Fully-Compressed Suffix Trees

Luís M. S. Russo  Gonzalo Navarro  Arlindo L. Oliveira

INESC-ID/IST
{lrs,aml}@algos.inesc-id.pt

Dept. of Computer Science, University of Chile
gnavarro@dcc.uchile.cl

Workshop on Compression, Text, and Algorithms 2007
Outline

1 Motivation
   - The Problem We Studied
   - Previous Work

2 Our Contribution
   - Performance
   - The kernel Operations
   - Further Operations

3 Conclusions
   - Summary
Suffix trees are important for several string problems:
- pattern matching
- longest common substring
- super maximal repeats
- bioinformatics applications
- etc
Example (Suffix Tree for *abbbab*)

Luís M. S. Russo, Gonzalo Navarro, Arlindo L. Oliveira

Fully-Compressed Suffix Trees
Problem (Suffix Trees need too much space)

*Pointer based representations require $O(n \log n)$ bits.*

This is much larger than the indexed string.
State of the art implementations require $[8, 10]n \log \sigma$ bits.
Sadakane proposed a new way to represent suffix trees.

Compressed Suffix Tree

- Tree Structure
- Compressed Index

+ Balanced parentheses representation

- Nodes represented as intervals
A node represented as an interval of leaves of a suffix tree.

**Example**

Interval $[3, 6]$ represents node $b$. 
Compressed indexes are compressed representations of the leaves of a suffix tree. Their success relies on:

- **Succinct structures**, based on RANK and SELECT.
- **Data compression**, that represent $T$ in $O(uH_k)$ bits.

### Examples

FM-index, Compressed Suffix Arrays, LZ-index, etc.

Sadakane used compressed suffix arrays. We need a compressed index that supports $\psi$ and LF. For example the Alphabet-Friendly FM-Index.
\[ \sigma = O(\text{polylog}(n)) \]

<table>
<thead>
<tr>
<th></th>
<th>Sadakane’s</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Space in bits</td>
<td>(nH_k + 6n + o(n \log \sigma))</td>
<td>(nH_k + o(n \log \sigma))</td>
</tr>
<tr>
<td>SDep/Locate</td>
<td>((\log_\sigma \log n) \log n)</td>
<td>((\log_\sigma \log n) \log n)</td>
</tr>
<tr>
<td>Count/Ancestor</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Parent/FirstChild</td>
<td>1</td>
<td>((\log_\sigma \log n) \log n)</td>
</tr>
<tr>
<td>SLink</td>
<td>1</td>
<td>((\log_\sigma \log n) \log n)</td>
</tr>
<tr>
<td>SLink'</td>
<td>((\log_\sigma \log n) \log n)</td>
<td>((\log_\sigma \log n) \log n)</td>
</tr>
<tr>
<td>Letter((v, i))</td>
<td>((\log_\sigma \log n) \log n)</td>
<td>((\log_\sigma \log n) \log n)</td>
</tr>
<tr>
<td>LCA</td>
<td>1</td>
<td>((\log_\sigma \log n) \log n)</td>
</tr>
<tr>
<td>Child</td>
<td>((\log \log n) \log n)</td>
<td>((\log \log n)^2 \log_\sigma n)</td>
</tr>
<tr>
<td>TDep</td>
<td>1</td>
<td>(((\log_\sigma \log n) \log n)^2)</td>
</tr>
<tr>
<td>LAQT</td>
<td>1</td>
<td>(((\log_\sigma \log n) \log n)^2)</td>
</tr>
<tr>
<td>LAQS</td>
<td>—</td>
<td>((\log_\sigma \log n) \log n)</td>
</tr>
<tr>
<td>WeinerLink</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
**Overall Performance**

\[ \sigma = O(\text{polylog}(n)) \]

<table>
<thead>
<tr>
<th>Space in bits</th>
<th>Sadakane’s</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>( nH_k + 6n + o(n \log \sigma) )</td>
<td>( nH_k + o(n \log \sigma) )</td>
<td></td>
</tr>
<tr>
<td>SD\text{EP}/LOCATE</td>
<td>( (\log_\sigma \log n) \log n )</td>
<td>( (\log_\sigma \log n) \log n )</td>
</tr>
<tr>
<td>COUNT/ANCESTOR</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>PARENT/F\text{CHILD}/</td>
<td>1</td>
<td>( (\log_\sigma \log n) \log n )</td>
</tr>
<tr>
<td>SL\text{INK}</td>
<td>1</td>
<td>( (\log_\sigma \log n) \log n )</td>
</tr>
<tr>
<td>SL\text{INK}'</td>
<td>( (\log_\sigma \log n) \log n )</td>
<td>( (\log_\sigma \log n) \log n )</td>
</tr>
<tr>
<td>LETTER((v, i))</td>
<td>( (\log_\sigma \log n) \log n )</td>
<td>( (\log_\sigma \log n) \log n )</td>
</tr>
<tr>
<td>LCA</td>
<td>1</td>
<td>( (\log_\sigma \log n) \log n )</td>
</tr>
<tr>
<td>\text{CHILD}</td>
<td>( (\log \log n) \log n )</td>
<td>( (\log \log n)^2 \log_\sigma n )</td>
</tr>
<tr>
<td>T\text{DP}</td>
<td>1</td>
<td>( ((\log_\sigma \log n) \log n)^2 )</td>
</tr>
<tr>
<td>LAQ\text{T}</td>
<td>1</td>
<td>( ((\log_\sigma \log n) \log n)^2 )</td>
</tr>
<tr>
<td>LAQ\text{S}</td>
<td>—</td>
<td>( (\log_\sigma \log n) \log n )</td>
</tr>
<tr>
<td>WE\text{INERLINK}</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
\[ \sigma = O(\text{polylog}(n)) \]

<table>
<thead>
<tr>
<th></th>
<th>Sadakane's</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Space in bits</td>
<td>( nH_k + 6n + o(n \log \sigma) )</td>
<td>( nH_k + o(n \log \sigma) )</td>
</tr>
<tr>
<td>SDep/Locate</td>
<td>((\log_\sigma \log n) \log n)</td>
<td>((\log_\sigma \log n) \log n)</td>
</tr>
<tr>
<td>Count/Ancestor</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Parent/FirstChild/</td>
<td>1</td>
<td>((\log_\sigma \log n) \log n)</td>
</tr>
<tr>
<td>SLink</td>
<td>1</td>
<td>((\log_\sigma \log n) \log n)</td>
</tr>
<tr>
<td>SLink'</td>
<td>((\log_\sigma \log n) \log n)</td>
<td>((\log_\sigma \log n) \log n)</td>
</tr>
<tr>
<td>Letter((v, i))</td>
<td>((\log_\sigma \log n) \log n)</td>
<td>((\log_\sigma \log n) \log n)</td>
</tr>
<tr>
<td>LCA</td>
<td>1</td>
<td>((\log_\sigma \log n) \log n)</td>
</tr>
<tr>
<td>Child</td>
<td>((\log \log n) \log n)</td>
<td>((\log \log n)^2 \log_\sigma n)</td>
</tr>
<tr>
<td>TDep</td>
<td>1</td>
<td>(((\log_\sigma \log n) \log n)^2)</td>
</tr>
<tr>
<td>LAQT</td>
<td>1</td>
<td>(((\log_\sigma \log n) \log n)^2)</td>
</tr>
<tr>
<td>LAQS</td>
<td>–</td>
<td>((\log_\sigma \log n) \log n)</td>
</tr>
<tr>
<td>WeinerLink</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
\[ \sigma = O(\text{polylog}(n)) \]

<table>
<thead>
<tr>
<th></th>
<th>Sadakane’s</th>
<th>Ours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Space in bits</td>
<td>( nH_k + 6n + o(n \log \sigma) )</td>
<td>( nH_k + o(n \log \sigma) )</td>
</tr>
<tr>
<td>SDep/Locate</td>
<td>((\log_\sigma \log n) \log n)</td>
<td>((\log_\sigma \log n) \log n)</td>
</tr>
<tr>
<td>Count/Ancestor</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Parent/FirstChild/</td>
<td>1</td>
<td>((\log_\sigma \log n) \log n)</td>
</tr>
<tr>
<td>SLink</td>
<td>1</td>
<td>((\log_\sigma \log n) \log n)</td>
</tr>
<tr>
<td>SLink'</td>
<td>((\log_\sigma \log n) \log n)</td>
<td>((\log_\sigma \log n) \log n)</td>
</tr>
<tr>
<td>Letter((v,i))</td>
<td>((\log_\sigma \log n) \log n)</td>
<td>((\log_\sigma \log n) \log n)</td>
</tr>
<tr>
<td>LCA</td>
<td>1</td>
<td>((\log_\sigma \log n) \log n)</td>
</tr>
<tr>
<td>Child</td>
<td>((\log \log n) \log n)</td>
<td>((\log \log n)^2 \log_\sigma n)</td>
</tr>
<tr>
<td>TDep</td>
<td>1</td>
<td>(((\log_\sigma \log n) \log n)^2)</td>
</tr>
<tr>
<td>LAQT</td>
<td>1</td>
<td>(((\log_\sigma \log n) \log n)^2)</td>
</tr>
<tr>
<td>LAQS</td>
<td>—</td>
<td>((\log_\sigma \log n) \log n)</td>
</tr>
<tr>
<td>WeinerLink</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>
We use sampling instead of balanced parentheses.
We use sampling instead of balanced parentheses.

Compressed Suffix Tree

- Tree Structure
- Compressed Index

+ Sampling

- Nodes represented as intervals

- LSA
The sampling has the property that in any sequence

- \( v \)
- \( \text{SLINK}(v) \)
- \( \text{SLINK}(	ext{SLINK}(v)) \)
- \( \text{SLINK}(	ext{SLINK}(	ext{SLINK}(v))) \)
- \( \ldots \)

of size \( \delta \) there is at least one sampled node.
Lemma

*When* $\text{LCA}(v, v') \neq \text{ROOT} *we have that:*

$$\text{SLINK}(\text{LCA}(v, v')) = \text{LCA}(\text{SLINK}(v), \text{SLINK}(v'))$$
Lemma

If \( \text{SLINK}^r(\text{LCA}(v, v')) = \text{ROOT} \), and let \( d = \min(\delta, r + 1) \).

Then \( \text{SDEP}(\text{LCA}(v, v')) = \max_{0 \leq i < d} \{ i + \text{SDEP}(\text{LCSA}(\text{SLINK}^i(v), \text{SLINK}^i(v'))) \} \)

Proof.

\[
\begin{align*}
\text{SDEP}(\text{LCA}(v, v')) & = i + \text{SDEP}(\text{SLINK}^i(\text{LCA}(v, v'))) \\
& = i + \text{SDEP}(\text{LCA}(\text{SLINK}^i(v), \text{SLINK}^i(v'))) \\
& \geq i + \text{SDEP}(\text{LCSA}(\text{SLINK}^i(v), \text{SLINK}^i(v'))) \\
\end{align*}
\]

The last inequality is an equality for some \( i \leq d \).
Lemma

If $\text{SLINK}^i(LCA(v, v')) = \text{ROOT}$, and let $d = \min(\delta, r + 1)$. Then $\text{SDEP}(LCA(v, v')) = \max_{0 \leq i < d} \{i + \text{SDEP}(\text{LCSA} (\text{SLINK}^i(v), \text{SLINK}^i(v'))))\}$

Proof.

$\text{SDEP}(LCA(v, v'))$

$= i + \text{SDEP}(\text{SLINK}^i(LCA(v, v'))))$

$= i + \text{SDEP}(LCA(\text{SLINK}^i(v), \text{SLINK}^i(v')))\)$

$\geq i + \text{SDEP}(\text{LCSA} (\text{SLINK}^i(v), \text{SLINK}^i(v'))))$

The last inequality is an equality for some $i \leq d$. 
Lemma

If \( \text{SLINK}^r(\text{LCA}(v, v')) = \text{ROOT} \), and let \( d = \min(\delta, r + 1) \).
Then \( \text{SDEP}(\text{LCA}(v, v')) \geq \max_{0 \leq i < d}\{i + \text{SDEP}(\text{LCSA}(\text{SLINK}^i(v), \text{SLINK}^i(v'))))\}\)

Proof.

\[
\text{SDEP}(\text{LCA}(v, v'))
= i + \text{SDEP}(\text{SLINK}^i(\text{LCA}(v, v')))) \\
= i + \text{SDEP}(\text{LCA}(\text{SLINK}^i(v), \text{SLINK}^i(v')))) \\
\geq i + \text{SDEP}(\text{LCSA}(\text{SLINK}^i(v), \text{SLINK}^i(v'))))
\]
The last inequality is an equality for some \( i \leq d \).
Lemma

If $\text{SLINK}^i(LCA(v, v')) = \text{ROOT}$, and let $d = \min(\delta, r + 1)$. Then $S\text{DEP}(LCA(v, v'))$?

$$\max_{0 \leq i < d} \{ i + S\text{DEP}(LCSA(\text{SLINK}^i(v), \text{SLINK}^i(v'))) \}$$

Proof.

$$S\text{DEP}(LCA(v, v'))$$

$$= i + S\text{DEP}(\text{SLINK}^i(LCA(v, v')))$$

$$= i + S\text{DEP}(LCA(\text{SLINK}^i(v), \text{SLINK}^i(v')))$$

$$\geq i + S\text{DEP}(LCSA(\text{SLINK}^i(v), \text{SLINK}^i(v')))$$

The last inequality is an equality for some $i \leq d$. 

Luís M. S. Russo, Gonzalo Navarro, Arlindo L. Oliveira

Fully-Compressed Suffix Trees
Lemma

If \( \text{SLINK}^r(LCA(v, v')) = \text{ROOT} \), and let \( d = \min(\delta, r + 1) \).

Then \( \text{SDEP}(LCA(v, v')) \geq \max_{0 \leq i < d} \{i + \text{SDEP}(\text{LCSA}(\text{SLINK}^i(v), \text{SLINK}^i(v'))))\} \)

Proof.

\[
\begin{align*}
\text{SDEP}(LCA(v, v')) & = i + \text{SDEP}(\text{SLINK}^i(LCA(v, v'))) \\
& = i + \text{SDEP}(LCA(\text{SLINK}^i(v), \text{SLINK}^i(v'))) \\
& \geq i + \text{SDEP}(\text{LCSA}(\text{SLINK}^i(v), \text{SLINK}^i(v'))) \\
\end{align*}
\]

The last inequality is an equality for some \( i \leq d \).
Lemma

\[ \text{If } \text{SLINK}^f(LCA(v, v')) = \text{ROOT, and let } d = \min(\delta, r + 1). \]
\[ \text{Then } \text{SDEP}(LCA(v, v')) = \max_{0 \leq i < d}\{i + \text{SDEP}(\text{LCSA}(\text{SLINK}^i(v), \text{SLINK}^i(v'))))\}\]

Proof.

\[ \text{SDEP}(LCA(v, v')) = i + \text{SDEP}(\text{SLINK}^i(LCA(v, v'))) \]
\[ = i + \text{SDEP}(\text{LCA}(\text{SLINK}^i(v), \text{SLINK}^i(v'))) \]
\[ \geq i + \text{SDEP}(\text{LCSA}(\text{SLINK}^i(v), \text{SLINK}^i(v'))) \]

The last inequality is an equality for some \(i \leq d\).
Fundamental lemma

Example ($\delta = 3$)

- sampled
- not sampled
- active
Fundamental lemma

Example \((\delta = 3)\)
Motivation

Our Contribution

Conclusions

Performance

KOps

+Ops

Fundamental lemma

Example ($\delta = 3$)

SDep : 5

Luís M. S. Russo, Gonzalo Navarro, Arlindo L. Oliveira

Fully-Compressed Suffix Trees
Example ($\delta = 3$)

SDep : 5
Example ($\delta = 3$)

\[
\text{SDep : 5}
\]

\[
\text{10}
\]
Fundamental lemma

Example \((\delta = 3)\)

SDEP : 5

10
Fundamental lemma

Example ($\delta = 3$)

SDep: 5  10  7
Fundamental lemma

Example ($\delta = 3$)

5+0
10+1
7+2
Fundamental lemma

Example ($\delta = 3$)

5+0 10+1 7+2
Why is the lemma important?

Tree Structure + Compressed Index

Sampling

Nodes represented as intervals

LCA S Dep
LSA
SLINK
S Dep
The lemma allows us to compute other operations:

- \( \text{SDep}(v) = \text{SDep}(\text{LCA}(v, v)) \).
- \( \text{SLink}(v) = \text{LCA}(\psi(v_l), \psi(v_r)) \),
  \( \text{SLink}^i(v) = \text{LCA}(\psi^i(v_l), \psi^i(v_r)) \).
- \( \text{LCA}(v, v') = \)
  \( \text{LF}(v[0..i - 1], \text{LCSA}(\text{SLink}^i(v), \text{SLink}^i(v'))) \),
  for the \( i \) in the lemma.

\( \text{SLink} \) depends on \( \text{LCA} \) and \( \text{LCA} \) on \( \text{SLink} \).
The lemma allows us to compute other operations:

- \( \text{SDEP}(v) = \text{SDEP}(\text{LCA}(v, v)) \). 
- \( \text{SLINK}(v) = \text{LCA}(\psi(v_l), \psi(v_r)) \),
  \( \text{SLINK}^i(v) = \text{LCA}(\psi^i(v_l), \psi^i(v_r)) \).
- \( \text{LCA}(v, v') = 
  \text{LF}(v[0..i-1], 
  \text{LCSA} (\text{SLINK}^i(v), \text{SLINK}^i(v'))) \),
  \text{for the } i \text{ in the lemma.} 

\text{SLINK} \text{ depends on LCA and LCA on SLINK.}
The lemma allows us to compute other operations:

- \( \text{SDEP}(v) = \text{SDEP}(\text{LCA}(v, v)) \).
- \( \text{SLINK}(v) = \text{LCA}(\psi(v_l), \psi(v_r)) \),
  \( \text{SLINK}^i(v) = \text{LCA}(\psi^i(v_l), \psi^i(v_r)) \).
- \( \text{LCA}(v, v') = \text{LF}(v[0..i-1], \text{LCSA}(\text{SLINK}^i(v), \text{SLINK}^i(v'))) \),
  for the \( i \) in the lemma.

\( \text{SLINK} \) depends on \( \text{LCA} \) and \( \text{LCA} \) on \( \text{SLINK} \).
The lemma allows us to compute other operations:

- \( \text{SDEP}(v) = \text{SDEP}(\text{LCA}(v, v)) \).
- \( \text{SLINK}(v) = \text{LCA}(\psi(v_l), \psi(v_r)) \),
  \( \text{SLINK}^i(v) = \text{LCA}(\psi^i(v_l), \psi^i(v_r)) \).
- \( \text{LCA}(v, v') = \)
  \( \text{LF}(v[0..i-1], \text{LCSA}(\text{SLINK}^i(v), \text{SLINK}^i(v'))) \),
  for the \( i \) in the lemma.

\( \text{SLINK} \) depends on \( \text{LCA} \) and \( \text{LCA} \) on \( \text{SLINK} \).
Example ($\delta = 3$)

- $5+0$
- $10+1$
- $7+2$
Entangled Operations

Example ($\delta = 3$)

Luís M. S. Russo, Gonzalo Navarro, Arlindo L. Oliveira

Fully-Compressed Suffix Trees
The lemma allows us to compute other operations:

- $\text{SDEP}(v) = \text{SDEP}(\text{LCA}(v, v))$.
- $\text{SLINK}(v) = \text{LCA}(\psi(v_l), \psi(v_r))$, $\text{SLINK}^i(v) = \text{LCA}(\psi^i(v_l), \psi^i(v_r))$.
- $\text{LCA}(v, v') = \text{LF}(v[0..i-1], \text{LCSA}(\text{SLINK}^i(v), \text{SLINK}^i(v')))$. for the $i$ in the lemma.

$\text{SLINK}$ depends on $\text{LCA}$ and $\text{LCA}$ on $\text{SLINK}$.
The lemma allows us to compute other operations:

- **SDep**($v$) = **SDep**($\text{LCA}(v, v)$).
- **SLink**($v$) = **LCA**($\psi(v_l), \psi(v_r)$),
  **SLink**$^i$(v) = **LCA**($\psi^i(v_l), \psi^i(v_r)$).
- **LCA**($v, v'$) =
  \[ \text{LF}(v[0..i-1], \text{LCSA}(\text{SLink}^i(v), \text{SLink}^i(v'))), \]
  for the $i$ in the lemma.

**SLink** depends on **LCA** and **LCA** on **SLINK**.
To avoid this circular dependency we use the next lemma.

**Lemma**

\[ \text{LCA}(v, v') = \text{LCA}(\min\{v_l, v'_l\}, \max\{v_r, v'_r\}) \]

**Example**

```
  α
 /\  \\
/   \  \\
Y     Z
/ \   / \  \\
 v   v'  
```

Luís M. S. Russo, Gonzalo Navarro, Arlindo L. Oliveira

Fully-Compressed Suffix Trees
To avoid this circular dependency we use the next lemma.

**Lemma**

\[
\text{LCA}(v, v') = \text{LCA}(\min\{v_l, v'_l\}, \max\{v_r, v'_r\})
\]
To avoid this circular dependency we use the next lemma.

**Lemma**

\[ \text{LCA}(v, v') = \text{LCA}(\min\{v_l, v'_l\}, \max\{v_r, v'_r\}) \]

**Example**

![Diagram showing the relationship between \(v\), \(v'\), \(min\{v_l, v'_l\}\), and \(max\{v_r, v'_r\}\).]
Hence we can use $\psi$ instead of $\text{SLINK}$. Therefore LCA no longer depends on $\text{SLINK}$.

The following operations simplify:

- $\text{SDEP}(v) = \text{SDEP}(\text{LCA}(v, v)) = \max_{0 \leq i < d}\{i + \text{SDEP}(\text{LCSA}(\psi^i(v_l), \psi^i(v_r)))\}$.
- $\text{LCA}(v, v') = \text{LF}(v[0..i - 1], \text{LCSA}(\psi^i(\min\{v_l, v'_l\}), \psi^i(\max\{v_r, v'_r\})))$, for the $i$ in the lemma.
Hence we can use $\psi$ instead of $\text{SLINK}$. Therefore LCA no longer depends on $\text{SLINK}$. The following operations simplify:

- $\text{SDEP}(v) = \text{SDEP}(\text{LCA}(v, v)) = \max_{0 \leq i < d} \{ i + \text{SDEP}(\text{LCSA}(\psi^i(v_l), \psi^i(v_r))) \}$.

- $\text{LCA}(v, v') =$
  
  $\text{LF}(v[0..i-1], \text{LCSA}(\psi^i(\min\{v_l, v'_l\}), \psi^i(\max\{v_r, v'_r\})))$, for the $i$ in the lemma.
Hence we can use $\psi$ instead of SLINK. Therefore LCA no longer depends on SLINK. The following operations simplify:

- $SDep(v) = Dep(LCA(v, v)) = \max_{0 \leq i < d}\{i + Dep(LCSA(\psi^i(v_l), \psi^i(v_r)))\}.
- LCA(v, v') =
  LF(v[0..i - 1],
  LCSA(\psi^i(\min\{v_l, v'_l\}), \psi^i(\max\{v_r, v'_r\})))
  for the $i$ in the lemma.
With these base operations we can also compute:

- \( \text{LETTER}(v, i) = \text{SLINK}^i(v)[0] = \psi^i(v_i)[0] \)
- \( \text{PARENT} \) is either
  \( \text{LCA}(v_i - 1, v_i) \) or
  \( \text{LCA}(v_r, v_r + 1) \), whichever is lowest.
With these base operations we can also compute:

- \( \text{LETTER}(v, i) = \text{SLINK}^i(v)[0] = \psi^i(v_i)[0] \)
- \( \text{PARENT} \) is either
  \( \text{LCA}(v_l - 1, v_l) \) or
  \( \text{LCA}(v_r, v_r + 1) \), whichever is lowest.
**Further Operations**

- **CHILD** can be computed with **LETTER** and binary searches.

- We can also use the fundamental lemma as
  \[
  \text{CHILD}(v, X) = \text{LF}(v[0..i - 1], \text{CHILD}(\text{SLINK}^i(v), X))
  \]

- The branching is computed over child lists in the sampled tree.

- We proposed a compromise between these approaches.
Further Operations

- **CHILD** can be computed with **LETTER** and binary searches.
- We can also use the fundamental lemma as
  \[ \text{CHILD}(v, X) = \text{LF}(v[0..i-1], \text{CHILD}(	ext{SLINK}^i(v), X)) \]
- The branching is computed over child lists in the sampled tree.
- We proposed a compromise between these approaches.
Further Operations

- CHILD can be computed with LETTER and binary searches.
- We can also use the fundamental lemma as
  \[ \text{CHILD}(v, X) = \text{LF}(v[0..i-1], \text{CHILD}(\text{SLINK}^i(v), X)) \]
- The branching is computed over child lists in the sampled tree.
- We proposed a compromise between these approaches.
We presented a representation of suffix tree that:

- occupies $uH_k + o(u \log \sigma)$ bits.
- supports usual operations in a reasonable time.
Acknowledgments

- FCT, grant SFRH/BD/12101/2003 of project POCI 2010.
- Fondecyt Grant 1-050493 (Chile).
Thanks for listening.