Factor Oracle for Machine Improvisation

Jaime Arias

Université de Bordeaux, LaBRI, UMR 5800
Inria - Bordeaux Sud-Ouest

August 2016

Preliminaries

Preliminaries

Word

A word s is a finite sequence $s = s_1 s_2 \dots s_m$ of length $|s| = m$ on a finite alphabet Σ.

$s =$

a	b	b	c	a	b	c	d	a	b	c

Factor

A word $x \in \Sigma^*$ is a factor of s if and only if s can be written $s = uxv$ with $u, v \in \Sigma^*$. Given integers i, j where $1 \leq i \leq j \leq m$, we denote a *factor* of s as $s[i \dots j] = s_i s_{i+1} \dots s_j$.

$s =$

a	b	b	c	a	b	c	d	a	b	c

$s[3, 5]$

Preliminaries

Prefix

A factor x of s is a prefix of s if $s = xu$ with $u \in \Sigma^*$. The ith *prefix* of s, denoted $pref_s(i)$, is the prefix $s[1 \ldots i]$.

$$s = \underbrace{\texttt{a}\,\texttt{b}\,\texttt{b}\,\texttt{c}}_{pref_s(4)}\,\texttt{a}\,\texttt{b}\,\texttt{c}\,\texttt{d}\,\texttt{a}\,\texttt{b}\,\texttt{c}$$

Suffix

A factor x of s is a suffix of s if $s = ux$ with $u \in \Sigma^*$. The ith *suffix* of s, denoted $suff_s(i)$, is the suffix $s[i \ldots m]$.

$$s = \texttt{a}\,\texttt{b}\,\texttt{b}\,\texttt{c}\,\texttt{a}\,\underbrace{\texttt{b}\,\texttt{c}\,\texttt{d}\,\texttt{a}\,\texttt{b}\,\texttt{c}}_{suff_s(6)}$$

Preliminaries

Longest Repeated Suffix (LRS)

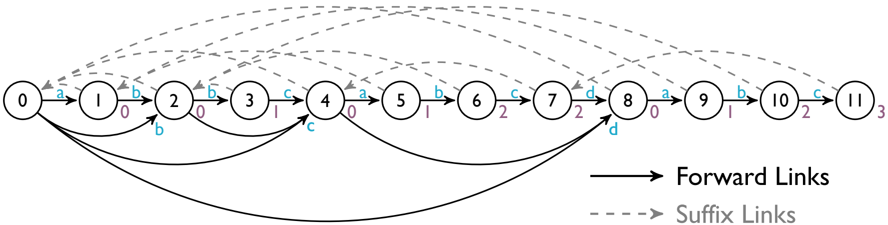
A factor x of s is the longest repeated suffix of s if x is a suffix of s and $|x|$ is maximal.

$$s = \begin{array}{|c|c|c|c|}\hline \texttt{a} & \texttt{b} & \texttt{b} & \texttt{c} \\ \hline\end{array}\underbrace{\begin{array}{|c|c|c|}\hline \texttt{a} & \texttt{b} & \texttt{c} \\ \hline\end{array}}_{lrs(s)}\begin{array}{|c|c|c|c|}\hline \texttt{d} & \texttt{a} & \texttt{b} & \texttt{c} \\ \hline\end{array}$$

Factor Oracle

Factor Oracle

Overview

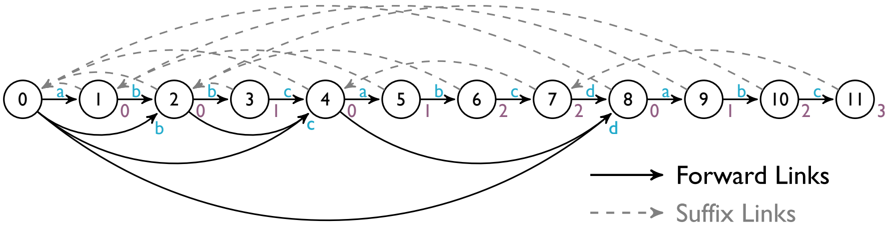


Factor Oracle

The factor oracle of a word s of length m is a *deterministic finite automaton* (Q, q_0, F, δ) where $Q = \{0, 1, \ldots, m\}$ is the set of states, $q_0 = 0$ is the starting state, $F = Q$ is the set of terminal states and δ is the transition function.

Factor Oracle

Overview

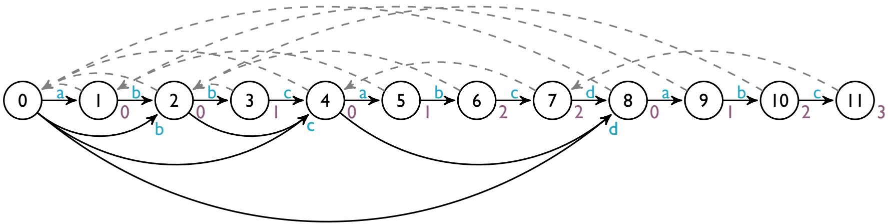


Suffix Link

The suffix link of a state i of the factor oracle of a word s, is equal to the state in which the *longest repeated suffix (lrs)* of $s[1 \ldots i]$ is recognized.

Factor Oracle

Overview

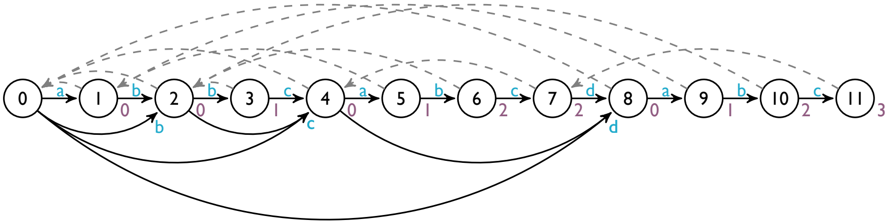


Suffix Links

- $s = \texttt{abbcabcdabc}$

Factor Oracle

Overview

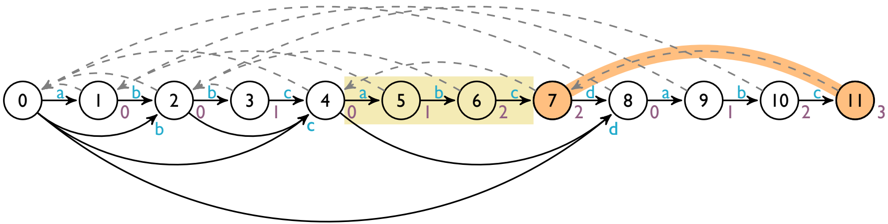


Suffix Links

- $s =$ abbcabcdabc
- $lrs(s) =$ abc

Factor Oracle

Overview



Suffix Links

- $s = \texttt{abbcabcdabc}$
- $lrs(s) = \texttt{abc}$
- $S(11) = 7$

Factor Oracle

Algorithm

Algorithm 1 Construction of a Factor Oracle

```
function FactorOracle(p = p_1 p_2 ... p_m)
    Create a new oracle P with an initial state 0
    S_P(0) ← −1
    for i ← 1, m do
        Oracle(p = p_1 p_2 ... p_i) ← AddLetter(Oracle(p = p_1 p_2 ... p_{i−1}), p_i)
    end for
    return Oracle(p = p_1 p_2 ... p_m)
end function
```

Factor Oracle

Algorithm

Algorithm 1 Construction of a Factor Oracle

```
function FactorOracle(p = p_1 p_2 ... p_m)
    Create a new oracle P with an initial state 0
    S_P(0) ← −1
    for i ← 1, m do
        Oracle(p = p_1 p_2 ... p_i) ← AddLetter(Oracle(p = p_1 p_2 ... p_{i−1}), p_i)
    end for
    return Oracle(p = p_1 p_2 ... p_m)
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

Factor Oracle

Algorithm

Algorithm 1 Construction of a Factor Oracle

function FactorOracle($p = p_1 p_2 \ldots p_m$)
    Create a new oracle P with an initial state 0
    $S_P(0) \leftarrow -1$
    for $i \leftarrow 1, m$ **do**
        $Oracle(p = p_1 p_2 \ldots p_i) \leftarrow \text{AddLetter}(Oracle(p = p_1 p_2 \ldots p_{i-1}), p_i)$
    end for
    return $Oracle(p = p_1 p_2 \ldots p_m)$
end function

$p =$

a	b	b	c	a	b	c	d	a	b	c

Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

function AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
  Create state $m + 1$
  Create a new transition from m to $m + 1$ labeled by σ $\triangleright\ \delta(m, \sigma) = m + 1$
  $k \leftarrow S_p(m)$
  $\pi_1 \leftarrow m$
  …
end function

slide 1

Factor Oracle

Algorithm

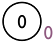
Algorithm 2 Incremental update of Factor Oracle

function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
  Create state $m + 1$
  Create a new transition from m to $m + 1$ labeled by σ $\quad \triangleright\ \delta(m, \sigma) = m + 1$
  $k \leftarrow S_p(m)$
  $\pi_1 \leftarrow m$
  …
end function

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 0$



slide 2

Factor Oracle

Algorithm

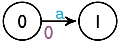
Algorithm 2 Incremental update of Factor Oracle

function AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
  Create state $m + 1$
  Create a new transition from m to $m + 1$ labeled by σ $\triangleright\ \delta(m, \sigma) = m + 1$
  $k \leftarrow S_p(m)$
  $\pi_1 \leftarrow m$
  …
end function

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 0$



slide 3

Factor Oracle

Algorithm

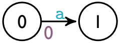
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter(Oracle(p = p_1, p_2 … p_m), σ)
    Create state m + 1
    Create a new transition from m to m + 1 labeled by σ        ▷ δ(m, σ) = m + 1
    k ← S_p(m)
    π_1 ← m
    …
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 0 \quad k = -1 \quad \pi_1 = 0$



slide 4

Factor Oracle

Algorithm

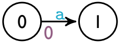
Algorithm 2 Incremental update of Factor Oracle

function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    while $k > -1$ and there is no transition from k by σ **do**
        Create a new transition from k to $m+1$ by σ $\triangleright\ \delta(k, \sigma) = m + 1$
        $\pi_1 \leftarrow k$
        $k \leftarrow S_p(k)$
    end while
    …
end function

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 0 \quad k = -1 \quad \pi_1 = 0$



slide 5

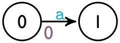
Algorithm 2 Incremental update of Factor Oracle

function AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
    …
    if $k = -1$ **then**
        $S_{p\sigma} \leftarrow 0$
        $lrs_{p\sigma} \leftarrow 0$
    else
        $S_{p\sigma} \leftarrow$ state that leads the transition from k by σ
        $lrs_{p\sigma} \leftarrow$ LengthCommonSuffix($\pi_1, S(m+1) - 1$) + 1
    end if
    …
end function

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 0 \quad k = -1 \quad \pi_1 = 0$



slide 6

Factor Oracle

Algorithm

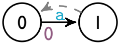
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter(Oracle(p = p_1, p_2 ... p_m), σ)
    ...
    if k = −1 then
        S_pσ ← 0
        lrs_pσ ← 0
    else
        S_pσ ← state that leads the transition from k by σ
        lrs_pσ ← LengthCommonSuffix(π_1, S(m + 1) − 1) + 1
    end if
    ...
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 0 \quad k = -1 \quad \pi_1 = 0$



Factor Oracle

Algorithm

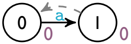
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter(Oracle(p = p_1, p_2 ... p_m), σ)
    ...
    if k = −1 then
        S_{pσ} ← 0
        lrs_{pσ} ← 0
    else
        S_{pσ} ← state that leads the transition from k by σ
        lrs_{pσ} ← LengthCommonSuffix(π_1, S(m + 1) − 1) + 1
    end if
    ...
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 0 \quad k = -1 \quad \pi_1 = 0$



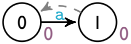
Algorithm 2 Incremental update of Factor Oracle

function AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
    Create state $m + 1$
    Create a new transition from m to $m + 1$ labeled by σ $\triangleright\ \delta(m, \sigma) = m + 1$
    $k \leftarrow S_p(m)$
    $\pi_1 \leftarrow m$
    …
end function

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 1$



slide 9

Factor Oracle

Algorithm

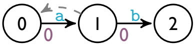
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter(Oracle(p = p_1, p_2 ... p_m), σ)
    Create state m + 1
    Create a new transition from m to m + 1 labeled by σ        ▷ δ(m, σ) = m + 1
    k ← S_p(m)
    π_1 ← m
    ...
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 1$



slide 10

Factor Oracle

Algorithm

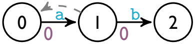
Algorithm 2 Incremental update of Factor Oracle

function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    Create state $m + 1$
    Create a new transition from m to $m + 1$ labeled by σ $\triangleright\ \delta(m, \sigma) = m + 1$
    $k \leftarrow S_p(m)$
    $\pi_1 \leftarrow m$
    …
end function

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 1 \quad k = 0 \quad \pi_1 = 1$



slide 11

Factor Oracle

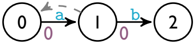
Algorithm

Algorithm 2 Incremental update of Factor Oracle

function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    while $k > -1$ and there is no transition from k by σ **do**
        Create a new transition from k to $m+1$ by σ $\triangleright\ \delta(k,\sigma) = m+1$
        $\pi_1 \leftarrow k$
        $k \leftarrow S_p(k)$
    end while
    …
end function

$p =$	a	b	b	c	a	b	c	d	a	b	c

$m = 1 \quad k = 0 \qquad \pi_1 = 1$



slide 12

Factor Oracle

Algorithm

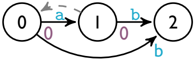
Algorithm 2 Incremental update of Factor Oracle

function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    while $k > -1$ and there is no transition from k by σ **do**
        Create a new transition from k to $m+1$ by σ ▷ $\delta(k, \sigma) = m + 1$
        $\pi_1 \leftarrow k$
        $k \leftarrow S_p(k)$
    end while
    …
end function

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 1 \quad k = 0 \qquad \pi_1 = 1$



slide 13

Factor Oracle

Algorithm

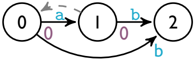
Algorithm 2 Incremental update of Factor Oracle

function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    while $k > -1$ and there is no transition from k by σ **do**
        Create a new transition from k to $m+1$ by σ $\qquad \triangleright\ \delta(k, \sigma) = m + 1$
        $\pi_1 \leftarrow k$
        $k \leftarrow S_p(k)$
    end while
    …
end function

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 1 \quad k = 0 \qquad \pi_1 = 0$



slide 14

Factor Oracle

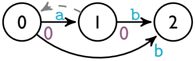
Algorithm

Algorithm 2 Incremental update of Factor Oracle

function AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
    …
    while $k > -1$ and there is no transition from k by σ **do**
        Create a new transition from k to $m + 1$ by σ ▷ $\delta(k, \sigma) = m + 1$
        $\pi_1 \leftarrow k$
        $k \leftarrow S_p(k)$
    end while
    …
end function

$p =$	a	b	b	c	a	b	c	d	a	b	c

$m = 1 \quad k = -1 \quad \pi_1 = 0$



slide 15

Factor Oracle

Algorithm

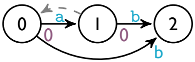
Algorithm 2 Incremental update of Factor Oracle

function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    if $k = -1$ **then**
        $S_{p\sigma} \leftarrow 0$
        $lrs_{p\sigma} \leftarrow 0$
    else
        $S_{p\sigma} \leftarrow$ state that leads the transition from k by σ
        $lrs_{p\sigma} \leftarrow$ LengthCommonSuffix($\pi_1, S(m+1) - 1$) + 1
    end if
    …
end function

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 1 \quad k = -1 \quad \pi_1 = 0$



Factor Oracle

Algorithm

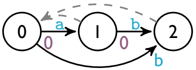
Algorithm 2 Incremental update of Factor Oracle

function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    if $k = -1$ **then**
        $S_{p\sigma} \leftarrow 0$
        $lrs_{p\sigma} \leftarrow 0$
    else
        $S_{p\sigma} \leftarrow$ state that leads the transition from k by σ
        $lrs_{p\sigma} \leftarrow$ LengthCommonSuffix($\pi_1, S(m+1) - 1$) + 1
    end if
    …
end function

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 1 \quad k = -1 \quad \pi_1 = 0$



slide 17

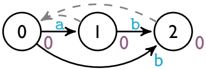
Algorithm 2 Incremental update of Factor Oracle

function AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
  …
  if $k = -1$ **then**
    $S_{p\sigma} \leftarrow 0$
    $lrs_{p\sigma} \leftarrow 0$
  else
    $S_{p\sigma} \leftarrow$ state that leads the transition from k by σ
    $lrs_{p\sigma} \leftarrow$ LengthCommonSuffix($\pi_1, S(m+1)-1$) + 1
  end if
  …
end function

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 1 \quad k = -1 \quad \pi_1 = 0$



Factor Oracle

Algorithm

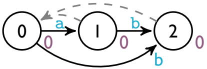
Algorithm 2 Incremental update of Factor Oracle

function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
  Create state $m + 1$
  Create a new transition from m to $m + 1$ labeled by σ $\quad \triangleright\ \delta(m, \sigma) = m + 1$
  $k \leftarrow S_p(m)$
  $\pi_1 \leftarrow m$
  …
end function

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 2$



slide 19

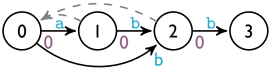
Algorithm 2 Incremental update of Factor Oracle

function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    Create state $m + 1$
    Create a new transition from m to $m + 1$ labeled by σ $\triangleright\ \delta(m, \sigma) = m + 1$
    $k \leftarrow S_p(m)$
    $\pi_1 \leftarrow m$
    …
end function

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 2$



slide 20

Factor Oracle

Algorithm

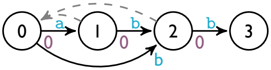
Algorithm 2 Incremental update of Factor Oracle

function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    Create state $m + 1$
    Create a new transition from m to $m + 1$ labeled by σ $\triangleright\ \delta(m, \sigma) = m + 1$
    $k \leftarrow S_p(m)$
    $\pi_1 \leftarrow m$
    …
end function

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 2 \quad k = 0 \qquad \pi_1 = 2$



slide 21

Factor Oracle

Algorithm

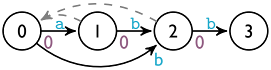
Algorithm 2 Incremental update of Factor Oracle

function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    while $k > -1$ and there is no transition from k by σ **do**
        Create a new transition from k to $m + 1$ by σ $\triangleright\ \delta(k, \sigma) = m + 1$
        $\pi_1 \leftarrow k$
        $k \leftarrow S_p(k)$
    end while
    …
end function

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 2 \quad k = 0 \qquad \pi_1 = 2$



slide 22

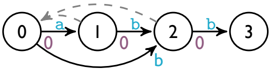
Algorithm 2 Incremental update of Factor Oracle

function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
  …
  if $k = -1$ **then**
    $S_{p\sigma} \leftarrow 0$
    $lrs_{p\sigma} \leftarrow 0$
  else
    $S_{p\sigma} \leftarrow$ state that leads the transition from k by σ
    $lrs_{p\sigma} \leftarrow$ LengthCommonSuffix($\pi_1, S(m+1) - 1$) + 1
  end if
  …
end function

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 2 \quad k = 0 \qquad \pi_1 = 2$



Factor Oracle

Algorithm

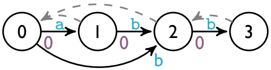
Algorithm 2 Incremental update of Factor Oracle

function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
  …
  if $k = -1$ **then**
    $S_{p\sigma} \leftarrow 0$
    $lrs_{p\sigma} \leftarrow 0$
  else
    $S_{p\sigma} \leftarrow$ state that leads the transition from k by σ
    $lrs_{p\sigma} \leftarrow$ LengthCommonSuffix($\pi_1, S(m+1) - 1$) + 1
  end if
  …
end function

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 2 \quad k = 0 \qquad \pi_1 = 2$



slide 24

Algorithm 2 Incremental update of Factor Oracle

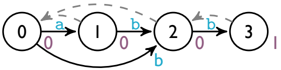
function AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
    …
    if $k = -1$ **then**
        $S_{p\sigma} \leftarrow 0$
        $lrs_{p\sigma} \leftarrow 0$
    else
        $S_{p\sigma} \leftarrow$ state that leads the transition from k by σ
        $lrs_{p\sigma} \leftarrow$ LengthCommonSuffix($\pi_1, S(m+1)-1$) + 1
    end if
    …
end function

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 2 \quad k = 0 \qquad \pi_1 = 2$

$lcs(2, 1) = 0$



Factor Oracle

Algorithm

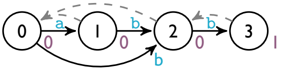
Algorithm 2 Incremental update of Factor Oracle

function AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
    Create state $m + 1$
    Create a new transition from m to $m + 1$ labeled by σ $\triangleright\ \delta(m, \sigma) = m + 1$
    $k \leftarrow S_p(m)$
    $\pi_1 \leftarrow m$
    …
end function

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 3$



slide 26

Factor Oracle

Algorithm

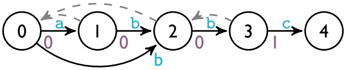
Algorithm 2 Incremental update of Factor Oracle

function AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
    Create state $m + 1$
    Create a new transition from m to $m + 1$ labeled by σ $\triangleright\ \delta(m, \sigma) = m + 1$
    $k \leftarrow S_p(m)$
    $\pi_1 \leftarrow m$
    …
end function

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 3$



slide 27

Factor Oracle

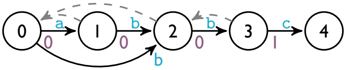
Algorithm

Algorithm 2 Incremental update of Factor Oracle

function AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
    Create state $m + 1$
    Create a new transition from m to $m + 1$ labeled by σ $\triangleright\ \delta(m, \sigma) = m + 1$
    $k \leftarrow S_p(m)$
    $\pi_1 \leftarrow m$
    …
end function

$p =$	a	b	b	c	a	b	c	d	a	b	c

$m = 3 \quad k = 2 \qquad \pi_1 = 3$



slide 28

Factor Oracle

Algorithm

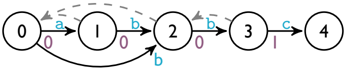
Algorithm 2 Incremental update of Factor Oracle

function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    while $k > -1$ and there is no transition from k by σ **do**
        Create a new transition from k to $m + 1$ by σ ▷ $\delta(k, \sigma) = m + 1$
        $\pi_1 \leftarrow k$
        $k \leftarrow S_p(k)$
    end while
    …
end function

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 3$ $k = 2$ $\pi_1 = 3$



slide 29

Factor Oracle

Algorithm

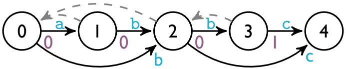
Algorithm 2 Incremental update of Factor Oracle

function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
  …
  while $k > -1$ and there is no transition from k by σ **do**
    Create a new transition from k to $m+1$ by σ ▷ $\delta(k, \sigma) = m + 1$
    $\pi_1 \leftarrow k$
    $k \leftarrow S_p(k)$
  end while
  …
end function

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 3 \quad k = 2 \qquad \pi_1 = 3$



slide 30

Factor Oracle

Algorithm

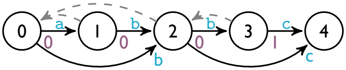
Algorithm 2 Incremental update of Factor Oracle

```
function AddLetter(Oracle(p = p_1, p_2 ... p_m), σ)
    ...
    while k > −1 and there is no transition from k by σ do
        Create a new transition from k to m + 1 by σ        ▷ δ(k, σ) = m + 1
        π_1 ← k
        k ← S_p(k)
    end while
    ...
end function
```

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 3 \quad k = 2 \qquad \pi_1 = 2$



slide 31

Factor Oracle

Algorithm

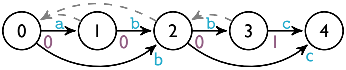
Algorithm 2 Incremental update of Factor Oracle

function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
  ...
  while $k > -1$ and there is no transition from k by σ **do**
    Create a new transition from k to $m + 1$ by σ ▷ $\delta(k, \sigma) = m + 1$
    $\pi_1 \leftarrow k$
    $k \leftarrow S_p(k)$
  end while
  ...
end function

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 3 \quad k = 0 \quad \pi_1 = 2$



slide 32

Factor Oracle

Algorithm

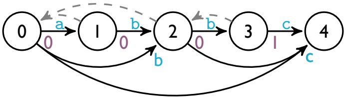
Algorithm 2 Incremental update of Factor Oracle

function AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
  …
  while $k > -1$ and there is no transition from k by σ **do**
    Create a new transition from k to $m+1$ by σ ▷ $\delta(k, \sigma) = m + 1$
    $\pi_1 \leftarrow k$
    $k \leftarrow S_p(k)$
  end while
  …
end function

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 3 \quad k = 0 \qquad \pi_1 = 2$



slide 33

Factor Oracle

Algorithm

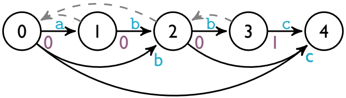
Algorithm 2 Incremental update of Factor Oracle

function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    while $k > -1$ and there is no transition from k by σ **do**
        Create a new transition from k to $m + 1$ by σ $\triangleright\ \delta(k, \sigma) = m + 1$
        $\pi_1 \leftarrow k$
        $k \leftarrow S_p(k)$
    end while
    …
end function

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 3 \quad k = 0 \qquad \pi_1 = 0$



slide 34

Factor Oracle

Algorithm

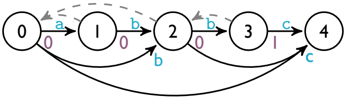
Algorithm 2 Incremental update of Factor Oracle

function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    …
    while $k > -1$ and there is no transition from k by σ **do**
        Create a new transition from k to $m + 1$ by σ $\triangleright\ \delta(k, \sigma) = m + 1$
        $\pi_1 \leftarrow k$
        $k \leftarrow S_p(k)$
    end while
    …
end function

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 3 \quad k = -1 \quad \pi_1 = 0$



slide 35

Factor Oracle

Algorithm

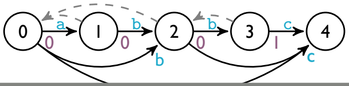
Algorithm 2 Incremental update of Factor Oracle

function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
    ...
    if $k = -1$ **then**
        $S_{p\sigma} \leftarrow 0$
        $lrs_{p\sigma} \leftarrow 0$
    else
        $S_{p\sigma} \leftarrow$ state that leads the transition from k by σ
        $lrs_{p\sigma} \leftarrow$ LengthCommonSuffix($\pi_1, S(m+1) - 1$) + 1
    end if
    ...
end function

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 3 \quad k = -1 \quad \pi_1 = 0$



Factor Oracle

Algorithm

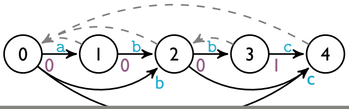
Algorithm 2 Incremental update of Factor Oracle

function AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
    …
    if $k = -1$ **then**
        $S_{p\sigma} \leftarrow 0$
        $lrs_{p\sigma} \leftarrow 0$
    else
        $S_{p\sigma} \leftarrow$ state that leads the transition from k by σ
        $lrs_{p\sigma} \leftarrow$ LengthCommonSuffix($\pi_1, S(m+1) - 1$) + 1
    end if
    …
end function

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 3 \quad k = -1 \quad \pi_1 = 0$



slide 37

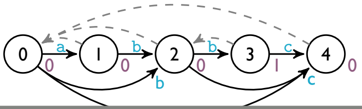
Algorithm 2 Incremental update of Factor Oracle

function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
  ...
  if $k = -1$ **then**
    $S_{p\sigma} \leftarrow 0$
    $lrs_{p\sigma} \leftarrow 0$
  else
    $S_{p\sigma} \leftarrow$ state that leads the transition from k by σ
    $lrs_{p\sigma} \leftarrow$ LengthCommonSuffix($\pi_1, S(m+1) - 1$) + 1
  end if
  ...
end function

$p =$

a	b	b	c	a	b	c	d	a	b	c

$m = 3 \quad k = -1 \quad \pi_1 = 0$



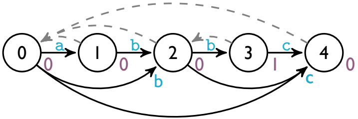
Factor Oracle

Algorithm

Algorithm 2 Incremental update of Factor Oracle

function AddLetter($Oracle(p = p_1, p_2 \dots p_m), \sigma$)
end function

$p =$	a	b	b	c	a	b	c	d	a	b	c



slide 39

Factor Oracle

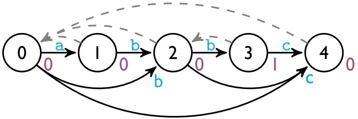
Algorithm

Algorithm 2 Incremental update of Factor Oracle

function AddLetter($Oracle(p = p_1, p_2 \ldots p_m), \sigma$)
end function

$p =$

a	b	b	c	a	b	c	d	a	b	c



slide 40

Factor Oracle

Algorithm

Algorithm 3 Find Better Algorithm

```
function FindBetter(i, a)
    for all the elements j of T(i) in increasing order do
        if lrs(j) = lrs(i) and p[j − lrs(i)] = a then
            return j
        end if
    end for
    return 0
end function
```

Algorithm 3 Find Better Algorithm

```
function FindBetter(i, a)
    for all the elements j of T(S(i)) in increasing order do
        if lrs(j) = lrs(i) and p[j − lrs(i)] = a then
            return j
        end if
    end for
    return 0
end function
```

Factor Oracle

Algorithm

Algorithm 4 Length Common Suffix Algorithm

function LengthCommonSuffix(π_1, π_2)
    if $S(\pi_1) = \pi_2$ **then**
        return $lrs(\pi_1)$
    else
        while $S(\pi_1) \neq S(\pi_2)$ **do**
            $\pi_2 \leftarrow S(\pi_2)$
        end while
    end if
    return $min(lrs(\pi_1), lrs(\pi_2))$
end function

Thank you for your attention! ☺

Factor Oracle for Machine Improvisation

Jaime Arias

Université de Bordeaux, LaBRI, UMR 5800
Inria - Bordeaux Sud-Ouest

August 2016