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.

Abstract

This master's thesis consists of four parts. Part I is an introduction into the field of research of this thesis. The history and motivation of the theory of grammar systems is presented and an informal introduction of the notion of teams is given. In contrast to classical formal language theory, which considers one grammar producing one language, a grammar system is a set of grammars, cooperating in a specified way, still producing one language.

After the introduction, Part II contains the prerequisites necessary for understanding this thesis. These are organized in such a way as to allow the reader to skip sections which would not extend his knowledge. The formal definitions of grammar systems can be found here as well.

In Part III, the forming of teams in grammar systems is studied. A team is formed from a finite number of components (sets of productions) and in every derivation step, one production from each component is used to rewrite a symbol of the sentential form. Hence rewriting is done in parallel. Several derivation modes are considered, varying from using a team exactly one time to using it a maximal amount of times. The possibility of different teams having different modes of derivation is defined as well, as is a weaker restriction on the application of a team. The power of all such mechanisms is investigated exhaustively and in many cases the forming of teams enlarges the generative power of the underlying grammar systems. Finally, some variations with mechanisms controlling the derivations are defined and also their generative power is investigated.

In Part IV, finally, an enhanced grammar system, called an eco-grammar system, is studied. 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. Their application in Artificial Life is shortly discussed, but attention is focused on their language generative capacity. Several different ways of forming teams of agents, with different L systems for the environment, are introduced to mirror the investigations in Part III. Also in this case the forming of teams is found to increase the generative power of the underlying systems.

Preface and acknowledgements

To trace the development of this master's thesis, I have to go back to the second half of 1994, when I was leading a quiet life in Leiden and studying at the local university. In that semester I discovered that my interest in computer science lay in the theoretical field. I had the honour of being taught very enthousiastically by Dr. Jetty Kleijn, Dr. Joost Engelfriet and Professor Grzegorz Rozenberg. I would like to thank them for showing me the beauty of the theory of formal languages as it was then that I decided to continue my studies in that direction.

Because my wife Emese is Hungarian I decided to try to combine pleasure with studying and to apply for a scholarship to study in Budapest for a year. A lot of documents were needed for this and time was short since everything needed to be sent in before the end of 1994. Professor Grzegorz Rozenberg gave me the very interesting research topic in the relatively new field of grammar systems that has resulted in this thesis. By this I would like to thank him for this challenging assignment (I have thoroughly enjoyed it) and Jetty and Emese for all their efforts which made it possible to complete the application in time.

The first half of 1995 was filled with finishing everything except my master's thesis for my graduation at Leiden University and with waiting impatiently for confirmation of the scholarship. When this confirmation from Hungary finally arrived, summer was approaching fast and by that the time to leave for Budapest.

Dr. Erzsébet Csuhaj-Varjú from the Computer and Automation Research Institute of the Hungarian Academy of Sciences had been found willing to supervise my work in Budapest. With the supervision of Professor Grzegorz Rozenberg and Jetty on distance, failure was excluded. Throughout the year Erzsébet has been of inexpressible help to me. Not only technical problems related to my study were solved by her, but also secondary conditions such as arranging me a room in the Academy have never been a problem. I would like to thank her very much for putting up with me all this time. It has been a year I will think back of for the rest of my life.

In this one year in Budapest, I have learned to speak Hungarian, made several valuable friends and frequented concert, opera and theater performances, as well as many other social activities (including our appartment being used as a hotel by our friends). It may be considered a miracle that next to enjoying life in the most beautiful city of Europe, I managed to finish this thesis and satisfactorily complete several courses at the Eötvös Loránd University as well.

Next to the persons mentioned above I would also like to thank Dr. Gheorghe Păun, whose comments and suggestions on an earlier version of this thesis have greatly improved the final version, as all comments from those persons above have.

Budapest, 1 July 1996

Maurice ter Beek

Contents

Part I
Introduction

A new direction in the theory of formal languages is that of *grammar systems*. To begin this introduction with, the history and motivation of grammar systems is explained. This theory of grammar systems has already resulted in the monograph [CDKP 94a], which contains an exhaustive survey of the state of the art in the area until ca. 1992.

When agents are unable to tackle a complex problem, or to perform a certain specific task, due to limited capabilities, it seems natural to try to tackle the problem, or perform the task, by more than one agent. This results in a *multi-agent system*. In the theory of *Artificial Intelligence* (AI), two basically different approaches are known.

Decentralized AI deals with the study of multi-agent systems of *autonomous* agents, i.e. communication between agents is minimalized and there is no central representation of the achievements of agents. Each autonomous agent exists in its own right, performing its own tasks, possibly cooperating with other agents.

Distributed AI, on the other hand, deals with multi-agent systems consisting of a group of agents that cooperate to reach a certain goal, making mutual sharing of information a necessity. These agents are fit into a larger system on the basis of a *strategy* under which *cooperation* is *distributed* in the architecture of the multi-agent system. The theory of grammar systems is evolved around this idea.

So-called *blackboard systems* are an example of the usage of multi-agent systems in AI as a way to achieve a goal which agents are unable to manage by themselves. The multi-agent system tries to reach the goal by distributed and cooperating agents. For details of blackboard systems, the reader is referred to [Nii 86] and [Nii 89].

Within this theory, the *blackboard model of problem solving* is specified along the following lines. The model consists of the following three parts.

- The knowledge sources, needed to solve the given problem.
- The blackboard, representing the current state of the problem solving.
- The strategy, regulating the order in which the knowledge sources work.

The blackboard model of problem solving starts with the given problem specified on the blackboard. The knowledge sources try to contribute to solve the problem by changing the current state of the blackboard. During the problem solving, the only way in which the knowledge sources can communicate with each other is by using the blackboard. Finally, in the case of successful cooperation to achieve the solution, this solution appears on the blackboard.

The link between this blackboard model of problem solving and formal languages was established in [CK 89]. The knowledge sources correspond to grammars, changing the current state of the blackboard corresponds to rewriting the sentential form, the strategy is regulated by so-called derivation modes and the solution is represented by a terminal word.

In [CD 90], *cooperating distributed grammar systems*, CD grammars systems for short, have been introduced as a formal realisation of this link. Sets of productions, called components, are used by picking one of its productions and rewriting the sentential form. However, there's a derivation mode (associated to the system) to be obeyed. A derivation mode specifies how many times this picking occurs, before using another component to continue. It can be precisely (at least, at most) a (positive) natural number, or an arbitrary amount of times, or as long as it can still rewrite the string. The latter one will turn out to be by far the most interesting and powerful.

These systems have been investigated intensively. Moreover, they have initiated the development of a theory which became known as the theory of grammar systems. A formal definition of this and of many other notions introduced in this Introduction can be found in Part II, the Prerequisites.

These CD grammar systems have only one mode of derivation associated to the grammar system as a whole, according to which all components rewrite the intermediate sentential forms. A more realistic approach to cooperation is to consider the cooperating components to have different capabilities. Such heterogeneous grammar systems were introduced in in [Mit 93] and they are called *hybrid CD grammar systems*. In those systems, a mode of derivation is associated not to the whole system, but to every component separately.

For readers interested in even more variants of (sequential) grammar systems, a survey is presented in Appendix A. Moreover, it is necessary to mention that next to the grammar systems that work sequential, also grammar systems that work in parallel have been defined and investigated. This has been done exhaustively enough to state that the theory of grammar systems consists of two main subfields. An informal definition of these *parallel communicating grammar systems*, PC grammars systems for short, can be found in Appendix B, along with references into this presently very popular research field.

Next to the above grammar systems that were motivated by the multi-agent systems originating from the theory of AI, also grammar systems whose introduction was motivated by *Artificial Life* (AL) were introduced. Whereas AI "has focused primarily on the production of intelligent solutions rather than on the production of intelligent behaviour", AL is the study of "man-made systems that exhibit behaviours characteristic of natural living systems" and "is not concerned with building systems that reach some sort of solution" ([Lang 89]).

In AL, a special multi-agent system, called an *ecosystem* is considered quite an important paradigm. Such an ecosystem can be modelled by a special grammar system, called an *eco-grammar system* ([CKKP 93]). An eco-grammar system is based on the following *postulates* which are formulated according to the properties for AL suggested in [Fd'A 91].

(1) An ecosystem consists of an *environment* and a set of *agents*. Both the state of the environment and the states of the agents are described by strings of symbols of given alphabets.

(2) In an ecosystem there is a *universal clock* which marks time units. These are the same for all agents as well as for the environment, according to which the evolution of the agents and of the environment takes place.

(3) Both the environment and the agents have characteristic *evolution rules*, which are rules of L systems and are thus applied in parallel to all the symbols describing the states of the agents and the environment. One such a rewriting step is done *in each time unit*.

(4) The evolution rules of the environment are *independent* of the agents and of the state of the environment itself. The evolution rules of the agents, however, *depend* on the state of the environment.

(5) The agents act on the environment according to *action rules*, which are pure rewriting rules used sequentially. In each time unit, each agent uses one action rule, which is chosen from a set depending on the current state of the agent.

(6) These *actions have priority over the evolution* of the environment. In each time unit, exactly those symbols of the environment which are not affected by the actions are rewritten, in parallel, by evolution rules.

An eco-grammar system integrates some features of both main directions in grammar systems, the CD grammar systems and the PC grammar systems.

One of the many variants listed in Appendix A is investigated in detail in the sequel: the notion of *teams* in the theory of grammar systems. This notion was introduced in [KMPS 95] and all the research in this thesis is concentrated around teams. Before introducing any formal definition, one needs to answer the question what a team is, or rather what it should be.

Everybody intuitively has an idea about what a team is: *a group of elements cooperating together to accomplish a certain task or to reach a certain goal.* Teams can be found in sport, sociology, society, to name but a few. However, the reason why certain elements are part of the team and how a team works to reach a set goal, may vary from team to team.

Thus *How to form teams?* is the first obvious question that needs to be answered. Several ways shall be considered in the sequel, based on this and the following questions. *Are teams formed once and then remain the same throughout the process (static) or can they vary from time to time (dynamic)? What is their size? How do they cooperate together, are different teams allowed to work in different ways (according to different strategies) or not?*

Also in the theory of formal languages, the concept of a team is not a new one. One may consider a scattered context rule as a team of productions. Its associated strategy is to rewrite all left-hand sides in parallel, even keeping the order of the left-hand sides from left to right if the scattered context grammar is not unordered. Also the matrices in matrix grammars can be seen as teams, albeit with totally different strategies. These two language classes are equal *but*, in the context-free case, larger than the underlying grammar without teams. Does the same hold for grammar systems, hence *Does the forming of teams enlarge the generative power?* This is the main question that will be answered in this thesis and in many cases the answer will be yes.

Another important question is what happens with the part of the string that is not affected by the rewriting of a team. *Do the parts of the string not rewritten by a team remain the same or are they rewritten according to another set of (complete) productions?* This establishes the link between grammar systems (for which the first case holds) and eco-grammar systems (for which the latter case holds). Though originally motivated from AL, only eco-grammar systems as generative devices will be considered in the sequel. For readers interested in the usage of eco-grammar systems within the frame of AL, a survey in that direction is presented in Appendix D.

The goal of this thesis is twofold. On the one hand, it provides a survey of the theory of teams in grammar systems, while on the other hand it enriches the theory where no research was done in order to shed light on the different ways of defining teams. Due to the extent of this area, this goal has been limited to the investigations of teams in CD grammar systems and eco-grammar systems. Moreover, the main topic of interest has been to investigate the generative power of teams in grammar systems. This is done extensively, for example the cases of a restriction to other productions than context-free ones are investigated for the first time for teams. Some directions for research not concerning the generative power are suggested in the sequel, though, just as some direct consequences of generative power results are mentioned.

The concept of teams is formally achieved in the theory of grammar systems by treating sets of components as teams. But, as already implied by the questions stated above, this is about the only thing all coming definitions of teams in grammar systems have in common.

How to form teams? Teams can be formed given a certain grammar system or they can be a part of the definition of the grammar system. This distinction is formally achieved by differentiating between *prescribed* and *free* teams. Furthermore, teams can be of *variable size* or a (positive) natural number, say s, can be given such that the number of components of every team is exactly this *constant size* s. In particular, when hybrid CD grammar systems are the basis, teams can be formed by grouping all components by their mode of derivation. This latter method as well as the use of prescribed teams means that the forming of teams takes place *automatically* or is inherent to the grammar system, whereas in the free case there are more possibilities to form teams.

A somewhat surprising result is the fact that whether the teams are of variable size or of constant size, or whether they are free or prescribed, they all lead to the same generative power, at least for the context-free case ([PR 94] and [FP 95]). It remains an open problem, though, whether the automatic forming of teams enhances the power of the underlying hybrid CD grammar systems.

Are teams formed once and then remain the same throughout the process (static) or can they vary from time to time (dynamic)? In this thesis, only the static variant is studied, but it is easy to come up with criteria deciding which teams are formed and doing so in every step. For example, every component that *can* rewrite a symbol of the current sentential form is put together in a team. Dynamic teams were already introduced in [MMS 94] and this topic has a lot of promising possibilities for future research.

What is their size? Teams could be formed with a size restriction or, as pointed out before, according to a constant, but they can also be completely variable. Some other conditions, such as arbitrary but at least two, will be considered briefly in the sequel.

It has already been proved that there are cases when teams of size two suffice ([CP 93]). Adding to such normal-form-like results, it will be shown that also restricting the size of the components leads to interesting consequences. For example, though not for all cases, teams with components consisting of single productions suffice. Much more research concerning restrictions, taking both the size of teams and components in consideration, remains to be done.

How do they cooperate together, are different teams allowed to work in different ways (according to different strategies) or not? Similar modes of derivation as those for CD grammar systems have been introduced in the case of teams as well, with the obvious modifications. For the maximal derivation mode, however, three different variants have been introduced so far and what's more: all of them have been proved to be equal with respect to their generative power ([KMPS 95], [PR 94] and [FP 95]).

Up to now, only one way of cooperation had been introduced in the literature: a rewriting step of a team consists of choosing one production from every component of the team and rewriting all these left-hand sides in parallel, according to the derivation mode associated to the grammar system, a team being unable to work if a component cannot rewrite any symbol of the current sentential form.

Two other ways are introduced in this thesis. The first way allows the mode of derivation to vary from team to team, thus creating a heterogeneous situation similar to that of the hybrid CD grammar systems. The second way allows a form of appearance checking in the teams. Only those components of the team that contain a production with a left-hand side that is present in the current sentential form, *must* be used, others can be ignored.

The so-called *hybrid teams* of the first way are proved to be equivalent to the homogeneous teams if those work in the maximal derivation mode. However, these hybrid teams seem to be able to reduce the syntactic complexity, that is to say there are cases when the amount of variables used can be reduced. A wide open field for future research can be found in this direction of syntactic complexity measures.

The so-called *weak rewriting step* of the second way leads to an equality with the particular case of appearance checking called unconditional transfer. This result has two direct interesting consequences. Firstly, already a prescribed team CD grammar system with only one production per component, operating in this weak rewriting step and one of the derivation modes $= 1$, ≥ 1, $*$ or $\leq k$ (for a $k \geq 1$), can generate more than a CD grammar system operating in mode t can. Secondly, a proper inclusion exists for these systems between the case with and without λ-free productions.

More research in this direction might lead to the solving of the open problem in formal language theory whether appearance checking is more powerful than unconditional transfer.

Do the parts of the string not rewritten by a team remain the same or are they rewritten according to another set of (complete) productions? This question splits the thesis in two. In Part III, teams in CD grammar systems are the subject while in Part IV prescribed teams in eco-grammar systems are introduced and investigated. The forming of teams is introduced in a restricted variant of eco-grammar systems, which was named *simple eco-grammar systems* ([CKKP 93]).

Part III starts with introducing the definitions concerning teams in CD grammar systems, some of which have already been informally discussed above, illustrated with examples. Furthermore, three different possibilities of adding control mechanisms are defined for the general case of prescribed team CD grammar systems with teams of variable size. These are a hypothesis language, a graph and a generalized sequential machine.

Consequently, the generative power of these definitions is investigated. The

results are grouped according to the different cases that arose by answering the above questions. Moreover, it will be shown that the control mechanisms are incapable of enlarging the generative power of the underlying grammar systems with teams. However, an advantage in syntactic complexity is conjectured and thus a possibility for further research remains.

First the case of context-free productions is investigated. In this case (prescribed) teams of constant or variable size, working in either of the three variants of the so-called t-mode, enlarge the power of CD grammar systems to equal that of the family of programmed grammars with appearance checking. For other modes of derivation than the maximal rewriting strategy, still the power of programmed grammars is reached, only limited to the ones without appearance checking.

In the case of CD grammar systems with prescribed teams of variable size operating in the weak rewriting step, the generative capacity is equal to that of the class of programmed grammars with unconditional transfer, for certain derivation modes. This implies that these families and those of the prescribed team CD grammar systems operating in the strong rewriting step and those certain modes of derivation do not coincide.

Secondly, the case of restrictions to regular, linear or metalinear productions is studied. For (hybrid) prescribed team CD grammar systems with teams of constant size and only regular productions, the team forming enlarges their generative power beyond the class of regular languages to the class of regular simple matrix grammars. Hence it extends also beyond the power of regular (hybrid) CD grammar systems. The same holds in the case of a restriction to linear productions.

In the case of teams of variable size, no more than the class of regular or linear languages can be generated by (hybrid) prescribed team CD grammar systems with only regular or linear productions, respectively. However, when restricted to metalinear productions, the generative power of (hybrid) prescribed team CD grammar systems extends beyond the class of metalinear languages. Moreover, prescribed team CD grammar systems with this restriction to metalinear productions are already equal to the class of programmed grammars with the same restriction and appearance checking in the case of the maximal rewriting strategies. For the other modes of derivation, the equalities hold only without appearance checking.

Finally, there are some modes of derivation for which only one production per component suffices in the case of prescribed team CD grammar systems with only regular, linear, metalinear or context-free productions and teams of variable size.

In Part IV simple eco-grammar system with prescribed teams are defined. These systems are nothing more than Lindenmayer systems with teams and therefore four different classes are defined, analogous to those of the Lindenmayer systems (by adding extension and/or tables). Concerning the usage of the

teams, two rewriting steps have been introduced, like in Part III. The difference between weak and *strong rewriting* is investigated, where strong rewriting means using a production from *each* component of the team. Once again, the similarity with appearance checking and unconditional transfer, respectively, is obvious.

The forming of teams will be shown to strictly enlarge the generative power of the underlying L systems in all cases. By the introduction of a normal form for the most complex variant (the extended simple eco-grammar systems with prescribed teams), working in the weak rewriting step, closure under several closure operations is proved. This is shown to lead to an equality with the recursively enumerable language class. This a surprising result since it gives a different view on the relation between appearance checking and unconditional transfer than was expected from Part III of the thesis.

Unfortunately, the power of the strong rewriting variant is not known to be strictly weaker or equal, perhaps the most important open problem in this area. Though a general picture of the generative power is given for all cases, many relations are still only inclusions, not known to be proper or equalities. Infinitely many possibilities thus remain for future research.

Especially interesting for future research is one of the most famous open problems in the theory of formal languages: *Can the conjecture that the inclusion of the family of languages generated by programmed grammars with unconditional transfer in the family of languages generated by programmed grammars with appearance checking is proper be shown to hold?* This thesis gives several new angles into this problem, but does not provide the answer. In Part III as well as in Part IV, open problems concerning the definitions in this thesis are stated which, if answered, would result in the solving of this famous open problem.

Part II
Prerequisites of formal languages

The general aim of this part is to define and explain the prerequisites necessary for this thesis, in a detailed way. If something remains unexplained, the reader is referred to [Sal 73] for formal languages in general, [DP 89] for regulated rewriting and [RS 80] for Lindenmayer systems. Several other references are given in the coming sections as well.

1 Languages and operations

Denote by $V = \{a_1, a_2, \ldots, a_n\}$ an *alphabet*; $a_1, a_2, \ldots, a_n$ are abstract symbols, called *letters*. A finite sequence consisting of zero or more letters of V is called a *word* (or a *string*) over V. The *empty word* (a *string* consisting of zero letters) is denoted by λ. The set of all words over V is denoted by V^*; V^+ denotes the set of all non-empty words. Every subset of V^* is a *language* over V; $L \subseteq V^*$ is called a λ-*free* language if $\lambda \notin L$.

To differentiate properly between sets and multisets, a *multiset* (a set with the possibility of multiple occurrences of the same elements) 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 $\emptyset$. Moreover, a set or multiset *appears* in a string or in a sentential form iff all of its elements are part of the string. For a multiset this means that an element which is listed x times in the multiset must have (at least) x appearances in the string as well.

For two words w_1 and w_2 the *concatenation* (or *product*) of these words is denoted by $w_1 w_2$. The n-fold product (n being a non-negative integer) of a word w is denoted by w^n; w^0 denotes the empty word. The *length* of a word w is denoted by $|w|$. The number of occurrences of a symbol a in a word w is denoted by $|w|_a$. Moreover, for $W \subseteq V$ and $w \in V^*$ as usual

$$|w|_W = \sum_{a \in W} |w|_a.$$

Denote by $alph(w)$ the set of symbols occurring in a word w, then for a language L $alph(L) = \bigcup_{w \in L} alph(w)$. In general, no distinction is made between the element x and the singleton set $\{x\}$.

The concatenation of two languages L_1 and L_2 is denoted by $L_1 L_2 = \{w_1 w_2 \mid w_1 \in L_1, w_2 \in L_2\}$. Furthermore, denote language $L^0 = \{\lambda\}$, $L^{i+1} = L^i L$ (for $i \geq 0$) and the $*$-*Kleene closure* ($+$-*Kleene closure*) by $L^* = \cup_{i \geq 0} L^i$ ($L^+ = \cup_{i \geq 1} L^i$). A language is called *regular* if it can be obtained from $V \cup \{\lambda\}$ by using a finite number of applications of the operations union, concatenation and $*$-Kleene closure.

Consider a mapping $s : V \to 2^{U^*}$, i.e. $s(a) \subseteq U^*$ for all $a \in V$, for V and U two (possibly equal) alphabets and 2^X the power set of the set X. Extend this mapping to $s : V^* \to 2^{U^*}$ by

$$s(\lambda) = \lambda \text{ and } s(w_1 w_2) = s(w_1)s(w_2), \text{ for } w_1, w_2 \in V^*.$$

For a language $L \subseteq V^*$, define

$$s(L) = \bigcup_{w \in L} s(w).$$

This s is called a *substitution* from V into U. A substitution s is called λ-*free* if $\lambda \notin s(a)$ for all $a \in V$. A *homomorphism*, sometimes also simply called a *morphism*, is a substitution s such that $s(a)$ consists of a single word for all $a \in V$. For a homomorphism $h : V^* \to U^*$, the mapping $h^{-1} : U^* \to 2^{V^*}$ defined by

$$h^{-1}(w) = \{v \in V^* \mid h(v) = w\}$$

is called an *inverse homomorphism*. 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.

A family of languages $\mathcal{L}$ is said to be closed under an n-ary operation τ if $\tau(L_1, L_2, \ldots, L_n) \in \mathcal{L}$ for all languages $L_1, L_2, \ldots, L_n \in \mathcal{L}$. A language family is called an *abstract family of languages*, denoted by AFL, if it is closed under union, concatenation, +-Kleene closure, λ-free homomorphisms, inverse homomorphisms and intersections with regular languages. A family of languages closed under all these operations except concatenation and +-Kleene closure is called a *semi*-AFL. Furthermore, an AFL or a semi-AFL is called *full* if it is closed under arbitrary homomorphisms.

The *left quotient* of a language L_1 by a language L_2 is defined by

$$L_2 \setminus L_1 = \{x \mid yx \in L_1,\ y \in L_2\},$$

similarly the *right quotient* of L_1 by L_2 is defined by

$$L_1/L_2 = \{x \mid xy \in L_1,\ y \in L_2\}.$$

The special cases of quotient by a singleton language, $L_2 = \{x\}$, are denoted by $d_x^l(L_1)$ and $d_x^r(L_1)$ and are called the *left derivative* and the *right derivative*, respectively, of a language L_1 with respect to a string x. Every semi-AFL is closed under left and right derivatives, while every full semi-AFL is closed under left and right quotient by regular languages.

Finally, an important convention made in this thesis is the fact that **whenever in a proof a new symbol is introduced, it is considered to be distinct from all previously introduced symbols**.

2 Grammars and automata

A *Chomsky grammar* of *type-i* ($i \in \{0,1,2,3\}$) is a construct $G = (N, T, P, S)$, where N and T are two disjoint alphabets, called the set of *nonterminals* and the set of *terminals*, respectively, $S \in N$ is the *axiom* and P is a finite set of *productions* of the form i ($i \in \{0,1,2,3\}$) as below.

(0) $\alpha \to \beta$, for $\alpha, \beta \in (N \cup T)^*$ and $|\alpha|_N \geq 1$,

(1) $w_1 A w_2 \to w_1 x w_2$, for $w_1, w_2 \in (N \cup T)^*$, $x \in (N \cup T)^+$ and $A \in N$. $S \to \lambda$ is allowed iff S does not appear in the right-hand side of any production,

(2) $A \to x$, for $A \in N$ and $x \in (N \cup T)^*$ and

(3) $A \to xB$ or $A \to x$, for $A, B \in N$ and $x \in T^*$.

Moreover, two other types of grammars will be used in the sequel, defined with a production set of the form j ($j \in \{\text{iv}, \text{v}\}$) as follows.

(iv) $A \to x_1 B x_2$ or $A \to x$, for $A, B \in N$ and $x, x_1, x_2 \in T^*$ and

(v) $A \to x_1 B x_2$ or $A \to x$, for $A, B \in N$ and $x, x_1, x_2 \in T^*$. Productions of the form $S \to x_1 A_1 x_2 A_2 \ldots x_n A_n x_{n+1}$ are allowed for $A_i \in N$, $x_i \in T^*$ and $1 \leq i \leq n$ iff S does not appear in the right-hand side of any production.

A string x *directly derives* a string y in G, denoted by $x \Longrightarrow_G y$, iff $x = w_1 \alpha w_2$, $y = w_1 \beta w_2$ and $\alpha \to \beta \in P$ for $w_1, w_2 \in (N \cup T)^*$. This is also called a one-step derivation in G; consequently a k-step derivation (for $k \geq 0$) in G, denoted by $\Longrightarrow_G^k$, is defined for $x, y \in (N \cup T)^*$ as follows: $x \Longrightarrow_G^k y$ iff there are words $x_0, x_1, \ldots, x_k$ such that $x = x_0$, $y = x_k$ and $x_i \Longrightarrow_G x_{i+1}$, $0 \leq i \leq k-1$. If G is clear from the context, it is omitted, thus writing only $\Longrightarrow$ and $\Longrightarrow^k$, respectively. This applies to all definitions of derivation steps in the sequel as well. The transitive (and reflexive) closure of the one-step derivation is denoted by $\Longrightarrow^+$ ($\Longrightarrow^*$).

The language generated by G is denoted by $L(G)$ and it is defined by

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

A word $w \in (N \cup T)^*$ is called a *sentential form* (*terminal word* if $w \in T^*$) of G iff $S \Longrightarrow_G^* w$. Hence the language generated by G consists of all terminal words of G.

A language is said to be of type-i iff it is generated by a Chomsky grammar of type-i ($i \in \{0,1,2,3\}$). Type-0 grammars are said to be *phrase structure grammars* and type-0 languages are called *recursively enumerable*; their family of languages is denoted by RE. Type-1 grammars and languages are also called

context-sensitive and their family of languages is denoted by CS. Type-2 grammars and languages are also called *context-free* and their family of languages is denoted by CF. In the literature, type-3 grammars and languages are called *right-linear*. When in condition (3) above the requirement $x \in T^*$ is replaced by $x \in T$, the definition of a *simply regular* grammar is obtained. Right-linear and simply regular grammars generate the same family of languages. In this thesis, grammars in the sense of condition (3) above will be considered and are called *regular*; their family of languages is denoted by REG. Grammars and languages of type (iv) are called *linear* and their family of languages is denoted by LIN. Grammars and languages of type (v) are called *metalinear* and their family of languages is denoted by $MLIN$. Furthermore, the family of all finite languages is denoted by FIN, i.e. $FIN = \{L \mid L \text{ is a finite subset of } V^* \text{ for some alphabet } V\}$. Finally, a grammar is called λ-*free* if it does not contain any production $\alpha \to \lambda$, or if the only λ-production it does contain is $S \to \lambda$, S being the axiom and not appearing in the right-hand side of any production of the particular grammar. The following proper hierarchy is well-known.

$$FIN \subset REG \subset LIN \subset MLIN \subset CF \subset CS \subset RE.$$

Notation 1 *In the sequel, the notation for language generating devices will be defined for λ-free productions of type-2 only. When those language generating devices are used with a restriction other than type-2, a subscript REG, LIN, $MLIN$, CS or RE is added; when λ-productions are allowed, the superscript λ is added.*

A *finite automaton* is a construct $\mathcal{A} = (K, V, s_0, f, H)$, where K is a finite non-empty 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.

A *generalized sequential machine* (gsm) is a construct $g = (K, I, O, s_0, \delta, H)$, where K is a finite non-empty set of *states*, I is the non-empty *input* alphabet, O is the non-empty *output* alphabet, $s_0 \in K$ is the *initial state*, $H \subseteq K$ is the set of *final states* and δ is a finite set of productions of the form

$$s_i v \to w s_j \text{ for } s_i, s_j \in K,\ v \in I \text{ and } w \in O^*.$$

Often the productions are written as tuples of the format $(s_i v, w s_j)$. For a gsm g and a word $v \in I^+$, denote

$$g(v) = \{w \mid s_0 v \Rightarrow^* w s_z \text{ for some } s_z \in H\},$$

where $\Rightarrow^*$ denotes the reflexive and transitive closure of $\rightarrow$ and for a language L over I

$$g(L) = \{z \mid z \in g(v) \text{ for some } v \in L\}.$$

(This mapping is called a *gsm mapping*.)

A gsm is called *deterministic* iff, for all $s_i \in K$ and $v \in I$, there exists at most one production of the above format in δ. Moreover, whenever all words w in those productions consist of one letter of O and the gsm is deterministic, it is called a *Mealy machine*. Every AFL is closed under λ-free gsm mappings, while every full AFL is closed under gsm mappings.

3 Regulated rewriting

Many ways of enhancing Chomsky grammars by adding sophisticated mechanisms to regulate the derivations have been introduced in the literature. Several of these will be used in the sequel and they are defined next.

A *matrix grammar with appearance checking* is a construct $G = (N, T, S, M, F)$, where N is the set of nonterminals, T is the set of terminals and $S \in N$ is the axiom, as in the case of Chomsky grammars, M is a finite set of *matrices* of the form $m : (r_1, r_2, \ldots, r_n)$, where $r_i : \alpha_i \rightarrow \beta_i$ are productions over $N \cup T$ and $|\alpha|_N \geq 1$, $1 \leq i \leq n$ and F, finally, is a set of occurrences of productions in M. A matrix grammar with appearance checking works as follows. For $w, w' \in (N \cup T)^*$ and $m : (\alpha_1 \rightarrow \beta_1, \alpha_2 \rightarrow \beta_2, \ldots, \alpha_n \rightarrow \beta_n) \in M$ it is said that w directly derives w', written as

$$\begin{array}{lll} w \Longrightarrow w' & \text{iff} & \text{there exist } w_0, w_1, \ldots, w_n \in (N \cup T)^* \text{ such that} \\ & & w_0 = w \text{ and } w_n = w' \text{ and for all } 0 \leq i \leq n-1 \\ & \text{either} & w_{i-1} = w'_{i-1}\alpha_i w''_{i-1} \text{ and } w_i = w'_{i-1}\beta_i w''_{i-1} \\ & & \text{for some } w'_{i-1}, w''_{i-1} \in (N \cup T)^* \\ & \text{or} & \text{the production } \alpha_i \rightarrow \beta_i \text{ cannot be applied to } w_{i-1}, \\ & & \alpha_i \rightarrow \beta_i \in F \text{ and } w_i = w_{i-1}. \end{array}$$

At any derivation step, the sentential form is rewritten by a matrix by using one by one all of its productions in the order in which they appear in that matrix. However, whenever a production cannot be applied and it is contained in F, it may be passed over and the next production can be considered. This is the only exception, in any other case the matrix cannot be applied. If $F = \emptyset$, the matrix grammar is called a *matrix grammar without appearance checking* and F is omitted from the construct. Moreover, if F contains all occurrences of productions in M, the matrix 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 matrix grammars with only λ-free context-free productions in M is denoted by MAT_{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.

In the following, a special case of matrix grammars is considered. A *simple matrix grammar* of *degree* n, $n \geq 1$, is a construct $G = (N_1, N_2, \ldots, N_n, T, S, M)$, where $N_1, N_2, \ldots, N_n$ (sets of nonterminals) and T (the set of terminals) are pairwise disjoint alphabets, $S \notin (\bigcup_{i=1}^{n} N_i \cup T)$ is the start symbol and M is a finite set of matrices, each of one of the following forms.

(a) $(S \to x)$, for $x \in T^*$,

(b) $(S \to A_1 A_2 \ldots A_n)$, for $A_i \in N_i$ and $1 \leq i \leq n$ or

(c) $(A_1 \to x_1, A_2 \to x_2, \ldots, A_n \to x_n)$, for $A_i \in N_i$, $x_i \in (N_i \cup T)^*$ and $|x_i|_{N_i} = |x_j|_{N_j}$ for all $1 \leq i, j \leq n$.

A simple matrix grammar of degree n works as follows.
For $w, w' \in (\bigcup_{i=1}^{n} N_i \cup T \cup \{S\})^*$ it is said that w directly derives w', written as

$$\begin{aligned} w \Longrightarrow w' \quad \text{iff} \quad & w = S \text{ and } (S \to w') \in M \\ \text{or} \quad & w = v_1 A_1 w_1 v_2 A_2 w_2 \ldots v_n A_n w_n,\ w' = v_1 x_1 w_1 v_2 x_2 w_2 \ldots v_n x_n w_n, \\ & A_i \in N_i,\ v_i \in T^*,\ w_i, x_i \in (N_i \cup T)^*,\ 1 \leq i \leq n \text{ and} \\ & (A_1 \to x_1, A_2 \to x_2, \ldots, A_n \to x_n) \in M. \end{aligned}$$

The language generated by G is

$$L(G) = \{w \in T^* \mid S \Longrightarrow^* w\},$$

where $\Longrightarrow^*$ denotes the reflexive and transitive closure of $\Longrightarrow$.

A simple matrix grammar is called regular, right-linear, linear, context-free or λ-free iff the productions appearing in matrices of type (c) in M are all regular, right-linear, linear, context-free or λ-free, respectively. The family of languages generated by λ-free context-free simple matrix grammars of degree n, $n \geq 1$, is denoted by $SM(n)$. Regular simple matrix grammars are also called equal matrix grammars. Furthermore, denote $SM = \bigcup_{n \geq 1} SM(n)$.

A *programmed grammar* 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 as in the case of Chomsky grammars 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 $\varphi(r) \subseteq Lab(P)$ is called 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

$$\begin{aligned}(w, r_1) \Longrightarrow (w', r_2) \quad \text{iff} \quad & w = w_1 \alpha w_2,\ w' = w_1 \beta w_2 \text{ and } r_2 \in \sigma(r_1) \\ \text{or} \quad & w = w',\ \alpha \to \beta \text{ cannot be applied to } w \\ & \text{and } r_2 \in \varphi(r_1).\end{aligned}$$

At any derivation step, if a production can rewrite the current sentential form it has to be applied and the next production to be executed has to be in the success field of this production; if it cannot be applied, a production from its failure field must be used next. If the failure fields are empty for all productions, 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

$$\begin{aligned}L(G) = \{w \in T^* \mid (S, r_0) \Longrightarrow (w_1, r_1) \Longrightarrow \cdots \Longrightarrow (w_s, r_s) = (w, r_s), \\ s \geq 1,\ w_i \in (N \cup T)^*,\ r_i \in Lab(P),\ 0 \leq i \leq s\}.\end{aligned}$$

For technical reasons, it is assumed in the sequel that the success fields can never be empty. In the original definition of programmed grammars in [Ros 69] the success fields could be empty; the derivation coming to a halt in case the next production should be selected from the empty set. This does not restrict any results in the sequel, since it can be proved that for every programmed grammar with empty success fields an equivalent one without empty success fields can be constructed.

The family of languages generated by programmed grammars with only λ-free context-free productions in P is denoted by 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.

A *scattered context grammar with appearance checking* is a construct $G = (N, T, S, P, F)$, where N is the set of nonterminals, T is the set of terminals and $S \in N$ is the axiom, as in the case of Chomsky grammars, $P = \{p_1, p_2, \ldots, p_n\}$ 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, $1 \leq i \leq n$. A 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

$$
\begin{aligned}
w \Longrightarrow w' \quad \text{iff} \quad & w = w_1\alpha_{i_1}w_2\alpha_{i_2}\ldots w_m\alpha_{i_m}w_{m+1},\ w' = w_1\beta_{i_1}w_2\beta_{i_2}\ldots w_m\beta_{i_m}w_{m+1}, \\
& p_i : (\alpha_1, \alpha_2, \ldots, \alpha_p) \to (\beta_1, \beta_2, \ldots, \beta_p) \in P,\ (\alpha_{i_1}, \alpha_{i_2}, \ldots, \alpha_{i_m}) \text{ is} \\
& \text{a subsequence of } (\alpha_1, \alpha_2, \ldots, \alpha_p),\ w_l \in (N \cup T)^* \\
& \text{and } 1 \leq l \leq m+1 \\
\text{and} \quad & \alpha_j \text{ in } \{\alpha_1, \alpha_2, \ldots, \alpha_p\} \text{ and not in } \{\alpha_{i_1}, \alpha_{i_2}, \ldots, \alpha_{i_m}\} \text{ implies} \\
& \text{that } \alpha_j \text{ does not appear in the sentential form between } \alpha_{j-1} \text{ and} \\
& \alpha_{j+1} \text{ and } \alpha_j \to \beta_j \in F.
\end{aligned}
$$

At any derivation step, a sentential form is rewritten by a rule p_i, $1 \leq i \leq n$, by using all the productions listed in it in parallel. Moreover, the symbols to be rewritten have to appear in the same order in the sentential form as their counterparts in the applied rule do. However, iff a production is contained in F and its left-hand side does not occur in the sentential form to be rewritten between the appearances of the symbols to its immediate left and right in the applied rule, it may be passed over. This is the only exception, in any other case the rule cannot be applied. If $F = \emptyset$, the scattered context grammar is called a *scattered context grammar without appearance checking* and F is omitted from the construct. 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 scattered context grammars with only λ-free context-free productions in P is denoted by SC_{ac} in the case of grammars with appearance checking; when grammars without appearance checking are considered the subscript ac is omitted.

An *unordered scattered context grammar with appearance checking* 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 is the finite set of finite sequences of productions and F is a set of occurrences of productions in P, as in the case of scattered context grammars with appearance checking. 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

$$
\begin{aligned}
w \Longrightarrow w' \quad \text{iff} \quad & w = w_1\alpha_{i_1}w_2\alpha_{i_2}\ldots w_m\alpha_{i_m}w_{m+1},\ w' = w_1\beta_{i_1}w_2\beta_{i_2}\ldots w_m\beta_{i_m}w_{m+1}, \\
& p_i : (\alpha_1, \alpha_2, \ldots, \alpha_p) \to (\beta_1, \beta_2, \ldots, \beta_p) \in P,\ (\alpha_{i_1}, \alpha_{i_2}, \ldots, \alpha_{i_m}) \text{ is} \\
& \text{a permutation of a subsequence of } (\alpha_1, \alpha_2, \ldots, \alpha_p), \\
& w_l \in (N \cup T)^* \text{ and } 1 \leq l \leq m+1 \\
\text{and} \quad & \alpha_j \text{ in } \{\alpha_1, \alpha_2, \ldots, \alpha_p\} \text{ and not in } \{\alpha_{i_1}, \alpha_{i_2}, \ldots, \alpha_{i_m}\} \text{ implies that} \\
& \alpha_j \text{ is not contained in } w \text{ and } \alpha_j \to \beta_j \in F.
\end{aligned}
$$

An unordered scattered context grammar with appearance checking thus works in the same way as a scattered context grammar with appearance checking

but without the ordering of the symbols to be replaced. The appearance checking now results in skipping those productions of the rule to be applied which are in F and whose left-hand sides do not occur at all in the parts of the sentential form that remain after choosing the symbols to be rewritten. Obviously, if $F = \emptyset$ 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 only λ-free context-free productions in P is denoted by 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.

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

It is known (see e.g. [DP 89], [GW 89], [HD 89], [HJ 94] and [Fer 95]) that (the family $EPT0L$ will be defined shortly)

$$CF \subset USC = PR = MAT \subset SC \subseteq CS \subset SC^\lambda = RE,$$
$$CF \subset USC^\lambda = PR^\lambda = MAT^\lambda \subset SC^\lambda = SC^\lambda_{ac} = RE,$$
$$CF \subset EPT0L \subset USC_{ut} = PR_{ut} = MAT_{ut} \subseteq USC_{ac} = PR_{ac} = MAT_{ac} \subset CS,$$
$$CF \subset EPT0L \subset USC_{ut} = PR_{ut} = MAT_{ut} \subset USC^\lambda_{ut} = PR^\lambda_{ut} = MAT^\lambda_{ut} \subseteq RE,$$
$$CF \subset USC = PR = MAT \subset USC^\lambda = PR^\lambda = MAT^\lambda \subset$$
$$USC^\lambda_{ac} = PR^\lambda_{ac} = MAT^\lambda_{ac} = SC^\lambda_{ac} = RE \text{ and}$$
$$CF \subset USC = PR = MAT \subset USC_{ac} = PR_{ac} = MAT_{ac} \subset$$
$$SC_{ac} = CS \subset USC^\lambda_{ac} = PR^\lambda_{ac} = MAT^\lambda_{ac} = SC^\lambda_{ac} = RE.$$

4 Lindenmayer systems

A *0L system* is a construct $G = (\Sigma, \sigma, \omega)$, where Σ is an alphabet, σ is a finite substitution from Σ into Σ^* and $\omega \in \Sigma^+$ is the axiom. The language generated by G is

$$L(G) = \{\sigma^i(\omega) \mid i \geq 0\}.$$

When σ is a morphism, the 0L system is called a deterministic 0L system or a *D0L system* since this means rewriting is deterministic. When σ is λ-free, the 0L system (D0L system) is called propagating or a *P0L system* (*PD0L system*).

An *E0L system* is a construct $G = (\Sigma, \Delta, \sigma, \omega)$, where $\Delta \subseteq \Sigma$ is called the *terminal alphabet* of G and $U(G) = (\Sigma, \sigma, \omega)$ is a 0L system, called the *underlying system* of G. The language generated by G is

$$L(G) = L(U(G)) \cap \Delta^*.$$

Like in the case of 0L systems, an *ED0L system*, an *EP0L system* and an *EPD0L system* are defined in the obvious way.

A *T0L system* is a construct $G = (\Sigma, S, \omega)$, where S is a finite set of finite substitutions, called *tables*, such that for each such a substitution $\sigma \in S$, (Σ, σ, ω) is a 0L system. The language generated by G is

$$L(G) = \{z \in \Sigma^* \mid z = \omega \vee z = \sigma_1\sigma_2 \cdots \sigma_r(\omega),\ \sigma_1, \sigma_2, \ldots, \sigma_r \in S\}.$$

When the number of tables in a T0L system equals one, a 0L system is obtained. Hence a 0L system can be considered a special case of a T0L system. Furthermore, a *DT0L system*, an *ET0L system*, a *PT0L system*, an *EDT0L system*, a *PDT0L system*, an *EPT0L system* and an *EPDT0L system* are defined in the obvious way.

In Lindenmayer systems rewriting thus takes place in parallel and in each step every symbol of the sentential form is rewritten, which differentiates these L systems from all grammars introduced before. Quite often in the literature and always in this thesis, substitutions in L systems are specified by productions. Then, for instance,

$$a \to \lambda,\ a \to a^3,\ a \to a^6$$

is equivalent to

$$\sigma(a) = \{\lambda, a^3, a^6\}.$$

The family of languages generated by 0L, P0L, E0L, EP0L, T0L, PT0L, ET0L and EPT0L systems is denoted by $0L$, $P0L$, $E0L$, $EP0L$, $T0L$, $PT0L$, $ET0L$ and $EPT0L$, respectively.

It is known (see e.g. [Sal 73], [RS 80] and [Sal 95]) that

$$P0L \subseteq 0L \subset T0L \subset PR_{ac},$$
$$CF \subset EP0L \subseteq E0L \subset EPT0L \subseteq ET0L \subset PR_{ac} \subset CS,$$
$$PD0L \subset D0L \subset ED0L,$$
$$PD0L \subset EPD0L \subset ED0L,$$
$$DT0L \subset EDT0L \subset ET0L,$$
$$PDT0L \subset DT0L \subset T0L \subset ET0L,$$
$$DT0L \subset EDT0L,$$
$$PDT0L \subset EPDT0L \text{ and}$$
$$PDT0L \subset PT0L \subset T0L.$$

In [RvS 78], ET0L systems were augmented with context conditions and priorities. One of these systems will be used in this thesis and is defined next. An *ET0L system with local undirected multiset context conditions and priorities (an [m,LU,p] KVET0L system)* 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 (for every symbol of the alphabet at least one production exists in each table) 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 $\Longrightarrow$.

An [m,LU,p] KVET0L system works only slightly different than an ET0L system. Rewriting still takes place in parallel, but when applying a table a production can only be applied when all symbols in its multiset appear somewhere in the sentential form and there is no production with the same left-hand side and a higher priority whose context condition is also satisfied.

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 the case of systems with local undirected multiset context conditions and priorities; when local undirected set context conditions and priorities are considered, the m in the construct is replaced by s. When all priorities are equal, the p in either construct is replaced by n. In [RvS 78] it was proved that all these variations result in an equality between the class of KVET0L (KVEPT0L) systems and the class of programmed grammars with appearance checking (and λ-free productions).

5 Grammar systems

The notions of this section are used intensively throughout this thesis. Therefore, they will be explained more detailed and illustrated with examples. For any

unexplained notions, as well as for anything which is unclear, the reader is referred to [CDKP 94a] and other references mentioned in the corresponding subsections.

5.1 Cooperating distributed grammar systems

Cooperating grammar systems were introduced already in [MR 78] and [MRV 78], the motivation coming from the theory of two-level grammars. About ten years later, cooperating grammar systems received more attention when the link between them and the notion of the blackboard model of problem solving from the theory of AI was established in [CK 89], as described in the Introduction. A more general form was introduced in [CD 90] and is presented next.

Definition 1 *A* cooperating distributed grammar system, *CD grammar system for short, is a construct* $\Gamma = (N, T, S, P_1, P_2, \ldots, P_n)$, *where* N *is the set of nonterminals,* T *is the set of terminals and* $S \in N$ *is the axiom as in the case of Chomsky grammars and each* P_i, $1 \leq i \leq n$, *is a finite set of productions over* $N \cup T$, *called a* component *of* Γ.

For the use in some proofs below, the *domain* of a (non-type-0 component) component P_i is defined by $dom(P_i) = \{A \in N \mid A \to x \in P_i\}$, $1 \leq i \leq n$.

Rewriting in CD grammar systems is quite different from the hitherto introduced ways.

Definition 2 *Consider a CD grammar system* $\Gamma = (N, T, S, P_1, P_2, \ldots, P_n)$. *Now rewriting in* Γ *can take place according to one of the following five* modes of derivation. *For all modes, consider* $x, y, z \in (N \cup T)^*$, $k \geq 1$, $1 \leq i \leq n$ *and* $\Longrightarrow_{P_i}^{l}$ *is used for the* l*-step derivation* $\Longrightarrow^{l}$ *as defined for a Chomsky grammar* (N, T, S, P_i).

($\leq k$*) This mode corresponds to* at most k *direct derivation steps in succession by some component* P_i *in the CD grammar system:*

$$x \Longrightarrow_{\Gamma}^{\leq k} y \text{ iff } \exists P_i \text{ such that } x \Longrightarrow_{P_i}^{0} y \text{ or } x \Longrightarrow_{P_i}^{k'} y \text{ for some } k' \leq k.$$

($= k$*) This mode corresponds to* exactly k *direct derivation steps in succession by some component* P_i *in the CD grammar system:*

$$x \Longrightarrow_{\Gamma}^{=k} y \text{ iff } \exists P_i \text{ such that } x \Longrightarrow_{P_i}^{k} y.$$

($\geq k$*) This mode corresponds to* at least k *direct derivation steps in succession by some component* P_i *in the CD grammar system:*

$$x \Longrightarrow_{\Gamma}^{\geq k} y \text{ iff } \exists P_i \text{ such that } x \Longrightarrow_{P_i}^{k'} y \text{ for some } k' \geq k.$$

(∗) This mode corresponds to an arbitrary number *of direct derivation steps in succession by some component P_i in the CD grammar system:*

$$x \Longrightarrow_\Gamma^* y \text{ iff } \exists P_i \text{ such that } x \Longrightarrow_{P_i}^0 y \text{ or } x \Longrightarrow_{P_i}^k y \text{ for some } k.$$

(t) This mode corresponds to maximal *derivations by some component P_i in the CD grammar system (the component must rewrite the sentential form as long as it is able to):*

$$x \Longrightarrow_\Gamma^t y \text{ iff } \exists P_i \text{ such that } x \Longrightarrow_{P_i}^* y \text{ and there is no } z \text{ such that } y \Longrightarrow_{P_i} z.$$

The language generated by a CD grammar system depends on the mode of derivation according to which it rewrites.

Definition 3 *Consider a CD grammar system $\Gamma = (N, T, S, P_1, P_2, \ldots, P_n)$. The language generated by Γ in derivation mode f, for $f \in \{*, t\} \cup \{\leq k, = k, \geq k \mid k \geq 1\}$, is denoted by*

$$L_f(\Gamma) = \{z \in T^* \mid S \Longrightarrow_\Gamma^f w_1 \Longrightarrow_\Gamma^f \cdots \Longrightarrow_\Gamma^f w_m = z,\ m \geq 1\}.$$

The family of languages generated by CD grammar systems with only λ-free context-free productions in each P_j, at most n components and working in mode f, is denoted by $CD_n(f)$, $1 \leq j \leq n$. Furthermore, denote $CD(f) = \bigcup_{n \geq 1} CD_n(f)$.

This definition is illustrated by the following example.

Example 1 *Consider the CD grammar system*

$$\Gamma_1 = (\{S, A, B, A', B'\}, \{a, b, c\}, S, P_1, P_2, P_3),$$

where

$$\begin{aligned} P_1 &= \{S \to AB, A' \to A, B' \to B\}, \\ P_2 &= \{A \to a^4 A' b^4, B \to c^4 B'\} \text{ and} \\ P_3 &= \{A \to ab, B \to c\}. \end{aligned}$$

Suppose that this CD grammar system works in the maximal derivation mode t. It is clear that the first component to be applied is P_1, resulting in AB. Then either component P_3 is used, resulting in the terminal word abc, or component P_2 is used. After using component P_2, the sentential form is $a^4 A' b^4 c^4 B'$ and only P_1 can be used, resulting in $a^4 A b^4 c^4 B$. Now, using P_2 and P_1 iteratively, strings of the form $a^i A b^i c^i B$ are generated, where $i \bmod 4 \equiv 0$. This string can be rewritten into a terminal one by using P_3 and it is clear that the language generated by Γ_1 operating in mode t is

$$L_t(\Gamma_1) = \{a^n b^n c^n \mid n \bmod 4 \equiv 1, n \geq 1\}.$$

This language is not a context-free language, demonstrating that CD grammar systems with context-free components are able to produce languages not in the context-free language class.

In fact, concerning the generative power of CD grammar systems, it is known from [CD 90] and [CDKP 94a] that for $f \in \{=k, \geq k \mid k \geq 2\}$, $k \geq 1$ and $k', k'' \geq 2$

$CF = CD(=1) = CD(\geq 1) = CD(*) = CD(\leq k) \subset (CD(=k') \cap CD(\geq k''))$,
$CF = CD_1(f) \subset CD_2(f) \subseteq CD_3(f) \subseteq \ldots \subseteq CD(f) \subseteq MAT$ and
$CF = CD_1(t) = CD_2(t) \subset CD_3(t) = CD(t) = ET0L$.

More results on the generative power of CD grammar systems can be found in [CDKP 94a].

5.1.1 Hybrid CD grammar systems

The idea to consider the agents of a multi-agent system to have different capabilities was introduced into the model of CD grammar systems in [Mit 93]. It is formally modelled by allowing different modes of derivation to be associated with different components.

Definition 4 *A* hybrid cooperating distributed (CD) grammar system *is a construct* $\Gamma = (N, T, S, (P_1, f_1), (P_2, f_2), \ldots, (P_n, f_n))$, *where* N *is the set of nonterminals,* T *is the set of terminals,* $S \in N$ *is the axiom and* $P_1, P_2, \ldots, P_n$ *are the components as in the case of usual CD grammar systems and* $f_i \in \{*, t\} \cup \{\leq k, =k, \geq k \mid k \geq 1\}$ *is the mode of derivation associated with the component* P_i, $1 \leq i \leq n$.
The language generated by Γ *is*

$$L(\Gamma) = \{z \in T^* \mid S \Longrightarrow_{P_{i_1}}^{f_{i_1}} w_{i_1} \Longrightarrow_{P_{i_2}}^{f_{i_2}} \cdots \Longrightarrow_{P_{i_m}}^{f_{i_m}} w_{i_m} = z, \quad 1 \leq i_j \leq n,\ 1 \leq j \leq m\}.$$

The family of languages generated by hybrid CD grammar systems with only λ-free context-free productions in each P_j and at most n components, is denoted by HCD_n, $1 \leq j \leq n$. Furthermore, denote $HCD = \bigcup_{n \geq 1} HCD_n$.

Also this definition is illustrated by an example.

Example 2 *Consider the hybrid CD grammar system*

$$\Gamma_2 = (\{S, A, B, C, D, A', B', C', D'\}, \{a, b, c, d\}, S, (P_1, =1), (P_2, =2), (P_3, =2), (P_4, t), (P_5, =4)),$$

where

$$\begin{aligned} P_1 &= \{S \to ABCD\}, \\ P_2 &= \{A \to aA', C \to cC'\}, \\ P_3 &= \{B \to bB', D \to dD'\}, \\ P_4 &= \{A' \to A, B' \to B, C' \to C, D' \to D\} \text{ and} \\ P_5 &= \{A \to a, B \to b, C \to c, D \to d\}. \end{aligned}$$

Obviously, every successful derivation starts with the application of P_1, thus generating $ABCD$. There are three possibilities to continue. When P_5 is applied, the terminal word $abcd$ is generated since P_5 operates in mode $=4$. Otherwise, either P_2 or P_3 has to be used in mode $=2$, resulting in $aA'BcC'D$ or $AbB'CdD'$, respectively. Then, in case P_2 (P_3) was used in the last step, the derivation can proceed with P_3 (P_2) or P_4. Using P_2 as well as P_3, the only possibility to continue is by using P_4. Hence, in all cases (basically only two different ones) there comes a time when P_4 is used. This P_4 works in mode t, which makes it possible to remove the primes from only A and C or from only B and D or from all four of them, in all cases leaving no primed nonterminals and thus fulfil the stop condition of mode t. Consequently, this process can be iterated until finally P_5 is used to replace A, B, C and D and thus fulfilling the stop condition for mode $=4$. It is clear from the above explanations that the generated language is

$$L(\Gamma_2) = \{a^n b^m c^n d^m \mid m, n \geq 1\}.$$

Like that of Example 1, this language is non-context-free.

Concerning the generative power of hybrid CD grammar systems, it is known from the definitions, [Mit 93] and [Păun 94] that for $f \in \{*, t\} \cup \{\leq k, = k, \geq k \mid k \geq 1\}$, $n' \geq 1$ and $n \geq 4$

$CD_{n'}(f) \subseteq HCD_{n'}$,
$CD(f) \subseteq HCD_3$,
$CF = CD_1(f) = HCD_1 \subset HCD_2 \subseteq HCD_3 \subseteq HCD_4 = HCD_n = HCD \subseteq MAT_{ac}$,
$ET0L = CD_3(t) \subseteq HCD_3$ and
$ET0L \subset HCD_4$.

Furthermore, in [Mit 93] it has been proved that for each hybrid CD grammar system Γ, an equivalent hybrid CD grammar system Γ' can be constructed, containing three components working in the t-mode and one in the $=k$-mode, for some $k \geq 1$.

5.2 Eco-grammar systems

The concept of an eco-grammar system was introduced in [CKKP 93], the first published paper on the topic is [CKKP 94a] and the definitions given next originate from [CKKP 94b].

Definition 5 *An* eco-grammar system *(of degree n, $n \geq 1$) is a construct*

$$\Sigma = (E, A_1, A_2, \ldots, A_n),$$

where

- $E = (V_E, P_E)$,
  - V_E *is a finite alphabet and*
  - P_E *is a finite set of 0L rewriting rules over* V_E *and*
- $A_i = (V_i, P_i, R_i, \varphi_i, \psi_i)$ *for* $1 \leq i \leq n$, *where*
  - V_i *is a finite alphabet,*
  - P_i *is a finite set of 0L rewriting rules over* V_i*,*
  - R_i *is a finite set of productions of the form* $\alpha \to \beta$ *for* $\alpha \in V_E^+$ *and* $\beta \in V_E^*$*,*
  - φ_i *is a computable function from* V_E^* *to* 2^{P_i} *and*
  - ψ_i *is a computable function from* V_i^+ *to* 2^{R_i}*.*

In this construct, E represents the environment with alphabet V_E and set of evolution rules P_E. $A_i = (V_i, P_i, R_i, \varphi_i, \psi_i)$ corresponds to the i-th agent, $1 \leq i \leq n$, with alphabet V_i, set of evolution rules P_i and set of action rules R_i. Mapping φ_i, which depends on the state of the environment, selects the evolution rules for A_i and mapping ψ_i, which depends on the state of the agent, selects the rules for the action. Note that mapping ψ_i is not defined for λ, whereas φ_i is.

The current state of the environment as well as that of the agents is represented by strings, which thus together indicate the current *state* of the eco-grammar system.

Definition 6 *A* state *of an eco-grammar system* $\Sigma = (E, A_1, A_2, \ldots, A_n)$ *is an* $n+1$*-tuple*

$$\sigma = (w_E, w_1, w_2, \ldots, w_n), \text{ where } w_E \in V_E^*, \ w_i \in V_i^* \text{ and } 1 \leq i \leq n.$$

(w_E is called the state of the environment and w_i is called the state of the i-th agent, $1 \leq i \leq n$.)

An eco-grammar system changes its states by rewriting the strings currently representing the environment and the agents.

Definition 7 *Let $\sigma = (w_E, w_1, w_2, \ldots, w_n)$ be a state of the eco-grammar system $\Sigma = (E, A_1, A_2, \ldots, A_n)$. Then*

- *agent A_i is defined to be* active *in state σ if the set of its action rules $\psi_i(w_i)$ is non-empty,*

- *an* action *of an active agent A_i in state σ is defined as an application of an action rule $r \in \psi_i(w_i)$ to the environmental state w_E,*

- *a* simultaneous action *of active agents $A_{i_1}, \ldots, A_{i_k}$, $\{i_1, \ldots, i_k\} \subseteq \{1, \ldots, n\}$, in state σ onto the environment is defined as a parallel derivation step*

$$w_E \Longrightarrow w'_E \quad \textit{such that} \quad \begin{array}{l} w_E = x_1\alpha_1x_2\alpha_2\ldots\alpha_kx_{k+1},\ w'_E = x_1\beta_1x_2\beta_2\ldots\beta_kx_{k+1}, \\ \alpha_j \to \beta_j \in \psi_{i_j}(w_{i_j}),\ 1 \leq j \leq k,\ x_i \in V_E^* \textit{ and} \\ 1 \leq i \leq k+1, \end{array}$$

- *w'_E is defined to be an* evolution of the environment *w_E if w'_E can be derived from w_E by 0L rewriting rules of P_E and*

- *w'_i is defined to be an* evolution of agent *A_i in state w_i if w'_i can be derived from w_i by 0L rewriting rules of $\varphi_i(w_E)$, $1 \leq i \leq n$.*

These notions explain the subtle difference in the definition of the two functions φ and ψ as pointed out before. Agents can perform an action only if they are in a non-empty state, but they can adapt to any state of the environment, including the empty one.

A change of the state of an eco-grammar system is achieved by an evolution of the state of every agent as well as an evolution of the environment at each place, except those ones where the currently active agents perform a simultaneous action.

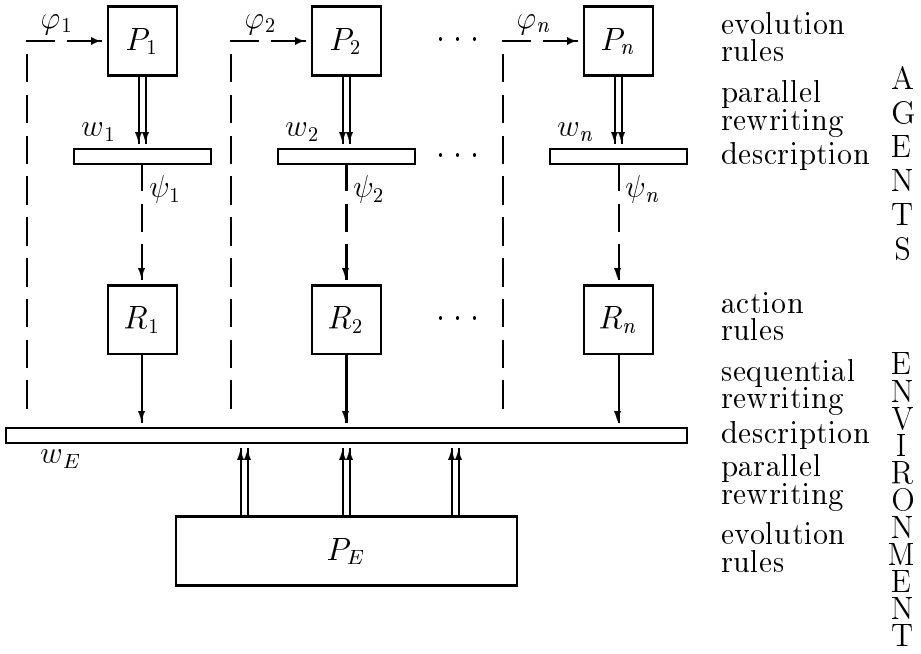
Definition 8 *A state $\sigma = (w_E, w_1, w_2, \ldots, w_n)$, of an eco-grammar system $\Sigma = (E, A_1, A_2, \ldots, A_n)$, directly derives a state $\sigma' = (w'_E, w'_1, w'_2, \ldots, w'_n)$, written as*

$$\sigma \Longrightarrow_\Sigma \sigma' \quad \textit{iff} \quad \begin{array}{l} w_E = x_1\alpha_1x_2\alpha_2\ldots x_k\alpha_kx_{k+1},\ w'_E = x'_1\beta_1x'_2\beta_2\ldots x'_k\beta_kx'_{k+1} \\ \textit{such that} \quad x_1\alpha_1x_2\alpha_2\ldots\alpha_kx_{k+1} \Longrightarrow x_1\beta_1x_2\beta_2\ldots\beta_kx_{k+1} \textit{ is a} \\ \textit{simultaneous action of all agents } A_{i_1}, \ldots A_{i_k}, \\ \{i_1, \ldots, i_k\} \subseteq \{1, \ldots, n\}, \textit{ that are active in state } \sigma \\ \text{and} \\ x'_1x'_2\ldots x'_{k+1} \textit{ is an evolution of } x_1x_2\ldots x_{k+1} \\ \textit{and} \quad w'_i \textit{ is an evolution of } A_i \textit{ in state } w_i, 1 \leq i \leq n. \end{array}$$

The transitive (and reflexive) closure of $\Longrightarrow_\Sigma$ is denoted by $\Longrightarrow_\Sigma^+$ ($\Longrightarrow_\Sigma^*$). If Σ is clear from the context, it is omitted, thus writing only $\Longrightarrow^+$ and $\Longrightarrow^*$, respectively.

Hence the next state of an agent is determined only by its own evolution rules, while the next state of the environment depends on both its own evolution rules and the currently active agents. When an agent enters the empty state λ, it remains non-active during the further functioning of the eco-grammar system (the mapping ψ_i is not defined for λ). On the other hand, an agent can be temporarily non-active as well, that is no action rule of the agent is selected for an action though it might be in the future.

Because of the complexity of eco-grammar systems, a pictorial representation of an eco-grammar system is provided next in order to make them easier understandable.



From the point of view of formal languages, eco-grammar systems can be seen as generative devices generating languages starting from a certain initial state.

Definition 9 *Consider an eco-grammar system $\Sigma = (E, A_1, A_2, \ldots, A_n)$ with an initial state σ_0. The* language of the environment of Σ *is defined by*

$$L_E(\Sigma, \sigma_0) = \{w_E \in V_E^* \mid \sigma_j = (w_E, w_1, \ldots, w_n),\ \sigma_0 \Longrightarrow_\Sigma^* \sigma_j,\ j \geq 0\}.$$

(Analogously, sometimes the language of the i-th agent is defined, but it shall not be used here.)

The family of languages of the environment generated by eco-grammar systems of degree n is denoted by EG_n, $n \geq 1$.

In Appendix C, some variations of eco-grammar systems as generative devices can be found. This section will continue with presenting one more such a variant in more detail since most of the continuation of this thesis will consider that restriction.

An important direction in the development of the theory of eco-grammar system is the study of restricted, i.e. simplified, systems. For convenience, the initial state is made inherent to the system in this definition.

Definition 10 *A* simple eco-grammar system *(of degree n, $n \geq 1$) is an $n+1$-tuple*

$$\Sigma = (E, A_1, A_2, \ldots, A_n),$$

where

- $E = (V_E, P_E, \omega)$,
  - V_E *is a finite alphabet,*
  - P_E *is a finite set of 0L rewriting rules over* V_E *and*
  - $\omega \in V_E^+$ *is the initial state of the environment and*
- $A_i = (V_i, P_i, R_i, \varphi_i, \psi_i)$ *for* $1 \leq i \leq n$, *where*
  - V_i *is a finite alphabet,*
  - P_i *is a finite set of P0L rewriting rules over* V_i,
  - R_i *is a finite set of productions of the form* $\alpha \to \beta$ *for* $\alpha \in V_E$ *and* $\beta \in V_E^*$,
  - $\varphi_i(w_E) = P_i$ *for every* $w_E \in V_E^*$ *and*
  - $\psi_i(w) = R_i$ *for every* $w \in V_i^+$.

In this construct, E is the environment with alphabet V_E, set of evolution rules P_E and initial state ω and $A_i = (V_i, P_i, R_i, \varphi_i, \psi_i)$ is the i-th agent, $1 \leq i \leq n$, with alphabet V_i, set of λ-free evolution rules P_i (i.e. no agent can disappear during evolution) and set of context-free action rules R_i (i.e. actions are independent of the environment). Mapping φ_i and mapping ψ_i are total (i.e. A_i is with full activity and non-adaptive).

Naturally, with the obvious modifications, Definitions 6, 7, 8 and 9 continue to hold for simple eco-grammar systems as well.

The family of languages of the environment generated by simple eco-grammar systems of degree n is denoted by SEG_n, $n \geq 1$. Moreover, by definition, $SEG_0 = 0L$.

An example is given to illustrate the way a simple eco-grammar system works.

Example 3 *Consider the simple eco-grammar system*

$$\Sigma_1 = (E, A_1),$$

where

- $E = (\{a\}, \{a \to a\}, a)$ *and*
- $A_1 = (\{b\}, \{b \to b\}, \{a \to a^5\}, \varphi_1, \psi_1)$, *where*
  $\varphi_1(w) = \{b \to b\}$, *for every* $w \in \{a\}^*$, *and* $\psi_1(u) = R_1$, *for every* $u \in \{b\}^+$.

In this example only the language of the environment is considered. The axiom is rewritten by the only action rule of the agent, resulting in a^5. *Then one of these five a's is again rewritten by the same production, while the other four a's are replaced by a, thus resulting in* a^9. *This process can be iterated and it is clear that the generated language of the environment is*

$$L_E(\Sigma_1) = \{a^{4i+1} \mid i \geq 0\}.$$

Hence, already a very simple simple eco-grammar system can generate an infinite non-0L language.

Some results from [CKKP 93] and [CKKP 94b] are presented in the following lemma.

Lemma 1 *For* $n \geq 0$

$$FIN \subset SEG_1 \text{ and}$$
$$SEG_n \subset SEG_{n+1}.$$

Part III
Teams in CD grammar systems

1 Definitions and examples

Cooperating distributed grammar systems consist of several grammars working together. It is a natural idea to increase the degree of cooperation by forming so-called *teams* of components and to investigate the effect on the generative power. This idea leads to the so-called *team CD grammar systems*, introduced in [KMPS 95].

Team CD grammar systems in some sense hold the middle between CD grammar systems and PC grammar systems, though the latter allows some communication between the grammars. In CD grammar systems at each moment in time one component is active, while the others are waiting; in PC grammar systems all components are simultaneously active and in team CD grammar systems several components are simultaneously active, leaving the others waiting.

The next section will start with a more general definition of *prescribed team CD grammar systems* from [PR 94], treating the original definition of [KMPS 95] as a special case.

Then, in Section 1.2, team cooperation is added to the hybrid CD grammar systems of [Mit 93]. These systems are heterogeneous, hence different components can work in different ways.

Finally, in Section 1.3 some mechanisms are added to the various grammar systems enabling them to have controlled derivations.

1.1 (Prescribed) teams of grammars

The definition of teams in CD grammar systems given next is a reformulation of the definition in [PR 94], which in turn is a more general definition than the one from [KMPS 95].

Definition 11 *A* prescribed team CD grammar system *(of variable size) is a construct*

$$\Gamma = (N, T, S, P_1, P_2, \ldots, P_n, Q_1, Q_2, \ldots, Q_m),$$

where $(N, T, S, P_1, P_2, \ldots, P_n)$ *is a CD grammar system and* $Q_i \subseteq \{P_1, P_2, \ldots, P_n\}$ *is called a prescribed team,* $1 \leq i \leq m$. *Such a team* $Q_i = \{P_{i_1}, P_{i_2}, \ldots, P_{i_{s_i}}\}$, $1 \leq s_i \leq n$ *and* $1 \leq i \leq m$, *is used in a one-step derivation, as follows. It is said that* x *derives* y *in one step, for* $x, y \in (N \cup T)^*$, *by using team* Q_i, *written as*

$$x \Longrightarrow_{Q_i} y \quad \textit{iff} \quad \begin{array}{l} x = x_1 A_1 x_2 A_2 \ldots x_{s_i} A_{s_i} x_{s_i+1},\ y = x_1 y_1 x_2 y_2 \ldots x_{s_i} y_{s_i} x_{s_i+1}, \\ x_l \in (N \cup T)^*,\ 1 \leq l \leq s_i + 1,\ A_k \to y_k \in P_{i_k} \textit{ and} \\ 1 \leq k \leq s_i. \end{array}$$

At any derivation step, from each component of the team being used one of its productions is used in parallel with all the other members of the team. As a team is a set, no order of the components is assumed and hence the rewritten symbols A_1 to A_{s_i} can appear in x in any order.

Definition 12 *Consider a prescribed team CD grammar system* $\Gamma = (N, T, S, P_1, P_2, \ldots, P_n, Q_1, Q_2, \ldots, Q_m)$. *Then the following modes of derivation can be defined for* Γ. *For* $x, y, z \in (N \cup T)^*$, $k \geq 1$ *and* $1 \leq i \leq m$,

$$
\begin{aligned}
x \Longrightarrow_{Q_i}^{\leq k} y \quad & \textit{iff} \quad x \Longrightarrow_{Q_i}^{0} y \textit{ or } x \Longrightarrow_{Q_i}^{k'} y \textit{ for some } k' \leq k, \\
x \Longrightarrow_{Q_i}^{=k} y \quad & \textit{iff} \quad x \Longrightarrow_{Q_i}^{k} y, \\
x \Longrightarrow_{Q_i}^{\geq k} y \quad & \textit{iff} \quad x \Longrightarrow_{Q_i}^{k'} y \textit{ for some } k' \geq k, \\
x \Longrightarrow_{Q_i}^{*} y \quad & \textit{iff} \quad x \Longrightarrow_{Q_i}^{0} y \textit{ or } x \Longrightarrow_{Q_i}^{k} y \textit{ for some } k, \\
x \Longrightarrow_{Q_i}^{t_0} y \quad & \textit{iff} \quad x \Longrightarrow_{Q_i}^{*} y \textit{ and there is no } z \textit{ such that } y \Longrightarrow_{Q_i} z, \\
x \Longrightarrow_{Q_i}^{t_1} y \quad & \textit{iff} \quad x \Longrightarrow_{Q_i}^{*} y \textit{ and for no component } P_{i_j} \in Q_i \textit{ and} \\
& \qquad \textit{no } z \textit{ there is a derivation } y \Longrightarrow_{P_{i_j}} z \textit{ and} \\
x \Longrightarrow_{Q_i}^{t_2} y \quad & \textit{iff} \quad x \Longrightarrow_{Q_i}^{*} y \textit{ and there is a component } P_{i_j} \in Q_i \\
& \qquad \textit{for which there is no derivation } y \Longrightarrow_{P_{i_j}} z.
\end{aligned}
$$

The three variants of the t-mode of derivation first appeared in [FP 95] (t_0), [KMPS 95] (t_1) and [PR 94] (t_2); the other modes of derivation are the natural extension of the modes in CD grammar systems to teams of grammars.

When a team is applied to a sentential form, the rewriting of this team in any of the modes of derivation can only come to an end when the stop condition (as defined above) of this mode of derivation is satisfied. Then another team can start its work. In case this stop condition is not (yet) satisfied the team must continue rewriting until it is satisfied. In the case when it is not satisfied and can never be satisfied, the derivation comes to a halt.

Hence, in the case of mode t_0 the work of a team ends successfully when *no further derivation step can be done as a team*, in the case of mode t_1 the work ends when *no component of the team can apply one of its productions any longer* and in mode t_2, finally, the work of a team ends when *there is at least one component that can no longer apply one of its productions.* Shortly, an example will illustrate these subtle differences.

Definition 13 *Consider a prescribed team CD grammar system* $\Gamma = (N, T, S, P_1, P_2, \ldots, P_n, Q_1, Q_2, \ldots, Q_m)$. *The language generated by* Γ, *operating in mode*

$f \in \{*, t_0, t_1, t_2\} \cup \{\leq k, =k, \geq k \mid k \geq 1\}$, *is then*

$$L_f(\Gamma) = \{z \in T^* \mid S \Longrightarrow^f_{Q_{i_1}} w_{i_1} \Longrightarrow^f_{Q_{i_2}} \cdots \Longrightarrow^f_{Q_{i_p}} w_{i_p} = z, \quad 1 \leq i_j \leq m, 1 \leq j \leq p\}.$$

The subtleties between the modes t_j for $j \in \{0, 1, 2\}$ are illustrated by the following example.

Example 4 *Consider the following prescribed team CD grammar system (with teams of variable size)*

$$\Gamma_3 = (\{A, B, C, A', B', C'\}, \{a, b, c\}, S, P_1, \ldots, P_7, \{P_1\}, \{P_2, P_3, P_4\}, \{P_5, P_6, P_7\}),$$

where

$$\begin{aligned} P_1 &= \{S \to ABC\}, \\ P_2 &= \{A \to aA', A \to a, A' \to aa\}, \\ P_3 &= \{B \to bB', B \to b\}, \\ P_4 &= \{C \to cC', C \to c\}, \\ P_5 &= \{A' \to A, A' \to A'\}, \\ P_6 &= \{B' \to B, B \to B\} \text{ and} \\ P_7 &= \{C' \to C, B \to B\}. \end{aligned}$$

In the beginning of the derivation, there is no difference between mode t_0, t_1 or t_2. All derivations start by applying the first team, which results in the string ABC. Then indeed the stop condition for all three modes is satisfied and another team can be used. It is clear that, because of the size of the other teams, if there are any nonterminals in a sentential form, then there need to be three to be able to continue the derivation. Hence, sentential forms which do not satisfy this condition shall not be mentioned in the continuation of this example. From ABC, only the second team can be used and in either mode this leads to $aA'bB'cC'$ or abc or to a string with one or two nonterminals. The reader can check that the stop condition for all three teams is indeed fulfilled.

Next the modes t_0, t_1 and t_2 will be considered separately. First, consider mode t_0. Applying the third team to $aA'bB'cC'$ results in $aAbBcC$ or $aA'bBcC$. In both cases the third team can no longer rewrite as a team. The stop condition for mode t_0 is thus satisfied and another team can be applied. The sentential form $aA'bBcC$ can only lead to the terminal string $a^3b^2c^2$, by applying the second team. However, applying the second team to $aAbBcC$ results in $a^2b^2c^2$ or $a^2A'b^2B'c^2C'$, in both cases satisfying the stop condition. This process can be iterated and it is clear that the generated language becomes

$$L_{t_0}(\Gamma_3) = \{a^n b^n c^n,\ a^{m+1} b^m c^m \mid n \geq 1,\ m \geq 2\}.$$

Secondly, consider mode t_1. Applying the third team to $aA'bB'cC'$ results in $aAbBcC$ or $aA'bBcC$. The sixth and seventh component both contain a production that could still be applied to the sentential form and thus, in both cases, the stop condition for mode t_1 is not satisfied. Nor can the third team be used any longer. It is thus clear that the generated language is

$$L_{t_1}(\Gamma_3) = \{abc\}.$$

Finally, consider mode t_2. Applying the third team to $aA'bB'cC'$ results in $aAbBcC$ or $aA'bBcC$. In the former case, the fifth component cannot rewrite the sentential form and the stop condition for mode t_2 is thus satisfied. In the latter case, every component of the third team can rewrite a symbol from the sentential form (though not all at the same time) and the stop condition for mode t_2 is thus not satisfied. No rewriting step can take place in this case either and the only way to continue is by rewriting $aAbBcC$. In a similar fashion as in the case of mode t_0 this process is iterated and the language becomes

$$L_{t_2}(\Gamma_3) = \{a^n b^n c^n \mid n \geq 1\}.$$

Having studied Γ_3 in such depth, it comes as no surprise that for modes $f \in \{=1, \geq 1, *\} \cup \{\leq k \mid k \geq 1\}$ the use of primes results in the language

$$L_f(\Gamma_3) = \{a^n b^n c^n,\ a^{m+1} b^m c^m \mid n \geq 1,\ m \geq 2\}.$$

Note that this example above "proves" that inclusions such as $L_{t_1}(\Gamma) \subseteq L_{t_0}(\Gamma)$ and $L_{t_2}(\Gamma) \subseteq L_{t_0}(\Gamma)$ can be proper.

The original definition of teams in CD grammar systems in [KMPS 95] can be considered as a special case of this definition. In [KMPS 95], teams are not prescribed, but each set of components can be a team, so-called *free* teams. Moreover, they are not of variable size, but a natural number $s \geq 1$ is given and the teams are formed such that the number of components of every team is exactly s; these teams are called of constant size s.

Combining these two definitions, it is thus clear that we can differentiate between the following four variants of team CD grammar systems, all being special cases of Definition 11 above.

Free teams of variable size: each subset of components can be a team.

Free teams of constant size: this is the original definition of [KMPS 95], as explained above.

Prescribed teams of variable size: these are defined in Definition 11.

Prescribed teams of constant size: all prescribed teams consist of the same number of components.

In the case of teams of constant size, whether prescribed or free, a finite set of axioms $W \subseteq (N \cup T)^*$, with only one string in it containing nonterminals, is allowed. This is done since otherwise in the case of λ-free productions no string shorter than s could be generated. In the case of free teams with teams of constant size, the construct thus becomes $\Gamma = (N, T, W, P_1, P_2, \ldots, P_n)$ and the generated language becomes $L_f(\Gamma, s)$. The modifications in the other cases are obvious.

The family of languages generated by CD grammar systems with prescribed teams of variable size and only λ-free context-free productions is denoted by PT_*CD.

Notation 2 *In the sequel, when teams of variable size are considered the subscript $*$ will be used in the notation; when teams of constant size are considered the subscript s will be used instead of $*$ and when the team-size is not constant but at least two, the subscript $+$ will be used. Moreover, the letter P is omitted from the notation when free teams are considered.*

To show the differences between the definition of prescribed teams and the original definition of free teams in [KMPS 95], also an example with free teams of constant size is presented.

Example 5 *Consider the team CD grammar system (with free teams of constant size)*

$$\Gamma_4 = (\{A, B\}, \{a, b, c\}, AB, P_1, P_2, P_3, P_4),$$

where

$$\begin{aligned} P_1 &= \{A \to a^4 A' b^4, A \to ab\}, \\ P_2 &= \{B \to c^4 B', B \to c\}, \\ P_3 &= \{A' \to A\} \text{ and} \\ P_4 &= \{B' \to B\}. \end{aligned}$$

It is left to the reader to check that the language generated by this team CD grammar system, with teams of size 1, is

$$L_f(\Gamma_4, 1) = \{a^m b^m c^n \mid m \bmod 4 \equiv 1,\ n \bmod 4 \equiv 1,\ m, n \geq 1\}$$

for $f \in \{=1, \geq 1, *, t_0, t_1, t_2\} \cup \{\leq k \mid k \geq 1\}$.

Next, the case of teams of a size larger than 1 is considered in more detail. Since at any moment in time a team cannot work when there is only one nonterminal in the sentential form, only the following two teams result in a team CD grammar system with constant size capable of generating a non-empty language.

$$\begin{aligned} Q_1 &= \{P_1, P_2\} \text{ and} \\ Q_2 &= \{P_3, P_4\}. \end{aligned}$$

These teams both have size 2 hence the grammar contains two 2-teams.

One can start from the axiom AB which, by using Q_1, leads to the terminal string abc or to $a^4A'b^4c^4B'$ or a sentential form with only one nonterminal appearing in it. In the latter case, no terminal word can be obtained since a team of size 2 requires at least two nonterminals to be present in the sentential form. From $a^4A'b^4c^4B'$, using Q_2 followed by Q_1, the terminal string $a^5b^5c^5$ or $a^8A'b^8c^8B'$ results. Once again, other sentential forms are incapable of ever resulting in a terminal word. Consequently, it is easy to see that interchanging Q_1 and Q_2 repeatedly generates $a^nAb^nc^nB$ for $n \bmod 4 \equiv 1$, which can be made terminal by using Q_1, hence yielding

$$L_f(\Gamma_4, 2) = \{a^nb^nc^n \mid n \bmod 4 \equiv 1, n \geq 1\}$$

for $f \in \{=1, \geq 1, *, t_0, t_1, t_2\} \cup \{\leq k \mid k \geq 1\}$.

Note that the following team CD grammar system (with free teams of size 2)

$$\Gamma_5 = (\{A, B\}, \{a, b, c\}, AB, P_1', P_2', P_3', P_4'),$$

where

$$\begin{aligned} P_1' &= \{A \to a^4Ab^4\}, \\ P_2' &= \{B \to c^4B\}, \\ P_3' &= \{A \to ab\} \text{ and} \\ P_4' &= \{B \to c\} \end{aligned}$$

can result in the teams

$$\begin{aligned} Q_1' &= \{P_1', P_2'\} \text{ and} \\ Q_2' &= \{P_3', P_4'\} \end{aligned}$$

being formed.

However, the generated language is

$$L_f(\Gamma_5, 2) = \{a^nb^nc^n \mid n \bmod 4 \equiv 1, n \geq 1\}$$

for $f \in \{=1, \geq 1, *\} \cup \{\leq k \mid k \geq 1\}$ only. This is due to the fact that the maximal rewriting strategy would mean that team Q_1', once started, could never fulfil its stop condition.

A final example according to the more general definition of prescribed teams of variable size is presented to make the notion of teams absolutely understood and to demonstrate the differences between free teams and prescribed teams.

Example 6 *The prescribed team CD grammar system (with teams of variable size)*

$$\Gamma_6 = (\{S, A, B\}, \{a, b, c\}, S, P_1, P_2, P_3, P_4, P_5, \{P_1\}, \{P_2, P_3\}, \{P_4, P_5\}),$$

where

$$\begin{aligned} P_1 &= \{S \to AB\}, \\ P_2 &= \{A \to a^4 A' b^4, A \to ab\}, \\ P_3 &= \{B \to c^4 B', B \to c\}, \\ P_4 &= \{A' \to A\} \text{ and} \\ P_5 &= \{B' \to B\} \end{aligned}$$

obviously yields the same language as Γ_4 *for teams of size* 2 *in Example 5.*

Thus,

$$L_f(\Gamma_6) = \{a^n b^n c^n \mid n \bmod 4 \equiv 1, n \geq 1\}$$

for $f \in \{=1, \geq 1, *, t_0, t_1, t_2\} \cup \{\leq k \mid k \geq 1\}$.

Note the duality between free teams and a string axiom. In Example 5 the team-size was the constant 2, which was possible by taking a string as axiom. In Γ_6 in this last example prescribed teams of variable size are needed to allow the production rewriting the axiom as a separate team of size 1.

1.1.1 The weak rewriting step

In [DKP 93], two different versions of a parallel derivation step in colonies (for a definition of colonies, see Appendix A) were introduced. The difference can occur only when two components can rewrite the same nonterminal; colonies excluding this possibility are called *non-competitive*. In a *strongly competitive* derivation step, every enabled component has to be used in a single derivation step. Thus, if two components have to rewrite the same nonterminal and this nonterminal appears only once in the sentential form, then the derivation is blocked. Hence if a component *can* be used, then it *must* be used.

In a *weakly competitive* derivation step, the derivation is not blocked in such a situation, but a maximum number of enabled components is used. Thus allowing a derivation to continue even if only one of two (or more) components that are capable of rewriting a specific nonterminal (that appears only once in the sentential form) can be used.

A rewriting step of a prescribed team CD grammar system as described in Definition 11 will be considered a strong rewriting step, since it resembles the strongly competitive derivation step in a colony. The idea of a weakly competitive derivation step can be introduced for prescribed team CD grammar systems as well.

Definition 14 *A weak rewriting step of a team Q_i in a prescribed team CD grammar system $\Gamma = (N, T, S, P_1, P_2, \ldots, P_n, Q_1, Q_2, \ldots, Q_m)$ is defined as follows. For $x, y \in (N \cup T)^*$, it is said that x directly derives y in the weak rewriting step by using team Q_i, written as*

$$\begin{array}{lll} x \overset{w}{\Longrightarrow}_{Q_i} y & \textit{iff} & x = x_1A_1x_2A_2 \ldots x_pA_px_{p+1},\ y = x_1y_1x_2y_2 \ldots x_py_px_{p+1} \\ & \textit{such that} & x_l \in (N \cup T)^*,\ 1 \leq l \leq p+1,\ A_k \to y_k \in P_{i_k} \textit{ for} \\ & & 1 \leq k \leq p,\ i_m \neq i_n \textit{ for } m \neq n,\ 1 \leq m, n \leq p \textit{ and} \\ & & \{P_{i_1}, P_{i_2}, \ldots, P_{i_p}\} \subseteq \{P_{i_1}, P_{i_2}, \ldots, P_{i_s}\} = Q_i \textit{ such that} \\ & & \textit{for all } P_{i_q} \in Q_i \setminus \{P_{i_1}, P_{i_2}, \ldots, P_{i_p}\} \textit{ there exists} \\ & & \textit{no production } \alpha \to \beta \in P_{i_q} \textit{ such that } \alpha \in x_1x_2 \ldots x_{p+1}. \end{array}$$

Next to the one-step derivation, the following modes of derivation are defined for Γ. For $x, y, z \in (N \cup T)^*$, $k \geq 1$ and $1 \leq i \leq m$,

$$\begin{array}{lll} x \overset{w}{\Longrightarrow}^{\leq k}_{Q_i} y & \textit{iff} & x \overset{w}{\Longrightarrow}^{0}_{Q_i} y \textit{ or } x \overset{w}{\Longrightarrow}^{k'}_{Q_i} y \textit{ for some } k' \leq k, \\ x \overset{w}{\Longrightarrow}^{=k}_{Q_i} y & \textit{iff} & x \overset{w}{\Longrightarrow}^{k}_{Q_i} y, \\ x \overset{w}{\Longrightarrow}^{\geq k}_{Q_i} y & \textit{iff} & x \overset{w}{\Longrightarrow}^{k'}_{Q_i} y \textit{ for some } k' \geq k, \\ x \overset{w}{\Longrightarrow}^{*}_{Q_i} y & \textit{iff} & x \overset{w}{\Longrightarrow}^{0}_{Q_i} y \textit{ or } x \overset{w}{\Longrightarrow}^{k}_{Q_i} y \textit{ for some } k \textit{ and} \\ x \overset{w}{\Longrightarrow}^{t_0}_{Q_i} y & \textit{iff} & x \overset{w}{\Longrightarrow}^{*}_{Q_i} y \textit{ and there is no } z \textit{ such that } y \overset{w}{\Longrightarrow}_{Q_i} z. \end{array}$$

The language generated by Γ, operating in mode $f \in \{*, t_0\} \cup \{\leq k, = k, \geq k \mid k \geq 1\}$, is

$$L^w_f(\Gamma) = \{z \in T^* \mid S \overset{w}{\Longrightarrow}^{f}_{Q_{i_1}} w_{i_1} \overset{w}{\Longrightarrow}^{f}_{Q_{i_2}} \cdots \overset{w}{\Longrightarrow}^{f}_{Q_{i_p}} w_{i_p} = z, \quad 1 \leq i_j \leq m, 1 \leq j \leq p\}.$$

This weak rewriting step will be considered only for CD grammar systems with prescribed teams of variable size. The family of languages generated by such systems, operating in derivation mode f and rewriting according to the weak rewriting step, is notated by $PT_wCD(f)$ in the case of λ-free context-free productions; when λ-productions are allowed the superscript λ is added.

To illustrate the differences between strong and weak rewriting in prescribed team CD grammar systems, the following example is presented.

Example 7 *Consider the prescribed team CD grammar system (with teams of variable size)*

$$\Gamma_7 = (\{A, B, A', B'\}, \{a, b, c\}, S, P_1, P_2, P_3, P_4, P_5, \{P_1\}, \{P_2, P_3\}, \{P_4, P_5\}),$$

where

$$\begin{aligned} P_1 &= \{S \to AB\}, \\ P_2 &= \{A \to aA'b, A \to ab\}, \\ P_3 &= \{B \to cB', B \to c\}, \\ P_4 &= \{A' \to A\} \text{ and} \\ P_5 &= \{B' \to B\}. \end{aligned}$$

Due to the formation of the teams, rewriting always consists of using a team for one step. Mode t_0 *thus equals mode* $= 1, \geq 1, *$ *as well as mode* $\leq k$ *for a* $k \geq 1$ *and therefore the mode is not mentioned explicitly throughout the explanation. Any derivation starts with using the first team, resulting in* AB*. This team will subsequently never be used again.*

Consider rewriting according to the weak rewriting step. From AB*, one can continue only with the second team, leading to one of the three sentential forms* $aA'bcB'$*,* $aA'bc$ *or* $abcB'$ *or to the terminal string* abc*. The only way to continue is by using the third team, resulting in the unprimed versions of the above sentential forms. Now this process can be iterated along one of the following steps, for* $n \geq 3$*.*

- $aAbcB \overset{w}{\Longrightarrow}_{Q_2} a^2b^2c^2$
- $aAbcB \overset{w}{\Longrightarrow}_{Q_2} a^2A'b^2c^2B' \overset{w}{\Longrightarrow}_{Q_3} a^2Ab^2c^2B \overset{w}{\Longrightarrow} \cdots \overset{w}{\Longrightarrow} a^nb^nc^n$
- $aAbcB \overset{w}{\Longrightarrow}_{Q_2} a^2A'b^2c^2 \overset{w}{\Longrightarrow}_{Q_3} a^2Ab^2c^2 \overset{w}{\Longrightarrow} \cdots \overset{w}{\Longrightarrow} a^nb^nc^2$
- $aAbcB \overset{w}{\Longrightarrow}_{Q_2} a^2b^2c^2B' \overset{w}{\Longrightarrow}_{Q_3} a^2b^2c^2B \overset{w}{\Longrightarrow} \cdots \overset{w}{\Longrightarrow} a^2b^2c^n$
- $aAbc \overset{w}{\Longrightarrow}_{Q_2} a^2b^2c$
- $aAbc \overset{w}{\Longrightarrow}_{Q_2} a^2A'b^2c \overset{w}{\Longrightarrow}_{Q_3} a^2Ab^2c \overset{w}{\Longrightarrow} \cdots \overset{w}{\Longrightarrow} a^nb^nc$
- $abcB \overset{w}{\Longrightarrow}_{Q_2} abc^2$
- $aAbc \overset{w}{\Longrightarrow}_{Q_2} abc^2B' \overset{w}{\Longrightarrow}_{Q_3} abc^2B \overset{w}{\Longrightarrow} \cdots \overset{w}{\Longrightarrow} abc^n$

Thus, Γ_7 *generates the language*

$$L_f^w(\Gamma_7) = \{a^mb^mc^n \mid m, n \geq 1\},$$

for $f \in \{= 1, \geq 1, *, t_0\} \cup \{\leq k \mid k \geq 1\}$.

Next, consider rewriting according to the strong rewriting step. From AB*, one can continue only with the second team. However, since a strong rewriting step requires using a production from every component of a team in every step, only the sentential form* $aA'bcB'$ *or the terminal string* abc *can result from using the*

second team. Consequently, using the third team, the primes in this sentential form can be removed and the process can be iterated. This clearly results in the generated language

$$L_f(\Gamma_7) = \{a^n b^n c^n \mid n \geq 1\},$$

for $f \in \{=1, \geq 1, *, t_0, t_1, t_2\} \cup \{\leq k \mid k \geq 1\}$.

1.2 Hybrid (prescribed) teams of grammars

Now, consider hybrid versions of the forming of teams in CD grammar systems, as defined in Section 1.1, in the style of the hybrid CD grammar systems. Two basically different versions can be defined. One can consider a hybrid CD grammar system and automatically form teams of its components according to some fixed strategy or one can consider a prescribed team CD grammar system and simply associate a (possibly different) mode of derivation with each team.

Both definitions will be presented in this section and their differences will be illustrated by some examples. To start with the first idea, consider a hybrid CD grammar system and automatically form teams by combining all components with a certain mode of derivation to form a team with that mode of derivation. Because the teams are formed automatically, they are not part of the system "hardware", but a way to define the work of the system. This results in the following definition.

Definition 15 *Consider a hybrid CD grammar system*

$$\Gamma = (N, T, S, (P_1, f_1), (P_2, f_2), \ldots, (P_n, f_n)).$$

Then teams $(Q_i, g_i) \subseteq \{(P_1, f_1), (P_2, f_2), \ldots, (P_n, f_n)\}$ *can be automatically formed in the following way. For* $g_i \in \{*, t_0, t_1, t_2\} \cup \{\leq k, =k, \geq k \mid k \geq 1\}$

$$(Q_i, g_i) = \{(P_k, f_k) \mid f_k = g_i,\ 1 \leq k \leq n\}.$$

Such a team $(Q_i, g_i) = \{(P_{j_1}, f_{j_1}), (P_{j_2}, f_{j_2}), \ldots, (P_{j_s}, f_{j_{s_i}})\}$, *is called an automatically formed team operating in mode* g_i.

The language generated by Γ *with automatically formed teams is*

$$L^{aut}(\Gamma) = \{z \in T^* \mid S \Longrightarrow_{Q_{i_1}}^{g_{i_1}} w_{i_1} \Longrightarrow_{Q_{i_2}}^{g_{i_1}} \cdots \Longrightarrow_{Q_{i_m}}^{g_{i_m}} w_{i_m} = z,\ m \geq 1\}.$$

The family of languages generated by hybrid CD grammar systems with automatically formed teams of variable size and only λ-free context-free productions is denoted by HT_*CD. Note that due to the automatical construction from a hybrid CD grammar system (with a one-symbol axiom), the notion of teams of constant size is very restricted. Only teams of constant size 1 could be constructed, but they obviously have the same generative power as the underlying

hybrid CD grammar system. Naturally, it is possible to consider hybrid CD grammar systems with a string axiom instead of a single nonterminal.

This new notion of forming teams in grammar systems is illustrated by the next example.

Example 8 *Consider the hybrid CD grammar system*

$$\Gamma_8 = (\{S, A, B\}, \{a, b, c\}, S, (P_1, \leq 1), (P_2, = 4), (P_3, = 4), (P_4, = 1), (P_5, = 1)),$$

where

$$\begin{aligned} P_1 &= \{S \to AB\}, \\ P_2 &= \{A \to aAb\}, \\ P_3 &= \{B \to cB\}, \\ P_4 &= \{A \to ab\} \text{ and} \\ P_5 &= \{B \to c\}. \end{aligned}$$

There is one component operating in mode ≤ 1 *and there are two components operating in mode* $= 4$*, as well as two in* $= 1$*. Hence the following three automatically formed teams result in a hybrid CD grammar system with automatically formed teams of variable size.*

$$\begin{aligned} (Q_1, \leq 1) &= \{(P_1, \leq 1)\}, \\ (Q_2, = 4) &= \{(P_2, = 4), (P_3, = 4)\} \text{ and} \\ (Q_3, = 1) &= \{(P_4, = 1), (P_5, = 1)\}. \end{aligned}$$

Often the modes are omitted in case of the components, thus writing

$$\begin{aligned} (Q_1, \leq 1) &= \{P_1\}, \\ (Q_2, = 4) &= \{P_2, P_3\} \text{ and} \\ (Q_3, = 1) &= \{P_4, P_5\}. \end{aligned}$$

One can start from the axiom to AB*, which leads to the terminal string* abc *using* Q_3 *in mode* $= 1$ *or to* $a^4Ab^4c^4B$ *using* Q_2 *in mode* $= 4$*. Using again* Q_3 *leads to the terminal string* $a^5b^5c^5$ *and once more* Q_2 *leads to* $a^8Ab^8c^8B$*. Consequently, it is easy to see that using* Q_2 *repeatedly generates* $a^nAb^nc^nB$*, where* $n \bmod 4 \equiv 1$*, which can be made terminal by using* Q_3*, thus yielding*

$$L^{aut}(\Gamma_8) = \{a^nb^nc^n \mid n \bmod 4 \equiv 1, n \geq 1\}.$$

Note that the hybrid CD grammar system

$$\Gamma_9 = (\{S, A, B\}, \{a, b, c\}, S, (P_1, = 1), (P_2, = 4), (P_3, = 4), (P_4, = 1), (P_5, = 1)),$$

where

$$\begin{aligned} P_1 &= \{S \to AB\}, \\ P_2 &= \{A \to aAb\}, \\ P_3 &= \{B \to cB\}, \\ P_4 &= \{A \to ab\} \text{ and} \\ P_5 &= \{B \to c\} \end{aligned}$$

results in the teams

$$\begin{aligned} (Q_1, =1) &= \{P_1, P_4, P_5\} \text{ and} \\ (Q_2, =4) &= \{P_2, P_3\} \end{aligned}$$

being formed automatically. This is not a very useful hybrid CD grammar system with automatically formed teams, however, since initially no team can be used to rewrite the axiom. Hence,

$$L^{aut}(\Gamma_9) = \emptyset.$$

This shows the importance of different modes for component 1 *than for components* 3 *and* 4.

Note that this example shows the way in which the choice of modes of derivations for components can influence the teams that are formed automatically. Moreover, note that the hybrid CD grammar system Γ_8 does not generate the same language with automatically formed teams as without any teams, since the reader can check that $L(\Gamma_8) = \{a^m b^m c^n \mid m \bmod 4 \equiv 1,\ n \bmod 4 \equiv 1,\ m, n \geq 1\}$.

As said before, one can also consider a hybrid version of the definition of prescribed teams. In contrast with Definition 15, one does not start from a hybrid CD grammar system, but simply lists the teams with their components and mode of derivation. Hence every team has a mode of derivation associated with it, but different teams may have the same mode of derivation. This results in the following definition, which is in fact more general than Definition 15.

Definition 16 *A* (prescribed) hybrid team CD grammar system *is a construct*

$$\Gamma = (N, T, S, P_1, P_2, \ldots, P_n, (Q_1, f_1), (Q_2, f_2), \ldots, (Q_m, f_m)),$$

where $(N, T, S, P_1, P_2, \ldots, P_n)$ *is a CD grammar system and* (Q_i, f_i) *is called a prescribed team operating in mode* f_i*, with* $Q_i \subseteq \{P_1, P_2, \ldots, P_n\}$ *and* $f_i \in \{*, t_0, t_1, t_2\} \cup \{\leq k, =k, \geq k \mid k \geq 1\}$, $1 \leq i \leq m$.

The language generated by Γ *is*

$$L(\Gamma) = \{z \in T^* \mid S \Longrightarrow_{Q_{i_1}}^{f_{i_1}} w_{i_1} \Longrightarrow_{Q_{i_2}}^{f_{i_2}} \cdots \Longrightarrow_{Q_{i_p}}^{f_{i_p}} w_{i_p} = z, \quad 1 \leq i_j \leq m,\ 1 \leq j \leq p\}.$$

The family of languages generated by hybrid CD grammar systems with prescribed teams of variable size and only λ-free context-free productions is denoted by HPT_*CD. When dealing with teams of constant size s a finite set of axioms $W \subseteq (N \cup T)^*$ with only one string in it containing nonterminals is allowed. Naturally, the S in the construct is replaced by this W and the subscript $*$ is replaced by s and the generated language becomes $L(\Gamma, s)$. All this is similar to the homogeneous case of Section 1.1.

This new notion of forming teams in grammar systems is demonstrated by the next example.

Example 9 *The prescribed hybrid team CD grammar system (with teams of variable size)*

$$\Gamma_{10} = (\{S, A, B\}, \{a, b, c\}, S, P_1, P_2, P_3, P_4, P_5, (Q_1, =1), (Q_2, =4), (Q_3, =1)),$$

where

$$\begin{aligned} P_1 &= \{S \to AB\}, \\ P_2 &= \{A \to aAb\}, \\ P_3 &= \{B \to cB\}, \\ P_4 &= \{A \to ab\}, \\ P_5 &= \{B \to c\}, \\ Q_1 &= \{P_1\}, \\ Q_2 &= \{P_2, P_3\} \text{ and} \\ Q_3 &= \{P_4, P_5\} \end{aligned}$$

obviously yields the same language as Γ_8 *in Example 8 did.*

Thus,

$$L(\Gamma_{10}) = \{a^n b^n c^n \mid n \bmod 4 \equiv 1, n \geq 1\}.$$

Note that due to the prescribed manner of the teams in prescribed hybrid team CD grammar systems, it is possible to have two different teams working in the same mode of derivation. Compare this to the hybrid CD grammar systems with automatically formed teams, where this would result in the forming of one team operating in that very mode.

The hybrid team CD grammar systems considered in this section combine the features of hybrid CD grammar systems and of teams in CD grammar systems. Until now, the mode of derivation was the same for all teams. Now, however, different teams can have different modes of derivation.

1.3 Controlled teams of grammars

The interchanging of teams during the derivation process in the grammar systems of the previous sections is not very enhanced, in the sense that there is no control on the order of the teams. Only very weak conditions are given to decide when one team stops and another one starts working.

In this section a generalization is considered by adding control mechanisms. Three different types will be considered in the following sections. In Subsection 1.3.1 *static external control* in the form of a *directed graph* is considered. It is called static control since the current state of the problem is not taken into consideration.

In Subsection 1.3.2 some conditions on the form of the sentential form are considered, in the form of a *hypothesis (or target) language*. This is a form of *dynamic control* since it takes the current state of the problem into consideration.

Finally, in Subsection 1.3.3 a *generalized sequential machine* (gsm) is used to intermediate between the uses of teams. Also this is an example of *dynamic control*, since the decision which team will be used next, after a certain team has stopped, is based on the current sentential form.

1.3.1 External control

In this section the definition of prescribed hybrid team CD grammar systems controlled by a directed graph is presented. The notion of derivations controlled by a directed graph was already presented in [Wood 73], [MR 78] and [CD 90] for Chomsky grammars, cooperating grammar systems and CD grammar systems, respectively. In [GP 90] and [Das 91a] results on graph controlled CD grammar systems with appearance checking and characterizations of graphs associated to specific language classes, respectively, can be found.

Definition 17 *A* prescribed hybrid team CD grammar system with graph control *is a construct*

$$\Gamma = (N, T, S, P_1, P_2, \ldots, P_n, (Q_1, f_1), (Q_2, f_2), \ldots, (Q_m, f_m), U),$$

where $(N, T, S, P_1, P_2, \ldots, P_n, (Q_1, f_1), (Q_2, f_2), \ldots, (Q_m, f_m))$ *is a prescribed hybrid team CD grammar system and* $U = (V, E)$ *a directed graph with set of nodes* V *and set of edges* E*, the* m *nodes of which are labelled by* Q_i*,* $1 \leq i \leq m$*.*

The language generated by Γ *and controlled by* U *is then*

$$L^U(\Gamma) = \{z \in T^* \mid S \Longrightarrow_{Q_{i_1}}^{f_{i_1}} w_{i_1} \Longrightarrow_{Q_{i_2}}^{f_{i_2}} \cdots \Longrightarrow_{Q_{i_p}}^{f_{i_p}} w_{i_p} = z, \\ (Q_{i_k}, Q_{i_{k+1}}) \in E,\ 1 \leq i_j \leq m,\ 1 \leq j \leq p,\ 1 \leq k \leq p-1\}$$

Note that the graph has exactly one node for each team.

The family of languages generated by prescribed hybrid team CD grammar systems with graph control and only λ-free context-free productions in P_j, $1 \leq j \leq n$, is denoted by $GCHPT_*CD$. For subclasses without teams, as suggested in Remark 1 below, the graph has exactly one node for each component (even labelled by the component).

Remark 1 *Definition 17 is the most general definition among those considered here; subclasses without teams, with non-hybrid teams, with (free) teams of constant size, with only regular productions and many more such restrictions can be imagined. The notation is adapted according to the conventions made in this thesis for such definitions.*

The next example illustrates one such a subclass.

Example 10 *Consider the hybrid CD grammar system with graph control*

$$\Gamma_{11} = (\{S, A, B\}, \{a, b\}, S, (P_1, t), (P_2, =1), (P_3, t), (P_4, t), (P_5, t), U),$$

where

$$\begin{array}{rcl} P_1 & = & \{S \to ABS, S \to AB\}, \\ P_2 & = & \{A \to a\}, \\ P_3 & = & \{B \to bB'\}, \\ P_4 & = & \{B \to b\}, \\ P_5 & = & \{B' \to B\} \text{ and} \\ U & \text{is} & \text{the following graph.} \end{array}$$

$$\begin{array}{ccccc} P_1 & \longrightarrow & P_2 & \longrightarrow & P_4 \\ & \nearrow & \downarrow & & \\ P_5 & \longleftarrow & P_3 & & \end{array}$$

Using P_1 for i times results in the sentential form $(AB)^i$, $i \geq 1$. Then every time P_2 changes one *A into an a, the components P_3 and P_5 change* all *B's into bB. This is repeated until there is only one A remaining in the sentential form, after which another path in the graph is taken to replace also this last A by a and consequently replacing all B's by b, thus obtaining a terminal string. No terminal string can be obtained if the last A's replacement by a is followed by replacing all B's by bB' since this would eventually require another application of P_2 before yielding a terminal string, which is not possible due to the absence of A's. It is thus clear that*

$$L^U(\Gamma_{11}) = \{(ab^n)^n \mid n \geq 1\}.$$

Hence already only one non-metalinear production, no teams and an easy graph suffice to generate a non-ET0L language.

1.3.2 Internal control

One can imagine that teams work together to reach a certain goal, which can be modelled by assuming a hypothesis (target) language with which the sentential forms are compared between every use of a team. Next the definition of prescribed hybrid team CD grammar systems with a (regular) hypothesis language is given. The notion of a hypothesis language (or the only slightly different concept of regular restriction) was already introduced in [Friš 68], [Das 91b] and [Păun 95a] for context-free grammars, CD grammar systems and colonies, respectively. See also [Král 73] for a result on context-free grammars with a weak regular restriction.

Definition 18 *A* prescribed hybrid team CD grammar system with a (regular) hypothesis language *is a construct*

$$\Gamma = (N, T, S, P_1, P_2, \ldots, P_n, (Q_1, f_1), (Q_2, f_2), \ldots, (Q_m, f_m), R),$$

where $(N, T, S, P_1, P_2, \ldots, P_n, (Q_1, f_1), (Q_2, f_2), \ldots, (Q_m, f_m))$ *is a prescribed hybrid team CD grammar system and* R *is a regular language in* $(N \cup T)^* \setminus T^*$.

A derivation consists of accepted derivation steps, where the derivation $x \Longrightarrow_{Q_i}^{f_i} y$ *(for* $f_i \in \{*, t_0, t_1, t_2\} \cup \{\leq k, = k, \geq k \mid k \geq 1\}$ *and* $1 \leq i \leq m$*) is accepted iff* $y \in R$ *or* $y \in T^*$.

The language generated by Γ *with hypothesis language* R *is*

$$L^R(\Gamma) = \{z \in T^* \mid S \Longrightarrow_{Q_{i_1}}^{f_{i_1}} w_{i_1} \Longrightarrow_{Q_{i_2}}^{f_{i_2}} \cdots \Longrightarrow_{Q_{i_p}}^{f_{i_p}} w_{i_p} = z, \\ w_{i_k} \in R,\ 1 \leq i_j \leq m,\ 1 \leq j \leq p,\ 1 \leq k \leq p-1\}$$

Note that no hypothesis is made about the final terminal string. The regularity of the hypothesis language is motivated by the fact that the test for the condition $y \in R$ can be done in linear time.

The family of languages generated by prescribed hybrid team CD grammar systems with a hypothesis language and only λ-free context-free productions in P_j, $1 \leq j \leq n$, is denoted by $HLHPT_*CD$. This is the most general definition considered; subclasses similar to the ones as proposed in Remark 1 can be imagined.

The definition is illustrated for one such a subclass mentioned in Remark 1 in the next example.

Example 11 *Consider the prescribed team CD grammar system with teams of constant size* 3 *and a hypothesis language*

$$\Gamma_{12} = (\{A, B, C\}, \{a, b, c\}, ABC, P_1, P_2, P_3, \{P_1, P_2, P_3\}, R),$$

where

$$\begin{aligned} P_1 &= \{A \to aA, A \to a\}, \\ P_2 &= \{B \to bB, B \to b\}, \\ P_3 &= \{C \to cC, C \to c\} \text{ and} \\ R &= T^+\{A\}T^+\{B\}T^+\{C\}. \end{aligned}$$

There is only one team and the regular language requires the presence of the three nonterminals A, B and C in every nonterminal sentential form. Thus using the team by the three second productions of its components leads to the terminal string abc. On the other hand, using the team by the three first productions of its components leads to the sentential form $a^{i-1}Ab^{i-1}Bc^{i-1}C$, $i \geq 2$. Then the team can make a terminal string $a^ib^ic^i$, $i \geq 2$, by using the three second productions of its components. No other selection of productions from the components than these described above is possible due to the form of the hypothesis language. It is thus clear that

$$L_f^R(\Gamma_{12}, 3) = \{a^nb^nc^n \mid n \geq 1\}$$

for $f \in \{=1, \geq 1, *, t_0, t_1, t_2\} \cup \{\leq k \mid k \geq 1\}$.

Hence already regular productions, *one* non-hybrid team of size 3 and a very easy regular language suffice to generate a non-context-free language.

In fact, adding the team $\{S \to ABC\}$ (S being the axiom) and changing the $+$'s in R into $*$'s results in a prescribed team CD grammar system with two teams of variable size (1 and 3) and only metalinear productions generating the same language. Using hybrid teams also the language $\{a^nb^nc^n \mid n \ mod \ 4 \equiv 1, n \geq 1\}$, which was used as running example in the previous sections, can be generated by a very concise grammar system using the same hypothesis language.

1.3.3 Intermediate control

A real life feature of team cooperation is the possibility that the teams involved speak different languages. To be able to reach a goal by cooperation, an interpreter is necessary to intermediate between them, i.e. in a grammar system a translation takes place after one team stops and before another one starts working. This idea can be modelled by using a generalized sequential machine (gsm) in team cooperation, formally presented in the next definition. This approach was already investigated in [Mit 90] and [Păun 95a] for CD grammar systems and colonies, respectively.

Definition 19 *A* prescribed hybrid team CD grammar system with a generalized sequential machine (gsm) *is a construct*

$$\Gamma = (N, T, S, P_1, P_2, \ldots, P_n, (Q_1, f_1), (Q_2, f_2), \ldots, (Q_m, f_m), g),$$

where $(N, T, S, P_1, P_2, \ldots, P_n, (Q_1, f_1), (Q_2, f_2), \ldots, (Q_m, f_m))$ *is a prescribed hybrid team CD grammar system and*

$$g = (K, N \cup T, N \cup T, s_0, \delta, H)$$

is a gsm.

For $f_i \in \{*, t_0, t_1, t_2\} \cup \{\leq k, = k, \geq k \mid k \geq 1\}$ *and* $1 \leq i \leq m$ *the language generated by* Γ *with gsm* g *is*

$$L^g(\Gamma) = \{z \in T^* \mid S \Longrightarrow_{Q_{i_1}}^{f_{i_1}} w_{i_1} \Longrightarrow g(w_{i_1}) \Longrightarrow_{Q_{i_2}}^{f_{i_2}} w_{i_2} \Longrightarrow g(w_{i_2}) \Longrightarrow_{Q_{i_3}}^{f_{i_3}} \cdots$$
$$\cdots \Longrightarrow_{Q_{i_p}}^{f_{i_p}} w_{i_p} = z,\ 1 \leq i_j \leq m,\ 1 \leq j \leq p\}$$

The family of languages generated by prescribed hybrid team CD grammar systems with a gsm and only λ-free context-free productions in P_j, $1 \leq j \leq n$, is denoted by $GSMHPT_*CD$. Again, this is the most general definition considered; subclasses as proposed in Remark 1 can be imagined.

Once again, the definition is illustrated for such a subclass in the next example.

Example 12 *Consider the prescribed team CD grammar system with teams of variable size and a gsm*

$$\Gamma_{13} = (\{S, A, B, C, D\}, \{a, b, c, d\}, S, P_1, P_2, \ldots, P_9, Q_1, Q_2, Q_3, Q_4, g),$$

where

$$\begin{array}{lcll} P_1 & = & \{S \to abcd\}, & Q_1 = \{P_1\}, \\ P_2 & = & \{A \to aa\}, & Q_2 = \{P_2, P_4\}, \\ P_3 & = & \{B \to bb\}, & Q_3 = \{P_3, P_5\}, \\ P_4 & = & \{C \to cc\}, & Q_4 = \{P_6, P_7, P_8, P_9\}, \\ P_5 & = & \{D \to dd\}, & \\ P_6 & = & \{A \to a\}, & \\ P_7 & = & \{B \to b\}, & \\ P_8 & = & \{C \to c\}, & \\ P_9 & = & \{D \to d\} \textit{ and} & \end{array}$$

$$g = (\{s_a, s_b, s_c, s_d, s_z\}, I, O, s_a, \delta, \{s_z\}),$$

where

$$I = O = \{S, A, B, C, D, a, b, c, d\} \textit{ and}$$

$$\delta = \{(s_a a, A s_b), (s_a A, A s_b), (s_b a, a s_b), (s_b b, B s_c), (s_b B, B s_c), (s_c b, b s_c),$$
$$(s_c c, C s_d), (s_c C, C s_d), (s_d c, c s_d), (s_d d, D s_z), (s_d D, D s_z), (s_z d, d s_z)\}.$$

Initially, only team Q_1 can be used, resulting in the sentential form abcd. This team can now never be used again. The gsm translates this string into ABCD. The gsm always translates the first a, b, c and d that it meets into A, B, C and D, respectively, if this particular nonterminal is not yet present in the sentential form, meanwhile skipping specific intermediate terminals and nonterminals. From ABCD the derivation can be continued by using either Q_2, Q_3 or Q_6. Team Q_6 results in the terminal string abcd. Using Q_2 (Q_3) leads to aaBccD (AbbCdd), which the gsm thus translates into AaBCcD (ABbCDd). Repeating this process, strings of the form $Aa^ib^jCc^id^j$ or $a^iBb^jc^iDd^j$ are translated into strings of the form $Aa^iBb^jCc^iDd^j$, $i \geq 1$ and $j \geq 0$ ($i \geq 0$ and $j \geq 1$). Finally, team Q_6 can then be used (only after translation since it requires the presence of A, B, C and D in the sentential form) to obtain terminal strings of the form $a^ib^jc^id^j$, $i \geq 1$ and $j \geq 1$. It is thus clear that

$$L_f^g(\Gamma_{13}) = \{a^nb^mc^nd^m \mid m, n \geq 1\}$$

for $f \in \{=1, \geq 1, *, t_0, t_1, t_2\} \cup \{\leq k \mid k \geq 1\}$.

Note that the gsm in the example is in fact a Mealy machine since it is deterministic and every production in the set δ has only one output-letter. Hence already regular productions, non-hybrid teams of variable size and a restricted gsm suffice to generate a non-context-free language.

In fact, teams are not even necessary. The hybrid CD grammar system $\Gamma'_{12} = (\{S, A, B, C, D\}, \{a, b, c, d\}, S, (P_1, = 1), (P'_2, = 2), (P'_3, = 2), (P'_4, = 4), g)$, where $P'_2 = P_2 \cup P_4$, $P'_3 = P_3 \cup P_5$ and $P'_4 = P_6 \cup P_7 \cup P_8 \cup P_9$ and the rest as in Γ_{13} in the example above, clearly also generates $L^g(\Gamma'_{12}) = \{a^nb^mc^nd^m \mid m, n \geq 1\}$.

1.4 Conclusion

In the first section, mostly definitions from [KMPS 95] and [PR 94] were recalled. The (prescribed) team CD grammar systems with teams of constant size form teams of a given constant size from a CD grammar system. Moreover, they require a string axiom with at least s nonterminals in it (s being the size of the teams) in order to guarantee any rewriting at all. In the (prescribed) team CD grammar systems with teams of variable size this is not necessary. One simply lists all the teams of possibly different size. Obviously, the definition of prescribed teams is a generalization of the original definition of team CD grammar systems.

Furthermore, a variation of the (strong) rewriting step was proposed. This new way of rewriting is called weak rewriting. It resembles the well-known concept of appearance checking; every component of a team which contains a production that can rewrite the sentential form must be used, but a component which does not contain any production with a left-hand side that is contained in the sentential form does not need to be used.

The notions of hybrid versions of the definitions from the first section are introduced in the second section. Two ways of forming hybrid team CD grammar systems are considered. In hybrid CD grammar systems with automatically formed teams, these teams are formed of all components with the same mode of derivation in the hybrid CD grammar system. Their size may thus differ. Prescribed hybrid team CD grammar systems simply list the different teams with their mode of derivation. This mode of derivation is the same for every member of a team, but there may be more teams with the same mode of derivation. This is in contrast with the hybrid CD grammar systems with automatically formed teams, where each team operates in a unique mode. Obviously, this latter definition is a generalization of all the three other ones appearing in Subsections 1.1 and 1.2.

All these team CD grammar systems defined in the first two sections allow several components to be active simultaneously. The non-context-free language $\{a^n b^n c^n \mid n \ mod \ 4 \equiv 1, n \geq 1\}$ is used as a running example throughout the sections. In the two new definitions concerning hybrid teams the grammar systems generating this language are surprisingly elegant. Hence an advantage over the usual (hybrid) CD grammar systems should not only be expected in generative power, but also in the elegance of the grammar systems. In this case, elegance could be understood as the amount of symbols used in the productions, to name just one possibility. However, the syntactical complexity of grammar systems with teams is beyond the scope of this thesis.

The forming of teams of grammars also allows the introduction of three variants of the maximal rewriting strategy known for CD grammar systems. In the variant closely based on the original one, the rewriting of a team comes to a halt when the team as a whole can no longer perform a rewriting step. In the other maximal modes of derivation, the rewriting of a team ends when no component of the team can work further on the current sentential form or if there is at least one component that can no longer rewrite the current sentential form. In the next section the generative power of the forming of teams in (hybrid) CD grammar systems for these, as well as for the usual modes of derivation of CD grammar systems extended to teams, is investigated.

The last section presents a survey of adding mechanisms to control the derivations in grammar systems. Three variants are considered, not previously defined for prescribed hybrid team CD grammar systems; naturally also subclasses, for example without teams, can be considered. When the control is imposed by a graph, the next team to be used has to be connected with the current team by an edge. In the second variant, the intermediate sentential forms have to be part of a regular language, called the hypothesis language. Finally, when a gsm is added to the grammar system, the intermediate sentential forms are translated before using a team again. The generative power of these controlled grammar systems will also be dealt with in the next section.

Finally, an interesting problem posed in [PR 94] remains to be investigated. In all definitions above, when more teams are able to rewrite the sentential form,

one of these teams is chosen non-deterministically. Perhaps a priority relation among the teams could solve this concurrency matter.

2 Generative power

In this section the generative power of the systems defined in Section 1 is investigated. To begin with, a lemma is stated that follows immediately from those definitions.

(Recall Notation 1, which states that the type of productions that are used in a language generating device appear as a subscript in the notation for the family of languages generated by this device.)

Lemma 2 *For* $s \geq 1$, $X \in \{REG, LIN, CF, CS, RE\}$, $Y \in \{MLIN\}$ *and* $f \in \{*, t_0, t_1, t_2\} \cup \{\leq k, = k, \geq k \mid k \geq 1\}$

(i) $T_sCD_X(f) \subseteq PT_sCD_X(f) \subseteq PT_*CD_X(f)$,
$T_*CD_X(f) \subseteq PT_*CD_X(f) \subseteq HPT_*CD_X$ *and*
$T_+CD_X(f) \subseteq PT_+CD_X(f) \subseteq PT_*CD_X(f)$,

(ii) $PT_sCD_X(f) \subseteq HPT_sCD_X \subseteq HPT_*CD_X$ *and*
$PT_+CD_X(f) \subseteq HPT_+CD_X \subseteq HPT_*CD_X$,

(iii) $HCD_X = HT_1CD_X \subseteq HPT_sCD_X \subseteq HPT_*CD_X$,
$HT_1CD_X \subseteq HT_*CD_X \subseteq HPT_*CD_X$ *and*
$HT_+CD_X \subseteq HPT_+CD_X$ *and*

(iv) $T_*CD_Y(f) \subseteq PT_*CD_Y(f) \subseteq HPT_*CD_Y$,
$HT_*CD_Y \subseteq HPT_*CD_Y$ *and*

(v) $T_sCD_X(f) \subseteq T_{s+1}CD_X(f)$,
$PT_sCD_X(f) \subseteq PT_{s+1}CD_X(f)$ *and*
$HPT_sCD_X \subseteq HPT_{s+1}CD_X$.

These relations continue to hold in the case when λ-productions are allowed.

Consequently, combining some results from [KMPS 95] and [CP 93] the following lemma is obtained.

Lemma 3 *For* $f \in \{=1, \geq 1, *\} \cup \{\leq k \mid k \geq 1\}$ *and* $s \geq 2$

$$CF = T_1CD(f) \subset T_2CD(f),$$
$$ET0L = T_1CD(t_1) \subset T_2CD(t_1) \text{ and}$$
$$T_sCD(t_1) \subseteq T_2CD(t_1).$$

The first inclusion was proved in [KMPS 95]. It shows that there are modes of derivation in which the forming of teams strictly increases the power of CD grammar systems, since for $f \in \{=1, \geq 1, *\} \cup \{\leq k \mid k \geq 1\}$ it is known that $CF = CD(f)$ holds (see Part II, Section 5.1). The second statement of the lemma also appeared in [KMPS 95]; it shows that also in the t_1-mode the forming of teams strictly increases the power of CD grammar systems, since $ET0L = CD(t)$ holds (see Part II, Section 5.1). The third statement says that teams of size two suffice and it was proved in [CP 93].

In the following sections the generative power of teams of grammars will be investigated, split into several sections according to the modes of derivation or the type of productions that are allowed in the grammars. Section 2.1 investigates the power of (prescribed) team CD grammar systems for the modes t_0, t_1 and t_2 in Subsection 2.1.1 and for the other modes of derivation in Subsection 2.1.2.

Section 2.2 deals with the generative power of the new definitions concerning hybrid teams of grammars that appeared in Section 1.2.

In Section 2.3 the generative power of (hybrid) prescribed teams of grammars with other productions than context-free productions in the grammars is studied. The linear and regular cases appear in Subsection 2.3.1 and the metalinear case appears in Subsection 2.3.2. Some special cases are considered in Subsection 2.4 and their generative power is investigated.

In Section 2.5 the effect on generative power by adding control mechanisms to the various versions of teams in grammar systems, as proposed in Section 1.3, is covered.

Finally, Section 3 presents a summary of the obtained results in the form of a hierarchy-diagram. Moreover, some remaining open problems concerning the generative power of teams of grammars are listed.

2.1 (Prescribed) teams of grammars: the context-free case

The proofs of all the results in this section can be found in [PR 94] and [FP 95].

2.1.1 The t_0, t_1 and t_2 modes of derivation

The next lemma was posed in [PR 94]. Its proof there is not covering all cases of the statement, however. The lemma is valid, though, as will be shown by the

following proof. It is based on the proof as given in [PR 94]. A different semantics in the notation is chosen and, of course, the cases which failed in that proof are covered here.

Lemma 4 *For $s \geq 2$*

$$PR_{ac} \subseteq T_sCD(t_1) \text{ and } PR_{ac}^{\lambda} \subseteq T_sCD^{\lambda}(t_1).$$

Proof Consider a language $L \in PR_{ac}$ generated by the programmed grammar

$$G' = (N', T', S', P').$$

(Recall that productions in a programmed grammar have the form $(r : A \to x, \sigma(r), \varphi(r))$, where $r : A \to x$ is a production over $(N \cup T)^*$).

Construct the following team CD grammar system (with free teams of constant size)

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

where

$$\begin{array}{rcl} N & = & N' \cup \{A' \mid A \in N'\} \cup \\ & & \{A_r, A'_r \mid (r : A \to x, \sigma(r), \varphi(r)) \in P'\} \cup \\ & & \{S_1, S_2, D, D'\} \cup M, \text{ where} \\ M & = & \{C_r^1, C_r^2, C_r^3, C_r^4 \mid r \in Lab(P')\}, \\ T & = & T' \cup \{\#_1, \#_2\}, \\ w & = & S_1S_2 \text{ and} \\ P_1, P_2, \ldots, P_n & \text{are} & \text{the components } P_{i,j}^r,\ 0 \leq i \leq 5,\ j \in \{1,2\} \text{ and} \\ & & P_{6,j},\ j \in \{1,2\} \text{ and } r \in Lab(P'), \text{ as below.} \end{array}$$

For all $r \in Lab(P')$, it will be shown that these components P_i, $1 \leq i \leq n$, can only form teams Q_i^r, $0 \leq i \leq 5$, consisting of two components $P_{i,1}^r$ and $P_{i,2}^r$ and Q_6 consisting of the two components $P_{6,1}$ and $P_{6,2}$ to generate a language at all. The first team Q_0^r is, for all $r \in Lab(P')$, formed by

$$\begin{array}{rcl} P_{0,1}^r & = & \{S_1 \to D\} \cup \{X \to X \mid X \in M \setminus \{C_r^1\}\} \text{ and} \\ P_{0,2}^r & = & \{S_2 \to S'C_r^1\}. \end{array}$$

These teams are initially used to set the general form $DS'C_r^1$. D is a dummy symbol which, interchanged with the other dummy symbol D', guarantees that at any moment in time a symbol can be rewritten by each component of a team. C_r^1 is a control symbol which, interchanged with the other control symbols C_r^2, C_r^3 and C_r^4, simulates the programming of the programmed grammar. Finally,

S' is the "axiom" which is going to generate L. In the end, the two control symbols will be replaced by $\#_1$ and $\#_2$, respectively, thus yielding $\#_1 L \#_2$. The productions $X \to X$ will be appearing throughout the entire construction and are constantly used as trap-rules to permit the forming of a team from $P^r_{i,1}$ and $P^{r'}_{i,2}$ for $1 \leq i \leq 5$ with $r \neq r'$. This is guaranteed by the production $C^j_r \to C^1_j$ in the second (or $C^j_{r'} \to C^j_{r'}$ in the first) component of the teams, $1 \leq j \leq 4$, which can always be applied in the t_1-mode even when the other component cannot apply any production anymore.

The simulation is continued from S' with a production r of $Lab(P')$. For every production $(r : A \to x, \sigma(r), \varphi(r)) \in P'$, consider the components

$$
\begin{aligned}
P^r_{1,1} &= \{A \to A_r, A \to A'\} \cup \{X \to X \mid X \in M \setminus \{C^2_r\}\},\\
P^r_{1,2} &= \{C^1_r \to C^1_r, C^1_r \to C^2_r\},\\
P^r_{2,1} &= \{A_r \to A'_r\} \cup \{X \to X \mid X \in M \setminus \{C^3_r\}\},\\
P^r_{2,2} &= \{C^2_r \to C^3_r\},\\
P^r_{3,1} &= \{A'_r \to x, A' \to A\} \cup \{X \to X \mid X \in M \setminus \{C^1_s \mid s \in \sigma(r)\}\},\\
P^r_{3,2} &= \{C^3_r \to C^3_r\} \cup \{C^3_r \to C^1_s \mid s \in \sigma(r)\},\\
P^r_{4,1} &= \{A \to A, D \to D'\} \cup \{X \to X \mid X \in M \setminus \{C^4_r\}\},\\
P^r_{4,2} &= \{C^1_r \to C^4_r\},\\
P^r_{5,1} &= \{D' \to D\} \text{ and}\\
P^r_{5,2} &= \{C^4_r \to C^1_s \mid s \in \varphi(r), \varphi(r) \neq \emptyset\} \cup \{C^4_r \to D' \mid \varphi(r) = \emptyset\}.
\end{aligned}
$$

The teams Q^r_1 choose which occurrence of A in the sentential form to rewrite next, mark that one by A_r and the other occurrences by A' and hand over control to the teams Q^r_2. Note that the t_1-mode guarantees that C^1_r is indeed once replaced by C^2_r and that the trap-rules guarantee this is done for the right production r, an essential trick in this proof. Moreover, since the teams Q^r_2 can work only one step due to their second component, it is guaranteed that only one occurrence of A is replaced by A_r indeed.

The teams Q^r_2 hand over control to Q^r_3 and prime A_r. The teams Q^r_3 simulate production r by replacing A'_r by the string x, taking the primes away from the A's again and handing over control to C^1_s, corresponding to a production s in the success field of r. This process can be iterated, but eventually the teams Q^r_4 start their work.

These teams Q^r_4 are used in case production r cannot be applied, thus when A is not present in the sentential form. Their first component tests this occurrence of A and rewrites forever if A is present. Otherwise it changes the dummy symbol and hands over control to the teams Q^r_5. These teams change the dummy symbol again and hand over control to one of the productions in the failure field of r. Do note that, in the case of an empty failure field, its second component replaces the control symbol by the dummy symbol D'.

Finally, the teams Q_6, consisting of the components

$$\begin{aligned} P_{6,1} &= \{C_r^1 \to \#_2 \mid r \in Lab(P')\} \cup \{D' \to \#_2\} \cup \{X \to X \mid X \in N\} \text{ and} \\ P_{6,2} &= \{D \to \#_1\} \cup \{X \to X \mid X \in N\}, \end{aligned}$$

are used to replace the control symbol and (or) dummy symbol(s) by terminals thus yielding a terminal string. The trap-rules guarantee that these last teams cannot be used when any nonterminals (other than the dummy or control symbols) are still present in the sentential form. Do note $D' \to \#_2$ in the first component of Q_6. The case of empty failure fields is handled by allowing not only C_r^1 to be replaced by $\#_2$ for all $r \in Lab(P')$, but allowing also D' to be replaced by $\#_2$. This generates a terminal string, since eventually D and D' will be replaced by $\#_1$ and $\#_2$, respectively.

The teams described above are the only ones that can be formed automatically and work properly. This is due to the fact that trap-rules and well-chosen symbols permit any other combination to work in the t_1-mode of derivation. Furthermore, the control symbols guarantee the ordering of the teams as described above. Hence $L_{t_1}(\Gamma, 2) = L_{t_1}(\Gamma, s) = \#_1 L \#_2$ for $s \geq 2$. From [KMPS 95] it is known that $T_sCD(t_1)$ is a full AFL and thus closed under left and right derivatives. Hence $L \in T_sCD(t_1)$ if $\#_1 L \#_2 \in T_sCD(t_1)$ and $PR_{ac} \subseteq T_sCD(t_1)$, $s \geq 2$, is proved.

$PR_{ac}^{\lambda} \subseteq T_sCD_{ac}^{\lambda}(t_1)$, $s \geq 2$, can be proved by a similar construction, even simplified since the control symbols can eventually be replaced by λ, making the symbols $\#_1$ and $\#_2$ unnecessary. □

Do note the importance of non-empty success fields in programmed grammars for this proof and the one in [PR 94]. The following corollary is a direct consequence of the proof of Lemma 4.

Corollary 1 *$PR_{ac} \subseteq T_+CD(t_1)$ and $PR_{ac}^{\lambda} \subseteq T_+CD^{\lambda}(t_1)$.*

In [PR 94] it is also proved that the same statement holds in the case of derivation mode t_2. This is done according to the same principles as in the case of mode t_1. Hence the proof needs to be similarly adjusted; this will not be done here, only the lemma is stated.

Lemma 5 *For $s \geq 2$*

$$PR_{ac} \subseteq T_sCD(t_2) \text{ and } PR_{ac}^{\lambda} \subseteq T_sCD^{\lambda}(t_2).$$

The proof of Lemma 5 in [PR 94] leads directly to the following corollary.

Corollary 2 *$PR_{ac} \subseteq T_+CD(t_2)$ and $PR_{ac}^{\lambda} \subseteq T_+CD^{\lambda}(t_2)$.*

Another lemma, indirectly proved in [PR 94], is the following.

Lemma 6 *For* $f \in \{t_1, t_2\}$

$$PT_*CD(f) \subseteq MAT_{ac} \text{ and } PT_*CD^{\lambda}(f) \subseteq MAT_{ac}^{\lambda}.$$

It is known that $PR_{ac} = MAT_{ac}$ and $PR_{ac}^{\lambda} = MAT_{ac}^{\lambda}$ (see the Prerequisites), hence the following corollary follows directly from Lemma 6.

Corollary 3 *For* $f \in \{t_1, t_2\}$

$$PT_*CD(f) \subseteq PR_{ac} \text{ and } PT_*CD^{\lambda}(f) \subseteq PR_{ac}^{\lambda}.$$

Combining the lemmas and corollaries above, one of the main results of [PR 94] is obtained, presented in the following theorem.

Theorem 1 *For* $s \geq 2$ *and* $f \in \{t_1, t_2\}$

$$T_sCD(f) = PT_sCD(f) = PT_*CD(f) = T_+CD(f) = PR_{ac} \text{ and}$$
$$T_sCD^{\lambda}(f) = PT_sCD^{\lambda}(f) = PT_*CD^{\lambda}(f) = T_+CD^{\lambda}(f) = PR_{ac}^{\lambda}.$$

The following four lemmas appeared in [FP 95].

Lemma 7

$$PT_*CD(t_0) \subseteq MAT_{ac} \text{ and } PT_*CD^{\lambda}(t_0) \subseteq MAT_{ac}^{\lambda}.$$

Lemma 8

$$MAT_{ac} \subseteq T_2CD(t_0) \text{ and } MAT_{ac}^{\lambda} \subseteq T_2CD^{\lambda}(t_0).$$

Lemma 9 *For* $z \in \{+, *\}$

$$MAT_{ac} \subseteq T_zCD(t_0) \text{ and } MAT_{ac}^{\lambda} \subseteq T_zCD^{\lambda}(t_0).$$

Lemma 10 *For $s \geq 3$*

$$MAT_{ac} \subseteq T_sCD(t_0) \text{ and } MAT_{ac}^{\lambda} \subseteq T_sCD^{\lambda}(t_0).$$

This is proved in [FP 95] to lead to the following corollary, thereby slightly improving the statements of Lemma 4 and Corollary 1.

Corollary 4 *For $z \in \{+, *\} \cup \{2, 3, 4, \ldots\}$*

$$MAT_{ac} \subseteq T_zCD(t_1) \text{ and } MAT_{ac}^{\lambda} \subseteq T_zCD^{\lambda}(t_1).$$

Combining these results from [FP 95], the main result of that paper is obtained, presented in the following theorem.

Theorem 2 *For $z \in \{+, *\} \cup \{2, 3, 4, \ldots\}$ and $f \in \{t_0, t_1\}$*

$$MAT_{ac} = PT_zCD(f) = T_zCD(f) \text{ and } MAT_{ac}^{\lambda} = PT_zCD^{\lambda}(f) = T_zCD^{\lambda}(f).$$

Combining Theorem 1 and 2, the following theorem is obtained.

Theorem 3 *For $s \geq 2$ and $f \in \{t_0, t_1, t_2\}$*

$$T_sCD(f) = PT_sCD(f) = PT_*CD(f) = T_*CD(f) \text{ and}$$
$$T_sCD^{\lambda}(f) = PT_sCD^{\lambda}(f) = PT_*CD^{\lambda}(f) = T_*CD^{\lambda}(f).$$

2.1.2 The other modes of derivation

The following lemma is stated and proved in [PR 94].

Lemma 11 *For $s \geq 2$ and $f \in \{*\} \cup \{\leq k, =k, \geq k \mid k \geq 1\}$*

$$PR \subseteq PT_sCD(f) \text{ and } PR^{\lambda} \subseteq PT_sCD^{\lambda}(f).$$

The next lemma is indirectly proved in [PR 94].

Lemma 12 *For* $f \in \{*\} \cup \{\leq k, =k, \geq k \mid k \geq 1\}$

$$PT_*CD(f) \subseteq MAT \text{ and } PT_*CD^\lambda(f) \subseteq MAT^\lambda.$$

It is known that $PR = MAT$ and $PR^\lambda = MAT^\lambda$ (see the Prerequisites), hence the following corollary follows directly from Lemma 12.

Corollary 5 *For* $f \in \{*\} \cup \{\leq k, =k, \geq k \mid k \geq 1\}$

$$PT_*CD(f) \subseteq PR \text{ and } PT_*CD^\lambda(f) \subseteq PR^\lambda.$$

Combining Lemma 2 and Corollary 5, the following corollary is obtained.

Corollary 6 *For* $f \in \{*\} \cup \{\leq k, =k, \geq k \mid k \geq 1\}$

$$T_sCD(f) \subseteq PR \text{ and } T_sCD^\lambda(f) \subseteq PR^\lambda.$$

Combining Lemma 2 and the two previous lemmas above, another main result of [PR 94] is obtained, presented in the following theorem.

Theorem 4 *For* $s \geq 2$ *and* $f \in \{*\} \cup \{\leq k, =k, \geq k \mid k \geq 1\}$

$$PT_sCD(f) = PT_*CD(f) = PR \text{ and } PT_sCD^\lambda(f) = PT_*CD^\lambda(f) = PR^\lambda.$$

The following theorem is an important corollary of the above results. Its proof is a combination of Theorem 1, 2 and 4 and some proper inclusions stated in the Prerequisites concerning programmed grammars with or without λ-free productions and/or appearance checking.

Theorem 5 *For* $s \geq 2$, $f \in \{*\} \cup \{\leq k, =k, \geq k \mid k \geq 1\}$ *and* $g \in \{t_0, t_1, t_2\}$

$$PT_sCD(f) \subset PT_sCD(g),\ PT_+CD(f) \subset PT_+CD(g),$$
$$PT_sCD^\lambda(f) \subset PT_sCD^\lambda(g) \text{ and } PT_+CD^\lambda(f) \subset PT_+CD^\lambda(g).$$

2.1.3 Weak versus strong rewriting

It is not hard to see that the principle of weak rewriting, not having to apply productions if they cannot be applied, resembles the appearance checking feature in regulated rewriting. Therefore, the following two lemmas do not come as a surprise. In this section, a restriction to only one production per component will be indicated by a 1 added as subscript. To be even more precise, denote U_mSC_{ut} for the class of unordered scattered context grammars with unconditional transfer and m scattered context rules and denote $P_mT_wCD_1(f)$ for the class of prescribed team CD grammar systems with m teams of variable size, 1 production per component, operating in mode f and rewriting in the weak rewriting step.

Lemma 13 *For $m \geq 1$ and $f \in \{=1, \geq 1, *\} \cup \{\leq k \mid k \geq 1\}$*

$$U_mSC_{ut} \subseteq P_mT_wCD_1(f) \text{ and } U_mSC_{ut}^{\lambda} \subseteq P_mT_wCD_1^{\lambda}(f).$$

Proof Consider an unordered scattered context grammar

$$G' = (N', T', S', P', F')$$

with unconditional transfer and m scattered context rules. Moreover, for $P' = \{p_1, p_2, \ldots, p_m\}$, $p_i : (\alpha_{i,1}, \alpha_{i,2}, \ldots, \alpha_{i,k_i}) \rightarrow (\beta_{i,1}, \beta_{i,2}, \ldots, \beta_{i,k_i})$ and $1 \leq i \leq m$, denote

$$r_{i,j} = \alpha_{i,j} \rightarrow \beta_{i,j} \text{ for } 1 \leq j \leq k_i.$$

To simulate this unordered scattered context grammar, construct the prescribed team CD grammar system

$$\Gamma = (N, T, S, P_1, P_2, \ldots, P_n, Q_1, Q_2, \ldots, Q_m),$$

where

$$\begin{array}{rcl} N & = & N', \\ T & = & T', \\ S & = & S', \\ P_1, P_2, \ldots, P_n & \text{are} & \text{the components } \{r_{i,j}\} \text{ for } 1 \leq j \leq k_i \text{ and } 1 \leq i \leq m \text{ and} \\ Q_1, Q_2, \ldots, Q_m & \text{are} & \text{the teams } \{\{r_{1,j}\}, \{r_{2,j}\}, \ldots, \{r_{m,j}\}\} \text{ for } 1 \leq j \leq k_i \text{ and} \\ & & 1 \leq i \leq m. \end{array}$$

A parallel rewriting step of an unordered scattered context grammar is simulated by a parallel rewriting step of a team, with its components being exactly the same productions as in the scattered context rule. Every component contains exactly one such a production and the number of teams equals the number of

scattered context rules. Any production in G' as well as in Γ does not have to be applied, if it cannot be applied to the sentential form.

Note that the proof requires the unordered character of the scattered context grammar, for a component of a team can rewrite any occurrence of the left-hand side of its production in the current sentential form. Since a team has to simulate the use of a scattered context rule, its mode of derivation is restricted to the cases as stated in the lemma. Clearly, $L(\Gamma) = L(G')$ and the lemma is proved for the case with as well as the case without λ-productions. □

Lemma 14 *For $m \geq 1$ and $f \in \{=1, \geq 1, *\} \cup \{\leq k \mid k \geq 1\}$*

$$P_m T_w CD_1(f) \subseteq U_m SC_{ut} \text{ and } P_m T_w CD_1^{\lambda}(f) \subseteq U_m SC_{ut}^{\lambda}.$$

Proof Consider a prescribed team CD grammar system

$$\Gamma = (N, T, S, P_1, P_2, \ldots, P_n, Q_1, Q_2, \ldots, Q_m)$$

with 1 production per component. Define for $Q_i = \{P_{i_1}, P_{i_2}, \ldots, P_{i_{k_i}}\}$ and $1 \leq i \leq m$

$$P_{i_j} = \{\alpha_{i_j} \rightarrow \beta_{i_j}\} \text{ for } 1 \leq j \leq k_i.$$

To simulate this prescribed team CD grammar system, construct the unordered scattered context grammar

$$G' = (N', T', S', P', F')$$

with m scattered context rules, where

$$\begin{array}{lcl} N' & = & N, \\ T' & = & T, \\ S' & = & S, \\ P' & = & \{p_1, p_2, \ldots, p_m\} \text{ with, for } 1 \leq i \leq m, \\ & & p_i : (\alpha_{i_1}, \alpha_{i_2}, \ldots, \alpha_{i_{k_i}}) \rightarrow (\beta_{i_1}, \beta_{i_2}, \ldots, \beta_{i_{k_i}}) \text{ and} \\ F' & \text{contains} & \text{every production in } P'. \end{array}$$

A parallel rewriting step of a prescribed team CD grammar system operating in mode $= 1$, ≥ 1, $*$ or $\leq k$ (for a $k \geq 1$) is simulated by exactly one, at least one, an arbitrary number of or at most k (for a $k \geq 1$) parallel rewriting step(s) of a scattered context rule, with the one production of every component of a team remaining exactly the same production in the corresponding scattered context rule. Every scattered context rule corresponds to a team and the amount

of scattered context rules thus equals the amount of teams. Moreover, every production in G' as well as in Γ does not have to be applied, as long as it cannot be applied to the sentential form.

Note that the proof requires the unordered character of the scattered context grammar, for a component of a team may rewrite any occurrence of the left-hand side of its production in the current sentential form. Since a scattered context grammar has to be able to simulate the use of a team by applying the corresponding scattered context rule several times, the modes of derivation for which the statement holds are restricted to the cases mentioned above. Clearly, $L(\Gamma) = L(G')$ and the lemma is proved for the case with as well as the case without λ-productions. □

The main result of this section is presented in the following theorem.

Theorem 6 *For* $m \geq 1$, $x \in \{s, *\}$, $f \in \{=1, \geq 1, *\} \cup \{\leq k \mid k \geq 1\}$ *and* $g \in \{*\} \cup \{\leq k, = k, \geq k \mid k \geq 1\}$

$$U_m SC_{ut} = P_m T_w CD_1(f) \not\subseteq PT_x CD(g) \text{ and}$$
$$U_m SC^{\lambda}_{ut} = P_m T_w CD^{\lambda}_1(f) \not\subseteq PT_x CD^{\lambda}(g).$$

Proof The equalities follow from Lemma 13 and 14. In [Fer 95] it was proved that $PR_{ut} \not\subseteq PR$ and $PR^{\lambda}_{ut} \not\subseteq PR^{\lambda}$, hence with some equalities stated in the Prerequisites and Theorem 4 one obtains the proof of this theorem. □

Since $ET0L \subset USC_{ut}$ and $CD(t) = ET0L$ (see the Prerequisites), a prescribed team CD grammar system with only one production per component, operating in the weak rewriting step and one of the derivation modes f of Theorem 6 can already generate more than a CD grammar system working in mode t can. The inclusion $PR_{ut} \subset PR^{\lambda}_{ut}$, which was proved in [Fer 95], leads to a corollary from Theorem 6 worth mentioning.

Corollary 7 *For* $f \in \{=1, \geq 1, *\} \cup \{\leq k \mid k \geq 1\}$

$$PT_w CD_1(f) \subset PT_w CD^{\lambda}_1(f).$$

2.2 Hybrid (prescribed) teams of grammars: the context-free case

It is natural to ask whether results similar to those that were stated in Section 2.1, can be obtained for the new definitions concerning hybrid teams of grammars that appeared in Section 1.2. Indeed, some similar results for the hybrid cases will

be proved below, but some open problems remain. All results in this section are new.

To begin with, some results on prescribed hybrid team CD grammar systems are presented. The next corollaries immediately follow from results in the previous section.

Corollary 8 *For* $s \geq 2$

$$PR_{ac} \subseteq HPT_sCD \text{ and } PR_{ac}^{\lambda} \subseteq HPT_sCD^{\lambda}.$$

Corollary 9 $PR_{ac} \subseteq HPT_+CD$ *and* $PR_{ac}^{\lambda} \subseteq HPT_+CD^{\lambda}$.

Some results concerning the heterogeneous case of teams in grammar systems are presented next.

Lemma 15 $HPT_*CD \subseteq MAT_{ac}$ *and* $HPT_*CD^{\lambda} \subseteq MAT_{ac}^{\lambda}$.

Proof Consider the hybrid prescribed team CD grammar system

$$\Gamma = (N, T, S, P_1, P_2, \ldots, P_n, (Q_1, f_1), (Q_2, f_2), \ldots, (Q_m, f_m)).$$

Define the homomorphism h from $(N \cup T)^*$ into $(\{A' \mid A \in N\} \cup T)^*$ by

$$h(a) = a \text{ for } a \in T \text{ and } h(A) = A' \text{ for } A \in N.$$

Moreover, associate to a team $Q_i = \{P_{i_1}, P_{i_2}, \ldots, P_{i_{s_i}}\}$, $1 \leq i \leq m$, all sequences of productions such that from each component P_{i_j}, $1 \leq j \leq s_i$, exactly one production is included in such a sequence. Denote such a sequence by $\sigma = (A_1 \rightarrow x_1, \ldots, A_s \rightarrow x_s)$ and all such sequences associated to a team Q_i by $Seq_i = \{\sigma_{i_1}, \sigma_{i_2}, \ldots, \sigma_{i_{i_j}}\}$, $1 \leq i \leq m$.

To simulate this hybrid CD grammar system with prescribed teams, construct the following matrix grammar

$$G' = (N', T', S', M', F'),$$

where

$$
\begin{aligned}
N' &= N \cup \{A' \mid A \in N\} \cup \{T, F\} \cup \{\Sigma_{i_j}, \Sigma'_{i_j} \mid 1 \le j \le l_i, 1 \le i \le m\} \cup \\
&\quad \{[Q_i, f_i, j] \mid (Q_i, f_i) \in \Gamma, f_i \in \{\le k, = k, \ge k\}, 1 \le i \le m, 0 \le j \le k\} \cup \\
&\quad \{[Q_i, g_i], [Q_i, t_0]' \mid (Q_i, g_i) \in \Gamma, g_i \in \{*, t_0, t_1, t_2\}, 1 \le i \le m\}, \\
T' &= T \cup \{z\}, \\
M' &= \{(S' \to ST)\} \cup \\
&\quad \{(T \to [Q_i, f_i, 0]) \mid f_i \in \{\le k, = k, \ge k\}, 1 \le i \le m\} \cup \\
&\quad \{(T \to [Q_i, g_i]) \mid g_i \in \{*, t_0, t_1, t_2\}, 1 \le i \le m\} \cup \\
&\quad \{([Q_i, f_i, j] \to [Q_i, f_i, j+1], A_1 \to h(x_1), A_2 \to h(x_2), \ldots, A_s \to h(x_s)) \mid \\
&\qquad 0 \le j \le k-1, (Q_i, f_i) = \{P_{j_1}, P_{j_2}, \ldots, P_{j_s}\}, A_r \to x_r \in P_{j_r}, \\
&\qquad f_i \in \{\le k, = k, \ge k\}, 1 \le i \le m, 1 \le r \le s\} \cup \\
&\quad \{([Q_i, \ge k, k] \to [Q_i, \ge k, k], A_1 \to h(x_1), A_2 \to h(x_2), \ldots, A_s \to h(x_s)) \mid \\
&\qquad (Q_i, \ge k) = \{P_{j_1}, P_{j_2}, \ldots, P_{j_s}\}, A_r \to x_r \in P_{j_r}, 1 \le i \le m, 1 \le r \le s\} \cup \\
&\quad \{([Q_i, g_i] \to [Q_i, g_i], A_1 \to h(x_1), A_2 \to h(x_2), \ldots, A_s \to h(x_s)) \mid \\
&\qquad (Q_i, g_i) = \{P_{j_1}, P_{j_2}, \ldots, P_{j_s}\}, A_r \to x_r \in P_{j_r}, g_i \in \{*, t_0, t_1, t_2\}, \\
&\qquad 1 \le i \le m, 1 \le r \le s\} \cup \\
&\quad \{([Q_i, t_0] \to \Sigma_{i_1}) \mid 1 \le i \le m\} \cup \\
&\quad \{(\Sigma_{i_j} \to \Sigma'_{i_{j+1}}, A_1 \to \varphi_1, A_2 \to \varphi_2, \ldots, A_s \to \varphi_s) \mid \\
&\qquad \sigma_{i_j} = (A_1 \to x_1, \ldots, A_s \to x_s), \varphi_r \in \{A'_r, F\}, \varphi_r = F \text{ must hold for} \\
&\qquad \text{at least one } r, 1 \le r \le s, 1 \le j \le l_i - 1, 1 \le i \le m\} \cup \\
&\quad \{(\Sigma'_{i_{l_i}} \to [Q_i, t_0]', A_1 \to \varphi_1, A_2 \to \varphi_2, \ldots, A_s \to \varphi_s) \mid \\
&\qquad \sigma_{i_{l_i}} = (A_1 \to x_1, \ldots, A_s \to x_s), \varphi_r \in \{A'_r, F\}, \varphi_r = F \text{ must hold for} \\
&\qquad \text{at least one } r, 1 \le r \le s, 1 \le i \le m\} \cup \\
&\quad \{(\Sigma'_{i_j} \to \Sigma_{i_j}, A'_1 \to F, A'_2 \to F, \ldots, A'_k \to F) \mid \\
&\qquad \{A_1, A_2, \ldots, A_k\} = N, 1 \le j \le l_i, 1 \le i \le m\} \cup \\
&\quad \{(A' \to A) \mid A \in N\} \cup \\
&\quad \{([Q_i, \le k, j] \to T), ([Q_i, = k, k] \to T), ([Q_i, \ge k, k] \to T), ([Q_i, *] \to T) \mid \\
&\qquad 1 \le i \le m, 0 \le j \le k\} \cup \\
&\quad \{([Q_i, t_0]' \to T, A'_1 \to F, A'_2 \to F, \ldots, A'_k \to F) \mid \\
&\qquad \{A_1, A_2, \ldots, A_k\} = N, 1 \le i \le m\} \cup \\
&\quad \{([Q_i, t_1] \to T, A_1 \to F, A'_1 \to F, A_2 \to F, A'_2 \to F, \ldots, A'_r \to F) \mid \\
&\qquad \{A_1, A_2, \ldots, A_r\} = \bigcup_{P_j \in (Q_i, t_1)} dom(P_j), 1 \le i \le m\} \cup \\
&\quad \{([Q_i, t_2] \to T, A_1 \to F, A'_1 \to F, A_2 \to F, A'_2 \to F, \ldots, A'_r \to F) \mid \\
&\qquad \{A_1, A_2, \ldots, A_r\} = dom(P_j) \text{ for some } P_j \in (Q_i, t_2), 1 \le i \le m\} \cup \\
&\quad \{(T \to z)\} \text{ and}
\end{aligned}
$$

in F' are all the productions $A \to F$ appearing in M'.

The simulation of Γ starts with introducing the sentential form ST, in which S is the start-symbol of Γ and T is a marker. The marker will control the derivation and S will generate the language of the hybrid CD grammar system with prescribed teams. This marker is non-deterministically replaced by a control symbol of the form $[Q_i, f_i, j]$ or $[Q_i, g_i]$. In these nonterminals, Q_i is the team operating in mode f_i or g_i and j is a counter, necessary for the modes $f_i \in \{\leq k, = k, \geq k\}$. With teams operating in mode $g_i \in \{*, t_0, t_1, t_2\}$ we do not need to count and the third component is omitted.

When the marker $[Q_i, f_i, j]$ ($[Q_i, g_i]$) is present in the sentential form a simulation by Q_i in mode f_i (g_i) is simulated. The homomorphism h priming all nonterminals in the matrices is necessary to guarantee that the productions are applied to nonterminals that were already existing in the sentential form before these matrices were applied and not to those introduced by a production from these matrices themselves. The counter in the case of modes $\leq k$, $= k$ and $\geq k$ guarantees that a team rewrites the sentential form less than k, exactly k or at least k times, respectively. In case of mode $*$, t_0, t_1 and t_2 there is no counting at all.

In case of t_1 and t_2, however, the productions in the set F guarantee that a team does not stop rewriting until no more component or at least one component of the team can no longer be used, respectively. Finally, in mode t_0 the symbol $[Q_i, t_0]$ can be replaced only by Σ_{i_1}. This symbol can then be replaced by $\Sigma'_{i_{j+1}}$ and back to $\Sigma_{i_{j+1}}$ until $\Sigma'_{i_{k_i}}$ is reached. In this way the correct termination of Q_i in mode t_0 is checked, by the following restrictions.

Firstly, Σ_{i_j} can only be replaced by $\Sigma'_{i_{j+1}}$ if the corresponding sequence of productions indeed cannot be used anymore. An F is introduced otherwise, since each sequence must have at least one $\varphi_r = F$. Secondly, $\Sigma'_{i_{j+1}}$ is allowed to be replaced by $\Sigma_{i_{j+1}}$ only after all primed symbols have been replaced by their originals. Finally, $\Sigma'_{i_{k_i}}$ can only be replaced by $[Q_i, t_0]'$ after indeed none of the sequences Σ_{i_j}, $1 \leq j \leq l_i$, can be used and then eventually be replaced by T.

In every case, afterwards the primes are removed and another team can non-deterministically take the marker spot and start its simulation in its mode. Eventually a terminal string results from S followed by the marker T. This marker is then replaced by z thus yielding $L(G') = L(\Gamma)\{z\}$. This symbol z can be removed by a morphism and thus, since it is known from [DP 89] that the family MAT_{ac} is closed under restricted morphisms, $L(\Gamma) \in MAT_{ac}$ and the first statement of the lemma is proved.

$HPT_*CD^{\lambda} \subseteq MAT^{\lambda}_{ac}$ can be proved directly by a similar construction, even simplified since the marker can eventually be replaced by λ, making the use of a morphism unnecessary. □

It is known that $PR_{ac} = MAT_{ac}$ and $PR^{\lambda}_{ac} = MAT^{\lambda}_{ac}$ (see the Prerequisites), hence the following corollary follows directly from Lemma 15.

Corollary 10 *$HPT_*CD \subseteq PR_{ac}$ and $HPT_*CD^\lambda \subseteq PR_{ac}^\lambda$.*

These results for hybrid CD grammar systems with prescribed teams immediately lead to a result for hybrid CD grammar systems with automatically formed teams, presented next. It is a direct consequence of combining the known equalities concerning programmed grammars and matrix grammars (see the Prerequisites) with Lemma 2 and 15.

Corollary 11 *For $s \geq 1$*

$$HT_1CD \subseteq HT_*CD \subseteq PR_{ac} \text{ and } HT_1CD^\lambda \subseteq HT_*CD^\lambda \subseteq PR_{ac}^\lambda.$$

Combining these lemmas and corollaries concerning the new definitions, the main result of this section is obtained.

Theorem 7 *For $s \geq 2$*

$$HT_*CD \subseteq HPT_*CD = HPT_sCD = PR_{ac} \text{ and}$$
$$HT_*CD^\lambda \subseteq HPT_*CD^\lambda = HPT_sCD^\lambda = PR_{ac}^\lambda.$$

2.3 (Hybrid) prescribed teams of grammars: other cases

In the previous section, results concerning hybrid (prescribed) team CD grammar systems with context-free productions were presented. In this section some results concerning a restriction to regular, linear or metalinear types of productions will be presented and not only for the hybrid case. All results in this section are new.

Recall the fact that, whether free or prescribed, teams with constant size are allowed to have a string axiom, whereas teams of variable size always have a single start symbol.

2.3.1 The regular and the linear case

First, a result for the regular case of prescribed team CD grammar systems with constant team-size 1 is presented.

Lemma 16 *For $f \in \{*, t\} \cup \{\leq k, = k, \geq k \mid k \geq 1\}$ and $g \in \{= k, \geq k \mid k \geq 2\} \cup \{t_0, t_1, t_2\}$*

$$REG = CD_{REG}(f) \subset PT_1CD_{REG}(g).$$

Proof The equality is proved in [CDKP 94a]. Moreover, it is obvious that $CD_{REG}(f) \subseteq PT_1CD_{REG}(g)$ for $f \in \{*, t\} \cup \{\leq k, = k, \geq k \mid k \geq 1\}$ and $g \in \{*, t_0, t_1, t_2\} \cup \{\leq k, = k, \geq k \mid k \geq 1\}$. Furthermore, the prescribed team CD grammar system, with teams of constant size 1,

$$\Gamma_{14} = (\{A_0, A_0', A_1, A_1', \ldots, A_{k-2}', B, B'\}, \{a, b\}, AB, P_1, P_2, P_3, \{P_1\}, \{P_2\}, \{P_3\}),$$

where

$$\begin{aligned} P_1 &= \{A_0 \to A_1, A_1 \to A_2, \ldots, A_{k-2} \to aA_0', B \to bB'\}, \\ P_2 &= \{A_0' \to A_1', A_1' \to A_2', \ldots, A_{k-2}' \to A_0, B' \to B\} \text{ and} \\ P_3 &= \{A \to A_1, A_1 \to A_2, \ldots, A_{k-2} \to a, B \to b\}. \end{aligned}$$

contains only regular productions and it generates $L_f(\Gamma_{14}, 1) = \{a^n b^n \mid n \geq 1\} \in PT_1CD_{REG}(f) \setminus REG$ for $f \in \{= k, \geq k \mid k \geq 2\} \cup \{t_0, t_1, t_2\}$. □

Hence already a prescribed team CD grammar system with only regular productions and teams of size 1 can generate more than the class of regular languages and more than a CD grammar system with only regular productions. The next lemma states that also in the linear case the prescribed team CD grammar systems with teams of any constant size can generate more than the class of linear languages as well as more than CD grammar systems with only linear productions.

Lemma 17 *For* $f \in \{*, t\} \cup \{\leq k, = k, \geq k \mid k \geq 1\}$, $g \in \{= k, \geq k \mid k \geq 2\} \cup \{t_0, t_1, t_2\}$ *and* $g' \in \{*, t_0, t_1, t_2\} \cup \{\leq k, = k, \geq k \mid k \geq 1\}$

$$LIN = CD_{LIN}(f) \subset PT_1CD_{LIN}(g) \subseteq PT_1CD(g') = CD(f).$$

Proof The first equality can be proved with a similar proof as for the regular case (see Lemma 16) and $CD_{LIN}(f) \subseteq PT_1CD_{LIN}(g)$ is obvious, for $f \in \{*, t\} \cup \{\leq k, = k, \geq k \mid k \geq 1\}$ and $g \in \{*, t_0, t_1, t_2\} \cup \{\leq k, = k, \geq k \mid k \geq 1\}$. Furthermore, the prescribed team CD grammar system, with teams of constant size 1,

$$\Gamma_{15} = (\{A_0, A_0', A_1, A_1', \ldots, A_{k-2}', B, B'\}, \{a, b, c\}, AB, P_1, P_2, P_3, \{P_1\}, \{P_2\}, \{P_3\}),$$

where

$$\begin{aligned} P_1 &= \{A_0 \to A_1, A_1 \to A_2, \ldots, A_{k-2} \to aA_0'b, B \to cB'\}, \\ P_2 &= \{A_0' \to A_1', A_1' \to A_2', \ldots, A_{k-2}' \to A_0, B' \to B\} \text{ and} \\ P_3 &= \{A \to A_1, A_1 \to A_2, \ldots, A_{k-2} \to ab, B \to c\}. \end{aligned}$$

contains only linear productions and it generates $L_f(\Gamma_{15}, 1) = \{a^n b^n c^n \mid n \geq 1\} \in PT_1CD_{LIN}(f) \setminus LIN$ for $f \in \{=k, \geq k \mid k \geq 2\} \cup \{t_0, t_1, t_2\}$. The last inclusion in the statement of the lemma is obvious and to prove the last equality, only the inclusion $PT_1CD(g') \subseteq CD(f)$ is not obvious. To prove this inclusion, all teams of size 1 become a component of the CD grammar system and a component $\{S \to S, S \to w \mid w \in W\}$, S being the axiom of the CD grammar system and W being the finite set of string axioms of the prescribed team CD grammar system with teams of constant size, is added. The mode of derivation remains the same, except that for t_0, t_1 and t_2 it becomes t. □

These two results lead to the following corollary for prescribed hybrid team CD grammar systems with teams of constant size 1 and only regular or linear productions.

Corollary 12

$$REG = HCD_{REG} \subset HPT_1CD_{REG} \subseteq HPT_1CD_{LIN} \text{ and}$$
$$LIN = HCD_{LIN} \subset HPT_1CD_{LIN} \subseteq HPT_1CD = HCD.$$

Proof The equality $REG = HCD_{REG}$ is proved in [Mit 93], a similar proof can prove this equality for the linear case as well. The inclusions of hybrid CD grammar systems with only regular or linear productions in prescribed hybrid team CD grammar systems with teams of constant size 1 and only regular or linear productions, respectively, are obvious. Moreover, the combination of Lemma 2, 16 and 17 proves their properness. The remaining two inclusions are obvious again and the last equality can be proven with a similar construction as for the proof of $PT_1CD(g') = CD(f)$ in Lemma 17. □

Hence also prescribed hybrid team CD grammar systems with only regular (linear) productions and teams of size 1 can generate more than the class of regular (linear) languages as well as more than hybrid CD grammar systems with only regular (linear) productions can.

In fact, the following holds. To be more precise, denote $(H)P_nT_1CD_X(f)$ for the class of (hybrid) prescribed team CD grammar systems with at most n occurrences of nonterminals in the string axiom, teams of size 1, only components of type X and operating in mode f (omitted in the hybrid case).

Theorem 8 *For $n \geq 1$, $X \in \{REG, LIN\}$ and $f \in \{t_0, t_1, t_2\} \cup \{=k, \geq k \mid k \geq 2\}$*

$$HP_nT_1CD_X = P_nT_1CD_X(f) = SM_X(n) =$$
$$SM_X^{\lambda}(n) = P_nT_1CD_X^{\lambda}(f) = HP_nT_1CD_X^{\lambda}.$$

Proof In [CDKP 94a], so-called extended CD' grammar systems were defined. In the terminology of this thesis, these systems are CD grammar systems with a string axiom. In [DP 92] these extended CD' grammar systems with only regular productions, at most n nonterminals in the string axiom and working in mode $f \in \{t\} \cup \{=k, \geq k \mid k \geq 2\}$ ($E_nCD'_{REG}(f)$) are proved to be equal to the regular simple matrix grammars of degree n.

Clearly, $E_nCD'_{REG}(f) = P_nT_1CD_{REG}(f)$ for $f \in \{*\} \cup \{\leq k, =k, \geq k \mid k \geq 1\}$ and $E_nCD'_{REG}(t) = P_nT_1CD_{REG}(g)$ for $g \in \{t_0, t_1, t_2\}$. When observing the proof, it can be seen that it holds for the linear case as well. Moreover, the construction can easily be modified to hold for the hybrid case as well. (One just has to code all nonterminals in a similar way as this was done in the proof of Lemma 15, thus indicating which mode is currently being simulated.) Finally, $SM_X^{\lambda}(n) = SM_X(n)$ for $X \in \{REG, LIN\}$ was proved in [Păun 81] and the proof can thus be seen to hold for both the case of forbidding as well as for the case of allowing λ-productions. □

This theorem has some interesting corollaries, since the families of regular and linear simple matrix grammars are well-investigated. A survey of simple matrix grammars can be found in [DP 89], where the proofs of the results corresponding to the coming corollaries can be found.

Corollary 13 *For $n \geq 1$ and $f \in \{t_0, t_1, t_2\} \cup \{=k, \geq k \mid k \geq 2\}$*

$$P_nT_1CD_{REG}(f) = HP_nT_1CD_{REG} \subset P_nT_1CD_{LIN}(f) = HP_nT_1CD_{LIN}.$$

Corollary 14 *The number of nonterminal occurrences in the axioms of prescribed team CD grammar systems, with teams of size 1 and only regular or only linear productions, defines an infinite hierarchy of languages generated in all modes $f \in \{t_0, t_1, t_2\} \cup \{=k, \geq k \mid k \geq 2\}$. The same holds for the hybrid versions of these families of languages.*

Corollary 15 *For $s \geq 1$ and $f \in \{t_0, t_1, t_2\} \cup \{=k, \geq k \mid k \geq 2\}$*

$(H)PT_sCD_{REG}(f)$ is incomparable with LIN and
$(H)PT_sCD_{LIN}(f)$ is incomparable with CF.

The question is now what can be said about the generative power of prescribed (hybrid) team CD grammar systems with only regular or linear productions and teams of constant size s, for $s \geq 2$. From Theorem 1, 2, 4 and 7, a comparison with the programmed (or matrix) grammars with only regular or linear productions seems natural. These, however, are equal to the classes of regular and linear languages, respectively, even when appearance checking is used. The proofs for

these equalities in the regular case can be found in [DP 89] and the proofs for the linear case can be proved similarly. Hence the inclusions $PR_{REG} = PR_{REG,ac} \subset PT_1CD_{REG}(f)$ and $PR_{LIN} = PR_{LIN,ac} \subset PT_1CD_{REG}(f)$ are obvious.

Moreover, the equalities between the programmed grammars and matrix grammars hold in the regular and linear case as well, even with appearance checking. Again, proofs of these equalities can be found in [DP 89]. Note that these proper inclusions hold already for teams of constant size 1. Thus to find a good comparison for the generative power of prescribed (hybrid) team CD grammar systems with only regular or linear productions and teams of constant size s, $s \geq 2$, remains an open problem. It is formulated in Section 3.

After these results for prescribed (hybrid) team CD grammar systems with teams of constant size, some results for the case of teams of variable size are presented next. The difference is the use of a single nonterminal as axiom in the case of teams of variable size, whereas in the case of teams with constant size a finite set of string axioms, with only one of them containing nonterminals, is used.

Lemma 18 *For* $f \in \{*, t_0, t_1, t_2\} \cup \{\leq k, = k, \geq k \mid k \geq 1\}$

$$HPT_*CD_{REG} = PT_*CD_{REG}(f) = REG \text{ and}$$
$$HPT_*CD_{LIN} = PT_*CD_{LIN}(f) = LIN.$$

Proof For teams with more than one component at least two nonterminals must be present in a sentential form, in order to use that team to rewrite that sentential form. This is in contradiction with the facts that every $\Gamma \in \{HPT_*CD_{REG}, PT_*CD_{REG}(f), HPT_*CD_{LIN}, PT_*CD_{LIN}(f) \mid f \in \{*, t_0, t_1, t_2\} \cup \{\leq k, = k, \geq k \mid k \geq 1\}\}$ has a single nonterminal as axiom and regular or linear productions, respectively. For teams of size one, the equality with (hybrid) CD grammar systems is obvious for the regular case as well as for the linear case, keeping in mind the use of a single nonterminal as axiom. From [CDKP 94a] ([Mit 93]) it is known that (hybrid) CD grammar systems with only regular productions do not generate more than the class of regular languages and similar proofs can be used to prove these results for the linear case as well. □

2.3.2 The metalinear case

From the proof of the last lemma in the previous section, it may be expected that the (hybrid) prescribed team CD grammar systems with teams of variable size and metalinear productions are able to generate languages beyond the class of regular or linear languages. Indeed, the following lemma holds.

Lemma 19 *For* $f \in \{*, t\} \cup \{\leq k, = k, \geq k \mid k \geq 1\}$ *and* $g \in \{= 1, \geq 1, *, t_0, t_1, t_2\} \cup \{\leq k \mid k \geq 1\}\}$

$$MLIN = CD_{MLIN}(f) = HCD_{MLIN} \subset PT_*CD_{MLIN}(g).$$

Proof The first two equalities can be proved by the proofs of $REG = CD_{REG}(f)$ ([CDKP 94a]) and $REG = HCD_{REG}$ ([Mit 93]), with the obvious modifications. Furthermore, it is clear that $MLIN \subseteq PT_*CD_{MLIN}(f)$, for $f \in \{*, t_0, t_1, t_2\} \cup \{\leq k, = k, \geq k \mid k \geq 1\}\}$. Moreover, the prescribed team CD grammar system

$$\Gamma_{17} = (\{S, A, B, C\}, \{a, b, c\}, S, P_1, P_2, \ldots, P_7, \{P_1\}, \{P_2, P_3, P_4\}, \{P_5, P_6, P_7\}),$$

with teams of variable size and the metalinear productions

$$\begin{array}{lll} P_1 = \{S \to ABC\}, & P_2 = \{A \to aA\}, & P_5 = \{A \to a\}, \\ & P_3 = \{B \to bB\}, & P_6 = \{B \to b\}, \\ & P_4 = \{C \to cC\} \text{ and} & P_7 = \{C \to c\}. \end{array}$$

generates $L(\Gamma_{17}) = \{a^n b^n c^n \mid n \geq 1\} \in PT_*CD_{MLIN}(f) \setminus CF$, for $f \in \{= 1, \geq 1, *, t_0, t_1, t_2\} \cup \{\leq k \mid k \geq 1\}\}$. Since it is known (see e.g. [Sal 73]) that $\{a^n b^n c^n \mid n \geq 1\} \in CS \setminus CF$ and from the Chomsky hierarchy (see the Prerequisites) that $MLIN \subset CF \subset CS$, the inclusion is proper indeed. □

Hence even languages beyond the class of metalinear languages can be generated already by a prescribed team CD grammar system with teams of variable size and metalinear productions for some modes of derivation. For prescribed team CD grammar systems with teams of constant size, no version containing only metalinear productions is defined due to the string axiom they already possess. Note, however, that the lemma above does not cover all modes of derivation, which Lemma 22 below will.

The following theorem is obtained by combining the results of the previous section and the results obtained so far in this section.

Theorem 9 *For* $f \in \{*, t_0, t_1, t_2\} \cup \{\leq k, = k, \geq k \mid k \geq 1\}$ *and* $g \in \{= 1, \geq 1, *, t_0, t_1, t_2\} \cup \{\leq k \mid k \geq 1\}$

$$\begin{aligned} PT_*CD_{REG}(f) = HPT_*CD_{REG} \subset PT_*CD_{LIN}(f) = HPT_*CD_{LIN} \subset \\ MLIN \subset PT_*CD_{MLIN}(g) \subseteq HPT_*CD_{MLIN}. \end{aligned}$$

The question is now how far these (hybrid) prescribed team CD grammar systems with teams of variable size and metalinear productions extend beyond the class of metalinear languages. Before presenting a theorem that answers this question, two lemmas are needed.

The proofs of these lemmas are given because the metalinear case is not covered in [DP 89]. Moreover, since the proofs are based on the proofs for the context-free case in [DP 89], they explain the techniques that are used to prove those frequently used results of the next lemmas in the context-free case.

Lemma 20

$$USC_{MLIN} \subseteq PR_{MLIN},\ USC^{\lambda}_{MLIN} \subseteq PR^{\lambda}_{MLIN},$$
$$USC_{MLIN,ac} \subseteq PR_{MLIN,ac}\ and\ USC^{\lambda}_{MLIN,ac} \subseteq PR^{\lambda}_{MLIN,ac}.$$

Proof Only the third inclusion is proved here, the others can be proved in a similar way. Consider an unordered scattered context grammar

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

with appearance checking and only metalinear productions. Define the homomorphism h from $(N \cup T)^*$ into $(\{A' \mid A \in N\} \cup T)^*$ by

$$h(a) = a \text{ for } a \in T \text{ and } h(A) = A' \text{ for } A \in N.$$

Next, for a rule $r : (\alpha_1, \alpha_2, \ldots, \alpha_n) \to (\beta_1, \beta_2, \ldots, \beta_n) \in P$, denote $h(\beta_1\beta_2 \ldots \beta_n) = w_1B'_1w_2B'_2 \ldots w_mB'_mw_{m+1}$ with $w_i \in T^*$ for $1 \leq i \leq m+1$. To simulate this scattered context grammar with appearance checking, construct the programmed grammar with appearance checking

$$G' = (N', T', S', P'),$$

where

$N' = N \cup \{A' \mid A \in N\} \cup \{S'\}$,

$T' = T$ and

P' contains for S' the starting productions $(s : S' \to S, \{[r, 1, 0] \mid r \in P\}, \emptyset)$
and for every production $r : (\alpha_1, \alpha_2, \ldots, \alpha_n) \to (\beta_1, \beta_2, \ldots, \beta_n)$
in P the productions
$([r, i, 0] : \alpha_i \to h(\beta_i), \{[r, i+1, 0]\}, \{\emptyset \mid \alpha_i \to \beta_i \notin F\} \cup \{[r, i+1, 0] \mid \alpha_i \to \beta_i \in F\})$,
$([r, n, 0] : \alpha_n \to h(\beta_n), \{[r, 1, 1]\}, \{\emptyset \mid \alpha_n \to \beta_n \notin F\} \cup \{[r, 1, 1] \mid \alpha_n \to \beta_n \in F\})$,
$([r, j, 1] : B'_j \to B_j), \{[r, j+1, 1]\}, \{[r, j+1, 1]\})$ and
$([r, m, 1] : B'_m \to B_m), \{[p, 1, 0] \mid p \in P\}, \{[p, 1, 0] \mid p \in P\})$
for $1 \leq i \leq n$ and $1 \leq j \leq m$.

Since the scattered context grammar contains only metalinear productions, it is clear that also the productions in this programmed grammar are all metalinear. Moreover, the productions in the programmed grammar simulating the productions in a scattered context rule are applied in a fixed order, possibly passing over a production in case it is contained in F. The use of primes guarantees that the simulating productions are applied only to nonterminals already appearing in the sentential form to be rewritten and not to the ones introduced by a former production of the scattered context rule that is being simulated.

This allows the parallel fashion of a scattered context rule to be simulated by the sequential order of programmed grammar productions. Note that the proof requires the unordered characteristic of the scattered context grammar, for a production $\alpha \to \beta$ can rewrite any occurrence of α in the current sentential form. Obviously, $L(G) = L(G')$ and thus $USC_{MLIN,ac} \subseteq PR_{MLIN,ac}$ holds. □

Lemma 21

$$MAT_{MLIN} \subseteq USC_{MLIN},\ MAT^{\lambda}_{MLIN} \subseteq USC^{\lambda}_{MLIN},$$
$$MAT_{MLIN,ac} \subseteq USC_{MLIN,ac}\ and\ MAT^{\lambda}_{MLIN,ac} \subseteq USC^{\lambda}_{MLIN,ac}.$$

Proof Again, only the third inclusion is proved, the others can be proved in a similar way. Consider a matrix grammar

$$G = (N, T, S, M, F)$$

with appearance checking and only metalinear productions. Denote

$$Lab(M) = \{m_{i,j} \mid m_i : (\alpha_1 \to \beta_1, \alpha_2 \to \beta_2, \ldots, \alpha_n \to \beta_n) \in M, \\ M = \{m_1, m_2, \ldots, m_m\}, 1 \leq i \leq m, 1 \leq j \leq n\}.$$

To simulate this matrix grammar with appearance checking, construct the unordered scattered context grammar with appearance checking

$$G' = (N', T', S', P', F'),$$

where

$$
\begin{array}{lll}
N' & = & N \cup \{[\alpha,\beta] \mid \alpha \in (N \cup T), \beta \in Lab(M)\} \cup \{S'\}, \\
T' & = & T, \\
P' & \text{contains} & \text{for } S' \text{ the starting rules } (S') \to ([S, m_{i,1}]) \text{ for } 1 \leq i \leq n \text{ and} \\
 & & \text{for every matrix (with only metalinear productions)} \\
 & & m_i : (\alpha_1 \to \beta_1\gamma_1, \alpha_2 \to \beta_2\gamma_2, \ldots, \alpha_n \to \beta_n\gamma_n) \in M, \\
 & & \beta_j \in (N \cup T),\ \gamma_j \in (N \cup T)^* \text{ and } 1 \leq j \leq n \\
 & & \text{the scattered context rules} \\
 & & ([\alpha_j, m_{i,j}]) \to ([\beta_j, m_{i,j+1}]\gamma_j), \\
 & & ([\alpha_n, m_{i,n}]) \to ([\beta_n, m_{k,1}]\gamma_n), \\
 & & ([\delta, m_{i,j}], \alpha_j) \to ([\delta, m_{i,j+1}], \beta_j\gamma_j), \\
 & & ([\delta, m_{i,n}], \alpha_n) \to ([\delta, m_{k,1}], \beta_n\gamma_n) \text{ and} \\
 & & ([\tau, m_{i,1}]) \to (\tau) \\
 & & \text{for } 1 \leq i, k \leq m,\ \delta \in (N \cup T) \text{ and } \tau \in T^* \text{ and} \\
F' & = & F.
\end{array}
$$

The matrices can be simulated by unordered scattered context rules by adding a label to every symbol of the alphabet. The matrices are split and for every production of it some scattered context rules are created. The labels define an ordering on the use of the various scattered context rules, thus simulating the strict order of matrices by unordered scattered context rules.

At any moment in time, the number of symbols with a label in the sentential form is zero or one. This can be seen from the definitions. If the number is zero, either the sentential form is a terminal one and the derivation is terminated or the sentential form contains a nonterminal and the derivation is blocked since every rule requires a labeled symbol (except the initial rules). If the number is one, it can be replaced by another labeled symbol (the label being the one of the next production in the matrix or the one of the first production of a new matrix if it was the last production of the matrix) while rewriting the symbol according to the production of a matrix being simulated.

Naturally, a production of a scattered context rule can be "passed over" if the same production could be passed over in the matrix grammar, in which case the other production of the scattered context rule replaces the label by the label from the next production in the matrix or the one from the first production of a new matrix if this was the last production of the matrix. Naturally, terminating rules eliminating the labels are present, to be used only when a matrix has been completely simulated. It can now be seen that $L(G) = L(G')$ and $MAT_{MLIN,ac} \subseteq USC_{MLIN,ac}$ holds. □

Theorem 10 *For* $f \in \{*\} \cup \{\leq k, = k, \geq k \mid k \geq 1\}$ *and* $g \in \{t_0, t_1, t_2\}$

$$USC_{MLIN} = PR_{MLIN} = MAT_{MLIN} = PT_*CD_{MLIN}(f) \subseteq HPT_*CD_{MLIN},$$
$$USC_{MLIN,ac} = PR_{MLIN,ac} = MAT_{MLIN,ac} = PT_*CD_{MLIN}(g) = HPT_*CD_{MLIN},$$

$$USC^{\lambda}_{MLIN} = PR^{\lambda}_{MLIN} = MAT^{\lambda}_{MLIN} = PT_*CD^{\lambda}_{MLIN}(f) \subseteq HPT_*CD^{\lambda}_{MLIN} \text{ and}$$
$$USC^{\lambda}_{MLIN,ac} = PR^{\lambda}_{MLIN,ac} = MAT^{\lambda}_{MLIN,ac} = PT_*CD^{\lambda}_{MLIN}(g) = HPT_*CD^{\lambda}_{MLIN}.$$

Proof It is easy to see from their proofs that Lemma 4 and 5 hold in the case of only metalinear productions as well. Likewise, also Lemma 11 holds in the metalinear case too. Finally, also Lemma 12 and 15 continue to hold in the case of a restriction to metalinear productions. The theorem is hence a simple combination of Lemma 20, 11, 2, 12 and 21 for the equalities without appearance checking; the inclusion being from Lemma 2. The equalities with appearance checking of the theorem follow from the combination of Lemma 20, 4 and 5, 2, 15 and 21. □

Remark 2 *Note that this theorem strengthens Theorem 9 to an equality for modes* t_0, t_1 *and* t_2 *instead of the last inclusion there. An open problem remains, however, for the other modes of derivation. It is formulated in Section 3.*

The next lemma strengthens Lemma 19, and thereby Theorem 9, even more.

Lemma 22 *For* $f \in \{*\} \cup \{\leq k, = k, \geq k \mid k \geq 1\}$ *and* $g \in \{t_0, t_1, t_2\}$

$$MLIN \subset USC_{MLIN} = PR_{MLIN} = MAT_{MLIN} = PT_*CD_{MLIN}(f) \subseteq PT_*CD_{MLIN}(g) = HPT_*CD_{MLIN}.$$

Proof The first three equalities are proved in Theorem 10. Moreover, the unordered scattered context grammar

$$G_1 = (\{S, A, B, C\}, \{a, b, c\}, S, \{p_1, p_2, p_3\}),$$

with the rules (consisting of only metalinear productions)

$$\begin{aligned} p_1 &: (S) \to (ABC), \\ p_2 &: (A, B, C) \to (aA, bB, cC) \text{ and} \\ p_3 &: (A, B, C) \to (a, b, c) \end{aligned}$$

generates $L(G_1) = \{a^n b^n c^n \mid n \geq 1\} \in USC_{MLIN} \setminus CF$. Since it is known (see e.g. [Sal 73]) that $\{a^n b^n c^n \mid n \geq 1\} \in CS \setminus CF$ and from the Chomsky hierarchy

(see the Prerequisites) that $MLIN \subset CF \subset CS$, the first inclusion is proper indeed. The last inclusion is obvious, since clearly $USC_{MLIN} \subseteq USC_{MLIN,ac}$ and, according to Theorem 10, $USC_{MLIN,ac} = PT_*CD_{MLIN}(f)$ for $f \in \{t_0, t_1, t_2\}$. Finally, the last equality was proved in Theorem 10. □

Hence already an unordered scattered context, programmed or matrix grammar without appearance checking and with λ-free productions and only metalinear productions can generate languages beyond the class of metalinear languages. Moreover, also a prescribed team CD grammar system (for all modes of derivation) and a hybrid prescribed team CD grammar system, both with teams of variable size and only metalinear productions, can already generate languages beyond this class of metalinear languages.

For matrix and programmed grammars (with appearance checking) and regular, linear, context-sensitive or recursively enumerable productions only, it is known (see e.g. [DP 89]) that these cannot generate more than the class of regular, linear, context-sensitive or recursively enumerable languages, respectively. Hence, no interesting results may be expected for unordered scattered context grammars in these cases either.

The next lemma says something about the relation between prescribed team CD grammar systems with teams of bounded size and exactly 1 metalinear production per component and linear simple matrix grammars of degree n. Because of Theorem 8 and its corollaries, this establishes a relation, presented in Corollary 16 right after the coming proof, with the (hybrid) prescribed team CD grammar systems with teams of size 1 and linear or regular productions only.

To be more precise, denote $PT_{(1,n)}CD_{MLIN,1}(f)$ for the class of prescribed team CD grammar systems with teams of size 1 for the teams containing a production with the axiom as its left-hand side, teams of size n for the teams containing other productions, only 1 production per component (the second 1 in the notation), operating in mode f and containing only metalinear productions.

Lemma 23 *For $n \geq 1$ and $f \in \{=1, \geq 1, *, t_0, t_1, t_2\} \cup \{\leq k \mid k \geq 1\}$*

$$SM_{LIN}(n) \subseteq PT_{(1,n)}CD_{MLIN,1}(f).$$

Proof Consider the linear simple matrix grammar

$$G' = (N'_1, N'_2, \ldots, N'_k, T', S', M')$$

of degree k, $k \geq 1$. To simulate this linear simple matrix grammar, construct the prescribed team CD grammar system

$$\Gamma = (N, T, S, P_1, P_2, \ldots, P_n, Q_1, Q_2, \ldots, Q_m),$$

where

$$
\begin{aligned}
N &= \bigcup_{i=1}^{n} N_i', \\
T &= T', \\
S &= S', \\
P_1, P_2, \ldots, P_n \text{ are } & \text{the components } \{\alpha \to \beta\} \text{ for every } \alpha \to \beta \text{ in matrices} \\
& \text{of } M',\ \alpha \in \{S'\} \cup \bigcup_{i=1}^{n} N_i' \text{ and } \beta \in (\bigcup_{i=1}^{n} N_i' \cup T)^* \text{ and} \\
Q_1, Q_2, \ldots, Q_m \text{ are } & \text{the teams } \{\{S \to \beta\}\} \text{ for every matrix } (S' \to \beta) \in M' \\
& \text{and } \beta \in \bigcup_{i=1}^{n} N_i' \cup T^* \text{ and} \\
& \text{the teams } \{\{A_j \to x_j\}\} \text{ for every matrix of the form} \\
& (A_1 \to x_1, A_2 \to x_2, \ldots, A_k \to x_k) \in M', A_i \in N_i' \\
& x_i \in (N_i' \cup T')^* \text{ and } 1 \leq i, j \leq k.
\end{aligned}
$$

Note that due to the pairwise disjoint alphabets of simple matrix languages a production $A_j \to x_j$, $1 \leq j \leq k$, does not rewrite a nonterminal introduced by a production $A_i \to x_j$, $1 \leq i < j \leq k$, of the same matrix, but a nonterminal already present in the sentential form before applying this particular matrix to it. It is this property of simple matrix grammars that allows the strict sequential order of rewriting of them to be simulated by one parallel rewriting step of a team of a prescribed team CD grammar system.

Do note also that a characteristic of linear simple matrix grammars is that there can never be two of the same nonterminals in any sentential form. Hence leftmost rewriting is equal to free rewriting in linear simple matrix grammars, thus free rewriting in the simulating prescribed team CD grammar system suffices. These notes imply the restriction to the modes of derivation as stated in the lemma. Moreover, the metalinear productions of the prescribed team CD grammar system allow exactly the axiom to be the left-hand side of non-linear productions, which are precisely the only non-linear productions in a linear simple matrix grammar.

It is clear that $L(G') = L(\Gamma)$. Furthermore, it is easy to see that teams with one component are constructed for productions with the axiom as left-hand side and teams of k components, k being the degree of the simple matrix grammar, are constructed for the other productions. Moreover, all these components of the prescribed team CD grammar system contain only one production. □

Compare the following relations with Theorem 9.

Corollary 16 *For* $f \in \{t_0, t_1, t_2\} \cup \{=k, \geq k \mid k \geq 2\}$, $g \in \{*\} \cup \{\leq k, =k, \geq k \mid k \geq 1\}$ *and* $g' \in \{t_0, t_1, t_2\}$

$$PT_1CD_{REG}(f) = HPT_1CD_{REG} \subset PT_1CD_{LIN}(f) = HPT_1CD_{LIN} \subseteq PT_*CD_{MLIN}(g) \subseteq PT_*CD_{MLIN}(g') = HPT_*CD_{MLIN}.$$

Remark 3 *According to Corollary 15, there is a language that can be generated by a (hybrid) prescribed team CD grammar system with teams of size* 1 *and only linear productions, but which cannot be generated by a context-free grammar. Hence the inclusion* $PT_1CD_{LIN}(f) \subseteq PT_*CD_{MLIN}(g)$ *in the above corollary is either proper and* $CF \subseteq PT_*CD_{MLIN}(g)$ *holds or the family of languages generated by prescribed team CD grammar systems with teams of variable size and only metalinear productions is incomparable with the class of context-free languages.*

My conjecture is an incomparability result. An intuition supporting a possible proof is the following. It is clear that the context-free grammar $G_2 = (\{S, A\}, \{a, b\}, S, \{S \to aAbS, S \to aAb, S \to ab, A \to aAb, A \to ab\})$ *generates* $L(G_2) = \{a^n b^n \mid n \geq 1\}^+ \in CF$. *A characteristic of this language is its unknown width and depth, i.e. the number of* $a^n b^n \mid n \geq 1$ *'s next to each other and for each the amount of* n *are unknown. Obviously, metalinear productions can simulate the depth with productions similar to the last four productions in* G_2. *The width, however, has to be known in advance in the case of metalinear productions since the axiom is the only production which can have more than one nonterminal on its right-hand side and should thus introduce a sufficient amount of them. This amount has to be known in advance and the set of productions is finite, hence it seems that* $\{a^n b^n \mid n \geq 1\}^+ \notin (PT_*CD_{MLIN}(f) \cup PT_*CD^{\lambda}_{MLIN}(f))$ *for* $f \in \{*, t_0, t_1, t_2\} \cup \{\leq k, =k, \geq k \mid k \geq 1\}$.

Finally, note that if this conjecture holds, then also the classes of unordered scattered context, matrix and programmed grammars (even with appearance checking) with only metalinear productions are incomparable with the class of context-free languages. For a proof of this, consider for example Theorem 10 and the proof of Lemma 22.

Hence for linear (and regular) simple matrix grammars, a prescribed team CD grammar system with only metalinear productions can be constructed generating the same language. Whether this also holds the other way around and for other modes of derivation, is an open problem formulated in Section 3.

Another open problem stated in that section is the relation between simple matrix grammars with only context-free productions and prescribed team CD grammar systems and hence also matrix grammars with only context-free productions (and appearance checking). The leftmost rewriting of simple matrix grammars makes it unlikely to have a similar relation between them and prescribed team CD grammar systems, though. For matrix grammars with only

metalinear productions, however, the following corollary does present an interesting relation with regular and linear simple matrix grammars.

Corollary 17 *For* $n \geq 1$

$$SM_{REG}(n) = SM^{\lambda}_{REG}(n) \subset SM_{LIN}(n) = SM^{\lambda}_{LIN}(n) \subseteq MAT_{MLIN} \subseteq MAT^{\lambda}_{MLIN}.$$

Proof It is known, see e.g. [DP 89], that $SM_{REG}(n) = SM^{\lambda}_{REG}(n) \subset SM_{LIN}(n) = SM^{\lambda}_{LIN}(n)$ for $n \geq 1$. From Lemma 23 follows $SM_{LIN}(n) \subseteq PT_{(1,n)}CD_{MLIN,1}(f)$ for $n \geq 1$ and $f \in \{=1, \geq 1, *, t_0, t_1, t_2\} \cup \{\leq k \mid k \geq 1\}$. Moreover, it is clear that $PT_{(1,n)}CD_{MLIN,1}(f) \subseteq PT_*CD_{MLIN}(f)$ for all modes of derivation and thus $SM_{LIN}(n) \subseteq PT_*CD_{MLIN}(f)$ is obtained for $n \geq 1$ and $f \in \{=1, \geq 1, *\} \cup \{\leq k \mid k \geq 1\}$. Finally, Theorem 10 finishes the proof of the corollary. □

A very interesting corollary indeed, knowing that $SM_{LIN}(n) \subseteq SM(n)$ for $n \geq 1$ (see e.g. [DP 89]) and keeping in mind the unknown relation between simple matrix grammars and matrix grammars, in the case of context-free productions only, already mentioned above.

2.4 A normal form result: one production per component suffices

Two lemmas from Subsection 2.1.3 lead to some interesting equalities, presented in the next theorem, concerning prescribed team CD grammar systems with only one production per component and the ones without this restriction. To be more precise, denote U_mSC_X for the class of unordered scattered context grammars with m scattered context rules and only productions of type X and denote $P_mT_*CD_{X,1}(f)$ for the class of prescribed team CD grammar system with m teams of variable size, 1 production per component, operating in mode f and containing only productions of type X.

Theorem 11 *For* $m \geq 1$, $X \in \{REG, LIN, MLIN, CF, CS, RE\}$ *and* $f \in \{=1, \geq 1, *\} \cup \{\leq k \mid k \geq 1\}$

$$P_mT_*CD_{X,1}(f) = U_mSC_X \text{ and } P_mT_*CD^{\lambda}_{X,1}(f) = U_mSC^{\lambda}_X.$$

Proof Clearly, the constructions given in the proofs of Lemma 13 and 14 continue to hold when no productions may be skipped under any condition whatsoever. This leads to unordered scattered context grammars and prescribed team

CD grammar systems with strong rewriting, i.e. the usual rewriting. Moreover, it is clear that the proofs hold for all other types of productions than context-free ones as well. □

Remark 4 *As said in the proof directly above, Lemma 13 and 14 continue to hold for other types of productions than context-free ones as well.*

The main result of this section is an interesting corollary of Theorem 11.

Theorem 12 *For* $X \in \{REG, LIN, MLIN, CF\}$ *and* $f \in \{=1, \geq 1, *\} \cup \{\leq k \mid k \geq 1\}$

$$PT_*CD_X(f) = PT_*CD_{X,1}(f) \text{ and } PT_*CD_X^{\lambda}(f) = PT_*CD_{X,1}^{\lambda}(f).$$

Proof The results for the regular and linear case follow from Lemma 18, Theorem 11 and the obvious equalities between (unordered) scattered context grammars, programmed grammars (both with only regular or linear productions and with or without λ-free productions) and regular or linear grammars, respectively. The results for the metalinear case follow from Theorem 10 and 11 and the results for the context-free case follow from Theorem 4 and 11 and some equalities concerning unordered scattered context grammars and programmed grammars, both with context-free productions only and with or without λ-free productions, stated in the Prerequisites. □

Hence when restricted to regular, linear, metalinear or context-free productions only, teams with one production per component suffice for prescribed team CD grammar systems with teams of variable size operating in derivation mode $=1$, ≥ 1, $*$ or $\leq k$ (for a $k \geq 1$).

Remark 5 *Note that* $CD(f) = CF$ *for* $f \in \{=1, \geq 1, *\} \cup \{\leq k \mid k \geq 1\}$ *(see Part II, Section 5.1), though* $CF \subset PT_*CD_1(f)$. *Hence in the context-free case even CD grammar systems with* n *components cannot generate all languages that can be generated by prescribed team CD grammar systems with teams of variable size and only* 1 *production per component, for modes* $f \in \{=1, \geq 1, *\} \cup \{\leq k \mid k \geq 1\}$.

2.5 Controlled grammar systems

In this section, something will be said about the generative power of the definitions in Section 1.3.

To begin with, every (prescribed) (hybrid) team CD grammar system is also a controlled (prescribed) (hybrid) team CD grammar system, formally stated in the next lemma.

Lemma 24 *For $s \geq 1$, $X \in \{REG, LIN, CF, CS, RE\}$, $X' \in \{MLIN\}$, $Y \in \{GC, HL, GSM\}$, $z \in \{s, *\}$ and $f \in \{*, t_0, t_1, t_2\} \cup \{\leq k, =k, \geq k \mid k \geq 1\}$*

(i) $T_sCD_X(f) \subseteq YT_sCD_X(f)$,

(ii) $PT_zCD_X(f) \subseteq YPT_zCD_X(f)$ *and* $PT_*CD_{X'}(f) \subseteq YPT_*CD_{X'}(f)$,

(iii) $HT_1CD_X \subseteq YHT_1CD_X$, $HT_*CD_X \subseteq YHT_*CD_X$ *and* $HT_*CD_{X'} \subseteq YHT_*CD_{X'}$ *and*

(iv) $HPT_zCD_X \subseteq YHPT_zCD_X$ *and* $HPT_*CD_{X'} \subseteq YHPT_*CD_{X'}$.

Proof The inclusions follow easily when the controlling devices impose no restriction. Thus for $Y = GC$, consider the complete graph with n nodes C_n. For $Y = HL$, consider the regular language $R = (N \cup T)^* \setminus T^*$. For $Y = GSM$, finally, consider the gsm that does not translate but only copies the string it receives as input, i.e. $g = (\{s_0\}, N \cup T, N \cup T, s_0, \{(s_0x, xs_0) \mid x \in (N \cup T)\}, \{s_0\})$. □

The following corollary follows immediately from Theorem 1, 2, 4 and 7 and Lemma 24.

Corollary 18 *For $s \geq 1$, $Y \in \{GC, HL, GSM\}$, $f \in \{*\} \cup \{\leq k, =k, \geq k \mid k \geq 1\}$, $g \in \{t_0, t_1, t_2\}$ and $z \in \{s, *\}$*

(i) $PR \subseteq YPT_zCD(f)$ *and* $PR^\lambda \subseteq YPT_zCD^\lambda(f)$,

(ii) $PR_{ac} \subseteq YT_sCD(g)$ *and* $PR^\lambda_{ac} \subseteq YT_sCD^\lambda(g)$,

(iii) $PR_{ac} \subseteq YPT_zCD(g)$ *and* $PR^\lambda_{ac} \subseteq YPT_zCD^\lambda(g)$ *and*

(iv) $PR_{ac} \subseteq YHPT_zCD$ *and* $PR^\lambda_{ac} \subseteq YHPT_zCD^\lambda$.

Clearly, this leaves little left to investigate for these type of controlled grammar systems with only context-free, context-sensitive or recursively enumerable productions. In the case of a restriction to regular, linear or metalinear productions, Lemma 16, 17, 18 and 22 lead to the following corollary.

Corollary 19 *For* $s \geq 1$, $Y \in \{GC, HL, GSM\}$, $f \in \{*, t_0, t_1, t_2\} \cup \{\leq k, = k, \geq k \mid k \geq 1\}$ *and* $g \in \{t_0, t_1, t_2\} \cup \{= k, \geq k \mid k \geq 2\}$

(i) $REG = YPT_*CD_{REG}(f) = YHPT_*CD_{REG} \subset YPT_sCD_{REG}(g)$,

(ii) $LIN = YPT_*CD_{LIN}(f) = YHPT_*CD_{LIN} \subset YPT_sCD_{LIN}(g)$ *and*

(iii) $MLIN \subset YPT_*CD_{MLIN}(f) \subseteq YHPT_*CD_{MLIN}$.

From the above results, it is clear that the team feature itself is so strong that an increase in generative power of controlled team grammar systems cannot be achieved. However, some languages might be easier to produce by controlled grammar systems, perhaps even with a smaller syntactic complexity, than by those without control. Hence a possibility for future research remains.

Concerning subclasses of the definitions of Section 1.3 (as suggested in Remark 1), however, only CD grammar systems with control devices were thoroughly investigated. See e.g. [CD 90], [GP 90], [Mit 90], [Das 91a] and [Das 91b] for more details. Furthermore, in Appendix E, some results concerning hybrid CD grammar systems with the three types of control of Section 1.3 are presented.

2.6 Conclusion

In the previous sections the power of forming teams in grammar systems has been investigated. In many cases this forming of teams was found to increase the generative power of the underlying grammar system. In this section the preceding sections are summarized, the most important results highlighted and their consequences considered.

To begin with, there are derivation modes in which the forming of teams strictly increases the power of CD grammar systems. Whether the same holds in the case of hybrid CD grammar systems is still unknown. The first main result is given in Theorem 1, shortly thereafter followed by Theorem 2 and together resulting in the most general Theorem 3. Theorem 3 says that the forming of (prescribed) teams of constant or variable size, working in either of the three variants of the t-mode, enlarges the power of CD grammar systems to equal that of the family of programmed grammars with appearance checking. Moreover, this power is achieved without any (external) control mechanism.

This theorem also proves the fact that teams of size two suffice, i.e. the number of teams implies no hierarchy, for more variants of team forming than for which it was proved in [CP 93]. Furthermore, it proves the unexpected equivalence between all three variants of the t-mode of derivation for maximal rewriting of teams. This is unexpected in the sense that t_1 seems far more restrictive than

t_2 and t_2 itself seems more restrictive than t_0 again. In fact, the inclusion of prescribed team CD grammar systems working in mode t_1 or t_2 into the ones working in mode t_0 can be proper.

For other modes of derivation than the maximal rewriting strategy, Theorem 4 says that still the power of programmed grammars is reached, only limited to the ones without appearance checking. This results in some proper inclusions for prescribed team CD grammar systems working in different modes of derivation, presented in Theorem 5.

In the case of CD grammar systems with prescribed teams of variable size operating in the weak rewriting step, Theorem 6 says that, for all derivation modes $*$, $\leq k$, $= k$ and $\geq k$ (for a $k \geq 1$), their generative capacity is equal to that of the class of programmed grammars with unconditional transfer. This implies that these families and those of the prescribed team CD grammar systems operating in the strong rewriting step and the same modes of derivation do not coincide.

Surprisingly enough, these two previous theorems also prove that in the case of prescribed teams there is no difference in generative power between the derivation modes $*$, $\leq k$, $= k$ and $\geq k$ (for a $k \geq 1$). This is surprising in the sense that for CD grammar systems $CF = CD(=1) = CD(\geq 1) = CD(*) = CD(\leq k) \subset (CD(=k') \cap CD(\geq k''))$ is known to hold.

Theorem 7 says that the forming of hybrid teams in CD grammar systems does not enlarge the generative power any further, in the case these hybrid teams are prescribed. When the teams are automatically formed from a hybrid CD grammar system, it is not known whether the forming of teams enlarges the power of hybrid CD grammar systems.

For (hybrid) prescribed team CD grammar systems with teams of constant size and only regular productions, the team forming enlarges their generative power beyond the class of regular languages to the class of regular simple matrix grammars. Hence it extends also beyond the power of regular (hybrid) CD grammar systems. The same holds in the case of a restriction to linear productions. These equalities are stated in Theorem 8 and lead to several corollaries, one of these being that (hybrid) prescribed team CD grammar systems with a restriction to regular or linear productions and teams of constant size are incomparable with the class of context-free languages. On the other hand, (hybrid) (team) CD grammar systems with context-free productions include that class, whereas for the metalinear case, incomparability is only conjectured.

In the case of teams of variable size, no more than the class of regular or linear languages can be generated by (hybrid) prescribed team CD grammar systems with only regular or linear productions, respectively. However, when restricted to metalinear productions, the generative power of (hybrid) prescribed team CD grammar systems extends beyond the class of metalinear languages. Moreover, already prescribed team CD grammar systems with this restriction to metalinear productions are equal to the class of programmed grammars with the

same restriction and appearance checking in the case of the maximal rewriting strategies. For the other modes of derivation, the equalities hold only without appearance checking. This summarizes Theorem 10.

Finally, in the special case of prescribed team CD grammar systems with only one production per component and teams of variable size, an equality with the class of unordered scattered context grammars is presented in Theorem 11. This leads to Theorem 12, which says that only one production per component suffices in the case of prescribed team CD grammar systems with only regular, linear, metalinear or context-free productions, teams of variable size and a mode of derivation $f \in \{=1, \geq 1, *\} \cup \{\leq k \mid k \geq 1\}$.

As stated in the beginning of this section, some new characterizations of recursively enumerable languages are thus obtained. The same holds for the programmed grammars with unconditional transfer. Moreover, a normal form for recursively enumerable languages follows from these results: every RE language can be generated by a CD grammar system with the t_0, t_1 or t_2 mode of derivation and teams consisting of only two members or by a prescribed hybrid team CD grammar system with teams of just two members.

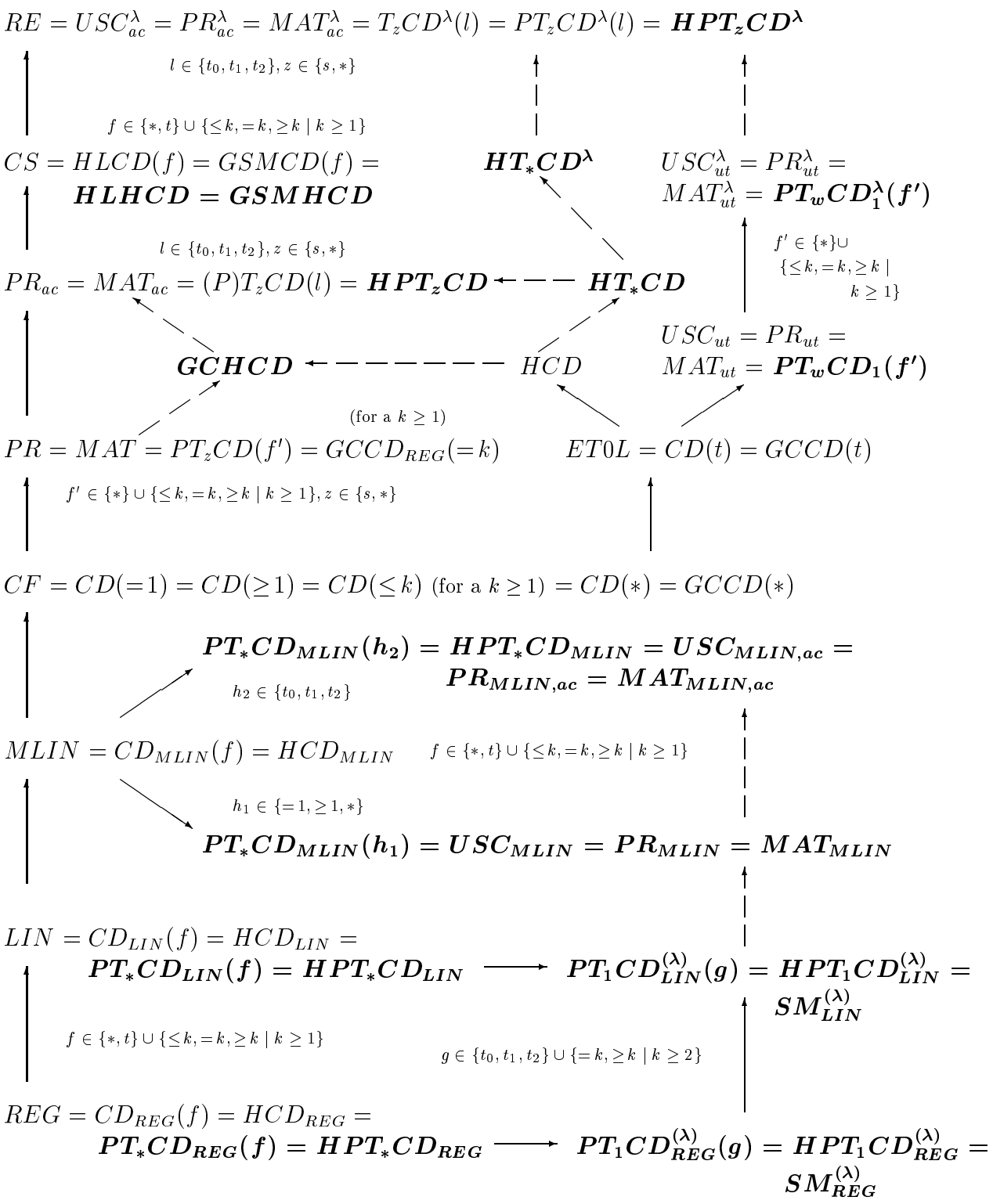
3 Summary and open problems

First, a summary of the results presented in this thesis so far will be given in the form of a diagram. For the sake of brevity, some minor results have been omitted. A hierarchy along the lines of the Chomsky hierarchy is chosen. In this way, readers will obtain a clear insight into the power of teams in grammar systems.

Although controlled (hybrid) CD grammar system have not really been discussed (the title of this thesis is, after all, *teams* in grammar systems), several relations concerning them are presented. These results can be found in Appendix E. In this way, the reader can notice that the team feature enlarges the power of grammar systems more than any of the three control mechanisms that have been considered in this thesis do.

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, all relations of which a proof is included in this thesis are printed in boldface. However, some of these proofs (those concerning controlled hybrid CD grammar systems) can be found in the appendices only.

Finally, the notation (λ) attached to a class of languages indicates that a restriction to λ-free productions does not influence the generative power. Likewise, the notation (P) must be understood.


$RE = USC^{\lambda}_{ac} = PR^{\lambda}_{ac} = MAT^{\lambda}_{ac} = T_zCD^{\lambda}(l) = PT_zCD^{\lambda}(l) = \boldsymbol{HPT_zCD^{\lambda}}$
$l \in \{t_0, t_1, t_2\}, z \in \{s, *\}$
$f \in \{*, t\} \cup \{\leq k, = k, \geq k \mid k \geq 1\}$
$CS = HLCD(f) = GSMCD(f) = \boldsymbol{HLHCD = GSMHCD}$
$\boldsymbol{HT_*CD^{\lambda}}$
$USC^{\lambda}_{ut} = PR^{\lambda}_{ut} = MAT^{\lambda}_{ut} = \boldsymbol{PT_wCD^{\lambda}_1(f')}$
$l \in \{t_0, t_1, t_2\}, z \in \{s, *\}$
$f' \in \{*\} \cup \{\leq k, = k, \geq k \mid k \geq 1\}$
$PR_{ac} = MAT_{ac} = (P)T_zCD(l) = \boldsymbol{HPT_zCD}$
$\boldsymbol{HT_*CD}$
$USC_{ut} = PR_{ut} = MAT_{ut} = \boldsymbol{PT_wCD_1(f')}$
$\boldsymbol{GCHCD}$
HCD
(for a $k \geq 1$)
$PR = MAT = PT_zCD(f') = GCCD_{REG}(= k)$
$ET0L = CD(t) = GCCD(t)$
$f' \in \{*\} \cup \{\leq k, = k, \geq k \mid k \geq 1\}, z \in \{s, *\}$
$CF = CD(=1) = CD(\geq 1) = CD(\leq k)$ (for a $k \geq 1$) $= CD(*) = GCCD(*)$
$\boldsymbol{PT_*CD_{MLIN}(h_2) = HPT_*CD_{MLIN} = USC_{MLIN,ac} = PR_{MLIN,ac} = MAT_{MLIN,ac}}$
$h_2 \in \{t_0, t_1, t_2\}$
$MLIN = CD_{MLIN}(f) = HCD_{MLIN}$
$f \in \{*, t\} \cup \{\leq k, = k, \geq k \mid k \geq 1\}$
$h_1 \in \{= 1, \geq 1, *\}$
$\boldsymbol{PT_*CD_{MLIN}(h_1) = USC_{MLIN} = PR_{MLIN} = MAT_{MLIN}}$
$LIN = CD_{LIN}(f) = HCD_{LIN} =$
$\boldsymbol{PT_*CD_{LIN}(f) = HPT_*CD_{LIN}}$
$\boldsymbol{PT_1CD^{(\lambda)}_{LIN}(g) = HPT_1CD^{(\lambda)}_{LIN} = SM^{(\lambda)}_{LIN}}$
$f \in \{*, t\} \cup \{\leq k, = k, \geq k \mid k \geq 1\}$
$g \in \{t_0, t_1, t_2\} \cup \{= k, \geq k \mid k \geq 2\}$
$REG = CD_{REG}(f) = HCD_{REG} =$
$\boldsymbol{PT_*CD_{REG}(f) = HPT_*CD_{REG}}$
$\boldsymbol{PT_1CD^{(\lambda)}_{REG}(g) = HPT_1CD^{(\lambda)}_{REG} = SM^{(\lambda)}_{REG}}$

After these results, many questions remain. Without pretending to be complete, a list of some of the most interesting open problems follows.

Open problem 1 *What is the generative power of prescribed team CD grammar systems operating in the weak rewriting step with a maximal derivation mode? In fact, the situation for all derivation modes not covered by Theorem 6 are of interest.*

Remark 6 *Many more open problems could be formulated about the weak rewriting step. For example, the equality between prescribed team CD grammar systems operating in this rewriting step and the programmed grammars with unconditional transfer could be used to finally prove* $PR \not\subseteq PR_{ut}$ *and/or* $PR^{\lambda} \not\subseteq PR^{\lambda}_{ut}$ *and/or to prove the strictness of (one of) the inclusions* $PR_{ut} \subseteq PR_{ac}$ *and* $PR^{\lambda}_{ut} \subseteq PR^{\lambda}_{ac}$ *(or perhaps even prove one or more equalities instead of the properness).*

Open problem 2 *Which relation holds between* HT_*CD *and* $PT_*CD(f)$*, for various modes of derivation* f *and with or without a restriction to* λ*-free productions?*

Open problem 3 *Does the automatic forming of teams strictly increase the power of hybrid CD grammar systems, with or without a restriction to* λ*-free productions? The conjecture is yes for at least the* λ*-free case.*

Remark 7 *Note that a proper inclusion of the relations of Open problem 3 in the* λ*-free case would result in confirmation of the conjecture stated in [Păun 94] that the inclusion* $HCD \subseteq MAT_{ac}$ *is proper. Unfortunately, this remains an open problem also after this thesis*

Open problem 4 *What is the relation between programmed grammars with appearance checking and hybrid CD grammar systems with automatically formed teams? Do the inclusions* $PR_{ac} \subseteq HT_*CD$ *and* $PR^{\lambda}_{ac} \subseteq HT_*CD^{\lambda}$ *hold (resulting in an equality) or are the inclusions* $HT_*CD \subseteq PR_{ac}$ *and* $HT_*CD^{\lambda} \subseteq PR^{\lambda}_{ac}$ *perhaps proper ones?*

Remark 8 *Note that a proof of these inclusions would result in the first inclusion in Theorem 7 to become an equality instead.*

Open problem 5 *What is the relation between prescribed (hybrid) team CD grammar systems with only regular or linear productions and teams of constant size and the class of metalinear languages? For teams of size* 1*, the prescribed (hybrid) team CD grammar systems with only regular or linear productions are incomparable with the class of context-free languages and properly include the classes of regular and linear languages, respectively.*

Open problem 6 *What is the generative power of prescribed (hybrid) team CD grammar systems with only regular or linear productions and teams of constant size s, for $s \geq 2$?*

Open problem 7 *Can the conjecture of Remark 3 be affirmed or denied?*

Open problem 8 *Which relation holds between simple matrix grammars with context-free productions and (hybrid) prescribed team CD grammar systems, for different modes of derivation?*

Remark 9 *Note that a proof of a relation as an answer to Open problem 8 would result in the proof of a relation between simple matrix grammars with context-free productions and matrix grammars with context-free productions (and appearance checking, if the relation is proved for modes of derivation t_0, t_1 and t_2), since the relation between matrix grammars and prescribed team CD grammar systems is a well-investigated one. An incomparability result is expected, though, as an answer for the relation between simple matrix grammars with context-free productions and prescribed team CD grammar systems as well as for the relation between simple matrix grammars with context-free productions and matrix grammars with context-free productions, an open problem stated in [DP 89].*

Open problem 9 *What is the generative power of (hybrid) prescribed team CD grammar systems with teams of variable size and at most p, $p \geq 2$, productions per component, for different types of productions and modes of derivation? And for exactly 1 production per component and other type of productions and modes of derivation than the ones stated in Theorem 12?*

Open problem 10 *Does control by a graph strictly increase the generative power of hybrid CD grammar systems? In fact, it is known that graph controlled hybrid CD grammar systems are included in the matrix grammars with appearance checking and include both the hybrid CD grammar systems and the matrix grammars without appearance checking, but are these inclusions proper or can equalities be proved. One of the inclusions of $MAT \subseteq GCHCD \subseteq MAT_{ac}$ must indeed be proper.*

Remark 10 *Note that a solution to Open problem 10 could shed light on the relation between hybrid CD grammar systems and matrix grammars without appearance checking, or perhaps even solve this open problem stated in [Păun 94].*

Part IV
Teams in eco-grammar systems

In the eco-grammar systems described in the Prerequisites, 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 rewrite parts of the string representing the environment in parallel. In this section, other ways of deciding which agents to rewrite the sentential form at a certain moment in time, based on the ideas of team forming discussed in Part III, are introduced.

1 Definitions and examples

The main subject of study in this section will be the generative power of the environmental language of simple eco-grammar systems with teams of agents. In the following definition therefore, the evolution rules, among others, will be omitted from the original definition of a simple eco-grammar system (Definition 10). Moreover, the most general definition will be stated with several subclasses defined afterwards.

Definition 20 *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 is a finite set of productions of the form $\alpha \to \beta$ for $\alpha \in V_E$ and $\beta \in V_E^*$, the* action rules of agent R_i *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 (for a definition of colonies, see Appendix A) introduced in [DKP 93] (see also Section 1.1.1).

Throughout this section, an environmental state of an eco-grammar system shall be simply called a state of an eco-grammar system.

Definition 21 *Consider an eco-grammar system* $\Sigma = (E, R_1, R_2, \ldots, R_n, Q_1, Q_2, \ldots, Q_m)$ *and let* w *and* w' *be two states of* Σ. *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.*

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

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.*

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

(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 strong (*$\_ = s$*) or weak (*$\_ = w$*) rewriting step, is*

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

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. The definition does consider "using a team" as the application of at least one agent's production.

Definition 22 *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, 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. Hence, consider by definition $PT_sETSEG_0^\lambda = PT_wETSEG_0^\lambda = ET0L$ and likewise for the λ-free case.

Definition 23 *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 22 is omitted.

(2) A extended simple eco-grammar system with prescribed teams if $\#P_E = 1$. Then the T in the notations of Definition 22 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 22 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 23 and for the λ-free cases as well.

Some examples are presented next to illustrate the above definitions.

Example 13 *Consider the following simple eco-grammar system with prescribed teams.*

$$\Sigma_2 = ((\{a\}, \{a \to aa\}, a), \{a \to aa\}, \{a \to aa\}).$$

The derivation starts by using the only team, thus rewriting the only symbol of the sentential form and leaving nothing to rewrite for the environment. This results in aa. Every next step the team replaces one occurrence of a by aa, the only

environmental table replaces all other occurrences of a by aa as well. Clearly, the generated language is

$$L_s(\Sigma_2) = L_w(\Sigma_2) = \{a^{2^n} \mid n \geq 0\}.$$

Example 14 *Consider the following simple eco-grammar system with prescribed teams.*

$$\Sigma_3 = ((\{a\}, \{a \to aa\}, a^4), \{a \to a\}, \{a \to a\}, \{\{a \to a\}, \{a \to a\}\}).$$

The derivation starts by using the only team, thus rewriting two out of the four a's by a and allowing the environmental table to replace the other two by aa each. This results in a^6. In every step, two a's are copied by the team and all others are doubled by the environmental table. The team actually allows the table to be applied to the sentential form after disabling two of the a's. Clearly, the generated language is

$$L_s(\Sigma_3) = L_w(\Sigma_3) = \{a^2\}\{a^{2^n} \mid n \geq 1\}.$$

Example 15 *Consider the following simple eco-grammar system with prescribed teams.*

$$\Sigma_4 = ((\{a, b, c\}, T, c^3), R_1, R_2, \ldots, R_6, \{R_1, R_2, R_3\}, \{R_4, R_5, R_6\}).$$

where

$$\begin{aligned} T &= \{a \to a, b \to b, c \to c\}, \\ R_1 &= R_2 = R_3 = \{c \to ca\} \text{ and} \\ R_4 &= R_5 = R_6 = \{c \to cb\}. \end{aligned}$$

The derivation starts with replacing all three c's by ca (or cb), leaving nothing to rewrite for the environment. From now on, the c's are rewritten in every step by ca or cb, while the environment copies all other symbols of the sentential form. It is clear that the generated language is

$$L_s(\Sigma_4) = L_w(\Sigma_4) = \{(cw)^3 \mid w \in \{a, b\}^*\}.$$

Finally, a more enhanced example is presented.

Example 16 *Consider the extended tabled simple eco-grammar system*

$$\Sigma_5 = ((V_E, T_E, P_E, S'), R_1, R_2, \ldots, R_7, \{R_1\}, \{R_2, R_3\}, \{R_4\}, \{R_5, R_6\}, \{R_7\}),$$

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*
- $R_7 = \{A' \to A\}$.

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. Next the above claim is proved.

If the only nonterminal appearing in the sentential form is Y (A, B), the derivation is blocked since no agent is capable of rewriting Y (A, B, both without an occurrence of X). Also if both A and Y and no other nonterminals appear in the sentential form no agent can be used. The same holds for only A and B, only B and Y and only A, B and Y. In fact, occurrences of A or B can only be rewritten in case also X is present in the sentential form. Only C's can never be replaced by anything else than B's and C's, but clearly never introduce a terminal. Finally, since C can never introduce an X and, without an X, A' can only derive F, A or A', the above claim holds.

Now the derivations of Σ_5 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 to the sentential form $(A'bC)^{i_1}abC(A'bC)^{i_2}Y$ (in the case of table T_1), 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

$$L_s(\Sigma_5) = \{(ab^nc)^nc \mid n \geq 1\}.$$

2 Generative power

In this section the generative power of the various kinds of eco-grammar systems, as defined in Section 1, 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 25 *For $X \in \{\lambda, E, T, ET\}$*

$$PT_sXSEG = PT_{s,1}XSEG \text{ and } PT_sXSEG^\lambda = PT_{s,1}XSEG^\lambda.$$

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

Theorem 13 *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 [CKKP 93] it was proved that already simple eco-grammar systems without teams generate strictly more than 0L systems. In fact, the simple eco-grammar system Σ_1 of Example 3 suffices to prove the properness of the first inclusion of this theorem. (ii) In Example 13 it was shown that there exists a system Σ_4 such that $L_x(\Sigma_4) = \{(cw)^3 \mid w \in \{a, b\}^*\} \in PT_xSEG_6$. It was proved in [RS 80] that this language cannot be generated by any E0L system. (iii) In Example 13 it was also shown that there exists a system Σ_3 such that $L_x(\Sigma_3) = \{a^2\}\{a^{2^n} \mid n \geq 1\} \in PT_xTSEG_2$. It was proved in [Roz 73] that this language cannot be generated by any T0L system. (iv) In Example 14 it was shown that there exists a system Σ_5 such that $L_s(\Sigma_5) = \{(ab^nc)^nc \mid n \geq 1\} \in PT_sETSEG_7$. 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 [ER 76]. □

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 [RvS 78] (see the Prerequisites) therefore seems appropriate.

Lemma 26

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

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 25 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

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

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

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

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 occuring 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. $\square$

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

Lemma 27

$$(i)\quad USC \subset PT_{s,1}ESEG \text{ and } USC^{\lambda} \subset PT_{s,1}ESEG^{\lambda} \text{ and}$$

$$(ii)\quad USC_{ut} \subseteq PT_{w,1}ESEG \text{ and } USC^{\lambda}_{ut} \subseteq PT_{w,1}ESEG^{\lambda}.$$

Proof First, the inclusions of (i) will be proved, later it will be shown that (ii) follows from this. Therefore, consider the unordered scattered context grammar

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

To simulate G, construct the following extended simple eco-grammar system with prescribed teams operating in the strong rewriting step.

$$\Sigma = ((V_E, T_E, T_1, S), R_1, R_2, \ldots, R_n, Q_1, Q_2, \ldots, Q_m),$$

where

$$\begin{aligned} V_E &= N \cup T, \\ T_E &= T, \\ T_1 &= \{\alpha \to \alpha \mid \alpha \in V_E\} \text{ and} \\ R_1, R_2, \ldots, R_n &\text{ and} \\ Q_1, Q_2, \ldots, Q_m &\text{ are defined below.} \end{aligned}$$

For every scattered context rule $p : (\alpha_1, \alpha_2, \ldots, \alpha_s) \to (\beta_1, \beta_2, \ldots, \beta_s) \in P$, construct the team $Q_p = \{R_1, R_2, \ldots, R_s\}$ with

$$R_j = \{\alpha_j \to \beta_j\}, \text{ for } 1 \leq j \leq s.$$

A parallel rewriting step of an unordered scattered context grammar is 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 13 it was shown that $L_1 = \{a^{2^n} \mid n \geq 0\} \in PT_sESEG$. In [HJ 94], however, it was proved that $L_1 \notin PR^\lambda$. Since $USC = PR$ and $USC^\lambda = PR^\lambda$ (see the Prerequisites), 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 and the proper inclusion $EPT0L \subset USC_{ut}$, stated in the Prerequisites, is a strengthening of Theorem 13.

Corollary 20

$$EPT0L \subset PT_wETSEG \text{ and } ET0L \subset PT_wETSEG^\lambda.$$

A second corollary of this lemma is the following.

Corollary 21

$$(i) \quad PUSC \subseteq PT_{s,1}SEG \text{ and } PUSC^\lambda \subseteq PT_{s,1}SEG^\lambda \text{ and}$$

$$(ii) \quad PUSC_{ut} \subseteq PT_{w,1}SEG \text{ and } PUSC_{ut}^\lambda \subseteq PT_{w,1}SEG^\lambda.$$

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

Lemma 28

$$PT_wETSEG \subseteq PR_{ac} \text{ and } PT_wETSEG^\lambda \subseteq PR_{ac}^\lambda.$$

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 [Sal 73]. 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

	σ	φ
$1 : S \to \omega$	*out*	$\emptyset$

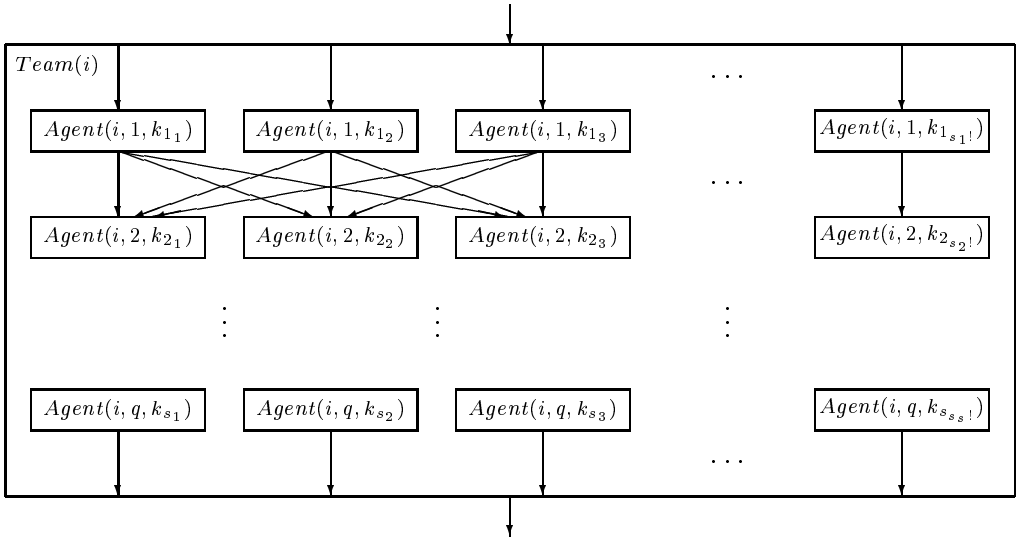
and simply introduces the (string) axiom of Σ.

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

		σ	φ
$\pi(1)$:	$\alpha_{\pi(1)} \to x'_{\pi(1)_1} x'_{\pi(1)_2} \ldots x'_{\pi(1)_{p_{\pi(1)}}}$	*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*	$\pi(3)$
:		:	:
$\pi(s)$:	$\alpha_{\pi(s)} \to x'_{\pi(s)_1} x'_{\pi(s)_2} \ldots x'_{\pi(s)_{p_{\pi(s)}}}$	*out*	*out*

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' 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

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

by a programmed grammar with appearance checking. Due to the known inclusion $E0L \subset PR_{ac}$, see e.g. [RS 80], 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 $Check$, to be defined shortly, will check that it does not do so before only primed symbols appear in the sentential form.

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

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

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

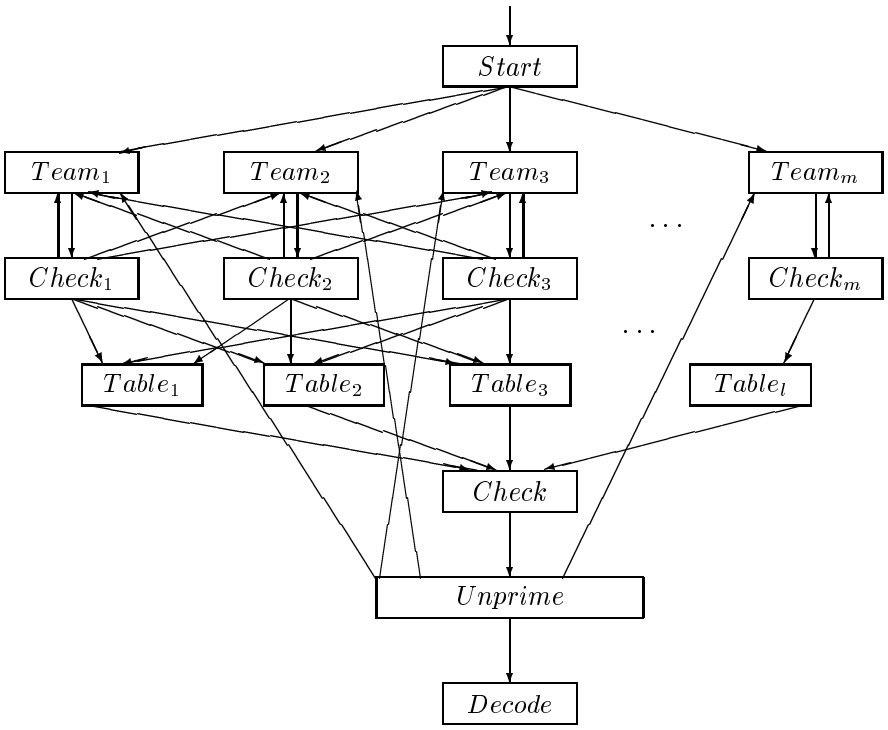
	σ	φ
$1:\ a \to \bar{a}$	$1, 2$	2
$2:\ b \to \bar{b}$	$2, 3$	3
$\vdots$	$\vdots$	$\vdots$
$z:\ z \to \bar{z}$	z	z

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 14 *For* $X \in \{E, T\}$

$$\begin{aligned} &(i) \quad PR \subset PT_sXSEG \subseteq PT_sETSEG \subseteq PR_{ac} \text{ and} \\ &\qquad PR_{ut} \subseteq PT_wXSEG \subseteq PT_wETSEG \subseteq PR_{ac} \text{ and} \\ \\ &(ii) \quad PR^\lambda \subset PT_sXSEG^\lambda \subseteq PT_sETSEG^\lambda \subseteq RE \text{ and} \\ &\qquad PR^\lambda_{ut} \subseteq PT_wXSEG^\lambda \subseteq PT_wETSEG^\lambda \subseteq RE. \end{aligned}$$

Proof The equalities 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

the Prerequisites). Furthermore, $[m, LU, p]\ KVEPT0L = PR_{ac}$ (see the Prerequisites), whereas the inclusion $[m, LU, p]\ KVET0L \subseteq RE$ is obvious. Hence all inclusion from left to right were proved in Lemma 26, 27 and 28 or follow directly from the definitions. □

3 Closure properties

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

Definition 24 *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 other team or table of* Σ *than* Q_T *and* T_T.

Theorem 15 *For every language in* PT_wETSEG *or* PT_wETSEG^λ *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 24. 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 29 *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 30 *The families PT_wETSEG and PT_wETSEG^λ 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 15, 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

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

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. $\square$

Lemma 31 *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 15, 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

$$Q_i' = \{\{[w_1 A w_2] \to [w_1 x w_2]' \mid [w_1 A w_2] \in N_1,\ A \to x \in R\} \mid R \text{ is an agent of } Q_i\}$$

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

$$\begin{aligned} T_j' \;=\; & \{[w_1 A w_2] \to [w_1 x w_2]' \mid [w_1 A w_2] \in N_1,\ A \to x \in T_j\} \cup \{[\alpha] \to [\alpha]' \mid \\ & [\alpha] \in V_E',\ \neg(\exists\ A \to x \in T_j,\ A \text{ appears in } \alpha,\ [\alpha] \in N_1)\} \cup \\ & \{[\alpha]' \to F \mid [\alpha]' \in V_E'\} \end{aligned}$$

and

$$\begin{aligned} T_j'' \;=\; & \{[\alpha]' \to [\beta][\gamma] \mid [\alpha] \in N_2,\ [\beta], [\gamma] \in N_1,\ \alpha = \beta\gamma\} \cup \{[\alpha]' \to [\alpha] \mid \\ & [\alpha]' \in V_E',\ \neg([\alpha] \in N_2,\ [\beta], [\gamma] \in N_1,\ \alpha = \beta\gamma)\} \cup \\ & \{[\alpha] \to F \mid [\alpha] \in V_E'\}. \end{aligned}$$

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 95] 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 14 leads to the main result of this section, presented in the next theorem.

Theorem 16

$$(i) \quad PT_wETSEG = PR_{ac} \text{ and}$$

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

Since RE is a full AFL, Theorem 16 leads to closure under more properties, see e.g. [Sal 73].

Corollary 22 *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.*

4 Conclusion

In the previous sections, simple eco-grammar system with prescribed teams have been introduced. These systems are nothing more than Lindenmayer systems with teams and therefore four different classes have been defined, analogous to those of the Lindenmayer systems. Hence next to the basic class, the extended, tabled and extended tabled systems have been defined.

Concerning the usage of the teams, two rewriting steps have been introduced, like in Part III. 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.

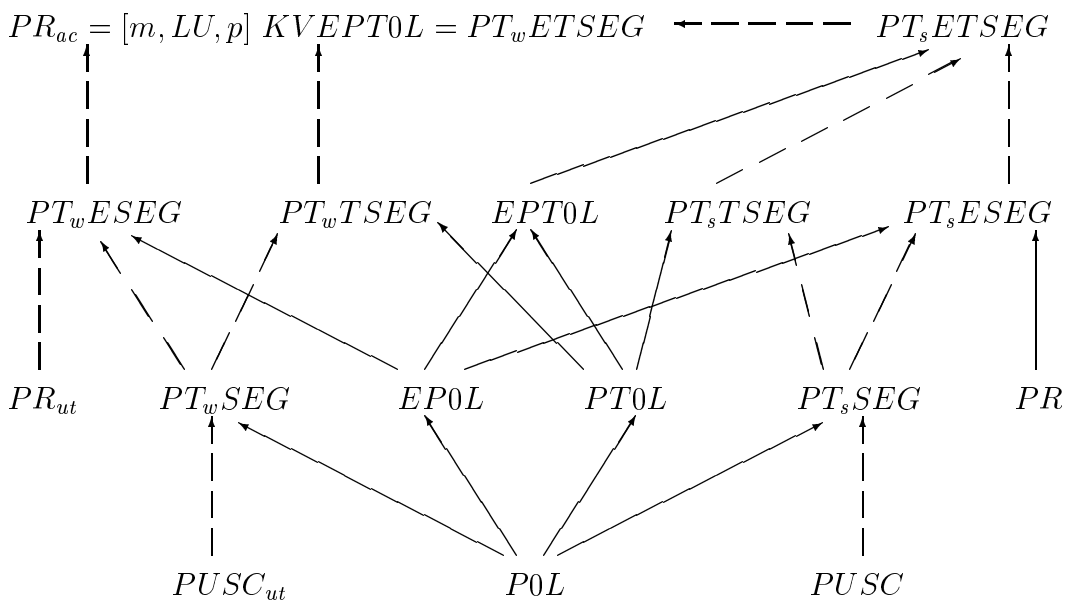
Consequently the power of forming teams in simple eco-grammar systems has been investigated. In all cases this forming of teams was found to strictly increase the generative power of the underlying (Lindenmayer) system. This result is formally presented in Theorem 13 and Corollary 20.

Theorem 14 places the power of simple eco-grammar systems with prescribed teams between 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, Theorem 15 shows a normal form for extended tabled simple eco-grammar systems with prescribed teams, operating in the weak rewriting step. With this normal form, closure of the family of languages generated by extended tabled simple eco-grammar systems with prescribed teams, operating in the weak rewriting step, under a number of closure properties has been proved. Moreover, due to some of these closure properties, in Theorem 16 these systems could be proved to be equivalent to 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 has been obtained.

5 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, the relation between weak and strong rewriting is unknown. An answer to that relation 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.

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 [Sto 71], 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 ([Fer 95]).

Finally, does the hierarchy that holds for L systems (see the diagram and note that, see e.g. [RS 80], $E(P)0L$ 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.

Appendices

A Other variants of CD grammar systems

Next to the hybrid CD grammar systems, several other types of CD grammar systems have been defined and investigated in the recent years. An attempt to give an exhaustive survey is presented below.

- Grammar systems controlled by a directed graph, hence derivations are guided by paths in the graph. See e.g. [MR 78], [CD 90], [GP 90] or [Das 91a].
- Grammar systems with memories, several different versions are defined and investigated. See e.g. [CD 88], [Csu 91] or [BC 92].
- Grammar systems with appearance checking. See e.g. [DP 90a].
- Grammar systems of finite index. See e.g. [DP 90a] or [GP 90].
- Grammar systems with registers, a special case of memories. See e.g. [DP 90b] or [DP 91].
- Grammar systems with L systems as components, in which a derivation step thus consists of a parallel rewriting step. See e.g. [CD 90] or [Wät 93].
- Grammar systems whose sentential forms are put into a gsm before another derivation step can take place. See e.g. [Mit 90] or [Păun 95a].
- Grammar systems with dynamical activation of the components (in which the component with currently maximal competence is used), or with balanced activation, etc. See e.g. [KP 91].
- Grammar systems with similar components, as in the grammar form theory. See e.g. [KP 91] or [CDMP 93].
- Grammar systems whose sentential forms are compared to a regular language, accepting only those sentential forms satisfying the format of the regular language. See e.g. [Das 91b] or [Păun 95a].
- Grammar systems with components generating finite languages, so-called *colonies*. These are a particular case of CD grammar systems for which hierarchies on the number of components, as well as other problems still open for CD grammar systems, are solved. Because of this and because the are referred to in the thesis, their definition is given here. A colony is a construct $\Gamma = (N, T, \omega, (S_1, L_1), (S_2, L_2), \ldots, (S_n, L_n))$, where N is the

set of nonterminals, T is the set of terminals, $\omega \in (N \cup T)^*$ is a string axiom, $S_i \in N$ and L_i is a finite subset of $((N \cup T) \setminus \{S_i\})^*$ for $1 \leq i \leq n$. Colonies are a special case of CD grammar systems. However, there are many variants and a difference can be that terminals of one component can be nonterminals of another component. See e.g. [KK 92a], [KK 92b], [DKP 93], [KC 94], [KK 94] or [Păun 95a].

- Grammar systems, or hybrid grammar systems, with time delay. See e.g. [CDMP 93].

- Grammar systems in which the sets of productions as components are replaced by grammars, thus allowing a terminal in one component to be a nonterminal in another component. Also variations in defining the terminal alphabet for the whole grammar system are considered. See e.g. [CDK 92].

- Grammar systems which are dynamically controlled, imposing internal control by start and stop conditions for the components. See e.g. [CDP 93].

- Grammar systems with non-context-free components, such as type-0 and context-sensitive components, are considered. See e.g. [CDKP 94a].

- Grammar systems which are structured by a hierarchy or an ordered relation on the set of components. See e.g. [CDKP 94b] or [MPR 94].

- Grammar systems, or hybrid grammar systems, with teams. These, together with many variants, form the topic of Part III of this thesis. See e.g. [CP 93], [PR 94], [MMS 94], [KMPS 95], [FP 95], [Mat 95] and Part III of this thesis.

- Grammar systems, or hybrid grammar systems, in which strings are replaced by other objects. See e.g. [DFP 95].

- Grammar systems, or hybrid grammar systems, with accepting grammars as components. See e.g. [FHB 96].

- Grammar systems that are internally hybrid, i.e. complex derivation modes, formed of the known derivation modes, are assigned to the components. See e.g. [FFH 96].

- Grammar systems with leftmost derivation. See e.g. [Păun 96].

Several of these variants have been described in [CDKP 94a], but not those that were defined after, roughly, 1992. Some other surveys of the theory of grammar systems and its variants can be found in [Păun 95b] and [Das 95].

B Parallel communicating grammar systems

In parallel communicating (PC) grammar systems, each component has an own sentential form associated to it, on which it works. If in each step, each component uses a production, the PC grammar system is called *synchronized*, otherwise it is called *non-synchronized*. These components can *communicate* with each other by sending their current sentential form on *request*, for which special *query symbols* are introduced. If these query symbols can only be introduced by the master, the PC grammar system is called *centralized*, otherwise it is called *non-centralized*. After having sent its sentential form to another component, a component may still have a copy of this sentential form or it goes back to its axiom again. In the latter case, the PC grammar system is called *returning*, otherwise it is called *non-returning*. If one component is considered the *master*, then its terminal words, possibly generated with the aid of communication, form the language of the whole PC grammar system. When there is no master, two modes can be considered. In the *competitive mode*, the derivation comes to an end as soon as one of the components generates a terminal word; in the *popular mode* the generated language is the collection of all terminal words generated by any component of the PC grammar system.

These grammar systems were introduced in [PS 89] for the regular case. The context-free case appeared in [Păun 89a], the context-sensitive case in [PIT 93] and the case of L systems in [Păun 92]. Non-synchronized PC grammar systems were introduced in [Păun 89b], non-centralized PC grammar systems appeared in [Păun 90], non-returning PC grammar systems in [Păun 91] and the competitive as well as the popular mode has been introduced in [Vasz 96]. A survey can be found in [Sân 90], [CDKP 94a] and [Păun 95b].

C Other variants of eco-grammar systems

An *extended eco-grammar system* is a usual eco-grammar system, except that each of the alphabets V_E and V_i are a disjoint union of two sets, N_E and T_E and N_i and T_i for $1 \leq i \leq n$, called a nonterminal alphabet and a terminal alphabet, respectively. The language generated by an extended eco-grammar system is the set of all states over the cartesian product of the terminal alphabets that can be reached from an initial state.

In [CKKP 93] and [CKKP 94b] it was proved that even very restricted extended eco-grammar systems can generate the class of context-sensitive languages; if λ-productions are allowed even the recursively enumerable language class can be obtained.

Not only restricted eco-grammar systems have been defined and investigated. Several enhanced eco-grammar systems have been introduced as well, one example being the *conditional tabled eco-grammar systems* which will be discussed in

more detail in Appendix D (see also [CPS 94] and [CPS 95]). Two more examples are the *multi-level eco-array grammars* introduced in [Freu 95] and the *eco-pattern systems* introduced in [Mit 95].

D Eco-grammar systems and Artificial Life

If one considers eco-grammar systems from the point of view of AL, it is clear that one's interests lie in the evolution of the agents (i.e. *organisms*) and the environment in an eco-grammar system. Therefore, the basic model is augmented by the following definition.

Definition 25 *Consider an eco-grammar system* $\Sigma = (E, A_1, A_2, \ldots, A_n)$ *with an initial state* σ_0. *Then for* $1 \leq i \leq n$

- *the* set of state sequences of Σ *is defined by*

$$Seq(\Sigma, \sigma_0) = \{\{\sigma_i\}_{i=0}^{\infty} \mid \sigma_0 \Longrightarrow_\Sigma \sigma_1 \Longrightarrow_\Sigma \sigma_2 \Longrightarrow_\Sigma \ldots\},$$

- *the* set of environmental state sequences of Σ *is defined by*

$$Seq_E(\Sigma, \sigma_0) = \{\{w_{E_i}\}_{i=0}^{\infty} \mid \{\sigma_i\}_{i=0}^{\infty} \in Seq(\Sigma, \sigma_0), \sigma_i = (w_{E_i}, w_{1_i}, \ldots, w_{n_i})\} \text{ and}$$

- *the* set of state sequences of the i-th agent A_i of Σ *is defined by*

$$Seq_i(\Sigma, \sigma_0) = \{\{w_{i_k}\}_{k=0}^{\infty} \mid \{\sigma_k\}_{k=0}^{\infty} \in Seq(\Sigma, \sigma_0), \sigma_k = (w_{E_k}, w_{1_k}, \ldots, w_{i_k}, \ldots, w_{n_k})\}.$$

The thus resulting definition of an eco-grammar system is advanced enough to capture many real life situations. This section is continued with a survey of such possibilities, without pretending to even approach completeness.

An organism can be considered *adult* when its state becomes statistically constant, i.e. the symbols that do change still represent only a very small fraction of the total string. This idea has already been investigated for generative devices, one can think of the adult L systems to name only one. Variations of adultness include *old* agents or almost dead *alive fossils*.

Considering the set of agents as a *population* a basic feature of real life, *birth*, was introduced in [CKKP 94b] and based on fragmented L systems. A *reproductive* eco-grammar system is a construct $\Sigma = (E, \mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n)$, where E is an environment defined as in the case of usual eco-grammar systems and $\mathcal{A}_i$, $1 \leq i \leq n$, is a multiset of agents of the i-th type, defined as follows. Every agent A_i in $\mathcal{A}_i$ has the same form $A_i = (V_i \cup \{\sqcup\}, P_i, R_i, \varphi_i, \psi_i)$, where V_i, P_i, R_i, φ_i and ψ_i are defined as in the case of usual eco-grammar systems but for one exception: P_i is allowed to have productions of the form $\alpha \to \beta_1 \sqcup \beta_2 \sqcup \ldots \sqcup \beta_k$ for $\alpha \in V_i$, $\beta_j \in V_i^*$, $\sqcup \notin \bigcup_{i=1}^{n} V_i$, $k \geq 2$ and $1 \leq j \leq k$. This $\sqcup$ is called the *reproduction symbol*.

A state of a reproductive eco-grammar system $\Sigma = (E, \mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n)$ is a construct $\sigma = (w_E, W_1, W_2, \ldots, W_n)$, where $w_E \in V_E^*$ and W_i is a multiset of strings $\langle w_{i_1}, w_{i_2}, \ldots, w_{i_{k_i}} \rangle$ for $w_{i_j} \in V_i^*$, $1 \leq j \leq k_i$ and $1 \leq i \leq n$. This w_E is the state of the environment and these w_{i_1} to $w_{i_{k_i}}$ are the states of all agents of the i-th type.

The birth of new agents is indicated by the occurrence of $\sqcup$ in the state of an agent. Whenever the state w of an agent A has the form $w_1 \sqcup w_2 \sqcup \ldots \sqcup w_n$, the agent is reproduced as a multiset of agents $\langle A^{(1)}, A^{(2)}, \ldots, A^{(n)} \rangle$ of the same type, each agent $A^{(i)}$ being in state w_i, $1 \leq i \leq n$. These new agents $A^{(1)}$ to $A^{(n)}$ inherit all properties of their ancestor; they thus have the same sets of evolution rules and action rules, as well as the same mappings φ and ψ.

In reproductive eco-grammar systems, a state $\sigma = (w_E, W_1, W_2, \ldots, W_n)$ directly derives a state $\sigma' = (w'_E, W'_1, W'_2, \ldots, W'_n)$, written as $\sigma \Longrightarrow_\Sigma \sigma'$, iff w'_E results from w_E as in the case of usual eco-grammar systems and W'_i is the multiset of states resulting from evolution of the states in W_i, $1 \leq i \leq n$.

Birth in reproductive eco-grammar systems is a form of *asexual* reproduction since the new agent is introduced by means of evolution. Slight modifications of the concept (e.g. allowing agents not only to rewrite the state of the environment, but the state of other agents as well) can capture *sexual* reproduction by means of an action of one agent on another one and *inheritance* by the new agent of both its ancestors as well.

When more and more organisms are introduced, a situation in which there exist more agents than symbols in the environment can occur, leading to *overpopulation*. This is in contrast with the fact when no evolution rule is selected by the mapping φ or the selected evolution rule is incapable of rewriting the state of the agent, leading to *death* of organisms.

Consider the set of action rules of agents in the eco-grammar system to be allowed to contain not only rules over the environment, but rules over the alphabets of the various other agents as well. Then an organisms whose action rules contain both type of rules can be called *omnivorous*, whereas organisms whose action rules are over the alphabets of the other agents only can be called *carnivorous*. Agents can be acting on other agents in a horizontal way as well as in a vertical way. Therefore, two different types of carnivorous agents have been introduced in [Csu 95], the first one being based on symbiotic colonies.

An eco-grammar system $\Sigma = (E, A_1, A_2, \ldots, A_n, \rho)$ is called *symbiotic* when there exists a symmetric binary relation ρ, called symbiosis, on $\{A_1, A_2, \ldots, A_n\}$. An agent A_i can perform an action on the environment iff each agent A_j for which $\rho(A_i, A_j)$ holds, $1 \leq i, j \leq n$, simultaneously performs an action on the environment. Moreover, these two agents have to act at connected places, that is the subwords of the environmental state on which they execute the action have to be adjacent.

An eco-grammar system $\Sigma = (E, A_1, A_2, \ldots, A_n, \kappa)$ is said to have *parasitic* agents when there exists a non-symmetric binary relation κ on $\{A_1, A_2, \ldots, A_n\}$.

An agent A_j is called a parasite of A_i iff $\kappa(A_i, A_j)$ holds, $1 \leq i, j \leq n$. These agents are not allowed to perform an action at the same time, even if they are both active. Moreover, the parasite A_j can only perform an action on a subword of the environmental state that is the result of an action by A_i at the preceding derivation step, $1 \leq i, j \leq n$.

One more possibility for both horizontal and vertical relations of agents, which can capture predator-and-prey cycles, was also presented in [Csu 95] and based on stratified grammar systems. A *stratified eco-grammar system* is a construct $\Sigma = (E, \mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n)$, where E is a usual environment and $\mathcal{A}_i = \langle A_{i_1}, A_{i_2}, \ldots, A_{i_{k_i}} \rangle$ is a multiset of usual agents A_{i_j}, $1 \leq j \leq k_i$ and $1 \leq i \leq n$, called a *stratum*. Furthermore, $\mathcal{A}_i \cap \mathcal{A}_j = \emptyset$.

A state of a stratified eco-grammar system $\Sigma = (E, \mathcal{A}_1, \mathcal{A}_2, \ldots, \mathcal{A}_n)$ is a construct $\sigma = (w_E, W_1, W_2, \ldots, W_n)$ for $w_E \in V_E^*$, $W_i = \langle w_{i_1}, w_{i_2}, \ldots, w_{i_{r_i}} \rangle$, $w_{i_j} \in V_{i_j}^*$, $1 \leq j \leq r_i$ and $1 \leq i \leq m$. A state $\sigma = (w_E, W_1, W_2, \ldots, W_n)$ directly derives a state $\sigma' = (w'_E, W'_1, W'_2, \ldots, W'_n)$, written as $\sigma \Longrightarrow_\Sigma \sigma'$, iff

- $w_{n_j} \Longrightarrow_{\varphi_n(w_E)} w'_{n_j}$ for $1 \leq j \leq r_n$, that is all new states of agents of stratum $\mathcal{A}_n$ are the result of an evolution,
- if $w_{j_k} = x_1 \ldots x_p \ldots x_s$ for $x_i \in V_j$, then $w'_{j_k} = y_1 \ldots z_p \ldots y_s$ such that y_i is the result of an evolution of x_i for $i \neq p$ and z_p results from x_p due to an action of some agent of stratum $\mathcal{A}_{j+1}$ for $1 \leq i \leq s$, $1 \leq k \leq r_j$ and $1 \leq j \leq n-1$,
- each active agent performs exactly one action and
- w'_E results from w_E by a simultaneous action of all active agents of stratum $\mathcal{A}_1$ and an evolution of those symbols that are not being rewritten by this simultaneous action.

The differences with the usual eco-grammar systems are obvious: an active agent does not perform an action on the environment, but on some agent in the stratum one level below it and the environment is modified exclusively by active agents of the stratum being at the bottom of the hierarchy.

Changes of season can be captured by eco-grammar systems if the 0L rules for evolution are replaced by T0L rules; in a given moment (season) one such a table (set of 0L rules) is used. The moment to change from one season to another can be triggered by a certain critical state (e.g. March 21st for the start of spring in the northern part of the world). The *conditional tabled eco-grammar systems* introduced in [CPS 95] can deal with seasonal changes. Moreover, this model can cope with *pollution* by strongly polluting sources (agents) such as damaged nuclear power plants or vulcanos. Therefore, the local action of agents on the environment has been removed and evolution of the environment has been made dependent of the states of the agents. This is done by so-called condition strings,

both permitting and forbidding ones; a certain string must be present while another may not be present to be able to apply a table. *Hibernating* animals can be described by agents that are almost dead during a certain season, but participate again after a change of tables.

In an eco-grammar system the agents can act on the whole environment, though in real life organisms are mostly bounded to certain places. This can be achieved in the model as well by the use of context-sensitive productions; a symbol can only be rewritten in the presence of another specific symbol. In this way *migration* of (groups of) organisms can be captured.

E Controlled hybrid CD grammar systems

To put the results in this section in a proper perspective, some results from [CD 90], [Das 91b] and [Mit 90] are restated in the following lemma.

Lemma 32 *For* $k, k' \geq 1$ *and* $f \in \{*, t\} \cup \{\leq k, =k, \geq k \mid k \geq 1\}$

(i) $GCCD_{REG}(=k) = GCCD(\leq k) = GCCD(=k) = PR$,
$GCCD(\geq k) = GCCD(\geq k') \subseteq PR$ *and*
$CF = GCCD(*) \subset GCCD_{REG}(t) = GCCD(t) = ET0L$,

(ii) $HLCD(f) = CS$ *and*

(iii) $GSMCD(f) = CS$.

Furthermore, it is clear that similar to Lemma 24 every (hybrid) CD grammar system is also a controlled (hybrid) CD grammar system.

Next, the generative power of controlled hybrid CD grammar systems will be investigated, hence a restriction of the grammar systems which are controlled. All the results in this appendix are new. First the case of a graph as external control is considered.

Lemma 33

$$GCHCD \subseteq MAT_{ac} \text{ and } GCHCD^{\lambda} \subseteq MAT_{ac}^{\lambda}.$$

Proof Consider a graph controlled hybrid CD grammar system

$$\Gamma = (N, T, S, (P_1, f_1), (P_2, f_2), \ldots, (P_n, f_n), U).$$

To simulate this hybrid CD grammar system with graph control, construct the following matrix grammar with appearance checking

$$G' = (N', T', S', M', F'),$$

where

$$\begin{aligned}
N' &= N \cup \{S'\} \cup \{[P_i, f_i], [P_i, \geq k]' \mid (P_i, f_i), (P_i, \geq k) \in \Gamma, 1 \leq i \leq n\}, \\
T' &= T \cup \{z\}, \\
M' &= \{(S' \to S[P_i, f_i]) \mid f_i \in \{*, t\} \cup \{\leq k, = k, \geq k \mid k \geq 1\}, 1 \leq i \leq n\} \cup \\
&\quad \{([P_i, \leq k] \to [P_j, f_j], A_1 \to x_1, A_2 \to x_2, \ldots, A_h \to x_h) \mid \\
&\quad A_g \to x_g \in P_i, f_j \in \{*, t\} \cup \{\leq k, = k, \geq k \mid k \geq 1\}, 1 \leq g \leq h \leq k, \\
&\quad (P_i, P_j) \in E, 1 \leq i, j \leq n\} \cup \\
&\quad \{([P_i, \leq k] \to [P_j, f_j]) \mid f_j \in \{*, t\} \cup \{\leq k, = k, \geq k \mid k \geq 1\}, \\
&\quad 1 \leq i, j \leq n\} \cup \\
&\quad \{([P_i, = k] \to [P_j, f_j], A_1 \to x_1, A_2 \to x_2, \ldots, A_k \to x_k) \mid \\
&\quad A_g \to x_g \in P_i, f_j \in \{*, t\} \cup \{\leq k, = k, \geq k \mid k \geq 1\}, 1 \leq g \leq k, \\
&\quad (P_i, P_j) \in E, 1 \leq i, j \leq n\} \cup \\
&\quad \{([P_i, \geq k] \to [P_, \geq k]', A_1 \to x_1, A_2 \to x_2, \ldots, A_k \to x_k) \mid \\
&\quad A_g \to x_g \in P_i, 1 \leq g \leq k, 1 \leq i \leq n\} \cup \\
&\quad \{([P_i, \geq k]' \to [P_i, \geq k]', A \to x) \mid A \to x \in P_i, 1 \leq i \leq n\} \cup \\
&\quad \{([P_i, \geq k]' \to [P_j, f_j]) \mid f_j \in \{*, t\} \cup \{\leq k, = k, \geq k \mid k \geq 1\}, \\
&\quad (P_i, P_j) \in E, 1 \leq i, j \leq n\} \cup \\
&\quad \{([P_i, *] \to [P_i, *], A \to x) \mid A \to x \in P_i, 1 \leq i \leq n\} \cup \\
&\quad \{([P_i, *] \to [P_j, f_j]) \mid f_j \in \{*, t\} \cup \{\leq k, = k, \geq k \mid k \geq 1\}, \\
&\quad (P_i, P_j) \in E, 1 \leq i, j \leq n\} \cup \\
&\quad \{([P_i, t] \to [P_i, t], A \to x) \mid A \to x \in P_i, 1 \leq i \leq n\} \cup \\
&\quad \{([P_i, t] \to [P_j, f_j], A_{i_1} \to F, A_{i_2} \to F, \ldots, A_{i_{s_i}} \to F) \mid \\
&\quad dom(P_i) = \{A_{i_1}, A_{i_2}, \ldots, A_{i_{s_i}}\}, f_j \in \{*, t\} \cup \{\leq k, = k, \geq k \mid k \geq 1\}, \\
&\quad (P_i, P_j) \in E, 1 \leq i, j \leq n\} \cup \\
&\quad \{([P_i, f_i] \to z) \mid f_i \in \{*, t\} \cup \{\leq k, = k, \geq k \mid k \geq 1\}, 1 \leq i \leq n\} \text{ and} \\
\text{in } F' \text{ are } & \text{ all the productions } A \to F \text{ appearing in } M'.
\end{aligned}$$

The simulation starts with applying a production $S' \to S[P_i, f_i]$, where S is the original axiom and $[P_i, f_i]$ a marker, indicating which component is being simulated and what its mode of derivation is. From S, the language of the graph controlled hybrid CD grammar system will be generated and the marker will control this generation.

When the mode is $\leq k$, indeed less than k times a production from the corresponding component is used before handing over control to another component.

Furthermore, this other component has to be connected to the current component by an edge in the controlling graph. This test is done throughout the whole construction, before handing over control to another component. In the case of mode $=k$, exactly k productions are used. For mode $\geq k$ first exactly k productions are used, after which the primed version of $[P_i, \geq k]$ is used to hand over control to another component only after another zero or more rewriting steps.

If the mode is $*$, an arbitrary number of productions is used before handing over control. Finally, in mode t the same construction is used to rewrite an arbitrary number of times. Moreover, the productions in the set F' guarantee that in this mode control can only be handed over to another component when no more production of the particular component can be used. In the case of mode $\leq k$ and $*$ the control can also directly be given to another component, corresponding to the case when the less than k or arbitrary number of rewriting steps is in fact zero.

Eventually, the marker is replaced by z thus yielding $L(G') = L(\Gamma)\{z\}$. This z can be removed by a morphism and thus, since it is known (see e.g. [DP 89]) that the family MAT_{ac} is closed under restricted morphisms, $L(\Gamma) \in MAT_{ac}$ and the first statement of the lemma is proved.

The second statement of the lemma can be proved by a similar construction, even simplified since the marker can be replaced by λ instead of z, making the use of a morphism unnecessary. □

Lemma 34

$$MAT \subseteq GCHCD \text{ and } MAT^{\lambda} \subseteq GCHCD^{\lambda}.$$

Proof Consider a matrix grammar

$$G = (N, T, S, M).$$

Denote $M = \{m_1, m_2, \ldots, m_m\}$. To simulate this matrix grammar, construct the following graph controlled hybrid CD grammar system

$$\Gamma' = (N, T, S, (P_1, f_1), (P_2, f_2), \ldots, (P_n, f_n), U),$$

where

- $(P_1, f_1), (P_2, f_2), \ldots, (P_n, f_n)$ contains for every matrix
  $m_i : (\alpha_{i_1} \to \beta_{i_1}, \alpha_{i_2} \to \beta_{i_2}, \ldots, \alpha_{i_{l_i}} \to \beta_{i_{l_i}}) \in M$, $1 \leq i \leq m$, the components

  $$(P_{i_j}, =1), \text{ where } P_{i_j} = \{\alpha_{i_j} \to \beta_{i_j}\} \text{ for } 1 \leq j \leq l_i \text{ and } 1 \leq i \leq m.$$

and

- $U = (V, E)$, where for every matrix
  $m_i : (\alpha_{i_1} \to \beta_{i_1}, \alpha_{i_2} \to \beta_{i_2}, \ldots, \alpha_{i_{l_i}} \to \beta_{i_{l_i}}) \in M$, $1 \leq i \leq m$,

  $V = \{P_{i_j} \mid 1 \leq j \leq l_i, 1 \leq i \leq m\}$ and
  $E = \{(P_{i_j}, P_{i_{j+1}}) \mid 1 \leq j \leq l_i, 1 \leq i \leq m\} \cup \{(P_{i_{l_i}}, P_{k_1}) \mid 1 \leq i, k \leq m\}$.

For every production in matrices of M, a component is constructed operating in mode $=1$. The strict ordering of productions in matrices of M is preserved by the graph. After all productions of a matrix are used, a new matrix can be started by beginning with its first production. This is all imposed by the graph. Since the productions are put in different components following each other, it is clear that a production can rewrite nonterminals introduced by a production from the same matrix which precedes it in the ordering. It is thus clear that $L(\Gamma') = L(G)$ and that both statements of the lemma are now proved. □

To put these result in a proper perspective, note that the following diagram now holds. An arrow in this diagram indicates an inclusion which is not known to be proper, whereas a double arrow indicates a proper inclusion. Families which are not connected by an arrow are not necessarily incomparable. The inclusions can be found in the Prerequisites and Lemma 33 and 34 or are obvious. Moreover, the same diagram holds when not restricted to λ-free productions.

$$\begin{array}{ccc} GCHCD & \to & MAT_{ac} \\ \uparrow & \nwarrow & \Uparrow \\ HCD & & MAT \end{array}$$

Note that one of the inclusions of $MAT \subseteq GCHCD \subseteq MAT_{ac}$ must thus be proper.

Now the case of a gsm as control mechanism is considered, leading to a result not only for control by a gsm but also for control by a hypothesis language.

Lemma 35 *$GSMHCD \subseteq CS$.*

Proof Consider a hybrid CD grammar system with a gsm

$$\Gamma = (N, T, S, (P_1, f_1), (P_2, f_2), \ldots, (P_n, f_n), g).$$

Furthermore, denote

$$g = (S, I, O, s_0, \delta, F) \text{ where } I = O = (N \cup T).$$

To simulate this hybrid CD grammar system with a gsm, construct the following Chomsky grammar of type-1.

$$G' = (N', T', S', P'),$$

where

$$
\begin{aligned}
N' &= N \cup \{S', T, T', s_0\} \cup \{[C_i, f_i, j], [C'_i, f_i, j], [C''_i, f_i, j] \mid \\
&\qquad (P_i, f_i) \in \Gamma, f_i \in \{\leq k, =k, \geq k \mid k \geq 1\}, 1 \leq i \leq n, 0 \leq j \leq k\} \cup \\
&\quad \{[C_i, g_i], [C'_i, g_i], [C''_i, g_i] \mid (P_i, g_i) \in \Gamma, g_i \in \{*, t\}, 1 \leq i \leq n\} \cup \\
&\quad \{C_{t_i}, C'_{t_i} \mid (P_i, t) \in \Gamma, 1 \leq i \leq n\}, \\
T' &= T \cup \{L, R\} \text{ and} \\
P' &= \{S' \to LTSR\} \cup \\
&\quad \{LT \to L[C_i, f_i, 0], LT \to L[C_i, g_i] \mid f_i \in \{\leq k, =k, \geq k \mid k \geq 1\}, \\
&\qquad g_i \in \{*, t\}, 1 \leq i \leq n\} \cup \\
&\quad \{[C_i, f_i, j]y \to y[C_i, f_i, j], [C'_i, f_i, j]y \to y[C'_i, f_i, j], \\
&\qquad y[C''_i, f_i, j] \to [C''_i, f_i, j]y \mid y \in (N \cup T), f_i \in \{\leq k, =k, \geq k \mid k \geq 1\}, \\
&\qquad 0 \leq j \leq k, 1 \leq i \leq n\} \cup \\
&\quad \{[C_i, g_i]y \to y[C_i, g_i], [C'_i, g_i]y \to y[C'_i, g_i], y[C''_i, g_i] \to [C''_i, g_i]y \mid \\
&\qquad y \in (N \cup T), g_i \in \{*, t\}, 1 \leq i \leq n\} \cup \\
&\quad \{[C_i, f_i, j]A \to x[C'_i, f_i, j] \mid A \to x \in P_i, f_i \in \{\leq k, =k, \geq k \mid k \geq 1\}, \\
&\qquad 0 \leq j \leq k, 1 \leq i \leq n\} \cup \\
&\quad \{[C_i, g_i]A \to x[C'_i, g_i] \mid A \to x \in P_i, g_i \in \{*, t\}, 1 \leq i \leq n\} \cup \\
&\quad \{[C_i, f_i, j]R \to [C''_i, f_i, j]R \mid f_i \in \{\leq k, =k, \geq k \mid k \geq 1\}, \\
&\qquad 0 \leq j \leq k, 1 \leq i \leq n\} \cup \\
&\quad \{[C_i, g_i]R \to [C''_i, g_i]R \mid g_i \in \{*, t\}, 1 \leq i \leq n\} \cup \\
&\quad \{L[C''_i, f_i, j] \to L[C_i, f_i, j+1] \mid f_i \in \{\leq k, =k, \geq k \mid k \geq 1\}, \\
&\qquad 0 \leq j \leq k-1, 1 \leq i \leq n\} \cup \\
&\quad \{L[C''_i, \leq k, j] \to Ls_0 \mid 0 \leq j \leq k, 1 \leq i \leq n\} \cup \\
&\quad \{L[C''_i, =k, k] \to Ls_0 \mid 1 \leq i \leq n\} \cup \\
&\quad \{L[C''_i, \geq k, k] \to L[C_i, \geq k, k], L[C''_i, \geq k, k] \to Ls_0 \mid 1 \leq i \leq n\} \cup \\
&\quad \{L[C''_i, *] \to L[C_i, *], L[C''_i, *] \to Ls_0 \mid 1 \leq i \leq n\} \cup \\
&\quad \{L[C_i, *] \to Ls_0 \mid 1 \leq i \leq n\} \cup \\
&\quad \{L[C_i, t] \to LC_{t_i}, L[C''_i, t] \to LC_{t_i} \mid 1 \leq i \leq n\} \cup \\
&\quad \{C_{t_i}y \to yC_{t_i} \mid y \to x \notin P_i, y \in (N \cup T)^+, x \in (N \cup T)^*, 1 \leq i \leq n\} \cup \\
&\quad \{C_{t_i}R \to C'_{t_i}R, yC'_{t_i} \to C'_{t_i}y, LC'_{t_i} \to Ls_0 \mid y \in (N \cup T)^+, 1 \leq i \leq n\} \cup \\
&\quad \{s_1x \to ys_2 \mid (s_1x, ys_2) \in \delta\} \cup \{sR \to TR \mid s \in F\} \cup \\
&\quad \{yT \to Ty \mid y \in O\} \cup \{LT \to LT'\} \cup \\
&\quad \{T'y \to yT' \mid y \in T\} \cup \{T'R \to R\}.
\end{aligned}
$$

The simulation starts by introducing the sentential form $LTSR$, where S is the original axiom, T is a marker and L and R are terminal symbols indicating

the left and right end, respectively, of the sentential form. From S the language of the hybrid CD grammar system with a gsm will be generated and the marker is non-deterministically replaced by a control symbol $[C_i, f_i, j]$ or $[C_i, g_i]$ indicating the use of a component P_i working in mode $f_i \in \{\leq k, = k, \geq k \mid k \geq 1\}$ or $g_i \in \{*, t\}$, respectively. In the case of a mode $f_i \in \{\leq k, = k, \geq k \mid k \geq 1\}$ a counter j, $0 \leq j \leq k$, is used; in the case of mode $*$ or t this is not necessary.

The simulation continues by moving the control symbol to the right (skipping terminals and nonterminals) until a nonterminal is replaced by a production from the component P_i, consequently priming the control symbol. Then this primed version of the control symbol is moved completely to the right of the sentential from, where it becomes double primed and is moved completely to the left again. When it is completely on the left of the sentential form, some different cases need to be considered.

In the case of mode $\leq k$, $= k$ or $\geq k$ (for a $k \geq 1$) the counter is increased by one and the process is repeated. When exactly k productions of the component P_i are used (when j reaches the value k) and the mode is $= k$, the control symbol is replaced by s_0. In the case of mode $\leq k$ this can happen for every value of j between zero and k. For mode $\geq k$ this production can be used when j is equal to k, but also another process guided by the initial control symbol with counter k can be started. This allows the use of a production more than k times, indeed corresponding to mode $\geq k$, before replacing the control symbol by s_0.

In the case of mode $*$, the control symbol can be replaced by s_0 after using P_i an arbitrary number of times indeed. The same holds for mode t, except that before introducing s_0 a test is done to check if there is indeed no production left from P_i that can be used on the current sentential form. For this test, a test symbol C_{t_i} is introduced, indicating for which component (P_i) this test is done. This test symbol is moved from left to right over the sentential form, allowed to skip any terminals but only those nonterminals for which there is no production in the component P_i being tested. When it reaches the right end, it is replaced by its primed version which is then moved completely to the left to be replaced by s_0.

In any mode, the result is the introduction of s_0 to indicate the end of the work of the component. This s_0 is the start-symbol of the gsm. Next, the usage of the gsm is simulated on the sentential form remaining after using a component. It does its work as usual and when it reaches the right side of the sentential form in a final state, this final state-symbol is replaced by the marker T again. This T is moved to the left, skipping only symbols from the output alphabet of the gsm, where it is replaced by T'. Finally, this T' is moved completely to the right skipping only terminals before it disappears.

From this detailed explanation it is clear that $L(\Gamma) = \{L\}L(\Gamma)\{R\}$ and, since it is known (see e.g. [Sal 73]) that the family CS is closed under cancellation of first and last letter, $L(G') \in CS$ and the lemma is thus proved. □

This lemma leads to the main result of this appendix, presented in the following theorem.

Theorem 17 *For* $f \in \{*, t\} \cup \{\leq k, =k, \geq k \mid k \geq 1\}$

$$CS = HLCD(f) = HLHCD = GSMHCD.$$

Proof The first equality can be found in Lemma 32. Furthermore, the inclusion $HLCD(f) \subseteq HLHCD$ is obvious. It is also clear that a gsm can check whether a given input-string is in a regular language; it can thus play the role of a hypothesis language and hence $HLHCD \subseteq GSMHCD$ holds. Finally, Lemma 35 finishes the proof of this theorem. □

References

[Ábr 65] S. Ábrahám, Some questions of phrase structure grammars. *Comput. Linguistics* 4 (1965), 61 - 70.

[BC 92] F.J. Brandenburg and E. Csuhaj-Varjú, Mailbox grammar systems. Submitted, 1992.

[CD 88] E. Csuhaj-Varjú and J. Dassow, Message handling systems. *Papers on Automata and Languages* 10 (1988), 71 - 82.

[CD 90] E. Csuhaj-Varjú and J. Dassow, On cooperating distributed grammar systems. *J. Inf. Process. Cybern. EIK* 26 (1990), 49 - 63.

[CDK 92] E. Csuhaj-Varjú, J. Dassow and J. Kelemen, Cooperating distributed grammar systems with different styles of acceptance. *Intern. J. Computer Math.* 42 (1992), 173 - 183.

[CDKP 94a] E. Csuhaj-Varjú, J. Dassow, J. Kelemen and Gh. Păun, *Grammar Systems. A Grammatical Approach to Distribution and Cooperation*, Gordon and Breach, London, 1994.

[CDKP 94b] E. Csuhaj-Varjú, J. Dassow, J. Kelemen and Gh. Păun, Stratified grammar systems. *Computers and AI* 13 (1994), 409 - 422.

[CDMP 93] E. Csuhaj-Varjú, J. Dassow, V. Mitrana and Gh. Păun, Cooperation in grammar systems: universality, similarity, timing. *Cybernetica* 4 (1993), 271 - 286.

[CDP 93] E. Csuhaj-Varjú, J. Dassow, and Gh. Păun, Dynamically controlled cooperating distributed grammar systems. *Inform. Sci.* 69 (1993), 1 - 25.

[CK 89] E. Csuhaj-Varjú and J. Kelemen, Cooperating grammar systems: a syntactical framework for the blackboard model of problem solving. In *Proc. AI and information-control systems of robots '89* (I. Plander, ed.), North-Holland Publ. Co., 1989, 121 - 127.

[CKKP 93] E. Csuhaj-Varjú, J. Kelemen, A. Kelemenová and Gh. Păun, Eco(grammar)systems: a language theoretic model for AL (Artificial Life). Manuscript, 1993.

[CKKP 94a] 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.

[CKKP 94b] E. Csuhaj-Varjú, J. Kelemen, A. Kelemenová and Gh. Păun, Eco-grammar systems - a generative framework for Artificial Life. Submitted, 1994.

[CP 93] E. Csuhaj-Varjú and Gh. Păun, Limiting the size of teams in cooperating grammar systems. *Bulletin EATCS* 51 (1993), 198 - 202.

[CPS 94] E. Csuhaj-Varjú, Gh. Păun and A. Salomaa, Conditional tabled eco-grammar systems versus (E)T0L systems. Submitted, 1994.

[CPS 95] E. Csuhaj-Varjú, Gh. Păun and A. Salomaa, Conditional tabled eco-grammar systems. In [Păun 95c] (1995), 227 - 239.

[Csu 91] E. Csuhaj-Varjú, On the size complexity of message handling grammar systems. *Computers and AI* 10 (1991), 143 - 157.

[Csu 95] E. Csuhaj-Varjú, Eco-grammar systems: recent results and perspectives. In [Păun 95c] (1995), 79 - 103.

[Das 91a] J. Dassow, A Remark on cooperating distributed grammar systems controlled by graphs. *Wiss. Zeit. T.U. Magdeburg* 35 (1991), 4 - 6.

[Das 91b] J. Dassow, Cooperating distributed grammar systems with hypothesis languages. *J. Exp. Th. AI* 3 (1991), 11 - 16.

[Das 95] J. Dassow, Cooperating grammar systems (definitions, basic results, open problems). In [Păun 95c] (1995), 40 - 52.

[DFP 95] J. Dassow, R. Freund and Gh. Păun, Cooperating array grammar systems. *Intern. J. of Pattern Recognition and AI* 9, 6 (1995), 1 - 25.

[DKP 93] J. Dassow, J. Kelemen and Gh. Păun, On parallelism in colonies. *Cybernet. Systems* 24 (1993), 37 - 49.

[DP 89] J. Dassow and Gh. Păun, *Regulated Rewriting in Formal Language Theory*, Springer-Verlag, 1989.

[DP 90a] J. Dassow and Gh. Păun, On some variants of cooperating distributed grammar systems. *Stud. Cerc. Matem.* 42 (1990), 153 - 165.

[DP 90b] J. Dassow and Gh. Păun, Cooperating distributed grammar systems with registers. *Found. Control Engineering* 15 (1990), 19 - 38.

[DP 91] J. Dassow and Gh. Păun, On the succinctness of descriptions of context-free languages by cooperating distributed grammar systems. *Computers and AI* 10 (1991), 513 - 527.

[DP 92] J. Dassow and Gh. Păun, Cooperating distributed grammar systems with regular components. *Computers and AI* (1992).

[ER 76] A. Ehrenfeucht and G. Rozenberg, On proving that certain languages are not ET0L. *Acta Informatica* 6 (1976), 407 - 415.

[Fd'A 91] J. D. Farmer and A. d'A Belin, Artificial Life: the coming evolution. In *Artificial Life II* (Ch. G. Langton et al., eds.), Addison-Wesley, Redwood City, Cal., 1991, 815 - 840.

[Fer 95] H. Fernau, Characterizations of unconditional transfer. Manuscript, 1995.

[FFH 96] H. Fernau, R. Freund and M. Holzer, Internally hybrid cooperating distributed grammar systems. *Techn. Report 185-2/FR-1/96*, Techn. Univ. Wien, 1996.

[FHB 96] H. Fernau, M. Holzer and H. Bordihn, Accepting multi-agent systems: the case of cooperating grammar systems. *Computers and AI* 15, 2-3 (1996), 123 - 139.

[FP 95] R. Freund and Gh. Păun, A variant of team cooperation in grammar systems. *J. UCS* 1, 2 (1995), 105 - 130.
http://hyperg.iicm.tu-graz.ac.at

[Freu 95] R. Freund, Multi-level eco-array grammars. In [Păun 95c] (1995), 175 - 201.

[Friš 68] I. Friš, Grammars with partial ordering of the rules. *Inform. Control* 12 (1968), 412 - 425. Correction in *Inform. Control* 14 (1969), 5.

[GH 69] S. A. Greibach and J. E. Hopcroft, Scattered context grammars. *J. Comput. Syst. Sci.* 3 (1969), 233 - 247.

[GP 90] M. Gheorghe and Gh. Păun, Further Remarks on cooperating distributed grammar systems. *Bull. Math. Soc. Sci. Math. Roumanie* 34, 82 (1990), 232 - 245.

[GW 89] J. Gonczarowski and M.K. Warmuth, Scattered versus context-sensitive rewriting. *Acta Informatica* 27 (1989), 81 - 95.

[HJ 94] D. Hauschildt and M. Jantzen, Petri net algorithms in the theory of matrix grammars, *Acta Informatica* 31 (1994), 719 - 728.

[Ib 70] O. H. Ibarra, Simple matrix languages. *Inform. Control* 17 (1970), 359 - 394.

[KC 94] A. Kelemenová and E. Csuhaj-Varjú, Languages of colonies. *Theor. Computer Sci.* 134 (1994), 119 - 130.

[KK 92a] J. Kelemen and A. Kelemenová, A subsumption architecture for generative symbol systems. In *Cybern. and Syst. Sci. '92* (R. Trappl, ed.), World Scientific, 1992, 1529 - 1536.

[KK 92b] J. Kelemen and A. Kelemenová, A grammar-theoretic treatment of multiagent systems. *Cybernet. Systems* 23 (1992), 621 - 633.

[KK 94] A. Kelemenová and J. Kelemen, From colonies to eco(grammar) systems. An overview. In *Proc. Important Results and Trends in Theoretical Computer Science*, (J. Karhumäki, H. A. Maurer and G. Rozenberg, eds.), *Lecture Notes in Computer Science 812*, Springer-Verlag, Berlin, 1994.

[KMPS 95] L. Kari, A. Mateescu, Gh. Păun and A. Salomaa, Teams in cooperating grammar systems, *J. Exper. Th. AI* 7 (1995), 347 - 359.

[KP 91] J. Kelemen and Gh. Păun, Generative paradigm in the study of complex systems: some variants of grammar systems. In *Grammar systems and colonies* (J. Kelemen, ed.), Dept. AI (Report CS4-91), Comenius Univ., Bratislava, 1991, 33 - 48.

[Král 73] J. Král, A note on grammars with regular restrictions. *Kybernetika* 9, 3 (1973), 159 - 161.

[Lang 89] Ch. G. Langton, Artificial life. In *Artificial Life I* (Ch. G. Langton, ed.), Addison-Wesley, Redwood City, Cal., 1989.

[Mat 95] A. Mateescu, A survey of teams in cooperating distributed grammar systems. In [Păun 95c] (1995), 137 - 151.

[May 72] O. Mayer, Some restricted devices for context-free languages. *Inform. Control* 20 (1972), 69 - 92.

[Mit 90] V. Mitrana, Pairs grammar systems - transducers. *Ann. Univ. Buc. Ser. Matem.-Inform.*, 39 (1990), 73 - 81.

[Mit 93] V. Mitrana, Hybrid cooperating distributed grammar systems. *Computers and AI* 2 (1993), 83 - 88.

[Mit 95] V. Mitrana, Eco-pattern systems. In [Păun 95c] (1995), 202 - 209.

[MMS 94] A. Mateescu, V. Mitrana and A. Salomaa, Dynamical teams of cooperating grammar systems. *Ann. Univ. Bucharest, Mat.-Inform.*. Submitted, 1994.

[MPR 94] V. Mitrana, Gh. Păun and G. Rozenberg, Structuring grammar systems by priorities and hierarchies. *Acta Cybern.* 11 (1994), 189 - 204.

[MR 78] R. Meersman and G. Rozenberg, Cooperating grammar systems. In *Proc. MFCS'78 Symp., LNCS* 64 (1978), 364 - 374.

[MRV 78] R. Meersman, G. Rozenberg and D. Vermeir, Cooperating grammar systems. *Techn. Report 78-12*, Univ. Antwerp, Dept. Math., 1978.

[Nii 86] P. H. Nii, Blackboard systems: The blackboard model of problem solving and the evolution of blackboard architectures. Part I. *The AI Magazine*, Summer 1986, 38 - 53.

[Nii 89] P. H. Nii, Blackboard systems. In *The Handbook of AI* 4 (A. Barr, P. R. Cohen and E. A. Feigenbaum, eds.), Addison-Wesley, Reading, Mass., 1989.

[Păun 81] Gh. Păun, On eliminating the λ-rules from simple matrix grammars. *Fundamenta Informaticae* 4 (1981), 185 - 195.

[Păun 89a] Gh. Păun, Parallel communicating grammar systems: the context-free case. *Found. Control Engineering* 14, 1 (1989), 39 - 50.

[Păun 89b] Gh. Păun, On the power of synchronization in parallel communicating grammar systems. *Stud. Cerc. Matem.* 41 (1989), 191 - 197.

[Păun 90] Gh. Păun, Non-centralized parallel communicating grammar systems. *Bulletin EATCS* 40 (1990), 257 - 264.

[Păun 91] Gh. Păun, On some questions about parallel communicating grammar systems. *Bull. Math. Soc. Sci. Math. Roumanie* 35, 83 (1991), 125 - 137.

[Păun 92] Gh. Păun, Parallel communicating systems of L systems. In *Lindenmayer Systems. Impacts on Theoretical Computer Science, Computer Graphics and Developmental Biology* (G. Rozenberg and A. Salomaa, eds.), Springer-Verlag, Berlin, 1992, 405 - 417.

[Păun 94] Gh. Păun, On the generative capacity of hybrid CD grammar systems, *J. Inform. Process. Cybern. EIK* 30, 4 (1994), 231 - 244.

[Păun 95a] Gh. Păun, On the generative capacity of colonies. *Kybernetika* 31 (1995), 83 - 97.

[Păun 95b] Gh. Păun, Grammar systems: a grammatical approach to distribution and cooperation. In *Automata, Languages and Programming; 22nd International Colloquium, ICALP'95, Szeged, Hungary, LNCS* 944 (1995), 429 - 443.

[Păun 95c] *Artificial life: grammatical models* (Gh. Păun, ed.), Black Sea Univ. Press, Bucharest, Romania, 1995.

[PIT 93] O. Procopiuc, C. M. Ionescu and F. L. Ţiplea, Parallel communicating grammar systems: the context-sensitive case. *Intern. J. Computer Math.* 49 (1993), 145 - 156.

[PR 94] Gh. Păun and G. Rozenberg, Prescribed teams of grammars. *Acta Informatica* 31 (1994), 525 - 537.

[PS 89] Gh. Păun and L. Sântean, Parallel communicating grammar systems: the regular case. *An. Univ. Buc., Ser. Matem.-Inform.* 38, 2 (1989), 55 - 63.

[Ros 69] D. J. Rosenkrantz, Programmed grammars and classes of formal languages. *J. ACM* 16, 1 (1969), 107 - 131.

[Roz 73] G. Rozenberg, T0L systems and languages. *Inform. Control* 23 (1973), 357 - 381.

[RS 80] G. Rozenberg and A. Salomaa, *The Mathematical Theory of L Systems*, Academic Press, New York, 1980.

[RvS 78] G. Rozenberg and S. H. von Solms, Priorities on context conditions. *Inform. Sci.* 14 (1978), 15 - 50.

[Sal 73] A. Salomaa, *Formal Languages*, Academic Press, New York, 1973.

[Sal 95] A. Salomaa, Developmental models for artificial life: basics of L systems. In [Păun 95c] (1995), 22 - 32.

[Sân 90] L. Sântean, Parallel communicating systems. *Bulletin EATCS* 42 (1990), 160 - 171.

[Sto 71] E. D. Stotskii, Control of the conclusion in formal grammars. *Problems of Information Transmission* 7, 3 (1971, translated 1973), 257 - 270.

[Vasz 96] Gy. Vaszil, Parallel communicating grammar systems without a master. *Computers and AI* 15, 2-3 (1996), 185 - 198.

[Wät 93] D. Wätjen, On cooperating distributed limited 0L systems. *J. Inform. Process. Cybern. EIK* 29 (1993), 129 - 142.

[Wood 73] D. Wood, Bicolored digraph grammar systems. *RAIRO* R-1 (1973), 45 - 50.