

Regularly Controlled Bidirectional Extended Linear Basic Grammars

Jan Anne Hogendorp*

*Department of Computer Science, University of Twente
P.O. Box 217, 7500 AE Enschede, The Netherlands*

Abstract — We study the concept of bidirectional application of productions – i.e., using a production of a grammar as a reduction too – with respect to regularly controlled extended linear basic (macro) grammars [3], provided with a restricted mode of derivation. So this new grammatical model is in essence equal to the regularly controlled bidirectional context-free grammars of [15] in which the underlying context-free grammar is replaced by an extended linear basic grammar. We establish closure properties of the corresponding family of languages; viz. for the outside-in or OI-mode we obtain a full substitution-closed AFL, and for the inside-out or IO-mode we obtain a full QAFL closed under deterministic substitution. The notion of bidirectionality gives rise to a dramatic increase of generating power; even under minor assumptions the OI [IO] instance of such grammars generate all OI-macro [IO-macro, respectively] languages. Furthermore, in case of free application of productions and reductions we obtain a generating capacity equal to the one of phrase-structure grammars.

1. Introduction

In [15] a new grammatical model has been introduced, the so-called regularly controlled bidirectional grammars, or RCB-grammars. An RCB-grammar is a pair consisting of a context-free grammar G (the underlying grammar), denoted by (V, Σ, P, S) , together with a regular language (the control language) C over the set of rules induced by P . This set of rules consists of the productions from P and of the reductions determined by P ; i.e., if $A \rightarrow \alpha$ is a production in P , then $\alpha \rightarrow A$ is the corresponding reduction.

The main reason to study this type of grammar stems from the concept of NTS-grammars; cf. [15] for background and motivation. An NTS-grammar is a context-free grammar such that each sentential form which can be derived from a nonterminal by applying both productions and reductions can be also derived by means of productions only. The NTS-property has also been defined in [14] for an extension of context-free grammars; viz. for macro grammars [10]. It is therefore also interesting to study macro grammars provided with a regular control language over both productions and the corresponding reductions. Now in general the aspect of bidirectionality gives a dramatic increase in generating power; cf. [4], where it has been shown that for several modes of derivation RCB-grammars are able to generate all recursively enumerable languages. Replacing the context-free grammar in the notion of RCB-grammar by a macro grammar results in a grammar model that can of course generate all recursively enumerable languages too. Therefore we restrict our attention in this paper to RCB-grammars in which the underlying context-free grammar is replaced by a so-called K -extended linear basic (macro) grammars. We motivate the choice of K -extended linear basic grammars by the fact that for some language families K , the corresponding family of languages $LB_m(K)$ is incomparable with the family CFL of context-free languages. This holds for instance if K equals $\emptyset NE$ or FIN – where $\emptyset NE$ equals the family ONE of all languages consisting of one word, together with the language \emptyset , or formally,

* The work of the author has been supported by the Netherlands Organization for Scientific Research (N.W.O.).

$\emptyset NE = ONE \cup \{\emptyset\}$; FIN is the family of all finite languages – and $m = IO$, since in these cases we have $LB = LB_{IO}(K)$. Remember that the family LB of linear basic (macro) languages, introduced in [10], is known to be incomparable with CFL [7]. Actually, we take extended linear basic (m, K) -grammars, introduced in [3], where m varies over the two well-known macro grammar modes of derivation OI (“outside-in”) and IO (“inside-out”). And K is a family of languages, which restricts the possible languages associated with language names in such grammars. Extended linear basic (m, K) -macro grammars or (m, K) -elb grammars are a generalization of the extended linear basic grammars introduced and investigated in [6] and [8]. We provide these (m, K) -elb grammars with bidirectional rewriting and regular control, and the resulting type of grammar will be called *bidirectional (m, REG, K) -elb grammar* or *(m, REG, K) -belb grammar* for short. REG denotes the family of regular (control) languages.

Together with the notion of RCB-grammar we have defined in [15] several modes of derivation which differ from the usual free application of rules. These modes are motivated by the observation in [5] that the family of languages generated by bidirectional context-free grammars (V, Σ, P, S) in which the start symbol S is replaced by some context-free language M over V is equal to the family of recursively enumerable languages. Therefore, to obtain new, interesting language families we have introduced restrictive modes of derivation. In this paper only one of the modes introduced in [15] will be studied within the framework of bidirectional (m, REG, K) -elb grammars. For sake of comparison some attention is also paid to the free mode of application.

The mode of derivation with which a grammar will be provided, determines three different aspects of the derivation process. In other words, each mode of derivation consists of three submodes. First, in a sentential form a string u can be rewritten by productions or reductions only if from the right end of this string u only terminals occur. This submode is called the *right-most string rewriting* submode, denoted by RS . The RS -submode is introduced in [4] as a more precise modeling of bidirectional right-most derivation in context-free grammars, whereas originally in [15] a slightly different variant of this submode has been investigated. Second, if a rule in a control word from the control language is not applicable to the sentential form derived so far, then the derivation process halts, and no string is derived. This submode is called the *block* mode and it is abbreviated by B . The third aspect discriminates between reductions of the form $\alpha \rightarrow A$, where α consists of terminals only, (the so-called terminal reductions) and reductions with at least one nonterminal at the left-hand side. In the *fair* submode, abbreviated by f , only such “non-terminal” or fair reductions are allowed. The name *fair* originates from the idea that such reductions respect the difference between terminals and nonterminals in this grammatical model. So the mode studied here is the $RS/B/f$ -mode. As remarked before, this mode is in fact a slight modification of the $RN/B/f$ -mode introduced in [15]. However, for context-free grammars, in both cases the family of languages generated by the resulting type of RCB-grammar is exactly the family of context-free languages [4,15]. We call an (m, REG, K) -belb grammar provided with the $RS/B/f$ -mode of derivation an (r, f, m, REG, K) -belb grammar. The corresponding family of languages is denoted by $RBLB_{r, f, m}(K)$. (Remember that m is equal to either OI or IO).

The structure of this paper is as follows. In section 2 we recall terminology and notational conventions for macro grammars, linear basic grammars and their generalizations. In Section 3 we introduce regularly controlled bidirectional (m, K) -elb grammars or (m, REG, K) -belb grammars. Then we formally define for these (m, REG, K) -belb grammars the $RS/B/f$ -mode of derivation. The corresponding type of grammar is called (r, f, m, REG, K) -belb grammar. In addition, we introduce c-trees to visualize the structure of sentential forms generated by (r, f, m, REG, K) -

belb grammars. These c-trees are also helpful in the proofs of Section 6. Closure properties of the family $RBLB_{r,f,m}(K)$ of (r,f,m,REG,K) -belb languages are established in Section 4. For both modes OI and IO and under weak assumptions on the family K , it is shown that the family $RBLB_{r,f,m}(K)$ is closed under the regular operations (union, concatenation, and Kleene +). Furthermore, we will prove that if K is a nontrivial family of languages closed under ngsm mappings, then $RBLB_{r,f,OI}(K)$ is a full substitution-closed AFL. Concerning the family $RBLB_{r,f,IO}(K)$, we establish – under appropriate conditions on K – closure under intersection with regular languages and deterministic substitution; hence this family is a full QAFL (in the sense of [3]) closed under deterministic substitution. In Section 5 we discuss the language generating capacity of (r,f,m,REG,K) -belb grammars. We show that the family $RBLB_{r,f,OI}(\emptyset NE)$ is equal to the family OI of OI-macro languages and that the family IO of IO-macro languages is included in the family $RBLB_{r,f,IO}(\emptyset NE)$. In Section 6 (m,REG,K) -belb grammars provided with free application of rules are studied. However, the restriction of allowing only fair reductions is maintained. Then for $m = OI$ and for $m = IO$ the family of languages generated by these so-called (f,m,REG,K) -belb grammars equals the family of recursively enumerable languages. Finally, Section 7 contains some concluding remarks.

2. Preliminaries

Terminology and notational conventions used in this paper follow in most cases standard texts in formal language theory like [13] or [20].

Macro grammars have been introduced by Fischer in [10] as a generalization of context-free grammars. The difference with context-free grammars is that in a macro grammar we associate with each nonterminal a nonnegative number of arguments. We take arguments from the set of terms build up from nonterminals, terminals, and variables. Therefore we can consider a macro grammar as a particular kind of term rewriting system. In order to define macro grammars in a precise way we use the concepts of ranked alphabet, and term over a ranked alphabet.

An *alphabet* is a finite set of symbols. A *ranked alphabet* Δ is an alphabet of which each symbol is provided with a nonnegative integer, called its *rank*. The ranked alphabet Δ is partitioned into sets Δ_i consisting of those symbols with rank equal to i . Thus $\Delta = \cup \Delta_i$ and if $i \neq j$, then $\Delta_i \cap \Delta_j = \emptyset$. Let PC be a set of punctuation symbols, consisting of the left and right parenthesis and the comma symbol. Furthermore, λ denotes the empty string.

Definition 2.1. Let Δ be a ranked alphabet. The set $T(\Delta)$ of *terms* over Δ is the smallest set of strings over $\Delta \cup PC$ such that

- (a) $\Delta_0 \cup \{\lambda\} \subseteq T(\Delta)$;
- (b) if $t_1, t_2 \in T(\Delta)$, then $t_1 t_2 \in T(\Delta)$;
- (c) if $A \in \Delta_n$ and $t_1, \dots, t_n \in T(\Delta)$, then $A(t_1, \dots, t_n) \in T(\Delta)$. □

We will write A instead of $A()$ if A has rank zero, thus if $A \in \Delta_0$.

Definition 2.2. A *macro grammar* G is 5-tuple (Φ, Σ, X, P, S) , where Φ is a ranked alphabet of *nonterminals*, Σ is an alphabet of *terminals*, X is a finite set of *variables*, S is an element of Φ_0 , called the *start symbol*. It is understood that each terminal and variable has rank zero and that the sets Φ , Σ and X are mutually disjoint. The set P consists of *productions* which have the form $A(x_1, \dots, x_n) \rightarrow t$ with $A \in \Phi_n$, the variables x_1, \dots, x_n are distinct elements of X , and t is an element from $T(\Sigma \cup \{x_1, \dots, x_n\} \cup \Phi)$. □

We need the following terminology to define several *modes of derivation* for a macro grammar. A string τ is a *subterm* of a term σ if τ is a term and τ is a substring of σ . A subterm τ of σ occurs at *top level* if there exist subterms τ_1 and τ_2 such that $\sigma = \tau_1 \tau_2$. A term over $\Sigma \cup \Phi$ that is a string over Σ is an *expanded term*.

Definition 2.3. Let $G = (\Phi, \Sigma, X, P, S)$ be a macro grammar and let σ and τ be terms over $\Sigma \cup \Phi$.

Then we write $\sigma \Rightarrow_{OI} \tau$ if

- there is a nonterminal A from Φ_n and terms ξ_1, \dots, ξ_n over $\Sigma \cup \Phi$ such that $A(\xi_1, \dots, \xi_n)$ is a subterm on top level in σ ;
- $A(x_1, \dots, x_n) \rightarrow t$ is a production from P ;
- τ is obtained from σ by replacing the designated term $A(\xi_1, \dots, \xi_n)$ by t' . The term t' is the result of substituting the terms ξ_1, \dots, ξ_n for x_1, \dots, x_n in t , respectively.

The relation \Rightarrow_{OI} on $T(\Sigma \cup \Phi)$ represents the *OI-derivation mode*, which can be considered as expanding macros by outermost calls first.

Secondly, we write $\sigma \Rightarrow_{IO} \tau$ if

- there is a nonterminal A from Φ_n and there are expanded terms ξ_1, \dots, ξ_n over Σ and $A(\xi_1, \dots, \xi_n)$ is a subterm of σ ;
- $A(x_1, \dots, x_n) \rightarrow t$ is a production from P ;
- τ is obtained from σ in the same way as formulated in the definition of \Rightarrow_{OI} .

The relation \Rightarrow_{IO} on $T(\Sigma \cup \Phi)$ represents the *IO-derivation mode*, which can be considered as expanding macros by innermost calls first.

The reflexive and transitive closure of \Rightarrow_{OI} [\Rightarrow_{IO}] is denoted by \Rightarrow_{OI}^* [\Rightarrow_{IO}^* , respectively]. \square

An *OI-macro* [*IO-macro*] *grammar* is a macro grammar provided with the mode of derivation OI [*IO*, respectively]. In the sequel, m denotes a mode of derivation. The language $L_m(G)$ generated by an m -macro grammar $G = (\Phi, \Sigma, X, P, S)$ is the set $\{w \in \Sigma^* \mid S \Rightarrow_m^* w\}$. The sets *OI* and *IO* denote the families of languages generated by OI- and IO-macro grammars, respectively. It is a well-known fact that *OI* and *IO* are incomparable [10]. Both families properly include the family of context-free languages and they are properly included in the family of context-sensitive languages.

In this paper we study (extensions of) a special kind of macro grammar, the so-called *linear basic grammars*. A *basic* term over $\Sigma \cup \Phi$ is a term in which no nonterminal appears within an argument list of another nonterminal, i.e., all macros are non-nested. A *linear basic* term is a basic term in which at most one nonterminal occurs. Then a [*linear*] *basic grammar* is a macro grammar in which the right-hand side of each production is a [*linear*] basic term. As a direct consequence we have that providing linear basic grammars with the OI- and IO-mode of derivation results in two equivalent type of grammars. So we can speak of linear basic languages without specifying the mode of derivation. Furthermore, the family of linear basic languages is properly included in $OI \cap IO$ [10].

According to Fischer [10], we can assume that each production in a linear basic grammar has one of the forms

$$A(x_1, \dots, x_n) \rightarrow B(w_1, \dots, w_k) \quad \text{or} \\ A(x_1, \dots, x_n) \rightarrow w$$

where w, w_1, \dots, w_k are words over $\Sigma \cup \{x_1, \dots, x_n\}$. We refer to this form as the *standard linear form*.

As a generalization of linear basic grammars “extended linear basic grammars” (or “elb grammars”) have been introduced. Starting from [6] where the words w, w_1, \dots, w_k in the standard linear form have been replaced by finite languages over $\Sigma \cup \{x_1, \dots, x_n\}$, via [8] where regular languages have been used instead of finite languages – however, resulting in no additional generating power – the ultimate extension possible in this way of generalizing the concept of linear basic grammar was defined in [3] in which K -elb grammars have been introduced by replacing each word w, w_1, \dots, w_k by a language from a given, arbitrary family of languages K . The precise definition of this latter grammatical model is as follows.

Definition 2.4. Let K be a family of languages. An *extended linear basic K grammar* or *K -elb grammar* is a 6-tuple $(\Phi, \Psi, \Sigma, X, P, S)$ where

- Φ is a ranked alphabet of *nonterminals*,
- Ψ is a ranked alphabet of *language names*,
- Σ is a *terminal alphabet*,
- X is a finite set of *variables*,
- $S \in \Phi_0$ is the *start symbol*,
- P is a finite set of *productions*. Each production has one of the following forms.

$$A(x_1, \dots, x_n) \rightarrow B(\psi_1(\vec{x})), \dots, \psi_k(\vec{x})), \quad (i)$$

where $A \in \Phi_n$ ($n \geq 0$), $B \in \Phi_k$ ($k \geq 1$), and \vec{x} is the abbreviation of x_1, \dots, x_n . Thus $\psi_i \in \Psi_n$ ($1 \leq i \leq k$). If $A = S$, then production (i) is a so-called *initial* production.

$$A(x_1, \dots, x_n) \rightarrow \psi(x_1, \dots, x_n), \quad (ii)$$

where $A \in \Phi_n - \{S\}$ and $\psi \in \Psi_n$.

$$\psi(x_1, \dots, x_n) \rightarrow L_0, \quad (iii)$$

where $\psi \in \Psi_n$ and $L_0 \subseteq (\Sigma \cup \{x_1, \dots, x_n\})^*$ is a language in K .

Moreover, we require that for each language name ψ from Ψ there is exactly one production of the form (iii) in P . □

We apply the same conventions on notation as we did in Definition 2.2, i.e., the list x_1, \dots, x_n consists of distinct elements of X , the sets Φ, Ψ, Σ and X are mutually disjoint, variables and terminals have rank zero, and if A is an element of either Φ_0 or Ψ_0 we write A instead of $A()$.

In Definition 2.4 we demand that $k \geq 1$ in a production of the form (i), whereas in the original definition in [3] $k \geq 0$ is permitted. However, we obtain no loss of generality; cf. Section 3.

Notice that each word w, w_1, \dots, w_k in the standard linear form is replaced by a language name $\psi(x_1, \dots, x_n), \psi_1(x_1, \dots, x_n), \dots, \psi_k(x_1, \dots, x_n)$, respectively. These language names constitute a special ranked alphabet. We associate to each language name ψ a unique production of the form $\psi(x_1, \dots, x_n) \rightarrow L_0$, where L_0 is a language from the family K with $L_0 \subseteq (\Sigma \cup \{x_1, \dots, x_n\})^*$. As in [1, 3] this approach allows us to make a distinction between OI and IO-derivations in a natural way. In particular, this implies that for many instances of the family K , the generating power of K -elb grammars depends on the mode of derivation.

The relation of K -elb grammars with macro grammars can be obtained in a natural way when we treat a K -elb grammar $G = (\Phi, \Psi, \Sigma, X, P, S)$ as a macro grammar G' with a countable (rather than a finite) number of productions. Viz., let G' be the macro grammar

$(\Phi \cup \Psi, \Sigma, X, P', S)$ where the (countable) set P' of productions is determined by G as follows.

- (1) Each production in P of the form 2.4(i) or 2.4(ii) is also in P' .
- (2) For each production $\psi(x_1, \dots, x_n) \rightarrow L_0$ of the form 2.4(iii) in P , P' contains the (countable number of) productions $\psi(x_1, \dots, x_n) \rightarrow w$ for each w in L_0 .

Now we can provide this countable macro grammar G' with either the OI-mode or with the IO-mode of derivation. In this indirect way one can define a K -elb grammar with the OI or IO-mode of derivation. Viz., an (m, K) -elb grammar G is a K -elb grammar provided with the mode m if the corresponding G' is an m -macro grammar with a countable set of productions. The language $L_m(G)$ generated by G is defined by $L_m(G) = L_m(G')$. The set $LB_m(K)$ denotes the family of languages generated by (m, K) -elb grammars.

We illustrate the concepts defined above by the following example. Recall that FIN is the family of finite languages.

Example 2.5. Consider the (m, FIN) -elb grammar G defined by $G = (\Phi, \Psi, \Sigma, X, P, S)$, where $\Phi = \{S, A\}$, $\Psi = \{\psi_0, \psi_1, \psi_2, \psi_3, \psi_4\}$, $X = \{x, y\}$, $\Sigma = \{0, 1\}$ and P consists of the productions

$$\begin{aligned} \pi_0 &= S \rightarrow A(\psi_0, \psi_1), & \pi_4 &= \psi_1 \rightarrow \{0, 1\}, \\ \pi_1 &= A(x, y) \rightarrow A(\psi_2(x, y), \psi_3(x, y)), & \pi_5 &= \psi_2(x, y) \rightarrow \{y\}, \\ \pi_2 &= A(x, y) \rightarrow \psi_4(x, y), & \pi_6 &= \psi_3(x, y) \rightarrow \{yx\}, \\ \pi_3 &= \psi_0 \rightarrow \{0, 1\}, & \pi_7 &= \psi_4(x, y) \rightarrow \{y\}. \end{aligned}$$

One can verify that

$$(i) \quad L_{IO}(G) = \cup \{h_{uv}(L_0) \mid u, v \in \Sigma\},$$

where for each $u, v \in \Sigma$ the length-preserving homomorphism $h_{uv} : \Sigma \rightarrow \Sigma$ is defined by $h_{uv}(0) = u$ and $h_{uv}(1) = v$. The language L_0 equals

$$\{1, 10, 101, 10110, 10110101, 1011010110110, \dots\},$$

which is the set of Fibonacci words over the alphabet $\{0, 1\}$. These Fibonacci words are given by the sequence $f : \mathbb{N} \rightarrow \Sigma^*$ defined by

$$f_0 = 1; \quad f_1 = 10; \quad f_{n+2} = f_{n+1}f_n \quad \text{for each } n \ (n \geq 0).$$

$$(ii) \quad L_{OI}(G) = \sigma(L_0) = \cup \{\{0, 1\}^{F_n} \mid n \geq 1\},$$

where the length-preserving substitution $\sigma : \Sigma \rightarrow 2^\Sigma$ is defined by $\sigma(0) = \Sigma$ and $\sigma(1) = \Sigma$. And F_n is the n th Fibonacci number; i.e., $F_0 = 0$, $F_1 = 1$, $F_{n+2} = F_n + F_{n+1}$ for each $n \ (n \geq 0)$.

In proving (i) we first observe that for all $w_1, w_2 \in \Sigma^*$ we have

$$A(w_1, w_2) \Rightarrow_{IO}^{\pi_1 \pi_6 \pi_5} A(w_2, w_2 w_1) \quad (1)$$

and

$$A(w_1, w_2) \Rightarrow_{IO}^{\pi_3 \pi_7} w_2. \quad (2)$$

These subderivations can be used to prove by induction that

$$A(0, 1) \Rightarrow_{IO}^{d_n} f_n, \quad \text{for all } n \ (n \geq 0), \quad (3)$$

where $d_n = (\pi_1 \pi_6 \pi_5)^n \pi_2 \pi_7$, for all $n \ (n \geq 0)$.

It is straightforward to show that

$$\{w \in \Sigma^* \mid A(0,1) \Rightarrow_{\text{IO}}^* w\} = L_0.$$

Then it follows that for all $u, v \in \Sigma$,

$$\{w \in \Sigma^* \mid A(u,v) \Rightarrow_{\text{IO}}^* w\} = h_{uv}(L_0).$$

Property (ii) easily follows from the former, considering the linear character of G . \square

3. Regularly Controlled Bidirectional (m, K) -elb Grammars

First, we note that in Definition 2.4(i) we required that $k \geq 1$, whereas in the original definition of (m, K) -elb grammars in [3] $k = 0$ is also permitted. However, the restriction to $k \geq 1$ causes no loss of generality, except that in our approach we need that the family K contains at least one nonempty language. This will be proved in Lemma 3.2.

Definition 3.1. An (m, K) -elb grammar $G = (\Phi, \Psi, \Sigma, X, P, S)$, is in *equal length form* if there is a natural number n ($n \geq 0$) such that each nonterminal in Φ is either in Φ_0 and equal to S , or in Φ_n and each language name in Ψ is either in Ψ_0 or in Ψ_n . \square

Lemma 3.2. Let K be a language family that contains a nonempty language L_0 . For each (m, K) -elb grammar G_0 there exists an (m, K) -elb grammar G in equal length form such that $L_m(G) = L_m(G_0)$.

Proof (sketch). Let $G_0 = (\Phi, \Psi, \Sigma, X, P, S)$ be an (m, K) -elb grammar with $X_0 = \{x_1, \dots, x_n\}$. We enlarge the rank of each nonterminal unequal to S in Φ_k , where $k \geq 0$, and of each language name in Ψ_k , where $k \geq 0$, to n by adding $n - k$ dummy arguments. To the resulting alphabet Ψ' we add each language name with zero rank which occurs in an initial production of P . The productions are changed accordingly by introducing two new language names ψ_e in Ψ_n and ψ_z in Ψ_0 , with $\psi_e(\vec{x}) \rightarrow L_0$ and $\psi_z \rightarrow L_0$.

This well-known construction – e.g., cf. [1] – can easily be written out in full detail. \square

Now we introduce regularly controlled bidirectional (m, K) -extended linear basic grammars, or (m, REG, K) -elb grammars, by a tuple (G, C, ϕ) , where $G = (\Phi, \Psi, \Sigma, X, P, S)$ is an (m, K) -elb grammar. The symbol ϕ does not occur in Φ, Ψ, Σ or X . We use ϕ in order to define \bar{P} , the set of reductions corresponding to productions in P . We associate with a production π in P of the form $\psi(\vec{x}) \rightarrow L_0$ – cf. Definition 2.4(iii) – i.e., a (countable) set of productions $\{\psi(\vec{x}) \rightarrow t \mid t \in L_0\}$, a corresponding set of reductions, defined by $\{t \rightarrow \psi(\gamma_1, \dots, \gamma_n) \mid t \in L_0\}$, where γ_i is equal to x_i in case x_i occurs in t , and otherwise γ_i equals the symbol ϕ ($1 \leq i \leq n$). Note that $\psi(\gamma_1, \dots, \gamma_n)$ depends on t . This set of reductions is also denoted by $\bar{\pi}$ or even by $L_0 \rightarrow \psi(\vec{x})$. Note that $L_0 = \emptyset$ implies that both π and $\bar{\pi}$ are empty.

For example, let π equal $\psi(x, y, z) \rightarrow \{axaz, yz\}$. Then π denotes the set $\{\psi(x, y, z) \rightarrow axaz, \psi(x, y, z) \rightarrow yz\}$, and the corresponding set of reductions is $\{axaz \rightarrow \psi(x, \phi, z), yz \rightarrow \psi(\phi, y, z)\}$. Therefore the reduction $\bar{\pi}$ associated with π is denoted by $\{axaz, yz\} \rightarrow \psi(x, y, z)$.

If π is of the form 2.4(i) or 2.4(ii), then $\bar{\pi}$ is defined by the rewriting rule $B(\psi_1(\vec{x}), \dots, \psi_k(\vec{x})) \rightarrow A(\vec{x})$ and $\psi(\vec{x}) \rightarrow A(\vec{x})$, respectively. Finally, \bar{P} is defined by $\bar{P} = \{\bar{\pi} \mid \pi \in P\}$ as usual. Furthermore, for each production π we define $\bar{\pi}$ to be equal to π . An element of $P \cup \bar{P}$ will be called a *rule*.

Notice that (m,K) -elb grammars provided with an arbitrary control language over P have been studied in [1, 3].

Definition 3.3. A *regularly controlled bidirectional (m,K) -elb grammar* or *(m,REG,K) -belb grammar* is a triple (G, C, ϕ) where

- G is an (m,K) -elb grammar $(\Phi, \Psi, \Sigma, X, P, S)$,
- C is a regular language with $C \subseteq (P \cup \bar{P})^*$,
- ϕ is a special symbol not occurring in Φ, Ψ, Σ or X .

We call G the *underlying grammar* of (G, C, ϕ) and C is called the *control language* of (G, C, ϕ) . Sentences of C will be referred to as *control words*. \square

Definition 3.4. A production $A(x_1, \dots, x_n) \rightarrow t$ of a macro grammar is called *argument preserving* [10] if each variable x_i ($1 \leq i \leq n$) occurs in the term t .

Let $G = (\Phi, \Psi, \Sigma, X, P, S)$ be an (m,K) -elb grammar. A production of the form $\Psi(x_1, \dots, x_n) \rightarrow L_0$ in P , where $\psi \in \Psi_n$, is called *argument preserving* if each variable x_i ($1 \leq i \leq n$) occurs in each word w from L_0 . \square

Note that productions of the form 2.4(i) and 2.4(ii) are argument preserving by definition.

For an (m,REG,K) -belb grammar (G, C, ϕ) , with $G = (\Phi, \Psi, \Sigma, X, P, S)$, let and $Term(G, \phi)$ denote the set of terms $T(\Sigma \cup X \cup \Phi \cup \Psi \cup \{\phi\})$. With each (m,REG,K) -belb grammar we associate – as usual – a derivation relation. This derivation relation formalizes bidirectional rightmost rewriting; cf. Definition 3.6.

Definition 3.5. Let (G, C, ϕ) be an (m,REG,K) -belb grammar, where $G = (\Phi, \Psi, \Sigma, X, P, S)$. Let ρ be a rule from $P \cup \bar{P}$, where $\alpha[\vec{x}]$ is the left-hand side of ρ , and let τ be a term in $Term(G, \phi)$.

- (a) If $\alpha[\vec{x}]$ is of the form $Z(\dots)$, with $Z \in \Phi \cup \Psi$, then we say that τ *fits in with* ρ in case there are arguments t_1, \dots, t_n from $Term(G, \phi)$ such that $\tau = \alpha[t_1, \dots, t_n]$, where $\alpha[t_1, \dots, t_n]$ is the result of substituting the terms t_1, \dots, t_n for x_1, \dots, x_n in $\alpha[\vec{x}]$, respectively.
- (b) If $\alpha[\vec{x}]$ is a language $L_0 \subseteq (\Sigma \cup X)^*$, i.e., ρ is a reduction of the form $L_0 \rightarrow \Psi(\vec{x})$, then τ *fits in with* ρ if there is at least one string t in L_0 such that $\tau = t[t_1, \dots, t_n]$, where $t[t_1, \dots, t_n]$ is the result of substituting the terms t_1, \dots, t_n for x_1, \dots, x_n in t , respectively. \square

Definition 3.6. Let (G, C, ϕ) be an (m,REG,K) -belb grammar, where $G = (\Phi, \Psi, \Sigma, X, P, S)$. Let ρ be rule from $P \cup \bar{P}$, and σ, τ be terms in $Term(G, \phi)$. We write $\sigma \Rightarrow_{r,m}^{\rho} \tau$ if there exists a term u in $Term(G, \phi)$, and strings v, x and y over the alphabet $\Phi \cup \Psi \cup \Sigma \cup X \cup PC$ such that $\sigma = xuy$ and $\tau = xvy$ and

- y contains no symbol from $\Phi \cup \Psi$,
- if λ fits in with ρ then $u = y = \lambda$,
- u is the only subterm in uy that fits in with ρ ,
- either ρ is a production, τ is the result of rewriting σ by ρ , and $\sigma \Rightarrow_m \tau$, or ρ is a reduction, σ is the result of rewriting τ by $\bar{\rho}$, and $\tau \Rightarrow_m \sigma$. \square

Let $C \subseteq (P \cup \bar{P})^*$ be a control language. A control word c in C is a sequence of rules. The application of a sequence of rules from $P \cup \bar{P}$ to a term τ is defined as the successive application of the rules which constitute c . Viz., if $c = \rho_1 \dots \rho_n$ ($n \geq 0$), then we write $\tau \Rightarrow_{r,m}^c \tau'$ if there are terms τ_i ($0 \leq i \leq n$) such that $\tau_0 = \tau$, $\tau_n = \tau'$ and for each i ($0 \leq i < n$) $\tau_i \Rightarrow_{r,m}^{\rho_{i+1}} \tau_{i+1}$ holds. In case a rule ρ_i in c is not applicable to the term τ_i , then further application of rules is blocked, and the application of c to τ yields no result, i.e., there is no term τ' such that $\tau \Rightarrow_{r,m}^c \tau'$ is defined.

Definition 3.7. An (m, REG, K) -belb grammar provided with right-most rewriting will be called an (r, m, REG, K) -belb grammar. Let (G, C, ϕ) be an (r, m, REG, K) -belb grammar with underlying grammar $G = (\Phi, \Psi, \Sigma, X, P, S)$ and control language $C \subseteq (P \cup \bar{P})^*$. Then the language generated by (G, C, ϕ) under the mode r, m is defined by

$$L_{r,m}(G, C, \phi) = \{w \in \Sigma^* \mid \exists c \in C \cdot S \Rightarrow_{r,m}^c w\}.$$

The family of languages generated by (r, m, REG, K) -belb grammars is denoted by $RBLB_{r,m}(K)$. \square

The derivation relation $\Rightarrow_{r,m}$ defined above corresponds to the RS/B-mode of derivation as defined in [15] for RCB grammars.

Example 3.8. Let L_1 be the language of equal length substrings, i.e.,

$$L_1 = \{x_1 c x_2 \dots c x_n \mid x_i \in \{a, b\}^*, |x_i| = m, 1 \leq i \leq n, n, m \geq 1\}.$$

In [10] it is shown that this language can be generated by an OI macro grammar. The language L_1 belongs to $RBLB_{r,OI}(\emptyset NE)$, i.e., it can also be generated by the following $(r, OI, REG, \emptyset NE)$ -belb grammar (G, C, ϕ) .

Define G by $(\Phi, \Psi, \Sigma, X, P, S)$, where the set of nonterminals Φ is equal to $\{S, A, B, D, E, F, H\}$, and S is the start symbol. The alphabet of language names Ψ equals $\{\psi_i \mid 0 \leq i \leq 9\}$. The set Σ of terminals is $\{a, b, c\}$, and $X = \{x, y\}$. Finally, the set of productions P of G consists of

$$\begin{array}{ll} \pi_0 = S \rightarrow A(\psi_0), & \pi_{12} = F \rightarrow \psi_6, \\ \pi_1 = A(x) \rightarrow A(\psi_1(x)), & \pi_{13} = F \rightarrow \psi_7, \\ \pi_2 = A(x) \rightarrow B(\psi_2(x)), & \pi_{14} = \psi_0 \rightarrow \emptyset, \\ \pi_3 = B(x) \rightarrow \psi_3(x), & \pi_{15} = \psi_6 \rightarrow \{a\}, \\ \pi_4 = A(x) \rightarrow \psi_2(x), & \pi_{16} = \psi_7 \rightarrow \{b\}, \\ \pi_5 = D(x) \rightarrow \psi_1(x), & \pi_{17} = H(x) \rightarrow \psi_4(x), \\ \pi_6 = D(x) \rightarrow E(\psi_2(x), \psi_4(x)), & \pi_{18} = H(x) \rightarrow \psi_8(x), \\ \pi_7 = E(x, y) \rightarrow \psi_5(x, y), & \pi_{19} = H(x) \rightarrow \psi_9(x), \\ \pi_8 = \psi_5(x, y) \rightarrow \{xy\}, & \pi_{20} = \psi_4(x) \rightarrow \emptyset, \\ \pi_9 = \psi_3(x) \rightarrow \{xcx\}, & \pi_{21} = \psi_8(x) \rightarrow \{a\}, \\ \pi_{10} = \psi_2(x) \rightarrow \{x\}, & \pi_{22} = \psi_9(x) \rightarrow \{b\}, \\ \pi_{11} = F \rightarrow \psi_0, & \end{array}$$

The rank of the symbols in $\Phi \cup \Psi$ are easy to infer from the productions in P . Define the control language C by the regular expression

$$\pi_0 \pi_1^* (\pi_2 \pi_3 \pi_9 \bar{\pi}_4)^* \pi_4 (\pi_{10} (\bar{\pi}_5 \pi_6 \pi_7 \pi_8 \bar{\pi}_{17} (\pi_{18} \pi_{21} + \pi_{19} \pi_{22}) + \bar{\pi}_{11} (\pi_{12} \pi_{15} + \pi_{13} \pi_{16})))^+.$$

First, a string $A(\psi_1(\dots \psi_1(\psi_0)\dots))$, in which the language name ψ_1 occurs n times, is generated by $\pi_0 \pi_1^n$ ($n \geq 0$). We represent the argument of A by $[n]$, where $[0]$ represents ψ_0 . By applying k times ($k \geq 0$) $\pi_2 \pi_3 \pi_9 \bar{\pi}_4$, followed by π_4 we obtain the string $\psi_2([n])c \dots c \psi_2([n])$, which contains k times the symbol c . In the following we discuss the expanding of a substring $\psi_2([n])$. By $\bar{\pi}_{10}$ we derive $\psi_2([n])$ to $[n]$, which is rewritten to $\psi_2([n-1])H([n-1])$ by the subsequence $\bar{\pi}_5 \pi_6 \pi_7 \pi_8 \bar{\pi}_{17}$ in case $n \geq 1$, and to F by the reduction $\bar{\pi}_{11}$ in case $n = 0$. Next, both $H([n-1])$ and F are expanded to a or b by the sequences $\pi_{18} \pi_{21} + \pi_{19} \pi_{22}$ and $\pi_{12} \pi_{15} + \pi_{13} \pi_{16}$, respectively.

Then $L_{r, \text{OI}}(G, C, \phi) = L_1$, which can now be easily verified. \square

Example 3.9. Consider the $(r, f, \text{OI}, \text{REG}, \emptyset \text{NE})$ -belb grammar (G, C) . We define the $(\text{OI}, \emptyset \text{NE})$ -elb grammar G by $G = (\Phi, \Psi, \{0, 1\}, X, P, S)$, with $X = \{x\}$, $\Phi = \{S, A, B, D\}$, $\Psi = \{\psi_1, \psi_2, \psi_3, \psi_4, \psi_5\}$, and P consists of the productions

$$\begin{array}{ll} \pi_0 = S \rightarrow A(\psi_1), & \pi_6 = B \rightarrow \psi_1, \\ \pi_1 = A(x) \rightarrow A(\psi_2(x)), & \pi_7 = B \rightarrow D(\psi_1), \\ \pi_2 = A(x) \rightarrow \psi_3(x), & \pi_8 = D(x) \rightarrow \psi_4(x), \\ \pi_3 = \psi_3(x) \rightarrow \{x\}, & \pi_9 = \psi_4(x) \rightarrow \{0x\}, \\ \pi_4 = \psi_2(x) \rightarrow \{xx\}, & \pi_{10} = D(x) \rightarrow \psi_5(x), \\ \pi_5 = \psi_1 \rightarrow \{1\}, & \pi_{11} = \psi_5(x) \rightarrow \{x0\}. \end{array}$$

First, a string $A(\psi_2(\dots(\psi_2(\psi_1))\dots))$ is generated, with n occurrences of the language name ψ_2 ($n \geq 0$). Then by the productions π_2 and π_3 , followed by a sequence of productions π_4 , we obtain a string with two language names ψ_1 at the right end. After each of these ψ_1 's has been rewritten into terminal strings in 0^*10^* , we continue to apply productions π_4 . This yields again two ψ_1 's from which strings in 0^*10^* can be derived. This continues until a completely terminal string is obtained. The total number of language names ψ_1 that show up during this derivation equals 2^n .

Each ψ_1 generates some string in 0^*10^* by the following sequences of rules. Let $c_1 = \pi_5$, $c_2 = \bar{\pi}_6\pi_7\pi_8\pi_9$, and $c_3 = \bar{\pi}_6\pi_7\pi_{10}\pi_{11}$. Then

$$\psi_1 \Rightarrow_{r, f, \text{OI}}^{c_1} 1, \quad \psi_1 \Rightarrow_{r, f, \text{OI}}^{c_2} 0\psi_1, \quad \text{and} \quad \psi_1 \Rightarrow_{r, f, \text{OI}}^{c_3} \psi_1 0.$$

As the control language we take the trivial control language, i.e., $C = (P \cup \bar{P})^*$. We have

$$L_{r, f, \text{OI}}(G, C) = \{w \in \{0, 1\}^* \mid \text{the number of 1's in } w \text{ is a power of } 2\},$$

which can easily be checked. Cf. [10], where it has also been proved that this language can be generated by an OI-macro grammar but not by an IO-macro grammar. \square

Although it is straightforward to define reductions associated with productions of the form 2.4(iii) (cf. Definition 3.6.), we do not study grammatical models in which such (arbitrary) reductions occur. These terminal reductions have the effect that they allow terminals to act as some kind of nonterminal symbol, which makes the distinction between terminals and nonterminals unclear. We have noticed this problem already in the case of regularly controlled bidirectional grammars that have a context-free grammar as its underlying grammar; cf. [15]. This restriction means that in this paper we only study the *fair mode* – cf. [15] – of bidirectional rewriting, in which we disallow terminal reductions. As a consequence, it enables us to omit the symbol ϕ . We will call this type of grammar an (r, f, m, REG, K) -belb grammar. The family of languages generated by (r, f, m, REG, K) -belb grammars is denoted by $RBLB_{r, f, m}(K)$. Note that the language of Example 3.8 can be generated by an $(r, f, \text{OI}, \text{REG}, \emptyset \text{NE})$ -belb grammar.

In the remainder of this section we clarify the structure of sentential forms generated by (r, f, m, REG, K) -belb grammars. Since the family of regular languages is closed under intersection, we can put regular restrictions on the control language. In the sequel, we assume that the set of productions P is the union of the disjoint sets P_1 , P_2 and P_3 , where P_1 , P_2 and P_3 consist of productions of the form 2.4(i), 2.4(ii) and 2.4(iii), respectively. Then we assume without loss of generality that for an $(r, f, \text{OI}, \text{REG}, K)$ -belb grammar (G, C) with underlying grammar $G = (\Phi, \Psi, \Sigma, X, P, S)$, the control language C is included in

$$((P_1\bar{P}_1 \cup P_2\bar{P}_2)^*(P_1 \cup \bar{P}_1 \cup P_2P_3^+(\bar{P}_2 \cup \{\lambda\})))^+. \quad (1)$$

For the same reason we can assume that an (r, f, IO, REG, K) -belb grammar (G, C) possesses a control language C which is included in

$$((\bar{P}_2 P_2 \cup P_3)^* (\bar{P}_2 \cup \{\lambda\})) (P_1 (\bar{P}_2 P_2)^* \bar{P}_1 \cup P_2 \bar{P}_2)^* (P_1 \cup P_2 P_3)^+. \quad (2)$$

In addition, for both modes OI and IO we may assume that the first rule of each control word in C is an initial production.

The restriction to control languages which are included in (1) or (2) becomes apparent when we inspect the structure of a sentential form occurring in the derivation according to an (r, f, m, REG, K) -belb grammar (G, C) . We represent terms from $Term(G)$ as follows. Define a *c-tree* as a variation on the well-known tree structure in which now the nodes are strings over $\Phi \cup \Psi \cup \Sigma \cup X \cup \{\#\}$. The symbol $\#$ is used to denote the concatenation operation in $T(\Phi \cup \Psi \cup \Sigma \cup X)$ explicitly. If A is an element of $\Phi \cup \Psi$ with rank n and $n \geq 1$, then A has n descendants which are again c-trees. If a node α is a string of symbols of rank zero, i.e., $\alpha \in (\Phi_0 \cup \Psi_0 \cup \Sigma \cup X \cup \{\#\})^*$, then α is called a *leave*. As an example, the term

$$A(\psi(x_1, ax_2b)x_1, ab)\psi(a, b)x_1$$

is represented by the c-tree in Figure 1.

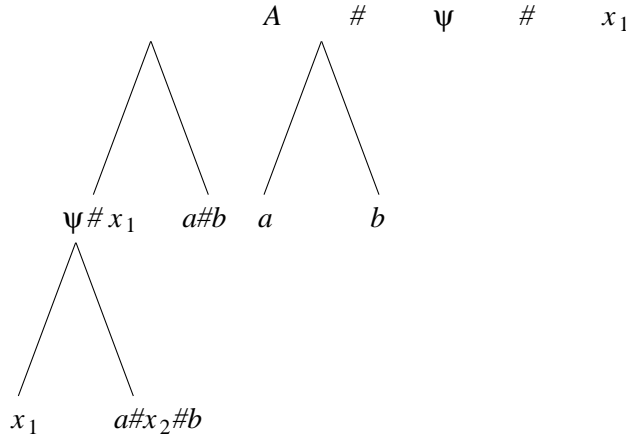


Figure 1.

Note that a c-tree does not represent a derivation of the grammar (G, C) .

A derivation corresponding to an (r, f, OI, REG, K) -belb grammar consists of a sequence of sentential forms which have the form

$$\psi_1(\vec{\psi}_1(\vec{t}_1))w_1 \dots w_{n-1} \psi_n(\vec{\psi}_n(\vec{t}_n))w_n A(\vec{\psi}_0(\vec{t}_0))w_0. \quad (3)$$

The formula $\vec{\psi}_i(\vec{t}_i)$ is the abbreviation of

$$\psi_{i1}(t_{i1}), \dots, \psi_{ir(i)}(t_{ir(i)}) \quad \text{with } 0 \leq i \leq n.$$

In (3) the following notational conventions are used. The symbols ψ_{ij} are language names. The number of language names may be zero, i.e., $n \geq 0$. The symbol A is either a language name or a nonterminal. A terminal string is a possible sentential form, so the substring $A(\dots)$ is optional in (3). Each t_{ij} is a list of terms over Ψ , each of which can be represented by a c-tree in which each node is a language name; the leaves of each term in t_{ij} are language names of rank zero. In the c-tree representation of (3), shown in Figure 2, these t_{ij} 's are represented by a triangle. The terms w_i are strings in $(\Sigma \cup X)^*$. Note that $r(i)$ is equal to the rank of ψ_i ($i \geq 1$), and

$r(0)$ is equal to the rank of A .

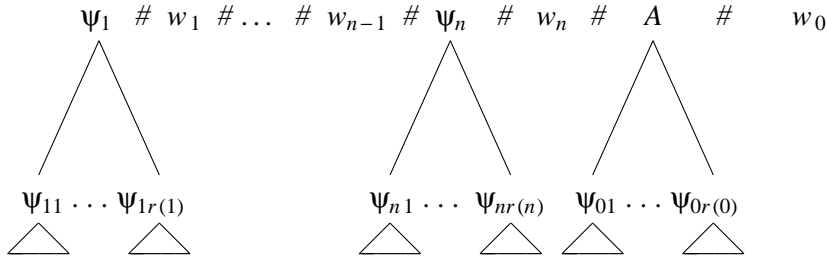


Figure 2.

A sentential form generated by an (r, f, IO, REG, K) -belb grammar (G, C) is of the form

$$A_0(\Psi_{01}(t_{01}), \dots, \Psi_{0i_0}(t_{0i_0}), A_1(\Psi_{11}(t_{11}), \dots, \Psi_{1i_1}(t_{1i_1}), A_2(\dots, A_n(\Psi_{n1}(t_{n1}), \dots, \Psi_{ni_n}(t_{ni_n}), w_{ni_n+1}, \dots, w_{r(n)}), \dots, w_{r(2)}, w_{1i_1+2}, \dots, w_{r(1)}, w_{0i_0+2}, \dots, w_{r(0)}). \quad (4)$$

In this sentential form (4) the A_i 's are nonterminals, the Ψ_{ij} 's denote language names, the w_{ij} 's are strings over $\Sigma \cup X$, and each t_{pq} denotes a list of w_{ij} 's the length of which is equal to the rank of Ψ_{pq} . Furthermore, $r(i)$ ($0 \leq i \leq n$) is equal to the rank of A_i . The sentential form (4) is represented by a c-tree in Figure 3 in which each t_{ij} is represented by a triangle.

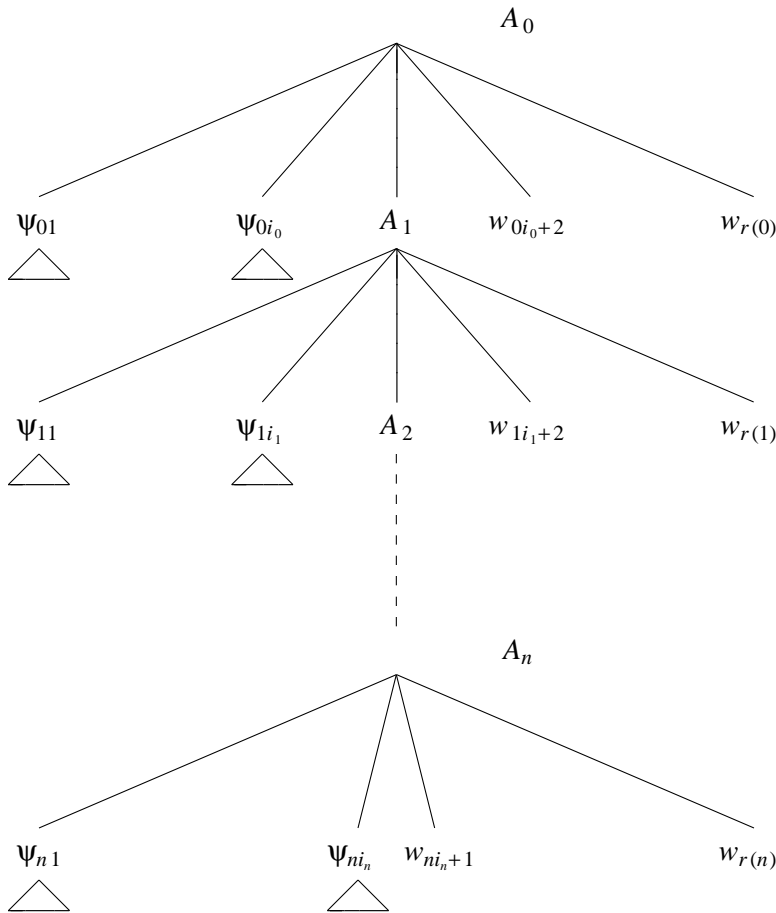


Figure 3.

4. Properties of $RBLB_{r,f,m}(K)$ -languages

In the proofs of the following propositions we assume that L_i ($i \geq 1$) is a language generated by an (r,f,m,REG,K) -belb grammar (G_i, C_i) with $G_i = (\Phi_{(i)}, \Psi_{(i)}, \Sigma_i, X_i, P_{(i)}, S_i)$. Thus $\Phi_{(i)}$ and $\Psi_{(i)}$ are ranked alphabets, i.e., $\Phi_{(i)} = \cup \Phi_{(i)j}$ and $\Psi_{(i)} = \cup \Psi_{(i)j}$. We assume that the sets of language names and the sets of variables of these grammars are mutually disjoint, i.e., $i \neq j$ implies $\Phi_{(i)} \cap \Phi_{(j)} = \emptyset$, $\Psi_{(i)} \cap \Psi_{(j)} = \emptyset$ and $X_i \cap X_j = \emptyset$.

Remember that a family of languages K is closed under *left- [right-] marking* if for each language L_0 in K , the language $\{\$\}L_0 [L_0\{\$\}]$, respectively] is in K , where the symbol $\$$ does not occur in the alphabet of L_0 . Frequently, we write $\$L_0$ instead of $\{\$\}L_0$.

The family *SYMBOL* is defined as the family of all languages consisting of a single word which is of length one, i.e., $SYMBOL = \{\{a\} \mid a \in \Sigma_\omega\}$ where Σ_ω is a countably infinite set of terminal symbols.

Proposition 4.1. *Let K be a family of languages closed under left- or right-marking. If $K \supseteq SYMBOL$, then $RBLB_{r,f,m}(K)$ is closed under union, concatenation, Kleene + and Kleene *.*

Proof. Union. Straightforward. This even holds without the premisses on the family K .

Concatenation. We construct an (r,f,m,REG,K) -belb grammar (G, C) from (G_1, C_1) and (G_2, C_2) such that $L_{r,m}(G, C) = L_1 L_2$. For both modes OI and IO we can use the same underlying grammar G . Define G equal to $(\Phi, \Psi, \Sigma_1, X_1, P, S)$, where $\Phi = \Phi_{(1)} \cup \Phi_{(2)} \cup \{S, Z\}$, $S \in \Phi_0$, $Z \in \Phi_2$, and where $\Psi = \Psi_{(1)} \cup \Psi_{(2)} \cup \{\psi, \psi_1, \psi_2\}$ such that $\psi_1, \psi_2 \in \Psi_0$ and $\psi \in \Psi_2$. Furthermore, we assume that S, Z, ψ, ψ_1 , and ψ_2 are new symbols. The set of productions P is equal to $P_{(1)} \cup P_{(2)} \cup \{\pi_\psi, \pi_Z, \pi_0, \pi_1, \pi_2\}$ with $\pi_\psi = \psi(x, y) \rightarrow \{xy\}$, $\pi_Z = Z(x, y) \rightarrow \psi(x, y)$, $\pi_0 = S \rightarrow Z(\psi_1, \psi_2)$, $\pi_1 = S_1 \rightarrow \psi_1$, and $\pi_2 = S_2 \rightarrow \psi_2$. Note that $\{xy\}$ belongs to K , as K includes the family *SYMBOL*, $x \neq y$, and K is closed under left- or right-marking. Now if $m = \text{OI}$, then define the control language C by $\pi_0 \pi_Z \pi_\psi \pi_2 C_2 \pi_1 C_1$. Otherwise, if $m = \text{IO}$, then we define C by $\pi_0 \pi_2 C_2 \pi_1 C_1 \pi_Z \pi_\psi$.

Kleene +. As in the case of concatenation, we construct an (r,f,m,REG,K) -belb grammar (G, C) from (G_1, C_1) , that generates L_1^+ . The underlying grammar G is for both modes OI and IO the same, but the control languages are different. Viz., define G by $(\Phi, \Psi, \Sigma, X, P, S)$, where $\Phi = \Phi_{(1)} \cup \{S, Z\}$ with $S \in \Phi_0$ and $Z \in \Phi_2$, and where the set Ψ is equal to $\Psi_{(1)} \cup \{\psi, \psi_1, \psi_2\}$, with $\psi_1, \psi_2 \in \Psi_0$ and $\psi \in \Psi_2$. Again S, Z, ψ, ψ_1 , and ψ_2 are new symbols. We also assume $\Phi_{(1)} \cap \{S, Z\} = \emptyset$ and $\Psi_{(1)} \cap \{\psi, \psi_1, \psi_2\} = \emptyset$. The set of productions P is formed by $P_{(1)} \cup \{\pi_\psi, \pi_Z, \pi_S, \pi_0, \pi_1\}$, where $\pi_S = S \rightarrow S_1$, $\pi_\psi = \psi(x, y) \rightarrow \{xy\}$, $\pi_Z = Z(x, y) \rightarrow \psi(x, y)$, $\pi_0 = S_1 \rightarrow Z(\psi_1, \psi_1)$, and $\pi_1 = S_1 \rightarrow \psi_1$. The control language C is equal to $\pi_S(\pi_0 \pi_Z \pi_\psi \pi_1 C_1 \pi_1)^* C_1$ if m is equal to OI, and in case of the IO-mode we take C equal to $\pi_S(\pi_0 \pi_1 C_1 \pi_1)^* C_1 (\pi_Z \pi_\psi)^*$.

Kleene *. Straightforward.

Note that in the proofs presented above the control languages have for both modes OI and IO a form in accordance with (1) and (2) from Section 2, respectively. \square

In the next proposition we show the closure under ngsms mappings of the language family $RBLB_{r,f,\text{OI}}(K)$. Therefore, we recall the following definition.

Definition 4.2. An *ngsm* or a *nondeterministic generalized sequential machine* is a 6-tuple $T = (Q, \Sigma, \Delta, \delta, q_0, Q_F)$, where

- Q is a finite alphabet of *states*,

- Σ is a finite alphabet of *input symbols*,
- Δ is a finite alphabet of *output symbols*,
- $q_0 \in Q$ is the *initial state*,
- $Q_F \subseteq Q$ is the set of *accepting states*,
- δ is a mapping from $Q \times \Sigma$ into the finite subsets of $Q \times \Delta^*$.

As usual, δ is extended to a function from $Q \times \Sigma^*$ into the finite subsets of $Q \times \Delta^*$ as follows.

(i) $\delta(q, \lambda) = \{(q, \lambda)\}$,

(ii) For $q \in Q, x \in \Sigma^*$ and $a \in \Sigma$,

$$\delta(q, xa) = \{(p, w) \mid w = w_1 w_2 \text{ and for some } r \text{ in } Q, (r, w_1) \text{ is in } \delta(q, x) \\ \text{and } (p, w_2) \text{ is in } \delta(r, a)\}.$$

The mapping associated with $T = (Q, \Sigma, \Delta, \delta, q_0, Q_F)$ – called an *ngsm mapping* and denoted by T too – is the function $T: \Sigma^* \rightarrow 2^\Delta$ defined by $T(w) = \{z \mid (q, z) \in \delta(q_0, w), q \in Q_F\}$. The extension of T to a language L_0 over Σ is defined by $T(L_0) = \cup\{T(w) \mid w \in L_0\}$. \square

The proof of the following proposition is performed by applying the well-known “triple” construction.

Proposition 4.3. *Let K be a family closed under ngsm mappings. Then $RBLB_{r,f, OI}(K)$ is closed under ngsm mappings.*

Proof. Let (G_1, C_1) be an (r, f, OI, REG, K) -belb grammar with $G_1 = (\Phi_{(1)}, \Psi_{(1)}, \Sigma, X_1, P_{(1)}, S_1)$, $C_1 \subseteq (P_{(1)} \cup \overline{P_{(1)}})^*$, and let T be an ngsm with $T = (Q, \Sigma, \Delta, \delta, q_0, Q_F)$. We construct an (r, f, OI, REG, K) -belb grammar (G, C) such that $L_{r, OI}(G, C) = T(L_{r, OI}(G_1, C_1))$. Define the set of variables X as $\{x_{i[p, q]} \mid 1 \leq i \leq |X_1|, p, q \in Q\}$, and let $Q = \{q_0, \dots, q_N\}$. With the list x_1, \dots, x_n we associate the list $x_{1[q_0, q_0]}, \dots, x_{n[q_N, q_N]}$, denoted by \tilde{x} . It consists of $n |Q|^2$ different variables from X ; $|Q|$ is the cardinality of the set Q . Let $\tilde{\Psi}(\tilde{x})$ denote

$$(q_0 \Psi q_0)(\tilde{x}), \dots, (q_N \Psi q_N)(\tilde{x}).$$

For each $p, q \in Q$, let T_{pq} be the ngsm mapping induced by the ngsm $(Q, \Sigma \cup X_1, \Sigma \cup X, \delta', p, \{q\})$, where $\delta': Q \times (\Sigma \cup X_1) \rightarrow 2^{(Q \times (\Sigma \cup X))^*}$ is the mapping defined by

$$\delta'(s, y) = \{(t, z) \mid (t, z) \in \delta(s, y), y \in \Sigma\} \cup \{(t, x_{i[s, t]}) \mid t \in Q, x_i = y, y \in X_1\}.$$

Furthermore, let U be the union of the sets

$$\{(p A q)(\tilde{x}) \rightarrow (p B q)(\tilde{\Psi}_1(\tilde{x}), \dots, \tilde{\Psi}_k(\tilde{x})) \mid A \in \Phi_n, B \in \Phi_k, p, q \in Q, \\ \Psi_i \in \Psi_n, n \geq 0, k \geq 1\},$$

$$\{(p A q)(\tilde{x}) \rightarrow (p \Psi q)(\tilde{x}) \mid A \in \Phi_n, \Psi \in \Psi_n, n \geq 0, p, q \in Q\}, \text{ and}$$

$$\{(p \Psi q)(\tilde{x}) \rightarrow T_{pq}(L_0) \mid \Psi \in \Psi_n, p, q \in Q, \Psi(\vec{x}) \rightarrow L_0 \in P_{(1)}\}.$$

Define a finite substitution $\tau: (P_{(1)} \cup \overline{(P_{(1)} - P_{(1)3})})^* \rightarrow 2^{(U \cup \overline{U})^*}$ by

$$\tau(S_1 \rightarrow A(\Psi_1, \dots, \Psi_n)) = \{(q_0 S_1 q_f) \rightarrow (q_0 A q_f)(\tilde{\Psi}_1, \dots, \tilde{\Psi}_n) \mid q_f \in Q_F\},$$

$$\tau(A(\vec{x}) \rightarrow B(\Psi_1(\vec{x}), \dots, \Psi_k(\vec{x}))) =$$

$$\{(p A q)(\tilde{x}) \rightarrow (p B q)(\tilde{\Psi}_1(\tilde{x}), \dots, \tilde{\Psi}_k(\tilde{x})) \mid p, q \in Q\},$$

$$\tau(A(\vec{x}) \rightarrow \Psi(\vec{x})) = \{(p A q)(\tilde{x}) \rightarrow (p \Psi q)(\tilde{x}) \mid p, q \in Q\},$$

$$\tau(\Psi(\vec{x}) \rightarrow L_0) = \{(p \Psi q)(\vec{x}) \rightarrow T_{pq}(L_0) \mid p, q \in Q\}.$$

Since there are no terminal reductions involved, a reduction $\bar{\pi}$ is always a reduction associated with an argument-preserving production of type 2.4(i) or 2.4(ii). Therefore, we can define $\tau(\bar{\pi})$ equal to $\bar{\tau}(\bar{\pi})$.

Then we define G equal to $(\Phi, \Psi, \Sigma, X, P, S)$, where Φ is given by

$$\Phi_0 = \{S\} \cup \{(p A q) \mid A \in \Phi_{(1)0}, p, q \in Q\},$$

$$\Phi_n |Q|^2 = \{(p A q) \mid A \in \Phi_{(1)n}, p, q \in Q\} \quad \text{for each } n (n \geq 1),$$

$$\Phi_l = \emptyset \text{ if there is no } n \in \mathbb{N} \text{ with } l = n |Q|^2,$$

and Ψ is given by

$$\Psi_n |Q|^2 = \{(p \Psi q) \mid \Psi \in \Psi_{(1)n}, p, q \in Q\} \quad \text{for each } n (n \geq 0),$$

$$\Psi_l = \emptyset \text{ if there is no } n \in \mathbb{N} \text{ with } l = n |Q|^2.$$

The set of productions P equals $\tau(P_{(1)}) \cup P_S$, where P_S is equal to $\{S \rightarrow (q_0 S_1 q_f) \mid q_f \in Q_F\}$, and, finally, we define the new control language C equal to $P_S \tau(C_1)$. Then $L_{r, \text{OI}}(G, C) = T(L_{r, \text{OI}}(G_1, C_1))$. \square

Remember that a family closed under ngsm mappings if and only if it is closed under intersection with regular languages and under finite substitution; cf. Lemma 9.3 in [18]. Therefore, a direct consequence of Proposition 4.3 is the following result.

Corollary 4.4. *Let K be a family closed under ngsm mappings. Then the family of languages $RBLB_{r, f, \text{OI}}(K)$ is closed under intersection with regular languages and under finite substitution.* \square

Next we establish closure under two types of substitution. First we give precise definitions of substituting words for symbols in a word “nondeterministically” (Definition 4.5) and “deterministically” (Definition 4.6).

Definition 4.5. Let K be a family of languages and let Σ_1 be an alphabet. A *nondeterministic K -substitution* (or *n K -substitution*) τ is a mapping from Σ_1 into the set of K -languages which is extended to words in Σ_1^* by $\tau(\lambda) = \{\lambda\}$ and $\tau(a_1 \dots a_n) = \tau(a_1) \dots \tau(a_n)$, where $a_i \in \Sigma_1$ ($1 \leq i \leq n$), or, equivalently,

$$\tau(a_1 \dots a_n) = \{w_1 \dots w_n \mid w_i \in \tau(a_i), 1 \leq i \leq n\}.$$

The mapping τ is extended to languages L_0 over Σ_1 by

$$\tau(L_0) = \cup \{\tau(w) \mid w \in L_0\}. \quad \square$$

Notice that in case the family K equals *ONE*, *FIN* or *REG*, an nK -substitution is known as a homomorphism, finite substitution and regular substitution, respectively.

The addition of the adjective “nondeterministic” suggests that we can also consider deterministic substitutions [2, 9]. The difference with the usual (nondeterministic) substitution – the additional “nondeterministic” may be omitted – is that in a deterministic K -substitution τ we choose in advance for each letter a in Σ_1 a fixed word w_a from the language $\tau(a)$; the language $\tau(a)$ belongs to the family K . Then in the application of τ to a word ω each occurrence of a is replaced by w_a . The choice of the words w_a determines a homomorphism $h : \Sigma_1 \rightarrow \Sigma_2^*$. Therefore $\tau(\omega)$ is defined to be equal to the set of the images of all homomorphisms $h : \Sigma_1 \rightarrow \Sigma_2^*$ such that $h(a)$ is in $\tau(a)$. We define this formally.

Definition 4.6. Let K be a family of languages and let Σ_1 be an alphabet. A *deterministic K -substitution* (or *d K -substitution*) τ is a mapping from Σ_1 into the set of K -languages. It is

extended to words in Σ_1^* by $\tau(\lambda) = \{\lambda\}$ and

$$\tau(a_1 \dots a_n) = \{h(a_1) \dots h(a_n) \mid h \text{ is a homomorphism such that } h(a) \in \tau(a) \text{ for each } a \in \Sigma_1\}, \text{ where } a_i \in \Sigma_1 \ (1 \leq i \leq n).$$

The extension of τ to languages L_0 over Σ_1 is defined by

$$\tau(L_0) = \cup \{\tau(w) \mid w \in L_0\}. \quad \square$$

From this definition it follows that $\tau(\omega) = \emptyset$ for each word ω in case $\tau(a) = \emptyset$ and at least one symbol a occurs in ω . It is also important to note that a dK -substitutions is not a special case of a nondeterministic substitution, but they are both different generalizations of the notion of homomorphism. In fact, a homomorphism is both a $dONE$ -substitution and an $nONE$ -substitution.

Definition 4.7. A family F is closed under nK -substitution [dK -substitution] if for each language L_0 in F and each nK -substitution [dK -substitution, respectively] τ the language $\tau(L_0)$ is in F . In case the family K equals the family F , the we say that F is closed under (n-)substitution [d -substitution, respectively]. \square

The following proposition shows that under weak assumptions on K the family of languages $RBLB_{r,f,OI}(K)$ is closed under n -substitution.

Proposition 4.8. *Let K be a family closed under isomorphism such that $SYMBOL \cup \{\emptyset\} \subseteq K$. Then $RBLB_{r,f,OI}(K)$ is closed under nondeterministic substitution.*

Proof. Let $L_1 = L_{r,OI}(G_1, C_1)$ be a language in $RBLB_{r,f,OI}(K)$, where $G_1 = (\Phi_{(1)}, \Psi_{(1)}, \Sigma_1, X_1, P_{(1)}, S_1)$. Let $\Sigma_1 = \{a_1, \dots, a_N\}$, and let $\sigma: \Sigma_1 \rightarrow 2^{\Sigma^*}$ be a nondeterministic $RBLB_{r,f,OI}(K)$ -substitution, such that for each a in Σ_1 the language $\sigma(a)$ is generated by the (r,f, OI, REG, K) -belb grammar (G_a, C_a) , where $G_a = (\Phi_{(a)}, \Psi_{(a)}, \Sigma_a, X_a, P_{(a)}, S_a)$. We construct an (r,f, OI, REG, K) -belb grammar (G, C) with underlying grammar $G = (\Phi, \Psi, \Sigma, X, P, S)$ such that $\sigma(L_1) = L_{r,OI}(G, C)$.

Essentially, we use the terminals in Σ_1 of (G_1, C_1) as variables in (G, C) via a transformation which associates with each a in Σ_1 a corresponding variable y_a in X . Each terminal a in Σ_1 which occurs in the K -languages at the right-hand side of the productions of type 2.4(iii) in (G_1, C_1) is replaced by the variable y_a . The original start symbol S_1 in (G_1, C_1) is transformed into S'_1 such that S'_1 has rank N . The new start symbol S is used in the new initial production $S \rightarrow S'_1(\psi_{a_1}, \dots, \psi_{a_N})$. The other productions of (G, C) are obtained from those in (G_1, C_1) by adorning them with additional variables y_{a_1}, \dots, y_{a_N} . Then throughout each derivation the language names $\psi_{a_1}, \dots, \psi_{a_N}$ will be passed on downwards. By applying reductions $\psi_a \rightarrow S_a$ followed by a control word from C_a (where $a \in \Sigma_1$) in the proper way, each $a \in \Sigma_1$ in a word from $L_{r,OI}(G_1, C_1)$ is substituted by the $RBLB_{r,f,OI}(K)$ -language $\sigma(a)$. Formally, we perform the construction of (G, C) in the following way.

Assume that the sets $\Phi_{(a)}$ with $a \in \Sigma_1$ are mutually disjoint. Let this property hold for the sets $\Psi_{(a)}$ and for the sets X_a too; in both cases a varies over Σ_1 . Then the alphabets Φ and Ψ are defined by

$$\Phi_0 = \{S\} \cup \cup \{\Phi_{(a)0} \mid a \in \Sigma_1\},$$

$$\Phi_n = \cup \{\Phi_{(a)n} \mid a \in \Sigma_1\} \text{ for each } n \text{ with } 1 \leq n < N,$$

$$\Phi_{n+N} = \{A' \mid A \in \Phi_{(1)n}\} \cup \cup \{\Phi_{(a)n+N} \mid a \in \Sigma_1\} \text{ for each } n \ (n \geq 0),$$

and

$$\Psi_0 = \{\psi_a \mid a \in \Sigma_1\} \cup \cup \{\Psi_{(a)0} \mid a \in \Sigma_1\},$$

$$\Psi_n = \cup \{\Psi_{(a)n} \mid a \in \Sigma_1\} \text{ for each } n \text{ with } 1 \leq n < N,$$

$$\Psi_{n+N} = \{\psi' \mid \psi \in \Psi_{(1)n}\} \cup \{\phi_{n,a} \mid a \in \Sigma_1\} \cup \cup \{\Psi_{(a)n+N} \mid a \in \Sigma_1\} \text{ for each } n (n \geq 0).$$

Define $X = X_1 \cup \{y_a \mid a \in \Sigma_1\} \cup \cup \{X_a \mid a \in \Sigma_1\}$. Let the sets U_1 , U_2 , and U_3 be defined in the following way, where $\vec{x} = (x_1, \dots, x_n)$ and $\vec{y} = (y_{a_1}, \dots, y_{a_N})$.

$$U_1 =$$

$$\{A'(\vec{x}, \vec{y}) \rightarrow B'(\psi'_1(\vec{x}, \vec{y}), \dots, \psi'_k(\vec{x}, \vec{y}), \phi_{n,a_1}(\vec{x}, \vec{y}), \dots, \phi_{n,a_N}(\vec{x}, \vec{y}))\}$$

$$A' \in \Phi_{n+N}, B' \in \Phi_{k+N}, \phi_{n,a} \in \Psi_{n+N}, a \in \Sigma_1, \psi'_i \in \Psi_{n+N}, 1 \leq i \leq k, n \geq 0\},$$

$$U_2 = \{A'(\vec{x}, \vec{y}) \rightarrow \psi'(\vec{x}, \vec{y}) \mid A' \in \Phi_{n+N}, \psi' \in \Psi_{n+N}, n \geq 0\},$$

$$U_3 = \{\psi'(\vec{x}, \vec{y}) \rightarrow i(L_0) \mid \psi' \in \Psi_{n+N}, \psi(\vec{x}) \rightarrow L_0 \in P_{(1)3}, n \geq 0\},$$

where $i : \Sigma_1 \cup X_1 \rightarrow X$ is the isomorphism defined by

$$\begin{aligned} i(x) &= x & \text{if } x \in X_1, \\ i(a) &= y_a & \text{if } a \in \Sigma_1. \end{aligned}$$

Let $U = U_1 \cup U_2 \cup U_3 \cup P_{\Sigma_1} \cup P_\phi$, where $P_{\Sigma_1} = \{S_a \rightarrow \psi_a \mid a \in \Sigma_1\}$ and $P_\phi = \{\phi_{n,a}(\vec{x}, \vec{y}) \rightarrow \{y_a\} \mid a \in \Sigma_1, n \geq 0\}$. Define the regular substitution $g : P_{(1)} \rightarrow 2^{U^*}$ by

$$g(A(\vec{x}) \rightarrow B(\psi_1(\vec{x}), \dots, \psi_k(\vec{x}))) =$$

$$\{A'(\vec{x}, \vec{y}) \rightarrow B'(\psi'_1(\vec{x}, \vec{y}), \dots, \psi'_k(\vec{x}, \vec{y}), \phi_{n,a_1}(\vec{x}, \vec{y}), \dots, \phi_{n,a_N}(\vec{x}, \vec{y}))\},$$

$$g(A(\vec{x}) \rightarrow \psi(\vec{x})) = \{A'(\vec{x}, \vec{y}) \rightarrow \psi'(\vec{x}, \vec{y})\},$$

$$g(\psi(\vec{x}) \rightarrow L_0) = \{(\psi'(\vec{x}, \vec{y}) \rightarrow i(L_0))(P_\phi^*(P_{\Sigma_1} \cup \{C_a \mid a \in \Sigma_1\})^*)^*\}.$$

We define the set P_\emptyset to be equal to $\{\psi_a \rightarrow \emptyset \mid a \in \Sigma_1\}$ in order to satisfy the condition that G ought to be an (OI, K) -elb grammar. Furthermore, if π is in $P_{(1)1} \cup P_{(1)2}$, then $g(\bar{\pi})$ is defined as $g(\bar{\pi}) = g(\pi)$.

Let $\pi_0 = S \rightarrow S'_1(\psi_{a_1}, \dots, \psi_{a_N})$. Then P is defined by

$$P = \cup \{P_{(a)} \mid a \in \Sigma_1\} \cup \{\psi'(\vec{x}, \vec{y}) \rightarrow i(L_0) \mid (\psi(\vec{x}) \rightarrow L_0) \in P_{(1)3}\} \cup$$

$$P_{\Sigma_1} \cup P_\phi \cup \{\pi_0\} \cup g(P_{(1)1} \cup P_{(1)2}) \cup P_\emptyset.$$

Finally, as the control language we take C equal to $\pi_0 g(C_1)$. \square

We recall the following concepts. A family of languages is called *nontrivial* if it contains a language which differs from \emptyset and from $\{\lambda\}$. A *full Abstract Family of Languages* or *full AFL* is a nontrivial family of languages which is closed under union, concatenation, Kleene +, homomorphism, inverse homomorphism and intersection with regular languages.

Corollary 4.9. *Let K be a nontrivial family closed under ngsml mappings. Then $RBLB_{r,f,OI}(K)$ is a full substitution-closed AFL.*

Proof. Recall that it is sufficient to prove closure under intersection with regular languages, regular substitution and union with a regular set in order to prove closure under inverse homomorphism [12]. We can easily show by the inclusion $LB_{OI}(K) \subseteq RBLB_{r,f,OI}(K)$ that under the

premisses on K the regular languages are included in $RBLB_{r,f, \text{OI}}(K)$. Then the statement follows immediately from Propositions 4.1, 4.3, and 4.8, and Corollary 4.4. \square

For the IO-mode closure under K -substitution or even under finite substitution is unlikely. On the other hand we can establish closure under intersection with regular languages and under deterministic substitution.

Proposition 4.10. *Let K be a family closed under intersection with regular languages. Then $RBLB_{r,f, \text{IO}}(K)$ is closed under intersection with regular languages.*

Proof. The proof is based on a modification of the technique of factored grammars [10]. Recall that each regular set R equals the (finite) union of a number of congruence classes which corresponds to a congruence relation \equiv – with respect to concatenation – over Σ^* of finite index; cf. [20]. Starting from an $(r, f, \text{IO}, \text{REG}, K)$ -belb grammar (G_1, C_1) we will construct an $(r, f, \text{IO}, \text{REG}, K)$ -belb grammar (G, C) such that in (G, C) we can tell just by looking at a nonterminal or a language name to which congruence class its arguments must belong if this nonterminal or language name ever appears in a sentential form. We also can determine to which congruence class any string generated by this nonterminal will belong. Therefore, we transform the grammar (G_1, C_1) in the following way. Each nonterminal and language name is adorned with $n+1$ congruence classes u_0, \dots, u_n , where n is the rank of that nonterminal or language name. However, an exception is made for the start symbol S_1 , which is left unchanged. The congruence class of the resulting terminal string generated by the transformed nonterminal – if any – is equal to u_0 .

The transformation of productions is arranged as follows. In connection with the exception made for the start symbol, the congruence class u_0 of the nonterminal on the right-hand side of an initial production ought to be taken from the finite number of congruence classes, the union of which equals R . The transformation of the remaining productions is such that with respect to the nonterminal or language name to the left-hand side we can freely choose the congruence classes u_0, \dots, u_n . Then the congruence class u_0 of the nonterminal or language name on top level of the right-hand side is equal to the corresponding congruence class on the left-hand side. And in case of non-initial productions of type 2.4(i) and productions of type 2.4(ii) the congruence class of each argument of the nonterminal on the left-hand side determines the congruence class of the corresponding argument of the language name on the right-hand side.

Concerning the productions of type 2.4(iii) we replace the language L by $L \cap R_{u_0, \vec{u}}$, where $\vec{u} = (u_1, \dots, u_n)$. The regular set $R_{u_0, \vec{u}}$ consists of those words in $(\Sigma \cup \{x_1, \dots, x_n\})^*$ such that substituting an element from the congruence class u_i for x_i ($1 \leq i \leq n$) results in a word from the congruence class u_0 . After applying this transformation, a combination u_0, \dots, u_n of congruence classes combined with a production $\psi(x_1, \dots, x_n) \rightarrow L_0$ of type 2.4(iii) gives a blocked derivation in case the intersection of the language L and the regular set $R_{u_0, \vec{u}}$ is empty. Viz., in that case there are no rules to rewrite the language name $[\psi, u_0, \vec{u}]$, and this particular guess of the grammar gives no contribution to the language generated by (G, C) .

It is left to the reader to check that the construction below formalizes the ideas presented above, and that the resulting grammar (G, C) generates the language $L_{r, \text{IO}}(G_1, C_1) \cap R$.

Let $G_1 = (\Phi_{(1)}, \Psi_{(1)}, \Sigma, X_1, P_{(1)}, S_1)$ and $C_1 \subseteq (P_{(1)} \cup \bar{P}_{(1)})^*$. Let R be a regular language accepted by the deterministic finite automaton $M = (Q, \Sigma, \delta, q_0, F)$. The relation \equiv on $\Sigma^* \times \Sigma^*$ is defined by

$$x \equiv y \quad \text{if and only if} \quad \forall q \in Q \cdot \delta(q, x) = \delta(q, y).$$

Because R is regular, it is possible to partition Σ^* by \equiv into a finite number of congruence classes. Let Σ^*/\equiv denote the set of congruence classes $[t_1], \dots, [t_k]$ induced by \equiv , where t_1, \dots, t_k are freely chosen but fixed representatives. Let R_{\equiv} be equal to $\{[t_i] \mid \delta(q_0, t_i) \in F, 1 \leq i \leq k\}$. Then we have the identity $R = \cup R_{\equiv}$. Define the sets $R_{u_0, \vec{u}}$, where $u_i \in \Sigma^*/\equiv$ ($0 \leq i \leq n$) by

$$R_{u_0, \vec{u}} = h_{\vec{u}}^{-1}(u_0),$$

where $h_{\vec{u}} : (\Sigma \cup X_1)^* \rightarrow \Sigma^*$ is the homomorphism defined by

$$\begin{aligned} h_{\vec{u}}(a) &= a && \text{for each } a \text{ in } \Sigma, \\ h_{\vec{u}}(x_i) &= t_i && \text{for each } x_i \text{ in } X_1, \text{ where } t_i \text{ equals } u_i = [t_i]. \end{aligned}$$

Note that $R_{u_0, \vec{u}}$ is regular.

Consider the following $(r, f, \text{IO}, \text{REG}, K)$ -belb grammar (G, C) which generates the language $L_{r, \text{IO}}(G_1, C_1) \cap R$. The underlying grammar G equals $(\Phi, \Psi, \Sigma, X_1, P, S_1)$, where P is given by $P = g(P_{(1)})$. The finite substitution g is defined in the following way for rules in $P_{(1)}$ of the types 2.4(i), 2.4(ii) and 2.4(iii). Let $\vec{u} = (u_1, \dots, u_n)$ and $\vec{v} = (v_1, \dots, v_k)$. Then

$$\begin{aligned} g(S_1 \rightarrow A(\psi_1, \dots, \psi_n)) &= \{S_1 \rightarrow [A, u_0, \vec{u}](\psi_1, u_1, \dots, \psi_n, u_n) \mid \\ &\quad u_0 \in R_{\equiv}, u_1, \dots, u_n \in \Sigma^*/\equiv\}, \\ g(A(\vec{x}) \rightarrow B(\psi_1(\vec{x}), \dots, \psi_k(\vec{x}))) &= \\ \{[A, u_0, \vec{u}](\vec{x}) \rightarrow [B, u_0, \vec{v}](\psi_1, v_1, \vec{u})(\vec{x}), \dots, \psi_k, v_k, \vec{u})(\vec{x}) \mid \\ &\quad u_0, \dots, u_n, v_1, \dots, v_k \in \Sigma^*/\equiv\}, \\ g(A(\vec{x}) \rightarrow \psi(\vec{x})) &= \{[A, u_0, \vec{u}](\vec{x}) \rightarrow [\psi, u_0, \vec{u}](\vec{x}) \mid u_0, \dots, u_n \in \Sigma^*/\equiv\}, \\ g(\psi(\vec{x}) \rightarrow L_0) &= \{[\psi, u_0, \vec{u}](\vec{x}) \rightarrow L_0 \cap R_{u_0, \vec{u}} \mid u_0, \dots, u_n \in \Sigma^*/\equiv\}. \end{aligned}$$

Since there are no terminal reductions involved, a reduction $\bar{\pi}$ is always a reduction associated with an argument-preserving production of type 2.4(i) or 2.4(ii). Therefore, we can define $g(\bar{\pi})$ to be equal to $g(\pi)$.

The ranked alphabet Φ of nonterminals is given by

$$\begin{aligned} \Phi_n &= \{[A, u_0, \vec{u}] \mid A \in \Phi_{(1)n}, u_0, \dots, u_n \in \Sigma^*/\equiv\} \text{ for each } n \ (n \geq 1), \\ \Phi_0 &= \{S_1\} \cup \{[A, u_0] \mid A \in \Phi_{(1)0} - \{S_1\}, u_0 \in \Sigma^*/\equiv\}. \end{aligned}$$

The sets Ψ_n of language names are given for each n ($n \geq 0$) by

$$\Psi_n = \{[\psi, u_0, \vec{u}] \mid \psi \in \Psi_{(1)n}, u_0, \dots, u_n \in \Sigma^*/\equiv\}.$$

Finally, as the control language we take $C = g(C_1)$. □

Although – as remarked before – for the IO-mode closure under finite substitution is unlikely, we have closure under deterministic substitution.

Proposition 4.11. *Let K be a family closed under isomorphism such that $\text{SYMBOL} \cup \{\emptyset\} \subseteq K$. Then $\text{RBLB}_{r, f, \text{IO}}(K)$ is closed under deterministic substitution.*

Proof. Let L_1 be an $\text{RBLB}_{r, f, \text{IO}}(K)$ -language, i.e., $L_1 = L_{r, \text{IO}}(G_1, C_1)$, where $G_1 = (\Phi_{(1)}, \Psi_{(1)}, \Sigma_1, X_1, P_{(1)}, S_1)$. Let $\Sigma_1 = \{a_1, \dots, a_N\}$, and let $\sigma : \Sigma_1 \rightarrow 2^{\Sigma^*}$ be a $\text{RBLB}_{r, f, \text{IO}}(K)$ -substitution, such that for each a in Σ_1 the language $\sigma(a)$ is generated by the $(r, f, \text{IO}, \text{REG}, K)$ -

belb grammar (G_a, C_a) , where $G_a = (\Phi_{(a)}, \Psi_{(a)}, \Sigma_a, X_a, P_{(a)}, S_a)$. We construct an $(r, f, \text{IO}, \text{REG}, K)$ -belb grammar (G, C) with underlying grammar $G = (\Phi, \Psi, \Sigma, X, P, S)$ such that $\sigma(L_1) = L_{r, \text{IO}}(G, C)$.

The construction resembles much to the proof of Proposition 4.8. We use the terminals in Σ_1 of (G_1, C_1) as variables in (G, C) . This is obtained via the isomorphism v which associates with each a in Σ_1 a corresponding variable y_a in X . Each terminal a in Σ_1 which occurs in the K -languages at the right-hand side of the productions of type 2.4(iii) in (G_1, C_1) is replaced by the variable y_a . The original start symbol S_1 in (G_1, C_1) is transformed into S'_1 such that S'_1 has rank N . The new start symbol S of G is used in the new initial production π_0 equal to $S \rightarrow S'_1(\psi_{a_1}, \dots, \psi_{a_N})$. The other productions of (G, C) are obtained from those in (G_1, C_1) by adorning them with additional variables y_{a_1}, \dots, y_{a_N} . A derivation in (G, C) starts with the initial production, followed by a sequence of control strings from the set $\cup\{(\psi_a \rightarrow S_a)C_a \mid a \in \Sigma_1\}$, until we have obtained a term in $\text{Term}(G)$ of the form $S'_1(w_1, \dots, w_N)$, where $w_i \in \sigma(a_i)$ ($1 \leq i \leq N$). From then on we can follow a derivation according to C_1 , where the construction is such that each letter a_i in Σ_1 ($1 \leq i \leq N$) is replaced by a fixed word w_i from $\sigma(a_i)$. The formal construction is as follows.

We assume that the sets $\Phi_{(a)}$ with $a \in \Sigma_1$ are mutually disjoint. Let the sets $\Psi_{(a)}$ and the sets X_a , where a varies over Σ_1 , possess this property too. Then the alphabets Φ and Ψ are defined by

$$\Phi_0 = \{S\} \cup \cup\{\Phi_{(a)0} \mid a \in \Sigma_1\},$$

$$\Phi_n = \cup\{\Phi_{(a)n} \mid a \in \Sigma_1\} \text{ for each } n \text{ with } 1 \leq n < N,$$

$$\Phi_{n+N} = \{A' \mid A \in \Phi_{(1)n}\} \cup \cup\{\Phi_{(a)n+N} \mid a \in \Sigma_1\} \text{ for each } n (n \geq 0),$$

and

$$\Psi_0 = \{\psi_a \mid a \in \Sigma_1\} \cup \cup\{\Psi_{(a)0} \mid a \in \Sigma_1\},$$

$$\Psi_n = \cup\{\Psi_{(a)n} \mid a \in \Sigma_1\} \text{ for each } n \text{ with } 1 \leq n < N,$$

$$\Psi_{n+N} = \{\psi' \mid \psi \in \Psi_{(1)n}\} \cup \cup\{\phi_{n,a} \mid a \in \Sigma_1\} \cup \cup\{\Psi_{(a)n+N} \mid a \in \Sigma_1\} \text{ for each } n (n \geq 0).$$

Define $X = X_1 \cup \{y_a \mid a \in \Sigma_1\} \cup \cup\{X_a \mid a \in \Sigma_1\}$. Let the sets U_1 , U_2 , and U_3 be defined in the following way, where $\vec{x} = (x_1, \dots, x_n)$ and $\vec{y} = (y_{a_1}, \dots, y_{a_N})$.

$$U_1 =$$

$$\{A'(\vec{x}, \vec{y}) \rightarrow B'(\psi'_1(\vec{x}, \vec{y}), \dots, \psi'_k(\vec{x}, \vec{y}), \phi_{n,a_1}(\vec{x}, \vec{y}), \dots, \phi_{n,a_N}(\vec{x}, \vec{y}))\}$$

$$A' \in \Phi_{n+N}, B' \in \Phi_{k+N}, \phi_{n,a} \in \Psi_{n+N}, a \in \Sigma_1, \psi'_i \in \Psi_{n+N}, 1 \leq i \leq k, n \geq 0\},$$

$$U_2 = \{A'(\vec{x}, \vec{y}) \rightarrow \psi'(\vec{x}, \vec{y}) \mid A' \in \Phi_{n+N}, \psi' \in \Psi_{n+N}, n \geq 0\},$$

$$U_3 = \{\psi'(\vec{x}, \vec{y}) \rightarrow v(L_0) \mid \psi' \in \Psi_{n+N}, \psi(\vec{x}) \rightarrow L_0 \in P_{(1)3}, n \geq 0\},$$

where $v : \Sigma_1 \cup X_1 \rightarrow X$ is the isomorphism defined by

$$\begin{aligned} v(x) &= x & \text{if } x \in X_1, \\ v(a) &= y_a & \text{if } a \in \Sigma_1. \end{aligned}$$

Let $U = U_1 \cup U_2 \cup U_3 \cup P_{\Sigma_1} \cup P_\phi$, where $P_{\Sigma_1} = \{S_a \rightarrow \psi_a \mid a \in \Sigma_1\}$ and $P_\phi = \{\phi_{n,a}(\vec{x}, \vec{y}) \rightarrow \{y_a\} \mid a \in \Sigma_1, n \geq 0\}$. Define the regular substitution $g : P_{(1)} \rightarrow 2^{U^*}$ by $g(A(\vec{x})) \rightarrow B(\psi_1(\vec{x}), \dots, \psi_k(\vec{x})) =$

$$\{A'(\vec{x}, \vec{y}) \rightarrow (B'(\Psi_1'(\vec{x}, \vec{y}), \dots, \Psi_k'(\vec{x}, \vec{y}), \Phi_{n,a_1}(\vec{x}, \vec{y}), \dots, \Phi_{n,a_N}(\vec{x}, \vec{y})))P_\phi^* \},$$

$$g(A(\vec{x}) \rightarrow \Psi(\vec{x})) = \{A'(\vec{x}, \vec{y}) \rightarrow \Psi'(\vec{x}, \vec{y})\},$$

$$g(\Psi(\vec{x}) \rightarrow L_0) = \{\Psi'(\vec{x}, \vec{y}) \rightarrow v(L_0)\}.$$

In order to satisfy the condition that G ought to be an (IO, K) -elb grammar, we define the set P_\emptyset to be equal to $\{\Psi_a \rightarrow \emptyset \mid a \in \Sigma_1\}$. Furthermore, if π is in $P_{(1)1} \cup P_{(1)2}$, then $g(\bar{\pi})$ is defined as $g(\pi) = g(\bar{\pi})$.

Let $\pi_0 = S \rightarrow S'_1(\Psi_{a_1}, \dots, \Psi_{a_N})$. Then P is defined by

$$P = \cup \{P_{(a)} \mid a \in \Sigma_1\} \cup \{\Psi'(\vec{x}, \vec{y}) \rightarrow v(L_0) \mid (\Psi(\vec{x}) \rightarrow L_0) \in P_{(1)3}\} \cup P_{\Sigma_1} \cup P_\phi \cup \{\pi_0\} \cup g(P_{(1)1} \cup P_{(1)2}) \cup P_\emptyset.$$

Finally, as the control language we take C equal to

$$\pi_0 \{(\Psi_a \rightarrow S_a) C_a \mid a \in \Sigma_1\}^* g(C_1). \quad \square$$

Remark that it follows from Proposition 4.11 that the language family $RBLB_{r,f,IO}(K)$ is closed under homomorphism in case K is closed under isomorphism.

Recall that a language family is a *full Quasi Abstract Family of Languages* or *full Q AFL* [2], if it is a family that contains at least one *SYMBOL*-language and that is closed under the regular operations (union, concatenation, and Kleene *), intersection with regular languages, and homomorphism.

Corollary 4.12. *Let K be a family with $K \supseteq \text{SYMBOL}$, and let K be closed under left or right-marking, intersection with regular languages, and isomorphism. Then the family $RBLB_{r,f,IO}(K)$ is a full Q AFL closed under deterministic substitution.* \square

5. Generating Power of (r, f, m, REG, K) -belb Grammars

In this section we determine a lower bound on the language generating capacity of (r, f, m, REG, K) -belb grammars. First, we establish some results which are analogously to the non-bidirectional (unidirectional) case. Let the family $BLB_{r,f,m}(K)$ denote the family of languages generated by (r, f, m, REG, K) -belb grammars (G, C) in which $C = (P \cup \bar{P})^*$. Such uncontrolled bidirectional grammars are called (r, f, m, K) -belb grammars in the sequel. Consequently, the control language C will be omitted in the pair (G, C) , so that we denote such a grammar by G only.

Lemma 5.1.

$$(i) \quad RBLB_{r,f,IO}(\emptyset NE) = RBLB_{r,f,IO}(FIN).$$

$$(ii) \quad BLB_{r,f,IO}(\emptyset NE) = BLB_{r,f,IO}(FIN).$$

$$(iii) \quad \text{For each family } K, \text{ we have } BLB_{r,f,m}(K) \subseteq RBLB_{r,f,m}(K), \text{ where either } m = \text{OI} \text{ or } m = \text{IO}.$$

Proof. (i). The inclusion from left to right is obvious. The converse inclusion can be shown by replacing each production $\Psi(x_1, \dots, x_n) \rightarrow L_0$ in an (r, f, IO, REG, FIN) -belb grammar by productions $\Psi_i(x_1, \dots, x_n) \rightarrow \{\eta_i\}$, where it is understood that $L_0 = \{\eta_1, \dots, \eta_k\}$ for some $k \geq 0$. Here $k = 0$ means that $L_0 = \emptyset$, in which case no replacement ought to be made. As a consequence, each rule ρ in which $\Psi(\vec{x})$ occurs l -times has to be replaced by k^l rules covering all combinations of Ψ_i 's ($1 \leq i \leq k$) possible in ρ . The corresponding alterations in the control language C are

allowed, for REG is closed under finite substitution.

(ii) and (iii). Obvious. \square

In the following proposition we show that the family IO is included in $BLB_{r,f,IO}(\emptyset NE)$. With respect to an m -macro grammar G equal to (Φ, Σ, X, P, S) we define for each $A \in \Phi$ the finite (possibly empty) language $L_{A,G}$ over $\Sigma \cup X$ by

$$L_{A,G} = \{\eta \in (\Sigma \cup X)^* \mid \exists \pi \in P \bullet \pi = A(\vec{x}) \rightarrow \eta\}.$$

Note that $L_{A,G}$ does not depend on the mode m .

Proposition 5.2. *The family IO of IO -macro languages is included in the families $BLB_{r,f,IO}(\emptyset NE)$ and $RBLB_{r,f,IO}(\emptyset NE)$.*

Proof. Let L_0 be an IO -macro language generated by the grammar $G_1 = (\Phi_{(1)}, \Sigma, X, P_{(1)}, S)$. We assume G_1 is in IO standard form [10]; i.e., each production is argument-preserving and it has either the form

(i) $A(x_1, \dots, x_n) \rightarrow B(D(y_1, \dots, y_l), z_2, \dots, z_k)$, where $A \in \Phi_{(1)n}$, $B \in \Phi_{(1)k}$

($k \geq 1$), $D \in \Phi_{(1)l}$ and $y_1, \dots, y_l, z_2, \dots, z_k \in X$, or

(ii) $A(x_1, \dots, x_n) \rightarrow \eta$, where $\eta \in (\Sigma \cup X)^*$.

We construct an (r, f, IO, FIN) -belb grammar G with underlying grammar $G = (\Phi, \Psi, \Sigma, X, P, S)$ such that $L_{r,IO}(G) = L_{IO}(G_1)$ as follows. Starting from $P = \emptyset$, for each production π in $P_{(1)}$ we add to P a sequence of productions. If π is of the form (i), then we add productions $p_{ABD\pi}$, $p_{D'\pi D}$ and $p_{D'\pi}$ to P , where $D' \in \Phi_n$ ($n \geq 0$) and

$$p_{ABD\pi} = A(x_1, \dots, x_n) \rightarrow B(\Psi_{D\pi}(\vec{x}), \Psi_{z_2}(\vec{x}), \dots, \Psi_{z_k}(\vec{x})),$$

$$p_{D'\pi D} = D'(x_1, \dots, x_n) \rightarrow D(\Psi_{y_1}(\vec{x}), \dots, \Psi_{y_l}(\vec{x})),$$

$$p_{D'\pi} = D'(x_1, \dots, x_n) \rightarrow \Psi_{D\pi}(x_1, \dots, x_n).$$

If π is of the form (ii), then we add to P the productions π_A and π_A'' , where

$$\pi_A = A(x_1, \dots, x_n) \rightarrow \Psi_A(x_1, \dots, x_n),$$

$$\pi_A'' = \Psi_A(x_1, \dots, x_n) \rightarrow L_{A,G_1}.$$

Furthermore, we add to P the elements of the set P_X , consisting of all productions $\Psi_x(x_1, \dots, x_n) \rightarrow \{x\}$, with $x \in X$ and x occurs in \vec{x} . A nonterminal D ought to be expanded by the corresponding language name Ψ_D . Therefore, we add to P all productions $\Psi_{D\pi}(\vec{x}) \rightarrow \emptyset$, in case D occurs in the argument list of the right-hand side of a nested production π in G_1 . From the construction of P one can easily determine Φ and Ψ . We observe that a production of the form (i) is simulated in G by some element of $\{p_{ABD\pi}\}P_X^*\{\bar{p}_{D'\pi}p_{D'\pi D}\}P_X^*$. In addition, a production of the form (ii) is simulated in G by $\pi_A\pi_A''$. However, it is not necessary to provide G with a control language in order to generate the language L_0 . The correct order of application is arranged implicitly by the derivation mode. Therefore, we can take for C the trivial control language $(P \cup \bar{P})^*$.

So far we have shown that $IO \subseteq BLB_{r,f,IO}(FIN)$. The conclusion now follows from Lemma 5.1. \square

Corollary 5.3. *If K is a language family that includes $\emptyset NE$, then the families $BLB_{r,f,IO}(K)$ and $RBLB_{r,f,IO}(K)$ both contain all IO -macro languages.* \square

For m equal to OI an analogous result holds.

Proposition 5.4. *The family OI of OI-macro languages is included in the families $BLB_{r,f,OI}(\emptyset NE)$ and $RBLB_{r,f,OI}(\emptyset NE)$.*

Proof. Let L_0 be an OI-macro language generated by the grammar $G_1 = (\Phi_{(1)}, \Sigma, X, P_{(1)}, S)$. Assume G_1 is in OI standard form [10]; that means that each production has either the form

$$(i) \quad A(x_1, \dots, x_n) \rightarrow B(D_1(x_1, \dots, x_n), \dots, D_k(x_1, \dots, x_n)), \text{ with } k, n \geq 0,$$

or

$$(ii) \quad A(x_1, \dots, x_n) \rightarrow \eta, \quad \text{where } \eta \in (\Sigma \cup X)^* \text{ and } n \geq 0.$$

Furthermore, we assume that in G_1 the symbol S only occurs at the left-hand side of productions of the form (i). This is no loss of generality, since we can eventually transform the grammar G_1 into the equivalent grammar G_2 equal to $(\Phi_{(2)}, \Sigma, X_{(2)}, P_{(2)}, S')$, where

$$\Phi_{(2)} = \Phi_{(1)} \cup \{S', S''\}, \quad X' = X \cup \{x\}, \text{ and}$$

$$P_{(2)} = P_{(1)} \cup \{S' \rightarrow S''(S), S''(x) \rightarrow x\}.$$

Analogously to Proposition 5.2 we construct an $(r, f, \text{OI}, \emptyset NE)$ -belb grammar G with $G = (\Phi, \Psi, \Sigma, X, P, S)$ such that $L_{r,\text{OI}}(G) = L_{\text{OI}}(G_1)$ as follows. For each nested production π in $P_{(1)}$ of the form (i), we add a corresponding production π' to P , where π' is defined by

$$A(\vec{x}) \rightarrow B(\psi_{D_1}(\vec{x}), \dots, \psi_{D_k}(\vec{x})).$$

Define $P'_{(1)i}$ by $P'_{(1)i} = \{\pi' \mid \pi \in P_{(1)}, \pi \text{ is of the form (i)}\}$.

Let Θ be the set of all nonterminals in $\Phi_{(1)}$ that occur in the argument list at the right-hand side of a nested production of the form (i). For each D in Θ we introduce a language name ψ_D and a production of type 2.4(ii), viz. $D(\vec{x}) \rightarrow \psi_D(\vec{x})$. In addition, we define for each such D a production of type 2.4(iii) by $\psi_D(\vec{x}) \rightarrow \emptyset$. The two sets of all productions of the form 2.4(ii) and 2.4(iii) obtained in this way are denoted by P_Θ and P_\emptyset , respectively. Thus

$$P_\Theta = \{D(\vec{x}) \rightarrow \psi_D(\vec{x}) \mid D \in \Phi_{(1)}\}, \text{ and}$$

$$P_\emptyset = \{\psi_D(\vec{x}) \rightarrow \emptyset \mid D \in \Phi_{(1)}\}.$$

Next, we define P by

$$P = P'_{(1)i} \cup P_\Theta \cup P_\emptyset \cup \{A(\vec{x}) \rightarrow \psi_\eta(\vec{x}), \psi_\eta(\vec{x}) \rightarrow \{\eta\} \mid A(\vec{x}) \rightarrow \eta \in P_{(1)}\}.$$

We take the set of nonterminals Φ equal to $\Phi_{(1)}$ and the set of language names Ψ is defined by

$$\Psi_n = \{\psi_\eta \mid A(\vec{x}) \rightarrow \eta \in P_{(1)}, A \neq S\} \cup \{\psi_D \mid D \in \Theta \cap \Phi_{(1)n}\},$$

for each n ($n \geq 0$).

The application of a nested production of the form (i) is simulated by the corresponding production $A(\vec{x}) \rightarrow B(\psi_{D_1}(\vec{x}), \dots, \psi_{D_k}(\vec{x}))$ in G . In case a language name ψ_D percolates at top level, it has to be rewritten into the corresponding nonterminal D . Thus a terminal production π equal to $B(\vec{x}) \rightarrow \eta$, with $\eta \in (\Sigma \cup X)^*$, in G_1 is simulated by the sequence of rules

$$(B(\vec{x}) \rightarrow \psi_\eta(\vec{x}))(\psi_\eta(\vec{x}) \rightarrow \{\eta\})(\bar{P}_\Theta \cup \{\lambda\}).$$

The additional $\{\lambda\}$ in this sequence is necessary to cover the case in which B has no arguments. Note that the trivial control language suffices, i.e., we can take C equal to $(P \cup \bar{P})^*$. \square

Corollary 5.5. *If K is a language family that includes \emptyset NE, then the families $BLB_{r,f,OI}(K)$ and $RBLB_{r,f,OI}(K)$ both contain all OI-macro languages. \square*

In the remaining part of this section we show that the language family $RBLB_{r,f,OI}(OI)$ equals the family OI; cf. Theorem 5.13.

Let G be an OI-macro grammar. Then we define the language $L_{r,OI}(G)$ over Σ^* by

$$L_{r,OI}(G) = \{w \in \Sigma^* \mid S \Rightarrow_{r,OI}^* w\},$$

where $\alpha \Rightarrow_{r,OI} \beta$ holds if and only if β is obtained from α by a single right-most OI-derivation step. The strings α and β are terms over the alphabet of G . For IO and OI-macro grammars, the right-most derivation relation can be defined analogously to Definition 2.6. An OI-macro grammar provided with right-most rewriting will be called an (r,OI) -macro grammar, and the language $L_{r,OI}(G)$ will be called an (r,OI) -macro language. Let OI_r denote the family of (r,OI) -macro languages. In addition, let $OI_r(REG)$ denote the family of languages generated by regularly controlled (r,OI) -macro grammars. Then we can prove the following result.

Proposition 5.6. *The family $RBLB_{r,f,OI}(OI)$ is included in the family $OI_r(REG)$.*

Proof. Let L_0 be generated by the (r,f,OI,REG,OI) -belb grammar (G_1, C_1) , where $G_1 = (\Phi_{(1)}, \Psi, \Sigma, X_{(1)}, P_{(1)}, S)$. We construct a regularly controlled OI-macro grammar (G, C) with $G = (\Phi, \Sigma, X, P, S)$ such that $L_0 = L_{r,OI}(G, C)$. Starting with $P = \emptyset$ we add for each rule in $P_{(1)} \cup \bar{P}_{(1)}$ one or more productions to P as follows. If $\rho \in P_{(1)2} \cup \bar{P}_{(1)2} \cup P_{(1)1}$, then ρ is added to P . If ρ is in $P_{(1)3}$, then ρ is of the form $\psi(x_1, \dots, x_n) \rightarrow L_\psi$. Let for each language name ψ in Ψ the language L_ψ be generated by some OI-macro grammar $G_\psi = (\Phi_\psi, \Sigma \cup \{x_1, \dots, x_n\}, Y_\psi, P_\psi, S_\psi)$. We assume that the alphabets of this finite number of grammars G_ψ are mutually disjoint. Then we add each production in P'_ψ to P , where P'_ψ equals $\{A'(\vec{y}^\rightarrow, \vec{x}^\rightarrow) \rightarrow t \mid A(\vec{y}^\rightarrow) \rightarrow t \in P_\psi\}$, and we define $P'_\Psi = \cup \{P'_\psi \mid \psi \in \Psi\}$. Next, we add to P productions $\psi(\vec{x}^\rightarrow) \rightarrow S_\psi(\vec{x}^\rightarrow)$, where $\psi \in \Psi$. Finally, if the rule ρ is in $\bar{P}_{(1)1}$, then ρ is a reduction of the form $B(\psi_1(\vec{x}^\rightarrow), \dots, \psi_k(\vec{x}^\rightarrow)) \rightarrow A(\vec{x}^\rightarrow)$. First consider the case that A is in $\Phi_{(1)0}$. Then the production π_{BA} equal to $B(\vec{x}^\rightarrow) \rightarrow A$ is added to P . Secondly, if A is in $\Phi_{(1)n}$ ($n \geq 1$), then we add productions π_{B1} equal to $B(\vec{x}^\rightarrow) \rightarrow x_1$ (Remember that $k \geq 1$.) and $\pi_{\psi_1 A}$ equal to $\psi_1(\vec{x}^\rightarrow) \rightarrow A(\vec{x}^\rightarrow)$ to P .

Now we define P to be equal to

$$\begin{aligned} & P_{(1)2} \cup \bar{P}_{(1)2} \cup P_{(1)1} \cup \cup \{\psi(\vec{x}^\rightarrow) \rightarrow S_\psi(\vec{x}^\rightarrow) \mid \psi(\vec{x}^\rightarrow) \rightarrow L_\psi \in P_{(1)3}\} \cup P'_\Psi \\ & \cup \{\pi_{\psi_1 A} \mid \exists B \in \Phi. \exists \psi_2, \dots, \exists \psi_k \in \Psi. A(\vec{x}^\rightarrow) \rightarrow B(\psi_1(\vec{x}^\rightarrow), \dots, \psi_k(\vec{x}^\rightarrow)) \in P_{(1)}\} \\ & \cup \{\pi_{B1} \mid \exists A \in \Phi. \exists \psi_1, \dots, \exists \psi_k \in \Psi. A(\vec{x}^\rightarrow) \rightarrow B(\psi_1(\vec{x}^\rightarrow), \dots, \psi_k(\vec{x}^\rightarrow)) \in P_{(1)}\} \\ & \cup \{\pi_{BA} \mid \exists \psi_1, \dots, \exists \psi_k \in \Psi. A(\vec{x}^\rightarrow) \rightarrow B(\psi_1(\vec{x}^\rightarrow), \dots, \psi_k(\vec{x}^\rightarrow)) \in P_{(1)}\}. \end{aligned}$$

Next define the regular substitution $\sigma : (P_{(1)} \cup \overline{(P_{(1)} - P_{(1)3})})^* \rightarrow 2^{P^*}$ by

$$\begin{aligned} \sigma(A(\vec{x}^\rightarrow) \rightarrow \psi(\vec{x}^\rightarrow)) &= \{A(\vec{x}^\rightarrow) \rightarrow \psi(\vec{x}^\rightarrow)\}, \\ \sigma(\psi(\vec{x}^\rightarrow) \rightarrow A(\vec{x}^\rightarrow)) &= \{\psi(\vec{x}^\rightarrow) \rightarrow A(\vec{x}^\rightarrow)\}, \\ \sigma(\psi(\vec{x}^\rightarrow) \rightarrow L_\psi) &= \{\psi(\vec{x}^\rightarrow) \rightarrow S_\psi(\vec{x}^\rightarrow)\} P'^*_\Psi, \\ \sigma(A(\vec{x}^\rightarrow) \rightarrow B(\psi_1(\vec{x}^\rightarrow), \dots, \psi_k(\vec{x}^\rightarrow))) &= \{A(\vec{x}^\rightarrow) \rightarrow B(\psi_1(\vec{x}^\rightarrow), \dots, \psi_k(\vec{x}^\rightarrow))\}, \\ \sigma(B(\psi_1(\vec{x}^\rightarrow), \dots, \psi_k(\vec{x}^\rightarrow)) \rightarrow A(\vec{x}^\rightarrow)) &= \{\pi_{B1} \pi_{\psi_1 A}\}, \\ \sigma(B(\psi_1, \dots, \psi_k) \rightarrow A) &= \{\pi_{BA}\}. \end{aligned}$$

From P we obtain Φ and X in a straightforward way. It easily follows from the construction of G that $L_0 = L_{r, \text{OI}}(G, \sigma(C_1))$. \square

Let the number of occurrences of a symbol σ in a word w be denoted by $\#_\sigma(w)$.

Example 5.7. Consider the $(r, f, \text{OI}, \text{REG}, \emptyset \text{NE})$ -belb grammar (G, C) of Example 3.9, where $G = (\Phi, \Psi, \{0, 1\}, X, P, S)$, $X = \{x\}$, $C = (P \cup \bar{P})^*$, and P consists of

$$\begin{array}{ll} \pi_0 = S \rightarrow A(\psi_1), & \pi_6 = B \rightarrow \psi_1, \\ \pi_1 = A(x) \rightarrow A(\psi_2(x)), & \pi_7 = B \rightarrow D(\psi_1), \\ \pi_2 = A(x) \rightarrow \psi_3(x), & \pi_8 = D(x) \rightarrow \psi_4(x), \\ \pi_3 = \psi_3(x) \rightarrow \{x\}, & \pi_9 = \psi_4(x) \rightarrow \{0x\}, \\ \pi_4 = \psi_2(x) \rightarrow \{xx\}, & \pi_{10} = D(x) \rightarrow \psi_5(x), \\ \pi_5 = \psi_1 \rightarrow \{1\}, & \pi_{11} = \psi_5(x) \rightarrow \{x0\}. \end{array}$$

According to Section 3, we have that $P_1 = \{\pi_0, \pi_1, \pi_7\}$, $P_2 = \{\pi_2, \pi_6, \pi_8, \pi_{10}\}$, and $P_3 = P - (P_1 \cup P_2)$. Now we replace production π_5 by the production $\pi'_5 = \psi_1 \rightarrow L_{\psi_1}$, where

$$L_{\psi_1} = \{w \in \{a, 1\}^+ \mid \#_1(w) = 2^k, k \geq 0\}.$$

The resulting grammar (G', C') is an $(r, f, \text{OI}, \text{REG}, \text{OI})$ -belb grammar, where $G' = (\Phi, \Psi, \{0, 1, a\}, X, P', S)$, with $P' = P \cup \{\pi'_5\} - \{\pi_5\}$ and C' is the resulting variant of C . It is easy to see that the language generated by this grammar equals

$$L_1 = \{w \in \{0, 1, a\}^+ \mid w = w_1 \dots w_k, k = 2^l, l \geq 0, w_i \text{ is in } 0^* a^* (1a^*)^{m_i} 0^*, \\ m_i = 2^{l_i}, l_i \geq 0\}.$$

The language L_{ψ_1} can be generated by the OI-macro grammar $G_{\psi_1} = (\Phi_{\psi_1}, \{1, a\}, Y, P_{\psi_1}, S_{\psi_1})$, where $\Phi_{\psi_1} = \{S_{\psi_1}, F, H\}$, $Y = \{y\}$, and P_{ψ_1} consists of the productions

$$\begin{array}{lll} S_{\psi_1} \rightarrow F(H), & F(y) \rightarrow y, & H \rightarrow Ha, \\ F(y) \rightarrow F(yy), & H \rightarrow aH, & H \rightarrow 1. \end{array}$$

The languages $\{u_i\}$, with $\psi_i(x) \rightarrow \{u_i\}$ ($2 \leq i \leq 5$), can be generated by OI-macro grammars which have a single production $S_{\psi_i} \rightarrow u_i$ ($2 \leq i \leq 5$).

Using the construction given in the proof of Proposition 5.6, we obtain the following regularly controlled (r, OI) -macro grammar (G_1, C_1) that generates L_1 , where $G_1 = (\Phi_{(1)}, \Sigma, X_{(1)}, P_{(1)}, S)$, and $P_{(1)}$ is formed by

$$P_1 \cup P_2 \cup \bar{P}_2 \cup \{\psi_i(x) \rightarrow S_{\psi_i}(x) \mid 2 \leq i \leq 5\} \cup \{S_{\psi_i}(x) \rightarrow u_i \mid 2 \leq i \leq 5\} \cup P_{\psi_1} \cup \\ \cup \{\psi_1 \rightarrow S_{\psi_1}, A(x) \rightarrow S, D(x) \rightarrow B, A(x) \rightarrow x, \psi_2(x) \rightarrow A(x)\}.$$

We define the regular substitution $\sigma : (P_{(1)} \cup \overline{(P_{(1)} - P_{(1)3})})^* \rightarrow 2^{P^*}$ by

$$\sigma(\rho) = \{\rho\} \text{ for each } \rho \in P_{(1)} \cup P_{(2)} \cup \bar{P}_{(2)},$$

$$\sigma(\psi_1 \rightarrow L_{\psi_1}) = \{\psi_1 \rightarrow S_{\psi_1}\} P_{\psi_1}^*,$$

$$\sigma(\psi_i(x) \rightarrow L_{\psi_i}) = \{(\psi_i(x) \rightarrow S_{\psi_i})(S_{\psi_i} \rightarrow u_i)\} \text{ for each } i (2 \leq i \leq 5),$$

$$\sigma(A(\psi_1) \rightarrow S) = \{A(x) \rightarrow S\},$$

$$\sigma(D(\psi_1) \rightarrow B) = \{D(x) \rightarrow B\},$$

$$\sigma(A(\psi_2(x)) \rightarrow A(x)) = \{(A(x) \rightarrow x)(\psi_2(x) \rightarrow A(x))\}.$$

Finally, $\Phi_{(1)} = \Phi \cup \Psi \cup \Phi_{\Psi_i} \cup \{S_{\Psi_i} \mid 2 \leq i \leq 5\}$, $X_{(1)} = X \cup Y$ and as the control language we define $C_1 = \sigma(C')$. \square

Next we give a characterization of the family $OI_r(REG)$. This is achieved by a proof method of Ginsburg and Spanier [11], who showed that the family of languages generated by regularly controlled context-free grammars provided with left-most derivation equals the family of context-free languages. First we give the obvious right-most version of Theorem 2.1 from [11] which is formulated in our notation.

Theorem 5.8. [11]. *Let K be a family of languages closed under λ -free regular substitution. Then the family of languages generated by K -controlled context-free grammars provided with right-most rewriting equals the family of all languages of the form $h(L_1 \cap L_2)$, where L_1 is in K , L_2 is a context-free language, and h is a homomorphism.* \square

If we replace in Theorem 5.8 “context-free” by “OI-macro” everywhere, then an analogous statement still holds; cf. Theorem 5.11. The proof according to [11] of Theorem 5.8 uses closure under inverse homomorphism of the family of context-free languages. Although this closure property does hold for the family OI [10], it is sufficient to have closure under isomorphism. To prove Theorem 5.11 we need the following two lemmas.

Lemma 5.9. $OI_r = OI$.

Proof. The proof is analogously to the context-free case, which is well known; cf. for instance [19]. \square

For each OI-macro grammar $G = (\Phi, \Sigma, X, P, S)$ we define the OI-macro grammar $H(G)$ by $(\Phi, \Sigma_1, X, P_1, S)$, where $\Sigma_1 = \Sigma \cup P$ and

$$P_1 = \{A(\vec{x}) \rightarrow \eta \pi \mid \pi \in P, \pi = A(\vec{x}) \rightarrow \eta\}.$$

Lemma 5.10. *Let G be an OI-macro grammar (Φ, Σ, X, P, S) and C be a control language over P . Then there exists a λ -free regular substitution τ and a homomorphism h such that $L_{r, OI}(G, C) = h(L_{r, OI}(H(G)) \cap \tau(C^R))$.*

Proof. Similarly to the proof of Lemma 2.1 in [11]. Viz., the homomorphism $h : \Sigma_1^* \rightarrow \Sigma^*$ is defined by $h(\pi) = \lambda$ if $\pi \in P$, and $h(a) = a$ if $a \in \Sigma$. The λ -free regular substitution τ is defined by $\tau(\pi) = \Sigma^* \{\pi\} \Sigma^*$, for each $\pi \in P$. \square

The following theorem has been adapted from Theorem 2.1 in [11].

Theorem 5.11. *Let K be a family of languages closed under reversal and λ -free regular substitution. Then the family of languages generated by K -controlled OI-macro grammars provided with right-most rewriting equals the family of all languages of the form $h(L_1 \cap L_2)$, where L_1 is in K , L_2 is an OI-macro language, and h is a homomorphism.*

Proof. By Lemma 5.10 there exists for each control language C in K and each OI-macro grammar G a λ -free regular substitution τ and a homomorphism h such that $L_{r, OI}(G, C) = h(L_{r, OI}(H(G)) \cap \tau(C^R))$. Then $\tau(C^R)$ is in K . The language $L_{r, OI}(H(G))$ is an OI-macro language (Lemma 5.9), so $L_{r, OI}(G, C)$ has the proper form.

Conversely, let L_1 and L_2 be languages over Σ_1 , where L_1 is a language in K and L_2 an OI-macro language. Let $h : \Sigma_1^* \rightarrow \Sigma_2^*$ be a homomorphism.

We may assume $\Sigma_1 \cap \Sigma_2 = \emptyset$, which is shown as follows. Let \bar{a} be a distinct symbol not in Σ_2 for each a in Σ_1 , and let $\bar{\Sigma}_1 = \{\bar{a} \mid a \in \Sigma_1\}$. Let h_1 be the isomorphism from $\bar{\Sigma}_1$ into Σ_1 defined by $h_1(\bar{a}) = a$ for each \bar{a} in $\bar{\Sigma}_1$. Then h_1^{-1} is an isomorphism too. Let $\bar{L}_1 = h_1^{-1}(L_1)$ and $\bar{L}_2 = h_1^{-1}(L_2)$. Then $\bar{\Sigma}_1 \cap \Sigma_2 = \emptyset$, \bar{L}_1 is in K , \bar{L}_2 is an OI-macro language [10], hh_1 is a

homomorphism and $h(L_1 \cap L_2) = hh_1(\bar{L}_1 \cap \bar{L}_2)$.

Now let $G_1 = (\Phi_1, \Sigma_1, X_1, P_1, S)$ be an OI-macro grammar that generates L_2 . Let $G_2 = (\Phi_2, \Sigma_2, X_1, P_2, S)$ be the OI-macro grammar with $\Phi_2 = \Phi_1 \cup \Sigma_1$ and $P_2 = P_1 \cup \{a \rightarrow h(a) \mid a \in \Sigma_1\}$. Let the production π_a be equal to $a \rightarrow h(a)$ for each a in Σ_1 . The homomorphism $h_3 : P_2^* \rightarrow \Sigma_1^*$ is defined by $h_3(\pi) = \lambda$ for $\pi \in P_1$ and $h_3(\pi_a) = a$ for a in Σ_1 . Then $h_3^{-1} : \Sigma_1^* \rightarrow 2^{P_2^*}$ is a λ -free regular substitution, $h_3^{-1}(L_1^R)$ is in K and $L_{r, \text{OI}}(G_2, h_3^{-1}(L_1^R)) = h(L_1 \cap L_2)$. Hence a language of the form $h(L_1 \cap L_2)$, with L_1 in K and L_2 an OI-macro language, can be generated by some K -controlled (r, OI) -macro grammar (G, C) . \square

Corollary 5.12. *The family $\text{OI}_r(\text{REG})$ equals the family OI .*

Proof. Recall that the family OI is closed under intersection with regular sets and under homomorphism [10]. \square

Theorem 5.13. *The language families $\text{RBLB}_{r,f, \text{OI}}(\text{OI})$ and $\text{BLB}_{r,f, \text{OI}}(\text{OI})$ are equal to the language family OI .*

Proof. Lemma 5.1(iii), Corollaries 5.5 and 5.12, Propositions 5.4 and 5.6. \square

Corollary 5.14. $\text{RBLB}_{r,f, \text{OI}}(\emptyset\text{NE}) = \text{RBLB}_{r,f, \text{OI}}(\text{OI}) = \text{OI}$.

Proof. Corollary 5.5 and Theorem 5.13. \square

Of course, a similar statement holds for any family of languages K which satisfies $\emptyset\text{NE} \subseteq K \subseteq \text{OI}$.

On the other hand, Corollary 5.14 may also be considered as a closure property of the family OI , viz., $\text{RBLB}_{r,f, \text{OI}}(\text{OI}) \subseteq \text{OI}$. Whether this property is stronger or weaker than the one established in [6] for the family OI remains open.

Now we show that the family $\text{BLB}_{r,f, \text{IO}}(\emptyset\text{NE})$ differs from the family $\text{BLB}_{r,f, \text{OI}}(\emptyset\text{NE})$.

Proposition 5.15. *The family $\text{BLB}_{r,f, \text{IO}}(\emptyset\text{NE})$ is not equal to the family OI .*

Proof. The language $L_0 = \{1^m(c1^m)^n \mid n = 2^m - 1, m \geq 0\}$ is an IO-macro language which is not an OI-macro language [10]. The language L_0 can be generated by the $(r, f, \text{IO}, \text{REG}, \emptyset\text{NE})$ -elb grammar (G, C) , where $G = (\Phi, \Psi, \Sigma, X, P, S)$ is defined by $\Phi = \{S, D, F, G\}$, $\Psi = \{\psi_i \mid 0 \leq i \leq 4\}$, $\Sigma = \{1, c\}$, $X = \{x\}$, and P consists of

$$\begin{aligned} \pi_0 &= S \rightarrow F(\psi_0), & \pi_6 &= \psi_1(x) \rightarrow \emptyset, \\ \pi_1 &= \psi_0 \rightarrow \{1\}, & \pi_7 &= G(x) \rightarrow \psi_3(x), \\ \pi_2 &= F(x) \rightarrow G(\psi_1(x)), & \pi_8 &= \psi_2(x) \rightarrow \{x1\}, \\ \pi_3 &= F(x) \rightarrow G(\psi_4(x)), & \pi_9 &= \psi_3(x) \rightarrow \{xcx\}, \\ \pi_4 &= D(x) \rightarrow \psi_1(x), & \pi_{10} &= \psi_4(x) \rightarrow \{x\}. \\ \pi_5 &= D(x) \rightarrow F(\psi_2(x)), \end{aligned}$$

Finally, take as the control language the trivial control language, i.e., $C = (P \cup \bar{P})^*$. That (G, C) generates L_0 can be shown in the following way. First, a term τ_m of the form $G(G(\dots G(F(1^m))\dots))$ is produced, where τ_m contains exactly $m-1$ symbols G ($m \geq 1$). This can be performed by a control word $\pi_0\pi_1(\pi_2\pi_4\pi_5\pi_8)^{m-1}$. Then applying $\pi_3\pi_{10}$, followed by repeatedly applying $\pi_7\pi_9$ yields the string $1^m(c1^m)^{2^m-1}$. \square

Proposition 5.15 shows a typical consequence of bidirectional rewriting. In case of unidirectional rewriting we have that with $K = \emptyset\text{NE}$ for both modes OI and IO, linear basic grammars have the same generating power; viz. they both generate the linear basic languages. The latter equality can also be expressed in terms of (m, K) -elb languages, i.e., let LB denote the family of linear basic languages. Then we have $\text{LB}_{\text{OI}}(\emptyset\text{NE}) = \text{LB}_{\text{IO}}(\emptyset\text{NE}) = \text{LB}$ ($= \text{EDTOL}$, [6]).

Due to the presence of bidirectional rewriting in (m, REG, K) -belb grammars we have that the family $BLB_{r,f, OI}(\emptyset NE)$ differs from the family $BLB_{r,f, IO}(\emptyset NE)$.

6. Free Rewriting of Nonterminals and Language Names

In this section we study another grammatical model that can be derived from (m, REG, K) -belb grammars. It is natural to investigate also (m, REG, K) -belb grammars provided with the (usual) derivation relation which models the free application of rules from the grammar; i.e., the restriction of right-most rewriting will be dropped in this section. We maintain the restriction of disallowing terminal reductions. Then we prove that the corresponding language family $RBLB_{f,m}(K)$ equals the family of recursively enumerable languages for $m = IO$ (Proposition 6.3) and for $m = OI$ (Proposition 6.4), provided some minor conditions on the family K hold.

Definition 6.1. Let (G, C, ϕ) be an (m, REG, K) -belb grammar, where $G = (\Phi, \Psi, \Sigma, X, P, S)$. Let ρ be rule from $P \cup \bar{P}$, and σ, τ be terms in $Term(G, \phi)$. We write $\sigma \Rightarrow_m^\rho \tau$ if

- either ρ is a production, σ is rewritten by ρ , and $\sigma \Rightarrow_m \tau$,
- or ρ is a reduction, τ is rewritten by $\bar{\rho}$, and $\tau \Rightarrow_m \sigma$.

In addition, let c be a control word in $C \subseteq (P \cup \bar{P})^*$, with $c = \rho_1 \dots \rho_n$ ($n \geq 0$, $\rho_i \in P \cup \bar{P}$, $0 \leq i \leq n$). Then \Rightarrow_m^c is defined (as usual) for terms τ and τ' from $Term(G, \phi)$ by $\tau \Rightarrow_m^c \tau'$ if there are terms τ_i ($0 \leq i \leq n$) such that $\tau_0 = \tau$, $\tau_n = \tau'$ and for each i ($0 \leq i < n$), $\tau_i \Rightarrow_m^{\rho_{i+1}} \tau_{i+1}$ holds.

In case a rule ρ_i in c is not applicable to the term τ_i , then further application of rules is blocked, and the application of c to τ yields no result, i.e., there is no term τ' such that $\tau \Rightarrow_m^c \tau'$ is defined. \square

In this section an (m, REG, K) -belb grammar will be provided with the derivation relation introduced in Definition 6.1.

Definition 6.2. If (G, C, ϕ) is an (m, REG, K) -belb grammar where $G = (\Phi, \Psi, \Sigma, X, P, S)$ is its underlying grammar, then the *language* generated by (G, C, ϕ) is

$$L_m(G, C, \phi) = \{w \in \Sigma^* \mid \exists c \in C \cdot S \Rightarrow_m^c w\},$$

where either $m = OI$ or $m = IO$. The family of languages generated by (m, REG, K) -belb grammars is denoted by $RBLB_m(K)$. \square

As in Section 2 and 3, only (m, REG, K) -belb grammars without terminal reductions will be studied. Such grammars are called (f, m, REG, K) -belb grammars, and their associated family of languages will be denoted by $RBLB_{f,m}(K)$. The next propositions characterize – under minor conditions on the family K – the families $RBLB_{f, IO}(K)$ and $RBLB_{f, OI}(K)$; viz. these families equal the family of recursively enumerable languages. The proofs are inspired by the equivalence of Turing machines and on-line acceptors provided with two pushdown stores as auxiliary storage. We refer to [13] for a precise definition of the concept of the Turing-machine as well as related notions and terminology. Given a Turing machine A and a mode m , we construct an (f, m, REG, K) -belb grammar (G, C) which simulates computations of A , using an encoding of two pushdown stores in the derivation tree. These simulations (Propositions 6.3 and 6.4) heavily rely on the use of reductions, which will be no surprise, in view of the main result of [4].

Let RE denote the family of recursively enumerable languages.

Proposition 6.3. *Let K be a family satisfying $\{\{\lambda\}, \emptyset\} \subseteq K \subseteq RE$ and let K be closed under left or right-marking. Then a language L_0 is in the family $RBLB_{f, IO}(K)$ if and only if L_0 is recursively enumerable.*

Proof. Let L_0 be equal to $T(A)$, the set of strings in Σ^* accepted by the deterministic single-tape Turing machine A , where $A = (Q, \Sigma, \Gamma, B, \delta, q_0, F)$. Furthermore, we assume that $\delta(q, a) = \emptyset$ for each q in F . We construct an $(f, IO, REG, \emptyset NE)$ -belb grammar (G, C) with $G = (\Phi, \Psi, \Sigma, X, P, S)$ such that $L_{IO}(G, C) = L_0$. The grammar (G, C) generates nondeterministically a representation of a word z in Σ^* and simulates the computation of A on z . The Turing machine A reaches a final state with z as its input if and only if (G, C) generates z . Let Σ_λ denote $\Sigma \cup \{\lambda\}$.

We define the alphabets Φ and Ψ of G by

$$\Phi_0 = \{S, A_0\} \cup \{R_{qDa}, U_{qDa} \mid q \in Q, D \in \Gamma, a \in \Sigma_\lambda\},$$

$$\Phi_1 = \{[D, a] \mid D \in \Gamma, a \in \Sigma_\lambda\},$$

$$\Phi_2 = \{A_1\},$$

and

$$\Psi_0 = \{\psi_1\} \cup \{\xi_q, \eta_q \mid q \in Q\},$$

$$\Psi_1 = \{\psi_a \mid a \in \Sigma_\lambda\},$$

$$\Psi_2 = \{\psi_{A_1}\}.$$

The set of variables X is defined by $X = \{x, y\}$ and the set P of productions by

$$\begin{aligned} P = & \{\pi_0, \pi_1, \pi_B, \pi_{q_0}, \pi_{A_1}, \pi_\psi\} \\ & \cup \{U_{pDa} \rightarrow \eta_q \mid p, q \in Q, D \in \Gamma, a \in \Sigma_\lambda\} \\ & \cup \{R_{pDa} \rightarrow \xi_q \mid p, q \in Q, D \in \Gamma, a \in \Sigma_\lambda\} \\ & \cup \{U_{pDa} \rightarrow [E, a](\eta_q) \mid p, q \in Q, D, E \in \Gamma, a \in \Sigma_\lambda\} \\ & \cup \{R_{pDa} \rightarrow [E, a](\xi_q) \mid p, q \in Q, D, E \in \Gamma, a \in \Sigma_\lambda\} \\ & \cup \{[D, a](x) \rightarrow \psi_a(x) \mid D \in \Gamma, a \in \Sigma_\lambda\} \\ & \cup \{\psi_a(x) \rightarrow \{xa\} \mid a \in \Sigma_\lambda\} \\ & \cup P_\xi \cup P_\Sigma \cup P_\eta \cup \{\psi_1 \rightarrow \emptyset\}, \end{aligned}$$

where

$$\begin{aligned} \pi_0 &= S \rightarrow A_1(\xi_{q_0}, \psi_1), & \pi_1 &= A_0 \rightarrow \psi_1, \\ \pi_B &= A_0 \rightarrow [B, \lambda](\psi_1), & \pi_{q_0} &= A_0 \rightarrow \eta_{q_0}, \\ \pi_{A_1} &= A_1(x, y) \rightarrow \psi_{A_1}(x, y), & \pi_\psi &= \psi_{A_1}(x, y) \rightarrow \{xy\}. \end{aligned}$$

Furthermore, let

$$P_\xi = \{\xi_q \rightarrow \{\lambda\} \mid q \in F\},$$

$$P_\eta = \{\eta_q \rightarrow \{\lambda\} \mid q \in F\},$$

$$P_\Sigma = \{A_0 \rightarrow [a, a](\psi_1) \mid a \in \Sigma\}.$$

The control language C is defined by

$$C = \pi_0 \bar{\pi}_1 (\pi_B \bar{\pi}_1)^* (P_\Sigma \bar{\pi}_1)^* \pi_{q_0} (E_1 \cup M_0 \cup M_{-1} E_{-1})^* M_L^* P_\eta E_\Sigma^+ P_\xi \pi_{A_1} \pi_\psi.$$

The control language C can be considered to consist of three major parts; viz.

$$\begin{array}{ll} \text{initialization part} & \pi_0 \bar{\pi}_1 (\bar{\pi}_B \bar{\pi}_1)^* (P_{\Sigma} \bar{\pi}_1)^* \pi_{q_0}, \\ \text{simulation part} & (E_1 \cup M_0 \cup M_{-1} E_{-1})^*, \\ \text{termination part} & M_L^* P_{\lambda} E_{\Sigma}^+ P_{\xi} \pi_{A_1} \pi_{\psi}. \end{array}$$

Before the actual simulation of the Turing machine A starts, the initialization part generates a sentential form $\alpha_{n,k}$, which has a corresponding c-tree of the form shown in Figure 4, where $a_i \in \Sigma$ ($1 \leq i \leq n$). If A accepts the

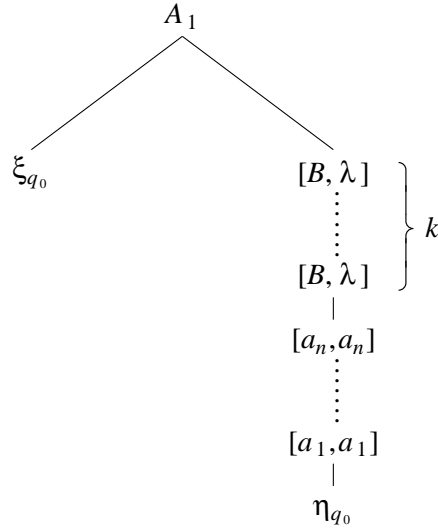


Figure 4.

string $a_1 \dots a_n$, then it will stop after some finite computation. The number k is a guess of the number of additional cells to the right of the n input cells, which the Turing machine A uses during this computation.

The simulation part $(E_1 \cup M_0 \cup M_{-1} E_{-1})^*$ simulates the computation of the Turing machine A . The sets E_1 , E_{-1} , M_0 and M_{-1} in the simulation part of C are defined as follows. Let $\Delta(r, q, D, a)$ be an abbreviation of

$$r \in \{-1, 0, 1\}, q \in Q, D \in \Gamma, a \in \Sigma_{\lambda}, \exists E \in \Gamma \cdot \exists p \in Q \cdot \delta(q, D) = (p, E, r),$$

and let $[(q, E, r)]_1 = q$ and $[(q, E, r)]_2 = E$. Then we define

$$\begin{aligned} E_1 = \{ & (\xi_{q_0} \rightarrow R_{qDa}) ([D, a] (\eta_{q_0}) \rightarrow U_{qDa}) (R_{qDa} \rightarrow [[\delta(q, D)]_2, a] (\xi_{[\delta(q, D)]_1})) \\ & (U_{qDa} \rightarrow \eta_{[\delta(q, D)]_1}) | \Delta(1, q, D, a) \}. \end{aligned}$$

The set E_1 simulates a 1-step of the Turing machine A . The first rule $\xi_{q_0} \rightarrow R_{qDa}$ of each control string in E_1 is such that D and a are guessed nondeterministically. By the second rule $([D, a] (\xi_{q_0}) \rightarrow U_{qDa})$ this guess is checked, and if the guess happens to be wrong the derivation is blocked.

$$M_s = \{ ([D, a] (\eta_{q_0}) \rightarrow U_{qDa}) (U_{qDa} \rightarrow [[\delta(q, D)]_2, a] (\eta_{[\delta(q, D)]_1})) | \Delta(s, q, D, a) \},$$

where s equals -1 or 0 .

The set M_0 simulates a 0-step of the Turing machine A .

$$E_{-1} = \{ (\eta_{q_0} \rightarrow U_{qDa}) ([D, a] (\xi_p) \rightarrow R_{qDa}) (U_{qDa} \rightarrow [D, a] (\eta_{q_0})) (R_{qDa} \rightarrow \xi_{q_0}) |$$

$$p, q \in Q, D \in \Gamma, a \in \Sigma_\lambda \}.$$

The sequence $M_{-1}E_{-1}$ simulates a (-1) -step of the Turing machine A .

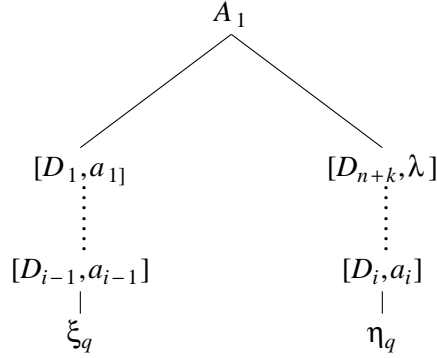


Figure 5.

We can show by induction on the number of Turing machine moves that if

$$q_0 a_1 \dots a_n \vdash_A^* D_1 \dots D_{i-1} q D_i \dots D_{n+k},$$

then for some control string c in the simulation part of C we have

$$\alpha_{n,k} \Rightarrow_{\text{IO}}^c \omega_{i,q}^{n+k},$$

where $\alpha_{n,k}$ is the sentential form generated by the initialization part of C , corresponding to $q_0 a_1 \dots a_n$ (cf. Figure 4) and $\omega_{i,q}^{n+k}$ is the sentential form associated with the c -tree represented in Figure 5, where $a_i = \lambda$ for all i with $n+1 \leq i \leq n+k$, and $D_i \in \Gamma$ for all i with $1 \leq i \leq n+k$.

If q is in F , then no rules in the simulation part of C are applicable.

In the termination part, sequences from M_L^* – with M_L defined by

$$M_L = \{ (\eta_q \rightarrow U_{qDa})([D, a](\xi_q) \rightarrow R_{qDa})(U_{qDa} \rightarrow [D, a](\eta_q))(R_{qDa} \rightarrow \xi_q) \mid \\ q \in F, D \in \Gamma, a \in \Sigma_\lambda \},$$

– transform the sentential form that has been derived after the actual simulation of the Turing machine A into one with a corresponding c -tree of the form shown in Figure 6, where $q \in F$, and $D_i \in \Gamma$ ($1 \leq i \leq n+k$).

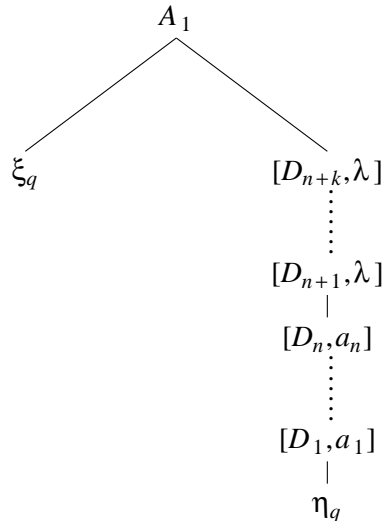


Figure 6.

Finally, $P_\eta E_\Sigma^\dagger P_\xi \pi_{A_1} \pi_\Psi$ derives the terminal string $a_1 \dots a_n$, where

$$E_\Sigma = \{([D, a](x) \rightarrow \Psi_a(x))(\Psi_a(x) \rightarrow \{xa\}) \mid a \in \Sigma_\lambda, D \in \Gamma\}.$$

By this construction, we have $T(A) \subseteq L_m(G, C)$. The converse inclusion can also be proved in a straightforward way. Thus, for each Turing machine A we have constructed an $(f, \text{IO}, \text{REG}, K)$ -belb grammar that generates $T(A)$. This proves the proposition from right to left. The converse implication can be proved using Church's Thesis. \square

The construction in the proof of Proposition 6.3 can serve as a base to prove a similar result with respect to the family $\text{RBLB}_{f, \text{OI}}(K)$.

Proposition 6.4. *Let K be a family satisfying $\{\{\lambda\}, \emptyset\} \subseteq K \subseteq \text{RE}$, and let K be closed under left or right-marking. Then a language L_0 is an $\text{RBLB}_{f, \text{OI}}(K)$ language if and only if L_0 is recursively enumerable.*

Proof. We give only the major steps of the construction. The construction of $G = (\Phi, \Psi, \Sigma, X, P, S)$ and C follows the proof of Proposition 6.3 directly. First, the initialization part of C has to generate a sentential form $\alpha_{n,k}$ with a c-tree structure as shown in Figure 7.

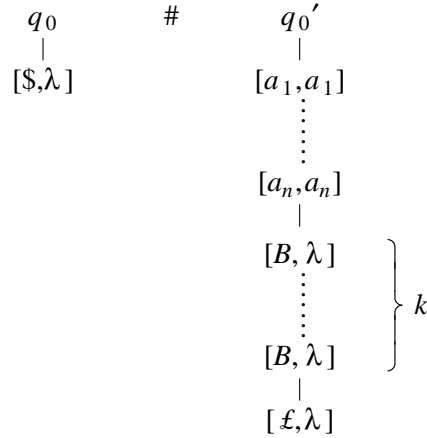


Figure 7.

This can be obtained easily. In general, if $q \in Q$, then q and q' are nonterminals in Φ_1 . Nonterminals of the form $[D, a]$, where $D \in \Gamma$ and $a \in \Sigma_\lambda$, are in Ψ_1 . Furthermore, $[\$, \lambda]$ and $[\pounds, \lambda]$ are in Ψ_0 , where $\$, \pounds$ are new symbols not in Γ .

The simulation part of C has the form $(E_1 \cup M_0 \cup M_{-1} E_{-1})^*$, which is identical to the simulation part of the control language in the proof of Proposition 6.3. However, the sets E_1 , E_{-1} , M_0 and M_{-1} are defined differently. Let R_{qDa} , U_{qDa} ($q \in Q$, $D \in \Gamma$, $a \in \Sigma_\lambda$) and $\Delta(r, q, D, a)$ be defined as in the proof of Proposition 6.3. In addition, we need language names Ψ_q , with $q \in Q$. Then

$$\begin{aligned}
 E_1 = & \{(q(x) \rightarrow q([D, a](x))) (q'([D, a](y)) \rightarrow U_{qDa}(y)) \\
 & (q([D, a](x) \rightarrow R_{qDa}(x)) (R_{qDa}(x) \rightarrow [\delta(q, D)]_1([\delta(q, D)]_2, a)(x))) \\
 & (U_{qDa}(y) \rightarrow \Psi_{[\delta(q, D)]_1}(y)) (\Psi_{[\delta(q, D)]_1}(y) \rightarrow [\delta(q, D)]_1'(y)) \mid \\
 & \Delta(1, q, D, a)\}.
 \end{aligned}$$

To obtain M_0 and $M_{-1} E_{-1}$ we define M_s for s equal to -1 or 0 by

$$M_s = \{(q'([D, a](y)) \rightarrow U_{qDa}(y))\}$$

$$(U_{qDa}(y) \rightarrow [\delta(q,D)]_1'([\delta(q,D)]_2,a](y)) | \Delta(s,q,D,a)\}.$$

Finally, E_{-1} is defined by

$$E_{-1} = \{(q'(y) \rightarrow q'([D,a](y))) (p([D,a](x)) \rightarrow R_{qDa}(x)) \\ (R_{qDa}(x) \rightarrow \Psi_q(x)) (\Psi_q(x) \rightarrow q(x)) | p,q \in Q, D \in \Gamma, a \in \Sigma_\lambda\}.$$

Then it easily follows that a sentential form generated during the simulation of the Turing machine A has a c-tree structure as shown in Figure 8. Note that $a_i = \lambda$ for all i with $n+1 \leq i \leq n+k$.

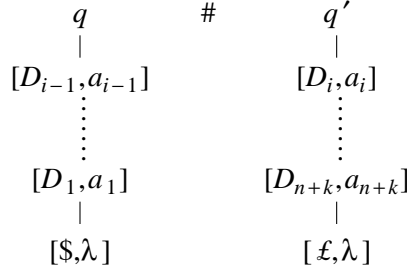


Figure 8.

The discussion of the termination part of C is left to the reader, as well as the remaining details of the proof. \square

7. Concluding Remarks

In this paper we have studied (m,K) -elb grammars provided with regular control and bidirectional rewriting. We have shown that if K is a nontrivial family closed under ngsm mappings, then the family $RBLB_{r,f,OI}(K)$ is a full substitution-closed AFL. Furthermore, if K is a family with $K \supseteq SYMBOL$, and closed under left or right-marking, intersection with regular languages, and homomorphism, then the family $RBLB_{r,f,IO}(K)$ is a full Quasi Abstract Family of Languages closed under deterministic substitution.

As for the generating power of these types of grammar we have seen that the family of IO-macro languages is included in each family $RBLB_{r,f,IO}(K)$, whenever $K \supseteq \emptyset NE$. And similarly, the family of OI-macro languages is included in each family $RBLB_{r,f,OI}(K)$, whenever $K \supseteq \emptyset NE$. Furthermore, we have that the family $RBLB_{r,f,OI}(K)$ equals the family OI whenever the family K satisfies $\emptyset NE \subseteq K \subseteq OI$. We also would like to establish upper bounds for the families $RBLB_{r,f,IO}(\emptyset NE)$. However, the proof techniques applied in the OI-case do not work for the IO-mode, since Lemma 5.10 does not hold for IO-macro grammars.

The results of Section 5 suggest that (m,K) -elb grammars provided with RCB-rewriting generate languages of a “nonlinear” character; i.e., languages generated by a type of grammar provided with symbols similar to nonterminals, such that each grammar, which generates such a language, derives – unidirectionally – at least one sentential form which contains at least two nonterminal-like items. It is likely that this is due to the interaction between the bidirectional rewriting and the presence of language names nested within nonterminals, which allow to obtain such nonlinear sentential forms. Therefore, it is interesting to study ordinary linear basic grammars provided with RCB-rewriting; cf. [16, 17].

Finally, with respect to Section 6 we remark that the use of control on the application of rules is indispensable to establish that – under weak assumptions on the family K – the family

$RBLB_{f,m}(K)$ equals the family of recursively enumerable languages for both $m = \text{OI}$ and $m = \text{IO}$.

Acknowledgment. I like to thank Peter Asveld for his comments on both text and contents of a preliminary version of this paper.

References

1. P.R.J. Asveld: *Iterated Context-Independent Rewriting — An algebraic approach to families of languages* (1978), Doctoral Dissertation, University of Twente, Enschede, The Netherlands.
2. P.R.J. Asveld & J. Engelfriet: Iterated deterministic substitution, *Acta Inform.* **8** (1977) 285-302.
3. P.R.J. Asveld & J. Engelfriet: Extended linear macro grammars, iteration grammars, and register programs, *Acta Inform.* **11** (1979) 259-285.
4. P.R.J. Asveld & J.A. Hogendorp: On the generating power of regularly controlled bidirectional grammars, Memorandum INF-89-68 (1989), Department of Computer Science, University of Twente, Enschede, The Netherlands.
5. L. Boasson: Dérivations et réductions dans les grammaires algébriques, *in*: J.W. de Bakker & J. van Leeuwen (Eds.): *7th International Colloquium on Automata Languages and Programming*, Lect. Notes Comp. Sci. **85** (1980) 109-118, Springer-Verlag, Berlin – Heidelberg – New York.
6. P.J. Downey: *Formal Languages and Recursion Schemes*, Ph.D. Thesis TR 16-74 (1974), Harvard University, Cambridge, Mass.
7. A. Ehrenfeucht & G. Rozenberg: On context-free languages not in EDTOL, *RAIRO Inform. théor./Theor. Inform.* **11** (1977) 273-291.
8. J. Engelfriet, E.M. Schmidt & J. van Leeuwen: Stack machines and classes of nonnested macro languages, *J. Assoc. Comput. Mach.* **27** (1980) 96-117.
9. J. Engelfriet, E.M. Schmidt: IO and OI (I), *J. Comput. System Sci.* **15** (1977) 328-353.
10. M.J. Fischer: *Grammars with Macro-like Productions*, Ph.D. Thesis (1968), Harvard University, Cambridge, Mass.
11. S. Ginsburg & E.H. Spanier: Control sets on grammars, *Math. Systems Theory* **2** (1968) 159-177.
12. S. Ginsburg: *Algebraic and Automata-Theoretic Properties of Formal Languages* (1975), North-Holland, Amsterdam.
13. M.A. Harrison: *Introduction to Formal Language Theory* (1978), Addison-Wesley, Reading, Mass.
14. J.A. Hogendorp: Nonterminal separating macro grammars, *in*: P.R.J. Asveld & A. Nijholt (Eds.): *Essays on Concepts, Formalisms, and Tools* (1987) 77-87, C.W.I. Tract no. 42, Centre for Mathematics and Computer Science, Amsterdam.
15. J.A. Hogendorp: Controlled bidirectional grammars, *Internat. J. Comput. Math.* **27** (1989) 159-180.
16. J.A. Hogendorp: Regularly controlled bidirectional linear basic grammars, Memorandum INF-90-40 (1990), Department of Computer Science, University of Twente, Enschede, The

Netherlands.

17. J.A. Hogendorp: *Controlled bidirectional grammars* (1990), Doctoral Dissertation, University of Twente, Enschede, The Netherlands.
18. J.E. Hopcroft & J.D. Ullman: *Formal Languages and Their Relation to Automata* (1969), Addison-Wesley, Reading, Mass.
19. H.R. Lewis & C.H. Papadimitriou: *Elements of the Theory of Computation* (1981), Prentice-Hall, Englewood Cliffs, NJ.
20. A. Salomaa: *Formal Languages* (1973), Academic Press, New York.