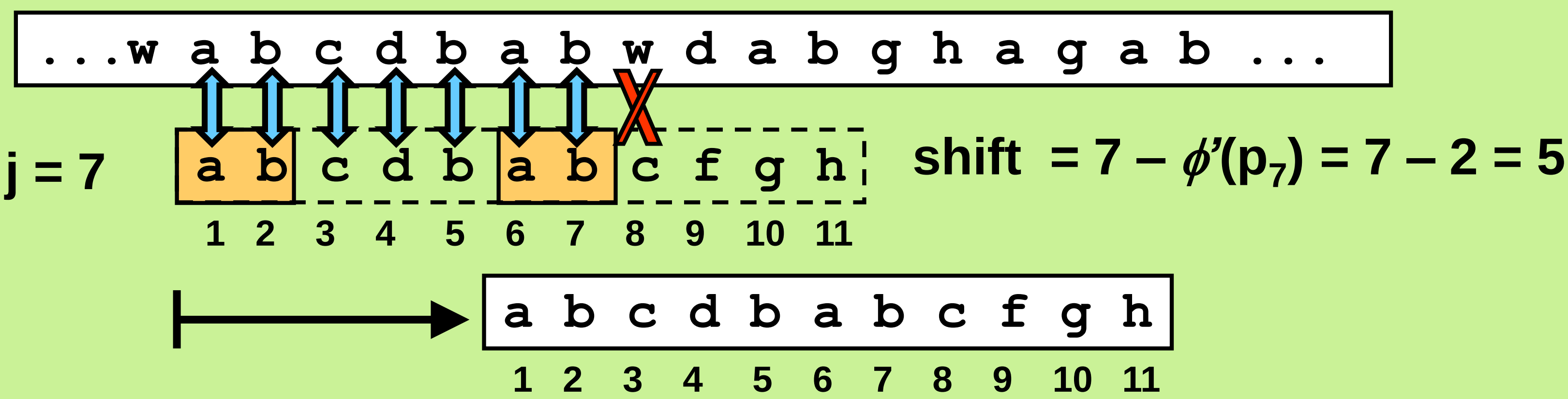


Knuth - Morris - Pratt algorithm

j	1	2	3	4	5	6	7	8	9	10	11
p_j	a	b	c	d	b	a	b	c	f	g	h
$\phi'(p_j)$	0	0	0	0	0	1	2	3	0	0	0

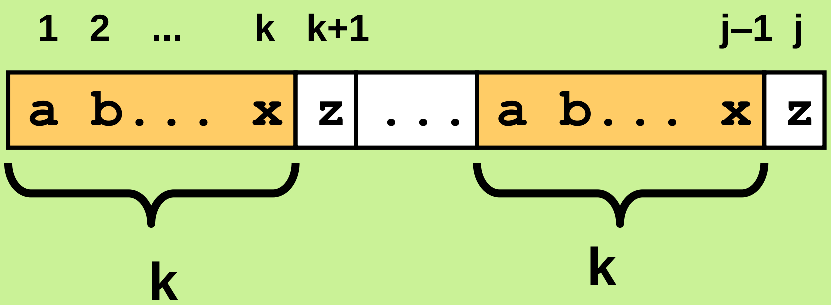
Longest border size in substr $(p_1,\ldots,p_j)$

shift after mismatch t_i X p_{j+1} = $j - \phi'(p_j)$



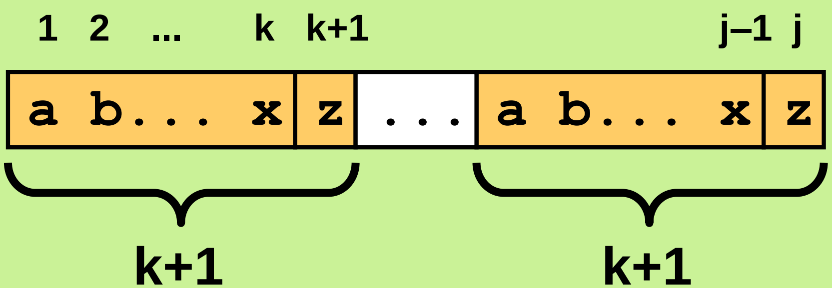
Knuth - Morris - Pratt algorithm

computing $\phi'(p_j)$ from $\phi'(p_{j-1})$



$k = \phi'(j-1)$ = size of longest border in $(p_1, p_2, \ldots p_{j-1})$

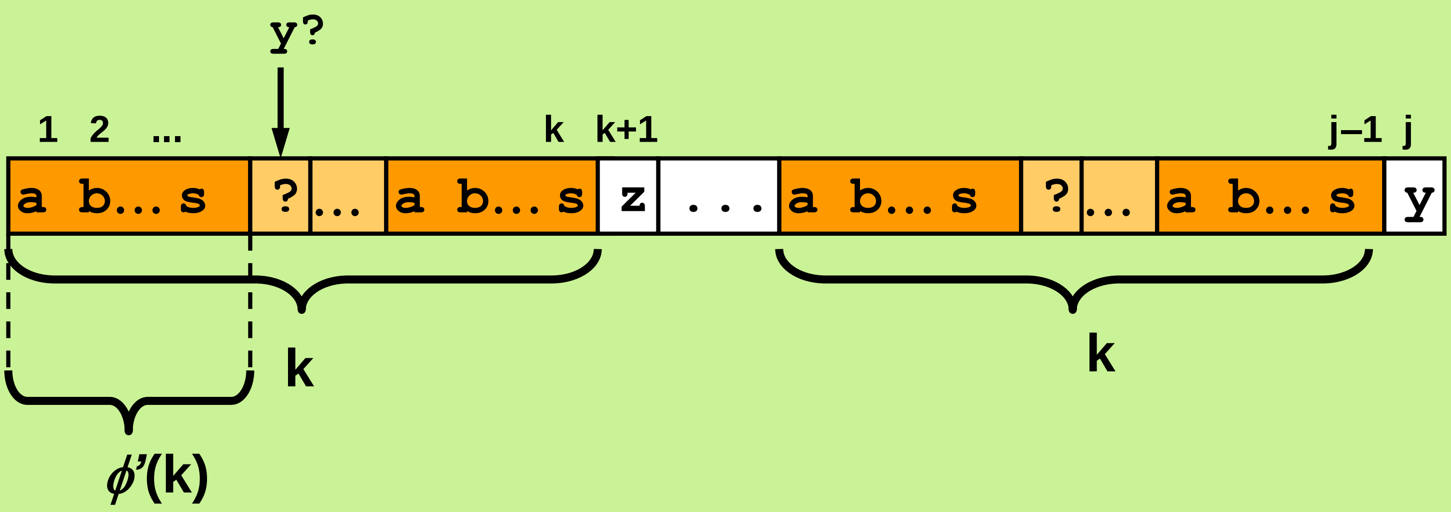
if $p_{k+1} = p_j$ then $\phi'(p_j) = k+1$



Knuth - Morris - Pratt algorithm

computing $\phi'(p_j)$ from $\phi'(p_{j-1})$

if $p_{k+1} \neq p_j$ then longest border of $(p_1,\ldots,p_j)$ might be the longest border of $(p_1,\ldots,p_k)$ ...



... and if it is not then continue search for the longest border in $(p_1,\ldots,p_{\phi'(k)})$ recursively

Knuth - Morris - Pratt algorithm

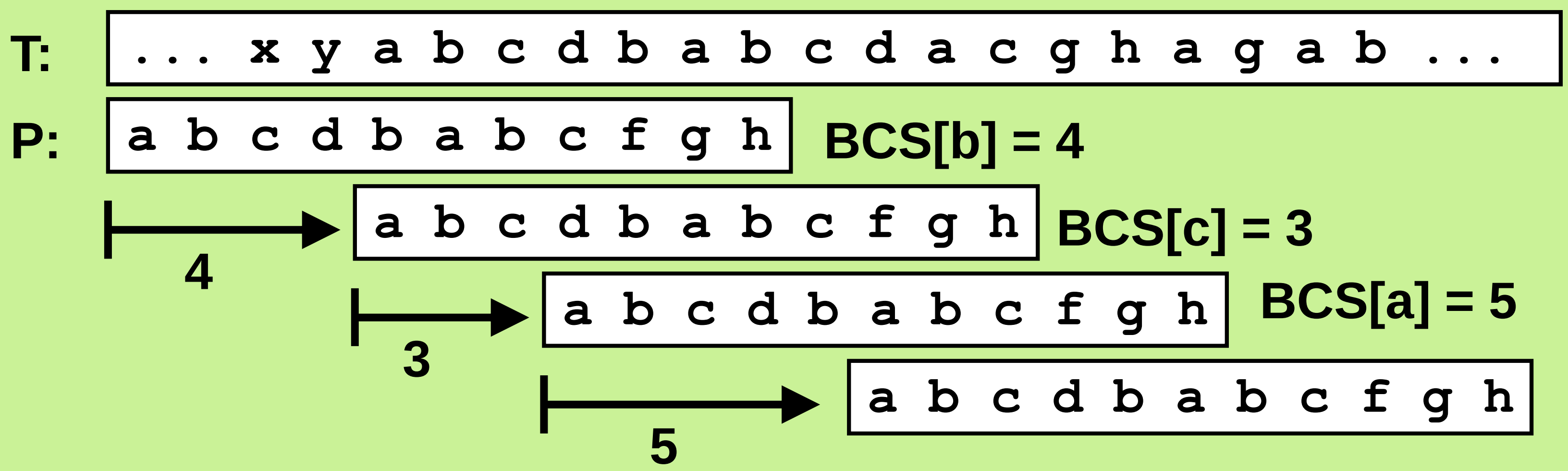
Compute shift function ϕ'

$\phi'(1) \leftarrow 0$
$k \leftarrow 0$
for $j \leftarrow 2..\text{length}(P)$ **do**
  while $k > 0$ **and** $p_{k+1} \neq p_j$ **do**
    $k \leftarrow \phi'[k]$
  end while
  if $p_{k+1} = p_j$ **then** $k \leftarrow k+1$ **end if**
  $\phi'(j) \leftarrow k$
end for

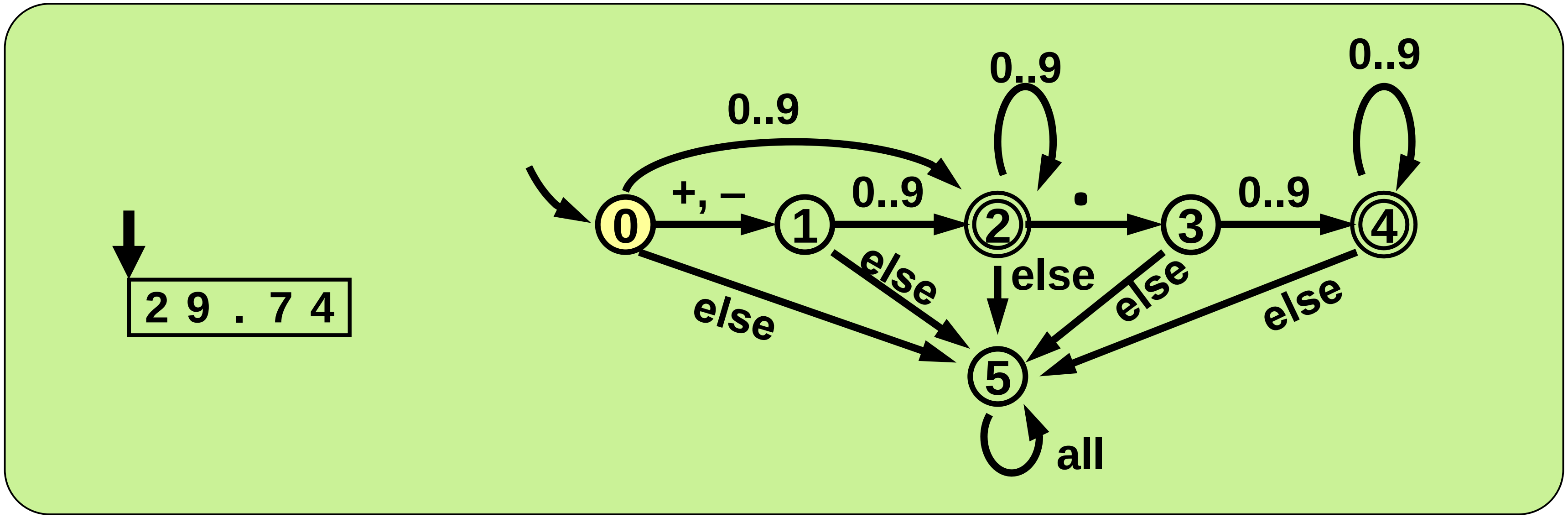
Boyer - Moore algorithm

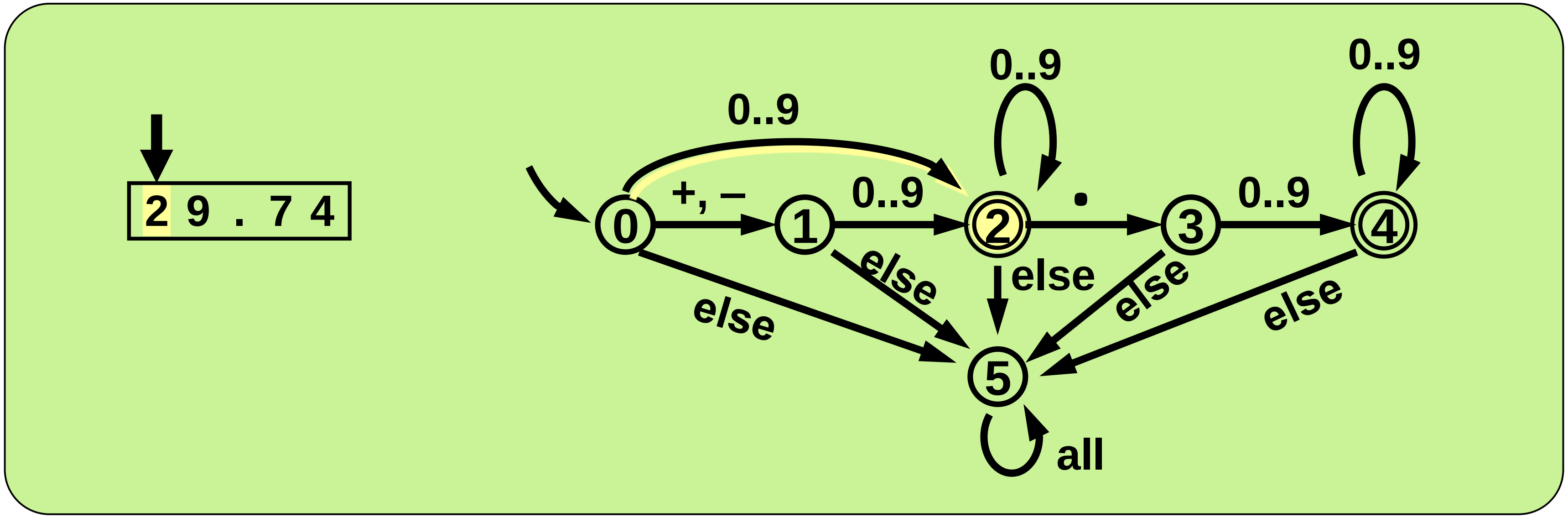
a b c d b a b c f g h

p	a	b	c	d	f	g	h	other
BCS[p]	5	4	3	7	2	1	11	11

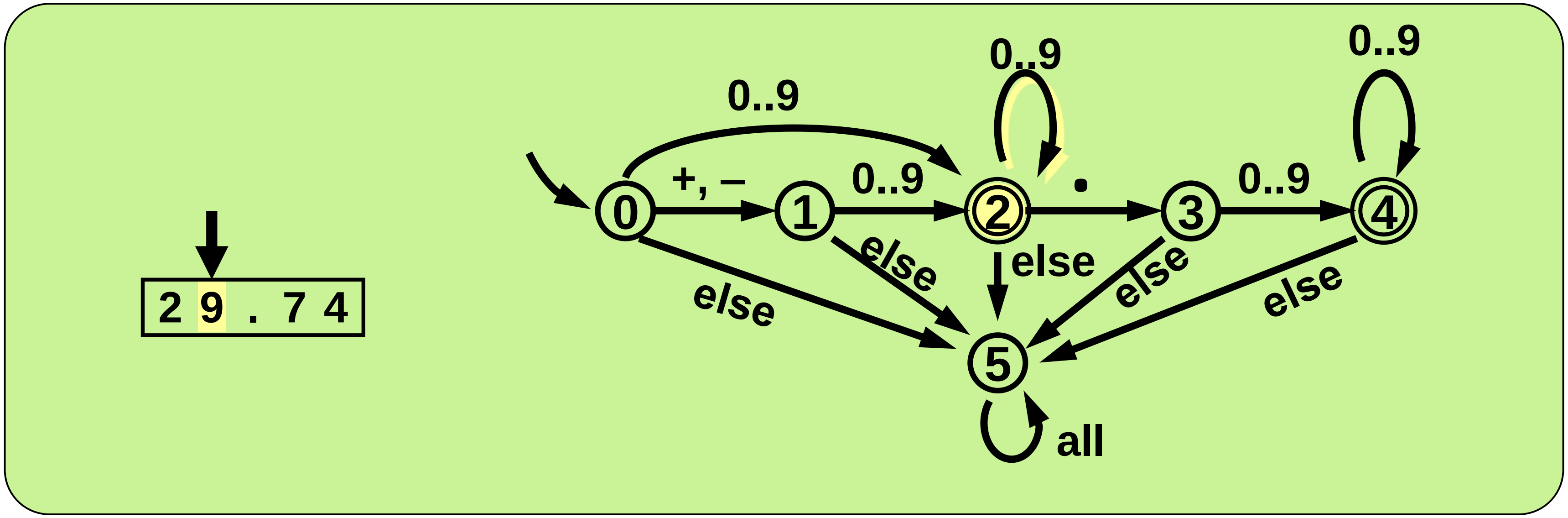


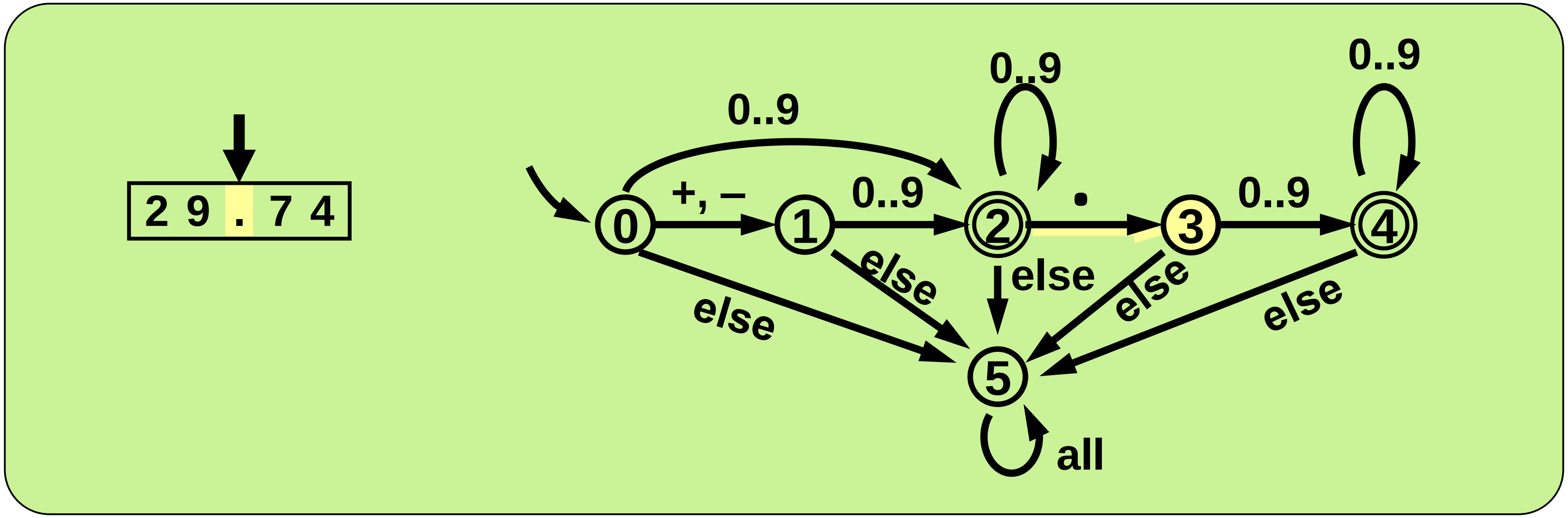
Konečný automat (finite-state automaton)



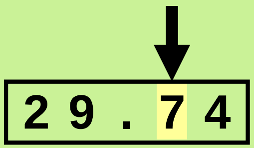


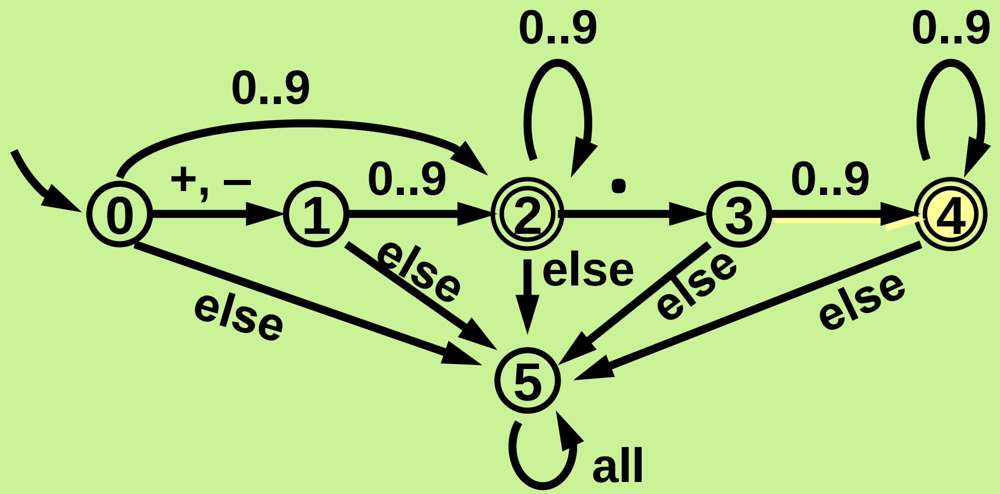
Konečný automat

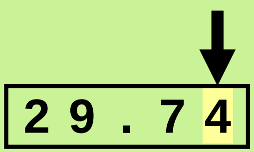


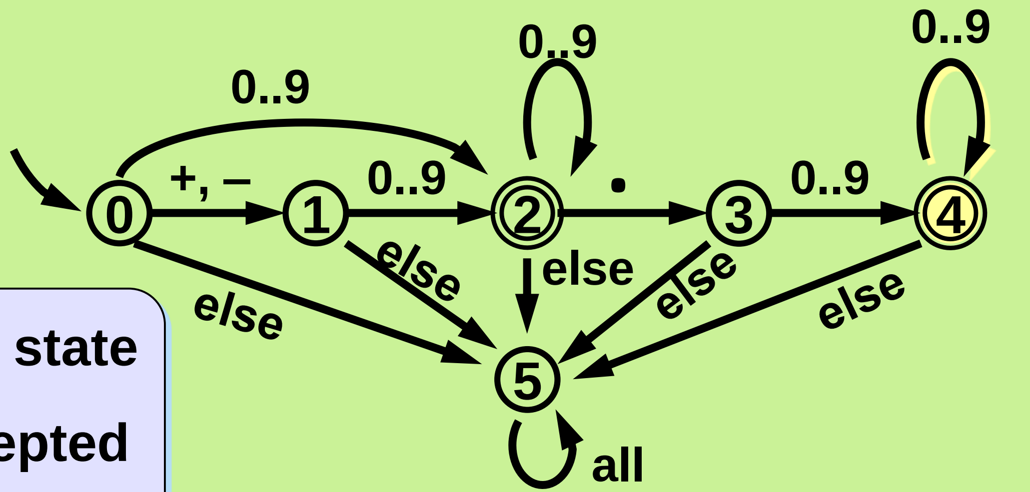


Konečný automat





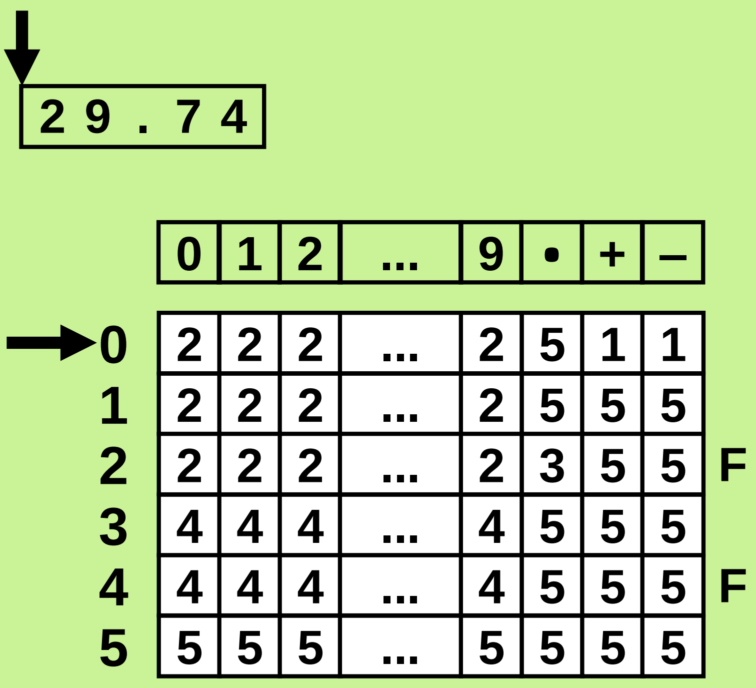




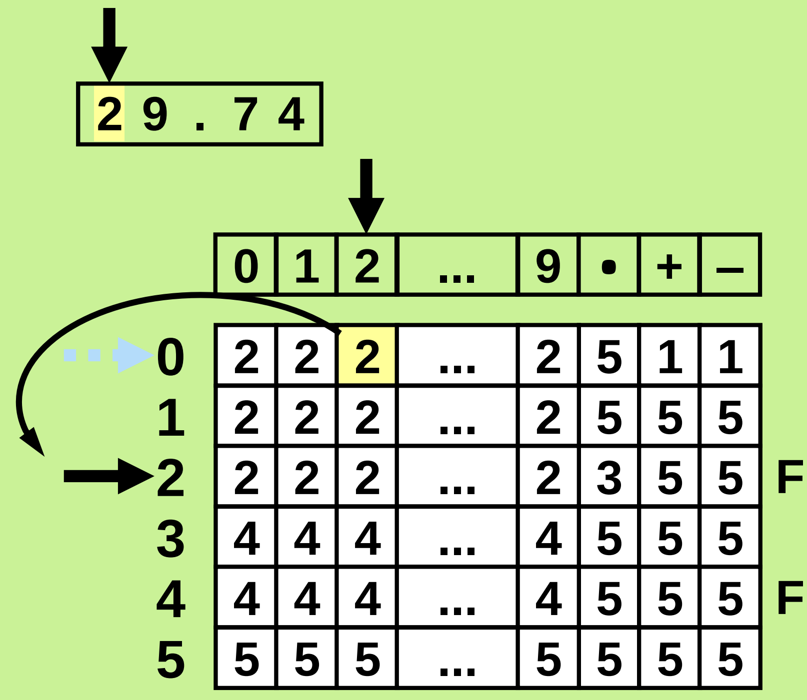
④ is an accepting state

— 29.74 is accepted

Konečný automat



	0	1	2	...	9	•	+	–	
0	2	2	2	...	2	5	1	1	
1	2	2	2	...	2	5	5	5	
2	2	2	2	...	2	3	5	5	F
3	4	4	4	...	4	5	5	5	
4	4	4	4	...	4	5	5	5	F
5	5	5	5	...	5	5	5	5	



	0	1	2	...	9	•	+	–	
0	2	2	2	...	2	5	1	1	
1	2	2	2	...	2	5	5	5	
2	2	2	2	...	2	3	5	5	F
3	4	4	4	...	4	5	5	5	
4	4	4	4	...	4	5	5	5	F
5	5	5	5	...	5	5	5	5	

Konečný automat

2 9 . 7 4

	0	1	2	...	9	•	+	–	
0	2	2	2	...	2	5	1	1	
1	2	2	2	...	2	5	5	5	
2	2	2	2	...	2	3	5	5	F
3	4	4	4	...	4	5	5	5	
4	4	4	4	...	4	5	5	5	F
5	5	5	5	...	5	5	5	5	

2 9 . 7 4

	0	1	2	...	9	•	+	–	
0	2	2	2	...	2	5	1	1	
1	2	2	2	...	2	5	5	5	
2	2	2	2	...	2	3	5	5	F
3	4	4	4	...	4	5	5	5	
4	4	4	4	...	4	5	5	5	F
5	5	5	5	...	5	5	5	5	

Konečný automat

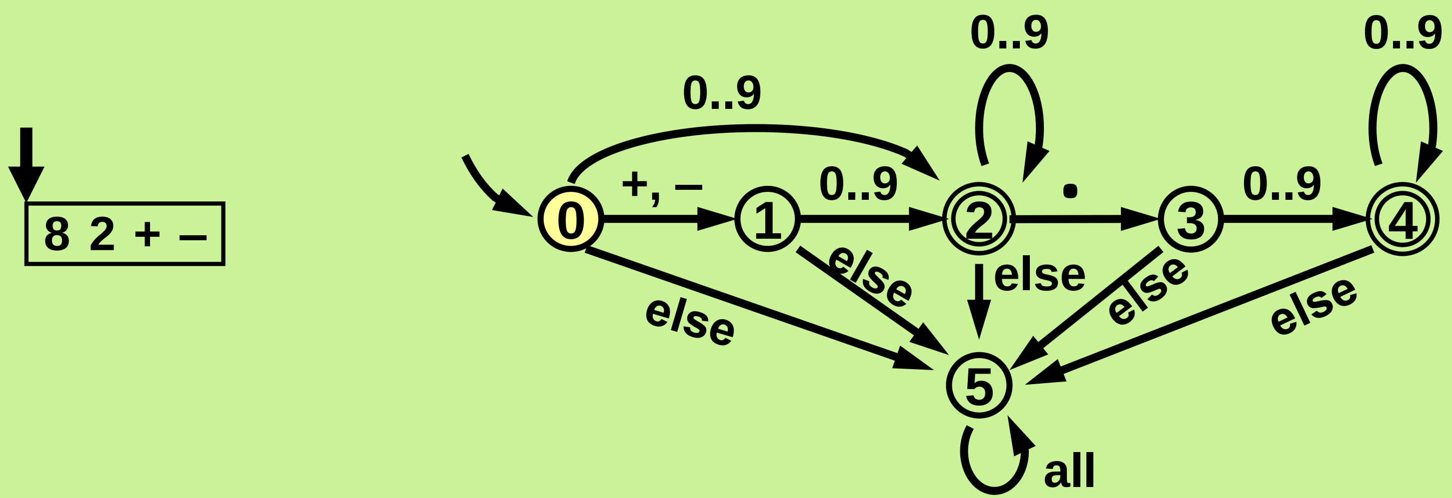
2 9 . 7 4

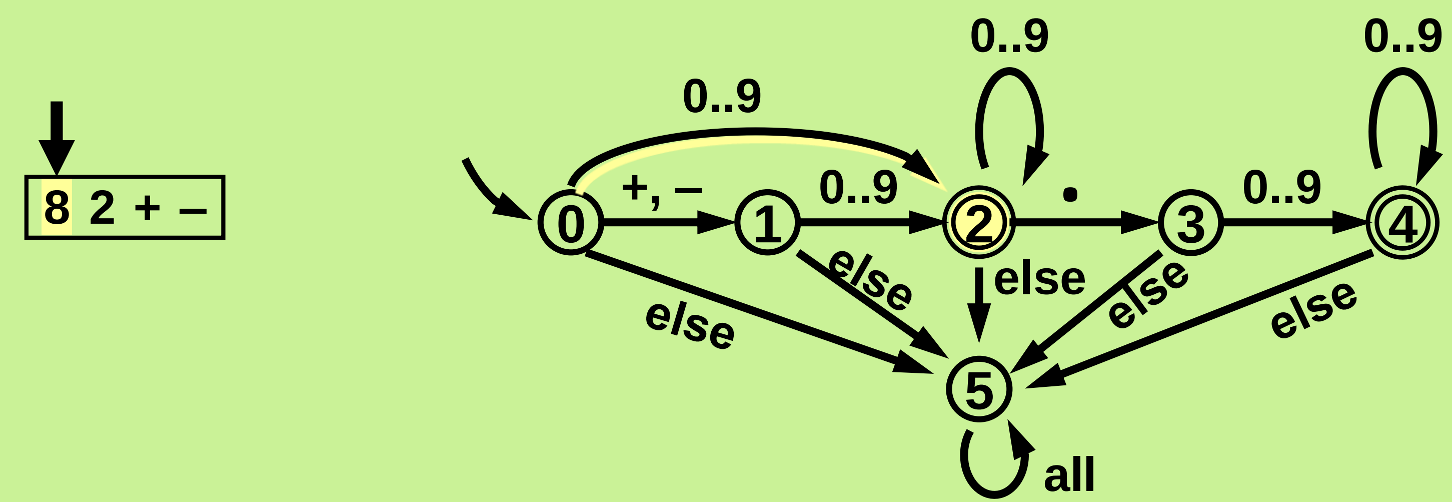
	0	1	2	…7	9	•	+	–	
0	2	2	2	…	2	5	1	1	
1	2	2	2	…	2	5	5	5	
2	2	2	2	…	2	3	5	5	F
3	4	4	4	…4	4	5	5	5	
4	4	4	4	…	4	5	5	5	F
5	5	5	5	…	5	5	5	5	

2 9 . 7 4

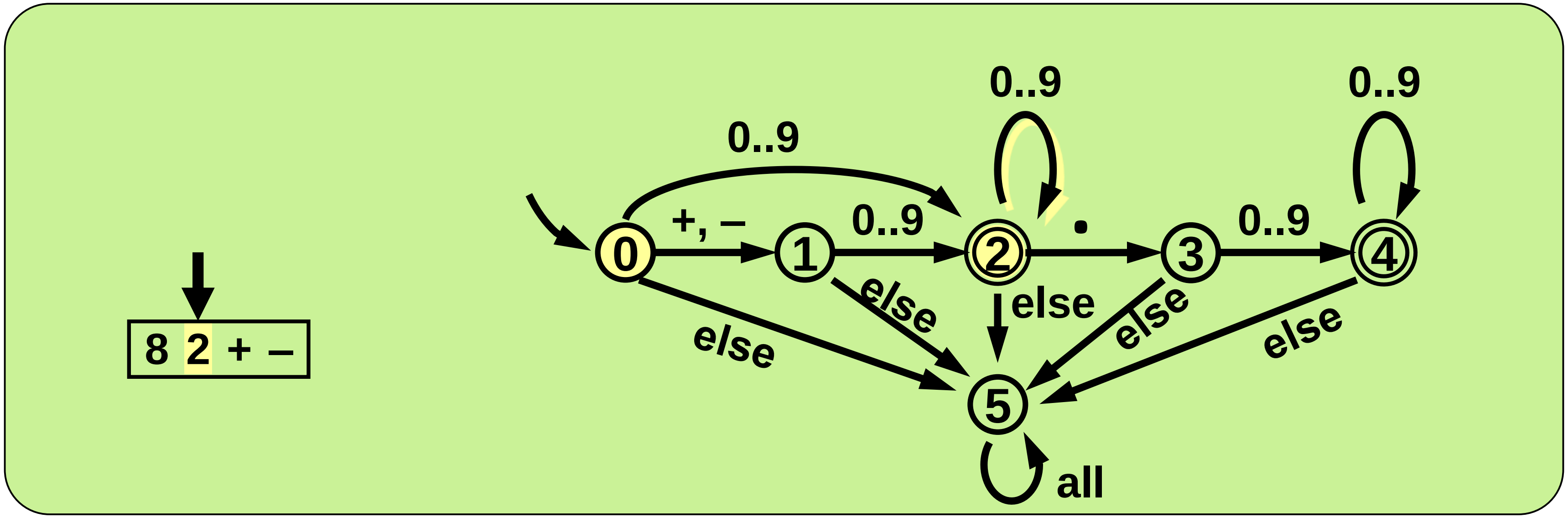
	0	1	2	4…	9	•	+	–	
0	2	2	2	…	2	5	1	1	
1	2	2	2	…	2	5	5	5	
2	2	2	2	…	2	3	5	5	F
3	4	4	4	…	4	5	5	5	
4	4	4	4	4…	4	5	5	5	F
5	5	5	5	…	5	5	5	5	

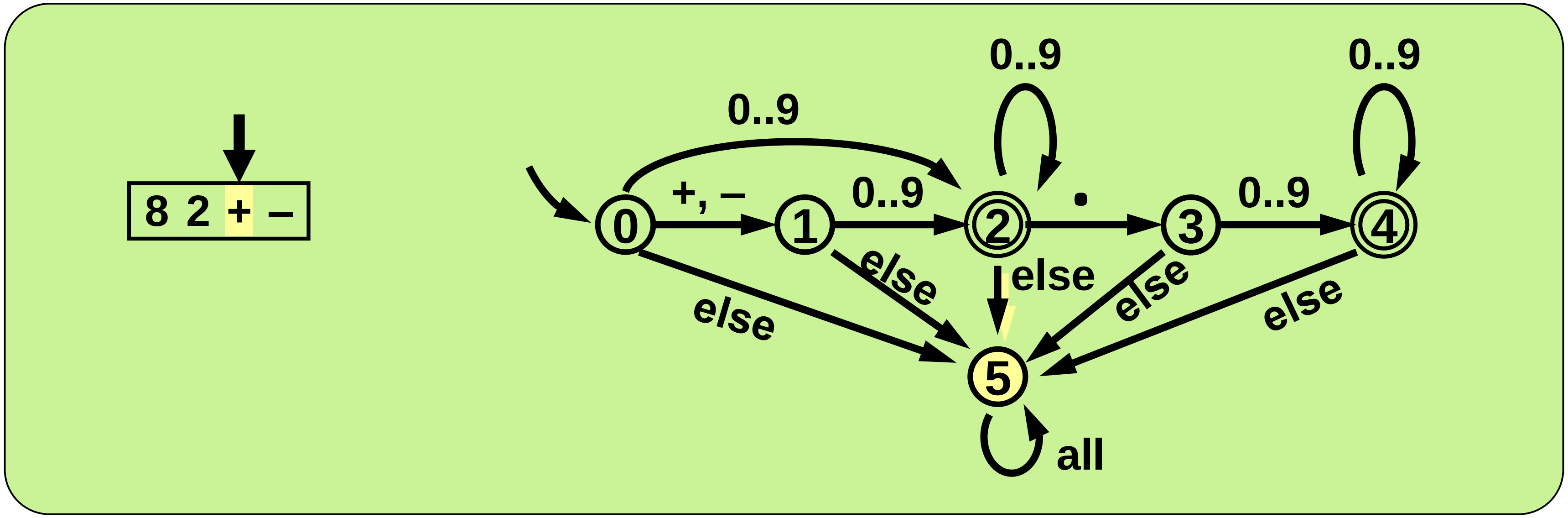
Konečný automat



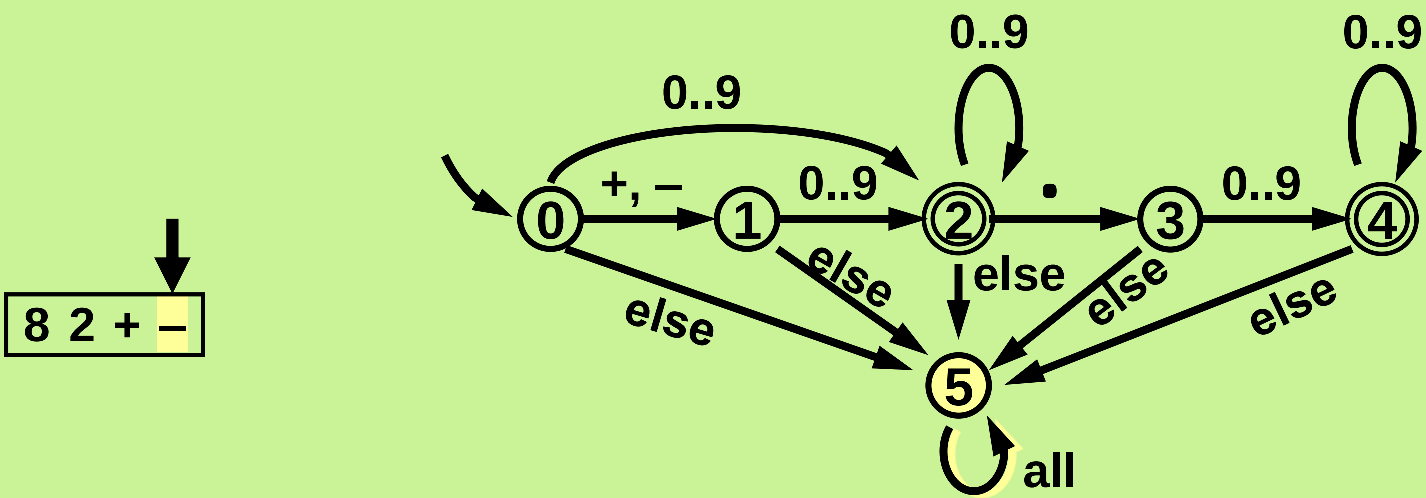


Konečný automat



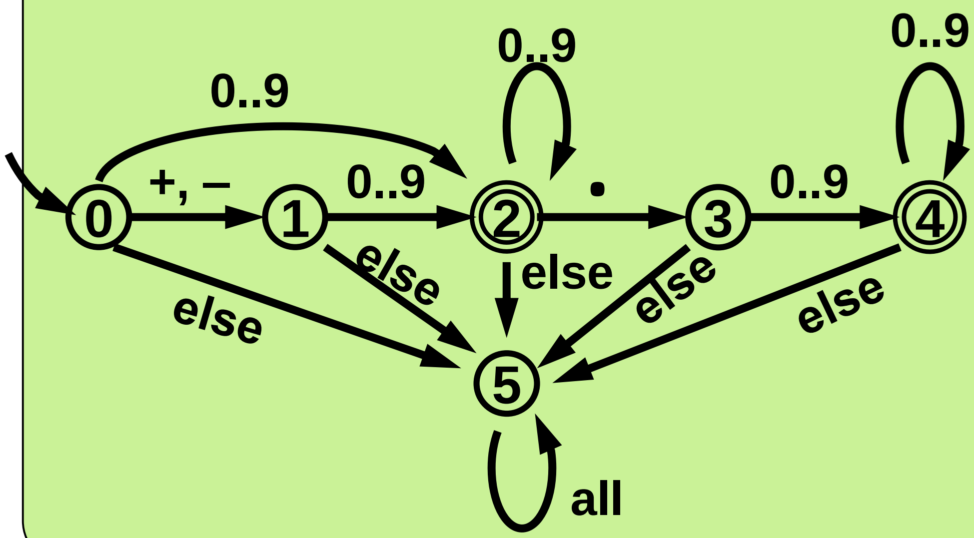


Konečný automat



⑤ is not accepting state — “82+–” is not accepted

Konečný automat



	0	1	2	...	9	•	+	–	
⓪	2	2	2	...	2	5	1	1	
①	2	2	2	...	2	5	5	5	
⓶	2	2	2	...	2	3	5	5	F
③	4	4	4	...	4	5	5	5	
⓸	4	4	4	...	4	5	5	5	F
⑤	5	5	5	...	5	5	5	5	

Q: finite set of states ⓪①⓶③⓸⑤

Σ: finite alphabet {0,1,2,...,9, •,+, –}

δ: mapping Q× Σ → Q, all labelled arows or the whole table

q_0: initial state ⓪

F: set of final (accepting) states ⓶ ⓸

Konstrukce KA pro „String matching“

		b	a	n	c	d	...	
	0	0	0	0	0	0	...	
b	1						...	
a	2						...	
n	3						...	
a	4						...	
n	5						...	
a	6						...	F

$M \leftarrow (\{q_0, q_1, \ldots, q_m\}, \Sigma, \delta, q_0, \{q_m\})$

for $\forall a \in \Sigma$ **do**

  $\delta(q_0, a) \leftarrow \{q_0\}$ {self-loop of the initial state}

end for

Konstrukce KA pro „String matching“

i = 1

		b	a	n	c	d	...
	0	1	0	0	0	0	...
b	1	1	0	0	0	0	...
a	2						...
n	3						...
a	4						...
n	5						...
a	6						... F

r = δ(0,b) = 0

5: **for** $i \leftarrow 1..m$ **do**
6: $r \leftarrow \delta(q_{i-1}, p_i)$
7: $\delta(q_{i-1}, p_i) \leftarrow q_i$ {forward transition}
8: **for** $\forall a \in \Sigma$ **do**
9: $\delta(q_i, a) \leftarrow \delta(r, a)$
10: **end for**
11: **end for**

Konstrukce KA pro „String matching“

		b	a	n	c	d	...	
	0	1	0	0	0	0	...	
b	1	1	2	0	0	0	...	
a	2	1	0	0	0	0	...	
n	3						...	
a	4						...	
n	5						...	
a	6						...	F

i = 2

r = δ(1,a) = 0

for $i \leftarrow 1..m$ **do**
    $r \leftarrow \delta(q_{i-1}, p_i)$
    $\delta(q_{i-1}, p_i) \leftarrow q_i$ {forward transition}
    for $\forall a \in \Sigma$ **do**
        $\delta(q_i, a) \leftarrow \delta(r, a)$
    end for
end for

Konstrukce KA pro „String matching“

		b	a	n	c	d	...
	0	1	0	0	0	0	...
b	1	1	2	0	0	0	...
a	2	1	0	3	0	0	...
n	3	1	0	0	0	0	...
a	4						...
n	5						...
a	6						... F

i = 3

r = δ(2,n) = 0

for $i \leftarrow 1..m$ **do**
  $r \leftarrow \delta(q_{i-1}, p_i)$
  $\delta(q_{i-1}, p_i) \leftarrow q_i$ {forward transition}
  for $\forall a \in \Sigma$ **do**
    $\delta(q_i, a) \leftarrow \delta(r, a)$
  end for
end for

Konstrukce KA pro „String matching“

		b	a	n	c	d	...
	0	1	0	0	0	0	...
b	1	1	2	0	0	0	...
a	2	1	0	3	0	0	...
n	3	1	4	0	0	0	...
a	4	1	0	5	0	0	...
n	5	1	6	0	0	0	...
a	6	1	0	0	0	0	...

i = 6

F r = δ(5,a) = 0

for $i \leftarrow 1..m$ **do**
    $r \leftarrow \delta(q_{i-1}, p_i)$
    $\delta(q_{i-1}, p_i) \leftarrow q_i$ {forward transition}
    for $\forall a \in \Sigma$ **do**
        $\delta(q_i, a) \leftarrow \delta(r, a)$
    end for
end for

Srovnání metod pro jeden vzorek

Algorithm	**Preprocessing time**	**Matching time**[1]
Naive string search algorithm	0 (no preprocessing)	$\Theta((n-m+1)\ m)$
Rabin–Karp string search algorithm	$\Theta(m)$	average $\Theta(n+m)$, worst $\Theta((n-m+1)\ m)$
Finite-state automaton based search	$\Theta(m\ \lvert\Sigma\rvert)$	$\Theta(n)$
Knuth–Morris–Pratt algorithm	$\Theta(m)$	$\Theta(n)$
Boyer–Moore string search algorithm	$\Theta(m + \lvert\Sigma\rvert)$	$\Omega(n/m)$, $O(n)$

kde m je délka vzorku a n délka prohledávaného textu

The End