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.

Fachbereich 14 Informatik
Universität des Saarlandes
Postfach 15 11 50
66041 Saarbrücken
Germany

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 $P \subset (V \times V^2) \cup (V \times T)$ 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 $x \longrightarrow u$ and $s \longrightarrow u_1 x u_2$ holds.

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

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

and we write $\mathbb{B}\langle V \rangle := \{\xi \mid \xi : V \longrightarrow \mathbb{B}\}$. We use the equivalent notation

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

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

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

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

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

Now we put

$$\xi * \eta := \sum_{x,y \in V} \xi(x) \cdot \eta(y) \cdot (x * y);$$

we use in this definition for $\alpha \in \mathbb{B}$ and $\xi \in \mathbb{B}\langle V\rangle$ the operation $(\alpha\cdot\xi)(v) = \alpha\cdot\xi(v)$ for $v \in V$. The product "$*$" is not associative. $(\mathbb{B}\langle V\rangle, +, *)$ is distributive. We use furthermore the notation

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

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

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

In this case $(\mathbb{B}\langle V\rangle, *)$ is a finite monoid and $P^{-1} : T^* \longrightarrow (\mathbb{B}(V), *)$ 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.

$$\begin{array}{rcl} K & \cup & \{(v,i,0) \mid v \in V, 1 < i \le n\} \\ & \cup & \{(v,i,1) \mid v \in V, 1 \le i < n+1\} \\ E & \cup & \{((v,i,1),(v,j,0)) \mid V \longrightarrow t_i \cdot \ldots \cdot t_{j-1}, 1 \le i < j \le n+1\} \end{array}$$

Obviously it holds

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

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

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

edges of Γ and

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

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

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

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

This closure property corresponds to

$$x \longrightarrow t_i \cdot \ldots \cdot t_{j-1},$$
$$y \longrightarrow t_j \cdot \ldots \cdot t_{m-1}$$

and

$$z \longrightarrow xy.$$

Therefore we have $z \longrightarrow t_1 \cdot \ldots \cdot t_{m-1}$ and from this follows by definition of Γ, 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 Γ producing under the explained operation the edge s_3, then we have the two different derivations

$$\begin{array}{llll} z \longrightarrow xy, & x \longrightarrow u_1, & y \longrightarrow u_2, & u_3 = u_1 \cdot u_2 \\ z \longrightarrow x'y', & x' \longrightarrow u'_1, & y' \longrightarrow u'_2, & u_3 = u'_1 \cdot u'_2. \end{array}$$

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

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

So we have more than one derivation of $\tilde{u} u_3 \overline{u}$ 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 Γ such that Γ_1 depends only on t_1 and with $\Gamma_n = \Gamma$. We give an operation which constructs Γ_{i+1} from Γ_i and estimate the complexity of this operation.

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

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

The construction of Γ_1 can be done in time $O(1)$.

We assume Γ_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:

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

Let Γ'_i the result of this construction.
Now we apply the closure operations

$$s_1 * s_2 \longrightarrow s_3$$

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

$$\begin{array}{llll} (x, j, 1) \xrightarrow{s_1} & (x, i+1, 0) & & \\ & & (y, i+1, 1) \xrightarrow{s_2} & (y, i+2, 0). \end{array}$$

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 Γ_{i+1} from Γ_i, because each edge of Γ'_i we have to consider only once.

To construct Γ_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 $x * y = 0$, then none of the pairs

$$\begin{array}{lll} \xrightarrow{s_1} & (x, l, 0) & \\ & (y, l, 1) & \xrightarrow{s_2} \end{array}$$

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 Γ. 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 $O(n^2)$.*

Corollar 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

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

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

Obviously we have

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

We assume Γ_i being constructed. We then get Γ_{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 $\overline{u}_i$ is prefix of $\overline{u}_{i+1}$. 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

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

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

[Ear70] J. Early. An efficient context-free parsing algorithm. *Com. ACM*, 13, 1970.

[Knu65] D. E. Knuth. On the translation of languages from left to right. *Information and Control*, 8, 1965.

[KT69] T. Kasami and K. Tori. A syntax-analysis procedure for unambiguous context-free grammars. *ACM*, 16, 1969.

[You67] D. H. Younger. Recognition and parsing of context-free languages in time n^3. *Information and Control*, 10, 1967.