A Graph Based Parsing Algorithm for Context-free Languages

Günter Hotz

Technical Report A 01/99 June 1999

e-mail: hotz@cs.uni-sb.de
WWW: http://www-hotz.cs.uni-sb.de

Abstract We present a simple algorithm deciding the word problem of c. f. languages in $O(n^3)$. It decides this problem in time $O(n^2)$ for unambiguous grammars and in time $O(n)$ in the case of $LR(k)$ grammars.
1 Introduction

There are several algorithms known deciding the word problem of general context-free languages in time $O(n^3)$. The algorithm of Younger [You67] is very simple and it solves the problem in time $O(n^3)$, but it takes no advantage of special cases. Kasami in [KT69] describes an algorithm, which decides this problem for unambiguous context-free grammars in time $O(n^2 \log n)$. Early [Ear70] developed an algorithm which decides the general word problem in time $O(n^3)$ but does it for unambiguous grammars in time $O(n^2)$ and for a wide class of grammars as $LR(k)$ grammars [Knu65] in time $O(n)$. His algorithm takes no advantage of grammars in a normal form. The proofs are hard to read. We present here a simple algorithm with the same runtime efficiency as Early’s algorithm.

2 Notations and Definitions

Let $V, T$ be finite alphabets, $V \cap T = \emptyset$, $S \in V$ and $P \subseteq (V \times V^2) \cup (V \times T)$ a c. f. production system in Chomsky normal form (Ch-NF). We assume that the grammar $G := (V, T, P, S)$ does not contain superfluous variables. That means for each $x \in V$ we find $u_1, u_2, u \in T^*$ such that $x \rightarrow u$ and $s \rightarrow u_1 xu_2$ holds.

We define linear forms with variables from $V$ and coefficients from the boolean algebra $\mathbb{B}$. These are mappings

$$\xi : V \rightarrow \mathbb{B}$$

and we write $\mathbb{B}(V) := \{\xi \mid \xi : V \rightarrow \mathbb{B}\}$. We use the equivalent notation

$$\xi := \sum_{v \in V} \xi(v) \cdot v$$

We define the sum and a product in $\mathbb{B}(V)$: As usual we put

$$(\xi + \eta)(v) := \xi(v) + \eta(v) \text{ for } v \in V.$$

The product $x \ast y$ for $x, y \in V$ gives all possible reductions of $xy$ relative to $P$. More formally we define

$$x \ast y := \sum_{z \in V} \zeta(z) \cdot z \iff (\zeta(z) = 1 \iff (z, xy) \in P).$$

Now we put

$$\xi \ast \eta := \sum_{x, y \in V} \xi(x) \cdot \eta(y) \cdot (x \ast y);$$
we use in this definition for $\alpha \in \mathbb{B}$ and $\xi \in \mathbb{B}(V)$ the operation $(\alpha \cdot \xi)(v) = \alpha \cdot \xi(v)$ for $v \in V$. The product "$\cdot$" is not associative. $(\mathbb{B}(V), +, \cdot)$ is distributive.

We use furthermore the notation

$$P^{-1}(t) = \sum_{z \in V} \alpha_z^t \cdot z, \quad \alpha_z^t = 1 \iff (z, t) \in P.$$ 

If the operation "$\cdot$" is associative then for $u = t_1 \cdot \ldots \cdot t_n$ and $\mu(u) := P^{-1}(t_1) \ast \ldots \ast P^{-1}(t_n)$ we have

$$u \in L(G) \iff \mu(u)(s) = 1.$$ 

In this case $(\mathbb{B}(V), \ast)$ is a finite monoid and $P^{-1} : T^* \longrightarrow (\mathbb{B}(V), \ast)$ is a homomorphism and therefore $L(G)$ is regular.

3 The Graph $\Gamma(G, u)$

We assign to the grammar $G$ and $u \in T^*$ an oriented graph $\Gamma = (K, E)$; $K$ is the set of vertices and $E$ the set of edges and $n := |u|$ the length of $u$.

$$K \cup \{(v, i, 0) \mid v \in V, 1 < i \leq n\}$$
$$\quad \cup \{(v, i, 1) \mid v \in V, 1 \leq i < n + 1\}$$
$$E \cup \{((v, i, 1), (v, j, 0)) \mid V \longrightarrow t_i \ldots \cdot t_{j-1}, 1 \leq i < j \leq n + 1\}$$

Obviously it holds

$$u \in L(G) \iff ((s, 1, 1), (s, n, 0)) \in E.$$ 

The graph $\Gamma$ is closed under the following operation: Let be $i < j < m$

$$(x, i, 1) \xrightarrow{s_1} (x, j, 0),$$
$$(y, j, 1) \xrightarrow{s_2} (y, m, 0)$$

edges of $\Gamma$ and

$$\zeta := x \ast y.$$ 

If $\zeta(z) = 1$, then the edge

$$(z, i, 1) \xrightarrow{s_3} (z, m, 0)$$

2
is in $\Gamma$. We write in this case $s_3 := s_1 \ast s_2$; in general there may be several edges $s'_3$ in the relation $s'_3 := s_1 \ast s_2$.

This closure property corresponds to

$$\begin{align*}
x &\longrightarrow t_1 \cdots t_{j-1}, \\
y &\longrightarrow t_j \cdots t_{m-1}
\end{align*}$$

and

$$z \longrightarrow xy.$$ 

Therefore we have $z \longrightarrow t_1 \cdots t_{m-1}$ and from this follows by definition of $\Gamma$, that $s_3$ is in $E$.

**Lemma 1.** If there are two different operations producing the same edge $s_3$, then $G$ is ambiguous.

**Proof 1.** Let $s_1, s_2$ and $s'_1, s'_2$ two pairs of edges from $\Gamma$ producing under the explained operation the edge $s_3$, then we have the two different derivations

$$\begin{align*}
z &\longrightarrow xy, \\
x &\longrightarrow u_1, \\
y &\longrightarrow u_2, \\
u_3 &\equiv u_1 \cdot u_2
\end{align*}$$

$$\begin{align*}
z &\longrightarrow x'y', \\
x' &\longrightarrow u'_1, \\
y' &\longrightarrow u'_2, \\
u_3 &\equiv u'_1 \cdot u'_2.
\end{align*}$$

Now we assume $G$ not containing superfluous variables. Therefore exist the derivations

$$s \longrightarrow \bar{u}z\bar{u} \longrightarrow \bar{u}u_1 \cdot u_2\bar{u} = \bar{u}u'_1 \cdot u'_2 \cdot \bar{u} \in T^*.$$ 

So we have more than one derivation of $\bar{u}u_3\bar{u}$ from $S$, i.e. $G$ is ambiguous.

4 The algorithm

We now construct a sequence $\Gamma_1, \Gamma_2, \ldots, \Gamma_n$ of subgraphs of $\Gamma$ such that $\Gamma_1$ depends only on $t_1$ and with $\Gamma_n = \Gamma$. We give an operation which constructs $\Gamma_{i+1}$ from $\Gamma_i$ and estimate the complexity of this operation.

Let $\Gamma_i := (K_i, E_i)$ for $i = 1, \ldots, n$ and

$$\begin{align*}
K_i &:= (v, l, \varepsilon) \in K \mid 1 \leq l \leq i, \varepsilon \in \{0, 1\} \cup \{(x, i + 1, 0) \mid x \in V\}, \\
E_i &:= \{s \in E \mid \text{source}(s), \text{sink}(s) \in K_i\}.
\end{align*}$$

The construction of $\Gamma_1$ can be done in time $O(1)$. 

3
We assume $\Gamma_i, i < n$ has been constructed.

We add $t_{i+1}$ and $\{(v,i+1,1) \mid v \in V \cup \{v,i+2,0 \mid v \in V\}$ to $K_i$. We in the first step add the following edges of $E$ to $E_i$:

$$(v, i+1, 1) \rightarrow (v, i+2, 0) \text{ for } v \rightarrow t_{i+1}.$$ 

Let $\Gamma'_i$ the result of this construction.

Now we apply the closure operations

$$s_1 * s_2 \rightarrow s_3$$

to edges $s_1, s_2$ from $\Gamma'_i$. $\Gamma_i$ being closed under these operations we have to begin with the new edges in $\Gamma'_i$. We have the following situation

$$(x,j,1) \xrightarrow{s_1} (x,i+1,0) \quad (y,i+1,1) \xrightarrow{s_2} (y,i+2,0).$$

We built from $s_1 * s_2$

$$(z,j,1) \xrightarrow{s_3} (z,i+2,0),$$

if $(z,xy) \in P$.

Iterating this construction in the worst case we need $O(n^2)$ elementary operations to construct $\Gamma_{i+1}$ from $\Gamma_i$, because each edge of $\Gamma'_i$ we have to consider only once.

To construct $\Gamma_n$ by this procedure therefore needs in the worst case $O(n^3)$ $*$-operations.

If the grammar is unambiguous we construct each edge only one time. Operations $s_1 * s_2$ which do not produce a new edge we are able to exclude by only once inspecting the pairs of vertices $(x,l,0), (y,l,1)$. If $x \neq y = 0$, then none of the pairs

$$(x,l,0) \xrightarrow{s_1} (x,l,0),$$

$$(y,l,1) \xrightarrow{s_2} (y,l,1),$$

has to be considered. Therefore in this case we need only $O(n^2)$ steps because this is the bound for the number of edges in $\Gamma$. So we proved the

**Theorem 1.** The algorithm defined here solves the word problem for c. f. languages in time $O(n^3)$. In the case of unambiguous grammars the running time of the algorithm is $O(n^2)$. 

4
**Corollary 1.** The algorithm solves the word problem in the case of grammars with $m$-bound ambiguity in time $O(n^2 \cdot m)$.

**Proof 2.** From the $m$-bound ambiguity it follows that the algorithm draws each new edge maximal $m$ times.

Now we study the case $G$ is a $LR(k)$ grammar.

$LR(k)$ grammars are characterized by the following property: For $uvu' \in L(G)$ and $|v| = k$ let $\overline{w}_1, \ldots, \overline{w}_l$ be the reduced words of $u \cdot v$ relative to $G$. Then the set of this words has a common prefix $\overline{u}$, where $\overline{u}$ is a reduced word of $u$, such that we can write

$$\overline{w}_1 + \ldots + \overline{w}_t = \overline{u} \cdot (\overline{v}_1 + \ldots + \overline{v}_l), \quad |v_i| \leq k \text{ for } i = 1, \ldots, l.$$

This property enables us to compute an upper bound for the number $|\Gamma_i|$ of edges in $\Gamma_i$.

Obviously we have

$$|\Gamma_1| \leq m \quad \text{for } m := \#V.$$

We assume $\Gamma_i$ being constructed. We then get $\Gamma_{i+1}$ by the following steps:

1. We compute $P^{-1}(t_{i+1})$, which produces not more than $m$ new edges.

2. We match the new edges with the existing edges. This leads to new edges connecting vertices belonging to

$$(\overline{v}_1 + \ldots + \overline{v}_l) \cdot P^{-1}(t_{i+1})$$

and edges connecting vertices belonging to $vt_{i+1}$ with edges belonging to $\overline{u}$.

The number of edges belonging to the first class is bound by a constant $c$ depending on $m = \#V$ and $k$. The number of the edges belonging to the second class is 0 if $\overline{v}_i$ is prefix of $\overline{v}_{i+1}$. It is 1 if $|u_{i+1}| = |u_i|$ and it is $|u_i| - |u_{i+1}|$ if reductions of the reduced word $u_i$ take place. So we have

$$|\Gamma_{i+1}| \leq |\Gamma_i| + C + |\overline{u}| - |\overline{u}_{i+1}| + 1.$$

From this we get

$$|\Gamma_n| = O(n).$$

From this follows

**Theorem 2.** The given graph algorithm solves the word problem for $LR(k)$ grammars $G := (V, T, P, S)$ and words $w \in T^*$ with $= O(n)$ *-operations.

It is obvious that the *-operations can be performed on a computer in time only depending on $G$. This means that it can be done in constant time relative to $|w|$.
Literatur


