Journal of Universal Computer Science, vol. 6, no. 9 (2000), 850-860
submitted: 1/3/00, accepted: 8/9/00, appeared: 28/9/00 © Springer Pub. Co.

Uniquely Parsable Accepting Grammar Systems

Carlos Martín-Vide
(Research Group in Mathematical Linguistics and Language Engineering
Rovira i Virgili University
Pça. Imperial Tàrraco 1, 43005 Tarragona, Spain
cmv@astor.urv.es)

Victor Mitrana
(University of Bucharest, Faculty of Mathematics
Str. Academiei 14, 70109, Bucharest, Romania
mitrana@funinf.math.unibuc.ro)

Abstract: We extend the restrictions which induce unique parsability in Chomsky grammars to accepting grammar systems. It is shown that the accepting power of global RC-uniquely parsable accepting grammar systems equals the computational power of deterministic pushdown automata. More computational power, keeping the parsability without backtracking, is observed for local accepting grammar systems satisfying the prefix condition. We discuss a simple recognition algorithm for these systems.

Category: F.4.2, F.4.3

1 Introduction

Cooperating distributed grammar systems (CD grammar systems) have been introduced in [Csuhaj-Varju and Dassow 1990] as a grammatical approach to the so-called "blackboard model" in the problem solving theory [Nii 1989]. A similar language generating device was considered in [Meersman and Rozenberg 1978], while a particular variant of it appeared in [Atanasiu and Mitrana 1989], with motivations coming from regulated rewriting area. Most of the results known in this field until the middle of 1992 can be found in [Csuhaj-Varju et al. 1994], while more recent results are surveyed in [Dassow et al. 1997].

However, there are still lots of classical topics in formal language theory or in related areas which have not been studied so far in the grammar systems set-up. The construction of parsers is such a topic which is not only of theoretical interest, but will make grammar systems more appealing to researchers in applied computer science. This will clearly bring to the user all the advantages of having a model which can cope with such phenomena as cooperation and distribution of the work carried out by several processors. Of interest to this aim are the results in [Fernau et al. 1996] and [Dassow and Mitrana 1999]. Thus, a comparison of the accepting capacity of CD grammar systems with respect to their generating power, or to that of other classes of grammars in the regulated rewriting area, is presented in [Fernau et al. 1996] and [Fernau and Holzer 1996]. Accepting multi-agents systems seem to be more adequate for the initial motivation of introducing grammar systems. [Dassow and Mitrana 1999] considers the same strategies of cooperation among the components of a grammar system for the stacks of a multi-stack pushdown automaton, in the aim of characterizing the languages generated by grammar systems in terms of recognizers; but that model turned out to be too powerful for the authors' scope.

In [Mihalache and Mitrana 1997], one investigates the effect of some syntactical constraints similar to those considered for strict deterministic context-free grammars applied to CD grammar systems. It is known that the family of languages generated by strict deterministic context-free grammars is the same as the family of languages generated by $LR(0)$ grammars (see Theorem 11.5.5 in [Harrison 1978]), which are ones of the most used class of grammars for parsing. They obtained a promising result in this respect, namely the *unambiguity* of derivations holds for some classes of grammar systems.

We belive that a more involved study of the derivations in a CD grammar system would be very useful to an eventual constructor of parsers for the languages generated by grammar systems. To this aim, the present paper introduces a new class of accepting devices called uniquely parsable accepting grammars systems (UPAGS, for short). These mechanisms have a restricted type of accepting rules such that parsing can be done without backtracking. Each component of a UPAGS is a so-called RC-uniquely parsable grammar [Morita et al. 1997] viewed as an accepting grammar. In [Morita et al. 1997] a hierarchy of uniquely parsable grammars that gave a simple grammatical characterization of the deterministic counterpart of the classical Chomsky hierarchy was introduced.

When extending the restrictions of unique parsability to accepting grammar systems, two variants should be taken into consideration, depending on the level, *local/global*, to which the restrictions address. In the global level case, the restrictions apply to all rules of a system considered altogether as a single set. The accepting capacity of these systems equals the accepting power of deterministic pushdown automata. In other words, each system collapses to an RC-uniquely parsable grammar. When the local level is considered, where conditions apply to all rules of each component independently, some more restrictive classes have more computational power but still keeping the parsability without backtracking. We propose a simple recognition algorithm for these systems.

2 RC-accepting grammars

An alphabet is always a finite and nonempty set of letters; if V is an alphabet, then V^* is the set of all words over V. The empty word is denoted by ε and the set of all nonempty words is $V^+ = V^* - \{\varepsilon\}$. For a word $x \in V^*$, we denote by $|x|$ the length of x and by $\tilde{x}$ the mirror image of x. For a finite set A, $\text{card}(A)$ denotes the number of elements in A. The reader is referred to [Rozenberg and Salomaa 1997] for basic elements of formal language theory.

Unlike usual, when grammars are understood as word generating devices, in this paper we are dealing with word accepting grammars. The idea is not new, variants of accepting grammars have been considered for the main regulated generative grammars, see, e. g., [Fernau et al. 1996] and the references therein. Furthermore, the accepting grammars which are to be defined here are of a special form which allows us to impose some syntactical conditions leading to parsing algorithms without backtracking.

An *RC-accepting grammar* (RC-AG for short) is a construct:

$$G = (N, T, S, P, \$)$$

where N and T are the sets of nonterminal and terminal symbols respectively, $S \in N$ is the goal symbol, $\$ \notin N \cup T$ is a special end-marker, and P is a finite set of accepting rules of the following forms:

(i) $\alpha x \longrightarrow \beta,\ \$\alpha x \longrightarrow \$\beta,\ \alpha x\$ \longrightarrow \beta\$,\ \$\alpha x\$ \longrightarrow \$\beta\$$,
with $\alpha \in N^*, \beta \in N^+, x \in T^+$. The rules of the form above are called *right-terminating rules* $(R - rules)$.

(ii) $\alpha \longrightarrow A,\ \$\alpha \longrightarrow \$A,\ \alpha\$ \longrightarrow A\$,\ \$\alpha\$ \longrightarrow \$A\$,\ \$\$ \longrightarrow \$A\$,$
with $A \in N,\ \alpha \in N^+$. The rules of the form above are called *context-free-like rules (C-rules).*

This definition is essentially similar to that introduced in [Morita et al. 1997] for generative grammars. The relation of direct reduction in G, denoted by $\Longrightarrow_P$, is defined as usual, namely $x \Longrightarrow_P y$ iff $x = x_1\alpha x_2, y = x_1\beta x_2$ and $\alpha \longrightarrow \beta \in P$. The reductions consisting of k direct reduction steps as above are denoted by $\Longrightarrow_P^k$, while an arbitrary reduction is denoted by $\Longrightarrow_P^*$.

The language accepted by an RC-accepting grammar G as above is

$$Acc(G) = \{w \in T^* \mid \$w\$ \Longrightarrow_P^* \$S\$\}.$$

We shall prove first that these grammars accept exactly the class of context-free languages. We consider that two languages are equal if they differ by at most the empty word.

Proposition 1. *The family of languages accepted by RC-accepting grammars is the family of context-free languages.*

Proof. We start with a context-free grammar G in the Greibach normal form [Rozenberg and Salomaa 1997], with productions of the form $A \to bA_1A_2\ldots A_n$, where $A, A_1, A_2, \ldots, A_n$ are nonterminals and b is a terminal symbol. We consider a new nonterminal, denoted by Λ, and for each production $A \to b\alpha$ of G, we consider the following RC-accepting rules:

- $\$b \to \tilde{\$}\alpha$, if $A = S$,
- $Ab \to \tilde{\alpha}$ and $A\Lambda b \to \tilde{\alpha}$, if $\alpha \neq \varepsilon$,
- $Ab \to \Lambda$ and $A\Lambda b \to \Lambda$, if $\alpha = \varepsilon$.

Add to the set of rules above the rules $\$\$ \to \$S\$$ and $\$\Lambda\$ \to \$S\$$. Now it is easy to prove formally, by induction on r, that

$$\begin{aligned} S \Longrightarrow_{ld}^r x\alpha \Longrightarrow_{ld}^* xy \text{ iff }\ & \$xy\$ \Longrightarrow^r \$\tilde{\alpha}y\$ \Longrightarrow^* \$S\$ \text{ or} \\ & \$xy\$ \Longrightarrow^r \$\tilde{\alpha}\Lambda y\$ \Longrightarrow^* \$S\$. \end{aligned}$$

We have denoted by $\Longrightarrow_{ld}$ the relation of leftmost derivation in G. Therefore, the RC-accepting grammar defined above accepts the language generated in the leftmost manner by G. In conclusion, all context-free languages can be accepted by RC-accepting grammars.

The converse inclusion follows from a result of Baker, see [Baker 1974]. This result states that each grammar having its productions of the form

$$x_0A_1x_1A_2\ldots x_{n-1}A_nx_n \to y_0B_1y_1B_2\ldots y_{m-1}B_my_m$$

where x_i, y_j are terminal words, $0 \leq i \leq n$, $0 \leq j \leq m$, and A_i, B_j are nonterminals, $1 \leq i \leq n$, $1 \leq j \leq m$, and satisfying one of the following two conditions:

1. $n = 1$,
2. there exists some j such that $|y_j| > |x_i|$ for all $0 \leq i \leq n$,

generates a context-free language.

Given an RC-accepting grammar $G = (N, T, S, P, \$)$ we construct a grammar $G' = (N, T \cup \{\$, c\}, S, P')$, where c is a new terminal and the rules of P' are defined as follows:

1. Take each C-rule of P and change its two sides with each other.
2. Do the same with each R-rule of P excepting those of the form $\$\alpha x \to \$\beta$, with $\alpha \neq \varepsilon$. For each such a rule, we add the rule $\$\beta \to \$\alpha xc$.

Clearly, G' satisfies the conditions required by the aforementioned result. Furthermore, $h(L) = Acc(G)$ holds, where h is a homomorphism which erases the symbols \$ and c and leaves unchanged the other ones, and L is the language generated by G'. As the class of context-free languages is closed under arbitrary homomorphisms, it follows that $Acc(G)$ is also context-free.

3 UP-conditions

As one can see, there are many possible rules that can be applied to a certain step of a reduction process in an RC-accepting grammar. Despite that the following conditions do not change the nondeterministic feature of the reduction process, they induce a kind of confluence property to all reductions. Furthermore, for each reduction there exists a unique leftmost reduction. Thus, parsing can be performed in a deterministic way by leftmost reductions.

An RC-accepting grammar $G = (N, T, S, P, \$)$ is an RC-uniquely parsable accepting grammar (RC-UPAG, for short) iff the following conditions, called *UP-conditions*, are satisfied:

1. Neither $S, \$S, S\$$ nor $\$S\$$ belong to $dom(P) = \{\alpha \mid \alpha \to \beta \in P\}$.
2. For any $1 \le j \le n$ and for any two rules $\alpha_i \longrightarrow \beta_i \in P,\ i = 1, 2$, if $\alpha_1 = \alpha_1'\delta$ and $\alpha_2 = \delta\alpha_2'$, for some $\delta, \alpha_1', \alpha_2' \in (N \cup T \cup \{\$\})^+$, then $\beta_1 = \beta_1'\delta$ and $\beta_2 = \delta\beta_2'$, for some $\beta_1, \beta_2 \in (N \cup T \cup \{\$\})^*$.
3. For any two rules $\alpha_i \longrightarrow \beta_i \in P,\ i = 1, 2$, if $\alpha_1 = \gamma\alpha_2\gamma'$, for some $\gamma, \gamma' \in (N \cup T \cup \{\$\})^*$, then $\beta_1 = \beta_2$ and $\gamma = \gamma' = \varepsilon$.

Note that for any two rules $\alpha_i \longrightarrow \beta_i,\ i = 1, 2$, if $\alpha_1 = \alpha_1'\delta,\ \alpha_2 = \delta\alpha_2'$, then $\delta \in N \cup \{\$\}$.

Otherwise stated, if a suffix and a prefix of the lefthand side of two rules are the same word, then this word remains unchanged by application of these rules no matter the order. Thus, if more rules are applicable at a given step, the application of one of them does not cancel the possibility of a further application of any other one. ¿From this property, one can derive that any given terminal word is either accepted or rejected by a process without backtracking. Consequently, RC-UPAGs preserve also a nice property of context-free grammars, namely for every accepted word w there exists a leftmost reduction which accepts w (a leftmost reduction is a reduction such that each direct reduction is made as far to the left as possible).

In [Morita et al. 1997] it was proved that the class of languages generated by RC-uniquely parsable grammars is the class of languages accepted by deterministic push-down automata.

4 RC-uniquely parsable accepting grammar systems

We extend the above definitions for accepting grammar systems, more precisely for accepting cooperating grammar systems [Fernau et al. 1996]. Roughly speaking, an RC-accepting grammar system (RC-AGS) consists of several RC-accepting grammars that are called the components of the system. Formally, an RC-AGS of degree n is a construct

$$\Gamma = (N, T, S, P_1, P_2, \ldots, P_n, \$),$$

where $(N, T, S, P_i, \$)$ is an RC-AG for all $1 \le i \le n$. For all grammars and grammar systems that are to be considered in the sequel contain only RC-rules we simply omit the prefix RC.

If all components of an AGS Γ are UPAG, then Γ is a *local uniquely parsable accepting grammar system*. If the grammar $G_\Gamma = (N,\ T,\ S,\ \bigcup_{i=1}^{n} P_i,\ \$)$ is a UPAG, then Γ is called a *global uniquely parsable accepting grammar system*. Obviously, each global UPAGS is a local UPAGS but the converse does not hold.

The relation of direct reduction in every set P_i, denoted by $\Longrightarrow_{P_i}$ is defined as usual, namely $x \Longrightarrow_{P_i} y$ iff $x = x_1\alpha x_2, y = x_1\beta x_2$ and $\alpha \longrightarrow \beta \in P_i$. The reduction consisting of exactly k direct reduction steps as above is denoted by $\Longrightarrow_{P_i}^{k}$, an arbitrary reduction is denoted by $\Longrightarrow_{P_i}^{*}$, whereas a maximal reduction is denoted by $\Longrightarrow_{P_i}^{t}$. Formally, we write $x \Longrightarrow_{P_i}^{t} y$ iff $x \Longrightarrow_{P_i}^{*} y$ and there is no $z \in (N \cup T \cup \{\$\})^*$ such that $y \Longrightarrow_{P_i} z$.

The language accepted by Γ, denoted by $Acc(\Gamma)$, is defined by

$$Acc(\Gamma) = \{w \in T^* \mid \$w\$ \Longrightarrow^t_{P_{i_1}} \$w_1\$ \Longrightarrow^t_{P_{i_2}} \dots \Longrightarrow^t_{P_{i_m}} \$w_m\$ = \$S\$,\ \text{for some } m \geq 1, 1 \leq i_j \leq n, 1 \leq j \leq m\}.$$

Example 1 *The AGS*

$$\Gamma = (\{S, A, B, C, A_1, B_1, C_1, X, Y, Z, F\}, \{a, b, c\}, S, P_1, P_2, P_3, \$),$$

where P_1 consists of the following rules

$$\begin{array}{ll} \$XYZ\$ \longrightarrow \$S\$, & AA_1 \longrightarrow X, \\ \$A_1B_1C_1\$ \longrightarrow \$S\$, & BB_1 \longrightarrow Y, \\ AXY \longrightarrow F, & CC_1 \longrightarrow Z \\ BYZ \longrightarrow F, & \end{array}$$

P_2 consists of the following rules

$$\begin{array}{lll} AX \longrightarrow A_1, & Bb \longrightarrow BB, & Ab \longrightarrow A_1B, \\ BY \longrightarrow B_1, & Bc \longrightarrow B_1C, & Aa \longrightarrow AA, \\ CZ \longrightarrow C_1, & Cc \longrightarrow CC, & \$a \longrightarrow \$A, \\ & C\$ \longrightarrow C_1\$, & \end{array}$$

and P_3 consists of the following rules

$$\$XB \longrightarrow \$F,$$
$$\$XYC \longrightarrow \$F,$$

is a local but no global UPAGS accepting the language $L = \{a^nb^nc^n | n \geq 1\}$.
Indeed, each word $a^nb^nc^n$, $n \geq 1$, is accepted by Γ as shown below for $n = 3$.

$$\$a^3b^3c^3\$ \Longrightarrow^t_{P_2} \$AAA_1BBB_1CCC_1\$ \Longrightarrow^t_{P_1} \$AXBYCZ\$ \Longrightarrow^t_{P_2} \$A_1B_1C_1\$ \Longrightarrow^t_{P_1} \$S\$.$$

On the other hand, the accepting process of any word $a^nb^mc^p$ is blocked in the third component provided $n \neq m$ or $m \neq p$.

4.1 Global UPAGS

In this section we shall prove that each global UPAGS can be replaced by a UPAG with the same accepting power. This can be simply done by putting together all components into just one component.

The following result, which is an immediate consequence of Theorem 2.1 in [Morita et al. 1997], is the main tool in our further reasoning.

Proposition 2. *Let $G = (N, T, S, P, \$)$ be a UPAG and let x be a word in $(N \cup T \cup \{\$\})^+$. If $x \Longrightarrow^* \$S\$$, then for any $y \in (N \cup T \cup \{\$\})^+$ such that $x \Longrightarrow^k y$, for some $k \geq 1$, the relation $y \Longrightarrow^* \$S\$$ holds.*

Now we can prove the main result of this section.

Proposition 3. *For each global UPAGS Γ there exists a UPAG that accepts the language $Acc(\Gamma)$.*

Proof. Let $\Gamma = (N, T, S, P_1, P_2, \ldots, P_n, \$)$ be a global UPAGS of degree n; we consider the accepting grammar $G = (N, T, S, P = \bigcup_{i=1}^{n} P_i, \$)$. We prove that $Acc(\Gamma) = Acc(G)$. Clearly, the non-trivial inclusion is $Acc(G) \subseteq Acc(\Gamma)$. Take an arbitrary word $x \in Acc(G)$ that is $\$x\$ \Longrightarrow_P^* \$S\$$. The following algorithm provides an accepting path in Γ.

Algorithm 1 begin
  $k := 1$;
  while $x \neq S$ do
    assume that $\$x\$ \Longrightarrow_{P_j}^{+} \$y\$ \Longrightarrow_P^* \$S\$$ *for some* $1 \leq j \leq n$.
    find the unique x_k *such that* $\$y\$ \Longrightarrow_{P_j}^{t} \$x_k\$$;
    $i_k := j$;
    $x := x_k$;
  endwhile;
  end.

Now, by this algorithm, it is easy to infer that

$$\$x\$ \Longrightarrow_{P_{i_1}}^{t} \$x_1\$ \Longrightarrow_{P_{i_2}}^{t} \ldots \Longrightarrow_{P_{i_k}}^{t} \$x_k\$ = \$S\$,$$

which concludes the proof.

A direct reduction of a word that is made as far to the left as possible is called a direct leftmost reduction. The direct leftmost reduction from x to y in P_i is denoted by $x = \!\!\gg_{P_i} y$. Formally, $x = \!\!\gg_{P_i} y$ iff $x = x_1 \alpha x_2, y = x_1 \beta x_2, \alpha \longrightarrow \beta \in P_i$, and for any $\alpha' \in dom(P_i)$ and any decomposition $x = x_1' \alpha' x_2'$ we have $|x_1| < |x_1'|$. In a similar way as above, the t-leftmost reduction in a component P_i is defined.

Proposition 4. *Let* $\Gamma = (N, T, S, P_1, P_2, \ldots, P_n, \$)$ *be a UPAGS. If* $x \Longrightarrow_{P_i}^{t} y$, *then* $x = \!\!\gg_{P_i}^{t} y$.

Proof. This is to be proved by induction on the number of direct reductions in $x \Longrightarrow_{P_i}^{t} y$, say k.

The case $k = 1$ is trivially true, since only one rule is applicable to x. Assume the assertion holds for k. Consider a $k + 1$ steps reduction

$$x \Longrightarrow_{P_i} z \Longrightarrow_{P_i}^{t} y$$

in which the first step is a leftmost reduction i.e. $x = \!\!\gg_{P_i} z$. By Proposition 2, this supposition does not induce any loss of generality. ¿From the induction hypothesis it follows $z = \!\!\gg_{P_i}^{t} y$, hence $x = \!\!\gg_{P_i}^{t} y$ and we are done.

The language accepted in the leftmost manner by a UPAGS $\Gamma = (N, T, S, P_1, P_2, \ldots, P_n, \$)$ is defined by

$$Acc(\Gamma, left) = \{w \in T^* \mid \$w\$ = \$w_1\$ = \!\!\gg_{P_{i_1}}^{t} \$x_1\$ = \!\!\gg_{P_{i_2}}$$
$$\$w_2\$ = \!\!\gg_{P_{i_2}}^{t} \$x_2\$ \quad = \!\!\gg \ldots = \!\!\gg_{P_{i_m}}^{t} \$x_m\$ = \$S\$,$$

for some $m \geq 1, 1 \leq i_j \leq n, 1 \leq j \leq m\}$.

In [Morita et al. 1997] it is proved that, parsing can be always performed in a unique way by leftmost reduction in any UPAG. By the last two propositions one can easily infer a similar result for UPAGSs, namely

Proposition 5. *For each global UPAGS* Γ, $Acc(\Gamma, left) = Acc(\Gamma)$ *holds. Moreover, each word in* $Acc(\Gamma)$ *there is a unique leftmost reduction in* Γ.

4.2 Local RC-UPAGS

As we have seen, the accepting process of every word in a global RC-UPAGS is done without backtracking. Unfortunately, no increase of the accepting capacity of these systems can be observed in comparison with that of RC-UPAGs.

On the other hand, as shown in Example 1, local RC-UPAGSs are more powerful than global RC-UPAGSs, but one loses the confluence property. Otherwise stated, one cannot accept every word without backtracking because the component which is to become active might be chosen from a set of components that are able to reduce the current word. However, once a component has been chosen, the reduction can be made without backtracking till the component is disabled.

In order to keep the general confluence property we restrict our investigation to leftmost reductions only. Even in this case the next active component may not be uniquely determined. For overcoming this drawback we add the following condition called the *prefix condition.* An RC-UPAGS $\Gamma = (N, T, S, P_1, P_2, \dots, P_n, \$)$, satisfies the prefix-condition iff

$$Pref(dom(P_i)) \cap dom(P_j) = \emptyset$$

for all $1 \leq i \neq j \leq n$. By $Pref(A)$ we have denoted the set of all prefixes of the words in A.

The accepting capacity of RC-UPAGSs satisfying the prefix condition seems to be very large. The system in Example 1 satisfies the prefix condition. A bit more intricate example is discussed below.

Example 2 *Let V be an alphabet and c be a symbol not in V. We define the RC-UPAGS*

$$\Gamma = (\{S, X, Y, F\}, V, S, (P_a)_{a \in V}, (P'_a)_{a \in V}, (P''_a)_{a \in V}, P, \$),$$

where

$$\begin{aligned} P_a &= \{\$a \to \$X, ca \to X\} \cup \{cb \to F \mid b \in V \setminus \{a\}\}, \\ P'_a &= \{Xa \to Y\} \cup \{Xb \to F \mid b \in V \setminus \{a\}\}, \\ P''_a &= \{Ya \to X\} \cup \{Yb \to F \mid b \in V \setminus \{a\}\}, \end{aligned}$$

for all $a \in V$, and

$$\begin{aligned} P = {} & \{\$XX\$ \to \$S\$, \$YY\$ \to \$S\$\} \cup \\ & \bigcup_{a \in V} \{\$XaX\$ \to \$F\$, \$XaY\$ \to \$F\$, \$YaX\$ \to \$F\$, \$YaY\$ \to \$F\$, \\ & \quad \$XXa \to \$F, \$YXa \to \$F, \$XYa \to \$F, \$YYa \to \$F\}. \end{aligned}$$

It is easy to notice that Γ is a local UPAGS that satisfies the prefix condition. The language accepted by Γ is $\{xcx \mid x \in V^+\}$.

In what follows, we shall discuss a simple recognition algorithm for the leftmost reduction in local UPAGSs satisfying the prefix condition. We start with some preliminary notations. For a set of words A we denote by

- $PSuf(A)$, the set of all proper suffixes of the words in A,
- $Long(A)$, the longest word in A.

For a word w we denote by $w_{(i)}$ the ith symbol in w, provided $1 \leq i \leq |w|$, or ε, otherwise.

Our algorithm has two distinct phases: one in which we find the unique component which is to become active and the other one in which a t-leftmost reduction is performed by this component. The former phase is based on the algorithm proposed in

[Aho and Corasick 1975] for solving the multiple pattern matching problem. The only difference is that the searching process is stopped as soon as a matching pattern has been found in the text. The text is the sentential form while the dictionary of matching patterns is formed by the words in the lefthand side of all rules of the given UPAGS.

For the latter phase we define $reduce(x, i, j, y)$ to be the word obtained by a t-leftmost reduction of x in the component i, where y is the lefthand side of the rule in P_i applicable in the leftmost manner to x at the position j. This function is computed by the next procedure.

Algorithm 2 Procedure $reduce(x, i, j, y)$;

```
begin
α := x(1)x(2) ... x(j−|1|)y(1) ... y(|y|−1);
β := y(|y|)x(j+|y|) ... x(|x|), where y → γ ∈ Pi for some γ;
while β ≠ ε do
  if there exists a rule v → δ ∈ Pi such that
        v(|v|) = β(1);
        v(1)v(2) ... v(|v|−1) = α(|α|−|v|+2) ... α(|α|)
 then α := α(1)α(2) ... α(|α|−|v|+1);
      β := δβ(2) ... β(|β|);
    else α := αβ(1);
      β := β(2) ... β(|β|);
  endif
endwhile
reduce := α;
end;
```

We now define the procedure $find\_component(x, i, j, y, OK)$ whose effect is to find the unique component that is able to reduce the sentential form in the leftmost manner. If such a component exists, the procedure returns the ordinal number i of that component, the position j in the sentential form where the rule from P_i with y in its lefthand side can be applied in the leftmost way, and $true$ for the boolean variable OK. Otherwise, it returns $false$ for OK. The procedure is a slight modification of the algorithm proposed in [Aho and Corasick 1975] for solving the multiple pattern matching problem. The matching patterns are the lefthand side of all rules while the text is the sentential form.

Algorithm 3 Procedure $find\_component(x, i, j, s, OK)$;

```
begin
s := ε; j := 1;
find := false;
while j ≤ |x| and not find do
  while sx(j) ∉ Pref(∪_{i=1}^n dom(Pi)) do
    s := Long(PSuf(x) ∩ Pref(∪_{i=1}^n dom(Pi)));
  endwhile
  s := sx(j);
  if s ∈ dom(Pi) for some 1 ≤ i ≤ n
    then find := true;
  endif
endwhile
if j > |x| then OK := false
  else OK := true
endif
end
```

The algorithm runs as follows: one computes the procedure $find\_component$ and once the component, the rule, and the position have been found, the algorithm computes the function $reduce$. Then, this process is resumed. When no reduction is possible anymore, the condition $end\_recognition$ is satisfied and the algorithm ends. We check whether the sentential form is $\$S\$$; in the affirmative case the word is accepted, otherwise it is rejected.

On the other hand, the aforementioned process might enter an infinite cycle in which only renamings of nonterminals are performed. In this case, the word is not accepted, but one needs a criterion to detect infinite loops in the derivation process. To this aim, whenever a new component is to be activated, we store the numbers of nonterminal and terminal occurrences in the sentential form, respectively, and check whether or not at least one of these two numbers has been modified in the reduction process performed by the function $reduce$.

If none of them has been modified for at least $\frac{(\mathrm{card}(N))^2}{2}$ applications of the function $reduce$, then we infer that the word is not accepted. Let us argue that this number of applications suffices for our decision. To this aim, let us denote by m and t the numbers of nonterminal and terminal occurrences in the sentential form, respectively, and by q the number of nonterminals that occur in the sentential form just before entering the program segment in which m and t will remain unchanged for a number of applications of $reduce$. Due to the working strategy of the system, it is easy to infer that if the function $reduce$ has been applied for more than $q(\mathrm{card}(N) - q) + \frac{q(q-1)}{2}$ times without modifying m or t, then the sentential form would not lead to $\$S\$$ forever.

The main algorithm is listed bellow. Let us assume that we have been given a UPAGS $\Gamma = (N, T, S, P_1, P_2, \ldots, P_n, \$)$ and the word $x \in T^*$. The algorithm outputs **YES** or **NO** if and only if x belongs or not to $Acc(\Gamma, left)$, respectively.

Algorithm 4 begin
  $k := 1$;
  $end\_recognition := false$;
  repeat
    $m := |x|_N$; $t := |x|_T$;
    $find\_component(x, i, j, s, OK)$;
    if OK then $x := reduce(x, i, j, s)$
      else $end\_recognition := true$
    endif
    if $m = |x|_N$ and $t = |x|_T$ then $inc(k)$
      else $k := 1$
    endif
  until $end\_recognition$ or $k = \frac{(\mathrm{card}(N))^2}{2}$
  if $end\_recognition$ and $x = \$S\$$ then $output$ **YES**
    else $output$ **NO**
  endif
  end.

The correctness of the algorithm follows immediately from Proposition 4, the prefix condition, and the considerations from above. We shall briefly discuss the complexity of this algorithm.

Procedure $find\_component$ preprocesses the pattern dictionary, only once, in time $O(d\ \log c)$ and searches the sentential form in time $O(n\ \log c)$, where d is the total size of the lefthand side of all rules of the system, n is the length of the current sentential form, and c is the number of symbols that occur in patterns, namely the cardinality of

$N \cup T$. Let

$$r = \max(\{\frac{|\beta|}{|(\alpha)_T|} \mid \alpha \to \beta \in \bigcup_{i=1}^{n} P_i, \alpha \in (N \cup \{\$\})^* T^+\} \cup \{1\}),$$

$$q = \min\{|\alpha| - 1 \mid \alpha \to \beta \in \bigcup_{i=1}^{n} P_i, \alpha \in (N \cup \{\$\})^+, |\alpha| \geq 2\}.$$

It is not hard to observe that the repeat...until loop is performed $O(|x| \cdot \frac{(\mathrm{card}(N))^2}{2} + \frac{r|x|}{q})$ times provided *reduce* computes $O(1)$ reductions per application.

5 Final remarks

We have discussed the effect of UP-conditions applied to cooperating distributed grammar systems working in the t-mode. As it was expected, the accepting capacity of global UPAGS is the same as the computational power of RC-uniquely parsable grammars that is the class of deterministic context-free languages.

Local UPAGS can express several natural language phenomena, while staying computationally tractable. More precisely, by the two examples, global UPAGS are able to capture some features of natural languages as *multiple agreements* [see Example 1], *marked duplication* [see Example 2] as well as *cross-serial dependencies* (the language $\{a^n b^m c^n d^m \mid n, m \geq 1\}$ can be accepted by a local UPAGS). By the recognition algorithm these mechanisms are polynomially parsable. We do not know whether all the languages accepted by local UPAGS are semiliniar. All these properties seem to advocate that local UPAGS might well have a good level of formal power needed in natural language processing, being inside the *mildly context sensitive* formalisms [Joshi 1985].

A natural direction of further work may consider these formalisms as a syntactic backbone upon which other formalisms of semantical structure can be grafted.

Acknowledgements

The work of the second author was supported by the Dirección General de Enseñanza Superior e Investigación Cientifica, SB 97–00110508.

References

[Aho and Corasick 1975] Aho, A. V., Corasick, M. J.: "Efficient string matching: an aid to bibliographic search"; Commun. ACM 18 (1975), 333–340.

[Atanasiu and Mitrana 1989] Atanasiu, A., Mitrana, V.: "The modular grammars"; Internat. J. Comp. Math. 30 (1989), 17–35

[Baker 1974] Baker, B. S.: "Non-context-free grammars generating context-free languages"; Inform. Control 24 (1974), 231–246.

[Csuhaj-Varju and Dassow 1990] Csuhaj-Varju, E., Dassow, J.: "On cooperating distributed grammar systems"; J. Inform. Process. Cybern., EIK, 26 (1990), 49 – 63.

[Csuhaj-Varju et al. 1994] Csuhaj-Varju, E., Dassow, J., Kelemen, J., Păun, Gh.: "Grammar Systems"; Gordon and Breach (1993).

[Dassow and Mitrana 1999] Dassow, J., Mitrana, V.: "Stack cooperation in multi-stack pushdown automata"; J. Comput. System Sci. 58 (1999), 611–621.

[Dassow et al. 1997] Dassow, J., Păun, Gh., Rozenberg, G.: "Grammar Systems"; [Rozenberg and Salomaa 1997] 155–213.

[Fernau et al. 1996] Fernau, H., Holzer, M., Bordihn, H.: "Accepting multi-agent systems: the case of cooperating grammar systems"; Computers and Artificial Intelligence 15, 2-3 (1996), 123–139.

[Fernau and Holzer 1996] Fernau, H., Holzer, M.: "Accepting multi-agent systems II"; Acta Cybernetica 12 (1996), 361–379.

[Harrison 1978] Harrison, M.: "Introduction to Formal Language Theory"; Addison-Wesley Publ. Co. (1978)

[Joshi 1985] Joshi, A.: "How much context-sensitivity is necessary for characterizing structural descriptions - Tree Adjoining Grammars"; Natural Language Processing - Theoretical Computational and Psychological Perspective, Cambridge University Press, New York (1985).

[Meersman and Rozenberg 1978] Meersman, R., Rozenberg, G.: "Cooperating grammar systems"; Proc. MFCS '78 Symp., LNCS 64, Springer-Verlag (1978) 364 – 374.

[Mihalache and Mitrana 1997] Mihalache, V., Mitrana, V.: "Deterministic cooperating distributed grammar systems"; New Trends in Formal Languages. Control, Cooperation, Combinatorics. LNCS 1218, Springer-Verlag (1997), 137–149.

[Morita et al. 1997] Morita, K., Nishihara, N., Yamamoto, Y., Zhang, Z.: "A hierarchy of uniquely parsable grammars and deterministic acceptors"; Acta Informatica 34 (1997), 389–410.

[Nii 1989] Nii, P. H.: "Blackboard systems"; The Handbook of AI, vol. 4, Addison-Wesley (1989).

[Rozenberg and Salomaa 1997] Rozenberg, G., Salomaa A.(eds.): "Handbook of Formal Languages"; Springer-Verlag, 3 vol. (1997).