Acta Cybernetica 12 (1996) 217-233.

On representing RE languages by one-sided internal contextual languages*

A. Ehrenfeucht † A. Mateescu ‡ Gh. Păun ‡
G. Rozenberg † § A. Salomaa ‡

Abstract

In this paper we prove that each recursively enumerable language L can be written in the form $L = cut_d(L' \cap R)$, where L' is a language generated by a one-sided internal contextual grammar with context-free choice, R is a regular language, and cut_d is the operation which removes the prefix bounded by the special symbol d, which appears exactly once in the strings for which cut_d is defined.

However, the context-free choice sets are always deterministic linear languages of a very simple form. Similar representations can be obtained using one-sided contextual grammars with finite choice and with erased or with erasing contexts.

Keywords. Formal languages, contextual grammars, recursively enumerable languages.

1 Introduction

In [3] it is proved that each recursively enumerable language L can be written in the form $L = cut_d(L' \cap R)$, where L' is a finite choice internal contextual language, R is a regular language, and cut_d is the operation which removes the prefix bounded by the special symbol d, which appears exactly once in the strings for which cut_d is defined. It is also asked in [3] whether or not the language L' can be generated by an internal contextual grammar with one-sided contexts only, and it is conjectured that the answer is negative. We prove that each recursively enumerable language L can be written in the form $L = cut_d(L' \cap R)$, where L' is a language generated by a one-sided internal contextual grammar with context-free choice, R is a regular

*Research supported by the Academy of Finland, project 11281, and the ESPRIT Basic Research Working Group ASMICS II. All correspondence to Alexandru Mateescu.

†Department of Computer science, University of Colorado at Boulder, Boulder, CO 80309, U.S.A.

‡Academy of Finland and University of Turku, Department of Mathematics, FIN-20014 Turku, FINLAND

§Leiden University, Department of Computer Science, P.O.Box 9512, NL-2300 RA Leiden, THE NETHERLANDS

language, and cut_d is the operation which removes the prefix bounded by the special symbol d, which appears exactly once in the strings for which cut_d is defined.

An *internal contextual* (*ic*, for short) grammar (as introduced in [9], as a counterpart of external contextual grammars in [7]), consists of an alphabet, a finite set of starting strings (axioms), and a finite set of context adjoining productions of the form $(C, u\$v)$, where C is a finite set of strings and u, v are strings over the given alphabet. For each $x \in C$ we can assume that there is a rewriting rule $x \to uxv$ (the context (u, v) is adjoined to x). When all productions are of the form $(C, \$v)$, then we say that the grammar is a one-sided one. All the strings obtained by finitely many adjoinings, starting from axioms, constitute the language generated by the grammar. When all sets C are of a given type F, we say that the grammar has F-choice (or F-selection).

The family of languages generated by a (one-sided) ic grammar with finite choice includes strictly the family of regular languages, is incomparable with each family intermediate between those of linear and of context-free languages and is strictly included in the family of context-sensitive languages [8], [10].

In [3] it is proved that every recursively enumerable (RE, for short) language L can be written as $L = cut_d(L' \cap R)$, where L' is a finite choice ic language, R is a regular language and cut_d is the operation which maps $x_1 d x_2$ into x_2, providing $x_1 x_2$ contains no occurrence of d. The last section of [3] asks wheter or not L' above can be a one-sided ic language. In [10] it is shown that the restriction to one-sided contexts decreases strictly the power of ic grammars, for all types of selection, which leads to the conjecture in [3] that the answer to this problem is negative. However, all context-free languages L can be written as $L = h(L' \cap R)$, for h a week coding, L' a finite choice one-sided ic language, and R a regular set. Moreover, there are non-context-free languages which can be represented in this way.

Here we contribute to this question by proving that for each RE language L there is a one-sided ic language L' *with context-free selection* and a regular language R such that $L = cut_d(L' \cap R)$. Therefore, we pay the price of using a one-sided ic language in the representation in [3] by involving context-free languages as choice sets. In fact, the used context-free languages are of three particular types, namely finite, deterministic linear, or of the form $\{z\}L_2^+$, where z is a string of length two and L_2 is a deterministic linear language. Further arguments supporting the conjecture in [3] are also discussed. It is shown that representations as above can be obtained when using grammars with finite selection, providing we also use erased or erasing contexts, in the sense of [10].

The proof of our results makes use of the one-sided normal form for context-sensitive grammars, [11].

2 Formal language prerequisites

We refer to [12] for basic formal language notions and results we use here, and we specify only some notations.

For an alphabet V, V^* is the free monoid generated by V, λ is the empty string,

$| x |$ is the length of $x \in V^*$, $| x |_a$ is the number of occurrences of the symbol a in x, and $V^+ = V^* - \{\lambda\}$. A morphism $h : V^* \to U^*$ is called *coding* if $h(a) \in U$ for all $a \in V$ and *week coding* if $h(a) \in U \cup \{\lambda\}$ for all $a \in V$.

The *left quotient* of a language L_2 with respect to a language L_1 is:

$$L_1 \backslash L_2 = \{w \mid uw \in L_2 \text{ for some } u \in L_1\}.$$

For the alphabet $V_n = \{a_1, b_1, \ldots, a_n, b_n\}$, the *Dyck language* D_n is defined as the smallest set $E \subseteq V_n^*$ such that:

1) $\lambda \in E$,

2) if $x, y \in E$, then $xy \in E$,

3) if $x \in E$, then $x_1 a_i b_i x_2 \in E$ for all $1 \leq i \leq n$ and $x_1 x_2 \in V_n^*$ such that $x = x_1 x_2$.

A Chomsky grammar is written in the form $G = (N, T, S, P)$, where N is the nonterminal alphabet, T is the terminal alphabet, $S \in N$ is the axiom, and P is the set of productions. The families of finite, regular, linear, context-free, context-sensitive, recursively enumerable and of arbitrary languages are denoted by FIN, REG, LIN, CF, CS, RE, ARB, respectively.

3 Internal contextual grammars

Let F be a family of languages. An *ic grammar* (with F choice) is a triple

$$G = (V, M, P)$$

where V is an alphabet, M is a finite set of strings over V, and P is a finite set of pairs $(C, u\$v)$, where $C \in F$, u, v are strings over V and \$ is a special symbol not in V.

The elements of M are called *axioms*, those of P are called *productions*; for a production $\pi = (C, u\$v)$, C is called the *selector* and (u, v) the *context* of π.

For any ic grammar $G = (V, M, P)$ and $x, y \in V^*$, we write $x \Rightarrow y$ iff $x = x_1 z x_2$, $y = x_1 u z v x_2$ and $(C, u\$v)$ is a production of P with $z \in C$. Denoting by $\Rightarrow^*$ the reflexive and transitive closure of $\Rightarrow$, the language generated by G is defined by:

$$L(G) = \{y \in V^* \mid x \Rightarrow^* y \text{ for some } x \in M\}.$$

If all productions of P are of the form $(C, \$v)$, then we say that G is a *one-sided* ic grammar.

We denote by $IC(F)$ the family of languages generated by ic grammars with F-choice and by $1IC(F)$ the family of languages generated by one-sided ic grammars with F choice.

Proofs of the following results can be found in [8], [9], [10], [3] :

1) $REG \subset 1IC(FIN) \subset IC(CS) \subset CS$

2) $IC(FIN) \subset IC(REG) \subset IC(CF) \subset IC(CS) \subset IC(RE)$

3) for all F containing the finite languages, $1IC(F)$ and $IC(F)$ are incomparable with each family F' such that $LIN \subseteq F' \subseteq CF$ (there are linear languages not in $IC(ARB)$ and there are non-context-free languages in $1IC(FIN)$).

For instance, the linear language

$$L = a^+ \cup \{a^n b^n \mid n \geq 1\}$$

is not in $IC(ARB)$ [8]. Consider also the ic grammar

$$G = (\{a, b, c, d, e\}, \{a\}, P),$$

$$P = \{(a, \$bc), (a, \$cb), (cbbc, \$b), (bccb, \$c), (cbb, \$d), (bcc, \$e)\}.$$

(Usually we write the singleton languages $\{z\}$ without parenthesis.)

In [3] it is proved that

$$L(G) \cap a(cbb)^+(de)^+ = \{a(cbb)^n(de)^m \mid m \geq 1, n \geq 4^m + 2(4^{m-1} - 1)/3, n \text{ even}\}.$$

which implies that $L(G)$ is not a context-free language.

4 Representing RE languages using ic languages

For an alphabet V and a symbol $d \notin V$, we define the operation

$$cut_d : V^* d V^* \to V^*$$

$$cut_d(x_1 d x_2) = x_2, x_1, x_2 \in V^*.$$

The main result in [3] is

Theorem 1 Every language $L \in RE$ can be written in the form $L = cut_d(L' \cap R)$, for $L' \in IC(FIN)$, $R \in REG$.

Because we shall use here a similar idea, we recall the construction in [3] :

Take $L \subseteq V^*$, $L \in R\dot{E}$, and a type 0 grammar for L, $G = (N, V, S, P)$. Denote

$$lhs(P) = \{u \in (N \cup V)^* \mid u \to v \in P\}$$

(the left hand sides of rules in P) and consider the new symbols $[,], \vdash, \#$. We construct the ic grammar

$$G = (N \cup V \cup \{[,], \vdash, \#\}, \{S\#\}, P')$$

with P' containing the following productions :

1) $(u, [\$]v)$, for each $u \to v \in P$,

2) $(\alpha[u], \vdash \$\alpha)$, $\alpha \in N \cup V, u \in lhs(P)$,

3) $(\alpha \vdash \beta, \vdash \$\alpha)$, for $\alpha, \beta \in N \cup V$,

4) $(a\#, \vdash \$a)$, for $a \in V$.

Consider also the regular language

$$R = (\{[u] \mid u \in lhs(P)\} \cup \{\vdash \alpha \mid \alpha \in N \cup V\})^* \# V^*.$$

Then

$$cut_{\#}(L(G') \cap R) = L(G).$$

The symbols [,], $\vdash$ are called *killers* ; a pair [ ] kills all symbols bracketed by [and], whereas $\vdash$ kills the symbol to the right of it. The intersection with R ensures that rules of type 1 are applied to alive symbols only, killing the substring u of the current string and introducing the alive string v. Rules of type 2 and 3 move alive symbols from the left to the right, crossing over dead symbols. By rules of type 4, the alive terminal symbols can be transported to the right of #. Eventually a string of the form $w\#z$ is obtained, with w containing only killers and dead symbols and $z \in L(G)$.

The main result of the present paper is:

Theorem 2 Every language $L \in RE$ can be written in the form $L = cut_d(L' \cap R)$, with $L' \in 1IC(CF)$ and $R \in REG$.

Proof Let us recall a result from [11]. A (grammatical) *transformation* (often also called a *rewriting system*) is a triple $\tau = (N, T, P)$, where N is a nonterminal alphabet, T is a terminal alphabet and P is a finite set of rewriting rules over $N \cup T$. For a language $L \subseteq N^*$ we define

$$\tau(L) = \{x \in T^* \mid y \Rightarrow^* x \text{ for some } y \in L\}.$$

According to Theorem 3 and the remarks following it in [11] (see also [6]), each language $L \in CS$, $L \subseteq T^*$, can be written in the form $L = \tau(L_0)$, for a regular language $L_0 \subseteq N^*$ and $\tau = (N, T, P)$ a transformation with the productions in P of the forms

$$A \to B, AB \to AC, A \to a, \text{ for } A, B, C \in N, a \in T.$$

Note that τ contains either context-free rules or left-context rules $AB \to AC$ (no one of them increasing the length of the current string).

Let us now take a language $L \in RE$, $L \subseteq T^*$. There are two new symbols $b, c \notin T$, and a language $L' \subseteq b^*cL$, $L' \in CS$, such that for every $w \in L$ there is a string b^icw in L'. We denote $T' = T \cup \{b, c\}$. For this language L', consider $L_0 \in REG$, and τ as above, $\tau = (N_0, T', P)$, $L_0 \subseteq N_0^*$, such that $N_0 \cap T' = \emptyset$ and $L' = \tau(L_0)$. Take a grammar $G_0 = (N_1, N_0, X_0, P_0)$ with $N_1 \cap N_0 = \emptyset$, $N_1 \cap T' = \emptyset$, generating L_0, with $N_1 = N_1' \cup \{X_f\}$ and with the rules in P_0 of the forms

a) $X_1 \to X_2 A$, for $X_1 \in N_1', C_2 \in N_1, A \in N_0$,

b) $X_f \to \lambda$.

(Such a grammar always exists for a regular language: take a left-regular grammar and replace each terminal rule $X \to A$ by $X \to X_f A$, then add the rule $X_f \to \lambda$.)

We construct the ic grammar

$$G = (W, M, P')$$

with $W = N_1 \cup N_0 \cup T' \cup \{], d\}$, for d a new symbol, $M = \{X_0\}$ and P' contains the following productions:

1) $(X_1, \$]X_2 A)$, for $X_1 \to X_2 A \in P_0, X_1 \in N_1', X_2 \in N_1, A \in N_0$,

2) $(X_f, \$])$,

3) $(A, \$]\alpha)$, for $A \to \alpha \in P, A \in N_0, \alpha \in N_0 \cup T'$,

4) $(AB, \$]]AC)$, for $AB \to AC \in P, A, B, C \in N_0$,

5) $(\{\alpha x]^{|x|} \mid x \in (N_0 \cup T')^+\}, \$]\alpha)$, for $\alpha \in N_0 \cup T'$,

6) $(X_f]\{x]^{|x|} \mid x \in (N_0 \cup T')^+\}^+ b^* c, \$d)$

Consider also the regular language

$$R = \{X] \mid X \in N_1'\}^+ \{X_f]\}(N_0 \cup T' \cup \{]\})^* dT^*.$$

We claim that:

$$L = cut_d(L(G) \cap R) \qquad (*)$$

Note that only the productions in groups 5,6 contain selectors which are not singleton languages.

Assume $N_0 \cup T' = \{\alpha_1, \ldots, \alpha_n\}$ and consider the Dyck language D_n. We define the coding φ by $\varphi(a_i) = \alpha_i$ $\varphi(b_i) =]$, for $1 \leq i \leq n$.

The intuition behind the previous construction is the following. The symbol] is a *killer*. Each occurrence of] kills a symbol α in $N_0 \cup T'$ according to the following rules:

1) if $x = x_1 \alpha] x_2$, for $\alpha \in N_1$, then the specified occurrence of α is killed by the specified occurrence of],

2) if $x = x_1 \alpha] x_2]^{|x_2|}] x_3$, for $\alpha \in N_0 \cup T'$ and $x_2 \in (N_0 \cup T')^*$, then α is killed by the occurrence of] in front of x_3.

The productions of type 1,2 in P' produce a string in G_0, together with a sequence of dead symbols and killers. The productions of type 3,4 simulate corresponding rules in τ. The productions of type 5 move alive symbols from left to the right, across dead symbols and killers. This is useful both for preparing substrings

AB for productions of type 4 and for transporting to the right alive copies of terminals. In order to obtain a string in R we must use exactly once the production of type 6. This checks whether the pairs $X]$, $X \in N_1$, appears in the left of symbols used when simulating the work of τ and that all symbols in the left of d are either killers or dead symbols (hence the derivation in τ is terminal), or alive b and c.

The inclusion $L \subseteq cut_d(L(G) \cap R)$ can be easily proved. Namely, take a derivation in the grammar G_0,

$$X_0 \Rightarrow X_1A_1 \Rightarrow X_2A_2A_1 \Rightarrow \ldots \Rightarrow X_kA_k \ldots A_2A_1 \Rightarrow A_k \ldots A_2A_1$$

According to the form of productions of G_0, $X_k = X_f$ and $X_i \neq X_f$ for all $1 \leq i \leq k-1$. Consider also a derivation in τ starting from $A_k \ldots A_2A_1$.

$$w_0 = A_k \ldots A_2A_1 \Rightarrow w_1 \Rightarrow w_2 \Rightarrow \ldots \Rightarrow w_s = b^icw,$$

for some $w \in T^*, w \in L, i \geq 0$.

The derivation

$$X_0 \Rightarrow X_0]X_1A_1 \Rightarrow X_0]X_1]X_2A_2A_1 \Rightarrow \ldots$$

$$\ldots \Rightarrow X_0]X_1] \ldots X_{k-1}]X_fA_k \ldots A_2A_1 \Rightarrow X_0]X_1] \ldots X_{k-1}]X_f]A_k \ldots A_2A_1$$

is obviously possible in G.

For a string z generated by G, let us denote by alive (z) the string obtained by erasing the dead symbols and the killers from z. Then

$$\text{alive } (X_0]X_1] \ldots X_{k-1}]X_f]A_k \ldots A_2A_1) = A_k \ldots A_2A_1 = w_0$$

and

$$X_0]X_1] \ldots X_{k-1}]X_f]A_k \ldots A_2A_1 \Rightarrow X_0]X_1] \ldots X_f]w_1'$$

can be obtained in G with alive $(X_0]X_1] \ldots X_f]w_1') = w_1$. (A rule of type 3, 4 is in P', corresponding to the rule used in$w_0 \Rightarrow w_1$.) Now, using rules of type 5, all the alive symbols in w_1' can be moved to the right, thus obtaining

$$X_0]X_1] \ldots X_f]w_1' \Rightarrow^* X_0]X_1] \ldots X_f]w_1''w_1.$$

The process can be iterated, each step $w_i \Rightarrow w_{i+1}$ can be simulated in G and after such a step the alive symbols can be transported to the right. Finally we obtain a string $z = X_0]X_1] \ldots X_f]w_s''w_s$, hence with $alive(z) = w_s$. The rule of type 6 is now applicable and we get a string in R ended with b^icdw. Using the operation cut_d we obtain the string $w \in L$.

Conversely, let us take a string $w \in cut_d(L(G) \cap R)$.

There is a successful derivation in G,

$$\delta : X_0 \Rightarrow^* z_1X_f]z_2dw,$$

for some $z_1 \in \{X] \mid X \in N_1\}^+$, $z_2 \in (N_0 \cup T' \cup \{]\})^*$, $w \in T^*$ (this is the form of strings in R).

Assertion 1 For every successful derivation δ as above there is a successful derivation of the form

$$\delta' : X_0 \Rightarrow^* z_1 X_f]w_1 \Rightarrow z_1 X_f]z_2 dw,$$

that is, a derivation with the rules of types 1,2 used before using any rule of other types.

Indeed, every derivation in G starts by a rule of type 1, $N_1 \cap (N_0 \cup T') = \emptyset$, and the symbols in N_1 do not appear in rules of type 5 (hence they cannot be moved). Consequently, the rules of types 1,2 do not interfere with rules of other types, if we have

$$X_0] \dots X_j] X_{j+1} A_{j+1} \dots A_1 \Rightarrow^* X_0] \dots X_j] X_{j+1} u \Rightarrow X_0] \dots X_j] X_{j+1}] X_{j+2} A_{j+2} u$$

then we may change the order of using rules, thus producing

$$\begin{aligned} & X_0] \dots X_j] X_{j+1} A_{j+1} \dots A_1 \Rightarrow \\ \Rightarrow\ & X_0] \dots X_j] X_{j+1}] X_{j+2} A_{j+2} A_{j+1} \dots A_1 \Rightarrow^* \\ \Rightarrow^*\ & X_0] \dots X_j] X_{j+1}] X_{j+2} A_{j+2} u \end{aligned}$$

By induction on the length of z_1 we have the assertion.

Assertion 2 If in a successful derivation δ' as above we use further rules of types 1,2 for deriving the prefix $z_1 X_f]$, then we obtain a derivation δ'' which is not successful.

Take $z = X_0] \dots X_k] X_f]$. If we use a production $(X_i, \$] Y A)$, then we obtain $X_0] \dots X_{i-1}] X_i] Y A] X_{i+1}] \dots X_f]$ and A, as well as, any symbol in $N_0 \cup T'$ derived from A by rules of types 3,4 cannot go to the right of $X_f]$, the obtained string will not be in R, the derivation is not a successful one. If we use $(X_f, \$])$, then we get $X_0] \dots X_k] X_f]]$ and again the form of strings in R is contradicted.

Consequently, in view of Assertions 1 and 2 we can consider from now on only derivations in G of the form $Z_1 X_f]w_1 \Rightarrow^* Z_1 X_f] Z_2 dw$ using productions of types 3-6 for deriving w_1. Thus we shall discuss only derivations $x \Rightarrow^* y$, for $x \in (N_0 \cup T' \cup \{]\})^*$. We call such a derivation successful when it is a part of a successful complete derivation in G.

Assertion 3 If in a derivation $u \Rightarrow v$ in G we use a production of type 3, $(A, \$]\alpha)$, $\alpha \in N_0 \cup T'$, for an occurrence of A which is alive in u, then A will be dead in v and the newly introduced occurrence of α is alive in v.

Indeed, if $u = u_1 A u_2$ and u cannot be written in the form $u = u_1 A u_2'] u_2''$ for $u_2' \in \varphi(D_n)$, then $u = u_1 A] \alpha u_2$ and we cannot have $u = u_1 A] \alpha v_2'] v_2''$ for some $v_2' \in \varphi(D_n)$ (otherwise $u = u_1 A v_2'] v_2''$), hence α is alive in v.

Assertion 4 If in a derivation $u \Rightarrow v$ in G we use a production of type 4, $(AB, \$]]AC)$, $A, B, C \in N_0$ for alive occurrences of A, B in u, then A, B are dead in v and A, C are alive in v.

Indeed, if $u = u_1ABu_2$ and u cannot be written in the form $u = u_1ABu_2']]u_2''$, for $u_2' \in \varphi(D_n)$, or in the form $u = u_1ABu_2']u_2'']u_2'''$, for $u_2', u_2'' \in \varphi(D_n)$, then $v = u_1AB]]ACu_2$ and we cannot have $v = u_1AB]]ACv_2']]v_2''$ for some $v_2' \in \varphi(D_n)$ (otherwise $u = u_1ABv_2']]v_2''$) or $v = u_1AB]]ACv_2']v_2'']v_2'''$, for $v_2', v_2'' \in \varphi(D_n)$ (otherwise $u = u_1ABv_2']v_2'']v_2''']$). Consequently, A and C are alive in V.

Assertion 5 If in a derivation $u \Rightarrow v$ in G we use a production of type 3, $(A, \$]\alpha)$, $\alpha \in N_0 \cup T'$, for an occurrence of A which is dead in v, then both A and α are dead in v.

Take $u = u_1Au_2]u_3$ for $u_2 \in \varphi(D_n)$. Then $v = u_1A]\alpha u_2]u_3$ and, clearly, both A and α are dead symbols in v.

Assertion 6 If in a derivation $u \Rightarrow v$ in G we use a production of type 4, $(AB, \$]]AC)$, $A, B, C \in N_0$, for dead A, B symbols in u, then in v all involved occurrences of A, B, C are dead.

Indeed, for $u = u_1ABu_2]]u_3$, with $u_2 \in \varphi(D_n)$, we get $v = u_1AB]]ACu_2]]u_3$, and for $u = u_1ABu_2]u_3]u_4$, with $u_2, u_3 \in \varphi(D_n)$, we get $v = u_1AB]]ACu_2]u_3]u_4$. Clearly, the specified occurrences of A, B, C are dead in v.

Assertion 7 If in a derivation $u \Rightarrow v$ in G we use a production of type 4, $(AB, \$]]AC)$, for A, B such that A is alive and B is dead in u, then the new occurrence of A will be alive in v, whereas the old occurrence of A, the occurrence of B and the new occurrence of C are dead in v.

If $u = u_1ABu_2]u_3$, for $u_2 \in \varphi(D_n)$, but we do not have $u = u_1ABu_2]]u_3$, for $u_2 \in \varphi(D_n)$, or $u = u_1ABu_2]u_3]u_4$, for $u_2, u_3 \in \varphi(D_n)$,then $v = u_1AB]]ACu_2]u_3$ (hence the first occurrence of A, as well as B and C are dead symbols), but we cannot have $v = u_1AB]]ACv_2]]v_3$, for $v_2 \in \varphi(D_n)$ (otherwise $u = u_1ABu_2]]v_3$ or $v = u_1AB]]ACv_2]v_3]v_4$); consequently, the newly introduced occurrence of A is alive.

(This is the place where the one-sided normal form for context-sensitive grammars is essentially useful; for the rest of the proof a grammar in the Kuroda normal form would suffice.)

As we cannot have A dead and B alive in a substring AB, there are all the cases of applying productions of types 3,4.

Consider now the use of a production of type 5.

Assertion 8 If in a derivation $u \Rightarrow v$ in G we use a production of type 5, that is to some $\alpha x]^{|x|}$ we adjoin $]\alpha, \alpha \in N_0 \cup T'$, and the used α is alive in u, then it will be dead in v and the newly introduced occurrence of α will be alive in v.

If $u = u_1\alpha x]^{|x|}u_2$ and we cannot have $u = u_1\alpha x]^{|x|}u_2']u_2''$ for $u_2' \in \varphi(D_n)$, then $v = u_1\alpha x]^{|x|}]\alpha u_2$ (clearly, the first occurrence of α is dead in v) and we cannot

have $v = u_1\alpha x]^{|x|}]\alpha]v_2]v_4$, for some $v_2 \in \varphi(D_n)$ (otherwise $u = u_1\alpha x]^{|x|}v_2]v_3$); consequently, the new occurrence of α is alive.

Assertion 9 If in a derivation $u \Rightarrow v$ in G we use a production of type 5, namely we replace some $\alpha x]^{|x|}$ by $]\alpha$, $\alpha \in N_0 \cup T'$, such that the used α is dead in u, then it will be dead in v, too, and also the new occurrence of α will be dead in v.

If $u = u\alpha x]^{|x|}]\alpha u_2]u_3$, for $u_2 \in \varphi(D_n)$, then $v = u\alpha x]^{|x|}]\alpha u_2]u_3$, which implies that both specified occurrences of α are dead in v.

As a consequence of Assertions 3-7, we have:

Assertion 10 If in a derivation $u \Rightarrow v$ in G we use a production π of types 3,4, then $alive(u) \Rightarrow alive(v)$ in τ, namely by using the rule in P corresponding to π.

Moreover, from Assertions 8,9 we get:

Assertion 11 If $u \Rightarrow v$ is a derivation in G using a production of type 5, then $alive(u) = alive(v)$.

Consequently, for each (successful) derivation in G,

$$\delta : X_0 \Rightarrow^* z_1X_f]w_1 \Rightarrow z_1X_f]w_2 \Rightarrow \ldots \Rightarrow z_1X_f]w_k$$

the subderivation

$$w_1 \Rightarrow w_2 \Rightarrow \ldots \Rightarrow w_k$$

corresponds to a derivation in τ

$$w_1 = alive(w_1) \Rightarrow alive(w_2) \Rightarrow \ldots \Rightarrow alive(w_k) \qquad (**)$$

Consider now the use of a production 6. It only introduces an occurrence of d (and no killer). Because d is not involved in other productions, in view of the form of strings in R, such a rule can be used exactly once and to the right of the introduced d we must have only symbols in T. Because the selector of the production 6 is bounded by the prefix $X_f]$ and a suffix b^ic, it follows that the whole string to which this rule is applied is of the form

$$z_1X_f]w_k = z_1X_f]u_1b^icu_2$$

for $u_1 \in (N_0 \cup T' \cup \{]\})^*$ such that $alive(u_1) = \lambda$, and $i \geq 0$, whereas $u_2 \in T^*$. Hence we obtain

$$z_1X_f]u_1b^icu_2 \Rightarrow z_1X_f]u_1b^icdu_2.$$

In view of (**), this implies that $b^icu_2 \in \tau(L_0) = L'$ hence $u_2 \in L$. Because in the above writing, $alive(u_1) = \lambda$, if we continue to apply to u_1 productions of types 3,4 or 5, then we have to use in selectors only dead symbols. According to Assertions 5,6 and 9, we obtain new dead symbols only, hence the string will remain in the form $z_1X_f]u_1'b^icdu_2$ with $alive(u_1') = \lambda$. In conclusion, the derivation will ultimately produce a string of the form xdu_2 with $u_2 \in L$. By the operation cut_d we

obtain u_2, hence $cut_d(L(G) \cap R) \subseteq L$, and this completes the proof of the equality (*), hence of Theorem 2. □

Example Here we exemplify the equality (*) from the above proof, for a simple case. Take the regular language

$$L_0 = CA^+B^+$$

generated by

$$G_0 = (\{X_1, X_2, X_3, X_f\}, \{A, B, C\}, X_1,$$
$$\{X_1 \to X_1B, X_1 \to X_2B, X_2 \to X_2A, X_2 \to X_3A, X_3 \to X_fC, X_f \to \lambda\})$$

and the transformation

$$\tau = (\{A, B, C\}, \{a_1, a_2, b, c\}, \{C \to c, A \to a_1, B \to a_2\}).$$

The produced language is $\tau(L_0) = ca_1^+a_2^+$. The associated ic grammar is

$$G = (\{X_1, X_2, X_3, X_f, A, B, C,], a_1, a_2, b, c, d\}, \{X_1\}, P')$$

$$P' = \{(X_1, \$]X_1B), (X_1, \$]X_2B), (X_2, \$]X_2A), (X_2, \$]X_3A), (X_3, \$]X_fC), (X_f, \$])\} \cup$$
$$\cup (C, \$]c), (A, \$]a_1), (B, \$]a_2), \} \cup$$
$$\cup \{(\{\alpha x]^{|x|} \mid x \in \{A, B, C, a_1, a_2, b, c,]\}^+\}, \$]\alpha) \mid \alpha \in \{A, B, C, a_1, a_2, b, c\}\} \cup$$
$$\cup \{(X_f]\{x]^{|x|} \mid x \in \{A, B, C, a_1, a_2, b, c\}^+b^*c, \$d)\}$$

whereas

$$R = \{X_1], X_2], X_3]\}^*\{X_f]\}\{A, B, C, a_1, a_2, b, c,]\}^*d\{a_1, a_2\}^*.$$

For a derivation in G_0

$$X_1 \Rightarrow X_1B \Rightarrow X_2BB \Rightarrow X_2ABB \Rightarrow X_3AABB \Rightarrow X_fCAABB \Rightarrow CAABB$$

followed by a derivation in τ

$$CAABB \Rightarrow cAABB \Rightarrow ca_1ABB \Rightarrow ca_1a_1BB \Rightarrow ca_1a_1a_2B \Rightarrow ca_1a_1a_2a_2,$$

we construct a derivation in G, leading to the same string $ca_1a_1a_2a_2$ as follows:

$$X_1 \Rightarrow X_1]X_1B \Rightarrow X_1]X_1]X_2BB \Rightarrow X_1]X_1]X_2]X_2]X_3AABB \Rightarrow$$
$$\Rightarrow X_1]X_1]X_2]X_2]X_3]X_fCAABB \Rightarrow X_1]X_1]X_2]X_2]X_3]X_f]CAABB$$

(we have simulated the derivation G_0)

$$\Rightarrow^* X_1]X_1]X_2]X_2]X_3]X_f]C]\underline{c}A]\underline{a}_1A]\underline{a}_1B]\underline{a}_2B]\underline{a}_2$$

(we have simulated the derivation τ; all the underlined symbols are alive, the other are dead)

$$\Rightarrow X_1]X_1]X_2]X_2]X_3]X_f]C]\underline{c}A]\underline{a}_1A]\underline{a}_1B]a_2B]]\underline{a}_2\underline{a}_2 \Rightarrow$$

$$\Rightarrow X_1]\dots X_f]C]\underline{c}A]\underline{a}_1A]a_1B]]\underline{a}_1a_2B]]\underline{a}_2\underline{a}_2 \Rightarrow$$
$$\Rightarrow X_1]\dots X_f]C]\underline{c}A]\underline{a}_1A]a_1B]]a_1a_2B]]]\underline{a}_1\underline{a}_2\underline{a}_2 \Rightarrow^*$$
$$\Rightarrow^* X_1]\dots X_f]C]cA]]ca_1A]]]ca_1a_1B]]]]ca_1a_1a_2B]]]]]\underline{c}\underline{a}_1\underline{a}_1\underline{a}_2\underline{a}_2$$

(the alive symbols are separated from the dead ones)

$$\Rightarrow X_1]\dots X_f]C]cA]]\dots ca_1a_1a_2B]]]]]cda_1a_1a_2a_2.$$

This is a string in R. Cutting the prefix bounded by d we get the string $a_1a_1a_2a_2$. □

5 Consequences, variants, comments

Because both the intersection by a regular set and the operation cut_d can be realized, at the same time, by a *gsm*, we obtain

Corollary 1 Every language $L \in RE$ can be written in the form $L = g(L')$, for g a *gsm* and $L' \in 1IC(CF)$.

Moreover, we have

Corollary 2 Every language $L \in RE$, $L \subseteq T^*$, can be written in the form $L = (R'\backslash L') \cap T^*$, for $L' \in 1IC(CF), R' \in REG$.

Proof Construct G and R as in the proof of Theorem 2 and take

$$R' = \{X] \mid X \in N_1'\}^*\{X_f]\}(N_0 \cup T' \cup T' \cup \{]\})^*\{d\}.$$

Then

$$cut_d(L(G) \cap R) = (R'\backslash L(G)) \cap T^*.$$

(We need the intersection with T^* in order to prevent cases when strings $u_1b^icdu_2$, $u_2 \notin T^*$, are produced in G; the intersection with R avoids such cases, but by a left quotient we can get u_2, which is a parasitic string.) □

The use of context-free selections is a very powerful feature. For instance, we have

Theorem 3 Every language $L \in RE, L \subseteq T^*$, can be written in the form $L = h(L' \cap R)$, where h is a morphism, $L' \in 1IC(LIN)$ and $R \in REG$.

Proof It is known [1] that each $L \in RE$ can be written as $L = h_1(L_1 \cap L_2)$, for h_1 a morphism and $L_1, L_2 \in LIN$. For such $L_1, L_2 \subseteq V^*$, we construct the *ic* grammar

$$G = (V \cup \{X_1, X_2, Y_1, Y_2\}, \{X_1\}, P)$$

with P containing the following productions:

1. $(X_1, \$a), a \in V$,
2. $(X_1, \$X_2)$,
3. $(X_1X_2L_1, \$Y_1)$,
4. $(X_1X_2L_2Y_1, \$Y_2)$.

Take also the regular language

$$R = X_1X_2V^*Y_1Y_2$$

and the morphism $h : (V \cup \{X_1, X_2, Y_1, Y_2\})^* \to T^*$ defined by $h(a) = h_1(a)$ for $a \in V$, and $h(X_1) = h(X_2) = h(Y_1) = h(Y_2) = \lambda$.

Then, clearly,

$$h(L(G) \cap R) = h(X_1X_2(L_1 \cap L_2)Y_1Y_2) = h_1(L_1 \cap L_2) = L$$

□

The representation in Theorem 2 cannot be obtained as a consequence of the results in [1], [4], because $LIN - 1IC(CF) \neq \emptyset$.

Note that the main difference between representations in Theorems 2 and 3 is the use of the operation cut_d in Theorem 2 and of an erasing morphism in Theorem 3. Moreover, from Theorem 3 we cannot obtain a consequence as Corollary 2 above.

From the previous representations we get

Theorem 4 The family $1IC(CF)$ is incomparable with each family F such that $LIN \subseteq F \subset RE$, which is closed under

1) left quotient and intersection with regular sets,

or

2) arbitrary morphisms and intersection with regular sets.

Proof We know that $LIN - 1IC(CF) \neq \emptyset$, hence $F - 1IC(CF) \neq \emptyset$ for all F as above. Conversely, $1IC(CF) \subseteq F$, together with the specified closure properties imply $RE \subseteq F$, a contradiction. □

Important families of languages having the properties in Theorem 4 are, for instance, the family of $ETOL$ languages and the family of languages generated by programmed (matrix, controlled, etc.) grammars with λ-rules but without appearance checking. The family $1IC(CF)$ contains languages outside these families. (The same conclusion follows from Theorem 1, with respect to the family $IC(FIN)$.)

Comparing the proof in [3] with the proof of Theorem 2, the main difficulty in the case above arises when changing the place of alive symbols with respect to dead symbols. For instance, instead of productions of type 6, one might try to use finite-choice productions of the form

$$(\alpha x]^{|x|}, \$]\alpha), \alpha \in N_0 \cup T',$$

for finitely many given strings x. Assume, that we have productions of this form for all $|\ x\ | \leq k$, for given k. Take a substring $abb^k]^k]$ (all occurrences of b are dead). Moving the first b, we get $abb^k]^k]b]$, hence we obtain a new dead occurrence of b. In order to move the alive a to the right, we cannot use rules as above. Suppose that we also have a rule

$$(ab, \$]a)$$

If this rule is used for both ab alive, then we get $ab]a$, now with b dead and two alive occurrences of a. If it is used to derive $abb^k]^k]$, we obtain $abb^k]^k] \Rightarrow ab]abb^k]^k]$. The first a is still alive, the second one is dead. Using also the rule $(ab], \$]a)$, we get $ab]]aab^k]^k]$ and the alive a is again at the distance of $k+1$ dead sumbols to the first] to its right. We have obtained nothing. This might be a further argument supporting the conjecture that we cannot represent RE languages starting from languages in $1IC(FIN)$.

The previous difficulty appears because we have substrings of killers, $]^i$, with $i \geq 2$. Such substrings are introduced, for instance, by rules of type 4. If we could "distribute" the killers to the killed symbols, then this difficulty would be avoided. A possible way to do this is to consider *paired contextual grammars*, that is constructs $G = (V, M, P)$, with P containing productions of the form $(z_1, z_2; \$v_1, \$v_2)$, where z_1, z_2, v_1, v_2 are strings over V. For $x, y \in V^+$, we write $x \Rightarrow y$, iff $x = x_1 z_1 z_2 x_2$, $y = x_1 z_1 v_1 z_2 v_2 x_2$, for $(z_1, z_2; \$v_1, \$v_2)$ a production of P. However, such (one-sided) grammars can simulate usual two-sided contextual grammars: for $\pi = (z, u\$v)$, we consider $\pi' = (\lambda, z, ; \$u, \$v)$ and $x \Rightarrow y$ by π if and only if $x \Rightarrow y$ by π'.

6 Erased and erasing contexts

Two extensions of contextual grammars were considered in [10], adding the possibility to remove symbols, not only to adjoin them.

An ic grammar *with erased contexts* is a construct

$$G = (V, M, P_1, P_2)$$

were V, M, P_1 are as in an usual ic grammar and P_2 is a finite set of pairs $(C, u\$v)$, $C \subseteq V^*, u, v \in V^*$. For any $x, y \in V^*$ we write $x \Rightarrow y$ if either

(1) $x = x_1 z x_2, y = x_1 u z v x_2$, for $(C, u\$v) \in P_1, z \in C$,

or

(2) $x = x_1 u z v x_2, y = x_1 z x_2$ and there is $(C, u\$v) \in P_2$ with $z \in C$.

Hence the contexts in P_1 are adjoined, those in P_2 are erased.

When all u as above are equal to λ in all productions of P_1, P_2, we say that G is one sided. The type of selectors C defines the type of selection of G. We denote by $ICD(F)$ the families of languages generated by ic grammars with erased contexts and F-choice; when only one-sided contexts are used, we write $1ICD(F)$.

In [10] it is proved that the extension to erased contexts increases strictly the generative power of ic grammars with F-choice. In view of the following result, this increase is considerable.

A variant of the operation cut_c has been considered in [3], namely, for $c \notin V$,

$$mcut_c : (V \cup \{c\})^*\{c\}V^*$$

defined by

$$mcut_c(x_1 c x_2) = x_2, x_2 \in V^*.$$

(The maximal prefix bounded by c is erased; c appears at least once in the strings for which $mcut_c$ is defined.)

Theorem 5 Every language $L \in RE, L \in T^*$, can be written in the form $L = mcut_c(L' \cap R)$, for $L' \in 1ICD(FIN), R \in REG$.

Proof Take again a left-linear grammar $G_0 = (N_1, N_0, X_0, P_0), N_1 = N_1' \cup \{X_f\}$, and a transformation $\tau = (N_0, T \cup \{b, c\}, P)$, as in the proof of Theorem 2, such that $\tau(L(G_0)) = L' \in CS, L' \subseteq b^*cL$, for $L \in RE$ given. We construct the ic grammar with erased contexts (and finite choice)

$$G' = (W, M, P_1, P_2)$$

where W, M are as in the grammar G constructed in the proof of Theorem 2, P_1 contains all the rules of types 1,2,3,4 in that proof, and

$$P_2 = \{(\lambda, \$A]) \mid A \in N_0\}.$$

Consider also the regular set.

$$R = \{X] \mid X \in N_1'\}^*\{X_f]\}T'^*.$$

Because no erasing of pairs $X], X \in N_1$, is possible, each derivation in G_1 ends by using the symbol X_f, which appears as a separator in R, all Assertions 1-7 (hence also Assertion 10) are still true for the use of rules in P_1. Moreover, the pairs $A]$, $A \in N_0$, can be freely erased, and A erased in this way is a dead symbol. Erasing such pairs, we make possible the use of productions of type 4 (the nonterminals A, B become neighbours) and we remove the useless symbols. More such erasings correspond to a string in $\varphi(D_n)$. Therefore, each derivation in G_0 followed by a derivation in τ can be simulated in G' and, conversely, a derivation in G' which ends by a string in R corresponds to a derivation in G_0 followed by a derivation in τ, in the sense that the produced string will be of the form $z_1 X_f]w, z_1 \in \{X] \mid X \in N_1'\}, w \in \tau(L(G_0))$. Consequently, $z_1 X_f]w = z_1 X_f]b^i cw', w' \in L$, hence using the operation $mcut_c$ we obtain $w' \in L$. □

An ic grammar *with erasing contexts* is a construct $G = (V, M, P_1, P_2)$, where V, M, P_1, P_2 are as in a grammar with erased context. For $x, y \in V^*$ we write $x \Rightarrow y$ if either

(1) $x = x_1zx_2, y = x_1uzvx_2$, for $(C, u\$v) \in P_1, z \in C$,
or
(2) $x = x_1uzvx_2, y = x_1uvx_2$ and there is $(C, u\$v) \in P_2$ with $z \in C$.

Hence the string z braketted by the context (u, v) is erased when $z \in C$ for some $(C, u\$v)$ in P_2.

We denote by $ICG(F), 1ICG(F)$, the families of languages generated by such grammars.

The use of erasing contexts increases strictly the generative power of ic grammars. As for erased contexts, we have

Theorem 6 Every language $L \in RE, L \subseteq T^*$, can be written in the form $L = mcut_c(L' \cap R)$, for $L' \in 1ICG(FIN), R \in REG$.

Proof We simply repeat the proof of Theorem 5 taking instead of P_2 the set

$$P_2' = \{(A], \$) \mid A \in N_0\}.$$

Erasing the context $(\lambda, A])$ when bracketing λ (as in the case of P_2) is the same with erasing $A]$ when bracketed by (λ, λ) (as in P_2'), hence all arguments in the proof of Theorem 5 remain valid. □

Again consequences as in Corollaries 1,2 and in Theorem 4 (point 1) can be obtained for families $1ICD(FIN), 1ICG(FIN)$.

These representations of RE languages remind the representations of linear languages by cancellation operations, as in [2], [5]. However, here we start from families of languages which are incomparable with LIN: again $L = a^+ \cup \{a^nb^n \mid n \geq 1\}$ cannot be generated by an ic grammar with erased or with erasing contexts and with F choice, whichever F is (in order to produce strings a^m with arbitrarly large m we either need a production $(C, a^i\$a^j)$ with $C \cap a^+ \neq \emptyset$ and $i + j \geq 1$ or the possibility to erase the occurrences of b from some a^nb^n; in the first case we can derive strings a^pb^p in L into strings a^qb^p with $q > p$; in the second case we have intermediate steps of derivation when strings a^sb^t with $s \geq n, t \geq n, s > t$, are produced; in both cases we have obtained parasitic strings).

Since the family RE is closed under all the operations involved in the previous proofs, all the representations given above are in fact characterizations of recursively enumerable languages.

References

[1] B. S. Baker, R. V. Book, Reversal-bounded multipushdown machines, *J. Comput. Systems Sci.* 8(1974), 315-332.

[2] F. J. Brandenburg, Concellations in linear context-free languages, Univ. Passau, Fak. fur Math. und Informatik, Report MIP-8904, 1989.

[3] A. Ehrenfeucht, Gh. Păun, G. Rozenberg, On representing recursively enumerable languages by internal contextual languages, Technical Report, Dept. of Comp. Sci., Leiden University, The Netherlands, 94-30, 1994, to appear also in *Th.Comp.Sci.*

[4] M. Latteux, B. Leguy, B. Ratoandromanana, The family of one-counter languages is closed under quotient, *Acta Informatica*, 22(1985), 579-588.

[5] M. Latteux, P. Turakainen, A characterization of recursively enumerable languages, *Acta Informatica*, 28(1990), 179-186.

[6] B. E. Katz, Synchronized left context-sensitive transformations of regular languages, *Nauch. Tekh. Inform.*, Ser. 2, 4(1974).

[7] S. Marcus, Contextual grammars, *Rev. Roum. Math. Pures Appl.*, 14(1969), 1525-1534.

[8] Gh. Păun, Contextual Grammars The Publ. House of the Romanian Academy of Sciences, Bucharest, 1982 (in Romanian).

[9] Gh. Păun, X. M. Nguyen, On the inner contextual grammars, *Rev. Roum. Math. Pures Appl.* 25(1980), 641-651.

[10] Gh. Păun, G. Rozenberg, A. Salomaa, Contextual grammars: erasing, determinism, one-sided contexts, *Developments in Language Theory* (G. Rozenberg, A.Salomaa, Eds.), World Sci., Singapore, 1994, 370-388.

[11] M. Penttonen, One-sided and two-sided contexts in formal grammars, *Inform. Control*, 25(1974), 371-392.

[12] A. Salomaa, Formal Languages, Academic Press, New York, London, 1973.

Received December, 1995