Controlled Rewriting Using Productions and Reductions

Jan Anne Hogendorp*

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

Abstract — We investigate context-free grammars the rules of which can be used in a productive and in a reductive fashion, while the application of these rules is controlled by a regular language. We distinguish several modes of derivation for this kind of grammar. The resulting language families (properly) extend the family of context-free languages. We establish some closure properties of these language families and some grammatical transformations which yield a few normal forms for this type of grammar. Finally, we consider some special cases (viz. the context-free grammar is linear or left-linear), and generalizations, in particular, the use of arbitrary rather than regular control languages.

1. Introduction

Usually one provides a grammatical model with a control mechanism to obtain additional generating power; cf. [10, 14, 15, 16, 17, 21] for controlled Chomsky-grammars, whereas in [1, 2, 3, 4] parallel grammars provided with control are studied. The use of control in defining programming languages may seem a bit awkward, but in fact in does occur in a non-trivial way in the definintion of modern programming languages, like e.g. Turbo Prolog™. Viz. a Turbo Prolog™ program is built up from different ‘‘program sections’’. Possible program sections are for instance *predicates* and *global predicates*. Concerning these two program sections we have the restriction that at most one global predicates section may be encountered during compilation and then only if there have been no ordinary predicates declarations earlier [20]. Let p denote a predicates declaration and q a global predicates declaration. With the production rules $\pi_1 = S \rightarrow qS$ and $\pi_2 = S \rightarrow pS$ this restriction can be represented by the control language $\pi_1\pi_2^* \cup \pi_2^*$.

In the literature, the rules of a controlled grammar are applied in a productive fashion only. In this paper, the control language is a regular language over both productions and reductions, as inspired by the concept of NTS-grammar. (NTS or nonterminal separating grammars form a subclass of the context-free grammars [18]; cf. also [5, 6, 7, 8, 19]. In this type of grammar, each sentential form which can be derived from a nonterminal by means of both productions and reductions can also be derived by the use of productions only. This property is also definable for some extensions of context-free grammars, for instance macro grammars; cf. [12].) The control language is associated with a so-called underlying context-free grammar. As reductions we consider rules of the form $\alpha \rightarrow A$ where $A \rightarrow \alpha$ is a production of the underlying grammar. The pair consisting of a context-free grammar and a regular control language is called a *regularly controlled bidirectional grammar* or *RCB-grammar*. Furthermore, we define the notions of *LRCB-* and of *LLRCB-grammar* which have as underlying grammar a linear context-free and a left-linear context-free grammar, respectively. Contrary to most grammatical models provided with a control device, the control is not just an additional feature. But it plays an essential part in many of our proofs in the sense that the corresponding proofs in the ‘‘uncontrolled’’ case become either unfeasible or probably impossible at all.

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

This paper is organized as follows. In Section 2 various modes of derivation are introduced. We do not allow free application of a production or a reduction but we only apply rules (i.e., productions and reductions) at the right side of the sentential form. Obviously, this choice is arbitrary and analogous results hold for the alternative case. We distinguish three kinds of derivation modes, each kind of which has two instances. First, given a sentential form, we consider two alternatives to select the nonterminal which ought to be rewritten. These are the RN-mode, which simply determines the right-most nonterminal, and the RO-mode, which determines the right-most occurrence of the nonterminal equal to the left-hand side of the current production. If a production cannot be applied we have another opportunity to choose between two possibilities. Either we skip the production (S-mode) or we simply break the derivational process, producing nothing at all. The applicability of a reduction under the two modes introduced above will be defined in terms of the applicability of the corresponding production. In the terminology of Salomaa [15, 16, 17] our B-mode corresponds to derivation without appearance checking and the S-mode corresponds to appearance checking with respect to the entire set of nonterminals. Because the RN-mode is comparable with the notion of derivation as defined in Ginsburg & Spanier [10], our treatment of the subject can be considered as a combination of those two approaches. Finally, we choose between either permitting or not terminal reductions. In fair mode (*f-mode*) we do not allow reductions of the type $\alpha \rightarrow A$ with α a string over the terminal alphabet, whereas in general mode (*g-mode*) we do.

In Section 3 we establish some closure properties of the language families defined by RCB-grammars. These closure properties consist of the regular ones (union, concatenation, and Kleene +) and also closure under homomorphism, inverse homomorphism, intersection with a regular set, and (regular or context-free) substitution. In Theorem 3.6 the most important results are summarized in AFL-terminology.

In Section 4 we introduce the notion of ‘‘weak Chomsky Normal Form’’. This is a variant of the Chomsky Normal Form in which productions of the form $A \rightarrow XY$ with X or $Y \in \Sigma$ are allowed. The main result of this section shows that every RCB/RN/B/f-language can be generated by an RCB/RN/B/f-grammar in this particular normal form.

The LRCB- and LLRCB-grammars are studied in Section 5. Besides some closure properties of the corresponding language families, we also establish a normal form for some of these grammars. It turns out that one can describe some of these languages by (L)LRCB-grammars having only a single nonterminal in its underlying grammar.

Section 6 is mainly devoted to the generalization to arbitrary control languages rather than regular ones. We mention which properties of the (regular and arbitrary) control languages are needed to prove the results of the previous sections.

2. Definitions and Examples

We define a regularly controlled bidirectional grammar as a pair (G,C) consisting of a context-free grammar G and a regular control language C over the productions and reductions (i.e., the reversed productions) of G. The control language C determines the order in which the productions and reductions of the grammar G ought to be applied.

For each context-free grammar $G = (V, \Sigma, P, S)$ with alphabet V, terminal alphabet Σ, set of productions P and initial symbol S, let $\overline{P}$ be the set of *reductions* corresponding to P, i.e., if an element π of P is equal to $A \rightarrow \alpha$, then $\overline{\pi}$ equals $\alpha \rightarrow A$ and $\overline{P} = \{\overline{\pi} \mid \pi \in P\}$.

Definition 2.1. A *regularly controlled bidirectional grammar* or *RCB-grammer* (G,C) consists of

- a context-free grammar $G = (V, \Sigma, P, S)$, called the *underlying* context-free grammar of (G,C), and
- a regular language C over $P \cup \overline{P}$. The language C is called the *control language* of (G,C).□

Before defining the language generated by an RCB-grammar (G,C), we first consider several *modes of derivation*, i.e., ways in which productions and reductions are applied to a sentential form of the underlying context-free grammar G, according to a word from the control language C. For each mode m, this results in a particular derivation relation $\Rightarrow_m$. Then using these derivation relations, we will associate to each mode m the language $L_m(G,C)$ generated by (G,C) under mode m. Roughly spoken, a terminal word w belongs to $L_m(G,C)$ if and only if it can obtained by means of applying a sequence of productions and reductions from $P \cup \overline{P}$ starting with S, according to some control word in the control language C. In the sequel a member of $P \cup \overline{P}$ will be called a *rule* of (G,C).

First we introduce two ways of selecting the nonterminal symbol from a string α in V^* to which a production π has to be applied, viz.

(1) *RN-mode*: determine the right-most nonterminal symbol of α,

(2) *RO-mode*: determine the right-most occurrence of the left-hand side of π in α.

The choice for determining the selected nonterminal symbol from the right end of α is arbitrary. Clearly, an analogous approach based on the nonterminal symbol selected from the left end is possible too and yields similar results. Let π be a production from P equal to $A \to \sigma$ and let m be either RN or RO. Now if the nonterminal selected by the mode m in a particular sentential form α is equal to the left-hand side A of π, then we say – as usual – that π is *applicable* to α, and we write $app_m(\pi,\alpha,\beta)$ in case β is the result of replacing that selected occurrence of A in α by the right-hand side σ of π. Next we call a reduction ρ, with $\rho = \overline{\pi}$ for some $\pi \in P$, applicable to a string α if there exists a string β with $app_m(\pi,\beta,\alpha)$, in case we also write $app_m(\rho,\alpha,\beta)$. It will be clear that there is at most one such a string β.

It may happen that in RN-mode the selected nonterminal is not equal to the left-hand side of a production π, and in both modes it may not even occur. With respect to reductions, in RO-mode it is possible that, when applied to a sentential form α, we cannot find a substring σ equal to the left-hand side of the reduction to the right of the right-most occurrence of the nonterminal symbol, if any is present. And in RN-mode, there may be no substring σ of α such that to the right of this σ only terminals occur. In these cases a production or a reduction is not applicable to a sentential form. Then we can follow two different strategies, giving us two additional mode instances independent of the nonterminal-selecting modes. In the block mode (*B-mode*) we do not allow to apply any rule to α once we have tried to apply a rule which was not applicable to α. In this mode the derivation relation $\Rightarrow^r_{m/B}$ – where r is a rule, i.e., either a production or a reduction – holds between strings α and β over V if $app_m(r, \alpha,\beta)$ holds. In the skip mode (*S-mode*) we still may apply rules to α after we have tried to apply a non-applicable rule with respect to α and m. In this mode the derivation relation $\Rightarrow^r_{m/S}$ holds between α and β, if either $app_m(r, \alpha,\beta)$ or $\neg\, app_m(r, \alpha,\beta) \wedge \alpha = \beta$ holds. Thus in B-mode applying a rule to a string over V may give no result, whereas in S-mode we will always end up with some string from V^*.

Next we define for $x \in (P \cup \overline{P})^*$ the relation $\Rightarrow^x_m$ which is the analogue of $\Rightarrow^*$ in uncontrolled grammars. In this notation m is a combination of different kinds of modes, separated by /'s, for instance RO/S or RN/B. This notational convention will also be applied to other mode

instances to be defined in the sequel. Now let $x = r_1 ... r_n$ ($n \geq 0$, $r_i \in P \cup \bar{P}$ for $1 \leq i \leq n$). Then $\alpha \Rightarrow_m^x \beta$ holds if there exists strings $\alpha_i \in V^*$ ($1 \leq i \leq n-1$), with

$$\alpha \Rightarrow_m^{r_1} \alpha_1 \Rightarrow_m^{r_2} \alpha_2 \Rightarrow_m^{r_3} \cdots \alpha_{n-1} \Rightarrow_m^{r_n} \beta .$$

With respect to applying a reduction ρ ($\rho \in \bar{P}$) we can distinguish another two mode instances – the *g-mode* and the *f-mode* – which are independent of the previously introduced modes of derivation. Viz. we can allow or disallow respectively, reductions of the form $\alpha \rightarrow A$ where $\alpha \in \Sigma^*$. If we allow such reductions of terminal strings we call this general reduction (*g-mode*); otherwise we call it fair reduction (*f-mode*). So in f-mode a terminal reduction causes blocking in B-mode and it is skipped in S-mode. From each regular control language C we can obtain an equivalent regular control language C' in which no terminal reductions occur, i.e., $C' \subseteq (P \cup (\bar{P} - \bar{P}_t))^*$ where P_t is the set of productions in P of which the right-hand side is a terminal string. This observation follows from the fact that the family of regular languages is closed under generalized sequential machine mappings.

An RCB-grammar in f-mode is in fact a special kind of a controlled phrase-structure grammar; cf. the proof of Proposition 2.4.(2). The distinction between f- and g-mode is also important when one considers chain rule deletion and when one studies LRCB- and LLRCB-grammars, i.e., RCB-grammars of which the underlying grammar is linear and left-linear, respectively; cf. Section 5.

Thus each RCB-grammar will be provided with three different types of modes, each of which may take one out of two values: RN versus RO, B versus S, and g versus f. In the sequel we will combine these mode values in an obvious fashion which results in notations like ''RN/B/f-mode'', and in concepts as ''RCB/RO/S/f-grammar''. If we do not specify a mode instance in a proposition or example, then we assume that it applies to both instances. For example, ''RN/f-mode'' means ''RN/B/f- and RN/S/f-mode''. Thus, in principle we now have 8 different types of grammars. However, not all these combinations of modes are equally important. Some interesting results will be established for certain mode combinations only; cf. Sections 3, 4 and 5. We will return to this matter in Section 6.

For each of the concrete modes of derivation, introduced above, we can now define the language generated by an RCB-grammar under that particular mode.

Definition 2.2. Let (G,C) be an RCB-grammar with underlying context-free grammar $G = (V, \Sigma, P, S)$ and control language $C \subseteq (P \cup \bar{P})^*$. For each mode m, the language $L_m(G,C)$ generated by (G,C) under mode m is $L_m(G,C) = \{w \in \Sigma^* \mid S \Rightarrow_m^x w, \text{ for some } x \in C\}$. □

In the following example the differences between the four possible combinations of mode instances of two modes are shown. We study the mode instances RO and RN together with the S- and B-mode, and we show that these modes are mutually independent.

Example 2.3. Consider the following RCB-grammar (G,C) with $G = (\{S,A,B,a,b\}, \{a,b\}, P, S)$ and P consists of $\pi_1 = S \rightarrow AB$, $\pi_2 = A \rightarrow a$, $\pi_3 = B \rightarrow A$, $\pi_4 = A \rightarrow AA$, $\pi_5 = A \rightarrow b$. As the control language we take $C = \{c_1, c_2\}$ with $c_1 = \pi_1\pi_2\pi_3\bar{\pi}_4\pi_5$ and $c_2 = \pi_1\pi_2\pi_3\pi_2$. With every combination of mode instances mentioned above, together with the g-mode, we obtain a different language.

$L_{RN/B/g}(G,C) = \emptyset$. This equality holds because in both control words the application of π_2 causes blocking.

$L_{RN/S/g}(G,C) = \{b\}$. Now π_2 is skipped, so we have the derivations $S \Rightarrow_{RN/S/g}^{c_1} b$ and $S \Rightarrow_{RN/S/g}^{c_2} Aa$.

$L_{RO/B/g}(G,C) = \{aa\}$. In this setting, π_2 is applicable. Now in c_1 $\bar{\pi}_4$ causes blocking, and c_2 gives $S \Rightarrow^{c_2}_{RO/B/g} aa$.

$L_{RO/S/g}(G,C) = \{aa, ab\}$. Now $\bar{\pi}_4$ is skipped in c_1, and so $S \Rightarrow^{c_1}_{RO/S/g} ab$. □

The generating power of RCB-grammars turns out to be rather strong. For instance, the family of context-free languages is included in the family of RCB/*m*-languages, independently of the mode *m*.

Proposition 2.4. (1) *The family of context-free languages is included in the family of regularly controlled bidirectional languages for each mode of derivation.*
(2) *The family of RCB/RN/B/f-languages coincides with the family of context-free languages.*

Proof: (1) Let $G = (V, \Sigma, P, S)$ be a context-free grammar. Then $L(G) = L_m(G,C)$ for each mode *m*, where (G,C) is the RCB-grammar with $C = P^*$.

(2) Because of (1) we only ought to prove the inclusion from left to right. In [10] the family of languages $L_C(G)$ generated by phrase-structure grammars G and control languages C has been investigated. In our notation the mode of derivation used in [10] reads LN/B where LN abbreviates left-most nonterminal (cf. RN-mode), or even, LN/B/f since in [10] no reductions are considered. For each RCB/RN/B/f-grammar (G,C) with $G = (V, \Sigma, P, S)$ we now consider the phrase-structure grammar $G' = (V, \Sigma, P', S)$ with $P' = P \cup \{\alpha \to \beta \mid \beta \to \alpha \in P,\ \alpha \in V^*(V - \Sigma)V^*\}$ and we modify C accordingly into C'. Then $L(G,C) = L_{C'}(G')$ provided in the latter case we take the RN/B/f-mode instead of the LN/B/f-mode. By a ''right-most nonterminal'' variant of Corollary 1 to Theorem 2.1 from [10] we obtain that $L_{C'}(G')$, and hence $L(G,C)$, is context-free. □

For some concrete modes, one can easily show that the generating power of RCB-grammars increases as compared with the underlying grammar. This fact is illustrated by the following examples.

Example 2.5. Consider the RCB/g-grammar (G,C) with $G = (V, \Sigma, P, S)$, $V = \{S\} \cup \Sigma$, $\Sigma = \{a,b,c\}$, and $P = \{\pi_1, \pi_2, \pi_3, \pi_4, \pi_5, \pi_6, \pi_7\}$, the set of productions, defined as $\pi_1 = S \to abc$, $\pi_2 = S \to a$, $\pi_3 = S \to aa$, $\pi_4 = S \to b$, $\pi_5 = S \to bb$, $\pi_6 = S \to c$, and $\pi_7 = S \to cc$. As the control language we take $C = \pi_1(\bar{\pi}_2\pi_3\bar{\pi}_4\pi_5\bar{\pi}_6\pi_7)^*$. Then $L_g(G,C) = \{a^n b^n c^n \mid n \geq 1\}$, as easily can be checked. □

Example 2.6. [17] The language in Example 2.5 is also generated by the following RCB/RO/f-grammar (G_0, C_0) with $G_0 = (V, \Sigma, P, S)$ where $V = \{S,A,B,C\} \cup \Sigma$, $\Sigma = \{a,b,c\}$ and P consists of the productions $\pi_1 = S \to ABC$, $\pi_2 = A \to Aa$, $\pi_3 = B \to Bb$, $\pi_4 = C \to Cc$, $\pi_5 = A \to a$, $\pi_6 = B \to b$, $\pi_7 = C \to c$. The control language C_0 of (G_0, C_0) equals $\pi_1(\pi_2\pi_3\pi_4)^*\pi_5\pi_6\pi_7$. Note that no reductions occur in any derivation of (G_0, C_0). □

Example 2.7. The language $\{a^n b^n c^n \mid n \geq 1\}$ can also be generated by a RCB/RN/S/f-grammar (G_1, C_1). Take $G_1 = (V, \Sigma, P, S)$ as follows. $\Sigma = \{a,b,c\}$, $V = \Sigma \cup \{A,B,C,D,E,F,G,S\}$. As the set of productions P we take $\{\pi_i \mid 0 \leq i \leq 13\}$ with

$$\begin{array}{lll}
\pi_0 = S \to abDc, & \pi_5 = C \to bDc, & \pi_{10} = F \to cDb, \\
\pi_1 = S \to aSbDc, & \pi_6 = C \to bc, & \pi_{11} = G \to bFb, \\
\pi_2 = A \to cbD, & \pi_7 = E \to DAb, & \pi_{12} = G \to bbcDb, \\
\pi_3 = B \to bDb, & \pi_8 = E \to DbcDb, & \pi_{13} = F \to bc. \\
\pi_4 = B \to bb, & \pi_9 = A \to bc, &
\end{array}$$

With the control language $C_1 = \pi_1^*\pi_0(\bar{\pi}_2\bar{\pi}_3\pi_4\bar{\pi}_5\pi_6\bar{\pi}_7\pi_8\pi_9\bar{\pi}_{10}\bar{\pi}_{11}\pi_{12}\pi_{13})^+$ we get the desired language. To understand this example it may be helpful to make the following observations.

First, the sequence $\bar{\pi}_3\pi_4\bar{\pi}_5\pi_6$ rewrites the nonterminal D into λ in the context $b—b$ or $b—c$. Then the sequence $\bar{\pi}_2\bar{\pi}_3\pi_4\bar{\pi}_5\pi_6\bar{\pi}_7\pi_8\pi_9$ rewrites *DcbDb* into *DbcDb* and *DcbDc* into *Dbcc*. Finally, we observe that the sequence $\bar{\pi}_{10}\bar{\pi}_{11}\pi_{12}\pi_{13}$ rewrites *bcDbb* into *bbcDb* and *cDbc* into *bcc*. The latter observation can also be formulated as: *cDb* becomes *bcD* in the context $b—b$ and *bc* in the context $—c$. □

3. Closure Properties

In this section we establish some closure properties of the family of languages generated by regularly controlled bidirectional grammars. In the sequel of this section we assume that L_i $(i \geq 1)$ is a language generated by an RCB-grammar (G_i, C_i) with $G_i = (V_i, \Sigma_i, P_i, S_i)$. In addition we assume that $N_i \cap N_j = \emptyset$ if $i \neq j$, where $N_i = V_i - \Sigma_i$ for every $i \geq 1$.

If not stated otherwise the results in this section hold for every combination of modes introduced in the previous section. By Proposition 2.4.(2) the family of RCB/RN/B/f-languages inherits all closure properties of the context-free languages. Therefore we mainly focus our attention in this section to modes different from RN/B/f.

Proposition 3.1.

- *The family of RCB-languages is closed under union.*
- *The family of RCB/B/f-languages and the family of RCB/RN/S/f-languages are closed under marked concatenation, marked Kleene +, and marked Kleene ∗.*
- *The family of RCB/RO/S/f-languages is closed under marked concatenation.*
- *The family of RCB/RO/f-languages is closed under concatenation.*
- *The family of RCB/RO/B/f-languages is closed under Kleene +, and Kleene ∗.*

Proof: Cf. [13]. □

Proposition 3.2. *The family of RCB/RO-languages is closed under intersection with regular languages.*

Proof: Let $L_1 = L(G_1, C_1)$ and R be a regular language, and let $(Q, \Sigma_R, \delta, q_0, F)$ be a deterministic finite automaton which accepts the reversal of R. We construct an RCB/RO-grammar (G, C) with $G = (V, \Sigma, P, S)$ such that $L_1 \cap R = L(G, C)$. The set of nonterminals N will be defined as follows. N contains two new symbols S and Z $(S, Z \notin V_1)$ and all triples of the form (u, A, t) where $u, t \in Q$ and $A \in V_1 \cup \{\lambda\}$. To complete N we add a symbol A_a for every $a \in \Sigma_1 \cup \{\lambda\}$. The set Σ of terminals of G equals $\Sigma_1 \cap \Sigma_R$. In order to define P we use the following notational conventions. For each $x \in V_1^*$, with $x = x_1 \ldots x_m$ $(m \geq 0)$, $x_i \in V_1$ $(1 \leq i \leq m)$, we define

$$\tilde{x} = \{(p_0, x_1, p_1) \ldots (p_{m-1}, x_m, p_m) \mid p_i \in Q,\ 0 \leq i \leq m\},$$

$$\tilde{\lambda} = \{(p_0, \lambda, p_1) \mid p_0, p_1 \in Q\},$$

and for every p, q in Q

$$\tilde{x}_p^q = \{(p, x_1, p_1) \ldots (p_{m-1}, x_m, q) \mid p_i \in Q,\ 1 \leq i \leq m-1\},$$

$$\tilde{\lambda}_p^q = \{(p, \lambda, q)\}.$$

We denote an element from $\tilde{x}$ by $\tilde{x}(p_0, \ldots, p_m)$. Consider for every $\pi = A \to \alpha$ in P_1,

$$P_\pi = \{(p, A, q) \to t \mid p, q \in Q,\ t \in \tilde{\alpha}_p^q\}$$

and for every $a \in \Sigma_1 \cup \{\lambda\}$,

$$P_a = \{(p,a,q) \to a \,|\, p,q \in Q,\ \delta(q,a) = p\}.$$

Because $P_a = \varnothing$ whenever $a \in \Sigma_1 - \Sigma$, we define $P_\Sigma = \cup\{P_a \,|\, a \in \Sigma \cup \{\lambda\}\}$. Now we define the set P of productions of G by

$$P = P_0 \cup P_F \cup P_E \cup P_\Sigma \cup \cup\{P_\pi \,|\, \pi \in P_1\}$$

where

$$P_0 = \{S \to Z(u,S_1,q_0) \,|\, u \in Q\},$$
$$P_F = \{A_a \to Z(u,a,t) \,|\, u = \delta(t,a),\ u \in F,\ a \in \Sigma_1 \cup \{\lambda\}\},$$
$$P_E = \{A_a \to a \,|\, a \in \Sigma \cup \{\lambda\}\}.$$

Consider the finite substitution $\sigma : P_1 \cup \bar{P}_1 \to 2^{(P \cup \bar{P})^*}$ defined by $\sigma(\pi) = P_\pi$ and $\sigma(\bar{\pi}) = \bar{P}_\pi$ for each $\pi \in P_1$. Finally, we define the control language C by $C = P_0 \sigma(C_1) \bar{P}_F P_E P_\Sigma^*$.

The fact that (G,C) exactly generates $L_1 \cap R$ is shown as follows. Let $T = \bar{P}_F P_E P_\Sigma^*$ and let $w \in L(G,C)$. Then there exist $\pi_0 \in P_0$, $d \in \sigma(C_1)$ and $t \in T$ such that $S \Rightarrow_{RO}^{\pi_0 dt} w$. Applying π_0 yields: there are $p \in Q$, $d \in \sigma(C_1)$ and $t \in T$, such that $Z(p,S_1,q_0) \Rightarrow_{RO}^{dt} w$. From the definition of $\sigma(C_1)$, this d yields $p,p_1,...,p_{m-1}$ in Q such that there exist $t \in T$ and $v \in L_1$ with $Z\tilde{v}(p,p_1,\ldots,p_{m-1},q_0) \Rightarrow_{RO}^{t} w$. Following the definitions of $\bar{P}_F$, P_E and P_Σ^*, we see that this implies that $p \in F$, $v = w$ and $w \in L_1 \cap R$. The second part of the proof is obtained by traversing this argument in the opposite direction. □

Proposition 3.3.

(a) *The family of RCB/RO/B/f-languages is closed under substitution*.

(b) *The family of RCB/RO-languages is closed under context-free substitution*.

Proof: (a) Let $L_1 = L(G_1,C_1)$ be an RCB/RO/B/f-language and let σ be an RCB/RO/B/f-substitution $\sigma : \Sigma_1 \to 2^{\Sigma^*}$. Assume that $\Sigma_1 = \{a_1,...,a_n\}$. Then for each $a \in \Sigma_1$, there exists an RCB/RO/B/f-grammar (G_a,C_a) with $G_a = (V_a,\Sigma,P_a,S_a)$ such that $L(G_a,C_a) = \sigma(a)$. We assume that for every $a \in \Sigma_1$, $N_1 \cap V_a = \varnothing$ and that $N_{a_i} \cap N_{a_j} = \varnothing$ if $i \neq j$ for every $1 \leq i, j \leq n$. Define alphabets $\Delta = \{S_{a_1},\ldots,S_{a_n}\}$ and $\Omega = \{Z_{a_1},...,Z_{a_n}\}$. Furthermore, consider an isomorphism $i : V_1 \to N_1 \cup \Omega$ defined by

$$i(A) = A \qquad \text{for each } A \text{ in } N_1,$$
$$i(a) = Z_a \qquad \text{for each } a \text{ in } \Sigma_1.$$

Let $U = \{A \to \alpha \,|\, A \in N_1, \alpha \in (N_1 \cup \Omega)^*\}$. Then we introduce a control set $T = \cup\{C_a \,|\, a \in \Sigma_1\}$ and a homomorphism $h : P_1 \cup \bar{P}_1 \to U \cup \bar{U}$ defined as follows

$$h(A \to \alpha) = A \to i(\alpha),$$
$$h(\alpha \to A) = i(\alpha) \to A.$$

Now we can define the RCB/RO/B/f-grammar (G,C) which generates the language $\sigma(L_1)$ by $G = (V, \Sigma, P, S)$ where

– $V = \cup\{V_a \,|\, a \in \Sigma_1\} \cup N_1 \cup \Delta \cup \Omega \cup \{Z\}$

– $P = \cup\{P_a \,|\, a \in \Sigma_1\} \cup h(P_1) \cup P_Z \cup \{Z \to \lambda\}$ with

$P_Z = \{Z_a \to Z\, S_a \,|\, a \in \Sigma_1\}$,

– $S = S_1$

and $C = h(C_1) P_Z^* T^* \{Z \to \lambda\}^*$

This construction works as follows. (A more detailed proof can be found in [13].) For each control word $c \in C_1$, with $S_1 \Rightarrow^c w_1 ... w_m \in \Sigma_1^*$, we obtain a string $Z_{w_1} ... Z_{w_m}$ after applying $h(c)$ to S_1. Next, every Z_{w_i} is rewritten to $Z\, S_{w_i}$ by applying P_Z^*. The nonterminals S_a are rewritten by the control sets C_a, $a \in \Sigma_1$. The nonterminal Z has as its purpose to prevent interaction between the rewriting of the nonterminals S_a. This is necessary, because applying a control word $c_1 \in C_a$ to S_a may yield a terminal string w, which, by means of reductions, may influence the derivation of a neighbor nonterminal S_a. Finally, all the Z's are rewritten to λ by applying $\{Z \to \lambda\}^*$.

The construction for 3.3(b) is easier, since in the derivations according to (G_a, C_a) only productions are used, and the control languages C_a are equal to P_a^* for each a in Σ_1. Therefore we do not need a nonterminal Z to prevent interaction. With the assumption that the nonterminal alphabets of the grammars G_a are mutually disjoint it is straightforward to prove that $L(G,C) = \sigma(L(G_1, C_1))$. □

Corollary 3.4. *The family of RCB/RO-languages is closed under homomorphism.* □

Proposition 3.5. *The family of RCB/RO-languages is closed under inverse homomorphism.*

Proof: Straightforward, cf. [9] and Proposition 2.4.(1). □

A family of languages is called *nontrivial* if it contains a language which differs from $\varnothing$ and from $\{\lambda\}$. Recall that a *full semi-Abstract Family of Languages* or *full semi-AFL* (cf. [9] for this and the following related concept) is a nontrivial family of languages which is closed under union, homomorphism, inverse homomorphisms and intersection with regular languages. Furthermore, a *full Abstract Family of Languages* or *full AFL* is a full semi-AFL which is also closed under concatenation, and Kleene +.

These concepts allow us to summarize some closure properties in the following form.

Theorem 3.6.

- *The family of RCB/RO/B/f-languages is a full AFL closed under substitution*.
- *The family of RCB/RO/S/f-languages is a full semi-AFL closed under concatenation.*
- *The family of RCB/RO/g-languages is a full semi-AFL.*

Proof: These results follow immediately from Propositions 3.1, 3.2, 3.3, 3.5 and Corollary 3.4. □

4. Grammatical Transformations

In this section we study certain transformations on RCB-grammars with the purpose to obtain normal forms for RCB-grammars. First we introduce the notion of ''weak Chomsky Normal Form''.

Definition 4.1. A context-free grammar $G = (V, \Sigma, P, S)$ is in *weak Chomsky Normal Form* or in *weak CNF* if each production of P has one of the following forms: $A \to XY$ or $A \to a$ with $A \in N$ $(N = V - \Sigma)$, whereas $X, Y \in V$ and $a \in \Sigma \cup \{\lambda\}$. An RCB-grammar (G,C) is in *weak CNF* if its underlying grammar G is in weak CNF. □

We allow X or Y to be an element of Σ, contrary to the usual Chomsky Normal Form where X and Y ought to be members of N only.

To transform an RCB-grammar into a weak CNF RCB-grammar it is not sufficient to transform the underlying grammar only, but we also ought to modify the corresponding control language. To obtain a weak Chomsky Normal Form for an RCB-grammar (G_0, C_0), we first

transform it into an equivalent RCB-grammar (G_1, C_1) in which G_1 has no chain rules. It turns out that this transformation works properly for one combination of modes only.

Definition 4.2. Let N be a set of nonterminal symbols. A *chain rule* is a rule $A \to B$ with $A, B \in N$, and $CH(N)$ is the set of all chain rules which can be formed with elements from N. □

Lemma 4.3. *Let (G_0, C_0) be an RCB/RN/B/f-grammar. Then there exists an equivalent RCB/RN/B/f-grammar (G_1, C_1) such that G_1 possesses no chain rules.*

Proof: The idea of the proof is based on similar arguments in [4, 2] for parallel rewriting systems. Viz. we construct a nondeterministic generalized sequential machine (or ngsm) $T = (Q, P_I, P_O, \delta, q_0, Q_F)$ such that $C_1 = T(C_0)$ and $G_1 = (V_0, \Sigma_0, P_1, S_0)$, with $P_1 = \{A \to \omega \mid A \in N_0, A \to \omega \in P_O\}$, and P_1 has no chain rules. Because the family of regular languages is closed under ngsm-mappings, C_1 is a regular language too.

Each state of T is an ordered pair (X, Y) where X is equal to the right-most nonterminal which appeared in the sentential form by the last non-chain rule in the derivation from S, or it is equal to S itself. Y equals the nonterminal to which X is rewritten by means of a nonempty consecutive sequence of chain rules. $Y = \lambda$ denotes the case that X is not rewritten by chain rules or that it is rewritten by such rules to X itself. The nondeterministic character of T appears when a nonterminal is rewritten to a terminal string. In that case another nonterminal becomes the right-most nonterminal which T ought to guess nondeterministically. T also ought to guess whether or not a reduction which is not a chain rule can be applied.

Before giving the formal description of T we introduce the following notation. Let (G, C) be an RCB-grammar, r be a rule of (G, C) and let $X \in N$. $R(\alpha)$ denotes the right-most nonterminal of α if $\alpha \in V^* - \Sigma^*$ and $R(\alpha) = \lambda$ if $\alpha \in \Sigma^*$. Let $lhs(r)$ and $rhs(r)$ denote the left-hand side and the right-hand side of r respectively. In the sequel we write r_X to denote the rule $([X/R(lhs(r))]\, lhs(r)) \to rhs(r)$, where $[X/R(\alpha)]\,\alpha$ denotes the string obtained from α by substituting X for the right-most nonterminal of α. Furthermore, we define the set $RN(r)$ as $\{R(rhs(r))\}$ if $rhs(r) \in V^* - \Sigma^*$ and $RN(r) = N \cup \{\lambda\}$ if $rhs(r) \in \Sigma^*$. Finally, we will use a function $act : Q \to N$ defined by

$$act((X,Y)) = X \qquad \text{if } Y = \lambda \text{ and}$$
$$act((X,Y)) = Y \qquad \text{otherwise.}$$

Now $act((X,Y)) = R(lhs(r))$ is a necessary condition for r to be applicable, and in most cases also sufficient, except when $r \in \bar{P}_0 - CH(N_0)$.

Formally, the ngsm T is defined as follows:

- the set of states is $Q = \{(X,Y) \mid X, Y \in N_0 \cup \{\lambda\}\}$,
- the input alphabet is $P_I = P_0 \cup \bar{P}_0$,
- the output alphabet equals

$$P_O = P_0 \cup \bar{P}_0 \cup \{r_X \mid r \in P_0 \cup \bar{P}_0,\ X \in N_0\} - CH(N_0),$$

- the initial state is $q_0 = (S_0, \lambda)$,
- the set of final states is $Q_F = \{(\lambda, \lambda)\}$,
- the transition mapping $\delta : Q \times P_I \to 2^{Q \times P_O^*}$ is defined by

$$\delta((X,Y), r) =$$
$$\{\, ((Z, \lambda), r) \mid Y = \lambda,\quad Z \in RN(r),\quad r \notin CH(N_0),\quad R(lhs(r)) = X \,\} \cup$$

$$\cup\{\ ((Z,\ \lambda),\ r_X)\,|\,Y \neq \lambda,\quad Z \in RN(r),\quad r \notin CH(N_0),\quad R(lhs(r)) = Y\ \}\cup$$

$$\cup\{\ ((X,\ \lambda),\ \lambda)\,|\,X = rhs(r),\quad r \in CH(N_0),\quad lhs(r) = act((X,Y))\ \}\cup$$

$$\cup\{\ ((X,\ rhs(r)),\ \lambda)\,|\,X \neq rhs(r),\quad r \in CH(N_0),\quad lhs(r) = act((X,Y))\ \}.$$

Note that $Y \neq \lambda$ implies $X \neq \lambda$, and consequently r_X is defined.

The correct behavior of T is easily checked. That T behaves correctly when it has to guess has been shown in [13]. □

By means of Lemma 4.3 we are able to prove the following normal form theorem.

Theorem 4.4. *For every RCB/RN/B/f-grammar* (G_1, C_1) *there exists an equivalent RCB/RN/B/f-grammar* (G, C) *in weak CNF.*

Proof: By Lemma 4.3 we assume that G_1 has no chain rules. Let $P_1 = \{\pi_1, ..., \pi_n\}$ be the set of productions of G_1 with $\pi_i = A_i \to B_{i,1}...B_{i,m_i}$. Let P be constructed as follows. Starting with the empty set, adjoin every production of P_1 to P which has a right-hand side with a length smaller than three. As the next step, for every $\pi_i \in P_1$ with $m_i \geq 3$ we construct $m_i - 1$ new productions from this production as follows. Take $\pi_{i,1} = A_i \to B_{i,1}D_{i,1}$, $\pi_{i,2} = D_{i,1} \to B_{i,2}D_{i,2}$,..., $\pi_{i,m_i-1} = D_{i,m_i-2} \to B_{i,m_i-1}B_{i,m_i}$. We assume that the $D_{i,j}$'s are distinct from each other, and that these $D_{i,j}$'s constitute the set D. The productions $\pi_{i,j}$ will be adjoined to P. Now we define a homomorphism $h : P_1 \to P^*$ with $h(\pi_i) = \pi_i$ if $m_i \leq 2$ and $h(\pi_i) = \pi_{i,1}, ..., \pi_{i,m_i}$ if $m_i \geq 3$. Furthermore, for a reduction $\bar{\pi} \in \bar{P}_1$ define $h(\bar{\pi}) = \overline{h(\pi)}$, using $\bar{\pi}\bar{\tau} = \bar{\tau}\bar{\pi}$ for every $\pi, \tau \in P_1$. Finally, we take $C = h(C_1)$ and $G = (V_1 \cup D, \Sigma_1, P, S_1)$.

Verifying the correctness of this construction is left to the reader as an easy exercise. □

It is unlikely that the arguments used in establishing Lemma 4.3 and Theorem 4.4 can be modified to obtain an RCB/RN/B/f-grammar in the usual Chomsky Normal Form, because of productions of the form $A \to \alpha\beta$ with $\alpha \in \Sigma_1^+$ and $\beta \in V_1^* - \Sigma_1^*$. For then we ought to remember to insert productions $F_a \to a$, $a \in \Sigma_1$ in the new control word after inserting productions which will derive β. Because this may get nested up to any level, an ngsm-mapping is not able to handle this. □

5. Linear and Left-Linear RCB-grammars

This section is devoted to the study of RCB-grammars of which the underlying grammar is linear or left-linear. The major part of the results in this section consists of straightforward consequences of propositions established in Sections 3 and 4.

Definition 5.1. If the underlying context-free grammar G of an RCB-grammar (G, C) happens to be linear, then we call (G, C) a *linear RCB-grammar* or *LRCB-grammar*. And (G, C) is a *left-linear RCB-grammar* or an *LLRCB-grammar* if G is a left-linear grammar. □

All the modes of derivation introduced in Section 2 are applicable to LRCB- and to LLRCB-grammars as well. However, the grammar types LRCB/RN/B/f and LRCB/RO/B/f, as well as LRCB/RN/S/f and LRCB/RO/S/f are strongly equivalent. This equivalence is due to the fact that fair reduction maps linear sentential forms into linear sentential forms, in which case the difference between RN-mode and RO-mode vanishes. The same remark applies to LLRCB-grammars.

Proposition 5.2. *The family of* [*left-*] *linear context-free languages is included in the family of* [*left-*] *linear regularly controlled bidirectional languages for each mode of derivation.* □

Clearly, the first construction in the proof of Proposition 3.1 also applies to LRCB-grammars, cf. [13]. Therefore we have

Corollary 5.3. (1) *The family of LRCB-languages is closed under (marked) union.*
(2) *The families of LRCB- and LLRCB-languages are closed under union with a regular set.*

Proof: (2) It is easy to see that the regular languages form a subset of the LLRCB-languages. □

Many of the constructions used in Section 3 fail to work in the LRCB- and in the LLRCB-case. Therefore we have less results for these language families. However, the families of LRCB/f-languages and of LLRCB/f-languages turn out to be closed under reversal.

Proposition 5.4. *The families of LRCB/f-languages and of LLRCB/f-languages are closed under reversal.*

Proof: Straightforward. □

Concerning the LLRCB-languages we have the following results.

Proposition 5.5.

- *The family of LLRCB-languages is closed under (marked) union.*
- *The family of LLRCB/f-languages is closed under marked concatenation, marked Kleene + and marked Kleene ∗.*

Proof: Cf. [13]. □

For LRCB/f- and LLRCB/f-grammars we can establish a very simple normal form.

Proposition 5.6. *Let* (G_0, C_0) *be an LRCB/f- or an LLRCB/f-grammar. Then there exists an equivalent LRCB/f- or an LLRCB/f-grammar* (G, C)*, respectively, which only possesses one nonterminal symbol.*

Proof: Let (G_0, C_0) be an LRCB/f- or an LLRCB/f-grammar. For this type of grammar we can easily construct, using a gsm, a grammar (G_0, C_1) where C_1 is such that for every two consecutive rules r_1 and r_2 in a control word $c \in C_1$, we have $R(rhs(r_1)) = R(lhs(r_2))$. (Cf. Lemma 4.3 for the definition of R. In this case R yields the nonterminal of a string $\alpha \in \Sigma^*(V-\Sigma)\Sigma^*$, and $R(\alpha) = \lambda$ if $\alpha \in \Sigma^*$.) If we replace each nonterminal in every rule occurring in G_0 and C_1 by the start symbol S_0 we obtain the grammar we aimed for. This latter step is now possible because the remaining nonterminals in (G_0, C_1) have as their single task to indicate at which position in a sentential form a rule ought to apply. This can be performed by one unique nonterminal as well.
□

6. Concluding Remarks

In this paper we extended regularly controlled context-free grammars to regularly controlled grammars with context-free rules which may be applied in a productive as well as a reductive fashion. In this approach we distinguished several (combinations of) modes of derivation. Some of these combinations have originally been introduced in the literature, i.e., the RN-mode in [10] (actually the LN-mode, cf. Proposition 2.4.(2).) and the B- and S-mode in [15, 16, 17] using somewhat different names. The introduction of the RO-mode has been inspired by the proof to establish closure under intersection with a regular set; cf. the proof of Proposition 3.2. A similar observation can be made for the f-mode with respect to closure under substitution; cf. [13]. However, the latter mode has also a justification in itself, for in g-mode some terminals play the part of ‘‘pseudo-nonterminals’’, i.e., they are in the terminal alphabet of the grammar but they can act as a nonterminal, for example a reduction $a \to A$, which is not a phrase-structure rule; cf.

Example 2.5. This phenomenon obscures the distinction between terminal and nonterminal symbols in grammatical models.

Closure properties of RCB-languages.								
	RN				RO			
	B		S		B		S	
	f	g	f	g	f	g	f	g
union	+	+	+	+	+	+	+	+
marked union	+	+	+	+	+	+	+	+
concatenation	+				+		+	
marked concatenation	+		+		+		+	
Kleene +	+				+			
marked Kleene +	+		+		+			
Kleene $*$	+				+			
marked Kleene $*$	+		+		+			
homomorphism	+				+	+	+	+
intersection with a regular set	+				+	+	+	+
context-free substitution	+				+	+	+	+
union with a regular set	+	+	+	+	+	+	+	+
inverse homomorphism	+				+	+	+	+
substitution	+				+			
substitution into a regular set	+				+			

Table 1.

The closure properties established in Section 3 are summarized in Table 1. We can make the following observations from Table 1. First, we ought to remark that a table entry which is empty does not mean a negative result, but a problem not yet solved. Concerning the positive results, we see that the combination of the modes B and f enables us to prove all the closure properties listed in the table. Intuitively, this is because in combination with the RO-mode other mode instances can cause ‘‘side effects’’ such as in case of the mode instances S or g. In addition we have the result of Theorem 4.4, which gives us a useful normal form for RCB/RN/B/f-grammars. These facts make the B/f-mode the most promising combination of modes, especially the RN/B/f-mode.

In establishing the closure properties of RCB-languages we used some (closure) properties of the family of regular languages (‘‘over the alphabet of productions and reductions’’). If we generalize from the family of regular languages we ought to know which of these properties are needed to obtain these closure properties of RCB-languages. Let **C** denote an arbitrary family of control languages. Then, for instance, closure under (marked) union is provable if **C** is closed under marked union, as one can see from the proof of Proposition 3.1, cf. [13]. In Table 2 results are shown based on the analysis of the proof of each closure property. Because **C** is no longer equal to the family of regular languages, we generalize RCB-grammars to Controlled bidirectional grammars (CB-grammars). Besides the properties of **C**, also a specific combination of

Closure property of CB-languages	Closure properties of **C**
(marked) union	marked union
concatenation	concatenation, left- and right-marking
marked concatenation	concatenation, left-marking
Kleene +	concatenation, left-marking, Kleene $*$, Kleene +
marked Kleene +	concatenation, left-marking, Kleene $*$
Kleene $*$	union, concatenation, left-marking, Kleene $*$
marked Kleene $*$	union, concatenation, left-marking
intersection by a regular set	union, concatenation, Kleene $*$, reversal, finite substitution
homomorphism	union, concatenation, Kleene $*$, homomorphism
regular substitution	union, concatenation, Kleene $*$, homomorphism
context-free substitution	union, concatenation, Kleene $*$, homomorphism
substitution	union, concatenation, Kleene $*$, homomorphism
union with a regular set	$P^* \in \mathbf{C}$
inverse homomorphism	union, concatenation, Kleene $*$, reversal, finite substitution, $P^* \in \mathbf{C}$
substitution into a regular set	union, concatenation, Kleene $*$, marked union, left- and right-marking

Table 2.

modes is necessary to establish each closure property for CB-languages. These modes are not included in the table, but can be extracted in a direct way from the results in Section 3. We conclude this subject with a final remark about the mode RN/B/f. Since most of the closure properties of the family of RCB/RN/B/f-languages heavily depend on **C** being the family of regular control languages, cf. Proposition 2.4.(2), we cannot expect to maintain all the closure properties if we generalize to an arbitrary family **C** of control languages.

To obtain closure properties for the family of **C**-controlled bidirectional languages we often need closure under left- or right-marking. A family of languages Φ is closed under *left-* and *right-marking* if for every language $L_0 \in \Phi$ also $\{\#\}L_0 \in \Phi$ and $L_0\{\#\} \in \Phi$, respectively, where # does not occur in the alphabet of L_0.

Consequently, we can also generalize Theorem 3.6 in the following way.

Theorem 6.1. *Let* **C** *be a family of control languages such that for every alphabet P, we have* $P^* \in \mathbf{C}$.

- *The family of CB/RO/S/g-languages is a full semi-AFL if* **C** *is closed under: union, concatenation, Kleene* $*$*, reversal and finite substitution.*
- *The family of CB/RO/S/f-languages is a full semi-AFL closed under concatenation if* **C** *is closed under: union, concatenation, Kleene* $*$*, left- and right-marking, reversal and finite*

substitution.

- *The family of CB/RO/B/f-languages is a full AFL closed under substitution if* **C** *is closed under: union, concatenation, Kleene + and ∗, left- and right-marking, reversal and finite substitution.* □

Similarly, as a generalization of Theorem 4.4 we have the following result.

Theorem 6.2. *Let* **C** *be a family of control languages closed under ngsm-mappings. Then for each CB/RN/B/f-grammar* (G_1, C_1) *with* $C_1 \in \mathbf{C}$ *we can obtain an equivalent CB/RN/B/f-grammar* (G, C) *in weak Chomsky Normal Form (and* $C \in \mathbf{C}$*).* □

As a continuation of the present paper, future research may spend attention to the following questions. Concerning the S-mode it may be interesting to study this mode in combination with a proper appearance checking set, i.e., a production is skipped only when it occurs in a set $F \subseteq P$; cf. [15, 16, 17]. In this paper only the case $F = P$ has been considered. For the grammars introduced in this paper, it will be interesting to know whether the corresponding language families lie (properly) in the family of context-sensitive languages. So far, only a result for the family of RCB/RN/B/f-languages has been established, since this family equals the family of context-free languages; cf. Proposition 2.4.(2). In the introduction it was mentioned that the subject of this paper has been inspired by the NTS-grammars. Therefore an application of the control mechanism introduced in this paper to underlying context-free grammars which are NTS lies in hand. It may also be possible to define the NTS-property for RCB-grammars in a direct way.

Acknowledgment. I thank Peter Asveld for his criticism and for helping to prepare this paper, and Rieks op den Akker and Leo Verbeek for their suggestions.

References

1. P.R.J. Asveld: Controlled iteration grammars and full hyper-AFL's, *Inform. and Contr.* **34** (1977) 248-269.
2. P.R.J. Asveld: Space-bounded complexity classes and iterated deterministic substitution, *Inform. and Contr.* **44** (1980) 282-299.
3. P.R.J. Asveld & J. Engelfriet: Iterated deterministic substitution, *Acta Inform.* **8** (1977) 285-302.
4. P.R.J. Asveld & J. van Leeuwen: Infinite chains of hyper-AFL's, TW-memorandum No. 99 (1975), Twente University of Technology, Enschede, The Netherlands.
5. J.M. Autebert, L. Boasson & G. Sénizergues: Langages de parenthèses, langages N.T.S. et homomorphismes inverses,

   18 *(1984) 327-344.*
6. 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.
7. L. Boasson & G. Sénizergues: NTS languages are deterministic and congruential, *J. Comput. System Sci.* **31** (1985) 332-342.

8. Ch. Frougny: Simple deterministic NTS languages, *Inform. Process. Lett.* **12** (1981) 174-178.

9. S. Ginsburg: *Algebraic and Automata-Theoretic Properties of Formal Languages* (1975), North-Holland, Amsterdam.

10. S. Ginsburg & E.H. Spanier: Control sets on grammars, *Math. Systems Theory* **2** (1968) 159-177.

11. M.A. Harrison: *Introduction to Formal Language Theory* (1978), Addison-Wesley, Reading, Mass.

12. J.A. Hogendorp: Nonterminal separating macro grammars, pp. 77-87 *in:* P.R.J. Asveld & A. Nijholt (Eds.): *Essays on Concepts, Formalisms, and Tools* (1987), C.W.I. Tract no. 42, Centre for Mathematics and Computer Science, Amsterdam.

13. J.A. Hogendorp: Controlled bidirectional grammars, Memorandum INF-88-16, Department of Computing Sciences, University of Twente.

14. D.J. Rosenkrantz: Programmed grammars and classes of formal languages, *J. Assoc. Comput. Mach.* **16** (1969) 107-131.

15. A. Salomaa: On grammars with restricted use of productions, *Ann. Acad. Sci. Fennicae Ser. AI.* **454** (1969).

16. A. Salomaa: On some families of formal languages obtained by regulated derivations, *Ann. Acad. Sci. Fennicae Ser. AI.* **479** (1970).

17. A. Salomaa: *Formal Languages* (1973), Academic Press, New York.

18. G. Sénizergues: A new class of C.F.L. for which the equivalence is decidable, *Inform. Process. Lett.* **13** (1981) 30-34.

19. G. Sénizergues: The equivalence and inclusion problems for NTS languages, *J. Comput. System Sci.* **31** (1985) 303-331.

20. Turbo Prolog™ Owner′s Handbook, Borland International Inc., Scotts Valley, Ca.

21 J. van Leeuwen: Rule-Labeled Programs – A Study of a Generalization of Context-Free and Some Classes of Formal Languages (1972), Doctoral Dissertation, University of Utrecht.