The Cocke-Younger-Kasami Algorithm
- Revised -

Ulrike Brandt and Hermann K.-G. Walter

University of Technology, Darmstadt
Department of Computer Science
[brandt,walter]@iti.informatik.tu-darmstadt.de
Fax:+49(0)6151-16-6185

Abstract

The wellknown algorithm of Cocke-Younger-Kasami, solving the
wordproblem for contextfree grammars in Chomsky-Normalform in
time $O(|w|^2)$ with the help of the recognition matrix can be extended
to arbitrary contextfree grammars. The resulting time bound is
$O(|w|^2+||G||^{-1})$ where $||G||$ is a very natural number associated to
$G$. Moreover for linear grammars we get time $O(|w|^2)$, the bound from
Earley's algorithm, and with small variations $O(|w|)$ for one-sided-
linear grammars.

Keywords: wordproblem, contextfree grammars, recognition ma-
trix, time-complexity

Introduction

The starting point for this note is the simple observation

$$w \in L(G) \iff \{w\} \cap L(G) \neq \emptyset$$

($G$ a (contextfree) grammar and $L(G)$ the generated language).

This is the reduction from the wordproblem to the emptiness-problem.

Since $\{w\}$ is a regular set with a very special minimal Rabin-Scott-acceptor,
one can use the wellknown triple construction for the intersection-theorem.

Rewriting this triple construction into the recognition-matrix, we can avoid
under special circumstances both the construction via the intersection-theo-
rem and the design of a good algorithm for the emptiness problem.
Furthermore we can avoid the transformation of the original grammar into 
some normal form, especially we need not prepare for erasing and chaining. 
A reasonable time bound results, giving the time bound of Cocke-Younger-
Kasami in the case of Chomsky-Normal form.

Moreover, for linear grammars the quadratic time bound of Earley’s algo-

rithm results. With a small variation of the basic algorithm we get for 
one-sided linear grammars (regular grammars) a linear time bound, as it 
should be.

Besides the knowledge, that by different approach we get the time bound 
\( O(|w|^3) \), our result may be of didactic value due to the simplicity of the 
argument.

1 Notations

If \( X \) is an alphabet, then \( X^* \) is the free monoid with the empty word \( \Box \).
Consider \( X' \subseteq X \). Let \( w \in X^* \), then \( w \) has a unique decomposition

\[
w = w_0x'_1w_1 \cdots x'_r w_r \text{ with } \quad x'_i \in X'(1 \leq i \leq r) \text{ and } w_i \in (X \setminus X')^*(0 \leq i \leq r).
\]

We call it the \( X' \) -decomposition.

Denote by \( |w|_{X'} = r \) the length with respect to \( X' \). Obviously, \( |w|_X = |w| \)
is the length of \( w \).

If \( w = x_1 \ldots x_n \) with \( x_m \in X \) for \( 1 \leq m \leq n \) and \( 0 \leq i \leq j \leq n \) denote by 
\( w[i,j] \) the word

\[
w[i,j] = x_i+1 \ldots x_n \text{ if } 0 < i < j \text{ and } w[i,i] = \Box.
\]

For \( j < i \) \( w[i,j] \) is undefined.

Note: \( w[i - 1, i] = x_i \) \( \quad 1 \leq i \leq n \).

A grammar \( G \) is a quadruple \( G = (\sigma, Z, T, P) \), where

- \( Z \) is the alphabet of variables
- \( T \) is the alphabet of terminals
- \( \sigma \in Z \) is the start symbol
- \( P \subseteq Z \times (Z \cup T)^* \) is the (finite) set of productions.

A rule \( r \in P \) is usually written in the form \( r = (p \rightarrow q) \).

By \( u \vdash v \) we denote the direct-derivation from \( u \) to \( v \), \( u \vdash v \) is the transitive 
and reflexive closure of \( \vdash \).
The generated language of $G$ is therefore defined by

$$L(G) = \{ w \in T^* \mid \sigma \vdash^* w \}.$$  

In this paper we are interested only in contextfree grammars and contextfree languages, just defined by contextfree grammars, i.e. for all $p \rightarrow q \in P \quad p \in Z$ holds.

More details on grammars and languages can be found in standard textbooks like [1] & [2] for example.

We introduce a measure for grammars $G$ by

$$||G|| = \text{Max}\{ |q|_2 \mid \exists \quad p \rightarrow q \in P \}.$$  

For example, a linear grammar $G$ is a contextfree grammar with $||G|| \leq 1$.

2 Preparations

The basic idea is the following connection between languages $L \subseteq X^*$ and $w \in X^*$:

$$w \in L \iff L \cap \{w\} \neq \emptyset.$$  

$\{w\}$ is a regular set (see [1], [2]). The minimal Rabin-Scott-acceptor is given by the following picture, provided $w = x_1 \ldots x_n(x_i \in X, 1 \leq i \leq n)$:

```
  0  x_1  x_2  2 ...  x_n  n
    |       |    |    |     |
    |       |    |    |     |
    v       v    v    v     v
  n+1
```

where 0 is the initial state, $n$ is the accepting state and $n+1$ is the fault state.

If $\delta$ is the transition-function and $\delta^*$ the extension of $\delta$ to words we immediately see:

$$\delta^*(u, i) = j \iff u = w[i, j] \quad (0 \leq i \leq j \leq n).$$

Consider a contextfree grammar $G$ and a word $w \in T^*$. Now we can use a modified triple-construction to create a grammar $G_w$ with

$$L(G_w) = L(G) \cap \{w\},$$
hence we have reduced the word problem to the emptiness-problem for contextfree grammars (see [1] for details). The modifications are elaborated in the way that terminal parts of the right-hand-side of a rule are processed directly.

The Rabin-Scott-acceptor for \( \{w\} \) has special properties (some kind of monotonicity for example). Therefore it is not necessary to construct \( G_w \) explicitly.

We make use of the recognition-matrix \( T_{w,G} \) (see [1]).

This is a matrix of format \((n + 1, n + 1)\), where numeration of columns and rows start with 0 instead of the usual 1. It is defined for arbitrary grammars by

\[
T_{w,G}[i,j] = \{ \xi \in Z \mid \xi \vdash w[i,j] \} \quad (0 \leq i,j \leq n).
\]

\( T_{w,G} \) is an upper triangular matrix.

The criterium for "\( w \in L(G)\)?" can be rewritten in the form \( \sigma \in T[0,n] \).

Since we use a modified version of the triple-construction we need a prepared table of statetransitions for the terminal parts of the grammar \( G \).

Let \( r = (p \to q) \in P \) and \( q = u_0 \xi_1 \ldots \xi_s u_s \) be the Z-decomposition of \( q \), then

\[
\text{Terminal}(r) = \{ u_i \mid 0 \leq i \leq s \}
\]

and

\[
\text{Terminal}(G) = \bigcup_{r \in P} \text{Terminal}(r)
\]

We prepare a table \( \Delta \) for all transitions

\[
\delta^*(u,i) = j \quad (u \in \text{Terminal}(G), 0 \leq i,j \leq n + 1).
\]

This table is of format \((n + 2, (|G| + 1) \cdot \#(P))\).

On a Random-Access-Machine (see[1]), the length of \( w(=n) \) must be part of the input, hence addressing an entry of \( \Delta \) takes constant time. Given \( w, |w| \) and \( G \) the preparation of \( \Delta \) needs linear time on a RAM.
Example:

\[ G : \sigma \rightarrow (\sigma)\sigma|()\square \]

generating the Dyck-language \( D_1 \) (see [1]). Let \( w = (((()()))), \) we get \( |w| = 8 \) and \( \Delta \) is given by

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \square )</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>8</td>
<td>9</td>
</tr>
<tr>
<td>(</td>
<td>1</td>
<td>2</td>
<td>9</td>
<td>4</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>)</td>
<td>9</td>
<td>9</td>
<td>3</td>
<td>9</td>
<td>5</td>
<td>6</td>
<td>9</td>
<td>8</td>
<td>9</td>
<td>9</td>
</tr>
<tr>
<td>(</td>
<td>9</td>
<td>9</td>
<td>4</td>
<td>9</td>
<td>6</td>
<td>9</td>
<td>8</td>
<td>9</td>
<td>9</td>
<td>9</td>
</tr>
</tbody>
</table>

Special treatment has to be given to the processes of erasing and chaining in context-free grammars.

Define for any \( Z' \subseteq Z \)

\[ \text{Chain}(Z') = \{ \xi \mid \exists \eta \in Z' : \xi \vdash \eta \}. \]

This operation can be done in constant time and can be prepared.

**Observation 1:**

Chain is a closure-operator, i.e.

1. \( Z' \subseteq \text{Chain}(Z') \) for \( Z' \subseteq Z \)
2. \( Z' \subseteq Z'' \subseteq Z \implies \text{Chain}(Z') \subseteq \text{Chain}(Z'') \)
3. \( Z' \subseteq Z \implies \text{Chain}(\text{Chain}(Z')) = \text{Chain}(Z') \)
4. \( \text{Chain}(\emptyset) = \emptyset \)
5. \( \text{Chain}(Z) = Z \).

**Observation 2:**

Let \( T(w) = \{ \xi | \xi \vdash w \} \) then

\( \text{Chain}(T(w)) = T(w) \) and therefore

\( \text{Chain}(T_w, \sigma[i,j]) = T_w, \sigma[i,j] \) for all \( 0 \leq i,j \leq n. \)
3 The algorithm

To compute $T_{w,G}$ we start with the initialization.

Observation 3:

(1) For all $1 \leq i \leq n$:

$$T_{w,G}[i,i] = T(\square)$$

(2) For all $0 \leq i \leq j \leq n$:

$$\text{Chain} \left( \left\{ \xi \mid \exists u \in T^* : u = w[i,j] \text{ and } \xi \rightarrow u \in P \right\} \right) \subseteq T_{w,G}[i,j]$$

Therefore we can initialize in the following way:

for $i = 0$ to $n$ do $T_{w,G}[i,i] := T(\square)$ od.

for $i = 0$ to $n$ do

for $j = i + 1$ to $n$ do

$T_{w,G}[i,j] := \text{Chain} \left( \left\{ \xi \mid \exists u \in T^* : u = w[i,j] \text{ and } \xi \rightarrow u \in P \right\} \right)$

od

od

The time costs are $O(|w|)$ for the first loop and $O(|w|^2)$ for the second and the third loop, in summary $O(|w|^2)$, since the internal operations take constant time. The complexity is measured on a RAM.

Example 1:

Consider the grammar $G$ given by

$$\sigma \rightarrow (\sigma)\sigma \mid ( ) \mid \square \text{ and } w = (()(()))$$

After initialization the current value of $T_{w,G}$ is

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>$\sigma$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>$\sigma$</td>
<td>$\emptyset$</td>
<td>$\sigma$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>$\sigma$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\sigma$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>$\sigma$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\sigma$</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>$\sigma$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\sigma$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>$\sigma$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\sigma$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>$\sigma$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\sigma$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\sigma$</td>
<td>$\emptyset$</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>$\sigma$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\sigma$</td>
<td>$\emptyset$</td>
<td>$\emptyset$</td>
<td>$\sigma$</td>
<td>$\emptyset$</td>
</tr>
<tr>
<td>8</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>$\sigma$</td>
</tr>
</tbody>
</table>

All other entries are $\emptyset$. 
Example 2:
Consider the grammar

\[
\begin{align*}
\sigma & \rightarrow \xi c \xi \mid c \mid \square \\
\xi & \rightarrow a \xi b \mid \square \text{ and } w = a^2 b^2 c a b
\end{align*}
\]

\(T(\square) = \{\xi\}\), \(\text{Chain}(\xi) = \{\sigma, \xi\}\), \(\text{Chain}(\sigma) = \{\sigma\}\)

After initialization the current value of \(T_{w,G}\) is

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>(\sigma, \xi)</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>(\sigma, \xi)</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>(\sigma, \xi)</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>(\sigma, \xi)</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>(\sigma, \xi)</td>
<td>(\sigma)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>(\sigma, \xi)</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>(\sigma, \xi)</td>
<td>0</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>(\sigma, \xi)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

All other entries are \(= 0\).

The whole computation of \(T_w\) can be easily derived from the following picture.

```
\begin{align*}
&\sigma_G \\
&\downarrow \\
&\xi \\
&\downarrow \\
&\xi_1 \\
&\downarrow \\
&u_0 \ w[i_1, j_1] \ u_1 \ \ldots \ w[i_l, j_l] \ u_l \\
&\downarrow \\
&w[i, j]
\end{align*}
```
with the following conditions:
- $\xi \to u_0\xi_1u_1 \ldots \xi_iu_i \in P$
- $\delta^*(u_0, i) = i_1, \delta^*(u_1, j_1) = j$
- $\delta^*(u_\lambda, j_\lambda) = i_{\lambda+1}$ for $0 < \lambda < l$
- $\xi_\lambda \in T_{w,G}[i_\lambda, j_\lambda]$
- $i \leq j_1 \leq j_2 \leq \ldots \leq j_{l-1} \leq j_l \leq j$
- $l \leq ||G||$

Therefore we get the following algorithm after initialization

for $i = 0$ to $n$ do

for $j = i + 2$ to $n$ do

$T_{w,G}[i, j] := \text{Chain}(T_{w,G}[i, j])$

$\cup \{\xi \in \mathcal{Z} \mid \exists l \geq 1, i \leq j_1 \leq j_2 \leq \ldots \leq j_l \leq j \text{ and } \xi \to q \in P\}$

with $\mathcal{Z}$-decomposition $q = u_0\xi_1 \ldots \xi_iu_i$:

1. $\exists 1 \leq \lambda \leq l : i < j_\lambda < j$
2. $l = 1 \implies u_0u_1 \neq \Box$
3. $\xi_1 \in T_{w,G}[\delta^*(u_0, i), j_1]$
4. $\delta^*(u_1, j_1) = j$
5. $\xi_\lambda \in T_{w,G}[\delta^*(u_{\lambda-1}, j_{\lambda-1}), j_\lambda]$ for $1 < \lambda \leq l$)

od

The criterion of success is simply

$\sigma \in T_{w,G}[0, n]$.

Example 1:
Consider the grammar $G : \sigma \to (\sigma)\sigma(\sigma)\Box$ and the word $w = (\sigma)\sigma(\sigma)(\sigma)$.

We compute $T_{w,G}[1, 5], w[1, 5] = (\sigma)(\sigma)$. The only production which can be used is $\sigma \to (\sigma)\sigma$. The only choice for the $j_\lambda$ is the following:

- $j_1 = 2$, since $\delta^*((, 1) = 2$ and $\sigma \in T_{w,G}(2, 2) = \{\sigma\}$

- $j_2 = 5$, since $\delta^*(\sigma, 2) = 3$ and $\sigma \in T_{w,G}(3, 5) = \{\sigma\}$ and

$\delta^*(\Box, 5) = 5$

hence $\sigma \in T_{w,G}[1, 5]$.

The chain operation is useless in this case.
The resulting recognition-matrix is

<table>
<thead>
<tr>
<th></th>
<th>0</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>σ</td>
<td>0</td>
<td>σ</td>
<td>0</td>
<td>σ</td>
<td>0</td>
<td>σ</td>
<td>0</td>
<td>σ</td>
</tr>
<tr>
<td>1</td>
<td>σ</td>
<td>0</td>
<td>σ</td>
<td>0</td>
<td>σ</td>
<td>0</td>
<td>σ</td>
<td>0</td>
<td>σ</td>
</tr>
<tr>
<td>2</td>
<td>σ</td>
<td>0</td>
<td>0</td>
<td>σ</td>
<td>0</td>
<td>σ</td>
<td>0</td>
<td>σ</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>σ</td>
<td>0</td>
<td>0</td>
<td>σ</td>
<td>0</td>
<td>σ</td>
<td>0</td>
<td>σ</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>σ</td>
<td>0</td>
<td>0</td>
<td>σ</td>
<td>0</td>
<td>σ</td>
<td>0</td>
<td>σ</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>σ</td>
<td>0</td>
<td>0</td>
<td>σ</td>
<td>0</td>
<td>σ</td>
<td>0</td>
<td>σ</td>
<td>0</td>
</tr>
<tr>
<td>6</td>
<td>σ</td>
<td>0</td>
<td>0</td>
<td>σ</td>
<td>0</td>
<td>σ</td>
<td>0</td>
<td>σ</td>
<td>0</td>
</tr>
<tr>
<td>7</td>
<td>σ</td>
<td>0</td>
<td>0</td>
<td>σ</td>
<td>0</td>
<td>σ</td>
<td>0</td>
<td>σ</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>σ</td>
<td>0</td>
<td>0</td>
<td>σ</td>
<td>0</td>
<td>σ</td>
<td>0</td>
<td>σ</td>
<td>0</td>
</tr>
</tbody>
</table>

The other entries are = 0.

Therefore \(( ) ( ) ( ) ( ) \in L(G)\)

**Example 2:**

Consider \(G : σ → ξσξ \mid c \mid ≠ \) and \(ξ → aξb \mid ≠ \) and \(a²b²cab\)

- \(T_{w,G}[0,2]\). We have two possible rules

1. \(σ → ξσξ\) i.e. \(l = 3, u_0 = u_1 = u_2 = u_3 = ≠ \implies j_3 = 2\).
   By definition(1) either \(j_1 = 1\) or \(j_2 = 1\).
   If \(j_1 = 1\) then \(ξ \in T_{w,G}[0,1] = 0\), a contradiction. If \(j_2 = 1\) then \(ξ \in T_{w,G}[1,2] = 0\), again a contradiction.

2. \(ξ → aξb, i.e. l = 1, u_0 = a, u_1 = b \implies j_1 = 2\), but \(δ^*(2, b) = 3\)
   a contradiction
   In summary \(T_{w,G}[0, 2] = 0\).

- \(T_{w,G}[1,3]\). Again two possible rules

1. \(σ → ξσξ, i.e. l = 3, u_0 = u_1 = u_2 = u_3 = ≠ \implies j_3 = 3\), again either \(j_1 = 2\) or \(j_2 = 2\).
   In both cases we get a contradiction.

2. \(ξ → aξb, i.e. l = 1, u_0 = a, u_1 = b, j_1 = 2\) and \(ξ \in T_{w,G}[2,2]\)
   hence \(ξ \in T_{w,G}[1,3]\).
By Chain we get $T_{w, G}[1, 3] = \{\sigma, \xi\}$

- $T_{w, i}[0, 4]$. Again two possible rules

(1) $\sigma \to \xi \sigma \xi$, i.e. $l = 3$, $u_0 = u_1 = u_2 = u_3 = \Box \implies j_3 = 4$

(i) $j_1 = 1$ impossible
(ii) $j_1 = 2$ impossible
(iii) $j_1 = 3 \implies j_2 = 3$ or $j_2 = 4$
   In the first case $\xi \in T_{w, G}[3, 4]$,
   in the second case $\sigma \in T_{w, G}[3, 4]$.
(iv) $j_2 = 2$ or $j_2 = 1$ or $j_2 = 3$ analogously.

(2) $\xi \to a\xi b$, i.e. $l = 1$, $u_0 = a, u_1 = b, j_1 = 3$,
$\xi \in T_{w, G}[1, 3] = \{\sigma, \xi\} \implies \xi \in T_{w, G}[0, 4]$

By Chain we get $T_{w, G}[0, 4] = \{\sigma, \xi\}$.

Remarks:

- The exclusion of chain-rules by condition (2) is compensated by the Chain-operation.

- By condition (1) we get $j_\lambda - j_{\lambda-1} < j - i$ ($2 \leq i \leq l$), together with condition $T_{w, G}[i, i] = T(\Box)$ for all $0 \leq i \leq n$, we can organize the algorithm in an ON-LINE-mode. Our version is OFF-LINE.

- Knowing the recognition-matrix it should be easy to construct a parser without increasing time-complexity.

We now turn our interest to time-complexity. Observe, that $j_l$ – if existent – is uniquely determined by $j$ and $u_l$ (Condition(4)).

Hence, we have "free" choices for $j_1, j_2, \ldots, j_{l-1}$. These leads to $l - 1$ loops. The crucial condition (1) can be checked by a boolean variable in the body of the loops.

Worst-case-bounds are $0 \leq j_\lambda \leq n$ ($1 \leq \lambda \leq l - 1$),
l $\leq ||G||$ and $0 \leq i, j \leq n$.

Hence, we get

$$O(n^2 \cdot n^{l-1}) = O(n^{2+||G||-1})$$

as the overall worst-case-time-bound, provided $||G|| \neq 0$.

Since preparation and initialization have time-bounds $O(n)$ and $O(n^2)$ resp., we get in whole the time-bound

$$O(||w||^{2+||G||-1}).$$
4 Special cases

I. Normalforms:
For a grammar $G$ in Chomsky-normalform all productions are of the form
$\xi_0 \rightarrow \xi_1 \xi_2$ ($\xi_{0,1,2} \in Z$) or
$\xi_0 \rightarrow t$ ($\xi_0 \in Z, t \in T$), hence
$|G| = 2$ and therefore time-complexity is $O(|w|^3)$. Indeed, in this case the
Cocke-Younger-Kasami-algorithm results.
For a grammar $G$ in 2-Greibach-normalform all productions are of the form
$\xi_0 \rightarrow t\xi_1 \xi_2$ ($\xi_{0,1,2} \in Z, t \in T$) or
$\xi_0 \rightarrow t\xi_1$ ($\xi_{0,1} \in Z, t \in T$) or
$\xi_0 \rightarrow t$ ($\xi_0 \in Z, t \in T$), hence
$|G| = 2$ and therefore time-complexity is $O(|w|^3)$, giving the same result
as in the Chomsky-normalform-case.

II. Linear grammars
Recall, a contextfree grammar is linear iff $|G| \leq 1$, hence we get the time-
complexity $O(|w|^2)$, which is the bound of Earley's algorithm, and is not
reached by the Cocke-Younger-Kasami-algorithm, without altering the al-
gorithm.

III. One-sided linear grammars
In a rightlinear grammar all productions are of the form $\xi_0 \rightarrow u\xi_1$ with
$\xi_0 \in Z, \xi_1 \in Z \cup \langle$ and $u \in T^*$.

In this case $j_1 = n$, the "target" state. Therefore we only have to compute
$T_w,\xi[n,n], \ldots, T_w,\xi[0,n]$, knowing that $T_w,\xi[n,n] = T(\langle)$.
Therefore, both phases -initialization and computation - can be simplified
drastically.

The resulting algorithm is:
Initialization:
for $i = n$ downto 0 do
$T_w,\xi[i,n] := \text{Chain}(\{\xi \in Z \mid \exists \xi \rightarrow u \in P \text{ with } u = w[i,n]\})$ od
Computation:
for $i = 0$ to $n$ do
\[ T_{w,G}[i,n] := \text{Chain}(\{\xi \in Z \mid \exists u \in T^*, \eta \in Z : \eta \in T_{w,G}[\delta^*(u,i),n] \text{ and } \xi \rightarrow u\eta \in P\}) \]
\[ \text{od} \]

Obviously, the time-complexity is $O(|w|)$ - as it should be.

The same kind of simplification can be used for leftlinear grammars, where all productions are of the form
\[ \xi_0 \rightarrow \xi_1 u \text{ with } \xi_0 \in Z, \xi_1 \in Z \cup \square \text{ and } u \in T^*. \]

In this case the "source" state 0 is fixed, hence we only have to compute $T_{w,G}[0,0], \ldots , T_{w,G}[0,n]$.

Therefore we get $O(|w|)$ as time-complexity-bound again.

Note, we do not need any normalform or a reduction to deterministic Rabin-Scott-acceptors to get the result.

5 Concluding remarks

We haven't discussed, whether it is possible to use some kind of Valiant-type reductions via interpreting $T_{w,G}$ as a "closure" and then reducing the computing of this closure to Boolean matrix-multiplication.

6 References

All what we used in this note is very familiar to those knowing the basics of formal language theory. Therefore two references will suffice
