Simple eco-grammar systems with prescribed teams

Maurice H. ter Beek*

Department of Computer Science, Leiden University,
P.O. Box 9512, 2300 RA Leiden, The Netherlands

e-mail: mtbeek@wi.leidenuniv.nl

Abstract

The notion of teams in grammar systems is applied to a restriction of eco-grammar systems, called simple eco-grammar systems. These systems were introduced by motivations coming from Artificial Life and (thus) consist of an environment in the form of a Lindenmayer system and agents working on this environment by context-free productions. Several different ways of forming teams of agents, with different L systems for the environment, are introduced. Furthermore, two different rewriting steps are introduced.

The forming of teams is found to increase the generative power of the underlying systems. Moreover, by proving some closure properties, one of the rewriting steps is shown to lead to a new characterization of the class of the recursively enumerable languages.

1 Introduction

Motivations coming from *Artificial Life* (AL) have resulted in a new field in the theory of grammar systems ([3]): *eco-grammar systems* ([4]). AL is the study of "man-made systems that exhibit behaviours characteristic of natural living systems" ([17]). Within AL, eco-grammar systems form a generative framework for a so-called *ecosystem*: a community of evolving agents and a shared evolving environment.

In an eco-grammar system, both the environment and the agents are described by strings and they develop according to Lindenmayer systems ([18]), which were originally introduced to describe the development of multicellular structures in biology. Since then, however, they have been exhaustively studied in computer science as well and have proved to be very rich in mathematical properties ([23]). The state of the environment of an eco-grammar system influences the development of the agents, while the agents can act on the environment. Such eco-grammar systems can model many life-like phenomena (e.g. birth, death, hibernation and overpopulation) and, moreover, viewed as generative devices they have a very large generative power, even when restricted variants (e.g. with one agent) are considered.

Here such a restricted variant, appropriately called a *simple* eco-grammar system ([4]), is considered. In these systems, a subset of the agents (the active ones) is used to rewrite the sentential form at certain places; the remaining places are rewritten by (evolution of) the environment. All of this happens in parallel and hence it can be seen as a kind of team-forming: the agents that are active form a team and

*This research was supported by a scholarship from the Hungarian Ministry of Culture and Education. Moreover, the facilities provided by the Department of General Computer Science of the Eötvös Loránd University and in particular by the Computer and Automation Research Institute of the Hungarian Academy of Sciences were essential.

rewrite parts of the string representing the environment in parallel. Here, other ways of deciding which agents to rewrite the sentential form at a certain moment in time, based on the ideas of team-forming in grammar systems (see, e.g., [16], [20] and [2]), are introduced and the result on the generative power is studied.

These resulting simple eco-grammar systems with prescribed teams are nothing more than Lindenmayer systems with teams and therefore four different classes will be defined, analogous to those of the Lindenmayer systems. Hence next to the basic class, the extended, tabled and extended tabled systems are defined. Concerning the application of the teams, two rewriting steps will be introduced, like in [2]. In the strong rewriting step, a production of each agent of the team has to be applied (the derivation coming to a halt if this is not possible), whereas in the weak rewriting step only those agents of the team (but at least one) that can rewrite a symbol from the current sentential form have to do so. The introduction of the weak rewriting step in [2] was motivated by the recent application of (generalized) eco-grammar systems as a framework for natural language generation, in [7], for which purpose the strong rewriting step is far too restrictive.

Consequently, the power of forming teams in simple eco-grammar systems is investigated. In all cases this forming of teams is found to increase strictly the generative power of the underlying (Lindenmayer) system. The power of simple eco-grammar systems with prescribed teams will be shown to lie inbetween the programmed grammars without and those with appearance checking, in the case of the strong rewriting step. In the case of the weak rewriting step, the simple eco-grammar systems with prescribed teams are placed between the programmed grammars with unconditional transfer and those with appearance checking.

Finally, a normal form for extended tabled simple eco-grammar systems with prescribed teams, operating in the weak rewriting step, is defined. With this normal form, closure of the family of languages generated by these systems under a number of closure properties will be proved. Moreover, due to some of these closure properties, these systems will be shown to equal the programmed grammars with appearance checking, in the λ-free case. When λ-productions are allowed, a new characterization of the class of recursively enumerable languages is obtained.

2 Preliminaries

In this section, some prerequisites necessary for understanding the sequel are defined. For details and unexplained notions, the reader is referred to [25] for formal languages, [9] for regulated rewriting, [23] for Lindenmayer systems and [6] for eco-grammar systems.

The set of all non-empty strings over an *alphabet* V is denoted by V^+. If the *empty string*, λ, is included, the notation becomes V^*. The *length* of a string x is denoted by $|x|$.

If $L \subseteq (V\{\lambda, c, c^2, \ldots, c^{k-1}\})^*$ for $k \geq 1$ and $c \notin V$ and h is a homomorphism defined by $h(c) = \lambda$ and $h(a) = a$ for $a \in V$ then h is called a *k-restricted homomorphism* on L.

An *inclusion* is denoted by $\subseteq$, whereas a *proper inclusion* is denoted by $\subset$.

Sometimes, the notation for a family of languages contains a λ between the brackets [and]. This means that the statement holds in the case of allowing λ-productions (indicated by the λ inbetween brackets) as well as in the case of a restriction to λ-free productions (thus neglecting the λ inbetween brackets).

To differentiate properly between sets and multisets, a *multiset* is denoted by listing its elements embraced by a $\langle$ and a $\rangle$. The notation for an *empty set* is the same as for an *empty multiset*, namely $\varnothing$. Moreover, a set or multiset *appears* in a string or sentential form iff all of its elements are part of the string.

Without definition, the family of recursively enumerable (RE) languages is used in the sequel. Its definition can be found in, e.g., [25].

A *finite automaton* is a construct $\mathcal{A} = (K, V, s_0, f, H)$, where K is a finite nonempty set of *states*, V is an alphabet, $s_0 \in K$ is the *initial state*, $f : K \times V \to 2^K$ is the *transition mapping* and $H \subseteq K$ is the set of *final states*. Such an automaton recognizes a string $x \in V^*$, where $x = y_1 y_2 \cdots y_n$ and $y_i \in V$ for $1 \leq j \leq n$, iff there exist $s_1, s_2, \ldots, s_n$ in K such that

$$s_1 \in f(s_0, y_1),\ s_{j+1} \in f(s_j, y_{j+1}) \text{ for } 1 \leq j \leq n-1,\ s_n \in H.$$

The set of all strings recognized by $\mathcal{A}$ is denoted by $L(\mathcal{A})$. The family of languages recognized by finite automata is exactly the family of regular languages.

An *ET0L system* is a construct $G = (\Sigma, \Delta, P, \omega)$, where Σ is the *total alphabet*, $\Delta \subseteq \Sigma$ is the *terminal alphabet*, P is a finite set of *tables* and $\omega \in \Sigma^+$ is the axiom. Each table $H \in P$ is *complete*, which means that for every symbol $\alpha \in V$ it contains at least one production of the form $\alpha \to \beta$, for $\beta \in V^*$.

When, in an ET0L system, $\Sigma = \Delta$, a *T0L system* is obtained; $\#P = 1$ results in an *E0L system* and when $\Sigma = \Delta$ and $\#P = 1$, a *0L system* is obtained. When, in any of those definitions, every production in P is λ-free, a propagating system is obtained. The family of languages generated by 0L, P0L, E0L, EP0L, T0L, PT0L, ET0L and EPT0L systems are denoted by $0L$, $P0L$, $E0L$, $EP0L$, $T0L$, $PT0L$, $ET0L$ and $EPT0L$, respectively.

An *ET0L system with local undirected multiset context conditions and priorities (an [m,LU,p] KVET0L system*, [24]) is a construct $G = (V, T, P, \omega)$, where V is the alphabet, T is the terminal alphabet, $\omega \in V^+$ is the string axiom and P is a finite set of complete tables of the form $\{(\tau_1, c_1, r_1), (\tau_2, c_2, r_2), \ldots, (\tau_m, c_m, r_m)\}$, in which τ_j is a production of the form $\alpha \to \beta$ with $\alpha \in V$ and $\beta \in V^*$, c_j is a context condition in the form of a multiset of symbols from the alphabet and $r_j \in \mathbb{N}$ is a priority, $1 \leq j \leq m$. An [m,LU,p] KVET0L system works as follows. For a table $H \in P$ of the form described above and $w, w' \in V^*$ it is said that w directly derives w', written as

$$\begin{array}{lll} w \Longrightarrow w' & \text{iff} & w = A_1 A_2 \ldots A_n,\ w' = \alpha_1 \alpha_2 \ldots \alpha_n \text{ and for every } 1 \leq i \leq n \\ & \text{either} & (A_i \to \alpha_i, c_i, r_i) \in H \text{ where } c_i \text{ appears in } w \text{ and for every} \\ & & \text{other } (A_i \to \alpha_i, c_i', r_i') \in H \text{ where } c_i' \text{ appears in } w, \\ & & r_i' \geq r_i \text{ holds} \\ & \text{or} & \text{there is no } (A_i \to \alpha_i, c_i', r_i') \in H \text{ where } c_i \text{ appears in } w \text{ and} \\ & & \text{then } \alpha_i = A_i. \end{array}$$

The language generated by G is $L(G) = \{z \in T^* \mid \omega \Longrightarrow^* z\}$, where $\Longrightarrow^*$ denotes the reflexive and transitive closure of $\Longrightarrow$.

The family of languages generated by [m,LU,p] KVET0L systems (with λ-free productions in every table in P) is denoted by $[m, LU, p]\ KVET0L$ ($[m, LU, p]\ KVEPT0L$). In [24] it was proved that these systems are equal to the class of programmed grammars with appearance checking, defined below.

An *unordered scattered context grammar with appearance checking* ([19]) is a construct $G = (N, T, S, P, F)$, where N is the set of nonterminals, T is the set of terminals, $S \in N$ is the axiom, $P = \{p_1, p_2, \ldots, p_n\}$ is a finite set of *rules* (rules are of the form $p_i : (\alpha_1, \alpha_2, \ldots, \alpha_{m_i}) \to (\beta_1, \beta_2, \ldots, \beta_{m_i})$, where $\alpha_j \to \beta_j$ are productions over $N \cup T$) and F is a set of occurrences of productions in P, $1 \leq i \leq n$. An unordered scattered context grammar with appearance checking works as follows. For $w, w' \in (N \cup T)^*$ and $1 \leq i \leq n$ it is said that w directly derives w', written as

$w \Longrightarrow w'$ iff $w = w_1\alpha_{i_1}w_2\alpha_{i_2}\ldots w_m\alpha_{i_m}w_{m+1}$, $w' = w_1\beta_{i_1}w_2\beta_{i_2}\ldots w_m\beta_{i_m}w_{m+1}$, $p_i : (\alpha_1, \alpha_2, \ldots, \alpha_p) \to (\beta_1, \beta_2, \ldots, \beta_p) \in P$, $(\alpha_{i_1}, \alpha_{i_2}, \ldots, \alpha_{i_m})$ is a permutation of a subsequence of $(\alpha_1, \alpha_2, \ldots, \alpha_p)$, $w_l \in (N \cup T)^*$ and $1 \leq l \leq m+1$

and α_j in $\{\alpha_1, \alpha_2, \ldots, \alpha_p\}$ and not in $\{\alpha_{i_1}, \alpha_{i_2}, \ldots, \alpha_{i_m}\}$ implies that α_j is not contained in w and $\alpha_j \to \beta_j \in F$.

If $F = \varnothing$, the unordered scattered context grammar is called an *unordered scattered context grammar without appearance checking* and F is omitted from the construct. Moreover, if F contains all occurrences of productions in P, the unordered scattered context grammar is called *with unconditional transfer*. The language generated by G is $L(G) = \{w \in T^* \mid S \Longrightarrow^* w\}$, where $\Longrightarrow^*$ denotes the reflexive and transitive closure of $\Longrightarrow$.

The family of languages generated by unordered scattered context grammars with (λ-free) context-free productions in P is denoted by USC^{λ}_{ac} (USC_{ac}) in the case of grammars with appearance checking; when grammars without appearance checking are considered the subscript ac is omitted and when grammars with unconditional transfer are considered the subscript ac is replaced by ut.

A *programmed grammar* ([21]) is a construct $G = (N, T, S, P)$, where N is the set of nonterminals, T is the set of terminals, $S \in N$ is the axiom and P is a finite set of productions of the form $(r : \alpha \to \beta, \sigma(r), \varphi(r))$, where $r : \alpha \to \beta$ is a production over $N \cup T$, labelled by r. Denote by $Lab(P) = \{r \mid (r : \alpha \to \beta, \sigma(r), \varphi(r)) \in P\}$ the set of labels of productions of G. Then $\sigma(r) \subseteq Lab(P)$ is called the *success field* of production r and $\varphi(r) \subseteq Lab(P)$ the *failure field*. A programmed grammar works as follows. For $(r_1 : \alpha \to \beta, \sigma(r_1), \varphi(r_1)) \in P$ and $w, w' \in (N \cup T)^*$ it is said that w directly derives w', written as

$(w, r_1) \Longrightarrow (w', r_2)$ iff $w = w_1\alpha w_2$, $w' = w_1\beta w_2$ and $r_2 \in \sigma(r_1)$

or $w = w'$, $\alpha \to \beta$ cannot be applied to w and $r_2 \in \varphi(r_1)$.

If the failure fields are empty for every production, the programmed grammar is called *without appearance checking*, otherwise it is called *with appearance checking*. Moreover, if the success field and the failure field coincide for every labeled production, the programmed grammar is called *with unconditional transfer*. The language generated by G is $L(G) = \{w \in T^* \mid (S, r_0) \Longrightarrow^* (w, r_1),\ r_0, r_1 \in Lab(P)\}$, where $\Longrightarrow^*$ denotes the reflexive and transitive closure of $\Longrightarrow$.

The family of languages generated by programmed grammars with (λ-free) context-free productions in P is denoted by PR^{λ}_{ac} (PR_{ac}) in the case of grammars with appearance checking; when grammars without appearance checking are considered the subscript ac is omitted and when grammars with unconditional transfer are considered the subscript ac is replaced by ut.

When, in any of the grammars above, no terminal alphabet is specified, a *pure* grammar is definied. Its generated language then becomes all strings that can be reached from the axiom. When pure grammars are considered, the notations are started with a P.

3 Definitions and examples

The most general definition is given next, its subclasses are defined afterwards.

Definition 1 *An* extended tabled simple eco-grammar system with prescribed teams *of degree* n, $n \geq 1$, *is a construct*

$$\Sigma = (E, R_1, R_2, \ldots, R_n, Q_1, Q_2, \ldots, Q_m),$$

where

- $E = (V_E, T_E, P_E, \omega)$,
  - V_E *is a finite alphabet, the* environmental alphabet,
  - $T_E \subseteq V_E$, *the* terminal alphabet,
  - P_E *is a finite set of ET0L tables over* V_E, *the* evolution tables of the environment *and*
  - $\omega \in V_E^+$, *the* initial state of the environment,
- R_i, $1 \leq i \leq n$, *is an* agent *which contains a finite set of productions of the form* $\alpha \to \beta$ *for* $\alpha \in V_E$ *and* $\beta \in V_E^*$, *the* action rules *and*
- Q_j, $1 \leq j \leq m$, *is a non-empty set of agents, a* prescribed team (of agents).

(The tables of P_E *are ET0L tables and are thus* complete, *i.e. there is at least one production for every symbol from* V_E *in every table.)*

Two different rewriting steps for teams in extended tabled simple eco-grammar systems are defined, based on the two versions of parallel rewriting for colonies introduced in [8].

In the sequel, an environmental state of an eco-grammar system shall be simply called a state of an eco-grammar system.

Definition 2 *Let* w *and* w' *be two states of an eco-grammar system* $\Sigma = (E, R_1, R_2, \ldots, R_n, Q_1, Q_2, \ldots, Q_m)$. *Then a team* $Q_i = \{R_{i_1}, R_{i_2}, \ldots, R_{i_{s_i}}\}$, $1 \leq s_i \leq n$ *and* $1 \leq i \leq m$, *is used in a* strong *rewriting step as follows.*

$$\begin{array}{lll} w \overset{s}{\Longrightarrow}_{Q_i} w' & \textit{iff} & w = x_1A_1x_2A_2\ldots x_sA_sx_{s+1},\ w' = y_1\alpha_1y_2\alpha_2\ldots y_s\alpha_sy_{s+1} \\ & \textit{such that} & A_k \to \alpha_k \in R_{i_k} \textit{ for } 1 \leq k \leq s,\ i_m \neq i_n \textit{ for } m \neq n, \\ & & 1 \leq m, n \leq s \textit{ and } \{R_{i_1}, R_{i_2}, \ldots, R_{i_s}\} = Q_i \textit{ and} \\ & & y_1y_2\ldots y_{s+1} \textit{ is the result of applying a table from} \\ & & P_E \textit{ to } x_1x_2\ldots x_{s+1}. \end{array}$$

For two states w *and* w' *of* Σ *and a team* $Q_i = \{R_{i_1}, R_{i_2}, \ldots, R_{i_{s_i}}\}$, $1 \leq s_i \leq n$ *and* $1 \leq i \leq m$, *a* weak *rewriting step is defined as follows.*

$$\begin{array}{lll} w \overset{w}{\Longrightarrow}_{Q_i} w' & \textit{iff} & w = x_1A_1x_2A_2\ldots x_pA_px_{p+1},\ w' = y_1\alpha_1y_2\alpha_2\ldots y_p\alpha_py_{p+1} \\ & \textit{such that} & A_k \to \alpha_k \in R_{i_k} \textit{ for } 1 \leq k \leq p,\ i_m \neq i_n \textit{ for } m \neq n, \\ & & 1 \leq m, n \leq p \textit{ and } \{R_{i_1}, R_{i_2}, \ldots, R_{i_p}\} \subseteq \\ & & \{R_{i_1}, R_{i_2}, \ldots, R_{i_s}\} = Q_i \textit{ such that for all } R_{i_q} \in \\ & & Q_i \setminus \{R_{i_1}, R_{i_2}, \ldots, R_{i_p}\} \textit{ there exists no production} \\ & & \alpha \to \beta \in R_{i_q} \textit{ such that } \alpha \in x_1x_2\ldots x_{p+1} \textit{ and} \\ & & y_1y_2\ldots y_{p+1} \textit{ is the result of applying a table from} \\ & & P_E \textit{ to } x_1x_2\ldots x_{p+1}. \end{array}$$

(In both variants, every rewriting step uses a team, i.e. if no team can be used the derivation comes to a halt. Here, "using a team" means applying at least one agent's production.)

The language generated by Σ and operating in the weak ($- = w$) or strong ($- = s$) rewriting step, is

$$L_-(\Sigma)=\{z \in T_E^* \mid \omega \stackrel{-}{\Longrightarrow}_{Q_{i_1}} w_{i_1} \stackrel{-}{\Longrightarrow}_{Q_{i_2}} \cdots \stackrel{-}{\Longrightarrow}_{Q_{i_k}} w_{i_k} = z,\ 1 \leq i_j \leq k,\ 1 \leq j \leq n\}.$$

Hence, a team can be used in two different derivation modes. If the strong rewriting step is used, a production from each agent of the team has to be applied. This means that if an agent of a team is unable to rewrite a symbol of the current sentential form, this team cannot be applied.

In the weak rewriting step, only every agent of the team that is able to rewrite the current sentential form has to do so. In the above sketched situation this would mean the application of a production from every agent of the team, except from those unable to rewrite a symbol of the current sentential form.

Definition 3 *The family of languages generated by extended tabled simple eco-grammar systems of degree n with prescribed teams and operating in the strong (weak) rewriting step, is denoted by PT_sETSEG_n (PT_wETSEG_n) when restricted to λ-free productions, otherwise a λ is added to the notation. Furthermore, denote $PT_sETSEG = \bigcup_{n\geq 1} PT_sETSEG_n$ and $PT_wETSEG = \bigcup_{n\geq 1} PT_wETSEG_n$, respectively. Similar notations apply when not restricted to λ-free productions.*

Extended tabled simple eco-grammar systems without agents correspond to the underlying ET0L system. Consider by definition $PT_sETSEG_0^\lambda = PT_wETSEG_0^\lambda = ET0L$ and likewise for the λ-free case.

Definition 4 *An extended tabled simple eco-grammar system with prescribed teams, $\Sigma = ((V_E, T_E, P_E, \omega), R_1, R_2, \ldots, R_n, Q_1, Q_2, \ldots, Q_m)$, is called*

(1) A tabled simple eco-grammar system with prescribed teams if $V_E = T_E$. Then the E in the notations of Definition 3 is omitted.

(2) A extended simple eco-grammar system with prescribed teams if $\#P_E = 1$. Then the T in the notations of Definition 3 is omitted.

(3) A simple eco-grammar system with prescribed teams if $V_E = T_E$ and $\#P_E = 1$. Then both the E and the T in the notations of Definition 3 are omitted.

Again, simple eco-grammar systems without agents correspond to the underlying L systems. Hence, consider by definition $PT_sTSEG_0^\lambda = PT_wTSEG_0^\lambda = T0L$, $PT_sTSEG^\lambda \subseteq PT_sETSEG^\lambda$ and $PT_wTSEG^\lambda \subseteq PT_wETSEG^\lambda$ and likewise for the other eco-grammar systems of Definition 4 and for the λ-free cases as well.

Some examples are presented next to illustrate the above definitions.

Example 1 *The following simple eco-grammar systems with prescribed teams*

$$\begin{aligned}
\Sigma_1 &= (\{a\}, \{a \to aa\}, a), \{a \to aa\}, \{a \to aa\}), \\
\Sigma_2 &= (\{a\}, \{a \to aa\}, a^4), \{a \to a\}, \{a \to a\}, \{\{a \to a\}, \{a \to a\}\}) \text{ and} \\
\Sigma_3 &= (\{a,b,c\}, T, c^3), R_1, R_2, \ldots, R_6, \{R_1, R_2, R_3\}, \{R_4, R_5, R_6\}), \text{ where} \\
&\quad T = \{a \to a, b \to b, c \to c\},\ R_1 = R_2 = R_3 = \{c \to ca\} \text{ and} \\
&\quad R_4 = R_5 = R_6 = \{c \to cb\}
\end{aligned}$$

generate the languages

$$\begin{aligned}
L_s(\Sigma_1) &= L_w(\Sigma_1) = \{a^{2^n} \mid n \geq 0\}, \\
L_s(\Sigma_2) &= L_w(\Sigma_2) = \{a^2\}\{a^{2^n} \mid n \geq 1\} \text{ and} \\
L_s(\Sigma_3) &= L_w(\Sigma_3) = \{(cw)^3 \mid w \in \{a,b\}^*\}.
\end{aligned}$$

Finally, a more enhanced example is presented.

Example 2 *Consider the extended tabled simple eco-grammar system*

$$\Sigma_4 = ((V_E, T_E, P_E, S'), R_1, R_2, \ldots, R_7, \{R_1\}, \{R_2, R_3\}, \{R_4\}, \{R_5, R_6\}, \{R_7\}),$$

where

- $V_E = \{S, S', A, A', B, B', C, F, X, X', Y, a, b, c\}$,
- $T_E = \{a, b, c\}$,
- $P_E = \{T_1, T_2, T_3\}$, *where*
  - $T_1 = \{A \to A', B \to bC\} \cup \{\alpha \to F \mid \alpha \in \{S', A', B', X, X', Y\}\} \cup \{\alpha \to \alpha \mid \alpha \in \{S, C, F\} \cup T_E\}$,
  - $T_2 = \{C \to B, Y \to X, X \to F, S' \to F\} \cup \{\alpha' \to \alpha \mid \alpha \in \{A, B, X\}\} \cup \{\alpha \to \alpha \mid \alpha \in \{S, A, B, F\} \cup T_E\}$ *and*
  - $T_3 = \{B \to c, X \to c\} \cup \{\alpha \to F \mid \alpha \in \{S', A', B', X', Y\}\} \cup \{\alpha \to \alpha \mid \alpha \in \{S, A, C, F\} \cup T_E\}$.
- $R_1 = \{S' \to A'B'S', S' \to A'B'X'\}$,
- $R_2 = \{A \to a\}$,
- $R_3 = \{X \to Y\}$,
- $R_4 = \{C \to B\}$,
- $R_5 = \{B \to c\}$,
- $R_6 = \{X \to c\}$ *and*
- $R_7 = \{A' \to A\}$.

It is left to the reader to verify that as soon as the "failure" symbol F is introduced in the sentential form or as soon as a sentential form whose only nonterminal occurrences are in one of the multisets $\langle A\rangle$, $\langle B\rangle$, $\langle C\rangle$, $\langle Y\rangle$, $\langle A, B\rangle$, $\langle A, Y\rangle$, $\langle B, Y\rangle$, $\langle A', C\rangle$ or $\langle A, B, Y\rangle$, a terminal string can no longer be obtained. When a sentential form containing only (one or more) nonterminals of one of these multisets is reached, this situation is identified by the multiset and further investigation becomes useless.

Now the derivations of Σ_4 are as follows. From S' one obtains sentential forms that are part of the regular expression $(AB)^*A'B'(S' + X')$ by applying agent R_1 and table T_2 or otherwise an F is introduced. To obtain a terminal string, one can continue with an application of another team than the one containing R_1 from every such sentential form that contains X'. Then the next step consists of using R_7 (no other agent can be used) and T_2 (or an F is introduced) and thus leads to $(AB)^iX$ for an $i \geq 1$.

Consider a sentential form $(AB)^iX$, $i \geq 1$. Then only $\{R_2, R_3\}$ or $\{R_5, R_6\}$ can be used. The latter leads to $\langle A', C\rangle$ (in the case of table T_1), $\langle A, B\rangle$ (in the case of table T_2) or $\langle A\rangle$ (in the case of table T_3), while the former leads to $\langle A, B, Y\rangle$ (in the case of table T_2), $\langle A, Y\rangle$ (in the case of table T_3) or, in the case of table T_1, to the sentential form $(A'bC)^{i_1}abC(A'bC)^{i_2}Y$, such that $i_1, i_2 \geq 0$ and $i_1 + i_2 + 1 = i$.

If $i = 1$, then only R_4 can be applied. This leads to $abBX$ if table T_2 is used and otherwise an F is introduced. The only way to continue is by using $\{R_5, R_6\}$ and the terminal string $abcc$ is obtained. If $i > 1$, then not only R_4, but also R_7 can be used. They both lead to an F if any of the tables T_1 or T_3 is used and to $(AbB)^{i_1}abB(AbB)^{i_2}X$, such that $i_1, i_2 \geq 0$ and $i_1 + i_2 + 1 = i$, if T_1 is used.

This process can be repeated until finally a sentential form $(ab^iB)^iX$, $i > 1$, is obtained. Then only $\{R_5, R_6\}$ can be used. This leads to $\langle C \rangle$ (in the case of table T_1), $\langle B \rangle$ (in the case of table T_2) or $(ab^ic)^ic$, $i > 1$, (in the case of table T_3). The generated language is thus

$$L_s(\Sigma_4) = \{(ab^nc)^nc \mid n \geq 1\}.$$

4 Generative power

In this section the generative power of the various kinds of eco-grammar systems, as defined in Section 3, is investigated. For all families of languages, a 1 is added as a subscript to the notation if every agent consists of only one production.

To begin with, a normal form result for several types of simple eco-grammar systems with prescribed teams operating in the strong rewriting step is presented in the next lemma. Its trivial proof is omitted.

Lemma 1 *For $X \in \{\lambda, E, T, ET\}$*

$$PT_sXSEG^{[\lambda]} = PT_{s,1}XSEG^{[\lambda]}.$$

Augmenting L systems with teams of agents strictly increases their generative power since the following theorem holds.

Theorem 1 *For $x \in \{s, w\}$*

(i) $P0L \subset PT_xSEG$ *and* $0L \subset PT_xSEG^\lambda$,

(ii) $EPOL \subset PT_xESEG$ *and* $E0L \subset PT_xESEG^\lambda$,

(iii) $PT0L \subset PT_xTSEG$ *and* $T0L \subset PT_xTSEG^\lambda$ *and*

(iv) $EPT0L \subset PT_sETSEG$ *and* $ET0L \subset PT_sETSEG^\lambda$.

Proof All inclusions follow immediately from the definitions, next their properness is proved. (*i*) In [5] it was proved that already simple eco-grammar systems without teams generate strictly more than 0L systems. (*ii*) In Example 1 it was shown that there exists a system Σ_3 such that $L_x(\Sigma_3) = \{(cw)^3 \mid w \in \{a, b\}^*\} \in PT_xSEG_6$. It was proved in [23] that this language cannot be generated by any E0L system. (*iii*) In Example 1 it was also shown that there exists a system Σ_2 such that $L_x(\Sigma_2) = \{a^2\}\{a^{2^n} \mid n \geq 1\} \in PT_xTSEG_2$. It was proved in [22] that this language cannot be generated by any T0L system. (*iv*) In Example 2 it was shown that there exists a system Σ_4 such that $L_s(\Sigma_4) = \{(ab^nc)^nc \mid n \geq 1\} \in PT_sETSEG_7$. This language cannot be generated by any ET0L system, however, since $ET0L$ is a full AFL and erasing the symbol c by a morphism leads to the language $\{(ab^n)^n \mid n \geq 1\}$, which was proved to be non-ET0L in [10]. □

The way an extended tabled simple eco-grammar system with prescribed teams works is like an enriched ET0L system. A comparison with the KVET0L systems of [24] therefore seems appropriate.

Lemma 2

$PT_sETSEG \subseteq [m, LU, p]\ KVEPT0L$ *and* $PT_sETSEG^{\lambda} \subseteq [m, LU, p]\ KVET0L$.

Proof Consider the following extended tabled simple eco-grammar system with prescribed teams operating in the strong rewriting step.

$$\Sigma = ((V_E, T_E, \{T_1, T_2, \ldots, T_k\}, \omega), R_1, R_2, \ldots, R_n, Q_1, Q_2, \ldots, Q_m).$$

According to Lemma 1 it is enough to consider Σ to have only one production per agent. Furthermore, assume that t is the maximum number of agents constituting a team. To simulate Σ, construct an [m,LU,p] KVET0L system

$$G = (V, T, P, \omega),$$

where

$$\begin{aligned} V &= V_E \cup \{[A, i, j] \mid A \in V_E, 1 \leq i \leq m, 1 \leq j \leq t\} \cup \{F\}, \\ T &= T_E \text{ and} \\ P_E &= \{H_1, H_2, \ldots, H_{k \cdot m+1}\}, \text{ as defined below.} \end{aligned}$$

For every team Q_i, $1 \leq i \leq m$, construct the table

$$\begin{aligned} [H_i, 0] = \{&(\alpha_1 \to [\beta_1, i, 1], c, 1), (\alpha_2 \to [\beta_2, i, 2], c, 1), \ldots, (\alpha_{s_i} \to [\beta_{s_i}, i, s_i], c, 1), \\ &\{(\alpha \to \alpha, \varnothing, 1) \mid \alpha \in V_E \cup F\}, \{([\alpha, i, j] \to F, \varnothing, 1) \mid [\alpha, i, j] \in V\} \mid \\ &c = \langle \alpha_1, \alpha_2, \ldots, \alpha_{s_i} \rangle, \alpha_r \to \beta_r \in R_{i_r}, Q_i = \{R_{i_1}, R_{i_2}, \ldots, R_{i_{s_i}}\}, \\ &1 \leq r \leq s_i \leq t, 1 \leq i \leq m\} \end{aligned}$$

and, for $1 \leq j \leq k$, the tables

$$\begin{aligned} [H_i, j] = \{&([\beta_1, i, 1] \to F, \{[\beta_1, i, 1], [\beta_1, i, 1]\}, 1), \\ &([\beta_2, i, 2] \to F, \{[\beta_2, i, 2], [\beta_2, i, 2]\}, 1), \ldots, \\ &([\beta_{s_i}, i, s_i] \to F, \{[\beta_{s_i}, i, s_i], [\beta_{s_i}, i, s_i]\}, 1), ([\beta_1, i, 1] \to \beta_1, c, 2), \\ &([\beta_2, i, 2] \to \beta_2, c, 2), \ldots, ([\beta_{s_i}, i, s_i] \to \beta_{s_i}, c, 2), \{(\alpha \to \beta, c, 2) \mid \\ &\alpha \to \beta \in T_j\}, (F \to F, \varnothing, 2), \{([\alpha, i, j] \to F, \varnothing, 3) \mid [\alpha, i, j] \in V\} \mid \\ &c = \langle [\beta_1, i, 1], [\beta_2, i, 2], \ldots, [\beta_{s_i}, i, s_i] \rangle, \alpha_r \to \beta_r \in R_{i_r}, \\ &Q_i = \{R_{i_1}, R_{i_2}, \ldots, R_{i_{s_i}}\}, 1 \leq r \leq s_i \leq t, 1 \leq i \leq m\}. \end{aligned}$$

Throughout the proof, the symbol F is used as a "failure" symbol, i.e. once introduced it can never be replaced by anything else than itself. In this construct, a rewriting step of Σ is simulated by the application of two tables from G.

First table $[H_i, 0]$ simulates team i by replacing one (or more, but this will be shown to be leading to the introduction of F) occurrence of the left-hand side of the one production of each agent of the team by its marked (indicating the simulated team and agent) right-hand side iff all these left-hand sides occur in the sentential form, copying all other symbols of the original alphabet and F without any context condition and replacing all marked symbols by F. For the context condition requiring the presence of every left-hand side of every agent of the simulated team in the sentential form, a multiset is needed since two (or more) agents could be able to rewrite the same symbol. It is clear that iff the condition is fulfilled all left-hand sides are rewritten. If any marked symbols occur, they must have been introduced

by a table simulating another team and first the other table for that team should be used, justifying replacing these marked symbols by F. All other symbols of the original alphabet and F need to be copied to obtain a complete table.

Then table $[H_i, j]$ simulates the rewriting (by an environmental table j) of all symbols in a sentential form not affected by team i. This is achieved by the following steps, in order of priority. (A production can only be applied when no production with a higher priority exists with a context condition that appears in the sentential form.)

(1) All marked symbols of the simulated team are replaced by F iff they occur more than once in the sentential form.

(2) All marked symbols of the simulated team are replaced by their unmarked version iff they all occur once in the sentential form. All unmarked symbols of the original alphabet are replaced by any of their right-hand sides that appeared in the environmental table, iff all marked symbols of the simulated team occur once in the sentential form. Finally, F is copied.

(3) All marked symbols are replaced by F.

Productions with priority 2 can only be applied if there is no marked symbol of the simulated team occurring more than once. Either there never was any such marked symbol occurring twice or it has been replaced by F by a production with priority 1. In the latter case, no terminal string can be derived. Hence assume there is no marked symbol of the simulated team occurring twice when productions with priority 2 are used. However, the condition that all marked symbols occur once is still necessary for productions of priority 2. The reason is that otherwise table $[H_i, j]$ could be used without $[H_i, 0]$ being used first, i.e. only simulating a table and no team. The productions of priority 2 unmark all these once occurring marked symbols. Furthermore, all unmarked symbols are rewritten according to the simulated table and F is copied. Then, when no more production with priority 2 can be used, the productions with priority 3 replace any remaining marked symbols by F.

From the description above it can be seen that every table is complete. It is also clear that the process described above can be repeated for any other team. Moreover, after simulating a team, any environmental table can be simulated. If after the simulation of a team (an environmental table) another team (environmental table) is simulated, no terminal string can be derived since F will be introduced. Also if the simulation of a team is followed by the simulation of a "wrong" environmental table, an F is introduced. This "failure" symbol is even introduced in many more occasions to enforce rewriting to take place as described.

When not restricted to λ-free productions, the proof continues to hold. In fact, it can be simplified by introducing separate marker symbols instead of augmenting the original symbols, thus guiding the simulation and replacing them by λ in the end. Hence it is clear that $L(G) = L(\Sigma)$ and the lemma is proved. □

From the equality $[m, LU, p]\ KVEPT0L = PR_{ac}$, which is proved in [24], one is lead to investigate the relation between programmed grammars without appearance checking and extended tabled simple eco-grammar systems with prescribed teams. Since $USC^{[\lambda]} = PR^{[\lambda]}$ (see, e.g., [9]), the next lemma suffices.

Lemma 3

(i) $USC^{[\lambda]} \subset PT_{s,1}ESEG^{[\lambda]}$ *and*

(ii) $USC_{ut}^{[\lambda]} \subseteq PT_{w,1}ESEG^{[\lambda]}$.

Proof The inclusions of (i) can be proved by a straightforward simulation, those from (ii) will be shown to follow from them.

For every scattered context rule $p : (\alpha_1, \alpha_2, \ldots, \alpha_s) \rightarrow (\beta_1, \beta_2, \ldots, \beta_s)$, the team $Q_p = \{R_1, R_2, \ldots, R_s\}$ with $R_j = \{\alpha_j \rightarrow \beta_j\}$, for $1 \leq j \leq s$, is constructed. A parallel rewriting step of an unordered scattered context grammar is thus simulated by a parallel rewriting step of a team, with its agents being exactly the productions in the scattered context rule. It is thus clear that the restriction of only one production per agent is obtained. The symbols in the sentential form that are not affected by the team are copied by the environmental table. Hence $L(\Sigma) = L(G)$. Clearly, the proof continues to hold when not restricted to λ-free productions.

In Example 1 it was shown that $L_1 = \{a^{2^n} \mid n \geq 0\} \in PT_sESEG$. In [15], however, it was proved that $L_1 \notin PR^{\lambda}$. Since $USC^{[\lambda]} = PR^{[\lambda]}$ (see, e.g., [9]), both inclusions of (i) are proper ones.

The proof of (ii) follows immediately from the above proof if the simulating simple eco-grammar system with prescribed teams is operating in the weak rewriting step. The (possible) "overpassing" of a production from a scattered context rule then results in the "overpassing" under the same restrictions (i.e. its left-hand side does not appear anywhere in the sentential form) of an agent. □

An immediate corollary of this lemma is a strengthening of Theorem 1.

Corollary 1 *$EPT0L \subset PT_wETSEG$ and $ET0L \subset PT_wETSEG^{\lambda}$.*

Proof The strict inclusions $EPT0L \subset O$ and $ET0L \subset O^{\lambda}$, where O denotes the family of languages generated by the ordered grammars (with context-free productions) as introduced in [14], can be found in [9]. Furthermore, in [11], $O^{[\lambda]} \subset PR_{ut}^{[\lambda]}$ is proved. In [13], finally, $PR_{ut}^{[\lambda]} = USC_{ut}^{[\lambda]}$ is proved. □

A second corollary of this lemma is the following.

Corollary 2

$$\begin{aligned} &(i) \quad PUSC^{[\lambda]} \subseteq PT_{s,1}SEG^{[\lambda]} \text{ and} \\ &(ii) \quad PUSC_{ut}^{[\lambda]} \subseteq PT_{w,1}SEG^{[\lambda]}. \end{aligned}$$

Continuing the investigation of the relation between extended tabled simple eco-grammar systems with prescribed teams and programmed grammars, the (λ-free case of) the following lemma is important.

Lemma 4

$$PT_wETSEG^{[\lambda]} \subseteq PR_{ac}^{[\lambda]}.$$

Proof Consider the following extended tabled simple eco-grammar system with prescribed teams operating in the weak rewriting step.

$$\Sigma = ((V_E, T_E, \{T_1, T_2, \ldots, T_l\}, \omega), R_1, R_2, \ldots, R_n, Q_1, Q_2, \ldots, Q_m).$$

The alphabets are considered to consist of barred symbols only. To simulate Σ, construct the programmed grammar $G = (N, T_E, S, P)$, where $N = \{a, a' \mid \bar{a} \in V_E\} \cup \{F\}$ and P is defined in the form of *subroutines*.

The use of subroutines is similar to that in Theorem 5.1 in Part II of [25]. The interconnection of the subroutines will be presented through flow charts. The expression *out* in a success or failure field indicates that the next subroutine is called. The production labelled by 1 is always the production through which the subroutine is entered.

The first subroutine is *Start*, which is defined as follows

$$\begin{array}{lll} & \sigma & \varphi \\ 1 : S \to \omega & out & \varnothing \end{array}$$

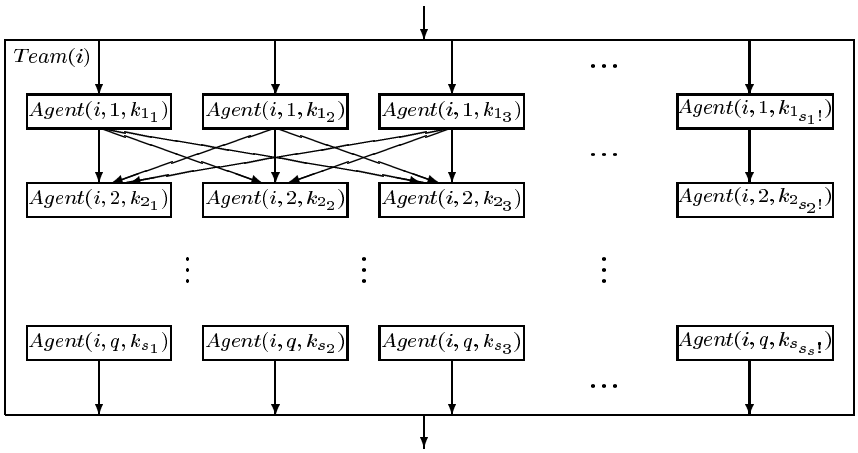
and simply introduces the (string) axiom of Σ.

For every team $Q_i = \{R_1, R_2, \ldots, R_q\}$, $1 \leq i \leq m$, some subroutines for each agent $R_j = \{\alpha_1 \to x_{1_1} x_{1_2} \ldots x_{1_{p_1}}, \alpha_2 \to x_{2_1} x_{2_2} \ldots x_{2_{p_2}}, \ldots, \alpha_s \to x_{s_1} x_{s_2} \ldots x_{s_{p_s}}\}$, $1 \leq j \leq q$, are constructed as follows. For every permutation $\pi(k)$, $1 \leq k \leq s!$ of $\{1, 2, \ldots, s\}$, construct the subroutine $Agent(i, j, k)$ as follows

$$\begin{array}{lll} & \sigma & \varphi \\ \pi(1): \ \alpha_{\pi(1)} \to x'_{\pi(1)_1} x'_{\pi(1)_2} \ldots x'_{\pi(1)_{p_{\pi(1)}}} & out & \pi(2) \\ \pi(2): \ \alpha_{\pi(2)} \to x'_{\pi(2)_1} x'_{\pi(2)_2} \ldots x'_{\pi(2)_{p_{\pi(2)}}} & out & \pi(3) \\ \vdots & \vdots & \vdots \\ \pi(s): \ \alpha_{\pi(s)} \to x'_{\pi(s)_1} x'_{\pi(s)_2} \ldots x'_{\pi(s)_{p_{\pi(s)}}} & out & out \end{array}$$

This subroutine replaces, for a certain permutation of the productions of a certain agent of a certain team, the left-hand side of one production by its primed right-hand side if this left-hand side is present in the sentential form. It tries every production of the agent in the order of the permutation. If none of the productions can be applied or as soon as one is applied, the subroutine is left.

The above subroutines for agents can be put together as depicted in the following flow chart, such that for every team Q_i, $1 \leq i \leq m$, the subroutine $Team(i)$ is defined. The flow chart is given for one team.



Any path from top to bottom can be taken and thus any production from an agent can be chosen; not only the first applicable one, as $Agent(i, j, k)$ would do if permutations were not used.

For every team $Q_i = \{R_1, R_2, \ldots, R_t\}$, $1 \leq i \leq m$, construct the subroutine $Check(i)$ as follows

	σ	φ
$1:\ x'_{1_1} \to x'_{1_1}$	out_σ	2
$2:\ x'_{1_2} \to x'_{1_2}$	out_σ	3
:	:	:
$p_1:\ x'_{1_{p_1}} \to x'_{1_{p_1}}$	out_σ	p_1+1
$p_1+1:\ x'_{2_1} \to x'_{2_1}$	out_σ	p_1+2
$p_1+2:\ x'_{2_2} \to x'_{2_2}$	out_σ	p_1+3
:	:	:
$p_1+p_2:\ x'_{2_{p_2}} \to x'_{2_{p_2}}$	out_σ	p_1+p_2+1
:	:	:
$p_1+p_2+\cdots+p_{s-1}+1:\ x'_{s-1_1} \to x'_{s-1_1}$	out_σ	$p_1+p_2+\cdots+p_{s-1}+2$
$p_1+p_2+\cdots+p_{s-1}+2:\ x'_{s-1_2} \to x'_{s-1_2}$	out_σ	$p_1+p_2+\cdots+p_{s-1}+3$
:	:	:
$p_1+p_2+\cdots+p_s:\ x'_{s_{p_s}} \to x'_{s_{p_s}}$	out_σ	out_φ

This subroutine checks, for a certain team, whether at least one of its agents their productions has been used. It tries every production in the team and the subroutine is left through out_σ as soon as the answer is affirmative. If every production has been tried, but none has been applied, the subroutine is left through out_φ.

For every (complete) environmental table T_h, $1 \leq h \leq l$, construct the subroutine $Table(h)$. Its programmed rules will not be specified, but they are those that would be constructed to simulate the E0L system $G' = (V', T', P', \omega')$, where

$$\begin{aligned} V' &= V_E \cup \{A' \mid A \in V_E\} \cup \{F\}, \\ T' &= \{a' \mid a \in V_E\}, \\ P' &= \{\beta \to y'_1 y'_2 \ldots y'_t \mid \beta \to y_1 y_2 \ldots y_t \in T_h\} \cup \{\gamma' \to F \mid \gamma' \in V'\} \text{ and} \\ \omega' &\text{ is the sentential form upon entering } Table(h), \end{aligned}$$

by a programmed grammar with appearance checking. Due to the known inclusion $E0L \subset PR_{ac}$ (see, e.g., [23]) this is possible. The subroutine is entered through the same production as the simulating programmed grammar's and it can be left through any production at any time. The subroutine *Check* (see below) will check that it does not do so before only primed symbols appear in the sentential form.

Assume, without loss of generality, $V_E = \{A, B, \ldots, Z\}$ and $T_E = \{a, b, \ldots, z\}$. Then the next subroutines are, from left to right, *Check*, *Unprime* and *Decode*.

	σ	φ
$1:\ A \to F$	1	2
$2:\ B \to F$	2	3
:	:	:
$Z:\ Z \to F$	Z	*out*

	σ	φ
$1:\ A' \to A$	1	2
$2:\ B' \to B$	2	3
:	:	:
$Z:\ Z' \to Z$	Z	*out*

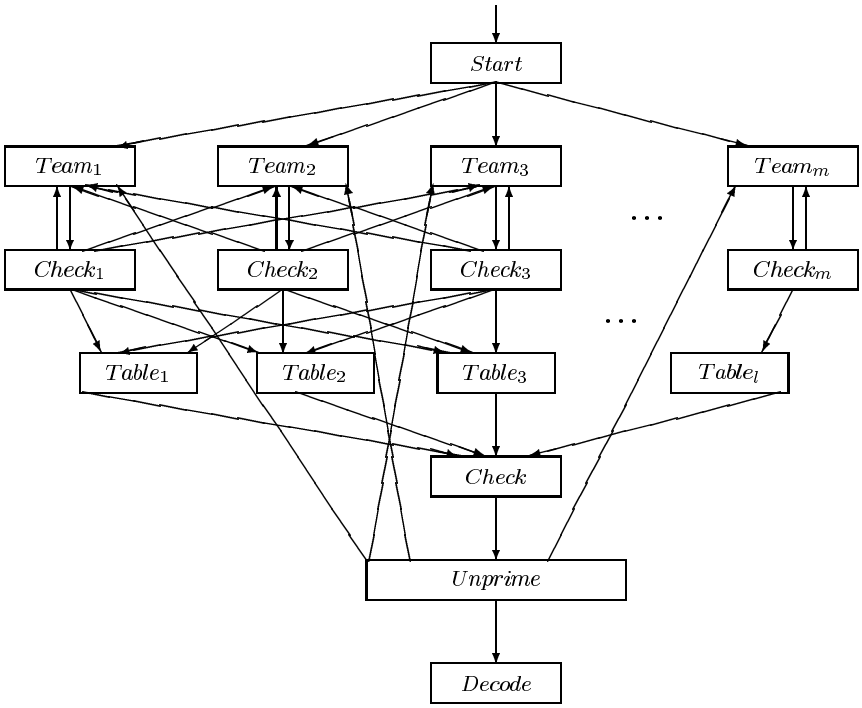
	σ	φ
$1:\ a \to \bar{a}$	1, 2	2
$2:\ b \to \bar{b}$	2, 3	3
:	:	:
$z:\ z \to \bar{z}$	z	z

The subroutine *Check* checks whether there is still an unprimed symbol in the sentential form, introducing an F if so. The success fields are only non-empty to satisfy the definition of a programmed grammar, but can be anything since once an F has been introduced the derivation can never be successfully ended anymore.

The subroutine *Unprime* removes all primes.

Finally, subroutine *Decode* is defined to decode all symbols to be able to end the derivation of the programmed grammar. Obviously, it only decodes symbols of the terminal alphabet of Σ. This subroutine can never be left once it is entered, thus when it is entered with nonterminal symbols, it can never lead to a terminal string. When the sentential form contains only terminal symbols upon entering *Decode*, a terminal string of the programmed grammar is obtained.

These are all the subroutines necessary to simulate Σ. The programmed grammar is now defined by the following flow chart



In this flow chart, all arrows coming from $Check_i$, $1 \leq i \leq m$, are labelled. Those pointing upwards by out_φ and those pointing downwards by out_σ.

It is clear that these subroutines can be combined into one long list of productions and that the proof remains valid in the case of λ-free productions. Thus $L(G) = L(\Sigma)$ and the proof is completed. □

A corollary of the results above is presented in the next theorem, which is the main result of this section.

Theorem 2

$$(i) \quad PR^{[\lambda]} \subset PT_sESEG^{[\lambda]} \subseteq PT_sETSEG^{[\lambda]} \subseteq PR_{ac}^{[\lambda]} \text{ and}$$

$$(ii) \quad PR_{ut}^{[\lambda]} \subseteq PT_wESEG^{[\lambda]} \subseteq PT_wETSEG^{[\lambda]} \subseteq PR_{ac}^{[\lambda]}.$$

Proof The equality $PR_{ac}^{\lambda} = RE$ and those between unordered scattered context grammars and programmed grammars, with or without λ-productions, with or without appearance checking and with or without unconditional transfer, are well-known (see, e.g., [9]). Furthermore, $[m, LU, p]\ KVEPT0L = PR_{ac}$ was proved in [24], whereas the inclusion $[m, LU, p]\ KVET0L \subseteq PR_{ac}^{\lambda} = RE$ is obvious. Hence all inclusion from left to right were proved in Lemma 2, 3 and 4 or follow directly from the definitions. □

5 Closure properties

Before presenting the main result of this section, a normal form for extended tabled simple eco-grammar systems is defined.

Definition 5 *An extended tabled simple eco-grammar system, with prescribed teams and operating in the weak rewriting step,*

$$\Sigma = ((V_E, T_E, P_E, \omega), R_1, R_2, \ldots, R_n, Q_1, Q_2, \ldots, Q_m),$$

is said to be in normal form *if*

(1) $\omega \in V_E \setminus T_E$, ω *does not appear on the right-hand side of any production in any team and if* ω *appears at the right-hand side of some production in some table, then this is a production* $\omega \to \omega$,

(2) there exists a unique team Q_I *among* Q_i, $1 \leq i \leq m$, *called the* initial team, *consisting of only one agent and this agent's only production is* $\omega \to \alpha$ *for* $\alpha \in V_E^*$ *and* $\alpha \neq \omega$,

(3) there exists a unique team Q_T *among* Q_i, $1 \leq i \leq m$, *called the* terminal team, *different from* Q_I, *such that this team's agents contain only productions of the form* $\alpha \to \beta$, *for* $\alpha \in (V_E \setminus T_E) \setminus \{\omega\}$ *and* $\beta \in T_E \cup \{F\}$,

(4) there exists a unique table T_T *in* P_E, *called the* terminal table, *such that it contains only productions of the form* $\alpha \to \beta$, *for* $\alpha \in (V_E \setminus T_E) \setminus \{\omega\}$ *and* $\beta \in T_E \cup \{F\}$,

(5) If $\alpha \to \beta \in T$, *for* $\alpha \in T_E$ *and* $T \in P_E$, *then* $\beta = F$ *and if* $\alpha \to \beta \in T$, *for* $\alpha \in V_E \setminus T_E$ *and* $T \in P_E \setminus \{T_T\}$, *then* $\beta \in (V_E \setminus T_E)^*$ *and*

(6) The above mentioned symbol F *is a distinguished symbol in* $V_E \setminus T_E$ *such that* $F \to F$ *is the only production with* F *as its left-hand side in any team or table of* Σ.

Theorem 3 *For every language in* $PT_wETSEG^{[\lambda]}$ *there exists an extended tabled simple eco-grammar system with prescribed teams, operating in the weak rewriting step and in normal form, generating it.*

Proof Consider the following system Σ such that $L_w(\Sigma) \in PT_wETSEG^{\lambda}$.

$$\Sigma = ((V_E, T_E, P_E, \omega), R_1, R_2, \ldots, R_n, Q_1, Q_2, \ldots, Q_m).$$

Denote $T'_E = \{\alpha' \mid \alpha \in T_E\}$, $V'_E = V_E \cup T'_E \cup \{S, F\}$ and let P'_E consist of every table $T \in P_E$ augmented by the productions

$$\alpha' \to F, \text{ for } \alpha' \in T'_E \cup \{S, F\}.$$

Furthermore, construct the initial team

$$Q_I = \{\{S \to \omega\}\},$$

the terminal team

$$Q_T = \{\{\alpha \to \alpha' \mid \alpha \in T_E\}, \{\alpha \to F \mid \alpha \in V'_E \setminus T_E\}\}$$

and the terminal table

$$T_T = \{\alpha \to \alpha', \beta \to F \mid \alpha \in T_E,\ \beta \in V'_E \setminus T_E\}.$$

Now the system

$$\Sigma' = ((V'_E, T'_E, P'_E \cup \{T_T\}, S), R'_1, R'_2, \ldots, R'_n, Q_1, Q_2, \ldots, Q_m, Q_I, Q_T),$$

where $R'_1, R'_2, \ldots, R'_n$ are the agents implied by the above construction, obviously satisfies all six conditions for the normal form as listed in Definition 5. Moreover, it is clear that $L_w(\Sigma') = L_w(\Sigma)$ and the theorem thus holds. □

This normal form is important since it leads to the following result, the easy proof of which is left to the reader.

Lemma 5 *If Σ is an extended simple eco-grammar system, with prescribed teams and operating in the weak rewriting step, in normal form and $H \in L(\Sigma)$, then in every derivation of H in Σ only the first step uses the initial team (and no table) and only the last step uses the terminal team and the terminal table.*

With this normal form, some closure properties with interesting consequences can be proved.

Lemma 6 *The families $PT_wETSEG^{[\lambda]}$ are closed under intersections with regular languages.*

Proof Consider the following extended tabled simple eco-grammar system with prescribed teams and operating in the weak rewriting step.

$$\Sigma = ((V_E, T_E, \{T_1, T_2, \ldots, T_k\}, \omega), R_1, R_2, \ldots, R_n, Q_1, Q_2, \ldots, Q_m).$$

According to Theorem 3, it can be assumed that Σ is in normal form. Moreover, consider the (deterministic) finite automaton

$$\mathcal{A} = (K, T_E, s_0, f, H).$$

Let $\bar{V}_E = \{[s, \alpha, s'] \mid s, s' \in K,\ \alpha \in V_E\}$ and $V'_E = \bar{V}_E \cup T_E \cup \{S, F\}$ and define for every agent R_i, $1 \leq i \leq n$, the set

$$C(R_i) = \{[s, \alpha, s'] \to [s, \beta_1, s_{l_1}][s_{l_1}, \beta_2, s_{l_2}] \cdots [s_{l_{p-1}}, \beta_p, s'] \mid \alpha \to \beta_1\beta_2 \cdots \beta_p \in R_i,\ s, s', s_{l_1}, s_{l_2}, \ldots, s_{l_{p-1}} \in K\}.$$

For the initial team $Q_I \in \{Q_1, Q_2, \ldots, Q_m\}$, construct the team

$$\begin{aligned} Q'_I = \{&\{S \to \gamma\}, \{S \to [s_0, \beta_1, s_{l_1}][s_{l_1}, \beta_2, s_{l_2}] \cdots [s_{l_{p-1}}, \beta_p, s]\} \mid \\ &\gamma = \lambda \text{ if } \lambda \in L(\mathcal{A}) \text{ and } \gamma = S \text{ otherwise}, \\ &S \to \beta_1\beta_2 \cdots \beta_p \in Q_I,\ s_{l_1}, s_{l_2}, \ldots, s_{l_{p-1}} \in K,\ s \in H\} \end{aligned}$$

and for every other team Q_i, $1 \leq i \leq m$ and $Q_i \neq Q_I$, construct the team

$$Q'_i = \{C(R) \mid R \text{ is an agent of } Q_i\}.$$

Furthermore, construct the team

$$Q'_T = \{\{[s, \alpha, s'] \to \alpha\}, \{[s, \beta, s'] \to F\} \mid \alpha \in T_E,\ s' = f(s, \alpha),\ s' \neq f(s, \beta),\ s, s' \in K\}.$$

For every table T_j, $1 \leq j \leq k$, construct the table

$$\begin{aligned} T'_j = \{&[s, \alpha, s'] \to [s, \beta_1, s_{l_1}][s_{l_1}, \beta_2, s_{l_2}] \cdots [s_{l_{p-1}}, \beta_p, s'] \mid \\ &\alpha \to \beta_1\beta_2 \cdots \beta_p \in T_j,\ s, s', s_{l_1}, s_{l_2}, \ldots, s_{l_{p-1}} \in K\} \cup \\ &\{\beta \to F \mid \beta \in V'_E \setminus \bar{V}_E\}. \end{aligned}$$

Furthermore, construct the table

$$T'_T = \{[s,\alpha,s'] \to \alpha \mid \alpha \in T_E,\ s' = f(s,\alpha),\ s,s' \in K\} \cup \\ \{[s,\alpha,s'] \to F \mid s' \neq f(s,\alpha),\ s,s' \in K\} \cup \{\beta \to F \mid \beta \in V'_E \setminus \bar{V}_E\}$$

Now consider the extended tabled simple eco-grammar system, with prescribed teams and operating in the weak rewriting step,

$$\Sigma' = ((V'_E, T_E, \{T'_1, T'_2, \ldots, T'_k, T'_T\}, S), R'_1, R'_2, \ldots, R'_n, Q'_1, Q'_2, \ldots, Q'_m, Q'_T),$$

where $R'_1, R'_2, \ldots, R'_n$ are the agents which are implied by the above constructions. The construction used is the standard "triple" construction, so a formal proof of $L_w(\Sigma') = L_w(\Sigma) \cap L(\mathcal{A})$ is omitted. This finishes the proof of this lemma. □

Lemma 7 *The family PT_wETSEG is closed under restricted homomorphisms.*

Proof Consider the following λ-free extended tabled simple eco-grammar system, with prescribed teams and operating in the weak rewriting step,

$$\Sigma = ((V_E, T_E, \{T_1, T_2, \ldots, T_l\}, \omega), R_1, R_2, \ldots, R_n, Q_1, Q_2, \ldots, Q_m)$$

and the homomorphism $h : V_T^* \to V_T'^*$, which is k-restricted with respect to $L(\Sigma)$. Note that $h(x) \neq \lambda$, for $x \in V_T^*$ and $|x| \geq k$. According to Theorem 3, it can be assumed that Σ is in normal form. Then construct the extended tabled simple eco-grammar system, with prescribed teams and operating in the weak rewriting step,

$$\Sigma' = ((V'_E, T'_E, \{T'_1, T''_1, T'_2, T''_2 \ldots, T''_l, T'_T\}, \omega), \\ R'_1, R'_2, \ldots, R'_n, Q'_1, Q''_1, Q'_2, Q''_2, \ldots, Q''_m, Q'_T),$$

as follows. Let $V'_E = \{[\alpha], [\alpha]' \mid \alpha \in (V_E \cup T_E)^+,\ |\alpha| \leq 2k\} \cup \{S', F\}$, $N_1 = \{[\alpha] \mid \alpha \in (V_E \cup T_E)^+,\ k \leq |\alpha| \leq 2k-1\}$ and $N_2 = \{[\alpha] \mid \alpha \in (V_E \cup T_E)^+,\ k \leq |\alpha| \leq 2k\}$.

Replace the initial team $Q_I \in \{Q_1, Q_2, \ldots, Q_m\}$, by the teams

$$Q'_I = Q''_I = \{\{S' \to h(\alpha), S' \to [\beta] \mid \alpha \in L(\Sigma),\ |\alpha| \leq k,\ [\beta] \in N_2,\ S \Rightarrow^* \beta\}\}$$

and for every other team Q_i, $1 \leq i \leq m$ and $Q_i \neq Q_I$, construct the teams

$$Q'_i = \{\{[w_1Aw_2] \to [w_1xw_2]' \mid [w_1Aw_2] \in N_1,\ A \to x \in R\} \mid R \text{ is an agent of } Q_i\}$$

and

$$Q''_i = \{\{[\alpha]' \to [\beta][\gamma] \mid [\alpha] \in N_2,\ [\beta], [\gamma] \in N_1,\ \alpha = \beta\gamma\}\}.$$

Furthermore, construct the team

$$Q'_T = \{\{[\alpha] \to h(\alpha) \mid \alpha \in T_E^+,\ k \leq |\alpha| \leq 2k-1\}, \\ \{[\beta] \to F \mid [\beta] \in V'_E \setminus \{[\alpha] \mid \alpha \in T_E^+\}\}\}.$$

For every table T_j, $1 \leq j \leq l$, construct the tables

$$T'_j = \{[w_1Aw_2] \to [w_1xw_2]' \mid [w_1Aw_2] \in N_1,\ A \to x \in T_j\} \cup \{[\alpha] \to [\alpha]' \mid \\ [\alpha] \in V'_E,\ \neg(\exists\ A \to x \in T_j,\ A \text{ appears in } \alpha,\ [\alpha] \in N_1)\} \cup \\ \{[\alpha]' \to F \mid [\alpha]' \in V'_E\}$$

and

$$T''_j = \{[\alpha]' \to [\beta][\gamma] \mid [\alpha] \in N_2,\ [\beta],[\gamma] \in N_1,\ \alpha = \beta\gamma\} \cup \{[\alpha]' \to [\alpha] \mid [\alpha]' \in V'_E,\ \neg([\alpha] \in N_2,\ [\beta],[\gamma] \in N_1,\ \alpha = \beta\gamma)\} \cup \{[\alpha] \to F \mid [\alpha] \in V'_E\}.$$

Furthermore, construct the table

$$T'_T = \{[\alpha] \to h(\alpha)\} \mid \alpha \in T_E^+,\ k \leq |\alpha| \leq 2k-1\} \cup \{\alpha \to F \mid \alpha \in V'_E\}.$$

Finally, $R'_1, R'_2, \ldots, R'_n$ are the agents that are implied by the above constructions. For each derivation $S \Rightarrow^* x$ according to Σ, the productions of the teams Q'_i and Q''_i (including $Q'_I = Q''_I$) and the tables T'_j and T''_j, $1 \leq i \leq m$ and $1 \leq j \leq l$, can construct a derivation $S \Rightarrow^* \alpha$ for some $\alpha = [\alpha_1][\alpha_2]\cdots[\alpha_p]$ such that $x = \alpha_1\alpha_2\ldots\alpha_p$, $k \leq |\alpha_f| \leq 2k-1$ and $1 \leq f \leq p$. When none of the initial teams is used, this is accomplished by interchanging the application of primed teams and tables with double primed ones. Note that at every step only one of the two primed versions can be used without introducing an F. Eventually, this x is a terminal string (or the sentential forms contains an F) and team Q'_T and table T'_T will rewrite α into $h(x)$. This team Q'_T can only be used in the final step of the derivation when the sentential form consists of only terminals, otherwise an F is introduced.

Throughout the proof, this symbol F is used as a "failure" symbol, i.e. once introduced it can never be replaced by anything else than itself. This implies that a rewriting step of Q'_T cannot be used to continue a derivation which would be blocked in the case of Σ. Note that the double primed teams and tables do not influence the generated language. Hence each derivation in Σ' corresponds to a derivation according to Σ, except that the brackets [and] occur in the sentential forms. It is clear that Σ' is λ-free and thus $L(\Sigma') = h(L(\Sigma)) \in PT_wETSEG$, which completes the proof. □

In [Fer 96a] it was proved that if for some language family $\mathcal{L}$ it is known that $PR^\lambda_{ut} \subseteq \mathcal{L} \subseteq PR^\lambda_{ac}$ [$PR_{ut} \subseteq \mathcal{L} \subseteq PR_{ac}$] and $\mathcal{L}$ is closed under intersection with regular languages [and restricted homomorphisms], then $\mathcal{L} = PR^\lambda_{ac}$ [$\mathcal{L} = PR_{ac}$]. Hence Theorem 2 leads to the main result of this section, presented in the next theorem.

Theorem 4

(i) $PT_wETSEG = PR_{ac}$ *and*

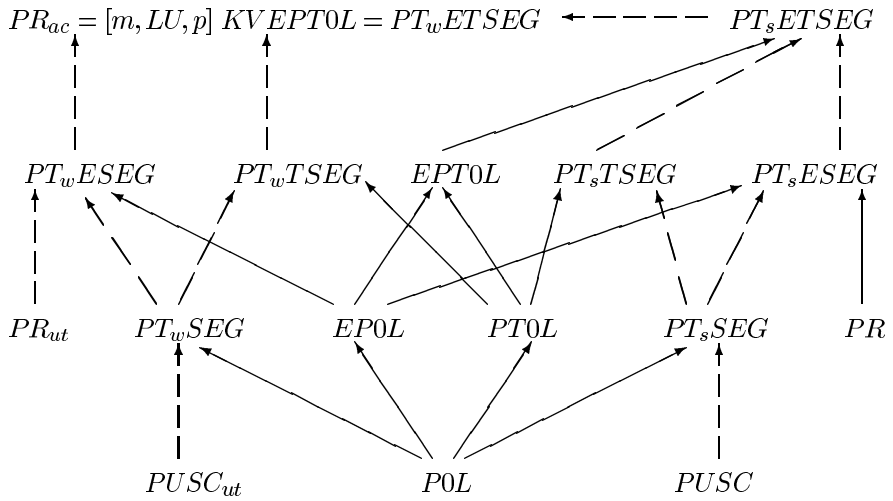
(ii) $PT_wETSEG^\lambda = RE$.

Since RE is a full AFL, Theorem 4 leads to closure under more properties (see, e.g., [25]).

Corollary 3 *The family PT_wETSEG^λ is closed under (i) union, (ii) concatenation, (iii) +-Kleene closure, (iv) arbitrary homomorphisms, (v) inverse homomorphisms, (vi) gsm mappings, (vii) inverse gsm mappings, (viii) left quotient by regular languages, (ix) right quotient by regular languages and (x) regular substitutions.*

6 Summary and open problems

To begin with, a summary of the results presented so far is given in the form of a diagram. In this diagram, a dashed arrow indicates an inclusion which is not known to be proper, whereas a straight arrow indicates a proper inclusion; in both cases the class the arrow leaves is included in the class the arrow points at. Families which are not connected are not necessarily incomparable. Moreover, the diagram continues to hold in the case when there is no restriction to λ-free productions.



Observing this diagram, it is clear that many open problems remain. Is weak rewriting more powerful than strong rewriting, or is the class of extended tabled simple eco-grammar systems with prescribed teams and operating in the weak rewriting step equal to the class of programmed grammars with appearance checking? And does the relation between weak and strong rewriting apply to all subclasses as well or do different relations exist for different subclasses?

It is interesting to note that also for colonies (for a definition of colonies, see, e.g., [8]) and for teams in CD grammar systems ([2]), the relation between weak and strong rewriting is unknown. An answer to those relations would not necessarily solve the case for extended tabled simple eco-grammar systems with prescribed teams, but it might shed light on some intrinsic characteristics of weak versus strong rewriting. However, in the case of colonies no relation between the two ways of rewriting is known yet, whereas in the case of CD grammar systems it was proved in [2] that weak rewriting can be simulated by strong rewriting.

Furthermore, the answer to one of the most interesting open problems in the theory of formal languages lies hidden in this diagram as well: is the inclusion of the family of languages generated by programmed grammars with unconditional transfer in the family of languages generated by those with appearance checking strict? In [26], the class of programmed grammars is claimed to be closed under intersection with regular sets (which would result in a proper inclusion indeed), but the proof is subject to disbelief (see, e.g., [12]).

Finally, does the hierarchy that holds for L systems (see the diagram and note that (see, e.g., [23]) $E[P]0L$ is incomparable with $[P]T0L$) extend to the "L systems with teams" considered here? (This could immediately confirm the above conjecture.) The conjecture for this last question is that a similar hierarchy does indeed hold within the weak, respectively strong, rewriting step, pretty much for the same reasons as why it holds for L systems.

Acknowledgements

This work has benefited from discussions with and comments and suggestions from E. Csuhaj-Varjú and H.C.M. Kleijn. This paper is a slightly shortened version of Part IV: *Teams in eco-grammar systems* of my master's thesis ([1]).

References

[1] M. H. ter Beek, Teams in grammar systems, *IR-96-32 (master's thesis)*, Leiden University, 1996.

[2] M. H. ter Beek, Teams in grammar systems: hybridity and weak rewriting. *Acta Cybernetica* 12, 4 (1996), 427 - 444.

[3] E. Csuhaj-Varjú, J. Dassow, J. Kelemen and Gh. Păun, *Grammar Systems. A Grammatical Approach to Distribution and Cooperation*, Gordon and Breach, London, 1994.

[4] E. Csuhaj-Varjú, J. Kelemen, A. Kelemenová and Gh. Păun, Eco(grammar) systems: a language theoretic model for AL (Artificial Life). Manuscript, 1993.

[5] E. Csuhaj-Varjú, J. Kelemen, A. Kelemenová and Gh. Păun, Eco-grammar systems. A preview. In *Proc. 12th European Meeting on Cybernetics and System Research* (R. Trappl, ed.), World Sci. Publ., Singapore, 1994, 941 - 948.

[6] E. Csuhaj-Varjú, Eco-grammar systems: recent results and perspectives. In *Artificial Life: Grammatical Models* (Gh. Păun, ed.), Black Sea Univ. Press, Bucharest, Romania, 1995, 79 - 103.

[7] E. Csuhaj-Varjú, Generalized eco-grammar systems: a framework for natural language generation. In *Lenguajes Naturales Y Lenguajes Formales XII (C. Martin-Vide, ed.)*, PPU, Barcelona, 1996, 13 - 27.

[8] J. Dassow, J. Kelemen and Gh. Păun, On parallelism in colonies. *Cybernet. Systems* 24 (1993), 37 - 49.

[9] J. Dassow and Gh. Păun, *Regulated Rewriting in Formal Language Theory*, Springer-Verlag, 1989.

[10] A. Ehrenfeucht and G. Rozenberg, On proving that certain languages are not ET0L. *Acta Informatica* 6 (1976), 407 - 415.

[11] H. Fernau, Membership for 1-limited ET0L languages is not decidable. *J. Inform. Process. Cybern. EIK* 30 (1994), 191 - 211.

[12] H. Fernau, On unconditional transfer. *Proceedings of the MFCS'96*, *Lecture Notes in Computer Science* 1113, Springer-Verlag, Berlin, 1996, 348 - 359.

[13] H. Fernau, Scattered context grammars with regulation. *Ann. Univ. Bucureşti, Math-Informatics Series* 45, 1 (1996), 41 - 50.

[14] I. Friš, Grammars with partial ordering of the rules. *Inform. Control* 12 (1968), 412 - 425. Correction in *Inform. Control* 14 (1969), 5.

[15] D. Hauschildt and M. Jantzen, Petri net algorithms in the theory of matrix grammars, *Acta Informatica* 31 (1994), 719 - 728.

[16] L. Kari, A. Mateescu, Gh. Păun and A. Salomaa, Teams in cooperating grammar systems, *J. Exper. Th. AI* 7 (1995), 347 - 359.

[17] Ch. G. Langton, Artificial life. In *Artificial Life I* (Ch. G. Langton, ed.), Addison-Wesley, Redwood City, Cal., 1989.

[18] A. Lindenmayer, Mathematical models for cellular interaction in development. *J. Theoretical Biology* 18 (1968), Part I: 280 - 299, Part II: 300 - 315.

[19] O. Mayer, Some restricted devices for context-free languages. *Inform. Control* 20 (1972), 69 - 92.

[20] Gh. Păun and G. Rozenberg, Prescribed teams of grammars. *Acta Informatica* 31 (1994), 525 - 537.

[21] D. J. Rosenkrantz, Programmed grammars and classes of formal languages. *J. ACM* 16, 1 (1969), 107 - 131.

[22] G. Rozenberg, T0L systems and languages. *Inform. Control* 23 (1973), 357 - 381.

[23] G. Rozenberg and A. Salomaa, *The Mathematical Theory of L Systems*, Academic Press, New York, 1980.

[24] G. Rozenberg and S. H. von Solms, Priorities on context conditions. *Inform. Sci.* 14 (1978), 15 - 50.

[25] A. Salomaa, *Formal Languages*, Academic Press, New York, 1973.

[26] E. D. Stotskii, Control of the conclusion in formal grammars. *Problems of Information Transmission* 7, 3 (1971, translated 1973), 257 - 270.