Investigations on Simple Eco-Grammar Systems

Judit Csima

Supervisor:
Dr. Erzsébet Csuhaj-Varjú

Eötvös Loránd University
Ph. D. Program in Computer Science
Foundations and Methodology of Computer Science
Director: Dr. János Demetrovics

Computer and Automation Research Institute
of the Hungarian Academy of Sciences

Budapest University of Technology and Economics
Department of Computer Science and Information Theory

Budapest, 2002

Contents

Table of contents **i**

Acknowledgement **iii**

1 Introduction **1**

1.1 Grammar systems theory 2

1.2 Eco-grammar systems 6

1.3 Structure of the dissertation 12

1.3.1 Team behaviour of variants of simple eco-grammar systems 12

1.3.2 Generative power of some variants of simple eco-grammar systems 15

1.3.3 Watson–Crick complementarity 16

1.4 Concluding remarks 18

2 Preliminaries **20**

2.1 Formal language prerequisites 20

2.2 Lindenmayer systems 22

2.3 Simple eco-grammar systems 23

3 Team behaviour of simple eco-grammar systems **28**

3.1 Definitions and previous results in the non-extended case . . . 29
3.2 Team behaviour in extended simple eco-grammar systems . . . 30

4 New operations in simple eco-grammar systems 43
4.1 Hybrid eco-grammar systems 45
4.2 Simple eco-grammar systems of type INS and REP 66

5 Generative power of variants of simple eco-grammar systems 80
5.1 Remarks on random-context grammars 81
5.2 Extended tabled simple eco-grammar systems 87
5.3 Weak extended simple eco-grammar systems 94

6 Watson–Crick systems 102
6.1 Definitions, examples 106
6.2 The main idea of the proofs 109
6.3 Standard Watson–Crick EDT0L systems 112
6.4 Standard Watson–Crick E0L systems 116
6.5 Standard Watson–Crick eco-grammar systems 124

Summary 130

Summary (in Hungarian) 131

Bibliography 132

List of figures 138

Acknowledgement

I am grateful to my supervisor Erzsébet Csuhaj-Varjú, who introduced me to the field of eco-grammar systems. Without her help and advice this work would not have been written.

I would like to thank SZTAKI, the Computer and Automation Research Institute of the Hungarian Academy of Sciences, for giving me the possibility of working there.

My thanks also go to my colleagues at the Department of Computer Science and Information Theory at the Budapest University of Technology and Economics for providing such a pleasant working atmosphere.

Special thanks to Dániel Marx for helping so much in the final phase of the work.

Chapter 1

Introduction

One of the main concept of traditional formal language theory is the notion of grammar, which can describe an infinite object, despite being finite. When the classical theory of formal grammars was elaborated, both the abstract and the real computational devices were centralised and based on the use of one processor only. By developments in computations, by the widespread use of parallel architectures and parallel processing a new approach has been raised in formal language theory as well which intends to study devices made up from independent components working together in order to generate a language.

Grammar systems theory defines languages through finite collections of formal grammars, cooperating and communicating with each other. In this dissertation the eco-grammar system model is investigated, which also provides a formal framework for describing emergent computation. This approach is aimed to obtain a better understanding of the properties of emergent systems by giving a theoretical basis for studying them.

In the following, first we review some fields of grammar systems theory and present the general ideas motivating them, emphasising those models and features which lead towards eco-grammar systems. Then we present our subject and summarise the results of our work.

1.1 Grammar systems theory

When one agent is unable to perform a complex task or tackle a specific problem due to its limited capabilities, it is natural to try to solve the problem by more than one agent. This idea motivates the notion of multi-agent system, which has two different approaches in Artificial Intelligence (AI): distributed and decentralised AI.

Distributed AI is concerned with the cooperative solution of problems. Here several agents interact with each other, following a given strategy; the communication and interaction protocols of the agents are well-defined in advance.

First models of grammar systems theory are mainly motivated by distributed AI. The idea of cooperating grammars was first presented in (Meersman and Rozenberg 1978), but the development of this area started only in 1988, when the notion of the cooperating distributed grammar system (CD grammar system, for short) was introduced in (Csuhaj-Varjú and Dassow 1990) and in (Csuhaj-Varjú and Kelemen 1989).

The concept of CD grammar systems was proposed as a syntactic model of the blackboard architecture of problem solving (see (Nii 1989)), where several independent agents work together on the solution of a problem by cooperating with each other only by modifying the common blackboard representing the current state of problem solving.

In a CD grammar system the agents are generative grammars and the global data base is represented by a string, called sentential form. The agents take turns in rewriting the sentential form according to a cooperation strategy, called derivation mode. The successful problem solving is achieved by generating a terminal word. CD grammar systems were the first formal language theoretical models formalising distributed computation. It demonstrated that complex behaviour, i.e. complex languages can be generated by simple grammars using a simple cooperation strategy. For an overview about CD grammar systems see (Csuhaj-Varjú, Dassow, Kelemen, and Păun 1994) and (Dassow, Păun, and Rozenberg 1997).

While CD grammar systems use sequential rewriting (in each derivation step only one grammar is active), parallel communicating grammar systems (PC grammar systems, for short) introduce parallelism into grammar systems theory, thus providing theoretical models for parallel, distributed computation. PC grammar systems are motivated by another problem solving model of distributed AI, the classroom model. In this model each agent can operate only on its own "notebook" and only one component, the master can use the blackboard. The agents work in parallel and they can communicate by sending their "notebooks" to each other.

In a PC grammar system (introduced in (Păun and Sântean 1989)) the components are generative grammars working on their own sentential forms in parallel and communicating with each other by sending their sentential forms by request. The language generated by the system consists of those words which can appear as the sentential form of the master. A good overview on PC grammar systems and several of the most important results about the role of synchronisation, different ways of communication, generative capacity, and size complexity can be found in (Vaszil 2000), (Csuhaj-Varjú and Salomaa 1998) and (Csuhaj-Varjú 1999).

Based on the general idea of grammar systems theory, several other models have been introduced and studied, for a short overall review see (Vaszil 2000) and (Csuhaj-Varjú 2000). We presented here the above two fundamental models, CD and PC grammar systems specially because eco-grammar systems, the subject of our study, can be viewed as a generalisation of both of them.

While the concept of CD and PC grammar systems is inspired by distributed AI, decentralised AI and Artificial Life (AL) give motivations for models like colony and eco-grammar system. Decentralised AI deals with the study of multi-agent systems of autonomous agents. In these systems the communication and cooperation is minimised, there is no predefined strategy (unlike in distributed AI), the properties of the system emerge only from the intensive interaction of the agents and the environment. AL studies man-made systems that exhibit behaviours characteric of natural living systems (see (Langton 1989)). Its models have similar properties to those of decentralised AI: they consist of a population of simple agents reacting on a common environment without any master component, which would direct their behaviour or coordinate their work. Life-like features are the result only of the interaction of the components and the environment.

The idea of emergence, that is, when intelligent, complex behaviour is resulted from the interaction of the agents with their shared environment, is common in these approaches and it is this paradigm, together with the general concept of grammar systems theory, that leads to models of colony and eco-grammar system.

The colony, (introduced in (Kelemen and Kelemenová 1992)), which is the first model describing the above ideas within formal language theory, is motivated mainly by reactive agents studied in robotics (see (Kelemen 1998) and (Brooks 1999)). This model retains the idea of CD and PC grammar systems of having grammars as components working together but in a colony these grammars are simple regular grammars generating finite languages and there is no cooperation between them. The model also has an environment component, which is represented by a string. The state of the system can be changed only by the actions of the agents; the environment is passive, it does not change its state autonomously. The agents can alter the string representing the environment by sequential rewritings. Due to the lack of any predefined strategy, each grammar participates in the rewriting whenever it can, conflicts are resolved nondeterministically. The language generated by the colony is the set of all the possible states of the environment.

Several derivation modes and acceptance styles were introduced and studied (see (Csuhaj-Varjú and Kelemenová 1994), (Dassow, Kelemen, and Păun 1993) and (Csuhaj-Varjú 1999) for formal definitions and results) and colonies proved to show emergent behaviour indeed: the components are very simple regular grammars, but powerful, large language classes can be generated in this way.

Although the colony formalises several features of multi-agent systems, for better understanding agents behaving in a dynamic environment this model is not suitable because its environment component is quite passive. To eliminate this shortcoming, a new model was created, the eco-grammar system, which is provided with an active environment component as well. The original eco-grammar system differs from colonies in the nature of the agents, too: here the agents are not purely reactive entities, they have an inner representation both for the environment and also for themselves. Eco-grammar systems are introduced in detail in the next section.

Grammar systems theory provides and studies several different models. It proves that language can be considered a collective product of several components and that key notions of present-day computation, like cooperation, communication and emergent behaviour can be formalised within its framework. By studying its models we obtain a better understanding of the phenomena of distributed computation and distributed systems. See (Dassow, Păun, and Rozenberg 1997) for an overview of the theory until 1996 and (Csuhaj-Varjú 2000) for a more recent, short summary. The reader interested in the motivations of grammar systems theory is referred to (Kelemen 1999).

1.2 Eco-grammar systems

Eco-grammar systems (EG systems, for short), as presented in (Csuhaj-Varjú, Kelemen, Kelemenová, and Păun 1994) and (Csuhaj-Varjú, Kelemen, Kelemenová, and Păun 1997), were introduced to formalise the concept of ecosystems consisting of several agents living together in a common environment. The model assumes that both the agents and also the environment evolve, the environment independently of the agents, the agents dependently on the environment. Moreover the agents can sense the environment and can change it.

The original model was based on the following six postulates:

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.

2. In an ecosystem there is a synchronised universal clock, which divides the time in units. In each time unit the environment and the agents together perform a derivation step.

3. Both the environment and the agents have evolution rules, which are rules of Lindenmayer systems, hence are applied in a parallel manner to all the symbols describing the states of the agents and of the environment; 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 depend on the state of the environment (at a given moment, a subset of the rules is chosen from a general set associated to each agent).

5. The agents act on the environment according to action rules, which are rewriting rules used in context-free manner. In each time unit, each agent uses one action rule, chosen from a set depending on the current state of the agent.

6. The action (of agents on the environment) has priority over the evolution of the environment; in a given time unit, those symbols of the environment which are not affected by the actions of the agents are rewritten (in parallel manner) by the evolution rules.

According to these postulates, an eco-grammar system can be imagined as presented in Figure 1.2.

The environment is described by a string w_E and evolves according to a set of rules P_E of a Lindenmayer system. The agent A_i is described by a string w_i and evolves according to the rules of the set P_i. At every moment, only a subset of these rules is active, this subset is chosen by φ_i depending on w_E, the string of the environment. Each agent A_i is provided by a set R_i of rewriting rules, too, by which it can modify the environment in a sequential

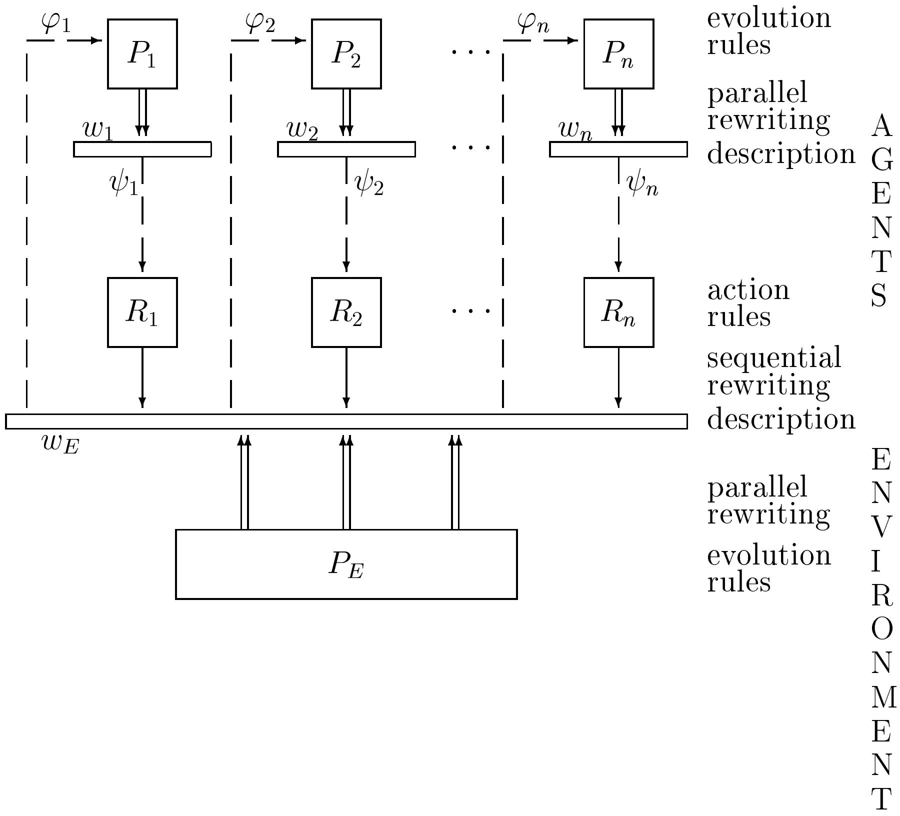

Figure 1.1: Original version of eco-grammar systems

manner. At each step, only a subset of R_i is active, depending on the current state of the agent. This subset is chosen by the function ψ_i.

The life of the system is governed by a universal clock, dividing the time in unit intervals: in each step first the agents act on the environment (each agent rewrites one of the symbols), then the evolution rules of the environment rewrites in parallel all the remaining symbols of the string of the environment. The strings of the agents are also rewritten in a parallel way, according to the chosen subsets of the sets P_i.

In this model both the environment and the agents have inner representations and the interaction is quite sophisticated between the different components. The difference between action and evolution (presented as sequential and parallel rewriting in the model) is due to the fact that evolution is a global, parallel process, while the actions of the agents are local. The behaviour of the whole system is represented by the environment's sentential form, thus the generated language consists of these strings.

Although motivated mainly by the ecosystem concept of Artificial Life, the eco-grammar system is a formal language theoretical model, being within the framework of grammar systems theory. Not only this model is a developed variant of colonies (as presented in the previous section), but it keeps some properties of both CD and PC grammar systems. It has only one sentential form (the string of the environment is the only one which determines the generated language) and this word can be changed by the component grammars like in a CD grammar system. However, the components can evolve in parallel, which is a feature common with PC grammar systems.

In (Csuhaj-Varjú, Kelemen, Kelemenová, and Păun 1997) eco-grammar systems were studied from different points of view. It was found there that from artificial life point of view, i.e. when the life-cycle and the derivation sequences are investigated, the evolution of the system can be extremely complex. From generative point of view it was proved that all recursively enumerable languages can be generated by eco-grammar systems with a very simple choice of the functions φ_i and ψ_i. These two facts together pointed in the direction of simplifying the original definition and led to simple eco-grammar systems.

The simple eco-grammar system model preserves the dynamic environment from the original definition, but otherwise the concept of the agent is similar to the one presented in colonies: the agents do not evolve and their acting on the environment is independent of the state of the system. This means that the functions φ_i, ψ_i and the inner representations of the agents can be omitted, thus getting the picture represented in Figure 1.2.

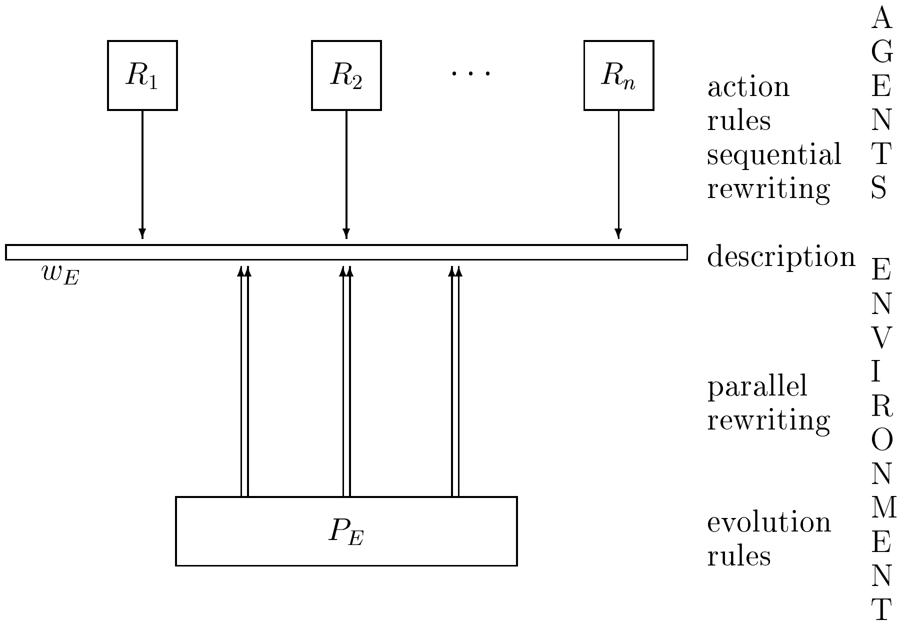

Figure 1.2: Simple eco-grammar system

Briefly, a simple eco-grammar system consists of several agents, represented by sets of context-free rules and an environment, given by a set of rules of a Lindenmayer system. The state of the system is described by the state of the environment, which is a string over the alphabet of the system. The environmental state changes by derivation steps. In a derivation step the agents act on the string by applying one of their context-free rules—each agent rewrites one letter—and the environment replaces those symbols where the agents do not perform any action, according to its set of rules, in a parallel way.

The advantages of simple eco-grammar systems are two-fold: on the one hand, simple eco-grammar systems have a closer connection with the ideas which motivate colonies, thus by examining it we better understand the ideas of emergent computations, while on the other hand we have a model which is not too strong in the generative sense but at the same time provides interesting features.

The subject of this dissertation is the study of simple eco-grammar systems. In the next section we summarise our results and settle them among the other works done on this field. Before doing so, we mention some models and fields grown from the original definition of eco-grammar system.

At the beginning, besides the study of generative power, eco-grammar systems were examined from artificial life point of view (see (Csuhaj-Varjú, Kelemen, Kelemenová, and Păun 1997) and (Csuhaj-Varjú 1995)). Developed models such as symbiotic eco-grammar systems, stratified eco-grammar systems, and reproductive eco-grammar systems were proposed to formalise the notions of population dynamics: symbiosis, parasitism, competition and reproduction.

Recently, eco-grammar systems have been also used to model natural languages (see (Csuhaj-Varjú 1996)) and to describe phenomenon of linguistics (see (Csuhaj-Varjú, Jiménez-López, and Martín-Vide 1999) and (Csuhaj-Varjú and Jiménez-López 1998)). For a thorough study of this subject, the reader is referred to (Jiménez-López 2000), which well demonstrates how grammar systems theory and especially eco-grammar systems can be used to model natural languages.

There is another variant of the model, which was studied intensively from formal language theoretical point of view, the conditional tabled eco-grammar system. It was intended to describe the interplay between an environment and some agents with non-local influence. In this model not only the evolution of the agents depend on the environment but the evolution of the environment depends on the agents as well. For technical details and result see (Csuhaj-Varjú, Păun, and Salomaa 1995) and (Sosik 1999).

1.3 Structure of the dissertation

The aim of this work is to examine how the behaviour of a simple eco-grammar system is affected by modifying some of its parameters or components. We concentrate on the generative power and study both the original model and also its different variants from this point of view.

In this section we display some of the works which have been done in the field, thus providing a context for our results, which we also summarise.

1.3.1 Team behaviour of variants of simple eco-grammar systems

As presented in the previous section, the original eco-grammar system model proved to have universal generative power. Simple eco-grammars systems were also examined from this point of view, one of the first works in this field was the study of the so called "team behaviour".

The concept of a team, which was introduced into grammar systems theory in (Kari, Mateescu, Păun, and Salomaa 1995), appears in eco-grammar systems theory in two different forms. Prescribed teams are discussed in (ter Beek 1999): in this case the definition of the system contains the specification of those groups of agents which can work together in a derivation step. This team concept is discussed later.

Another approach is to consider team derivation modes. In the original definition of simple eco-grammar systems all the agents have to work, otherwise the derivation is blocked. Derivation mode $= k$ modifies this definition in a natural way: it prescribes that in each derivation step exactly k agents work. Thus, in this derivation mode any k agents can form a team and can work together.

The first investigations concerning the derivation mode $= k$ were done about non-extended systems where all the generated words are in the corresponding language. The results appeared in (Csuhaj-Varjú and Kelemenová 1998) and showed that the parameters, the number of the agents in the system and the number of the agents working in a derivation step, are very important. In Chapter 3 first we review these results, then we deal with the extended case, which was examined in (Csima 1998a). In an extended simple eco-grammar system the generated strings are filtered: only words over a distinguished sub-alphabet are in the corresponding language. It is proved that, unlike the non-extended case, in the extended case the parameters are not important: we obtain a collapsing hierarchy.

In Chapter 4 we continue our investigations on the team behaviour but we consider simple eco-grammar systems with agents using new operations, motivated by genetics: insertion and replacement. An insertion rule can insert a word into the sentential form if both the left and the right context is present somewhere in the string. By a replacement rule, the agent can change a letter if the prescribed neighbouring context is present.

We combine these operations with the concept of simple eco-grammar systems in two different ways. We can consider simple eco-grammar systems of type INS and REP, where the agents are homogeneous or we can study a more complicated model by allowing the possibility of different agents having different types of rules. These models intend to examine nature-motivated decentralised systems.

For hybrid eco-grammar systems we define the derivation modes $= k$ and the $= k_1, = k_2$. In the $= k$ case we prescribe how many agents (chosen freely without any further restriction) have to work in one derivation step, while in the derivation mode $(= k_1, = k_2)$ the number of INS-type and REP-type agents is also specified.

In the first part of Chapter 4 the relation of language classes generated by hybrid eco-grammar systems with different parameters is studied as presented in (Csima 1999). It is proved that the number of agents working in one step is a significant parameter, while the number of agents being in the system is not.

Simple eco-grammar systems of type INS and REP are special cases of hybrid eco-grammar systems, therefore some techniques used for hybrid systems can be applied to homogeneous systems as well. Although in this way the results about INS- and REP-type systems can be obtained easily, in the second part of the chapter we present these results in detail. The reason to do this is to emphasise the striking similarity between the original case, i.e. when the agents are represented by CF rules and this case, when the agents use insertions and replacements.

At the end of Chapter 4 we compare language classes generated by homogeneous CF-, INS-, or REP-type simple EG systems and we prove that these language classes are pairwise incomparable.

1.3.2 Generative power of some variants of simple eco-grammar systems

While in Chapter 3 and Chapter 4 we study generative power by comparing language classes generated with different parameters, in Chapter 5 we introduce two variants of the original model and we show that the generative power of Turing machines can be achieved in both cases.

In Chapter 3 we found that the hierarchy of extended simple eco-grammar language classes collapses into one language class, but the place of this language class remained an open problem. From (Dassow and Mihalache 1995) we know that it is between two well-known language classes, the class of languages generated by extended Lindenmayer systems (E0L) and the class of languages generated by matrix grammars with appearance checking with λ-rules. It is also known from an example that this language class contains at least one non-E0L language.

While trying to determine the generative power of this language class more precisely, we found two different modifications of simple eco-grammar systems which have universal generative power. Our achievements improve the results of (ter Beek 1999), where the generative power of extended tabled simple eco-grammar systems with prescribed teams was examined in two different derivation modes. By the strong derivation all the agents of a team must work in a derivation step, by the weak rewriting only those agents of the team have to work which have rules applicable for the sentential form. Extended tabled simple eco-grammar systems with teams of agents with prescribed members and operating according to a weak derivation mode proved to be very powerful devices: it was proved in (ter Beek 1999) that they can generate all recursively enumerable languages. As far as the strong derivation mode is concerned, it was stated as an open problem whether universal power can be achieved in this case, too.

In Chapter 5 we consider two variants of extended simple eco-grammar systems, neither of which contain prescribed teams: extended tabled simple eco-grammar systems and weak extended simple eco-grammar systems. In an extended tabled simple eco-grammar system the environment can be represented by more than one set of rules, called tables, thus the environment can vary its behaviour step by step. In weak extended simple eco-grammar systems the definition of the derivation step is modified: during the computation the derivation is not blocked if some of the agents cannot perform any action on the sentential form.

We solve the open problem presented in (ter Beek 1999) in a stronger form by proving that extended tabled simple eco-grammar systems without prescribed teams can generate all recursively enumerable languages by using the original, i.e. the strong derivation mode. We also prove that weak extended simple eco-grammar systems (without tables and without prescribed teams) achieve the same generative power.

1.3.3 Watson–Crick complementarity

In Chapter 6, we study models which combine formal language theory and Watson–Crick complementarity, a fundamental concept of DNA computing.

For a longer introduction into DNA computing see the introductory part of Chapter 6, here we simply note that the effectivity of computing by DNA is thought to originate from two properties: massive parallelism and the phenomenon of Watson–Crick complementarity, which motivates the intensive research done on Watson–Crick complementarity in formal language theory.

From our point of view Watson–Crick complementarity provides a new operation. For each letter of the alphabet there exists a complementary symbol, thus for each word there exists a complementary word. In a Watson–Crick system during the derivation, when some conditions hold on the computed string, the computation continues from the complementary string. It is Watson–Crick complementarity that makes it possible to suppose that this complementary word is always present.

Our main goal is to examine the effect of introducing Watson–Crick complementarity into the framework of eco-grammar systems, in this chapter however, we do not restrict our investigation to the eco-grammar system models but study Lindenmayer systems as well. We follow this approach because the techniques presented for Lindenmayer systems, with only a slight modification, are also useful when combining simple eco-grammar systems and Watson–Crick complementarity.

Watson–Crick D0L system, the first model in formal language theory using Watson–Crick complementarity as an operation, was introduced and first studied in (Mihalache and Salomaa 1997) and (Mihalache and Salomaa 2001). This model raises a lot of interesting questions for study, such as the stability, the periodicity, and the growth functions of these systems. For further information see (Salomaa 1998), (Salomaa 1999), (Salomaa 2001a), (Salomaa 2001b), (Honkala and Salomaa 2001), (Csuhaj-Varjú and Salomaa 2000).

The computational capacity of these constructs is also among the particularly important problems. In (Sosik 2001) it was shown that any Turing computable function can be computed by a Watson–Crick D0L system.

In Chapter 6 first we also study the problem of the generating power of several variants of standard Watson–Crick Lindenmayer systems: we study standard Watson–Crick tabled EDT0L systems and standard Watson–Crick E0L systems. In the first case the environmental component of the system is represented by more than one deterministic Lindenmayer set, in the latter the environment has only one set, but this set is not necessarily deterministic. The adjective "standard" refers, in both cases, to the mechanism triggering the complementary transitions during the derivation. We show that all recursively enumerable languages can be generated by both types of systems.

Although these results are about Lindenmayer systems and not about eco-grammar systems, they are presented however because the constructions of their proofs are useful when examining standard Watson–Crick eco-grammar systems, which emerges from extended simple eco-grammar systems by adding the general idea of Watson–Crick complementarity. The functioning of a standard Watson–Crick eco-grammar system is similar to the original (i.e. non-Watson–Crick) case, with the only difference that when during the computation the sentential form satisfies some requirements, it is replaced by its complementary string. We prove that standard Watson–Crick eco-grammar systems with only one agent are able to generate all recursively enumerable languages.

1.4 Concluding remarks

The aim of our research was to study simple eco-grammar systems and to answer questions raised within the framework of grammar systems theory. Since eco-grammar system models are intended to formalise notions of emergent computation and features of living systems, by better understanding these models we hope to understand those notions and features as well.

Our work on the role of the parameters in team derivation modes helps us to perceive the important parts of the model. By proving that several variants of simple eco-grammar systems are computationally complete tools, on the one hand powerful, simple models are proposed for possible applications, while on the other hand the ways of future research is appointed, by closing several directions.

Chapter 2

Preliminaries

In this chapter we present the formal language theoretical notions and notations we use in the dissertation, we present the basic definitions concerning Lindenmayer systems, and finally we introduce the model we are going to study: the simple eco-grammar system.

The definitions concerning Watson–Crick complementarity, the subject of the last chapter, can be found in Chapter 6 because that is the only part of the dissertation where we use them.

2.1 Formal language prerequisites

We present the basic notions and notations used in the dissertation, for further information the reader is referred to (Salomaa 1973), (Rozenberg and Salomaa 1980) and (Dassow and Păun 1989).

The set of all non-empty words over a finite alphabet V is denoted by V^+, the empty word is denoted by λ; $V^* = V^+ \cup \{\lambda\}$. For a set V, we denote by $\mathrm{card}(V)$ or by $|V|$ the cardinality of V. For a word x, we denote by $|x|$ the length of x. If $a \in V$ and x is a word over V, then $|x|_a$ denotes the number of letters a in x. When $V_1 \subseteq V$ and $x \in V^*$, then $|x|_{V_1} = \sum_{a \in V_1} |x|_a$.

If V is an alphabet, then $V^{(2)}$, V', $[V, i, j]$, and $[V, j]$ denote the sets $\{x^{(2)} \mid x \in V\}$, $\{x' \mid x \in V\}$, $\{[x, i, j] \mid x \in V\}$, and $\{[x, j] \mid x \in V\}$ for $1 \leq i, j$, respectively.

If $x = x_1 \cdots x_n$ is a word over an alphabet V, where $x_j \in V$ for $1 \leq j \leq n$, then $[x, i, k]$ and $[x, i]$ denote the words $[x_1, i, k] \cdots [x_n, i, k] \in [V, i, k]^*$ and $[x_1, i] \cdots [x_n, i] \in [V, i]^*$, respectively, for $i, k \geq 1$; x' and $x^{(2)}$ denote the words ${x_1}' \cdots {x_n}' \in V'^*$ and $x_1^{(2)} \cdots x_n^{(2)} \in {V^{(2)}}^*$.

Let V be an alphabet and $x, y \in V^*$. We use the notation $x \preceq y$ if x is a sub-word of y, that is, if $y = x_1 x x_2$ with some words $x_1, x_2 \in V^*$.

By a permutation of the words $w_1, \ldots, w_n$, written as $[w_1, w_2, \ldots, w_n]^P$, where $w_i \in V^*$ for $1 \leq i \leq n$, we mean a word in the form of $w_{i_1} w_{i_2} \cdots w_{i_n}$, where $i_{j_1} \neq i_{j_2}$ if $j_1 \neq j_2$ and $1 \leq i_j \leq n$ for $1 \leq j \leq n$.

By a language over an alphabet V we mean a subset L of V^*. We denote by $\mathrm{alph}(L)$ the set of all letters appearing in at least one word of L. For a word w, $\mathrm{alph}(w)$ denotes the set of all letters appearing in w.

By a context-free production or by a context-free rule (a CF rule, for short) over an alphabet V we mean a production in the form of $a \rightarrow u$, where $a \in V$ and $u \in V^*$. A CF rule is a λ-rule (or a deletion rule) if $u = \lambda$.

For a set of CF rules R, $\mathrm{dom}(R)$ denotes the set of all letters appearing on the left-hand side of a rule in R.

2.2 Lindenmayer systems

A 0L system is a triplet $H = (V, P, \omega)$, where V is a finite alphabet, P is a set of context-free rules over V, and $\omega \in V^*$ is the axiom. Moreover, P has to be complete, that is, for each symbol a in V there must be at least one rule $a \to x$ in P with this letter a on the left-hand side.

0L systems use parallel derivations: we say that x directly derives y in a 0L system $H = (V, P, \omega)$, with $x, y \in V^*$, written as $x \overset{\text{0L}}{\Longrightarrow}_H y$, if $x = x_1 x_2 \cdots x_n$, $y = y_1 y_2 \cdots y_n$, where $x_i \in V$, $y_i \in V^*$, and the rules $x_i \to y_i$ are in P for $1 \leq i \leq n$.

A T0L system is a triplet $H = (V, T, \omega)$, where V is a finite alphabet, $T = \{T_1, \ldots, T_k\}$ is a finite set of tables over V, where each table T_i for $1 \leq i \leq k$ is a complete set of CF rules over V, and $\omega \in V^*$ is the axiom. We say that x directly derives y in a T0L system $H = (V, T, \omega)$, with $x, y \in V^*$, written as $x \overset{\text{T0L}}{\Longrightarrow}_H y$, if $x \overset{\text{0L}}{\Longrightarrow}_{H_i} y$ for some i , $1 \leq i \leq k$, with the 0L system $H_i = (V, T_i, \omega)$.

An ET0L system is a quadruple $H = (V, T, \omega, \Delta)$, where $\overline{H} = (V, T, \omega)$ is a T0L system, and $\emptyset \neq \Delta \subseteq V$ is a subset of V, the terminal alphabet. In an ET0L system $H = (V, T, \omega, \Delta)$ x directly derives y, with $x, y \in V^*$, written as $x \overset{\text{ET0L}}{\Longrightarrow}_H y$, if $x \overset{\text{T0L}}{\Longrightarrow}_{\overline{H}} y$.

The transitive and reflexive closure of $\overset{\text{ET0L}}{\Longrightarrow}_H$ is denoted by $\overset{\text{ET0L}}{\Longrightarrow}{}^*_H$. The generated language of the ET0L system H (denoted by $L(H)$) is

$$L(H) = \{\ w \in \Delta^* \mid \omega \overset{\text{ET0L}}{\Longrightarrow}{}^*_H\ w\ \}.$$

Thus in an ET0L system only words over a distinguished sub-alphabet are in the generated language. A language is said to be an ET0L language if there is an ET0L system generating it.

T0L and 0L systems are special cases of ET0L systems: $\Delta = V$ stands in both cases; moreover, in the case of 0L systems $T = \{T_1\}$ also holds. In an E0L system there is only one table but Δ is not necessarily equal to V. Therefore the above definition gives the generated language for these systems as well.

2.3 Simple eco-grammar systems

We give only the definition of simple eco-grammar systems. The formal definition of the original (i.e. non-simple) version, which was introduced in (Csuhaj-Varjú, Kelemen, Kelemenová, and Păun 1997) is not presented, because in this dissertation we study only simple eco-grammar systems. For more details about the original version see (Csuhaj-Varjú, Kelemen, Kelemenová, and Păun 1997).

Definition 2.1

A simple eco-grammar system (a simple EG system, for short) is a construct $\Sigma = (V_E, P_E, R_1, \ldots, R_n, \omega)$*, where*

- V_E *is a finite, non-empty alphabet,*
- P_E *is a complete set of CF rules over* V_E*,*
- $n \geq 1$*,*
- R_i *is a non-empty set of CF rules over* V_E *for* $1 \leq i \leq n$*, and*
- $\omega \in {V_E}^*$*.*

In this construct V_E is the alphabet and P_E is the set of evolution rules of the environment. The ith agent is represented by the set R_i, $1 \leq i \leq n$. String ω is the axiom, the initial sentential form.

The system changes its state by a simultaneous action of the agents and by a parallel rewriting done according to P_E: in each derivation step first each agent rewrites one letter in the sentential form, then the environment rewrites the remaining letters in a parallel way.

Definition 2.2

Consider a simple eco-grammar system $\Sigma = (\, V_E, P_E, R_1, \ldots, R_n, \omega\,)$. We say that x directly derives y in Σ (with $x, y \in {V_E}^*$, written as $x \overset{eco}{\Longrightarrow}_\Sigma y$), if

- $x = x_1 Z_1 x_2 Z_2 \cdots x_n Z_n x_{n+1}$, *with* $Z_i \in V_E$, $x_j \in {V_E}^*$, $1 \leq i \leq n$, *and* $1 \leq j \leq n+1$,
- $y = y_1 w_1 y_2 w_2 \cdots y_n w_n y_{n+1}$, *with* $y_i, w_j \in {V_E}^*$, $1 \leq i \leq n$, *and* $1 \leq j \leq n+1$,
- *there exists a permutation of the agents, namely* $R_{j_1}, R_{j_2}, \ldots, R_{j_n}$, *such that* $Z_i \rightarrow w_i \in R_{j_i}$, *for* $1 \leq i \leq n$, *and*
- $x_1 \cdots x_{n+1} = y_1 \cdots y_{n+1} = \lambda$ *or* $x_1 \cdots x_{n+1} \overset{0L}{\Longrightarrow}_E y_1 \cdots y_{n+1}$ *where* $E = (V_E, P_E, \omega)$ *is the 0L system of the environment.*

Let us note here that according to this definition, all the agents have to work in each derivation step. In (ter Beek 1999) and in Chapter 5 another version of simple eco-grammar systems is studied, the so called weak simple eco-grammar system. In this case the derivation is not blocked if some of the agents cannot perform any action on the sentential form.

We denote the transitive and reflexive closure of $\overset{\text{eco}}{\Longrightarrow}_\Sigma$ by $\overset{\text{eco}}{\Longrightarrow}{}^*_\Sigma$. The generated language consists of all the words which can be derived from the initial string.

Definition 2.3

Consider a simple eco-grammar system $\Sigma = (V_E, P_E, R_1, \ldots, R_n, \omega)$. *The generated language of* Σ *is the following:*

$$L(\Sigma) = \{ v \in {V_E}^* |\, \omega \stackrel{eco}{\Longrightarrow}{}_{\Sigma}^{*} v \}.$$

The extended version of simple eco-grammar systems, where a subset of the alphabet, called the terminal alphabet is distinguished, was introduced in (Csuhaj-Varjú, Kelemen, Kelemenová, and Păun 1997).

Definition 2.4

An extended simple eco-grammar system is a construct $\Sigma = (V_E, P_E, R_1, \ldots, R_n, \omega, \Delta)$, *where*

- V_E, P_E , n, R_i *for* $1 \leq i \leq n$, *and* ω *are the same as in Definition 2.1, that is, the alphabet of the system, the rule-set of the system, the number of the agents* ($n \geq 1$), *the rule-sets of the agents and the axiom, and*
- Δ *is a non-empty subset of* V_E, *the terminal alphabet.*

In an extended simple eco-grammar system, the derivation goes in the same way as presented in Definition 2.2, the only difference is that the generated language contains only words over the terminal alphabet.

Definition 2.5

Consider an extended simple eco-grammar system $\Sigma = (V_E, P_E, R_1, \ldots, R_n, \omega, \Delta)$. *The generated language of* Σ *is the following:*

$$L(\Sigma) = \{ v \in \Delta^* |\, \omega \stackrel{eco}{\Longrightarrow}{}_{\Sigma}^{*} v \}.$$

Now we present two examples which illustrates our model and its power.

Example 2.6

Let $\Sigma = (\ V_E, P_E, R_1, R_2, R_3, \omega, \Delta\ )$ *be the following extended simple EG system:*

- $V_E = \{\ A, B, C, a, b, c\ \}$,
- $P_E = \{\ A \to a, B \to b, C \to c, a \to a, b \to b, c \to c\ \}$,
- $R_1 = \{\ A \to A^2, A \to a\ \}$,
- $R_2 = \{\ B \to B^2, B \to b\ \}$,
- $R_3 = \{\ C \to C^2, C \to c\ \}$,
- $\omega = ABC$, *and*
- $\Delta = \{\ a, b, c\ \}$.

It is easy to see that the generated language is a non-context-free language: $L(\Sigma) = \{\ a^n b^n c^n \mid n \geq 1\ \}$.

Example 2.7

Let $\Sigma = (\ V_E, P_E, R_1, R_2, R_3, \omega, \Delta\ )$ *be the following extended simple EG system:*

- $V_E = \{\ S_1, S_2, S_3, a, b, c, K\ \}$,
- $P_E = \{\ a \to a, b \to b, c \to K, S_1 \to K, S_2 \to K, S_3 \to K, K \to K\ \}$,
- $R_1 = \{\ S_1 \to S_1 a, S_1 \to S_2 b, S_1 \to c\ \}$,
- $R_2 = \{\ S_2 \to S_2 a, S_2 \to S_3 b, S_2 \to c\ \}$,
- $R_2 = \{\ S_3 \to S_3 a, S_3 \to S_1 b, S_3 \to c\ \}$,
- $\omega = S_1 S_2 S_3$, *and*
- $\Delta = \{\ a, b, c\ \}$.

When deriving a terminal word, in a derivation step we have only three possibilities: we can use the first rules of the agents together or we can use the second ones or the third ones, otherwise the derivation is blocked because in the next step one of the agents could not work. Therefore either all the agents introduces the same letter a or b or the agents finish the derivation by bringing in a c. This implies that the generated language is $L(\Sigma) = \{ (cw)^3 \mid w \in \{a,b\}^* \}$, which is a non-$E0L$ language (see (Ehrenfeucht and Rozenberg 1976)).

Chapter 3

Team behaviour of simple eco-grammar systems

As presented in Chapter 1, the generative power of eco-grammar systems was studied in several papers. In the original model of simple eco-grammar systems all the agents have to work in each derivation step, otherwise the derivation is blocked. In this section a different derivation mode is examined: in each derivation step exactly k agents have to work.

First we summarise the results about the relation between language classes generated by simple eco-grammar systems with n agents working in derivation mode $= k$. These results appeared in (Csuhaj-Varjú and Kelemenová 1998), and were summarised and completed in (Csima 1997), where non-extended simple eco-grammar systems were studied.

The notion of extended simple eco-grammar systems was introduced in (Csuhaj-Varjú, Kelemen, Kelemenová, and Păun 1997) and was examined first in (Dassow and Mihalache 1995) under the name of 0-terminal EG system. In the second part of the chapter we study the derivation mode $= k$ in extended systems. The results, which appeared in (Csima 1998a), show that unlike the non-extended case, where the size parameters of the teams and the agent population have influence on the power of the system, in the extended case a collapsing hierarchy is obtained.

3.1 Definitions and previous results in the non-extended case

The first team derivation mode which was studied for simple eco-grammar systems is the derivation mode $= k$, where in each step exactly k agents have to work.

Definition 3.1

Consider a simple eco-grammar system $\Sigma = (\, V_E, P_E, R_1, \ldots, R_n, \omega\,)$.
We say that x *directly derives* y *in* Σ *in derivation mode* $= k$ *(with* $x, y \in {V_E}^*$ *and* $1 \leq k \leq n$*, written as* $x \overset{=k}{\Longrightarrow}_\Sigma y$*), if*

- $x = x_1 Z_1 x_2 Z_2 \cdots x_k Z_k x_{k+1}$*, with* $Z_i \in V_E$*,* $x_j \in {V_E}^*$*,* $1 \leq i \leq k$*, and* $1 \leq j \leq k+1$*,*
- $y = y_1 w_1 y_2 w_2 \cdots y_k w_k y_{k+1}$*, with* $y_i, w_j \in {V_E}^*$*,* $1 \leq i \leq k$*, and* $1 \leq j \leq k+1$*,*
- *there exist* k *different agents in* Σ*, namely* $R_{j_1}, R_{j_2}, \ldots, R_{j_k}$*, such that* $Z_i \to w_i \in R_{j_i}$ *for* $1 \leq i \leq k$*, and*
- $x_1 \cdots x_{k+1} = y_1 \cdots y_{k+1} = \lambda$ *or* $x_1 \cdots x_{k+1} \Rightarrow_E y_1 \cdots y_{k+1}$*, where* $E = (V_E, P_E, \omega)$ *is the 0L system of the environment.*

We denote the transitive and reflexive closure of $\overset{=k}{\Longrightarrow}_\Sigma$ by $\overset{=k}{\Longrightarrow}{}^*_\Sigma$. The generated language consists of those words which can be generated from the axiom in some derivation steps.

Definition 3.2

Consider a simple eco-grammar system $\Sigma = (\, V_E, P_E, R_1, \ldots, R_n, \omega\,)$. The generated language in the derivation mode $= k$ is the following:

$$L(\Sigma, = k) = \{\, v \in {V_E}^* \,|\, \omega \overset{=k}{\Longrightarrow}{}^*_\Sigma \, v \,\}.$$

In the following theorem, we summarise the results which are presented in (Csuhaj-Varjú and Kelemenová 1998) and (Csima 1997), where relations between language classes generated by simple eco-grammar systems with different parameters n and k were examined. (The results of (Csuhaj-Varjú and Kelemenová 1998) are summarised and completed in (Csima 1997).) Let $\mathcal{L}(E(n, = k))$ denote the class of languages generated by a simple EG system containing n agents and working in the derivation mode $= k$.

Theorem 3.3

For $1 \leq k \leq n$ and $1 \leq \ell \leq m$

1. $\mathcal{L}(E(n, = k)) \subset \mathcal{L}(E(m, = k))$ *if* $k \geq 2$ *and* $m > n$,
2. $\mathcal{L}(E(n, = 1)) = \mathcal{L}(E(m, = 1))$,
3. *otherwise* $\mathcal{L}(E(n, = k))$ *and* $\mathcal{L}(E(m, = \ell))$ *are incomparable.*

3.2 Team behaviour in extended simple eco-grammar systems

In an extended simple eco-grammar system we distinguish a subset of the alphabet, and only words over this terminal alphabet are in the generated language.

The definition of the derivation mode $= k$ is the same as in Definition 3.1, the difference lies in the definition of the generated language.

Definition 3.4

Consider an extended simple eco-grammar system $\Sigma = (\, V_E, P_E, R_1, \ldots, R_n, \omega, \Delta\,)$. *The generated language in the derivation mode* $= k$ *is the following:*

$$L(\Sigma, = k) = \{\, v \in \Delta^* |\, \omega \overset{=k}{\Longrightarrow}{}^*_{\Sigma}\, v \,\}.$$

In this section we study extended simple eco-grammar systems with or without λ-rules. When we allow λ-rules in P_E and in the sets R_i, we use the following notations: the class of languages generated by extended simple eco-grammar systems is denoted by $\mathcal{L}(EE^{\lambda})$ (in this notation the first E means "extended", the second one refers to "eco"); the class of languages generated by systems containing n agents and operating in the derivation mode $= k$ is denoted by $\mathcal{L}(EE^{\lambda}(n, = k))$. We omit the notation λ if λ-rules are not allowed neither in P_E nor in the sets R_i: $\mathcal{L}(EE)$ and $\mathcal{L}(EE(n, = k))$.

We present results about the hierarchy of language classes generated by n agents in the derivation mode $= k$. Most of the statements and proofs are true with or without λ-rules, this fact is denoted by the superscript (λ); sometimes a statement is true only when λ-rules are allowed, in this case we will emhasize this fact.

First we examine the role of the first parameter: the number of agents.

Lemma 3.5

For $1 \leq k \leq n \leq m \quad \mathcal{L}(EE^{(\lambda)}(n, = k)) \subseteq \mathcal{L}(EE^{(\lambda)}(m, = k))$.

PROOF.

We show that for any $1 \leq k \leq n \leq m$ and for any extended simple EG system $\Sigma = (\, V_E, P_E, R_1, \ldots, R_n, \omega, \Delta\,)$ there exists another extended simple EG system $\overline{\Sigma} = (\, \overline{V_E}, \overline{P_E}, \overline{R_1}, \ldots, \overline{R_m}, \overline{\omega}, \overline{\Delta}\,)$, such that their generated languages are equal, that is, $L(\Sigma, = k) = L(\overline{\Sigma}, = k)$. Moreover, if Σ does not contain

λ-rules, then $\overline{\Sigma}$ is λ-free, too.

Let $\overline{\Sigma}$ be the following:

- $\overline{V_E} = V_E \cup \{K\}$, where $K \notin V_E$,
- $\overline{P_E} = P_E \cup \{K \to K\}$,
- $\overline{R_i} = R_i \quad$ for $1 \le i \le n$,
- $\overline{R_i} = \{K \to K\} \quad$ for $n+1 \le i \le m$,
- $\overline{\omega} = \omega$, and
- $\overline{\Delta} = \Delta$.

It is clear that $L(\Sigma, = k) = L(\overline{\Sigma}, = k)$ and $\overline{\Sigma}$ is λ-free if and only if Σ does not contain λ-rules. ∎

The following result shows that the above inclusion is not proper. The statement is true both with or without λ-rules.

Lemma 3.6

For $1 \le k \le n \quad \mathcal{L}(EE^{(\lambda)}(n+1, = k)) \subseteq \mathcal{L}(EE^{(\lambda)}(n, = k))$.

PROOF.

Without λ-rules

We show that for any λ-free extended simple EG system $\Sigma = (V_E, P_E, R_1, \ldots, R_{n+1}, \omega, \Delta)$ working in the derivation mode $= k$ we can construct another λ-free extended simple EG system $\overline{\Sigma} = (\overline{V_E}, \overline{P_E}, \overline{R_1}, \ldots, \overline{R_n}, \overline{\omega}, \overline{\Delta})$ working also in the derivation mode $= k$ simulating Σ.

Let $\overline{\Sigma}$ be the following system:

- $\overline{V_E} = V_E \cup {V_E}^{(2)} \cup {V_E}' \cup \{K\}$, where $K \notin V_E$,

- $\overline{P_E} = P_E{}^{\to(2)} \cup \{ A^{(2)} \to A \mid A \in V_E \} \cup \{ A' \to K \mid A \in V_E \} \cup$
  $\cup \{ K \to K \}$, where $P_E{}^{\to(2)} = \{ A \to v^{(2)} \mid A \to v \in P_E \}$,

- $\overline{R_1} = R_1{}^{\to(2)} \cup R'^{\to(2)}_{n+1} \cup \{A' \to A \mid A \in V_E \} \cup$
  $\cup \{A^{(2)} \to A \mid A \in V_E \}$,

- $\overline{R_i} = R_i{}^{\to(2)} \cup R'^{\to(2)}_{n+1} \cup \{A^{(2)} \to A \mid A \in V_E \}$ for $2 \leq i \leq n$, where
  $R_j{}^{\to(2)} = \{ A \to w^{(2)} \mid A \to w \in R_j \}$ for $1 \leq j \leq n$, and
  $R'^{\to(2)}_{n+1} = \{ A \to w'^{(2)} \mid A \to w \in R_{n+1} \}$, where
  $w'^{(2)} = x_1^{(2)} \cdots x_{m-1}^{(2)} x_m{}'$ if $w = x_1 \cdots x_{m-1} x_m$, $x_i \in V_E$, and $1 \leq i \leq m$,

- $\overline{\omega} = \omega$, and

- $\overline{\Delta} = \Delta$.

The main idea of this construction is the following. We simulate one derivation step of Σ by two corresponding derivation steps of $\overline{\Sigma}$. In the first simulation step we use the rules corresponding to the derivation step of Σ, the second step checks if the simulation is correct.

In $\overline{\Sigma}$, the set R_{n+1} is included in the set of rules of each agent in the form of $R'^{\to(2)}_{n+1}$. When an agent uses a rule corresponding to R_{n+1}, a primed letter is introduced as well. The only agent which is able to make these primed letters disappear is $\overline{R_1}$; this construction guarantees (considering the rules of $\overline{P_E}$) that the derivation cannot terminate with a terminal word if the agents use more than one rule from $R'^{\to(2)}_{n+1}$ at the same derivation step.

If in a derivation step of Σ the agents $R_{i_1}, \ldots, R_{i_k}$ work and $i_j \leq n$ for $1 \leq j \leq k$, then the simulation goes as follows.

In the first simulation step in $\overline{\Sigma}$ the agents $\overline{R_{i_1}}, \ldots, \overline{R_{i_k}}$ work using rules from $R_{i_j}{}^{\to(2)}$, corresponding to the step of Σ. Then the environment uses the same rules as in Σ (in the form of $P_E{}^{\to(2)}$).

In the second simulation step the agents $\overline{R_{i_1}}, \ldots, \overline{R_{i_k}}$ rewrite k letters and the environment rewrites the remaining letters by using the rules in the form of $A^{(2)} \to A$. (There are at least k letters in the sentential form because there are no λ-rules in Σ.)

If R_{n+1} was among the agents $R_{i_1}, \ldots, R_{i_{k-1}}, R_{i_k} = R_{n+1}$ in a derivation step in Σ, then the two simulation steps of $\overline{\Sigma}$ are the following.

In the first simulation step we choose one agent $\overline{R_s}$ in $\overline{\Sigma}$, such that $s \neq i_j$ for any $1 \leq j \leq k-1$. This is possible because $k \leq n$. This agent simulates the work of R_{n+1} by using the corresponding rule of ${R'}_{n+1}^{\to(2)}$.

The agents $\overline{R_{i_1}}, \ldots, \overline{R_{i_{k-1}}}$ simulate the other rules of the agents and the environment rewrites the remaining letters in the same way as in Σ.

In the second simulation step $\overline{R_1}$ rewrites the primed letter, other $k-1$ agents rewrite $k-1$ other letters and the environment rewrites the remaining letters into symbols of V_E.

Since we can simulate every derivation step of Σ by a derivation sequence of $\overline{\Sigma}$, we obtain $L(\Sigma, = k) \subseteq L(\overline{\Sigma}, = k)$.

On the other hand, for any derivation sequence of $\overline{\Sigma}$ resulting in a terminal word, there exists a derivation in Σ generating the same word, which gives the other inclusion $L(\overline{\Sigma}, = k) \subseteq L(\Sigma, = k)$. We will simulate two derivation steps of these sequences of $\overline{\Sigma}$ by one step of Σ. For each $v \in L(\overline{\Sigma}, = k)$ (which implies $v \in \Delta^+$ because λ-rules are not allowed) there exists a derivation sequence in $\overline{\Sigma}$ in the form of

$$\overline{\omega} \Rightarrow_{\overline{\Sigma}} v_1 \Rightarrow_{\overline{\Sigma}} \cdots \Rightarrow_{\overline{\Sigma}} v_t = v.$$

Since $v \in {V_E}^+$, there are neither primed letters nor letters K in v.

Considering the rules in $\overline{P_E}$, it is easy to see that there can be at most one primed letter in the previous word v_{t-1}.

The fact that there is no primed letter in v_{t-1} means that in the $(t-1)$th step k agents, $\overline{R_{i_1}}, \ldots, \overline{R_{i_k}}$, use rules from the sets ${R_{i_j}}^{\rightarrow(2)}$ with $1 \le i_j \le n$. In this case we can simulate the last two derivation steps of $\overline{\Sigma}$ by one derivation step of Σ, using the corresponding rules of the agents $R_{i_1}, \ldots, R_{i_k}$ and the environment.

The other possibility gives that $k-1$ agents, $\overline{R_{i_1}}, \ldots, \overline{R_{i_{k-1}}}$, use rules from the sets ${R_{i_j}}^{\rightarrow(2)}$ with $1 \le i_j \le n$ and one agent uses a rule of $R'^{\rightarrow(2)}_{n+1}$. Then we can simulate the last two steps by one step in Σ when the agents $R_{i_1}, \ldots, R_{i_{k-1}}, R_{n+1}$ and the environment use the corresponding rules.

In both cases $v_{t-2} \in {V_E}^+$, thus we can continue the simulation by giving the role of v to v_{t-2}. Since the axiom is over V_E and t is an even number (see the rules of $\overline{\Sigma}$), it can be proved by induction on the length of the derivation that we can simulate the whole derivation sequence.

With λ-rules

The main idea is the same as it was in the λ-free case: we show that we can simulate any extended simple EG system $\Sigma = (\,V_E, P_E, R_1, \ldots, R_{n+1}, \omega, \Delta\,)$ working in the derivation mode $= k$ by another extended simple EG system $\overline{\Sigma} = (\,\overline{V_E}, \overline{P_E}, \overline{R_1}, \ldots, \overline{R_n}, \overline{\omega}, \overline{\Delta}\,)$ working also in the derivation mode $= k$.

Let $\overline{\Sigma}$ be the following system:

- $\overline{V_E} = V_E \cup {V_E}^{(2)} \cup \{K, T_1, T_2\}$, where $K, T_1, T_2 \notin V_E$,
- $\overline{P_E} = {P_E}^{\rightarrow(2)} \cup \{\, A^{(2)} \rightarrow A \mid A \in V_E \,\} \cup$
  $\{\, K \rightarrow K, T_1 \rightarrow K, T_2 \rightarrow \lambda \,\}$,
- $\overline{R_1} = {R_1}^{\rightarrow(2)} T_2 \cup R^{\rightarrow(2)}_{n+1} T_1 \cup \{\, T_1 \rightarrow \lambda \,\} \cup \{\, T_2 \rightarrow \lambda \,\}$,
- $\overline{R_i} = {R_i}^{\rightarrow(2)} T_2 \cup R^{\rightarrow(2)}_{n+1} T_1 \cup \{\, T_2 \rightarrow \lambda \,\}$ for $2 \le i \le n$,

  where

$$R_j{}^{\to(2)}T_2 = \{\ A \to w^{(2)}T_2 \mid A \to w \in R_j\ \} \quad \text{for } 1 \le j \le n \text{ and}$$
$$R_{n+1}^{\to(2)}T_1 = \{\ A \to w^{(2)}T_1 \mid A \to w \in R_{n+1}\ \},$$

- $\overline{\omega} = \omega$, and
- $\overline{\Delta} = \Delta$.

The statement $L(\Sigma, = k) = L(\overline{\Sigma}, = k)$ can be proved similarly to the λ-free case: T_1 plays the role of the primed letters, it controls the usage of the rules of $R_{n+1}^{\to(2)}$; T_2 letters guarantee that k agents could work in the second simulation step even if in the first simulation step λ-rules were applied. ■

From the two previous lemmas we obtain the following corollary, which says that the first parameter has no influence on the class of generated languages.

Corollary 3.7
For $1 \le k \le n \quad \mathcal{L}(EE^{(\lambda)}(n, = k)) = \mathcal{L}(EE^{(\lambda)}(k, = k))$.

Now we turn our attention to the second parameter: the number of agents working in a derivation step. Considering Corollary 3.7, it is enough to examine the relations of language classes $\mathcal{L}(EE^{(\lambda)}(k, = k))$.

For the λ-free case, it was proved in (Dassow and Mihalache 1995) that $\mathcal{L}(EE(n+1, = n+1))$ is included in $\mathcal{L}(EE(n, = n))$. In the following lemma we present the same result for extended simple EG systems with λ-rules as well. We give a simulation which is based on the construction used in (Dassow and Mihalache 1995), hence in the first part of our proof we briefly summarise their construction and their explanation for the λ-free case. Then in the second part of the proof we give the modifications which are necessary to obtain the same result when λ-rules are allowed.

Lemma 3.8
For $1 \le n \quad \mathcal{L}(EE^{(\lambda)}(n+1, = n+1)) \subseteq \mathcal{L}(EE^{(\lambda)}(n, = n))$.

PROOF.

The λ-free case (from (Dassow and Mihalache 1995)):

For any extended simple EG system $\Sigma = (\ V_E, P_E, R_1, \dots, R_{n+1}, \omega, \Delta\ )$ we give another extended simple EG system $\overline{\Sigma} = (\overline{V_E}, \overline{P_E}, \overline{R_1}, \dots, \overline{R_{n+1}}, \overline{\omega}, \overline{\Delta})$, such that $L(\Sigma, = n+1) = L(\overline{\Sigma}, = n)$.

Let $\overline{\Sigma}$ be the following:

- $\overline{V_E} = V_E \cup V_E{}' \cup \{\bar{A} \mid A \to \alpha \in R_{n+1}\} \cup \{K\}$, where $K \notin V_E$,
- $\overline{P_E} = \{\ A \to \alpha' \mid A \to \alpha \in P_E\ \} \cup \{\ A \to \bar{A} \mid A \to \alpha \in R_{n+1}\ \} \cup$
  $\cup \{\ A' \to A \mid A \in V_E\ \} \cup \{\ \bar{A} \to K \mid A \to \alpha \in R_{n+1}\ \} \cup \{\ K \to K\ \}$,
- $\overline{R_1} = \{A \to \alpha' \mid A \to \alpha \in R_1\ \} \cup \{\bar{A} \to \alpha \mid A \to \alpha \in R_{n+1}\ \}$,
- $\overline{R_i} = \{A \to \alpha' \mid A \to \alpha \in R_i\ \} \cup \{A' \to A \mid A \in V_E\ \}$ for $2 \le i \le n$,
- $\overline{\omega} = \omega$, and
- $\overline{\Delta} = \Delta$.

Let

$$x = x_1 Z_1 x_2 Z_2 \cdots x_{n+1} Z_{n+1} x_{n+2} \Rightarrow y_1 w_1 y_2 w_2 \cdots y_{n+1} w_{n+1} y_{n+2} = y$$

be a derivation step in Σ, where the rules used by the agents are $Z_i \to w_i$ for $1 \le i \le n+1$ and the derivations $x_j \Rightarrow y_j$ for $1 \le j \le n+2$ are performed by the environment. We can simulate this step by two steps of $\overline{\Sigma}$ in the following way. (We suppose that R_{n+1} used the rule $Z_{n+1} \to w_{n+1}$.)

$$x = x_1 Z_1 x_2 Z_2 \cdots x_{n+1} Z_{n+1} x_{n+2} \Rightarrow y_1{}' w_1{}' y_2{}' w_2{}' \cdots y'_{n+1} \overline{Z}_{n+1} y'_{n+2} \Rightarrow$$

$$\Rightarrow y_1 w_1 y_2 w_2 \cdots y_{n+1} w_{n+1} y_{n+2} = y$$

It follows from the construction that the environment cannot introduce two barred letters, because in this case the derivation would never result in a terminal word because of the symbol K. Therefore, only the sequences consisting of pairs of steps of this type can derive terminal words in $\overline{\Sigma}$, but these derivations can be performed in Σ as well.

In the case when λ-rules are allowed we have to modify the above construction in the following way.

- $\overline{V_E} = V_E \cup {V_E}' \cup \{\bar{A} \mid A \to \alpha \in R_{n+1}\} \cup \{K,\ Z\}$, where $K,\ Z \notin V_E$,
- $\overline{P_E} = \{\ A \to \alpha' \mid A \to \alpha \in P_E\ \} \cup \{\ A \to \bar{A} \mid A \to \alpha \in R_{n+1}\ \} \cup$
  $\cup \{\ A' \to A \mid A \in V_E\ \} \cup \{\ \bar{A} \to K \mid A \to \alpha \in R_{n+1}\ \} \cup$
  $\cup \{\ K \to K\ \} \cup \{\ Z \to \lambda\ \}$,
- $\overline{R_1} = \{A \to \alpha' Z \mid A \to \alpha \in R_1\ \} \cup \{\bar{A} \to \alpha \mid A \to \alpha \in R_{n+1}\ \}$,
- $\overline{R_i} = \{A \to \alpha' Z \mid A \to \alpha \in R_i\ \} \cup \{Z \to \lambda\ \}$ for $2 \leq i \leq n$,
- $\overline{\omega} = \omega$, and
- $\overline{\Delta} = \Delta$.

Here the letters Z guarantee that the agents can apply a rule in $\overline{\Sigma}$ in the second simulation step, even if in the first step λ-rules were used. Besides this modification, the construction is the same as in the λ-free case. Thus, we have $L(\Sigma, = n{+}1) = L(\overline{\Sigma}, = n)$ by the same reason as in the λ-free case. ■

There remains only one relation to examine: whether the language classes $\mathcal{L}(EE^{(\lambda)}(n, = n))$ are included in $\mathcal{L}(EE^{(\lambda)}(n+1, = n+1))$ or the inclusion in Lemma 3.8 is strict. The answer depends on λ-rules. When λ-rules are allowed, the two language classes are equal.

Lemma 3.9

For $1 \leq n \quad \mathcal{L}(EE^{\lambda}(n, = n)) \subseteq \mathcal{L}(EE^{\lambda}(n+1, = n+1))$.

PROOF.

For any extended simple EG system $\Sigma = (\ V_E, P_E, R_1, \ldots, R_n, \omega, \Delta\ )$ we give another extended simple EG system $\overline{\Sigma} = (\ \overline{V_E}, \overline{P_E}, \overline{R_1}, \ldots, \overline{R_{n+1}}, \overline{\omega}, \overline{\Delta}\ )$ containing λ-rules, such that $L(\Sigma, = n) = L(\overline{\Sigma}, = n+1)$.

Let $\overline{\Sigma}$ be the following:

- $\overline{V_E} = V_E \cup \{K\}$, where $K \notin V_E$,
- $\overline{P_E} = P_E \cup \{K \to K\}$,
- $\overline{R_i} = R_i$ for $1 \leq i \leq n$,
- $\overline{R_{n+1}} = \{K \to K, K \to \lambda\}$,
- $\overline{\omega} = \omega K$, and
- $\overline{\Delta} = \Delta$.

For any $v \in L(\Sigma, = n)$ there is a derivation in Σ in the form of

$$\omega \Rightarrow v_1 \Rightarrow \cdots \Rightarrow v_t = v$$

where in the ith step $(1 \leq i \leq t)$ the agents R_j use the rules r_{j_i}. Let us consider the following derivation in $\overline{\Sigma}$

$$\omega K \Rightarrow v_1 K \Rightarrow \cdots \Rightarrow v_{t-1} K \Rightarrow v_t = v$$

where in the ith step $(1 \leq i \leq t-1)$ the agents $\overline{R_j}$ $(1 \leq j \leq n)$ apply the rules r_{j_i} corresponding to the above derivation and the agent $\overline{R_{n+1}}$ uses the rule $K \to K$; in the last step the agents $\overline{R_j}$ use the corresponding rules r_{j_t}

(for $1 \leq j \leq n$) and $\overline{R_{n+1}}$ uses the rule $K \to \lambda$. In this way we can simulate Σ by $\overline{\Sigma}$.

Now we show that $L(\overline{\Sigma}, = n+1) \subseteq L(\Sigma, = n)$. For any $v \in L(\overline{\Sigma}, = n+1)$ there is a derivation in $\overline{\Sigma}$ in the form

$$\overline{\omega} = \omega K \Rightarrow v_1 \Rightarrow \cdots \Rightarrow v_{t-1} \Rightarrow v_t = v$$

where in each derivation step all the $n+1$ agents work. Since v is over $\overline{\Delta} = \Delta$, there are no letters K in it. But in the last step $n+1$ agents must work and the last agent can use only the rules $K \to K$ or $K \to \lambda$. Moreover, considering that $\overline{\omega}$ contains one K and that there is no rule producing new letters K, we get that all words in the above derivation, except v, contain exactly one letter K. We also get that in the first $t-1$ step the $(n+1)$th agent has to use the rule $K \to K$ and it has to use the rule $K \to \lambda$ in the last step. But in this case we can simulate this derivation of $\overline{\Sigma}$ by the following derivation in Σ:

$$\omega = f(\overline{\omega}) \Rightarrow f(v_1) \Rightarrow \cdots \Rightarrow f(v_{t-1}) \Rightarrow f(v_t) = v_t = v$$

where f is a homomorphism over $V_E \cup \{K\}$ defined as follows:

$$f(z) = \begin{cases} z & \text{if } z \in V_E \\ \lambda & \text{if } z = K \end{cases}$$

In the above derivation the agents $R_1, R_2, \ldots, R_n$ use the rules corresponding to the derivation of $\overline{\Sigma}$. ∎

On the other hand, in the λ-free case the inclusion is proper between the language classes $\mathcal{L}(EE(n, = n))$ and $\mathcal{L}(EE(n+1, = n+1))$.

Lemma 3.10

For any $1 \leq n$ there exists a language L, such that

$$L \in \mathcal{L}(EE(n, = n)) \setminus \mathcal{L}(EE(n+1, = n+1)).$$

Proof.

Let L be the finite language $\{\ a^n, b^n\ \}$. It is clear that it can be generated by an extended simple EG system working in the derivation mode $= n$. Moreover, it cannot be generated by using the derivation mode $= \ell$ if $\ell > n$ because in this case either the letters in the axiom are not enough to perform an action of ℓ agents or we should use λ-rules which is not allowed. ∎

Concluding our investigations, from Corollary 3.7, Lemma 3.8 and Lemma 3.9 we obtain the following theorem. It says that when λ-rules are allowed, neither the number of the agents being in the system nor the number of the agents working in a derivation step have any influence on the generated languages.

Theorem 3.11

For $1 \leq k \leq n \quad \mathcal{L}(EE^{\lambda}(n, = k)) = \mathcal{L}(EE^{\lambda}(1, = 1))$.

In the λ-free case we use the notation $\mathcal{L}(EE(= k))$ for the class of languages which can be generated by extended simple EG systems working in the derivation mode $= k$, without λ-rules. From Corollary 3.7 we have $\mathcal{L}(EE(n, = k)) = \mathcal{L}(EE(= k))$ for $1 \leq k \leq n$.

Using this fact together with Lemma 3.8 and Lemma 3.10, we can present the result of the λ-free case in the following theorem.

Theorem 3.12

$\mathcal{L}(EE(= 1)) \supset \mathcal{L}(EE(= 2)) \supset \cdots \supset \mathcal{L}(EE(= k)) \supset \cdots \quad$ *for* $1 \leq i$.

We summarise the results of this section in the form of a diagram, where a straight arrow indicates a proper inclusion; the class the arrow leaves is included in the class the arrow points at. The inclusion $\mathcal{L}(EE(1, = 1)) \subset$

$\mathcal{L}(EE^{\lambda}(1,=1))$ can be proved by any language of $\mathcal{L}(EE^{\lambda}(1,=1))$ containing the empty word, λ.

We use the notation n_i for $1 \leq i$, where n_i denotes an arbitrary positive integer, such that $i \leq n_i$.

$$
\begin{array}{c}
\mathcal{L}(EE^{\lambda}(1,=1)) = \mathcal{L}(EE^{\lambda}(n,=k)) \\
\uparrow \\
\mathcal{L}(EE(1,=1)) = \mathcal{L}(EE(n_1,=1)) \\
\uparrow \\
\mathcal{L}(EE(2,=2)) = \mathcal{L}(EE(n_2,=2)) \\
\uparrow \\
\vdots \\
\uparrow \\
\mathcal{L}(EE(k,=k)) = \mathcal{L}(EE(n_k,=k)) \\
\uparrow \\
\vdots
\end{array}
$$

Figure 3.1: Hierarchy in team derivation mode

Chapter 4

New operations in simple eco-grammar systems

In this chapter we consider simple eco-grammar systems with agents using new operations, namely insertion and replacement. These operations, motivated by genetics, became a topic in the focus of interest and have been studied intensively in formal language theory. (See (Kari 1991), (Kari and Thierrin 1996) and (Dassow and Mitrana 1997).)

Combining insertion and replacement with the concept of the simple eco-grammar system, in the simple case we obtain the definition of the simple eco-grammar system of type X, with $X \in \{\mathrm{INS}, \mathrm{REP}\}$. A simple eco-grammar system of type X is nothing more than a simple eco-grammar system with a slight modification: all agents are represented by sets of type X rules. (According to this notation, the original model, when the agents are represented by CF rules, is a simple eco-grammar system of type CF.)

We get a more complicated model if we allow the possibility of different agents having different types of rules. In these hybrid eco-grammar systems there are agents of type INS and REP at the same time.

For hybrid systems we define two derivation modes: the derivation modes $= k$ and $(= k_1, = k_2)$. In the former case only the number of agents working in a derivation step is prescribed while in the latter case both the number of INS-type and the number of REP-type agents are determined. In the first part of the chapter the relation of language classes generated by hybrid eco-grammar systems with different parameters n and k (or k_1, k_2) is studied and it is shown that parameters k and k_1,k_2 are significant, the language families defined by these restrictions with different k (or k_1, k_2) are pairwise incomparable.

In homogeneous systems we consider the same two parameters as in Chapter 3: the number of agents being in the system and the number of agents working in a derivation step. In the second part of the chapter we examine the role of these parameters by using the same ideas as in the hybrid case. Since simple eco-grammar systems of type INS and REP are special cases of hybrid eco-grammar systems, the techniques used and the results obtained in the derivation modes $(= 0, = k_2)$ and $(= k_1, = 0)$ in the hybrid case can be applied for these homogeneous systems as well.

The results about simple eco-grammar systems of type INS and REP show striking similarity to the results presented in Chapter 3 about EG systems of type CF: only the value of the parameter k proves to be important.

At the end of the chapter the relation between language classes generated by systems of different types (i.e. INS, REP or CF) is examined and it is shown that the corresponding language classes are pairwise incomparable. Thus we prove that the type of the rules of the agents is also an important parameter of the system.

The results about the homogeneous systems were presented in (Csima 1997). The stronger results about hybrid eco-grammar systems appeared first in (Csima 1998b), then a refined and completed version was published in (Csima 1999).

4.1 Hybrid eco-grammar systems

In the definition of a simple eco-grammar system the agents have context-free rules; now we modify this condition and we use new operations, insertion and replacement.

An insertion rule (an INS-type rule, for short) over an alphabet V is an ordered triple (u, z, v), where $u, z, v \in V^+$. We derive y from x by applying the insertion rule (u, z, v) if $x = x_1x_2$, $y = x_1zx_2$, and $u \preceq x_1$, $v \preceq x_2$, with $x_1, x_2 \in V^*$.

By a replacement rule (a REP-type rule, for short) over an alphabet V we mean a rule in the form of $(a, b, c) \to (a, d, c)$, where $a, b, c, d \in V$. We derive y from x by the replacement rule $(a, b, c) \to (a, d, c)$ if $x = x_1abcx_2$, $y = x_1adcx_2$, and $x_1, x_2 \in V^*$.

By an insertion rule we can insert a new sub-word into the sentential form, while by a replacement rule we can change only one letter. We can see that there is also a difference between the context restrictions in the two definitions. When applying an INS-type rule, the two parts of the context condition, u and v, must be present in the sentential form in the same order as presented in the rule, and the position of the insertion can be anywhere between them. In a REP-type rule the restriction is more strict: the close contexts (that is, the neighbouring letters) are prescribed by the rule. This difference is motivated by the properties of DNA strands: local point-mutations are determined by the close context of the position where the mutation happens, while a global change in the DNA strand (e.g. an insertion) is directed by the presence or absence of some substrings in the strand.

We study two different models using these new operations. The first type of models is the simple eco-grammar system of type INS or REP, where all the agents are of the same type. We study these systems in Section 4.2.

In the other model, in the hybrid eco-grammar system, there can be agents of both type.

Definition 4.1

A hybrid eco-grammar system is a construct $\Sigma = (V_E, P_E, n_1, R_1, \ldots, R_n, \omega)$, *where*

- V_E *is a finite, non-empty alphabet,*
- P_E *is a complete set of CF rules over* V_E*,*
- $1 \leq n$, $0 \leq n_1 \leq n$,
- R_i *is a set of INS-type rules for* $1 \leq i \leq n_1$ *and* R_i *is a set of REP-type rules for* $n_1 + 1 \leq i \leq n$,
- $\omega \in {V_E}^*$.

In this construct V_E, P_E, and ω are defined in the same way as in the original definition of a simple eco-grammar system. The first n_1 sets, R_i, $1 \leq i \leq n$, represent the agents of type INS, the others represent the agents of type REP. Notice that $n_1 = 0$ or $n_1 = n$ are also allowed, that is, the system can contain only REP-type or only INS-type agents, which gives that simple eco-grammar systems of type INS or REP are special hybrid eco-grammar systems.

We study different derivation modes in hybrid eco-grammar systems. The difference lies in the restrictions prescribed for the number of agents being active in a derivation step. One of these possibilities is the derivation mode $(=k_1,=k_2)$, defined below.

Definition 4.2

Consider a hybrid eco-grammar system $\Sigma = (V_E, P_E, n_1, R_1, \ldots, R_n, \omega)$. *We say that* x *directly derives* y *in* Σ *in the derivation mode* $(=k_1,=k_2)$, *(with* $x, y \in {V_E}^*$, $0 \le k_1 \le n_1$, $0 \le k_2 \le n - n_1$, *and* $1 \le k = k_1 + k_2$, *written as* $x \overset{(=k_1,=k_2)}{\Longrightarrow}_{\Sigma} y$*), if*

- *there exists a word* $\overline{x} = x_1 o_1 x_2 o_2 \cdots x_k o_k x_{k+1}$ *over* $V_E \cup \{I\}$, $I \notin V_E$, *such that* $x_t \in {V_E}^*$, $o_j \in V_E \cup \{I\}$ *for* $1 \le t \le k+1$, $1 \le j \le k$, $|\{j \mid 1 \le j \le k, o_j = I\}| = k_1$, $|\{j \mid 1 \le j \le k, o_j \in V_E\}| = k_2$, *and* $f(\overline{x}) = x$, *where* f *is the homomorphism defined over* $V_E \cup \{I\}$ *as follows:*
$$f(z) = \begin{cases} z & \text{if } z \in V_E \\ \lambda & \text{if } z = I \end{cases}$$
- $y = y_1 \gamma_1 y_2 \gamma_2 y_3 \cdots \gamma_k y_{k+1}$, *with* $y_t, \gamma_j \in {V_E}^*$ *for* $1 \le t \le k+1, 1 \le j \le k$,
- *there exist* k *different agents* $R_{i_1}, \ldots, R_{i_k}$, *such that*

  if $o_j = I$, *then there exists an INS-type rule* $(\alpha_j, \gamma_j, \beta_j) \in R_{i_j}$, *such that* $\alpha_j \preceq f(x_1 o_1 \cdots o_{j-1} x_j)$ *and* $\beta_j \preceq f(x_{j+1} o_{j+1} \cdots o_k x_{k+1})$,

  if $o_j \in V_E$, *then there exists a REP-type rule* $(a_j, o_j, b_j) \to (a_j, \gamma_j, b_j) \in R_{i_j}$, *such that* $f(x_1 o_1 \cdots o_{j-1} x_j) = u a_j$ *and* $f(x_{j+1} o_{j+1} \cdots o_k x_{k+1}) = b_j v$, *where* $u, v \in {V_E}^*$, *and*
- $x_1 x_2 \cdots x_{k+1} = y_1 y_2 \cdots y_{k+1} = \lambda$ *or* $x_1 x_2 \cdots x_{k+1} \overset{0L}{\Longrightarrow}_E y_1 y_2 \cdots y_{k+1}$, *where* $E = (V_E, P_E, \omega)$ *is the 0L system of the environment.*

Informally speaking, a derivation step goes in the following way. We choose k different agents and k sites in the sentential form, k_1 sites for insertion, k_2 sites for replacement, $k_1 + k_2 = k$. The positions where insertions are to be performed are marked by I, the positions chosen for replacements are marked by the letter to be replaced. These positions must be compatible with the corresponding operations: for each position there must be a chosen agent which is of the corresponding type and which has a rule applicable at that position. When the agents have applied the chosen rules, the environment rewrites the remaining letters.

We emphasise the following consequences of the definition: more than one agent can use the same context to apply an insertion and the contexts of several replacements can be overlapping, too. That is, for example in a sentential form $a_1a_2a_3a_4$ (where the a_i's are letters) the following two insertion rules and the following two replacement rules can be used in the same derivation step: (a_1, α_1, a_2) and (a_1, α_2, a_2), $(a_1, a_2, a_3) \rightarrow (a_1, d_1, a_3)$, and $(a_2, a_3, a_4) \rightarrow (a_2, d_2, a_4)$. If the environment does not change the letters, with these four rules we obtain either the word $a_1\alpha_1\alpha_2d_1d_2a_4$ or the word $a_1\alpha_2\alpha_1d_1d_2a_4$.

The above definition also implies that if we consider the derivation mode $(= k_1, = k_2)$ in a hybrid EG system $\Sigma = (\, V_E, P_E, n_1, R_1, \ldots, R_n, \omega\,)$, then $0 \leq k_1 \leq n_1$, $0 \leq k_2 \leq n - n_1$ and $1 \leq k_1 + k_2$ hold, where n_1 is the number of INS-type agents in the system.

In a derivation step $(= k_1, = k_2)$ we prescribe how many INS-type and how many REP-type agents work in a derivation step. If we prescribe only the total number of agents working in a derivation step, then we obtain derivation mode $= k$.

Definition 4.3

Consider a hybrid eco-grammar system $\Sigma = (V_E, P_E, n_1, R_1, \ldots, R_n, \omega)$*. We say that* x *directly derives* y *in* Σ *in the derivation mode* $= k$*, (with* $x, y \in {V_E}^*$ *and* $1 \leq k \leq n$*, written as* $x \overset{=k}{\Longrightarrow}_\Sigma y$*), if* $x \overset{(=k_1,=k_2)}{\Longrightarrow}_\Sigma y$ *for some* $0 \leq k_1 \leq n_1$*,* $0 \leq k_2 \leq n - n_1$*, where* $k_1 + k_2 = k$*.*

We denote the transitive and reflexive closure of $\overset{(=k_1,=k_2)}{\Longrightarrow}_\Sigma$ and $\overset{=k}{\Longrightarrow}_\Sigma$ by $\overset{(=k_1,=k_2)}{\Longrightarrow}{}^*_\Sigma$ and $\overset{=k}{\Longrightarrow}{}^*_\Sigma$.

Definition 4.4

Consider a hybrid eco-grammar system $\Sigma = (V_E, P_E, n_1, R_1, \ldots, R_n, \omega)$*. The languages generated by* Σ *in the derivation modes* $(= k_1, = k_2)$ *and* $= k$ *are the following:*

$$L(\Sigma, = k_1, = k_2) = \{\, v \in {V_E}^* |\, \omega \overset{(=k_1,=k_2)}{\Longrightarrow}{}^*_\Sigma \; v \,\},$$

$$L(\Sigma, = k) = \{\, v \in {V_E}^* |\, \omega \overset{=k}{\Longrightarrow}{}^*_\Sigma \, v \,\}.$$

The class of languages which can be generated by hybrid eco-grammar systems is denoted by $\mathcal{L}(E_H)$; in the two special cases, when the agents are of the same type, the language classes are denoted by $\mathcal{L}(E_{INS})$ and $\mathcal{L}(E_{REP})$. The class of languages generated by hybrid EG systems containing n agents and operating in the derivation mode $(= k_1, = k_2)$ or in the derivation mode $= k$ is denoted by $\mathcal{L}(E_H(n, = k_1, = k_2))$ and $\mathcal{L}(E_H(n, = k))$. We replace the subscript H by INS or REP in the case of homogeneous INS- or REP-type EG systems.

We present an example, which shows how a hybrid eco-grammar system works and which proves to be useful in the proofs of the chapter, too.

Example 4.5

Let $0 \leq k_1, k_2$ *and* $1 \leq n$*, such that* $1 \leq k_1 + k_2 \leq n$ *and let* $\Sigma = (V_E, P_E, k_1, R_1, \ldots, R_n, \omega)$ *be the following hybrid eco-grammar system:*

- $V_E = \{a_1, a_2, c_1, \ldots, c_n\}$,
- $P_E = \{a_1 \to a_2, a_2 \to {a_1}^2\} \cup \bigcup_{1 \leq j \leq n} \{c_j \to a_1\}$,
- $R_i = \{(a_1, c_i, a_1), (a_2, a_1, a_2)\}$ *for* $1 \leq i \leq k_1$,
- $R_i = \{(a_1, a_1, a_1) \to (a_1, c_i, a_1)\} \cup \{(e, c_i, f) \to (e, a_1, f) \mid e, f \in \{a_2, c_1, \ldots, c_n\}\}$ *for* $k_1 + 1 \leq i \leq n$,
- $\omega = {a_1}^{2+k_2}$.

In this way we define a hybrid eco-grammar system for any $0 \leq k_1, k_2$, $1 \leq n$, where $1 \leq k_1 + k_2 \leq n$ holds. For each such system we consider two derivation modes, the derivation modes $(= k_1, = k_2)$ and $= k$ with $k = k_1 + k_2$. The generated languages $L(\Sigma, = k_1, = k_2)$ and $L(\Sigma, = k)$ are denoted by $L_H(n, = k_1, = k_2)$ and $L_H(n(k_1), = k)$ and proves to be useful in several proofs of the chapter.

The parameters n, k and k_1, k_2 in the notations $L_H(n, = k_1, = k_2)$ and $L_H(n(k_1), = k)$ refer to the following: these languages are generated by a hybrid EG system defined in Example 4.5, in the derivation mode $(= k_1, = k_2)$ or $= k$. The generating system contains n agents, k_1 agents of type INS, $n - k_1$ agents of type REP and the length of the axiom is equal to $k_2 + 2$, where $k_2 = k - k_1 \leq n - k_1$.

Having examined the system defined in Example 4.5 and the derivation process, we can see that both generated languages, $L_H(n, = k_1, = k_2)$ and $L_H(n(k_1), = k)$, contain two types of words: ${a_1}^+$ (the shortest word of this type is the axiom, ${a_1}^{2+k_2}$) and $a_2[c_1, c_2, \ldots, c_{k_1}, c_{j_1}, \ldots, c_{j_{k_2}}, a_2, \ldots, a_2]^P a_2$, that is, words containing the first k_1 letters c and more k_2 different letters c, besides some letters a_2. In this latter type of words the symbols c can stand anywhere except the first and the last position of the word.

In the following we present some auxiliary statements which are used in the chapter. In these lemmas we use the following notions:

Definition 4.6

If $a \in \mathrm{alph}(L)$ for a language L and for any $K_0 > 0$ there exists a word $w \in L$ with $|w|_a \geq K_0$, then the symbol a is said to be a letter with unbounded occurrence, otherwise a is said to be a letter with bounded occurrence.

Lemma 4.7

Let L be a language with the following properties:

(i) *$L = L_1 \cup L_2$, where both L_1 and L_2 are infinite languages and their alphabets, $\mathrm{alph}(L_1)$ and $\mathrm{alph}(L_2)$, are disjoint sets.*

(ii) *For any letter $a \in \mathrm{alph}(L_i)$ with unbounded occurrence ($i = 1, 2$), for any $w \in L_i$, and for any $K > 0$ there exists a word $w' \in L_i$, such that $w \preceq w'$ and $|w'|_a > K$.*

If L has these properties and $L = L(\Sigma, = k)$ or $L = L(\Sigma, = k_1, = k_2)$ with a hybrid EG system $\Sigma = (\, V_E, P_E, n_1, R_1, \ldots, R_n, \omega \,)$, then for each $u, v \in L$ the facts that $v \in L_i$ and $u \stackrel{=k}{\Longrightarrow}_\Sigma v$ or $u \stackrel{(=k_1,=k_2)}{\Longrightarrow}_\Sigma v$ implies that $u \in L_j$ and $i \neq j$.

Informally speaking, this lemma says that if a language has the above properties and it can be generated by a hybrid eco-grammar system, then the words following each other in a derivation sequence are over different sub-alphabets.

In this section, when applying this lemma to a language L, the division $L = L_1 \cup L_2$ will be always straightforward, in general the letters of $\mathrm{alph}(L_1)$

and the letters of $\mathrm{alph}(L_2)$ will be the letters with subscript 1 and 2, respectively.

PROOF.

First we prove a slightly weaker statement: we show that there exists a positive integer N, such that for $u, v \in L$ if $v \in L_i$, $|v| > N$, and $u \overset{=k}{\Longrightarrow}_\Sigma v$ or $u \overset{(=k_1,=k_2)}{\Longrightarrow}_\Sigma v$, then $u \in L_j$ and $i \neq j$. That is, first we show that words over one of the sub-alphabets and being longer than N can be derived only from words being over the other sub-alphabet.

Let us define N as follows:

- in derivation mode $= k$ let $N = k \cdot \max_{1 \leq l \leq n_1} \max_{(u,z,v) \in R_l} |z| + k + M \cdot \max_{s \to t \in P_E} |t|$, where M is the maximum number of letters with bounded occurrence in a word of L;

- in derivation mode $(= k_1, = k_2)$ let $N = k_1 \cdot \max_{1 \leq l \leq n_1} \max_{(u,z,v) \in R_l} |z| + k_2 + M \cdot \max_{s \to t \in P_E} |t|$, where M is the same as above.

Suppose on the contrary that there exist words $u, v \in L$ such that $u \overset{=k}{\Longrightarrow}_\Sigma v$ or $u \overset{(=k_1,=k_2)}{\Longrightarrow}_\Sigma v$, $|v| > N$ and $u, v \in L_i$ (for $i = 1$ or $i = 2$).

Since $|v| > N$, it is sure that in the step $u \overset{=k}{\Longrightarrow}_\Sigma v$ or $u \overset{(=k_1,=k_2)}{\Longrightarrow}_\Sigma v$ the system Σ uses at least one production in the form of $a \to w_1$ from P_E, where $a \in \mathrm{alph}(L_i)$ is a letter with unbounded occurrence and $w_1 \in (\mathrm{alph}(L_i))^+$ is a non-empty word being over $\mathrm{alph}(L_i)$ (otherwise $|v| \leq N$ would hold because of the definition of N).

The existence of such a rule implies that Σ cannot perform any derivation step $u_1 \overset{=k}{\Longrightarrow}_\Sigma v_1$ or $u_1 \overset{(=k_1,=k_2)}{\Longrightarrow}_\Sigma v_1$, where $u_1 \in L_i$ and $\lambda \neq v_1 \in L_j$, $i \neq j$, that is, such a derivation step where from a word over $\mathrm{alph}(L_i)$ a non-empty word over the other sub-alphabet would be derived. This is true because in this case, by combining the above rule $a \to w_1$ of P_E and the rules used in the step $u_1 \overset{=k}{\Longrightarrow}_\Sigma v_1$ or $u_1 \overset{(=k_1,=k_2)}{\Longrightarrow}_\Sigma v_1$, we would be able to derive words in which letters from both sub-alphabets occur. (Here we use condition (ii), which guarantees that there is a word in L_i in which u_1 is a sub-word and a occurs at least once.)

Therefore the words over $\text{alph}(L_j)$ can be derived only from words over the same sub-alphabet, which is also true for those words over $\text{alph}(L_j)$ which are longer than N. Since we know that such words exist (L_j is an infinite language) we can conclude that there exists a rule in P_E in the form of $b \to w_2$, where $b \in \text{alph}(L_j)$ is a letter with unbounded occurrence and $w_2 \in (\text{alph}(L_j))^+$.

The existence of such a rule implies that Σ cannot perform any derivation step where from a word over $\text{alph}(L_j)$ a non-empty word over the other sub-alphabet $\text{alph}(L_i)$ would be derived because in this case, combining this derivation step with the application of the above rule $b \to w_2$, we could generate words in which letters from both sub-alphabets occur.

Hence we have obtained that words over one of the sub-alphabets can be derived only from words over this sub-alphabet, which is a contradiction because in this way only the words of L_i or only the words of L_j could be derived, depending on the alphabet of the axiom.

Thus we have proved that words longer than N being over one of the sub-alphabets can be derived only from words being over the other sub-alphabet.

Now we prove the statement of the lemma. Let us suppose that there exists a derivation step $u \overset{=k}{\Longrightarrow}_\Sigma v$ or $u \overset{(=k_1,=k_2)}{\Longrightarrow}_\Sigma v$, such that $u, v \in (\text{alph}(L_i))^+$. Since we have seen that sufficiently long words over $\text{alph}(L_j)$ can be derived only from words over $\text{alph}(L_i)$ and since L_i and L_j are infinite languages, there must be at least one rule in P_E in the form of $a \to w_1$, where

$a \in \mathrm{alph}(L_i)$ is a letter with unbounded occurrence, $w_1 \in (\mathrm{alph}(L_j))^+$ and $i \neq j$. Combining this rule with the derivation step $u \stackrel{=k}{\Longrightarrow}_\Sigma v$ or $u \stackrel{(=k_1,=k_2)}{\Longrightarrow}_\Sigma v$, we could derive words in which symbols from both sub-alphabets occur. ■

In Section 4.2 we use a modified version of the above lemma for simple eco-grammar systems of type CF.

Lemma 4.8

Let L be a language with properties (i) and (ii) mentioned in Lemma 4.7. If $L = L(\Sigma, = k)$ with a simple eco-grammar system $\Sigma = (V_E, P_E, R_1, \ldots, R_n, \omega)$ of type CF, then for each $u, v \in L$ the facts that $v \in L_i$ and $u \stackrel{=k}{\Longrightarrow}_\Sigma v$ imply that $u \in L_j$ and $i \neq j$.

Proof.

The proof uses analogous arguments to the ones presented in the proof of Lemma 4.7. There is only one difference, namely that N is defined as $N := k \cdot \max_{1 \leq l \leq n} \max_{A \to z \in R_l} |z| + M \cdot \max_{s \to t \in P_E} |t|$, where M is the maximum number of letters with bounded occurrence in a word of L. ■

The following lemma proves to be useful when proving incomparability of language classes defined with different parameters.

Lemma 4.9

Let L be a language with properties (i) and (ii) mentioned in Lemma 4.7. Moreover, let L have the following properties, too: there exist two positive integers k and n, such that $2 \leq k \leq n$ and

(iii) *there exist n different letters $c_1, \ldots, c_n \in \mathrm{alph}(L_2)$, such that for every $w \in L_2$:*

- $\sum_{i=1}^{n} |w|_{c_i} = k$,
- $|w|_{c_i} \leq 1$ *for* $1 \leq i \leq n$,

(iv) *for any* i, j, $1 \leq i, j, \leq n$, $i \neq j$ *and for any* $M_0 > 0$ *there exists a word* w' *in* L_2, *such that* $w' = x_1 c_i x_2 c_j x_3$, *with* $x_1, x_2, x_3 \in (\mathrm{alph}(L_2))^*$ *and* $|x_2| > M_0$, *and*

(v) $|\mathrm{alph}(L_1)| = 1$.

If $L = L(\Sigma, = \ell)$ *or* $L = L(\Sigma, = \ell_1, = \ell_2)$ *with a hybrid eco-grammar system* $\Sigma = (\, V_E, P_E, m_1, R_1, \ldots, R_m, \omega \,)$, *then* $m \geq n$ *and* $\ell \geq k$ *or* $\ell_1 + \ell_2 \geq k$, *respectively.*

PROOF.
Let us consider the rules whose application can introduce letters c into the strings of L_2. These rules are either from P_E or are rules of the agents, that is, (see Lemma 4.7) they are in the form of (α, β, γ), where $\alpha, \gamma \in (\mathrm{alph}(L_1))^+$, $\beta \in (\mathrm{alph}(L_2))^+$ or in the form of $(a, b, e) \rightarrow (a, d, e)$, where $a, b, e \in \mathrm{alph}(L_1)$, $d \in \mathrm{alph}(L_2)$.

1. Rules of P_E cannot introduce letters c because in this case it would be possible to generate words with more than k letters c (see condition (v), which gives that the only one letter in $\mathrm{alph}(L_1)$ is a letter with unbounded occurrence).

2. If $\ell < k$ or $\ell_1 + \ell_2 < k$ or if $m < n$, then there must be at least one agent R_{i_1} in Σ which is able to introduce two different letters c: c_i and c_j (otherwise words of L_2 would contain at most $k - 1$ letters c or there would be at most $n - 1$ different letters c in $\mathrm{alph}(L_2)$). Since the distance between the occurrences of c_i and c_j is unbounded (see

condition (iv)), there must be at least one more agent R_{i_2} in Σ which can also introduce c_i or c_j and we can use these two agents, R_{i_1} and R_{i_2}, together in the same derivation step. But in this case we could derive words with two c_i or c_j (see condition (v)), which is not possible in the generated language.

$\blacksquare$

In the following we present results concerning the relations of language classes $\mathcal{L}(E_H(n,=k))$ and $\mathcal{L}(E_H(m,=k_1,=k_2))$. To begin with, we examine the relation between language classes $\mathcal{L}(E_H(n,=k))$ defined with different k or n. The following theorem says that if the second parameter, k, is fixed, then there is a hierarchy according to the number of agents, but the language classes generated with different parameters k are incomparable.

Theorem 4.10

For $1 \leq k \leq n$ and $1 \leq \ell \leq m$

1. $\mathcal{L}(E_H(n,=k)) \subset \mathcal{L}(E_H(m,=k))$ *if* $k \geq 2$ *and* $m > n$,
2. $\mathcal{L}(E_H(1,=1)) \subset \mathcal{L}(E_H(2,=1))$,
3. $\mathcal{L}(E_H(n,=1)) = \mathcal{L}(E_H(m,=1))$ *if* $n, m \geq 2$,
4. *otherwise* $\mathcal{L}(E_H(n,=k))$ *and* $\mathcal{L}(E_H(m,=\ell))$ *are incomparable.*

PROOF.

1.

For a language $L = L(\Sigma,=k)$ generated by a hybrid eco-grammar system $\Sigma = (V_E, P_E, n_1, R_1, \ldots, R_n, \omega)$, we construct the following hybrid EG system $\overline{\Sigma} = (\,\overline{V_E}, \overline{P_E}, \overline{m_1}, \overline{R_1}, \ldots, \overline{R_m}, \overline{\omega}\,)$:

- $\overline{V_E} = V_E \cup \{K\}$, where $K \notin V_E$,
- $\overline{P_E} = P_E \cup \{K \to K\}$,
- $\overline{m_1} = n_1$,
- $\overline{R_i} = R_i \quad$ for $1 \leq i \leq n$,
- $\overline{R_j} = \{(K, K, K) \to (K, K, K)\} \quad$ for $n+1 \leq j \leq m$, and
- $\overline{\omega} = \omega$.

We have $L(\overline{\Sigma}, = k) = L(\Sigma, = k)$ because the agents $\overline{R_{n+1}}, \ldots, \overline{R_m}$ can never participate in any derivations of $\overline{\Sigma}$. This gives the inclusion $\mathcal{L}(E_H(n, = k)) \subseteq \mathcal{L}(E_H(m, = k))$ if $m > n$.

To complete the proof, we show that the inclusion is proper if $k > 1$. $L_H(m(k), = k)$ (see Example 4.5) has the properties required by Lemma 4.9 if $k \geq 2$, thus the lemma proves that it cannot be generated by a hybrid eco-grammar system containing less than m agents.

2.

It follows from the above proof that $\mathcal{L}(E_H(1, = 1)) \subseteq \mathcal{L}(E_H(2, = 1)$. The following example shows that the inclusion is proper.

Let $\Sigma = (\, V_E, P_E, 1, R_1, R_2, \omega\,)$ be the following hybrid eco-grammar system:

- $V_E = \{a_1, a_2, b_1, b_2, c, d\}$,
- $P_E = \{a_1 \to a_2, a_2 \to a_1, b_1 \to {b_2}^2, b_2 \to {b_1}^2, c \to a_1, d \to a_1\}$,
- $R_1 = \{(a_1, c^2, a_1), (a_2, a_1, a_2)\}$,
- $R_2 = \{(a_1, a_1, b_1) \to (a_1, d, b_1), (a_2, d, b_2) \to (a_2, a_1, b_2)\}$, and
- $\omega = {a_1}^2 b_1$.

In mode $= 1$ we have two possibilities for the first two steps:

$$a_1{}^2 b_1 \overset{=1}{\Longrightarrow}_\Sigma a_2 c^2 a_2 b_2{}^2 \overset{=1}{\Longrightarrow}_\Sigma a_1{}^5 b_1{}^4$$

or

$$a_1{}^2 b_1 \overset{=1}{\Longrightarrow}_\Sigma a_2 d b_2{}^2 \overset{=1}{\Longrightarrow}_\Sigma a_1{}^2 b_1{}^4.$$

The generated language contains only words of type $a_1{}^+ b_1{}^+$, $a_2{}^+ c^2 a_2{}^+ b_2{}^+$ or $a_2{}^+ d b_2{}^+$.

Let us suppose that $L(\Sigma, = 1) = L(\overline{\Sigma}, = 1)$ with a hybrid eco-grammar system $\overline{\Sigma} = (\overline{V_E}, \overline{P_E}, 1, \overline{R_1}, \overline{\omega})$ containing only one agent.

We can apply Lemma 4.7 to $L(\Sigma, = 1)$, with $L_1 = L(\Sigma, = 1) \cap \{a_1, b_1\}^*$, $L_2 = L(\Sigma, = 1) \cap \{a_2, b_2, c, d\}^*$, therefore words containing letters c or d must be derived in $\overline{\Sigma}$ from words in the form of $a_1{}^+ b_1{}^+$.

By the rules of $\overline{P_E}$ we cannot introduce letters c or d because in this case the number of letters c or d would not be bounded in the generated words.

If $a_1 \to \alpha$ and $b_1 \to \beta$ are rules of $\overline{P_E}$, then $\alpha \in a_2{}^*$ and $\beta \in b_2{}^*$, otherwise words with mixed occurrences of letters a_2 and b_2 or words starting with b_2 or ending with a_2 would be in the generated language. Moreover, there must be at least one rule in $\overline{P_E}$ in the form of $a_1 \to \alpha$ and $b_1 \to \beta$ with $\alpha \neq \lambda$ and $\beta \neq \lambda$, otherwise a_2 or b_2 would be letters with bounded occurrence, which is not the case.

The only possibility to generate words with a sub-word c^2 is to use insertion rules because neither $\overline{P_E}$ nor a replacement rule can generate it. In $\overline{\Sigma}$ there is only one agent and so this agent must be an INS-type one, that is, the letter d is also generated by an insertion rule. This rule is in the form of $(a_1{}^{i_1}, \gamma d \delta, a_1{}^{i_2} b_1{}^{i_3})$, $(a_1{}^{i_1}, \gamma d \delta, a_1{}^{i_2})$, $(a_1{}^{i_1}, \gamma d \delta, b_1{}^{i_2})$, $(a_1{}^{i_1} b_1{}^{i_2}, \gamma d \delta, b_1{}^{i_3})$, or $(b_1{}^{i_1}, \gamma d \delta, b_1{}^{i_2})$, where $\gamma, \delta \in \{a_2, b_2\}^*$ and $i_1, i_2, i_3 \geq 1$. But none of these insertion rules is able to determine the exact position of the letter d: using the non-erasing rules of $\overline{P_E}$, we could generate words where the letter d occurs at a wrong position, in the middle of a sequence of letters a_2 or b_2.

3.

We have to prove that $\mathcal{L}(E_H(m, = 1)) \subseteq \mathcal{L}(E_H(n, = 1))$ if $m \geq n \geq 2$, since the other direction follows from the proof of part 1.

Let us suppose that $L = L(\Sigma, = 1)$ with a hybrid eco-grammar system $\Sigma = (\, V_E, P_E, m_1, R_1, \ldots, R_m, \omega\,)$. Then let $\overline{\Sigma} = (\, \overline{V_E}, \overline{P_E}, \overline{n_1}, \overline{R_1}, \ldots, \overline{R_n}, \overline{\omega}\,)$ be the following system:

- $\overline{V_E} = V_E$,
- $\overline{P_E} = P_E$,
- $\overline{n_1} = 1$ if $m > m_1 > 0$, $\overline{n_1} = 0$ if $m_1 = 0$, and $\overline{n_1} = n$ if $m_1 = m$,
- $\overline{R_1} = \bigcup_{1 \leq i \leq m_1} R_i$ if $m_1 > 0$, otherwise $\overline{R_1} = \bigcup_{1 \leq j \leq m} R_j$,
- $\overline{R_i} = \bigcup_{m_1+1 \leq j \leq m} R_j$ if $m_1 < m$, otherwise $\overline{R_i} = \bigcup_{1 \leq j \leq m} R_j$, for $2 \leq i \leq n$,
- $\overline{\omega} = \omega$.

It is clear that $L(\Sigma, = 1) = L(\overline{\Sigma}, = 1)$ because the derivations are the same in the two systems, the only difference is that in $\overline{\Sigma}$ both the INS-type rules and the REP-type rules of Σ are contracted into one set.

4.

We have to show that two language classes $\mathcal{L}(E_H(n, = k))$ and $\mathcal{L}(E_H(m, = \ell))$ are incomparable if $k \neq \ell$.

If $k > \ell \geq 1$, then $L_H(n(k), = k) \in \mathcal{L}(E_H(n, = k)) \setminus \mathcal{L}(E_H(m, = \ell))$ holds by Lemma 4.9 (for the definition of the language $L_H(n(k), = k)$ see Example 4.5).

On the other hand, first we deal with the case $2 \leq \ell < k$ and we prove that $L_H(m(\ell), = \ell) \in \mathcal{L}(E_H(m, = \ell)) \setminus \mathcal{L}(E_H(n, = k))$ if $2 \leq \ell < k$. First we show that if $L_H(m(\ell), = \ell) = L(\Sigma, = k)$ with a hybrid eco-grammar system $\Sigma = (\, V_E, P_E, n_1, R_1, \ldots, R_n, \omega\,)$, then P_E does not contain the rule $a_1 \to \lambda$.

If P_E has this rule, then the REP-type agents could not introduce letters c, since by using a replacement rule which introduce a letter c and the erasing rule of P_E for all letters preceding the first letter c, we would generate a word whose first symbol is a letter c. (Here we use that Lemma 4.7 can be applied for $L_H(m(\ell), = \ell)$ with $L_1 = L_H(m(\ell), = \ell) \cup \{a_1\}^*$ and hence words containing letters c can be derived only from words being over $\{a_1\}^*$).

Because of the same reason, all insertion rules introducing a letter c_i must be in the form $({a_1}^{i_1}, {a_2}^{j_1}\gamma {a_2}^{j_2}, {a_1}^{i_2})$ (with $i_1, i_2, j_1, j_2 \geq 1$, $c_i \preceq \gamma$), so that we could avoid the letters c_i as the first or as the last symbol of the words.

Since the environment cannot introduce letters c (a_1 is a letter with unbounded occurrence), the only possibility to introduce them is to use these insertion rules. But with these rules we cannot introduce sub-words $c_i c_j$, which is a contradiction. (It is not possible either to produce these sub-words by one agent; we can see this by using the same idea as the one used in Lemma 4.9.) Thus we have proved that $a_1 \to \lambda$ is not a possible rule in P_E.

In $L_H(m(\ell), = \ell)$ the shortest word containing letters c is in the form of $a_2[c_1, c_2, \ldots, c_\ell]^P a_2$. This word must be derived in Σ from the word ${a_1}^2$ because this is the only shorter word in the set $L_H(m(\ell), = \ell) \cap {a_1}^*$ (here we use again the fact that we can apply Lemma 4.7 to the language $L_H(m(\ell),=\ell))$.

We know that P_E cannot use erasing rule during this step and we also know that REP-type agents cannot perform any actions on ${a_1}^2$ because it contains only two symbols. It means that in this step at most ℓ (INS-type) agents can work, that is, $k \leq \ell$, which is a contradiction because we supposed that $k > \ell$.

To complete the proof, we show that if $2 \leq k$, then $L = \{a^2, b\} \in \mathcal{L}(E_H(m,=1) \setminus \mathcal{L}(E_H(n,=k))$.
It is clear that L can be generated by the following hybrid EG system $\Sigma = (\ V_E, P_E, m, R_1, \ldots, R_m, \omega\ )$ in mode $=1$:

- $V_E = \{a, b\}$,
- $P_E = \{a \to \lambda, b \to \lambda\}$,
- $R_i = \{(a, b, a)\}$ for $1 \leq i \leq m$, and
- $\omega = a^2$.

On the other hand, if L is generated by a hybrid EG system $\overline{\Sigma} = (\overline{V_E}, \overline{P_E}, \overline{n_1}, \overline{R_1}, \ldots, \overline{R_n}, \overline{\omega})$ in mode $=k$, then a^2 must be the axiom in $\overline{\Sigma}$ because agents cannot apply rules to words consisting of one letter. Thus $a^2 \Rightarrow_{\overline{\Sigma}} b$ is a derivation step in $\overline{\Sigma}$. But in this step at most one agent can be active, which gives $k \leq 1$, a contradiction with the supposition $k \geq 2$. ■

Now we turn our attention to language classes $\mathcal{L}(E_H(n,=k_1,=k_2))$ generated with different n, k_1 and k_2. The results are similar to the $=k$ case: there is a hierarchy according to the number of the agents if k_1 and k_2 are fixed, otherwise the language classes are incomparable.

Theorem 4.11

For $0 \leq k_1, k_2, \ell_1, \ell_2,\ \ 1 \leq k_1 + k_2 \leq n$ *and* $1 \leq \ell_1 + \ell_2 \leq m$

1. $\mathcal{L}(E_H(n, = k_1, = k_2)) \subset \mathcal{L}(E_H(m, = k_1, = k_2))$ *if* $k_1 + k_2 \geq 2$ *and* $m > n$,
2. $\mathcal{L}(E_H(n, = 1, = 0)) = \mathcal{L}(E_H(m, = 1, = 0))$,
3. $\mathcal{L}(E_H(n, = 0, = 1)) = \mathcal{L}(E_H(m, = 0, = 1))$,
4. *otherwise* $\mathcal{L}(E_H(n, = k_1, = k_2))$ *and* $\mathcal{L}(E_H(m, = \ell_1, = \ell_2))$ *are incomparable.*

Proof.

1.

Using the same construction as in part 1 of Theorem 4.10, we obtain that $\mathcal{L}(E_H(n, = k_1, = k_2)) \subseteq \mathcal{L}(E_H(m, = k_1, = k_2))$ if $k_1 + k_2 \geq 1$ and $m > n$. Applying Lemma 4.9 to $L_H(m, = k_1, = k_2)$, we can see that the inclusion is proper if $k_1 + k_2 \geq 2$ and $m > n$ (for the definition of the language $L_H(m, = k_1, = k_2)$ see Example 4.5).

2. and 3.

These parts can be proved by using constructions similar to the ones used in the proof of part 3 of Theorem 4.10: we can contract all the REP-type rules and all the INS-type rules into one set.

4.

We have to prove that the two language classes $\mathcal{L}(E_H(n, = k_1, = k_2))$ and $\mathcal{L}(E_H(m, = \ell_1, = \ell_2))$ are incomparable if $k_1 \neq \ell_1$ or $k_2 \neq \ell_2$. For any $0 \leq k_1, k_2, n$, $1 \leq k_1 + k_2 \leq n$ we present a language which cannot be derived by any hybrid eco-grammar system in mode $(= \ell_1, = \ell_2)$ if $\ell_1 \neq k_1$ or $\ell_2 \neq k_2$.

First we consider the case when $k_1 + k_2 \geq 2$. Applying Lemma 4.9 to $L_H(n, = k_1, = k_2)$ (see Example 4.5), we obtain that if $L_H(n, = k_1, = k_2) = L(\Sigma, = \ell_1, = \ell_2)$ with a hybrid eco-grammar system $\Sigma = (V_E, P_E, m_1, R_1, \ldots, R_m, \omega\ )$, then $\ell_1 + \ell_2 \geq k_1 + k_2$. Since $k_1 + k_2 \geq 2$, we can use the same idea as in part 4 of Theorem 4.10 and we can conclude that P_E cannot contain the rule $a_1 \to \lambda$. Using this fact and examining the rules used in the derivation step ${a_1}^{k_2+2} \overset{(=\ell_1,=\ell_2)}{\Longrightarrow_\Sigma} a_2[c_1, \ldots, c_{k_1}, c_{i_1}, c_{i_2}, \ldots, c_{i_{k_2}}]^P a_2$ (which is the only possibility to derive the latter word in Σ) we obtain that $\ell_2 \leq k_2$ and $\ell_1 \leq k_1$. Since $k_1 + k_2 \leq \ell_1 + \ell_2$, we have $\ell_1 = k_1$ and $\ell_2 = k_2$. This gives that $L_H(n, = k_1, = k_2) \in \mathcal{L}(E_H(n, = k_1, = k_2)) \setminus \mathcal{L}(E_H(m, = \ell_1, = \ell_2))$, if $k_1 + k_2 \geq 2$ and $\ell_1 \neq k_1$ or $\ell_2 \neq k_2$.

For completing the proof, we construct two languages which show the incomparability when $k_1 + k_2 = 1$.

In the case k_1=1, $k_2 = 0$ $L = \{a^2, b\} \in \mathcal{L}(E_H(n, = 1, = 0)) \setminus \mathcal{L}(E_H(m, = \ell_1, = \ell_2))$ holds if $\ell_1 \neq 1$ or $\ell_2 \neq 0$ by the same reason as in part 4 of Theorem 4.10.

In the other case, that is, when k_1=0 and k_2=1, the language generated by the following hybrid EG system Σ in mode $(= 0, = 1)$ shows the incomparability: this language cannot be generated by hybrid EG systems in mode (ℓ_1, ℓ_2) if $\ell_1 \neq 0$ or $\ell_2 \neq 1$.

Let $\Sigma = (\ V_E, P_E, 0, R_1, \ldots, R_n, \omega\ )$ be the following hybrid EG system:

- $V_E = \{a_1, a_2, b_1, b_2, d\}$,
- $P_E = \{a_1 \to {a_2}^2, a_2 \to {a_1}^2, b_1 \to {b_2}^2, b_2 \to {b_1}^2, d \to a_1\}$,
- $R_i = \{(a_1, a_1, b_1) \to (a_1, d, b_1), (a_2, d, b_2) \to (a_2, a_1, b_2)\}$ for $1 \leq i \leq n$,
- $\omega = a_1 a_1 b_1$.

It is clear that the generated language in derivation mode $(=0,=1)$ is

$$L_{REP}(=1) := L(\Sigma,=0,=1) = \{{a_1}^{4^k} a_1 {b_1}^{4^k}, {a_2}^{2\cdot 4^k} d {b_2}^{2\cdot 4^k} \mid 0 \leq k \}.$$

Let us suppose that $L(\Sigma,=0,=1) = L(\overline{\Sigma},=\ell_1,=\ell_2)$ with a hybrid EG system $\overline{\Sigma} = (\,\overline{V_E}, \overline{P_E}, \overline{m_1}, \overline{R_1}, \ldots, \overline{R_m}, \overline{\omega}\,)$. By Lemma 4.7 we can say that in $\overline{\Sigma}$ the words in a derivation sequence are alternately over $\{a_1, b_1\}$ and $\{a_2, b_2, d\}$.

$\overline{P_E}$ cannot introduce the letter d, because in this case more than one letters d would be in some words (since a_1 and b_1 are letters with unbounded occurrence). Moreover, for all the rules $a_1 \to \alpha$ and $b_1 \to \beta$ of $\overline{P_E}$ $\alpha = {a_2}^{i_1}$ and $\beta = {b_2}^{i_2}$ hold, otherwise words with mixed letters a and b or words starting with a letter b or ending with a letter a would be in the generated language. $\overline{P_E}$ cannot contain rules in the form of $a_1 \to \lambda$ or $b_1 \to \lambda$ because in this case the number of letters a_2 or b_2 would be bounded in the generated words. Similarly, for all rules $a_2 \to \alpha$ or $b_2 \to \beta$ of $\overline{P_E}$ $\alpha \in {a_1}^*$ and $\beta \in {b_1}^*$ must hold and $\overline{P_E}$ cannot contain the rules $a_2 \to \lambda$ and $b_2 \to \lambda$.

Since $\overline{P_E}$ cannot contain erasing rules, the only possibility to derive the word ${a_2}^2 d {b_2}^2$ is the derivation step ${a_1}^2 b_1 \Rightarrow_\Sigma {a_2}^2 d {b_2}^2$.

Considering the lengths of these two words, there are five possibilities for (ℓ_1, ℓ_2):

1. $\ell_1 = 0,\ \ell_2 = 1$
2. $\ell_1 = 1,\ \ell_2 = 0$
3. $\ell_1 = 1,\ \ell_2 = 1$
4. $\ell_1 = 2,\ \ell_2 = 0$
5. $\ell_1 = 2,\ \ell_2 = 1.$

Case $\ell_2 = 0$ is not possible, i.e. at least one REP-type agent must apply a rule in each derivation step, because otherwise we could not determine the exact position of the letter d and we could derive words where the letter d does occur in the middle of a sequence of letters a or b. This excludes cases 2 and 4.

In case 5 the replacement rule used in the derivation step must be $(a_1, a_1, b_1) \to (a_1, d, b_1)$. (The letter d cannot be introduced by insertion because in this case we could not determine the exact position of it.) It is also sure that we have to use the rule (a_1, b_2, b_1) or $(a_1 a_1, b_2, b_1)$ for introducing a letter b_2. But with this last insertion rule we could also insert a letter b_2 in the middle of a sequence of letters a_2, which is a contradiction.

In case 3 when $\ell_1 = \ell_2 = 1$, we have two possibilities for the rules in the derivation step $a_1{}^2 b_1 \Rightarrow_\Sigma a_2{}^2 d b_2{}^2$. The letter d is introduced by a replacement and either the environment uses the rule $a_1 \to a_2{}^2$ and one INS-type agent introduces a b_2 by a rule of type (a_1, b_2, b_1), $(a_1 a_1, b_2, b_1)$ or the environment uses the rule $a_1 \to a_2$ and one INS-type agent uses the rule (a_1, a_2, a_1), (a_1, a_2, b_1), or $(a_1, a_2, a_1 b_1)$.

The first case is not possible because of the same reason as in case 5. The second case implies that $a_1 \to a_2$ is the only rule in $\overline{P_E}$ to rewrite a_1, otherwise more then one word in the form of $a_2{}^i d b_2{}^2$ could be generated. But in this case for all derivation steps $a_1{}^{4^k} a_1 b_1{}^{4^k} \Rightarrow_{\overline{\Sigma}} a_2{}^{2\cdot 4^j} d b_2{}^{2\cdot 4^j}$ the difference $2 \cdot 4^j - 4^k$ would be bounded, which is impossible. Thus the only possibility is $\ell_1 = 0, \ell_2 = 1$. ∎

Finally, we show that hybrid EG systems are more powerful than homogeneous INS- or REP-type systems.

Remark 4.12

$\mathcal{L}(E_I) \cup \mathcal{L}(E_R) \subset \mathcal{L}(E_H)$

PROOF.

The inclusion holds by definition. On the other hand, using the same language and the same idea as in the proof of part 2 of Theorem 4.10, we obtain that the language defined there cannot be generated by a hybrid EG system which contains only INS- or only REP-type agents. ∎

4.2 Simple eco-grammar systems of type INS and REP

Simple eco-grammar systems with only INS- or with only REP-type agents are special cases of hybrid eco-grammar systems. In this part of the chapter we present some results about these homogeneous systems. The first results are almost direct consequences of the results presented in the first part of the chapter, we present them however again, because we find it worth showing the similarity between them and the results presented in Section 3.1.

Definition 4.13

A simple eco-grammar system of type INS is a hybrid eco-grammar system $\Sigma = (V_E, P_E, n_1, R_1, \dots, R_n, \omega)$ *where* $n_1 = n$. *A simple eco-grammar system of type REP is a hybrid eco-grammar system* $\Sigma = (V_E, P_E, n_1, R_1, \dots, R_n, \omega)$ *where* $n_1 = 0$.

Since in simple eco-grammar systems of type INS or REP the value of n_1 is determined by the type of the system, we omit it in the notation and thus a simple eco-grammar system of type INS or REP is denoted by $\Sigma = (V_E, P_E, R_1, \ldots, R_n, \omega)$.

Those simple eco-grammar systems, where the agents are represented by sets of CF rules will be referred as EG systems of type CF in this section.

We want to define the derivation mode $= k$ $(1 \leq k \leq n)$ for simple eco-grammar systems of type INS and REP in the similar way as it was presented in the CF case. We do not have to introduce this derivation mode as a new one, because we can consider it as a special case of Definition 4.3. In this way the same derivation mode is defined as in Definition 3.1 for simple eco-grammar systems of type CF.

Definition 4.14

Consider a simple eco-grammar system $\Sigma = (V_E, P_E, R_1, \ldots, R_n, \omega)$ of type INS. We say that x directly derives y in Σ in the derivation mode $= k$ (with $x, y \in {V_E}^*$, $1 \leq k \leq n$, written as $x \stackrel{=k}{\Longrightarrow}_\Sigma y$), if $x \stackrel{=k,=0}{\Longrightarrow}_{\overline{\Sigma}} y$ holds with the hybrid eco-grammar system $\overline{\Sigma} = (V_E, P_E, n, R_1, \ldots, R_n, \omega)$.

Similarly:

Definition 4.15

Consider a simple eco-grammar system $\Sigma = (V_E, P_E, R_1, \ldots, R_n, \omega)$ of type REP. We say that x directly derives y in Σ in the derivation mode $= k$ (with $x, y \in {V_E}^*$, $1 \leq k \leq n$), written as $x \stackrel{=k}{\Longrightarrow}_\Sigma y$), if $x \stackrel{=0,=k}{\Longrightarrow}_{\overline{\Sigma}} y$ holds with the hybrid eco-grammar system $\overline{\Sigma} = (V_E, P_E, 0, R_1, \ldots, R_n, \omega)$.

We denote the transitive and reflexive closure of $\stackrel{=k}{\Longrightarrow}_\Sigma$ by $\stackrel{=k}{\Longrightarrow}^*_\Sigma$.

Definition 4.16

Consider a simple eco-grammar system $\Sigma = (V_E, P_E, R_1, \ldots, R_n, \omega)$ of type

X ($X \in \{INS, REP\}$). *The generated language in the derivation mode* $= k$ *is the following:*

$$L(\Sigma, = k) = \{\, v \in {V_E}^* \,|\, \omega \overset{=k}{\Longrightarrow}{}^*_\Sigma\, v \,\}$$

As already presented in Section 4.1, the class of languages which can be generated by a simple eco-grammar system of type X is denoted by $\mathcal{L}(E_X)$ for $X \in \{\mathrm{CF}, \mathrm{INS}, \mathrm{REP}\}$. The class of languages generated by systems containing n agents and operating in the derivation mode $= k$ is denoted by $\mathcal{L}(E_X(n, = k))$ for $X \in \{\mathrm{CF}, \mathrm{INS}, \mathrm{REP}\}$.

In the following we summarise some consequences of Theorem 4.11. In these results language classes $\mathcal{L}(E_X(n, = k))$ with fixed X ($X \in \{\mathrm{INS}, \mathrm{REP}\}$) and with different parameters k and n, with $1 \leq k \leq n$, are compared. We can see that the results are similar to the ones presented in Chapter 3 about EG systems of type CF.

To begin with, results concerning simple eco-grammar systems of type INS are presented.

Theorem 4.17

For $1 \leq k \leq n$ *and* $1 \leq \ell \leq m$

1. $\mathcal{L}(E_{INS}(n, = k)) \subset \mathcal{L}(E_{INS}(m, = k))$ *if* $k \geq 2$ *and* $m > n$,

2. $\mathcal{L}(E_{INS}(n, = 1)) = \mathcal{L}(E_{INS}(m, = 1))$,

3. *otherwise* $\mathcal{L}(E_{INS}(n, = k))$ *and* $\mathcal{L}(E_{INS}(m, = \ell))$ *are incomparable.*

Proof.

These statements are corollaries of Theorem 4.11. The equality and the inclusions are true because $\mathcal{L}(E_H(n, = k_1, = 0)) \subset \mathcal{L}(E_H(m, = k_1, = 0))$ holds for $m \geq n \geq k_1 \geq 2$, $\mathcal{L}(E_H(n, = 1, = 0)) = \mathcal{L}(E_H(m, = 1, = 0))$ holds for $n, m \geq 1$ and because for each hybrid eco-grammar system working in the derivation mode $(=k_1,=0)$ there exists another hybrid eco-grammar system containing the same number of agents, all of which are of type INS, working also in the derivation mode $(=k_1,=0)$ and generating the same language.

The incomparability also follows from the incomparability results of Theorem 4.11 and from the above fact that we can suppose that in a hybrid eco-grammar system working in the derivation mode $(=k_1,=0)$ there are only INS-type agents. ■

About simple eco-grammar systems of type REP we have similar results.

Theorem 4.18

For $1 \leq k \leq n$ and $1 \leq \ell \leq m$

1. $\mathcal{L}(E_{REP}(n,=k)) \subset \mathcal{L}(E_{REP}(m,=k))$ *if* $k \geq 2$ *and* $m > n$,
2. $\mathcal{L}(E_{REP}(n,=1)) = \mathcal{L}(E_{REP}(m,=1))$,
3. *otherwise* $\mathcal{L}(E_{REP}(n,=k))$ *and* $\mathcal{L}(E_{REP}(m,=\ell))$ *are incomparable.*

PROOF.
These statements follow also from Theorem 4.11. Here we have to use the results in the case $(=0,=k_2)$ and the fact that we can suppose that in a hybrid eco-grammar system working in the derivation mode $(=k_1,=0)$ there are only REP-type agents. ■

In the following, we compare the language classes $\mathcal{L}(E_{CF})$, $\mathcal{L}(E_{INS})$, $\mathcal{L}(E_{REP})$ and prove that they are pairwise incomparables.

Lemma 4.19

$\mathcal{L}(E_{INS})$ and $\mathcal{L}(E_{REP})$ are incomparables.

PROOF.

The languages $L = \{ a, b^2 \}$ and $L_{REP}(=1)$ (for the definition see part 4 of Theorem 4.11) proves the incomparability by the same reason as in part 4 of Theorem 4.10 and 4.11. ■

For the incomparability results concerning $\mathcal{L}(E_{CF})$, we need some special languages. They have a structure similar to the one presented in Example 4.5 but they have some new features as well.

Example 4.20

Let $\Sigma = (V_E, P_E, R_1, \ldots, R_n, \omega)$ *be the following simple eco-grammar system of type INS for* $1 \leq k \leq n$*:*

- $V_E = \{a_1, b_1, a_2, b_2, c_1, \ldots, c_n\}$,
- $P_E = \{b_1 \to b_2, a_1 \to a_2, b_2 \to b_1, a_2 \to a_1\} \cup \bigcup_{1 \leq j \leq n} \{c_j \to a_1\}$,
- $R_i = \{(b_1, c_i, a_1), (b_2, a_1, a_2)\}$ *for* $1 \leq i \leq n$,
- $w = a_1 b_1 {a_1}^k$.

Having examined the system and the derivation process, we can see that in the derivation mode $= k$, after $2i$ $(i \geq 0)$ derivation steps we have the word $a_1 b_1 {a_1}^{k+2ik}$; after $2i+1$ $(i \geq 0)$ derivation steps the sentential form is in the form of $a_2 b_2 [c_{i_1}, c_{i_2}, \ldots, c_{i_k}, a_2, a_2, \ldots, a_2]^P a_2$, where the letters c are different.

Requirements of Lemma 4.8 hold for the generated language $L_{INS}(n, = k) := L(\Sigma, k)$ and we note that in the words being over the second subalphabet $\{a_2, b_2, c_1, \ldots, c_n\}$ the only letter b_2 precedes all occurrences of the letters c. We also note that the distance between b_2 and the letters c is not bounded in the words over $\{a_2, b_2, c_1, \ldots, c_n\}$, that is, for any $M_0 > 0$ there exists a word over this sub-alphabet where the sub-word between b_2 and the first letter c is longer than M_0.

Lemma 4.21

For $1 \leq k \leq n$

$$L_{INS}(n, = k) \in \mathcal{L}(E_{INS}(n, = k)) \setminus \mathcal{L}(E_{CF}).$$

PROOF.

Since $L_{INS}(n, = k)$ (defined above in Example 4.20) has the properties required by Lemma 4.8, if $L_{INS}(n, = k) = L(\Sigma, = \ell)$ with a simple eco-grammar system $\Sigma = (\ V_E, P_E, R_1, \dots, R_m, \omega\ )$ of type CF, then in each derivation sequence words over $\mathrm{alph}(L_1) = \{a_1, b_1\}$ are followed by words over $\mathrm{alph}(L_2) = \{a_2, b_2, c_1, \dots, c_n\}$, and conversely.

We examine how the system Σ can introduce letters c during a derivation step $v_1 \Rightarrow_\Sigma v_2$ where $v \in \{a_1, b_1\}^*$ and $v_2 \in \{a_2, b_2, c_1, \dots, c_n\}^*$.

- P_E cannot introduce letters c because in this case more than k letters c could occur in the words of $L(\Sigma)$.

- For any i, $1 \leq i \leq n$ there must be in Σ at least one agent-rule in the form of $a_1 \to \alpha c_i \beta$ with $\alpha, \beta \in (\mathrm{alph}(L_2))^*$ because if there is a letter c_i which can be derived only from b_1, then this letter c_i could not occur at distance of any length from the beginning of the word (considering that the position of b_1 is fixed).

- In P_E there must be at least one rule in the form of $a_1 \to u$, with $u \neq \lambda$, otherwise the lengths of the words in L_2 would be bounded.

- For any agent-rule being in the form of $a_1 \to \alpha c_i \beta$, it follows that $|\alpha|_{b_2} = 0$ holds. Otherwise, by applying this rule to the last letter a_1

of a long word in L_1 and by applying a P_E rule in the from of $a_1 \rightarrow u$, $u \in (\mathrm{alph}(L_2))^+$ to all the other letters which are not concerned by the actions of the agents, we would get a word where b_2 is not standing at the second position of the word.

So we have $|\alpha|_{b_2} = 0$. But we can also apply this rule $a_1 \rightarrow \alpha c_i \beta$ to the first letter a_1 of a word in L_1 and so we would get a word where a letter c_i stands before b_2. It is a contradiction because there are no words like this in $L_{INS}(n, = k)$.

■

Example 4.22

Let $\Sigma = (V_E, P_E, R_1, \dots, R_n, \omega)$ *be the following simple eco-grammar system of type CF for* $1 \leq k \leq n$*:*

- $V_E = \{a_1, b_1, a_2, b_2, c_1, \dots, c_n\}$,
- $P_E = \{b_1 \rightarrow {b_2}^2, a_1 \rightarrow {a_2}^2, b_2 \rightarrow {b_1}^2, a_2 \rightarrow {a_1}^2\} \cup \bigcup_{1 \leq j \leq n} \{c_j \rightarrow a_1\}$,
- $R_i = \{a_1 \rightarrow c_i, c_i \rightarrow a_1\}$ *for* $1 \leq i \leq n$,
- $w = b_1 {a_1}^{1+k}$.

In the derivation mode $= k$, after $2s$ $(s \geq 0)$ derivation steps we have the word ${b_1}^{4^s} {a_1}^{4^s+k}$; after $2s+1$ $(s \geq 0)$ derivation steps we have a word in the form of ${b_2}^{2 \cdot 4^s} [c_{j_1}, c_{j_2}, \dots, c_{j_k}, {a_2}^2, {a_2}^2, \dots, {a_2}^2]^P$ (in the latter formula the letters c are different).

The requirements of Lemma 4.7 holds for the generated language $L_{CF}(n, = k) := L(\Sigma, = k)$ and we note that in the words being over the second sub-alphabet $\{a_2, b_2, c_1, \ldots, c_n\}$ the letters c can occur anywhere after the sequence of letters b.

Lemma 4.23

For $1 \leq k \leq n$

$$L_{CF}(n, = k) \in \mathcal{L}(E_{CF}(n, = k)) \setminus \mathcal{L}(E_{INS}).$$

PROOF.

Let us suppose that $L_{CF}(n, = k) = L(\Sigma, = \ell)$ with a simple eco-grammar system $\Sigma = (\ V_E, P_E, R_1, \ldots, R_m, \omega\ )$ of type INS (where $L_{CF}(n, = k)$ is defined in Example 4.22 above).

As mentioned above, $L_{CF}(n, = k)$ has all the properties required by Lemma 4.7, therefore we can say that if $u \stackrel{=k}{\Longrightarrow}_\Sigma v$ holds, then u and v are over the different sub-alphabets $\{\ b_1, a_1\ \}$ and $\{\ b_2, a_2, c_1, \ldots, c_n\ \}$.

We examine the rules of P_E which can be used to generate words in $L_{CF}(n, = k) \cap \{a_2, b_2, c_1, \ldots, c_n\}^*$ from words being over $\{a_1, b_1\}$. These rules are in the form of $a_1 \rightarrow \alpha$ or $b_1 \rightarrow \beta$, where $\alpha, \beta \in \{a_2, b_2, c_1, \ldots, c_n\}^*$.

- Neither α nor β can contain letters c because in this case more than k letters c could stand in some generated words.

- Neither α nor β can contain letters a_2 and b_2 together because in this case we would be able to derive words with mixed letters a_2 and b_2.

- α cannot be in the form of ${b_2}^i$, where $i \geq 1$ and β cannot be in the form of ${a_2}^j$, where $j \geq 1$ because in these cases b_2 could stand in the last or a_2 could stand in the first position of a word in the generated language.

- Thus $\alpha = {a_2}^i$ and $\beta = {b_2}^j$, $i, j \geq 0$. Moreover, there must be at least one rule with $i, j \geq 1$, otherwise the number of letters a_2 and b_2 would be bounded in the generated words.

We have seen that only agents can introduce letters c during a derivation, so the rules which can introduce letters c are insertion rules and are in one of the following forms (see the words of $L_{CF}(n, = k) \cap \{a_1, b_1\}^*$):

(1) $({b_1}^{i_1}{a_1}^{i_2}, \gamma c_i \delta, {a_1}^{i_3})$

(2) $({b_1}^{i_1}, \gamma c_i \delta, {b_1}^{i_2})$

(3) $({b_1}^{i_1}, \gamma c_i \delta, {b_1}^{i_2}{a_1}^{i_3})$

(4) $({b_1}^{i_1}, \gamma c_i \delta, {a_1}^{i_2})$

(5) $({a_1}^{i_1}, \gamma c_i \delta, {a_1}^{i_2})$

where $i_1, i_2, i_3 \geq 1$ and $\gamma, \delta \in \{a_2, b_2, c_1, \ldots, c_n\}^*$.

In the cases (2), (3) and (4), a letter c_i could stand before letters b_2 if we use the rule $b_1 \to {b_2}^j$, $j \geq 1$, of P_E.

So all the rules producing letters c must be in the form of (1) or (5) but from this fact it follows that P_E must contain a deletion rule in the form of $a_1 \to \lambda$ because this is the only way to introduce a letter c just after the sequence of letters b. If there exists a deletion rule in P_E in the form of $a_1 \to \lambda$, then from a word long enough, being over $\mathrm{alph}(L_1)$, we could derive a word which would contain a lot of letters b_2 but only a few letters a_2 (by using this deletion rule and the rule $b_1 \to {b_2}^j$, $j \geq 1$ at the same step). It is a contradiction because in $L_{CF}(n, = k)$, in each word, the number of letters a_2 is at least the half of the number of letters b_2. ∎

Example 4.24

Let $\Sigma = (V_E, P_E, R_1, \ldots, R_n, \omega)$ *be the following simple eco-grammar system of type REP for* $2 \leq k \leq n$*:*

- $V_E = \{a_1, a_2, c_1, \ldots, c_n\}$,
- $P_E = \{a_1 \to {a_2}^2, a_2 \to {a_1}^2\} \cup \bigcup_{1 \leq j \leq n} \{c_j \to a_1\}$,
- $R_i = \{(a_1, a_1, a_1) \to (a_1, c_i, a_1)\}$
  $\cup \{(e, c_i, f) \to (e, a_1, f) | e, f \in \{a_1, c_1, \ldots, c_n\}\}$ *for* $1 \leq i \leq n$,
- $w = {a_1}^{2+k}$.

In mode $= k$, the generated language $L_{REP}(n, = k) := L(\Sigma, = k)$ contains words in the form of ${a_1}^{2 \cdot 4^i + k}$ or $a_2[c_{i_1}, \ldots, c_{i_k}, a_2, \ldots, a_2]^P a_2$ and satisfies the requirements of Lemma 4.8. In each word over the second sub-alphabet $\{a_2, c_1, \ldots, c_n\}$ the letters c can occur anywhere except the first and the last position.

Example 4.25

Let $\Sigma = (V_E, P_E, R_1, \ldots, R_n, \omega)$ *be the following simple eco-grammar system of type REP:*

- $V_E = \{a_1, a_2, b_1, b_2, c_1, \ldots, c_n\}$,
- $P_E = \{a_1 \to {a_2}^2, a_2 \to {a_1}^2, b_1 \to {b_2}^2, b_2 \to {b_1}^2\} \cup \bigcup_{1 \leq j \leq n} \{c_j \to a_1\}$,
- $R_i = \{(b_1, a_1, a_1) \to (b_1, c_i, a_1)\} \cup \{(b_2, c_i, a_2) \to (b_2, a_1, a_2)\}$
  for $1 \leq i \leq n$,
- $w = b_1 a_1 a_1$.

The first few steps of a derivation in derivation mode $=1$ can be the following:

$$b_1a_1a_1 \stackrel{=1}{\Longrightarrow}_\Sigma {b_2}^2c_{i_1}{a_2}^2 \stackrel{=1}{\Longrightarrow}_\Sigma {b_1}^4a_1{a_1}^4 \stackrel{=1}{\Longrightarrow}_\Sigma {b_2}^8c_{i_2}{a_2}^8 \stackrel{=1}{\Longrightarrow}_\Sigma \cdots$$

Hence,

$$L_R(n,=1) := L(\Sigma,=1) = \{{b_1}^{4^s}a_1{a_1}^{4^s} \mid 1 \leq s\} \cup \{{b_2}^{2\cdot 4^s}c_i{a_2}^{2\cdot 4^s} \mid 1 \leq s, 1 \leq i \leq n\}$$

The generated language $L_R(n,=1)$ satisfies requirements of Lemma 4.8 and the letters c can occur only between the sequence of letters b_2 and a_2 in the words being over $\{a_2, b_2, c_1, \ldots, c_n\}$.

Lemma 4.26

1. *For* $2 \leq k \leq n$ $\;L_{REP}(n,=k) \in \mathcal{L}(E_{REP}(n,=k)) \setminus \mathcal{L}(E_{CF})$.
2. *For* $1 \leq n$ $\;L_R(n,=1) \in \mathcal{L}(E_{REP}(n,=1)) \setminus \mathcal{L}(E_{CF})$.

PROOF.

1.

Suppose that $L_{REP}(n,=k) = L(\Sigma,=\ell)$ holds with a simple eco-grammar system $\Sigma = (\, V_E, P_E, R_1, \ldots, R_m, \omega\,)$ of type CF (where $L_{REP}(n,=k)$ is defined in Example 4.24 above).

First, we note that letters c can be introduced only by agents, otherwise more than k letters c could be in the generated words. Hence letters c can be introduced only by the agent-rules being in the form of $a_1 \to \alpha c_i \beta$, where $\alpha, \beta \in \{a_2, b_2, c_1, \ldots, c_n\}^*$ (since we can apply Lemma 4.8 to $L_{REP}(n,=k)$ with $\mathrm{alph}(L_1) = \{a_1\}$).

Since $k \geq 2$, there exist two different letters c in V_E, c_1 and c_2, and these letters can occur at distance of any length from each other in the words of

$L_{REP}(n, = k)$. Therefore $\ell = 1$ is not possible. Since $\ell \geq 2$, for each letter c there exists at most one agent which can introduce this letter c, otherwise words with two letters c with the same subscript could be generated.

One agent could not introduce two different letters c either because in this case the distance between these letters c would be bounded in the generated words.

Thus in an agent-rule in the form of $a_1 \to \alpha c_i \beta$ $(1 \leq i \leq m)$ α cannot contain a letter c, therefore $\alpha = {a_2}^j$, $j \geq 0$. Moreover, $j = 0$ is not possible because in this case a letter c_i could stand at the first position of a generated word (by applying this agent-rule to the first letter of a word).

Hence we obtain that any rule, used by the agents to introduce letters c, is in the form of $a_1 \to {a_2}^j c_i \beta$, $j \geq 1$ and $\beta \in \{a_2\}^*$. But in this case words like $x c_i c_j y$ $(x, y \in \{a_2, c_1, \ldots, c_n\}^*)$ are impossible to derive, which is a contradiction.

2.

Let us suppose that $L_R(n, = 1) = L(\Sigma, = \ell)$ holds with a simple eco-grammar system $\Sigma = (V_E, P_E, R_1, \ldots, R_m, \omega)$ of type CF (where $L_R(n, = 1)$ is defined in Example 4.25 above). We can apply Lemma 4.8 to $L_R(n, = 1)$, so the words in a derivation sequence are alternately over $\{a_1, b_1\}$ and $\{a_2, b_2, c_1, \ldots, c_n\}$. Following the line of the proof of Lemma 4.23, we can say that the rules of P_E used to produce words being over the sub-alphabet $\{a_2, b_2, c_1, \ldots, c_n\}$ are either in the form $a_1 \to {a_2}^j$ or in the form $b_1 \to {b_2}^i$, where $i, j \geq 0$. We can also say that there are rules of the above form with $i, j \geq 1$.

The agents can introduce letters c only by rules in the form of $a_1 \to \alpha c_i \beta$ or $b_1 \to \alpha c_i \beta$, where $\alpha, \beta \in \{a_2, b_2, c_1, \ldots, c_n\}^*$. By applying a rule $b_1 \to \alpha c_i \beta$ to the first b_1 of a word of $L_1 = L_R(n, = 1) \cap \{a_1, b_1\}^*$ and the rule $b_1 \to {b_2}^j$,

$j \geq 1$ to all other letters b_1 which are not concerned by the actions of the agents, we obtain a word where a letter c stands before a letter b_2. Similarly, by applying a rule $a_1 \to \alpha c_i \beta$ to the last letter a_1 and the rule $a_1 \to {a_2}^i$, with $i \geq 1$, to the remaining letters a_1, we get a word where a letter c is after a letter a_2. In both cases we generate a word which is not in $L_R(n, = 1)$. ■

Example 4.27

Let $\Sigma = (V_E, P_E, R_1, \ldots, R_n, \omega)$ *be the following simple eco-grammar system of type CF for* $1 \leq k \leq n$*:*

- $V_E = \{a_1, a_2, c_1, \ldots, c_n\}$,
- $P_E = \{a_1 \to a_2, a_2 \to a_1\} \cup \bigcup_{1 \leq i \leq n} \{c_i \to a_1\}$,
- $R_i = \{a_1 \to {c_i}^2, c_i \to a_1\}$, *for* $1 \leq i \leq n$,
- $w = {a_1}^k$.

The generated language $L_{n,=k} := L(\Sigma, = k)$ in the derivation mode $= k$ contains two types of words: ${a_1}^+$ and $[{c_{i_1}}^2, {c_{i_2}}^2, \ldots, {c_{i_k}}^2, a_2, a_2, \ldots, a_2]^P$, with different letters c. The generated language has the properties required in Lemma 4.7.

Lemma 4.28

For $1 \leq k \leq n$

$$L_{n,=k} \in \mathcal{L}(E_{CF}(n, = k)) \setminus \mathcal{L}(E_{REP}).$$

Proof.

If $L_{n,=k} = L(\Sigma, = \ell)$ holds with an eco-grammar system $\Sigma = (V_E, P_E, R_1, \ldots, R_m, \omega)$ of type REP (where $L_{n,=k}$ is defined in Example 4.27 above), then the words over $\{a_2, c_1, \ldots, c_n\}$ are derived in Σ from words over $\{a_1\}$ (since we can apply Lemma 4.7 for $L_{n,=k}$). Moreover, similarly to the previous proofs, we can say that only agents can introduce letters c. Since each agent can introduce at most one letter c, Σ has to have for each $1 \leq i \leq n$ at least two agents which can introduce the same letter c_i and $\ell \geq 2$ must hold as well.

These c_i-producing agent-rules must be in the form of $(a_1, a_1, a_1) \rightarrow (a_1, c_i, a_1)$ but in this way we cannot guarantee that they are applied at neighbouring positions, thus we could also derive words with separated letters c_i. ■

The consequence of the above lemmas is the following theorem:

Theorem 4.29

Language classes $\mathcal{L}(E_{INS})$, $\mathcal{L}(E_{REP})$, and $\mathcal{L}(E_{CF})$ are pairwise incomparable.

PROOF.

It is a direct consequence of Lemma 4.19, 4.21, 4.23, 4.26, and 4.28. ■

Chapter 5

Generative power of variants of simple eco-grammar systems

In Chapter 3 results about team behaviour of extended simple eco-grammar systems were collected (see Figure 3.2). There we found that in the general case (i.e. when λ-rules are allowed) the hierarchy of extended simple eco-grammar language classes is a collapsing one: all the language classes are equal to $\mathcal{L}(EE^{\lambda}(1, = 1))$.

The place of this language class remained an open problem, the only known result is that it is strictly larger than the class of languages generated by extended 0L systems (see Example 2.7) and that it is included in the class of languages generated by matrix grammars with appearance checking with λ-rules ($\mathcal{M}_{ac}^{\lambda}$) (see (Dassow and Mihalache 1995)).

The aim of our research was to determine more precisely the generative power of extended simple eco-grammar systems. By examining this question we found two variants of the original model which have universal generative power: the extended tabled simple eco-grammar system and the weak extended simple eco-grammar system.

In an extended tabled simple eco-grammar system instead of only one 0L system, the environment can be represented by more than one set: these 0L systems are called tables. In weak extended simple eco-grammar systems the derivation is different from the original model: in the weak derivation mode the derivation is not blocked if some agents cannot perform any action on the sentential form.

By showing that both models reach the generative power of Turing machines, we solve an open problem in a stronger form as stated in (ter Beek 1999). The results appeared in (Csima 2000).

At the beginning of the chapter, first we summarise some preliminaries about random-context grammars which we use in the following constructions.

5.1 Remarks on random-context grammars

Before starting to study the generative power of some variants of extended simple eco-grammar systems, we recall the notion of random-context grammars, which is used in the constructions of the chapter.

Definition 5.1

A random-context grammar is a quadruple $G = (N, T, P, S)$ where N is the finite set of non-terminals, T is the finite set of terminals, $S \in (N \cup T)^+$ is the axiom, and P is a finite set of random-context rules, that is, triplets in the form of $(C \to \alpha, Q, R)$, $C \to \alpha$ is a CF rule over $N \cup T$, where $C \in N$, and Q and R are sub-sets of N. For $x, y \in (N \cup T)^*$, we write $x \overset{rc}{\Longrightarrow} y$ iff $x = x_1Cx_2$, $y = x_1\alpha x_2$ for some $x_1, x_2 \in (N \cup T)^*$, $(C \to \alpha, Q, R)$ is a triplet in P, all symbols of Q appear and no symbol of R appears in x_1x_2 (Q is called the permitting context, and R is called the forbidding context of the rule $C \to \alpha$. If Q and/or R are empty, then no check is necessary.)

If the forbidding context is empty for every rule, then we speak about a random-context grammar without appearance checking, otherwise the grammar is with appearance checking.

In the dissertation all random-context grammars are with appearance checking, so for the sake of brevity sometimes we simply refer to a random-context grammar with appearance checking and with λ-rules as a random-context grammar.

We denote the transitive and reflexive closure of $\stackrel{\text{rc}}{\Longrightarrow}$ by $\stackrel{\text{rc}}{\Longrightarrow}^*$. The generated language of a random-context grammar consists of all the words which can be generated in some steps from axiom S.

Definition 5.2

The generated language of a random-context grammar $G = (N, T, P, S)$ is the following:

$$L(\Sigma) = \{ \, v \in T^* \mid S \stackrel{rc}{\Longrightarrow}^* v \, \}.$$

The class of languages which can be generated by random-context grammars with appearance checking is denoted by $\mathcal{RC}_{ac}^{\lambda}$. It is well known (see (Dassow and Păun 1989)) that $\mathcal{RC}_{ac}^{\lambda} = \mathcal{RE}$, that is, all recursively enumerable languages can be generated by random-context grammars with appearance checking and with λ-rules.

Based on this result, we prove the universality of our models by simulating random-context grammars with different variants of simple eco-grammar systems. In these simulations we use another definition for random-context grammars, so first we show that the relation $\mathcal{RC}_{ac}^{\lambda} = \mathcal{RE}$ holds for this modified definition, too.

Informally speaking, in a random-context grammar we check the appearance of the permitting and the forbidding letters and if the result is positive (there are no forbidding letters in the sentential form and all the permitting ones are present), we apply the corresponding rule. In Definition 5.1 this check is done over x_1x_2, that is, the symbol which is about to be rewritten is left out. In such a way we can make such a requirement that the letter on the left-hand side of a rule appears at least twice in the current sentential form or that the left-hand side appears at most once.

Another possibility is to check the appearance of the symbols in the whole sentential form, this case is presented in the following definition.

Definition 5.3

A random-context grammar with complete checking is a quadruple $G = (N, T, P, S)$, where N, T,S, and P are the same as in Definition 5.1. For $x, y \in (N \cup T)^*$, we write $x \overset{rc_c}{\Longrightarrow} y$ iff $x = x_1Cx_2$, $y = x_1\alpha x_2$ for some $x_1, x_2 \in (N \cup T)^*$, $(C \to \alpha, Q, R)$ is a triplet in P, all symbols of Q appear and no symbol of R appears in x_1Cx_2.

The generated language is defined as in Definition 5.2.

Definition 5.4

The generated language of a random-context grammar with complete checking $G = (N, T, P, S)$ is the following:

$$L(\Sigma) = \{\, v \in T^* \mid S \overset{rc_c}{\Longrightarrow}{}^* \, v \,\}.$$

Here we denote the transitive and reflexive closure of $\overset{\mathrm{rc_c}}{\Longrightarrow}$ by $\overset{\mathrm{rc_c}}{\Longrightarrow}{}^*$.

In the following we present two auxiliary statements which make it possible to suppose that a random-context grammar has some special properties. These statements are mentioned in several places in the literature, for example in (Dassow and Păun 1989), but without proof. Therefore we decided to present the proofs here.

The first lemma shows that all the languages which can be generated by random-context grammars working according to Definition 5.1, can also be generated by random-context grammars with complete checking, that is, by random-context grammars working according to Definition 5.3.

In the following, for the sake of simplicity, when a set contains only one element, C, for the set we use the notation C instead of the notation $\{C\}$.

Lemma 5.5

For any random-context grammar $G = (N, T, P, S)$ we can construct a random-context grammar with complete checking $\overline{G} = (\overline{N}, \overline{T}, \overline{P}, \overline{S})$, such that $L(G) = L(\overline{G})$.

PROOF.

We denote by r the number of rules in P. The rules of P are enumerated as $p_i = (C_i \to \alpha_i, Q_i, R_i)$. We refer to the components of the ith rule as C_i, α_i, Q_i, and R_i.

Let the components of $\overline{G}$ be the following:

$\overline{T} = T,$

$\overline{S} = S,$

$\overline{N} = N \cup \{ \overline{C_j} \mid 1 \le j \le r \}.$

For any i $(1 \le i \le r)$, there are some rules in $\overline{P}$ according to the ith rule of P in the following way:

- if for the ith rule of P, $(C_i \to \alpha_i, Q_i, R_i)$, both $C_i \in Q_i$ and $C_i \in R_i$ hold, then we do not put any rule into $\overline{P}$, because a rule like this is never applied in G;

- if for the ith rule of P, $(C_i \to \alpha_i, Q_i, R_i)$, neither $C_i \in Q_i$ nor $C_i \in R_i$ holds, then we put $(C_i \to \alpha_i, Q_i, R_i \cup \{ \overline{C_j} \mid 1 \le j \le n \})$ into $\overline{P}$;

- if for the ith rule of P, $(C_i \to \alpha_i, Q_i, R_i)$, $C_i \in Q_i$ holds but $C_i \in R_i$ does not hold, then we put two rules into $\overline{P}$: $(C_i \to \overline{C_i}, Q_i \setminus \{C_i\}, R_i \cup \{ \overline{C_j} \mid 1 \leq j \leq n \})$ and $(\overline{C_i} \to \alpha_i, C_i, \{ \overline{C_j} \mid 1 \leq j \leq n, i \neq j \})$;

- if for the ith rule of P, $(C_i \to \alpha_i, Q_i, R_i)$, $C_i \in Q_i$ does not hold but $C_i \in R_i$ does, then we put the following two rules into $\overline{P}$: $(C_i \to \overline{C_i}, Q_i, (R_i \setminus \{ C_i \}) \cup \{ \overline{C_j} \mid 1 \leq j \leq n \})$ and $(\overline{C_i} \to \alpha_i, \emptyset, \{C_i\} \cup \{ \overline{C_j} \mid 1 \leq j \leq n, i \neq j \})$.

It follows directly from the construction of $\overline{P}$ that $L(\overline{G}) = L(G)$, since $\overline{G}$ works according to Definition 5.3 and G works according to Definition 5.1. The main idea is that if $C_i \in Q_i$ or $C_i \in R_i$ holds for a rule of P, then we check the appearance of the symbols in two steps. The additional symbols $\overline{C_j}$ are used to prevent the grammar $\overline{G}$ from starting to simulate a rule of P while another rule of P is being simulated. ■

Let us note here that the construction of $\overline{P}$ presented above gives that we can suppose that $C_i \notin Q_i$, $C_i \notin R_i$, and $R_i \cap Q_i = \emptyset$ hold for the rules $(C_i \to \alpha_i, Q_i, R_i)$ of $\overline{P}$.

By the following lemma, we can suppose that both the permitting and the forbidding contexts of the rules of a random-context grammar with complete checking (that is, working according to Definition 5.3) contain at most one symbol.

Lemma 5.6

For any random-context grammar with complete checking $G = (N, T, P, S)$ we can construct another random-context grammar with complete checking $\overline{G} = (\overline{N}, \overline{T}, \overline{P}, \overline{S})$, such that $L(G) = L(\overline{G})$, $|Q| \leq 1$ and $|R| \leq 1$ hold for all the rules $(C \to \alpha, Q, R) \in \overline{P}$.

PROOF.

As in Lemma 5.5, we denote by r the number of rules in P. The rules of P are enumerated as $p_i = (C_i \to \alpha_i, Q_i, R_i)$ and we also use the notations $Q_i = \{ A_{i,1}, \ldots, A_{i,k_i} \}$, where $k_i = |Q_i|$ and $R_i = \{ B_{i,1}, \ldots, B_{i,\ell_i} \}$, where $\ell_i = |R_i|$. We will refer to the components of the ith rule as C_i, α_i, Q_i, R_i and to the members of Q_i and R_i as $A_{i,s}$, $B_{i,t}$, $1 \le s \le k_i$, $1 \le t \le \ell_i$.

Let the components of $\overline{G}$ be the following:

$\overline{T} = T$,

$\overline{S} = SS_0$,

$\overline{N} = N \cup \{ \overline{C_j} \mid 1 \le j \le r \} \cup \{ S_0, S_i^j \mid 1 \le i \le r, 0 \le j \le k_i + \ell_i + 1 \}$.

The set $\overline{P}$ is defined as follows. For each rule $(C_i \to \alpha_i, \{A_{i,1}, \ldots, A_{i,k_i}\}, \{B_{i,1}, \ldots, B_{i,\ell_i}\})$ of P ($k_i = |Q_i|$, $\ell_i = |R_i|$) we put the following $3+k_i+\ell_i+2$ rules into $\overline{P}$:

- $(S_0 \to S_i^0, \emptyset, \emptyset)$,
- $(C_i \to \overline{C_i}, S_i^0, \overline{C_i})$,
- $(S_i^0 \to S_i^1, \overline{C_i}, \emptyset)$,
- $(S_i^s \to S_i^{s+1}, A_{i,s}, \emptyset)$ for $1 \le s \le k_i$,
- $(S_i^{k_i+t} \to S_i^{k_i+t+1}, \emptyset, B_{i,t})$ for $1 \le t \le \ell_i$,
- $(\overline{C_i} \to \alpha_i, S_i^{k_i+\ell_i+1}, \emptyset)$, and
- $(S_i^{k_i+\ell_i+1} \to S_0, \emptyset, \overline{C_i})$.

Moreover, we also put the rule $(S_0 \to \lambda, \emptyset, \emptyset)$ into $\overline{P}$.

All the rules of P can be simulated by the corresponding $k_i + \ell_i + 5$ rules of $\overline{P}$: first we check the appearance of symbols in Q_i, then the non-appearance of symbols in R_i. Thus $L(G) \subseteq L(\overline{G})$. On the other hand, it follows from the construction that the rules of $\overline{P}$ which correspond to the same rule of P, must be used together, one after the other, in the same order as explained above. The forbidding context of the second rule prevents the grammar $\overline{G}$ from starting the simulation of the same rule more than once. Two different rules cannot be simulated at the same time because only one symbol S is in the sentential form and its index determines the rule to be simulated. This gives $L(\overline{G}) \subseteq L(G)$. ∎

We note here, that like in Lemma 5.5, $C_i \notin Q_i$, $C_i \notin R_i$, and $R_i \cap Q_i = \emptyset$ hold for the rules $(C_i \to \alpha_i, Q_i, R_i)$ of $\overline{P}$.

The two lemmas together with the result $\mathcal{RC}_{ac}^{\lambda} = \mathcal{RE}$ gives that any recursively enumerable language can be generated by a random-context grammar which uses complete checking and whose permitting and forbidding contexts contain at most one symbol.

5.2 Extended tabled simple eco-grammar systems

In this section extended tabled simple eco-grammar systems are investigated. In such a system the environment can be represented by more than one 0L system.

Definition 5.7
An extended tabled simple eco-grammar system (an ETEG system, for short) is a construct $\Sigma = (\, V_E, T_E, R_1, \ldots\,, R_n, \omega, \Delta\,)$, *where*

- $(V_E, T_E, \omega, \Delta)$ *is an ET0L system,*
- $n \geq 1$, *and*
- R_i *is a non-empty set of CF rules over* V_E *for* $1 \leq i \leq n$.

Here V_E, n, R_i (for $1 \leq i \leq n$), ω and Δ have the same meaning as in Definition 2.4, that is, these are the alphabet of the system, the number of the agents, the production sets representing the agents, the axiom, and the terminal alphabet. $T_E = \{T_i \mid 1 \leq i \leq k\ \}$ is the set of tables of the environment, where each T_i is a complete set of CF rules over V_E.

In an extended tabled simple eco-grammar system the environment in each derivation step can choose a table to perform a parallel rewriting.

Definition 5.8

Consider an extended tabled simple eco-grammar system $\Sigma = (V_E, T_E, R_1, \ldots, R_n, \omega, \Delta\ )$, *where* $T_E = \{T_i \mid 1 \leq i \leq k\ \}$. *We say that* x *directly derives* y *in* Σ *(with* $x, y \in {V_E}^*$, *written as* $x \overset{t\text{-}eco}{\Longrightarrow}_{\Sigma} y$*) if* $x \overset{eco}{\Longrightarrow}_{\Sigma_i} y$ *holds with the extended simple eco-grammar system* $\Sigma_i = (V_E, T_i, R_1, \ldots, R_n, \omega, \Delta)$ *for some* $1 \leq i \leq k$.

We denote the transitive and reflexive closure of $\overset{\text{t-eco}}{\Longrightarrow}_{\Sigma}$ by $\overset{\text{t-eco}}{\Longrightarrow}{}^*_{\Sigma}$.

Definition 5.9

The generated language of an extended tabled simple eco-grammar system $\Sigma = (\ V_E, T_E, R_1, \ldots, R_n, \omega, \Delta\ )$ *is the following:*

$$L(\Sigma) = \{\ v \in \Delta^* \mid \omega \overset{t\text{-}eco}{\Longrightarrow}{}^*_{\Sigma} v\ \}.$$

Now we present an example to illustrate the power of ETEG systems.

Example 5.10

Let $\Sigma = (\ V_E, T_E, R_1, \omega, \Delta\ )$ *be the following ETEG system:*

- $V_E = \{ S, B, N, K, S', a, b \}$,
- $T_E = \{ T_1, T_2, T_3, T_4 \}$, *where*
  $T_1 = \{ S \to K, N \to N, a \to a, b \to b, K \to K, S' \to S', B \to B \}$,
  $T_2 = \{ S \to \lambda, N \to N, a \to a, b \to b, K \to K, S' \to K, B \to B \}$,
  $T_3 = \{ S \to K, N \to N, a \to a, b \to b, K \to K, S' \to K, B \to bB \}$,
  $T_4 = \{ S \to K, N \to N, a \to a, b \to b, K \to K, S' \to K, B \to b \}$,
- $R_1 = \{ S \to aBNS, N \to \lambda, S' \to \lambda \}$,
- $\omega = SS'$, *and*
- $\Delta = \{ a, b \}$.

We show that the generated language of this system is

$$L(\Sigma) = \{ (ab^n)^m \mid 1 \leq n \leq m \} \cup \{\lambda\}.$$

First we note that S can be deleted only by the environment and S' can be deleted only by the agent, and that these two events must happen in the same derivation step. If the environment deletes S, that is, it uses table T_2, and the agent does not apply the rule $S' \to \lambda$, then the environment rewrites S' to K, which blocks the derivation since this K will never disappear from the sentential form. If the agent deletes S' in a derivation step and the environment does not delete S, that is, does not use table T_2, then it introduces symbol K again.

Before this step, when S and S' disappear from the sentential form, in every step the agent has to use the rule $S \to aBNS$, or otherwise the environment introduces a K, and the environment has to use the first table T_1, or otherwise S' would be rewritten to K. Thus either the derivation is $SS' \overset{\text{t-eco}}{\Longrightarrow} \lambda$, or the first few steps are $SS' \overset{\text{t-eco}}{\Longrightarrow}{}^{*} (aBN)^m SS' \overset{\text{t-eco}}{\Longrightarrow} (aBN)^m$. After these steps the only possibility for the agent is to use rule the $N \to \lambda$.

During these steps the environment can use any of its tables, therefore it can introduce letters b before all the B's or it can rewrite either all letters B to B or all letters B to b. Because there are only m symbols N in the sentential form, the derivation lasts exactly m more steps. Hence at the end of the derivation, when there are no more N symbols left, there are at least one but at most m symbols b after each a.

The above explication shows that all non-empty generated words are in the form $(ab^n)^m$, $1 \leq n \leq m$. It follows from the construction that all words of this form and the empty word, λ, too, are in the generated language, which is thus indeed:

$$L(\Sigma) = \{ \ (ab^n)^m \mid 1 \leq n \leq m \ \} \cup \{\lambda\}.$$

This example shows that even a very simple extended tabled simple eco-grammar system with only one agent is able to produce a quite complicated language, namely a language which is not an ET0L one (see (Rozenberg and Salomaa 1980)).

We show that these systems can generate all recursively enumerable languages. This result is a direct consequence of the following lemma.

Lemma 5.11

Let L be a language in $\mathcal{RC}^{\lambda}_{ac}$. Then there exists an extended tabled simple eco-grammar system Σ with one agent, such that $L(\Sigma) = L$.

PROOF.

Let L be a language in $\mathcal{RC}^{\lambda}_{ac}$. By Lemma 5.5 we can suppose that $L = L(G)$ with a random-context grammar with complete checking $G = (N, T, S, P)$. By Lemma 5.6 we can suppose that the rules in P have the form $(C \to \alpha, Q, R)$, with $C \in N$, $\alpha \in (N \cup T)^*$, $\mathrm{card}(Q) \leq 1$, and $\mathrm{card}(R) \leq 1$.

Moreover, by the note at the end of the proof of Lemma 5.6 we can assume that there are no rules with $Q = R$, $R = \{C\}$ or $Q = \{C\}$.

We denote by r the number of rules in P and by V the set $N \cup T$. The rules of P are enumerated as $p_i = (C_i \to \alpha_i, Q_i, R_i)$. We will refer to the components of the ith rule as C_i, α_i, Q_i, and R_i.

Now we will construct an extended tabled simple eco-grammar system $\Sigma = (V_E, T_E, R_1, \omega, \Delta)$, such that $L(\Sigma) = L(G) = L$. Let

$$
\begin{aligned}
V_E = V \quad & \cup \{[X, i] \mid X \in V, 1 \leq i \leq r\} \\
& \cup \{[X', i] \mid X \in V, 1 \leq i \leq r\} \\
& \cup \{K, U, Z, Z'\} \\
& \cup \{[U, i] \mid 1 \leq i \leq r, Q_i = \emptyset\} \\
& \cup \{[U', i] \mid 1 \leq i \leq r\} \\
& \cup \{[\widehat{U}, i] \mid 1 \leq i \leq r, Q_i \neq \emptyset\}, \\
& \text{where } K, U, Z, Z' \notin V,
\end{aligned}
$$

$$
\begin{aligned}
T_E = \quad & \{T_i, T'_i, T''_i \mid 1 \leq i \leq r\}, \text{ where} \\
T_i = \quad & \{X \to [X, i] \mid X \in V\} \\
& \cup \{U \to [U, i] \mid Q_i = \emptyset\} \\
& \cup \{U \to [\widehat{U}, i] \mid Q_i \neq \emptyset\}, \\
& \text{for } 1 \leq i \leq r, \\
T'_i = \quad & \{[X, i] \to [X', i] \mid X \in V, \{X\} \neq R_i\} \\
& \cup \{[B, i] \to K \mid \{B\} = R_i\} \\
& \cup \{Z' \to Z'\} \\
& \cup \{[\widehat{U}, i] \to [U', i] \mid Q_i \neq \emptyset\} \\
& \text{for } 1 \leq i \leq r, \\
T''_i = \quad & \{[X', i] \to X \mid X \in V\} \\
& \cup \{Z' \to Z, Z' \to \lambda, [U', i] \to U, [U', i] \to \lambda\} \\
& \text{for } 1 \leq i \leq r,
\end{aligned}
$$

$$\begin{aligned} R_1 = \quad & \{Z \to Z'\} \\ & \cup \{[U, i] \to [U', i] \mid 1 \leq i \leq r, Q_i = \emptyset\} \\ & \cup \{[A, i] \to [A', i] \mid 1 \leq i \leq r, Q_i = \{A\}\} \\ & \cup \{[C'_i, i] \to \alpha_i \mid 1 \leq i \leq r\}, \\ \omega = \quad & SUZ, \text{ and} \\ \Delta = \quad & T. \end{aligned}$$

Those symbols which are not mentioned above in the tables are rewritten into K; these rules make the tables complete.

We introduce different alphabets according to the rules of P in the following way. These alphabets are versions of V with different indexes: for the ith rule of P we have alphabets $\{[X, i] \mid X \in V\}$ and $\{[X', i] \mid X \in V\}$. Moreover, we have some special additional symbols in the ETEG system in order to coordinate the derivation. These are $\{K, U, Z, Z'\} \cup \{[U', i] \mid 1 \leq i \leq r\} \cup \{[U, i] \mid 1 \leq i \leq r, Q_i = \emptyset\} \cup \{[\widehat{U}, i] \mid 1 \leq i \leq r, Q_i \neq \emptyset\}$. By K the derivation is blocked: if this symbol appears, then the derivation never results in a terminal word. The different versions of symbol U allow the agent to work when $Q_i = \emptyset$.

First we show how a derivation step of G can be simulated by Σ. During the simulation the sentential form has the form wUZ, where the word w corresponds to the sentential form of G, and U and Z coordinate the simulation. Let us suppose that in a derivation step the rule $(C_i \to \alpha_i, Q_i, R_i)$ is used for a sentential form x. The simulation in the ETEG system goes as follows.

In the first step the environment applies table T_i to rewrite xU into $[x, i][U, i]$ or into $[x, i][\widehat{U}, i]$ depending on whether or not $Q_i = \emptyset$; the agent rewrites Z into Z'. This is the only role of Z: it allows the agent to work during the first step of the simulation.

In the second step the agent applies the rule $[U, i] \to [U', i]$ if $Q_i = \emptyset$ or applies the rule $[A, i] \to [A', i]$ if $Q_i = \{A\}$. The environment rewrites the remaining letters by using table T'_i.

In the third simulation step the agent applies its rule corresponding to the rule of G, namely the rule $[C'_i, i] \to \alpha_i$, while the environment rewrites the remaining letters by using table T''_i. During this last step, the environment can delete the special symbols Z' and $[U', i]$, thus allowing the possibility of finishing the derivation if the sentential form would be a terminal word.

Now we have showed that we can simulate the derivation steps of the random-context grammar. It follows from the construction of the simulating ETEG system that the behaviour described above is the only one which can result in a terminal word. The only possibility to start a derivation from a word over $V \cup \{U, Z\}$ is to use one of the tables T_i and the rule $Z \to Z'$ of the agent. If the sentential form contains some forbidding letters from R_i, then the environment blocks the derivation in the next step by introducing a K; if the permitting symbol of the non-empty set Q_i does not appear in the sentential form, the derivation is blocked because the agent cannot work. (It cannot use the other rule $[U, i] \to [U', i]$, because this symbol appears in the sentential form only if $Q_i = \emptyset$.) In the next step the agent has to use the rule $[C'_i, i] \to \alpha_i$ and the environment has to use table T''_i. These three consecutive steps simulate the application of one of the rules of P. ■

Using the fact that $\mathcal{RC}_{ac}^{\lambda} = \mathcal{RE}$ we obtain the following theorem:

Theorem 5.12

Let L be a recursively enumerable language. Then there exists an extended tabled simple eco-grammar system Σ with one agent, such that $L(\Sigma) = L$.

5.3 Weak extended simple eco-grammar systems

In this section another variant of extended simple eco-grammar systems is studied: the weak extended simple eco-grammar system. This variant has the same components as an extended simple eco-grammar system but it works in a different way.

Definition 5.13

A weak extended simple eco-grammar system (a wEEG system, for short) is a construct $\Sigma = (\, V_E, P_E, R_1, \ldots, R_n, \omega, \Delta\,)$*, where all the components are the same as in Definition 2.4.*

In a weak extended simple eco-grammar system $\Sigma = (V_E, P_E, R_1, \ldots, R_n, \omega, \Delta)$ x *directly derives* y *(with* $x, y \in {V_E}^*$*, written as* $x \stackrel{w\text{-}eco}{\Longrightarrow}_{\Sigma} y$*), if*

- $x = x_1 Z_1 x_2 Z_2 \cdots x_k Z_k x_{k+1}$*, with* $Z_i \in V_E$*,* $x_j \in {V_E}^*$*,* $0 \le k \le n$*,* $1 \le i \le k$*, and* $1 \le j \le k+1$*,*
- $y = y_1 w_1 y_2 w_2 \cdots y_k w_k y_{k+1}$*, with* $y_i, w_j \in {V_E}^*$*,* $0 \le k \le n$*,* $1 \le i \le k$*, and* $1 \le j \le k+1$*,*
- *there exists a permutation of some agents, namely* $R_{j_1}, R_{j_2}, \ldots, R_{j_k}$*, such that* $Z_i \to w_i \in R_{j_i}$ *for* $1 \le i \le k$*,*
- $\mathrm{dom}(R_t) \cap \mathrm{alph}(x_1 x_2 \cdots x_{k+1}) = \emptyset$*, for* $1 \le t \le n, t \ne j_i$*,* $1 \le i \le k$ *and*
- $x_1 x_2 \cdots x_{k+1} = y_1 y_2 \cdots y_{k+1} = \lambda$ *or* $x_1 x_2 \cdots x_{k+1} \stackrel{0L}{\Longrightarrow}_E y_1 y_2 \cdots y_{k+1}$*, where* $E = (V_E, P_E, \omega)$ *is the 0L system of the environment.*

We denote by $\overset{\text{w-eco}}{\Longrightarrow}{}^*_\Sigma$ the transitive and reflexive closure of $\overset{\text{w-eco}}{\Longrightarrow}_\Sigma$.

Informally speaking, in a weak extended simple eco-grammar system we choose some agents to perform a common action in the following way: the chosen agents can perform an action together and there is no symbol among the remaining letters where any of the other agents could act. The chosen agents perform their actions, the remaining letters are rewritten by the environment. In that particular case when there is only one agent in the system, this definition implies that the agent has to work if it is able to but if no letter can be rewritten by the agent, then it is the environment itself that continues the derivation.

Definition 5.14

For a weak extended simple eco-grammar system
$\Sigma = (\, V_E, P_E, R_1, \ldots, R_n, \omega, \Delta \,)$ *the generated language is the following:*

$$L(\Sigma) = \{\, v \in \Delta^* |\; \omega \overset{\text{w-eco}}{\Longrightarrow}{}^*_\Sigma v \,\}.$$

In the following we show that wEEG systems can generate all recursively enumerable languages. The result is based on the following lemma.

Lemma 5.15

Let L be a language in $\mathcal{RC}^\lambda_{ac}$. Then there exists a weak extended simple eco-grammar system Σ with one agent, such that $L(\Sigma) = L$.

PROOF.

Let L be a language in $\mathcal{RC}^\lambda_{ac}$. As in Lemma 5.11, we can suppose that L can be generated by a a random-context grammar with complete checking $G = (N, T, S, P)$, where the rules in P have the form $(C \to \alpha, Q, R)$, with $C \in N$, $\alpha \in (N \cup T)^*$, $\text{card}(Q) \leq 1$, and $\text{card}(R) \leq 1$. By the note at the end of the proof of Lemma 5.6, we can assume that there are no rules with $Q = R$, $R = \{C\}$ or $Q = \{C\}$. We will give a weak extended simple eco-grammar system $\Sigma = (V_E, P_E, R_1, \Delta, \omega)$, such that $L(G) = L(\Sigma)$.

We present the definition of this system Σ and explain its functioning. Similar to Lemma 5.11, the notation V stands for $N \cup T$, r denotes the number of rules in P, the rules in P are enumerated as $p_i = (C_i \to \alpha_i, Q_i, R_i)$.

Let

- $V_E = V \cup \{[X, i, j] \mid X \in V, 1 \le i \le r, 1 \le j \le 5\}$
  $\cup \{[Z, i, j], [\widehat{Z}, i, k] \mid 1 \le i \le r, 1 \le j \le 5, 1 \le k \le 4\}$
  $\cup \{[\widehat{C}_i, i, k] \mid 1 \le i \le r, 1 \le k \le 4\}$
  $\cup \{[U, i, \ell] \mid 1 \le i \le r, Q_i = \emptyset, 1 \le \ell \le 2\}$
  $\cup \{K\}$
  $\cup \{[K, i, 3] \mid 1 \le i \le r\}$ where $U, K, Z \notin V$,

- $P_E = \{[X, i-1, 5] \to [X, i, 1] \mid 2 \le i \le r, X \in V \cup \{Z\}\}$
  $\cup \{[X, r, 5] \to [X, 1, 1] \mid X \in V \cup \{Z\}\}$

  $\cup \{[X, i, 1] \to [X, i, 2] \mid 1 \le i \le r, X \in V \cup \{Z, \widehat{Z}\}\}$
  $\cup \{[\widehat{C}_i, i, 1] \to [\widehat{C}_i, i, 2] \mid 1 \le i \le r\}$
  $\cup \{[U, i, 1] \to [U, i, 2] \mid 1 \le i \le r, Q_i = \emptyset\}$

  $\cup \{[X, i, 2] \to [X, i, 3] \mid 1 \le i \le r, X \in V \cup \{Z, \widehat{Z}\}\}$
  $\cup \{[\widehat{C}_i, i, 2] \to [\widehat{C}_i, i, 3] \mid 1 \le i \le r\}$
  $\cup \{[U, i, 2] \to K \mid 1 \le i \le r, Q_i = \emptyset\}$

  $\cup \{[X, i, 3] \to [X, i, 4] \mid 1 \le i \le r, X \in V \cup \{Z, \widehat{Z}\}\}$
  $\cup \{[\widehat{C}_i, i, 3] \to [\widehat{C}_i, i, 4] \mid 1 \le i \le r\}$
  $\cup \{[K, i, 3] \to K \mid 1 \le i \le r\}$

  $\cup \{[X, i, 4] \to [X, i, 5] \mid 1 \le i \le r, X \in V \cup \{Z\}\}$
  $\cup \{[\widehat{Z}, i, 4] \to [Z, i, 5] \mid 1 \le i \le r\}$

$$\cup \{[\widehat{C}_i, i, 4] \to [C_i, i, 5] \mid 1 \le i \le r\}$$
$$\cup \{[X, i, 5] \to X \mid 1 \le i \le r, X \in V \cup \{Z\}\}$$
$$\cup \{X \to K \mid X \in V \cup \{K\}\},$$

- $R_1 = \{[Z, i-1, 5] \to [\widehat{Z}, i, 1] \mid 2 \le i \le r\}$
$$\cup \{[Z, r, 5] \to [\widehat{Z}, 1, 1]\}$$
$$\cup \{[C_i, i-1, 5] \to [\widehat{C}_i, i, 1] \mid 2 \le i \le r, Q_i \ne \emptyset\}$$
$$\cup \{[C_1, r, 5] \to [\widehat{C}_1, 1, 1] \mid Q_1 \ne \emptyset\}$$
$$\cup \{[C_i, i-1, 5] \to [\widehat{C}_i, i, 1][U, i, 1] \mid 2 \le i \le r, Q_i = \emptyset\}$$
$$\cup \{[C_1, r, 5] \to [\widehat{C}_1, 1, 1][U, 1, 1] \mid Q_1 = \emptyset\}$$
$$\cup \{[\widehat{Z}, i, 1] \to [\widehat{Z}, i, 2] \mid 1 \le i \le r\}$$
$$\cup \{[B, i, 1] \to K \mid 1 \le i \le r, \{B\} = R_i \ne \emptyset\}$$
$$\cup \{[A, i, 2] \to [A, i, 3][K, i, 3] \mid 1 \le i \le r, \{A\} = Q_i \ne \emptyset\}$$
$$\cup \{[U, i, 2] \to [K, i, 3] \mid 1 \le i \le r, Q_i = \emptyset\}$$
$$\cup \{[\widehat{Z}, i, 2] \to [\widehat{Z}, i, 3] \mid 1 \le i \le r\}$$
$$\cup \{[K, i, 3] \to \lambda\}$$
$$\cup \{[\widehat{C}_i, i, 3] \to K \mid 1 \le i \le r, \}$$
$$\cup \{[\widehat{C}_i, i, 4] \to [\alpha_i, i, 5] \mid 1 \le i \le r, \alpha_i \ne \lambda\}$$
$$\cup \{[\widehat{C}_i, i, 4] \to \lambda \mid 1 \le i \le r, \alpha_i = \lambda\}$$
$$\cup \{[Z, i, 5] \to \lambda \mid 1 \le i \le r\},$$

- $\Delta = T$, and
- $\omega = [S, r, 5][Z, r, 5]$.

The main point of the simulation is that we simulate the application of the rules in their order from 1 to r, each time either simulating the rule or skipping it. After having simulated or skipped the rth rule we continue with the first one.

We do the simulation of a rule by introducing five different alphabets for each rule of G: for the ith rule we introduce the alphabets $[V, i, j]$ for $1 \leq j \leq 5$. We start the simulation or the skipping of the ith rule with a word over the alphabet $[V, i-1, 5]$, then during the simulation we go through the alphabets $[V, i, j]$ for $1 \leq j \leq 4$, and finish with a word over the alphabet $[V, i, 5]$. Consequently we can finish the whole derivation or we can continue with the next rule.

There are more additional alphabets for coordinating the simulation: the letters $[Z, i, j]$ and $[\widehat{Z}, i, k]$ for $1 \leq i \leq r$, $1 \leq j \leq 5$, and $1 \leq k \leq 4$ make it possible to skip the ith rule of G; the symbols $[\widehat{C}_i, i, k]$ let the agent simulate the ith rule of G; the symbols $[U, i, \ell]$ are introduced only if Q_i, the permitting set is empty and make it possible to deal with this case; the symbols $[K, i, 3]$ ensure that the derivation is blocked if the agent simulates the ith rule of G while the non-empty permitting condition is missing.

In the following, we first show how the application of a rule of G can be simulated and we also show how the application can be skipped. Then we show why the construction of the above wEEG system guarantees that only those derivations result in a terminal word which follow a derivation of the random-context grammar G.

Let us suppose that we want to simulate the application of the first rule of G: $(C_1 \rightarrow \alpha_1, Q_1, R_1)$ (the case of the other rules is similar) and let us first suppose that $Q_1 \neq \emptyset$. Before the simulation the sentential form in Σ is over $\{[W, r, 5] \mid W \in V \cup \{Z\}\}$.

In the first step the agent "decides" whether the current rule (in this case the first rule of G) will be simulated or will be skipped. Let us suppose that the rule is to be simulated. In this case the agent uses the rule $[C_1, r, 5] \to [\widehat{C}_1, 1, 1]$. The other letters are rewritten by the environment, using the rules $\{[X, r, 5] \to [X, 1, 1] \mid X \in V \cup \{Z\}\}$.

In the next step the agent checks whether or not the forbidding context is present in the sentential form. This is done in the following way: the agent introduces a K if $[B, 1, 1]$ is present (where $\{B\} = R_1$ is the forbidding context), while otherwise the agent does not work because $[\widehat{Z}, 1, 1]$ is not present in the sentential form. The environment increases the second index of the symbols from 1 to 2 in this step.

In the third step the agent uses the rule $[A, 1, 2] \to [A, 1, 3][K, 1, 3]$ for $\{A\} = Q_1$; the environment increases the second indexes from 2 to 3 in the other symbols.

In the fourth step the agent deletes $[K, 1, 3]$ while the environment increases the second indexes from 3 to 4. In the fifth and final step the agent applies the rule $[\widehat{C}_1, 1, 4] \to [\alpha_1, 1, 5]$ or the rule $[\widehat{C}_1, 1, 4] \to \lambda$, which correspond to the first rule of G; the environment increases the second indexes. Therefore we obtain a word over $\{[W, 1, 5] \mid W \in V \cup \{Z\}\}$.

If $Q_1 = \emptyset$, that is, when the permitting condition is empty, then the simulation is different. While the environment does the same as in the previous case, the agent applies different rules. The rule the agent uses in the first step is $[C_1, r, 5] \to [\widehat{C}_1, 1, 1][U, 1, 1]$ and thus $[U, 1, 1]$ is introduced. In the third step this symbol is used to introduce $[K, 1, 3]$ and from that point the simulation continues in the same way as described above, that is, when $Q_1 \neq \emptyset$.

Now we show how we can do the skipping of the first rule (the case of the other rules is the same). Let us suppose again that we have a word over the alphabet $\{[W, r, 5] \mid W \in V \cup \{Z\}\}$.

The environment works in the same way as it did in the previous case, the difference is in the behaviour of the agent. In the first step the agent chooses the rule $[Z, r, 5] \to [\widehat{Z}, 1, 1]$, in the next step the rule $[\widehat{Z}, 1, 1] \to [\widehat{Z}, 1, 2]$, and in the third step the rule $[\widehat{Z}, 1, 2] \to [\widehat{Z}, 1, 3]$. In the fourth and in the fifth step the agent no longer has any rule to apply, hence it does not perform any action. By the end of these five steps we have the same word as we had before, apart from the first indexes in the symbols: we have the same word over the alphabet $\{[W, 1, 5] \mid W \in V \cup \{Z\}\}$.

At this point the simulation or the skipping of the second rule can start and can be carried out in the previous manner. We can continue this process until the last rule, the rth one, when we can restart the whole procedure with the first rule again.

In order to finish the derivation, after having finished the simulation of a rule of G the agent chooses the rule in the form of $[Z, i, 5] \to \lambda$ while the environment rewrites the remaining letters according to its rules $[X, i, 5] \to X$. Thus we have seen that $L(G) \subseteq L(\Sigma)$.

In the following we show that the eco-grammar system must follow one of the sequences presented above, or otherwise the derivation would never terminate.

In the first step, when the sentential form is over $[W, i-1, 5]$, the agent can work because either the left-hand side of the current rule of G is present (and thus the agent can rewrite $[C_i, i-1, 5]$) or the symbol $[Z, i-1, 5]$ can be rewritten. (At the end of the proof we explain why we can suppose that Z has not yet disappeared from the sentential form.)

Therefore, in this first step the agent marks a position where it can perform the application of the current rule or it can mark Z. If it marks a position for the current rule, then in the next steps it must check the appearance of the forbidding and the permitting context. The derivation can result in a word not containing letters K only if the check is successful. This is done in the following way: the derivation is blocked by the rule $[B, i, 1] \to K$ if the forbidding context is present, or by the rule $[\widehat{C}_i, i, 3] \to K$ if the non-empty permitting condition is missing. In the last step the agent must apply the rule corresponding to the rule of G.

Thus we have seen that if the agent decides to mark a position for applying the current rule, then it must check whether or not the rule is applicable, and, if the checking is successful, then finally it must simulate it. If the agent chooses the other possibility and marks Z, then in the next two steps he must increase the second index of $[\widehat{Z}, i, j]$ from 1 to 2 and from 2 to 3. In the next two steps the agent cannot work. Hence if the agent chooses to mark $[Z, i, 5]$, then the work of the whole system follows the strategy of skipping the current rule, or otherwise the derivation would be blocked.

As far as the end of the derivation is concerned, the environment has to apply the rules in the form of $[X, i, 5] \to X$ for all the letters in the same derivation step, or otherwise the derivation is blocked in the next step. It can happen that the agent deletes Z before the end of the derivation but this fact does not allow any new word to be generated, so we can safely assume that the deletion of Z happens in the same derivation step as the rewritings $[X, i, 5] \to X$. We have seen the other direction of the inclusion, $L(\Sigma) \subseteq L(G)$, which completes the proof of the lemma. ■

Because $\mathcal{RC}_{ac}^{\lambda} = \mathcal{RE}$, we obtain the following theorem:

Theorem 5.16

Let L be a recursively enumerable language. Then there exists a weak extended simple eco-grammar system Σ with one agent, such that $L(\Sigma) = L$.

Chapter 6

Watson–Crick systems

The idea of using DNA strands for computation is not a new one but intensive investigations started only after the successful experiment of Adleman (see (Adleman 1994)), when he solved an instance of the Hamiltonian path problem by using DNA strands for computation. Although the graph in which he found a Hamiltonian path was small, this experiment turned the interest to the applicability of DNA strands as a support for computing.

Although the ultimate goal, to create a practical molecular computer, is still elusive, a large amount of work has already been done concerning the theoretical foundation of DNA computing. Many theoretical models have been proposed (see (Păun, Rozenberg, and Salomaa 1998)), where universal computing capacity of Turing machines have been achieved.

Universality seems to be a built-in feature of DNA computing. The main question is to determine which are those common crucial properties of the models motivated by the phenomena of formation of double DNA strands, which make these models so powerful. An answer to this question was first given in (Rozenberg and Salomaa 1996): the effectivity of computing by DNA strand originates from two properties, namely from massive parallelism and from the phenomenon of Watson–Crick complementarity. Parallelism drastically reduces time complexity, thus makes computationally intractable problems tractable, while Watson–Crick complementarity gives us a very useful operation, which is provided by nature "for free". In the following we briefly summarise the main characteristics of this latter phenomenon.

DNA (deoxyribonucleic acid) is found in any living organism, it consists of polymer chains, called DNA strands. DNA strands are composed of nucleotides (sometimes called bases): A (adenine), G (guanine), C (cytosine), and T (thymine). A and G are called purines, C and T are called pyrimidines. In nature the strands form the well-known double helix by a bondage of two separate strands. This bonding always takes place by pairwise attraction of the bases: A bonds with T, G bonds with C. This phenomenon is called Watson–Crick complementarity and the pairs (A, T) and (G, C) are called complementary pairs. Watson–Crick complementarity gives us some fundamental information for free: whenever in a double strand a bondage takes place, we know that the bases at the corresponding positions are complementary. Moreover, for a single DNA strand nature gives its complementary pair; therefore if we have one member of a double strand, then we have its complementary pair as well.

This feature was generalised and formalised in (Salomaa 1998), where it was interpreted as a computational paradigm, called the paradigm of complementarity. Suppose that the alphabet of the strings to be computed is DNA-like, that is, a complementary relation is present among the letters: half of the letters are called purines, their complementary letters are called pyrimidines. In such a situation any string induces its complementary string.

The first notion, called a Watson–Crick D0L system, where the notion of complementarity is considered as an operation, was introduced in (Mihalache and Salomaa 1997). A Watson–Crick D0L system consists of a D0L system over a so-called DNA-like alphabet $V = \{a_1, \dots, a_n, \bar{a}_1, \dots, \bar{a}_n\}$, $n \geq 1$, and a mapping ϕ called the trigger for complementarity transition. Letters a_i and $\bar{a}_i$, $1 \leq i \leq n$, are called complementary letters. Following the analogy to the alphabet of DNA, we speak of purines and pyrimidines. Barred letters will be referred to as pyrimidines, and non-barred ones as purines. The mapping ϕ is from the set of strings over the DNA-like alphabet V to $\{0, 1\}$ and has the following property: the ϕ-value of the axiom of the D0L system is 0 and whenever the ϕ-value of a string is 1, then the ϕ-value of its complementary word must be 0. (The complementary word of a string is obtained by replacing each letter in the string with its complementary letter.) The derivation in a Watson–Crick D0L system proceeds as follows: when a new string is computed by applying the homomorphism of the D0L system, then it is checked according to the trigger. If the ϕ-value of the obtained string is 0 (the string is a "good" word), then the derivation continues in the usual manner. If the obtained string is a "bad" one, that is, its ϕ-value is equal to 1, then the string is changed to its complementary word and the derivation continues from this complementary string.

Particularly important variants of Watson–Crick D0L systems are the so-called standard systems. In this case those words which contain more pyrimidines (barred letters) than purines are "bad" words and those containing at least as many purines as pyrimidines are "good" words. In the dissertation we always deal with standard systems. Analogously to standard Watson–Crick D0L systems, we define standard Watson–Crick EDT0L, E0L and eco-grammar systems, where the underlying systems are EDT0L systems, E0L systems or extended simple eco-grammar systems, respectively.

The computational and generative capacity of these systems is among the most important problems of the field. In (Sosik 2001) it was shown that any Turing computable function can be computed by a standard Watson–Crick D0L system. In the dissertation we study standard Watson–Crick systems as generating devices, that is we are interested in their generative capacity.

We prove that any recursively enumerable language can be generated by a standard Watson–Crick EDT0L system, by a standard Watson–Crick E0L system or by a standard Watson–Crick eco-grammar system. Moreover, the standard Watson–Crick EDT0L system and the standard Watson–Crick eco-grammar system can be chosen with bounded size parameters: we can consider EDT0L systems with two tables or EDT0L systems with no limitation on the number of tables but with the number of non-terminals bounded by a constant and we can also consider standard Watson–Crick eco-grammar systems with only one agent.

Although our main goal in this chapter is to study standard Watson–Crick eco-grammar systems and to prove their universal generative power, the first, longer part of this chapter deals with standard Watson–Crick Lindenmayer systems. The reason behind this is the following: on the one hand the constructions used for Lindenmayer systems show why Watson–Crick complementary extends the generative capacity of the underlying generative device and give useful techniques for using Watson–Crick complementarity in formal language theory, while on the other hand the construction used in the case of standard Watson–Crick E0L systems can be used for standard Watson–Crick eco-grammar systems with only a slight modification.

The main part of the results of this chapter is presented in (Csima, Csuhaj-Varjú, and Salomaa 2001).

6.1 Definitions, examples

By a DNA-like alphabet we mean an alphabet of $2n$ symbols, $n \geq 1$, in the following form: $V = \{a_1, \ldots, a_n, \overline{a_1}, \ldots, \overline{a_n}\}$. We say that letters a_i and $\overline{a_i}$ are complementary letters and we call $V_{PUR} = \{a_1, \ldots, a_n\}$ the alphabet of purines of V and $V_{\mathrm{PYR}} = \{\overline{a_1}, \ldots, \overline{a_n}\}$ is said to be the alphabet of pyrimidines of V.

The complementary word of a string z in V^* is obtained by replacing each letter in z with its complementary letter and it is denoted by $\overline{z}$. The complementary word of the empty word is itself. The morphism h_w, which assigns to each word $z \in V^*$ its complementary word, $\overline{z}$, is called the Watson–Crick morphism.

A standard Watson–Crick ET0L system is a construct $\Sigma = (V, T, \omega, \Delta)$ with the following properties. V is a DNA-like alphabet, $T = \{T_1, \ldots, T_k\}$, $k \geq 1$, is a finite set of tables over V, where each table T_i, $1 \leq i \leq k$, is a complete set of context-free rules over V, $\omega \in V^*$ is the axiom (with $|\omega|_{V_{\mathrm{PUR}}} \geq |\omega|_{V_{\mathrm{PYR}}}$), and Δ is a non-empty subset of V, the terminal alphabet of the system.

The derivation step in a standard Watson–Crick ET0L system $\Sigma = (V, T, \omega, \Delta)$ is defined as follows. For two words, $x = x_1 x_2 \cdots x_n$ and $y = y_1 y_2 \cdots y_n$, where $x_i \in V$, $y_i \in V^*$, $1 \leq i \leq n$, we say that x directly derives y in Σ, written as $x \overset{\text{W-0L}}{\Longrightarrow}_{\Sigma} y$, if the following holds: for some j, $1 \leq j \leq k$, $x_i \rightarrow y_i{}' \in T_j$ for every $1 \leq i \leq n$, and furthermore, if $|y_1{}' y_2{}' \cdots y_n{}'|_{V_{\mathrm{PUR}}} \geq |y_1{}' y_2{}' \cdots y_n{}'|_{V_{\mathrm{PYR}}}$, then $y_i = y_i'$ for every $1 \leq i \leq n$ and $y_i = h_w(y_i{}')$ for every $1 \leq i \leq n$ otherwise. In the latter case we say that a complementary transition takes place.

Thus, if in the string obtained by parallel derivation there are not more pyrimidines than purines, then the derivation continues in the usual manner, like in the case of ET0L systems. Otherwise, if the pyrimidines are in a strict majority in the word, then the derivation continues from the complementary word of the generated string.

We denote the transitive and reflexive closure of $\overset{\text{W-0L}}{\Longrightarrow}_\Sigma$ by $\overset{\text{W-0L}}{\Longrightarrow}_\Sigma^{\;*}$. The generated language consists of those words over Δ which can be obtained from the axiom in some derivation steps:

$$L(\Sigma) = \{\, v \in \Delta^* \mid \omega \overset{\text{W-0L}}{\Longrightarrow}_\Sigma^{\;*} v \,\}.$$

If $T = \{P\}$, that is, if there is only one table, then the system is called a standard Watson–Crick E0L system. In this case we use the notation $\Sigma = (V, P, \omega, \Delta)$, for P being the unique table of the system.

If all the tables are deterministic, that is, there is exactly one rule for each letter in each table, then the system is called a standard Watson–Crick EDT0L system.

Now we present an example which illustrates the generative power of standard Watson–Crick Lindenmayer systems and shows how these systems work.

In this example we have a standard Watson–Crick E0L system and we generate the language (the same as in Example 5.10) $L(\Sigma) = \{\ (ab^n)^m \mid 1 \leq n \leq m\ \}$, which is known not to be in ET0L (see (Rozenberg and Salomaa 1980)).

Example 6.1

Let $\Sigma = (\ V, P, \omega, \Delta\ )$ be the following standard Watson–Crick E0L system:

- $V = \{S, S_1, F, E, A, B, X, X_1, X_2, a, b, \overline{S}, \overline{S_1}, \overline{F}, \overline{E}, \overline{A}, \overline{B}, \overline{X}, \overline{X_1}, \overline{X_2}, \overline{a}, \overline{b}\}$,
- $P = \{\ S \to SF\overline{E} \mid \overline{S_1},\ F \to F,\ \overline{E} \to \overline{E},\ S_1 \to \overline{X_1}A,\ \overline{F} \to \overline{X_1X_1}X \mid \overline{X_1}X,\ E \to A,\ \overline{X_1} \to \overline{X_1},\ X \to XX_2,\ A \to A\overline{B},\ \overline{B} \to \overline{B},\ X_2 \to X_2,\ X_1 \to \lambda,\ \overline{X} \to \lambda,\ \overline{X_2} \to \lambda,\ \overline{A} \to a,\ B \to b\ \}$,

- *$\omega = S$, and*
- *$\Delta = \{ a, b \}$.*

The form of the rules for the remaining letters of the alphabet (i.e. the ones not mentioned in P) is not important because these letters never occur in any derivation, but in order to have a complete table we also put the rules $Y \to Y$ into P, for each letter $Y \in V$ not mentioned above.

Now we explain how this standard Watson–Crick E0L system works. First the system uses the rules $S \to SF\overline{E}, F \to F, \overline{E} \to \overline{E}$ and derives a word in the form of $S(F\overline{E})^{m-1}$, where $1 \leq m$. If in the next step the system uses the rule $S \to \overline{S_1}$ instead of the rule $S \to SF\overline{E}$, then there will be more pyrimidines than purines in the sentential form, thus a complementary transition takes place and we get the word $S_1(\overline{F}E)^{m-1}$. Here the system has to use the rules $S_1 \to \overline{X_1}A$, $\overline{F} \to \overline{X_1 X_1} X \mid \overline{X_1} X$, $E \to A$, and by them a word is derived which consists of one sub-word $\overline{X_1}A$, $n-1$ sub-words $\overline{X_1}XA$, and $m-n$ sub-words $\overline{X_1X_1}XA$, where $1 \leq n \leq m$. The number $n-1$ shows how many times the system used the rule $\overline{F} \to \overline{X_1}X$. The order of the sub-words is not important, so we can suppose that the word is in the form of $(\overline{X_1}XA)^{n-1}(\overline{X_1X_1}XA)^{m-n}\overline{X_1}A$. In this word the number of purines (which is $2m-1$) exceeds the number of pyrimidines (which is $2m-n$) by $n-1 \geq 0$.

In the following, the system uses the rules $\overline{X_1} \to \overline{X_1}$, $X \to XX_2$, $A \to A\overline{B}$, $\overline{B} \to \overline{B}$, $X_2 \to X_2$ and performs exactly n steps. In each step $m-1$ new purines and m new pyrimidines are generated, hence by the end of the nth derivation step there are more pyrimidines than purines in the sentential form. Therefore here a complementary transition takes place and we get the word $(X_1\overline{XX_2{}^nA}B^n)^{n-1}(X_1X_1\overline{XX_2{}^nA}B^n)^{m-n}X_1\overline{A}B^n$.

Here the system uses the rules $X_1 \to \lambda, \overline{X} \to \lambda, \overline{X_2} \to \lambda, \overline{A} \to a, B \to b$ and derives the word $(ab^n)^m$, where $1 \leq n \leq m$.

It is clear that all the words in this form can be generated by the system (the values of the parameters m and n depend on the number of steps the system makes at the beginning of the derivation and on the number of positions it uses the rule $\overline{F} \to \overline{X_1}X$). It is also clear that other words cannot be generated by this system, the derivations have to follow the above line.

6.2 The main idea of the proofs

In this section we present the idea on which both main proofs of the chapter are based. First we recall a definition from (Geffert 1988).

Definition 6.2

Let $\Delta = \{a_1, a_2, \ldots, a_\ell\}$ be an alphabet, $1 \leq \ell$. An Extended Post Correspondence (an EPC) is a set

$$P = (\{(u_1, v_1), \ldots, (u_m, v_m)\}, (z_{a_1}, \ldots, z_{a_\ell})),$$

where $1 \leq m$, and $u_i, v_i, z_{a_j} \in \{0,1\}^*$ for $1 \leq i \leq m$ and $1 \leq j \leq \ell$.
The language represented by P in Δ, written as $L(P)$, is the following set of words:

$$L(P) = \{x_1 \cdots x_n \in \Delta^* \mid \exists s_1, s_2, \ldots, s_r \in \{1, 2, \ldots, m\}, \quad such\ \ that$$

$$1 \leq r \quad and \quad u_{s_1}u_{s_2}\cdots u_{s_r} = v_{s_1}v_{s_2}\cdots v_{s_r}z_{x_1}z_{x_2}\cdots z_{x_n}\}$$

We want to show that for any recursively enumerable language L we can construct a standard Watson–Crick EDT0L system, a standard Watson–Crick E0L system, and also a standard Watson–Crick eco-grammar system, such that the generated language of the systems is L. The constructions have the same idea, which is based on the following theorem presented in (Geffert 1988):

Theorem 6.3

For any recursively enumerable language L there exists an Extended Post Correspondence P, such that $L(P) = L$.

We note here that Definition 6.2 remains correct and Theorem 6.3 remains true if we suppose that the words u_i, v_i, and z_{a_j} are over $\{1, 2\}$ instead of $\{0, 1\}$ for $1 \leq i \leq m$ and $1 \leq j \leq \ell$. This modification is important from our point of view because it makes our construction simpler, so we will use this version of the EPC and Theorem 6.3.

We can consider the words $u_{s_1}u_{s_2}\cdots u_{s_r}$ and $v_{s_1}v_{s_2}\cdots v_{s_r}z_{x_1}z_{x_2}\cdots z_{x_n}$ as numbers in base three notation and therefore we can speak about their values. From this point of view, Theorem 6.3 says that a word $w = x_1 \cdots x_n$ is in L if and only if there exist indices $s_1, \ldots, s_r \in \{1, 2, \ldots, m\}$, such that $1 \leq r$ and the values of the two words $u_{s_1}u_{s_2}\cdots u_{s_r}$ and $v_{s_1}v_{s_2}\cdots v_{s_r}z_{x_1}z_{x_2}\cdots z_{x_n}$ are equal.

Using this fact, we will generate words of a recursively enumerable language L in three phases in the following way. By the end of the first phase we generate a sentential form representing the values of the words $u_{s_1}u_{s_2}\cdots u_{s_r}$ and $v_{s_1}v_{s_2}\cdots v_{s_r}$. By the end of the second phase we will have a sentential form which represents the word $w = x_1 \cdots x_n$ and the values of the words $u_{s_1}u_{s_2}\cdots u_{s_r}$ and $v_{s_1}v_{s_2}\cdots v_{s_r}z_{x_1}z_{x_2}\cdots z_{x_n}$.

The construction used in these phases guarantees that all sentential forms representing the values of a pair in the form of $(u_{s_1}u_{s_2}\cdots u_{s_r}, v_{s_1}v_{s_2}\cdots v_{s_r}z_{x_1}z_{x_2} \cdots z_{x_n})$ and the corresponding word $w = x_1 \cdots x_n$ can be generated in this way.

In the third phase we check the condition presented above: the equality of the values of the two words $u_{s_1}u_{s_2}\cdots u_{s_r}$ and $v_{s_1}v_{s_2}\cdots v_{s_r}z_{x_1}z_{x_2}\cdots z_{x_n}$. The corresponding word $w = x_1\cdots x_n$ is accepted if and only if this checking is successful. Therefore only those words can be generated for which there exist indices $(s_1,\ldots,s_r) \in \{1,2,\ldots,m\}$, such that $1 \leq r$ and $u_{s_1}u_{s_2}\cdots u_{s_r} = v_{s_1}v_{s_2}\cdots v_{s_r}z_{x_1}z_{x_2}\cdots z_{x_n}$.

Now we describe how we can represent values of words in the sentential form. Besides the letters from alph(L), we have additional letters in the sentential form: A and B; the number of A's represents the value of the word $u_{s_1}u_{s_2}\cdots u_{s_r}$, the number of B's represents the value of the other word $v_{s_1}\cdots v_{s_r}z_{x_1}z_{x_2}\cdots z_{x_n}$.

The derivation proceeds in the following way. In the first phase we increase the number of A's and B's according to the concatenation of the pairs (u_j, v_j). Hence by the end of this phase the sentential form represents the values of the words $u_{s_1}u_{s_2}\cdots u_{s_r}$ and $v_{s_1}\cdots v_{s_r}$.

In the second phase we put letters x_i into the sentential form and at the same time increase the number of letters B according to the value of the corresponding word z_{x_i}. Thus by the end of this phase the sentential form contains a word $w = x_1\cdots x_n$ and represents the two values of the words $u_{s_1}u_{s_2}\cdots u_{s_r}$ and $v_{s_1}\cdots v_{s_r}z_{x_1}z_{x_2}\cdots z_{x_n}$.

In the third phase we check whether the number of A's and B's are equal. If the answer is positive, then all symbols but the ones representing the word w are deleted and the word $w = x_1\cdots x_n$ is generated. Otherwise the derivation is blocked and never results in a terminal word.

We use this same idea in both main constructions of the chapter, the difference between the two proofs lies in the techniques used for coordinating the derivation.

6.3 Standard Watson–Crick EDT0L systems

Now we present the construction by which we can give for any recursively enumerable language a generating standard Watson–Crick EDT0L system, where the number of non-terminals is bounded by a constant.

Theorem 6.4

Let L be a recursively enumerable language. Then there exists a standard Watson–Crick EDT0L system Σ, such that $L = L(\Sigma)$. Moreover, the number of non-terminals of Σ is bounded by a constant.

PROOF.

Let L be a recursively enumerable language over an alphabet $\Delta = \{a_1, \ldots, a_l\}$. Then, according to Theorem 6.3, there exists an EPC instance $P = (\{(u_1, v_1), \ldots, (u_m, v_m)\}, (z_{a_1}, \ldots, z_{a_l}))$, where $1 \leq m$, and $u_i, v_i, z_{a_j} \in \{1,2\}^*$ for $1 \leq i \leq m$ and $1 \leq j \leq \ell$, such that $L(P) = L$ holds.

Now we define the simulating standard Watson–Crick EDT0L system $\Sigma = (V, T, \omega, \Delta \cup \overline{\Delta})$, and explain its functioning. For the sake of readability, we use a notation for complementary symbols slightly different from the usual one: the complementary symbol of $[X, i]$ is denoted by $[\overline{X}, i]$.

The alphabet V of the system Σ is the following:

$$V = \{[S, 0], [\overline{S}, 0]\} \cup [W, 1] \cup [W, 2] \cup [W, 3] \cup [W, 4] \cup \{K, \overline{K}\} \cup \Delta \cup \overline{\Delta},$$

where $W = \{A, B, \overline{A}, \overline{B}, S, \overline{S}\}$, $\Delta = \{a_j \mid 1 \leq j \leq l\}$, and $\overline{\Delta} = \{\overline{a}_j \mid 1 \leq j \leq \ell\}$. Symbols A and B have the role explained in Section 6.2; symbols S are used to start the derivation and to introduce the letters a_j; the barred letters, the pyrimidines, play role only in the checking phase.

The axiom is $\omega = [S, 0]$.

The set of the tables is

$$\mathcal{T} = \{T_{i,0}, T_{i,1}, \mid 1 \leq i \leq m\} \cup \{T_{k,2}, \mid 1 \leq k \leq \ell\} \cup \{T_2, T_3, T_4, T_5\}.$$

There are two tables corresponding to each pair (u_i, v_i) for $1 \leq i \leq m$ ($T_{i,0}$ and $T_{i,1}$), and there is a table corresponding to each word z_{a_k} for $1 \leq k \leq \ell$ ($T_{k,2}$). Tables T_2, T_3, T_4 and T_5 are used in switching from one phase to another and in finishing the derivation of the terminal words.

The tables contain the following rules and are used in the following way. All the symbols which are not mentioned in a table are rewritten into K, these rules make the tables complete. For $1 \leq i \leq m$, let

$$T_{i,0} = \left\{[S,0] \to [S,1][A,1]^s[B,1]^t\right\}$$

where s and t are the values of u_i and v_i according to the base three notation. (Actually, s and t, as well as p and q introduced below, depend on i. We have disregarded the matter in our notation, to improve readability.) One of these tables is used to start the derivation and to introduce A's and B's into the sentential form according to the corresponding pair (u_i, v_i). To start with, we have to use one of these tables $T_{i,0}$, thus it is ensured that at least one pair (u_i, v_i) will be represented in the sentential form.

For $1 \leq i \leq m$, let

$$T_{i,1} = \left\{[S,1] \to [S,1][A,1]^s[B,1]^t, [A,1] \to [A,1]^{3^p}, [B,1] \to [B,1]^{3^q}\right\}$$

where s and t are the values, and p and q are the lengths of u_i and v_i. Using one of these tables, we can modify the sentential form in such a way that it reflects the act of concatenating the pair (u_i, v_i) at the end of the pair of words obtained so far. By this concatenation, the values t_1, t_2 of the previously obtained words become $3^p \cdot t_1 + s$ and $3^q \cdot t_2 + t$, respectively. Using these tables several times, by the end of the first phase of the derivation we obtain a sentential form representing a pair $(u_{s_1} \cdots u_{s_r}, v_{s_1} \cdots v_{s_r})$ with $r \geq 1$.

Let

$$T_2 = \{[A,1] \to [A,2], [B,1] \to [B,2], [S,1] \to [S,2]\}.$$

This table switches a word with letters from $[W,1]$ to the corresponding word with letters from $[W,2]$.

For $1 \leq k \leq \ell$, let

$$\begin{aligned} T_{k,2} = \quad & \{[S,2] \to a_k[S,2][B,2]^t, [A,2] \to [A,2], [B,2] \to [B,2]^{3^q}\} \\ & \cup\{a_j \to a_j \mid 1 \leq j \leq \ell\}, \end{aligned}$$

where t is the value and q is the length of z_{a_k}. By these tables we can introduce letters a_k into the sentential form and at the same time we can modify the sentential form in such a way that it represents the act of concatenating the pair (λ, z_{a_k}) to a pair $(u_{s_1} \cdots u_{s_r}, v_{s_1} \cdots v_{s_r} z_{x_1} \cdots z_{x_v})$, $1 \leq r$, $0 \leq v$. Hence by the end of the second phase we have a sentential form representing $(u_{s_1} \cdots u_{s_r}, v_{s_1} \cdots v_{s_r} z_{x_1} \cdots z_{x_n})$ and the corresponding word $x_1 \cdots x_n$.

From now on, the derivation is done by table T_3. This consists of the following rules:

$$T_3 = \{[A,2] \to [A,3], [B,2] \to [\overline{B},3], [S,2] \to \lambda\} \cup \{a_j \to a_j\overline{a}_j \mid 1 \leq j \leq \ell\}.$$

By this set of rules we check whether or not the obtained sentential form w satisfies $|w|_{[A,3]} \geq |w|_{[\overline{B},3]}$. If there are at least as many $[A,3]$'s as $[\overline{B},3]$'s in w, then no complementary transition takes place and the derivation can go on. Otherwise in the next derivation step a blocking situation occurs, the system will introduce symbols K. (We mentioned at the beginning of the proof, that all the letters not mentioned here, including $[\overline{A},3]$ and $[B,3]$, are rewritten to K.)

The next table is T_4, with the rules

$$T_4 = \{[A,3] \to [\overline{A},4], [\overline{B},3] \to [B,4]\} \cup \{a_j, \to a_j, \overline{a_j} \to \overline{a_j} \mid 1 \leq j \leq \ell\}.$$

These rules are for checking whether or not the obtained sentential form w satisfies $|w|_{[\overline{A},4]} \leq |w|_{[B,4]}$. We do it in the same way as we did in the previous case: if the sentential form goes through this checking as well, that is, if no complementary transition takes place, then the number of A's and B's are equal in w. Otherwise the system blocks the derivation in the next step.

Then the application of table T_5 follows, with rules

$$T_5 = \{[\overline{A},4] \to \lambda, [B,4] \to \lambda\} \cup \{a_j \to a_j, \overline{a_j} \to \lambda \mid 1 \leq j \leq \ell\}.$$

These rules finish the derivation by deleting all symbols not in Δ, therefore the derivation results in a word over $\mathrm{alph}(L)$.

The construction of the standard Watson–Crick EDT0L system is based on the EPC representing the recursively enumerable language L and it is guaranteed that only those terminal words $w = x_1 \cdots x_n$ can be obtained by Σ for which there exist some pairs (u_{s_i}, v_{s_i}) in the EPC, such that $u_{s_1}u_{s_2}\cdots u_{s_r} = v_{s_1}v_{s_2}\cdots v_{s_r}z_{x_1}z_{x_2}\cdots z_{x_n}$. It is also clear that all these words can be derived, due to the working of the construct presented above. Moreover, from the construction we can also see that any terminal word that can be generated by the system is over Δ, letters from $\overline{\Delta}$ do not occur in a the generated words. Since these words are exactly the words of L, we have proved that $L(\Sigma) = L$. Moreover, we also can observe that the number of non-terminal letters is bounded by a constant, namely 28. ■

We note here that the only place where Watson–Crick complementarity is used is the third phase, where we check the equality of letters A and B.

6.4 Standard Watson–Crick E0L systems

By a similar technique as we used in the previous section, we prove that any recursively enumerable language can be generated by a standard Watson–Crick ET0L system. This will not be a deterministic system, but has only one table, that is, it is a standard Watson–Crick E0L system. We will see from the construction that the non-determinism of the system can be eliminated by splitting the set of rules into two tables, thus we will obtain another representation of recursively enumerable languages by standard Watson–Crick EDT0L systems.

Theorem 6.5

Let L be a recursively enumerable language. Then there exists a standard Watson–Crick E0L system Σ, such that $L = L(\Sigma)$.

PROOF.

The main ideas of the proof are the same as those of the previous proof: for the recursively enumerable language L we take the representation given by Theorem 6.3. Starting from this, we construct a system whose working can be divided into three different phases as explained in Section 6.2.

Because in a standard Watson–Crick E0L system we have only one table, we have to use other techniques to ensure the possibility of introducing any pair (u_i, v_i) in the first phase and any pair (λ, z_{a_j}) in the second one. In order to do this, we have $2m + 1$ sub-alphabets for the first phase, where m is the number of the pairs (u_i, v_i). These alphabets are in the form of $[W, i]$ for $0 \leq i \leq m$ and $[W, i, 1]$ for $1 \leq i \leq m$. They are used in the following way: we can modify the sentential form by introducing the pair (u_i, v_i) into it only during the two steps while the sentential form is rewritten from a word over $[W, i-1]$ into a word $[W, i, 1]$ and from a word over $[W, i, 1]$ into a word over

$[W, i]$. In this two-step period we can modify the sentential form according to the current pair (u_i, v_i) or we can skip this possibility, going on without altering the sentential form. At the end of this process, that is, when the sentential form is over $[W, m]$, we can continue the first phase by rewriting the sentential form into a word over $[W, 0]$ or we can finish this phase and start the second one with a word over $[W, m + 1]$.

In the second phase we use the same technique: we have $2\ell + 1$ alphabets in the form of $[W, m + t + 1]$ for $0 \leq t \leq l$ and $[W, m + t + 1, 1]$ for $1 \leq t \leq \ell$, where ℓ is the number of a_j's. We can put the pair (λ, z_{a_k}) into the sentential form during the two steps from $[W, m + k]$ to $[W, m + k + 1, 1]$ and from $[W, m+k+1, 1]$ to $[W, m+k+1]$ or we can skip this possibility by not altering the sentential form during these two steps. At the end of this process (that is, when the sentential form is over $[W, m+\ell+1]$) we can continue the second phase by the pair (λ, z_{a_1}) or we can finish it by rewriting the sentential form into a word over $[W, m + \ell + 2]$.

The last part of the derivation is analogous to the one used in the previous section: the checking is done through the alphabets $[W, m + \ell + 2]$, $[W, m + \ell + 3]$, $[W, m + \ell + 4]$. Only those words $w = x_1 \cdots x_n$ are generated for which there is a pair $(u_{s_1} u_{s_2} \cdots u_{s_r}, v_{s_1} v_{s_2} \cdots v_{s_r} z_{x_1} z_{x_2} \cdots z_{x_n})$ whose members are equal.

As in the previous proof, for the sake of readability, we use notations slightly different from the usual ones: the complementary symbol of $[X, i]$ is denoted by $[\overline{X}, i]$ and the complementary symbol of $[X, k, 1]$ is denoted by $[\overline{X}, k, 1]$.

Let $\Sigma = (V, P, \omega, \Delta)$ be the standard Watson–Crick E0L system given as follows.

$$V = \cup_{i=0}^{m+l+4} [W, i] \cup_{k=1}^{m+l+2} [W, k, 1] \cup W \cup \{K, \overline{K}\},$$

where m is the number of the pairs (u_i, v_i) in the representation given by Theorem 6.3 and ℓ is the cardinality of alph(L), and $W = \Big\{A, B, \overline{A}, \overline{B}, C, D, \overline{C}, \overline{D}, S_{in}, S_{out}, \overline{S_{in}}, \overline{S_{out}}, S^0_{out}, \overline{S^0_{out}}\Big\} \cup \{a_j, \overline{a_j}, b_j, \overline{b_j}, \mid 1 \leq j \leq \ell\}$. The symbols A, B and a_j for $1 \leq j \leq \ell$ have the same meaning as in the previous proof, the symbols S coordinate the derivation, while the other symbols are necessary for using Watson–Crick complementarity.

The terminal alphabet is $\Delta = \{a_j \mid 1 \leq j \leq \ell\}$, and the axiom is $\omega = [S^0_{out}, 0]$.

Let us note here that it will follow from the construction of Σ that the sentential forms in any derivation are either over

$$\cup_{i=0}^{m+l+4}[W_1, i] \cup_{k=1}^{m+l+2} [W_1, k, 1] \cup W_1 \cup \{K\}$$

or over

$$\cup_{i=0}^{m+l+4}[W_2, i] \cup_{k=1}^{m+l+2} [W_2, k, 1] \cup W_2 \cup \{K\},$$

where

$$W_1 = \Big\{A, B, \overline{C}, \overline{D}, S_{in}, \overline{S^0_{out}}, \overline{S_{out}}\Big\} \cup \{a_j, \overline{b_j}, \mid 1 \leq j \leq \ell\} \text{ and}$$

$$W_2 = \{\overline{A}, \overline{B}, C, D, \overline{S_{in}}, S^0_{out}, S_{out}\} \cup \{\overline{a_j}, b_j \mid 1 \leq j \leq \ell\}.$$

As it will turn out, the construction also indicates that the sentential form contains only one letter of type S, and this one letter is a purine. If it is an S_{in}, then the word is over W_1, otherwise the word is over W_2.

Due to these properties, in spite of the fact that we have only one table in the system, we can use different sets of rules. By changing S_{in} and S_{out} we can decide which set is to be applied.

Now we give the rules of P for each alphabet, in the same order as they are used in a derivation. This means that starting from $[W, 0]$, we give all the rules of P which can be used for rewriting the letters in the current alphabet and a short explanation, too.

First we give the rules used in the first phase. If the sentential form is over $[W, i-1]$ for $1 \leq i \leq m$, then the system has the following rules:

$[S_{in}, i-1] \rightarrow [S_{in}, i, 1] \mid [\overline{S_{out}}, i, 1]$,
$[S_{out}, i-1] \rightarrow [S_{out}, i, 1] \mid [\overline{S_{in}}, i, 1]$,
$[S^0_{out}, i-1] \rightarrow [S^0_{out}, i, 1] \mid [\overline{S_{in}}, i, 1]$, and
$[X, i-1] \rightarrow [X, i, 1]$ for the other letters $X \in W$.

In this step the system decides whether the current pair (u_i, v_i) is to be introduced into the sentential form or not. Having $[S_{in}, i, 1]$ in the sentential form implies that the pair is to be introduced, otherwise the current pair is to be skipped. This decision can be made via Watson–Crick complementarity in the following way. In the sentential form all the purines but the symbol S appear together with a pyrimidine, thus, apart from S, in the sentential form the number of purines and pyrimidines is equal. By rewriting S to $\overline{S}$ the number of pyrimidines becomes greater than the number of purines, thus each letter transforms into its complementary, that is, the sentential form switches from W_1 to W_2, or vice versa. At the beginning of the derivation we have to distinguish the state when no pairs (u_i, v_i) have been introduced yet. During this state the symbol S has a superscript 0, which disappears only when the first pair has been represented in the sentential form. We can start the second phase only when the symbol S does not have the superscript 0.

If the sentential form is over $[W, i, 1]$ for $1 \leq i \leq m$, then the system has the following rules (in the rules below p and q denote the lengths of u_i and v_i, and s and t denote the values of u_i and v_i):

$[S_{in}, i, 1] \to [S_{in}(A\overline{C})^s(B\overline{D})^t, i]$, $[A, i, 1] \to [(A\overline{C})^{3^p-1}A, i]$,
$[B, i, 1] \to [(B\overline{D})^{3^q-1}B, i]$, and $[X, i, 1] \to [X, i]$ for the other letters $X \in W$.

If at the beginning of this step the sentential form is over $[W_1, i, 1]$, then the current pair (u_i, v_i) is introduced by increasing the number of A's and B's according to (u_i, v_i). Otherwise, the system does not do anything but changes the indices of the letters.

If the sentential form is over $[W, m]$, then the system has the following set of rules:

$[S_{in}, m] \to [S_{in}, m+1, 1] \mid [\overline{S_{out}}, m+1, 1]$,
$[S_{out}, m] \to [S_{out}, m+1, 1] \mid [\overline{S_{in}}, m+1, 1]$,
$[S^0_{out}, m] \to [S^0_{out}, m+1, 1]$, and
$[X, m] \to [X, m+1, 1]$ for the other letters $X \in W$.

Here the system decides whether to finish the first phase or to continue it. If the system makes the sentential form to be over $[W_1, m+1, 1]$, then in the next step the system finishes the first phase, otherwise, when the sentential form is over $[W_2, m+1, 1]$, it rewrites the sentential form into a word over $[W, 0]$ and thus the first phase continues.

If the sentential form is over $[W, m+1, 1]$, then the system has the following rules:

$[X, m+1, 1] \to [X, m+1]$ for $X \in W_1$, $[X, m+1, 1] \to [X, 0]$ for $X \in W_2$.

In the second phase we have similar rules and the working of the system is the same as it was in the first phase: first the system decides whether to put the current z_{a_k} and a_k into the sentential form or not, then in the second step it works according to this decision.

If the sentential form is over $[W, m+k]$ for $1 \leq k \leq \ell$, then the system has the following rules:

$[S_{in}, m+k] \rightarrow [S_{in}, m+k+1, 1] \mid [\overline{S_{out}}, m+k+1, 1]$,
$[S_{out}, m+k] \rightarrow [S_{out}, m+k+1, 1] \mid [\overline{S_{in}}, m+k+1, 1]$, and
$[X, m+k] \rightarrow [X, m+k+1, 1]$ for the other letters $X \in W$.

If the sentential form is over $[W, m+k+1, 1]$ for $1 \leq k \leq \ell$, then the system has the following rules (in the rules below q and t denote the length and the value of z_{a_k}):

$[S_{in}, m+k+1, 1] \rightarrow [a_k\overline{b_k}S_{in}(B\overline{D})^t, m+k+1]$, $[B, m+k+1, 1] \rightarrow [(B\overline{D})^{3^q-1}B,$ $m+k+1]$, and $[X, m+k+1, 1] \rightarrow [X, m+k+1]$ for the other letters $X \in W$.

If the sentential form is over $[W_1, m+k+1]$, then the number of B's is increased according to z_{a_k} and a_k is also introduced. Otherwise, the sentential form is not altered, except the indices.

If the sentential form is over $[W, m+\ell+1]$, then the system has the following rules:

$[S_{in}, m+\ell+1] \rightarrow [S_{in}, m+\ell+2, 1] \mid [\overline{S_{out}}, m+\ell+2, 1]$,
$[S_{out}, m+\ell+1] \rightarrow [S_{out}, m+\ell+2, 1] \mid [\overline{S_{in}}, m+\ell+2, 1]$, and
$[X, m+\ell+1] \rightarrow [X, m+\ell+2, 1]$ for the other letters $X \in W$.

Here the system decides whether to finish the second phase or to continue it; this decision is done in the same way as in the first phase.

If the sentential form is over $[W, m+\ell+2, 1]$, then the system has the following rules:

$[X, m+\ell+2, 1] \to [X, m+\ell+2]$ for $X \in W_1$, $[X, m+\ell+2, 1] \to [X, m+1]$ for $X \in W_2$. In the third phase we check whether the two parts of the pair represented in the sentential form are equal or not. It is done in an analogous way as in the construction in Section 6.3. Therefore we give the rules without any further explanation.

If the sentential form is over $[W, m+\ell+2]$, then the system has the following rules:

$[A, m+\ell+2] \to [A, m+\ell+3]$, $[\overline{C}, m+\ell+2] \to \lambda$, $[B, m+\ell+2] \to [\overline{B}, m+\ell+3]$, $[\overline{D}, m+\ell+2] \to \lambda$, $[a_j, m+\ell+2] \to [a_j, m+\ell+3]$, $[\overline{b_j}, m+\ell+2] \to [\overline{b_j}, m+\ell+3]$, for $1 \leq j \leq \ell$, $[S_{in}, m+\ell+2] \to \lambda$ and $[X, m+\ell+2] \to K$ for the other letters $X \in W$.

If the sentential form is over $[W, m+\ell+3]$, then the system has the following rules:

$[A, m+\ell+3] \to [\overline{A}, m+\ell+4]$, $[\overline{B}, m+\ell+3] \to [B, m+\ell+4]$, $[a_j, m+\ell+3] \to [a_j, m+\ell+4]$, $[\overline{b_j}, m+\ell+3] \to [\overline{b_j}, m+\ell+4]$ for $1 \leq j \leq \ell$, and $[X, m+\ell+3] \to K$ for the other letters $X \in W$.

If the sentential form is over $[W, m+\ell+4]$, then the system has the following rules:

$[\overline{A}, m+\ell+4] \to \lambda$, $[B, m+\ell+4] \to \lambda$, $[a_j, m+\ell+4] \to a_j$, $[\overline{b_j}, m+\ell+4] \to \lambda$ for $1 \leq j \leq \ell$, and $[X, m+\ell+4] \to K$ for the other letters $X \in W$.

To make the set P complete, we also add the rules $X \to K$ for $X \in W \cup \{K, \overline{K}\}$.

The construction guarantees that the system Σ generates the language L because of the following reasons:

1. All the words of L can be generated according to the representation given by Theorem 6.3: in the first two phases the pairs (u_i, v_i), the words z_{x_k} and also the symbols x_k are introduced into the sentential form, while in the third phase the checking deletes all non-terminal symbols.
2. Only those words can be generated which have a representation given in Theorem 6.3, because otherwise the system introduces symbols K, and thus the derivation is blocked. ■

We note that, unlike the construction presented in Section 6.3, the above proof uses Watson–Crick complementarity in the first and the second phase as well. Furthermore, non-determinism appears only in these two phases.

As mentioned above, with a slight modification of the system Σ we can give another construction which shows that standard Watson–Crick EDT0L systems can generate all recursively enumerable languages.

Theorem 6.6

Let L be a recursively enumerable language. There exists a standard Watson–Crick EDT0L system Σ with two tables, such that $L = L(\Sigma)$.

PROOF.

We modify the construction of the standard Watson–Crick E0L system given in the previous proof in the following way. Let us note, that the only place, where the system defined in Theorem 6.5 uses non-determinism is at the purine symbols S. Therefore now we construct two tables, and all the rules except the ones rewriting a purine S symbol are put in both tables. The remaining rules, that is, those which rewrite a purine S symbol, are split into two sub-sets, one of which is put into the first table, the other one is put into the second one.

The first sub-set contains the rules rewriting a purine S to a purine; the second sub-set contains the other rules, that is, those switching a purine S to a pyrimidine. Hence, the system can use the first table if it does not want to switch the sentential form into its complementary word, otherwise it uses the second table. This modification does not affect the generated language; the new system works in the same way as the system defined in Theorem 6.5. Therefore for every recursively enumerable language we can give a standard Watson–Crick EDT0L system with only two tables. ■

Let us note here that in this EDT0L system the number of tables is only two, but the number of non-terminals depends on m and ℓ, given by the EPC representation of L.

6.5 Standard Watson–Crick eco-grammar systems

In this part of the chapter we introduce standard Watson–Crick eco-grammar systems and we show that for any recursively enumerable language L we can construct a standard Watson–Crick eco-grammar system generating it. Furthermore, in this standard Watson–Crick eco-grammar system there is only one agent and the set of the environment is deterministic.

Definition 6.7

A standard Watson–Crick eco-grammar system is a construct $\Sigma = (V_E, P_E, R_1, R_2, \ldots, R_n, \omega, \Delta)$ *with the following properties:* V_E *is a DNA-like alphabet,* P_E *is a complete set of CF rules over* V_E*,* $n \geq 1$*,* R_i *is a set of CF rules over*

V_E (for $1 \le i \le n$), which is the set of the ith agent, $\omega \in V^*$ is the axiom (which contains more purines than pyrimidines), and Δ is a subset of V_E.

The derivation goes in the same way as in an extended simple EG system, the only difference is that whenever a word with more pyrimidines than purines would be derived, a complementary transition takes place.

Definition 6.8

Consider a standard Watson–Crick eco-grammar system $\Sigma = (V_E, P_E, R_1, R_2, \ldots, R_n, \omega, \Delta)$. We say that x directly derives y in Σ (with $x, y \in {V_E}^*$, written as $x \overset{W\text{-}eco}{\Longrightarrow}_\Sigma y$), if $x = x_1 A_1 x_2 \cdots A_{n-1} x_n A_n x_{n+1}$, $y = y_1 z_1 y_2 \cdots z_{n-1} y_n z_n y_{n+1}$, with $x_\ell \in V^*$, $A_j \in V$, $y_\ell, z_j \in V^*$ for $1 \le \ell \le n+1$, $1 \le j \le n$, and there is a permutation of the agents, namely $R_{k_1}, \ldots, R_{k_n}$, such that there are rules $A_i \to {z_i}'$ in R_{k_i} for $1 \le i \le n$ and $x_1 x_2 \cdots x_{n+1} = {y_1}'{y_2}' \cdots {y_{n+1}}' = \lambda$ or $x_1 x_2 \cdots x_{n+1} \overset{0L}{\Longrightarrow}_{\overline{\Sigma}} {y_1}'{y_2}' \cdots {y_{n+1}}'$ holds with the E0L system $\overline{\Sigma} = (V_E, P_E, \omega, \Delta)$ (considering it as a non-Watson–Crick system), and if $|y' = {y_1}'{z_1}'{y_2}' \cdots {z_n}'$ ${y_{n+1}}'|_{V_{PUR}} \ge |y' = {y_1}'{z_1}'{y_2}' \cdots {z_n}'{y_{n+1}}'|_{V_{PYR}}$, then $y_\ell = {y_\ell}'$ and $z_j = {z_j}'$ for every $1 \le \ell \le n+1$ and for every $1 \le j \le n$, otherwise $y_\ell = h_w({y_\ell}')$ and $z_j = h_w({z_j}')$ for every $1 \le \ell \le n+1$ and for every $1 \le j \le n$.

In the above definition h_w is the Watson–Crick morphism which assigns to each word its complementary word.

Informally speaking, in a derivation step first each agent chooses one position to apply a rule, then the remaining letters are rewritten by the environment. If after this rewriting the word contains more pyrimidines than purines, then all letters turn into their complementary letter and the derivation continues from this word, otherwise the derivation continues without a complementary transition.

We denote the transitive and reflexive closure of $\overset{\text{W-eco}}{\Longrightarrow}\Sigma$ by $\overset{\text{W-eco}}{\Longrightarrow}\Sigma^*$.

The generated language consists of the words over Δ which can be obtained from the axiom in some derivation steps:

$$L(\Sigma) = \{\, v \in \Delta^* |\, \omega \overset{\text{W-eco}}{\Longrightarrow}{}_{\Sigma}^{*}\, v \,\}.$$

Now we give a standard Watson–Crick eco-grammar system, as an example, and show that although the system contains only one agent and its rules are simple, it is able to generate a non-ET0L language (the same as in Example 5.10 and 6.1).

Example 6.9

Let $\Sigma = (\ V, P, R_1, \omega, \Delta\ )$ *be the following standard Watson–Crick eco-grammar system:*

- $V = \{a, b, K, \overline{a}, \overline{b}, \overline{K}\} \cup [W, 0] \cup [W, 1]$, *where* $W = \{a, \overline{a}, c, \overline{c}, b, \overline{b}, d, \overline{d}, B, \overline{B}, D, \overline{D}, N, \overline{N}, M, \overline{M}, L, \overline{L}, K, \overline{K}, S_1, \overline{S_1}, S_2, \overline{S_2}\}$,

- $P = \{\ [a, 0] \to [a, 0], [\overline{c}, 0] \to [\overline{c}, 0], [B, 0] \to [B, 0], [\overline{D}, 0] \to [\overline{D}, 0],$
$[N, 0] \to [N, 0], [\overline{M}, 0] \to [\overline{M}, 0]\} \ \cup$

  $\{[\overline{a}, 0] \to [a, 1], [c, 0] \to [\overline{c}, 1], [\overline{B}, 0] \to [B, 1], [D, 0] \to [\overline{D}, 1],$
$[\overline{N}, 0] \to [N, 1], [M, 0] \to [\overline{M}, 1]\} \ \cup$

  $\{[a, 1] \to [a, 1], [\overline{c}, 1] \to [\overline{c}, 1], [B, 1] \to [b, 1][\overline{d}, 1][B, 1], [\overline{D}, 1] \to [\overline{D}, 1],$
$[N, 1] \to [N, 1], [\overline{M}, 1] \to [\overline{M}, 1], [L, 1] \to [L, 1], [b, 1] \to [b, 1],$
$[\overline{d}, 1] \to [\overline{d}, 1][S_1, 1] \to [S_1, 1] \mid [\overline{S_2}, 1]\ \} \cup$

  $\{[\overline{a}, 1] \to a, [c, 1] \to \lambda, [\overline{b}, 1] \to b, [d, 1] \to \lambda, [\overline{B}, 1] \to \lambda,$
$[D, 1] \to \lambda, [\overline{L}, 1] \to \lambda, [M, 1] \to \lambda, [\overline{N}, 1] \to \lambda\} \cup$

  $\{[X, 0] \to K, [X, 1] \to K \mid$ *for the other* $X \in W\} \cup$
$\{a \to K, b \to K, K \to K, \overline{a} \to K, \overline{b} \to K, \overline{K} \to K\ \}$

- $\omega = [S_1, 0]$,
- $\Delta = \{ a, b \}$, *and*
- $R_1 = \{ [S_1, 0] \rightarrow [a, 0][\overline{c}, 0][B, 0][\overline{D}, 0][N, 0][\overline{M}, 0][S_1, 0] \mid$ $[a, 0][\overline{c}, 0][B, 0][\overline{D}, 0][N, 0][\overline{M}, 0][\overline{S_2}, 0], [S_2, 0] \rightarrow [S_1, 1], [N, 1] \rightarrow [L, 1],$ $[S_2, 1] \rightarrow \lambda \}$.

We show that the generated language of this system is

$$L(\Sigma) = \{ (ab^n)^m \mid 1 \leq n \leq m \},$$

which is not an ET0L language (see (Rozenberg and Salomaa 1980)).

The working of the system is divided into three phases. In the first phase the agent uses several times the rule $[S_1, 0] \rightarrow [a, 0][\overline{c}, 0][B, 0][\overline{D}, 0][N, 0]$ $[\overline{M}, 0][S_1, 0]$ and the environment does not alter anything. In this way the sentential form becomes a word in the form $[(a\overline{c}B\overline{D}N\overline{M})^{m-1}S_1, 0]$, with $m \geq 1$. When finally the agent uses the rule $[S_1, 0] \rightarrow [a, 0][\overline{c}, 0][B, 0][\overline{D}, 0][N, 0]$ $[\overline{M}, 0][\overline{S_2}, 0]$, a word with more pyrimidines than purines would be generated, thus a complementary transition takes place and the word $[(\overline{a}c\overline{B}D\overline{N}M)^m S_2, 0]$ results. Then the agent rewrites $[S_2, 0]$ to $[S_1, 1]$ and the environment changes the indices from 0 to 1 and rewrites the letters into their complementary symbols. Hence the sentential form becomes $[(a\overline{c}B\overline{D}N\overline{M})^m S_1, 1]$.

At this point the only rule which is available for the agent is $[N, 1] \rightarrow$ $[L, 1]$. While the agent rewrites the letters N to letters L, the environment writes words $b\overline{d}$ before the letters B. This can happen at most m times, because after this the agent cannot work, the derivation is blocked. At the nth iteration of this step, at most in the mth step, the environment must use the rule $[S_1, 1] \rightarrow [\overline{S_2}, 1]$, which results in a complementary transition and the sentential form becomes a word in the form of $[(\overline{a}c(\overline{b}d)^n\overline{B}D\overline{N}M)^{m-n}$ $(\overline{a}c(\overline{b}d)^n\overline{B}D\overline{L}M)^n S_2, 1]$.

From this point the derivation is finished in one step: the agent deletes the symbol S_2, the environment deletes all letters, except the letters $\overline{a}$ and $\overline{b}$, which are rewritten into terminal form.

In this way all the words in the form of $(ab^n)^m$, with $1 \leq n \leq m$ can be generated and it is clear from the construction that the working explained above is the only possibility to generate a terminal word.

This example shows how a standard Watson–Crick eco-grammar system works and illustrates the power of these systems as well. Let us note that in this example the environment is not deterministic, there are two rules for the symbol $[S_1, 1]$. The construction of the following theorem prove that we could avoid this non-determinism: all recursively enumerable language can be generated by standard Watson–Crick eco-grammar systems with only one agent, and with a deterministic environment.

Theorem 6.10

Let L be a recursively enumerable language. Then there exists a standard Watson–Crick eco-grammar system $\Sigma_1 = (V_1, P_1, R_1, \omega_1, \Delta_1)$, such that $L = L(\Sigma_1)$, and P_1 is deterministic.

PROOF.

For L, we construct Σ_1 by modifying the construction of the standard Watson–Crick E0L system Σ presented in the proof of Theorem 6.5. The alphabet, the terminal alphabet and the axiom of Σ_1 are the same as the ones in Σ. Moreover, P_1, the set of the environment in Σ_1, contains all the rules of P except those which rewrites an S letter being in any form, therefore P_1 is a deterministic set. Instead of these rules, to make the set P_1 complete, we put the rules $[X, i] \to K$ and $[X, k, 1] \to K$ into P_1, where $X \in \{S_{in}, S_{out}, S^0_{out}, \overline{S_{in}}, \overline{S_{out}}, \overline{S^0_{out}}\}$, $0 \leq i \leq m+\ell+4$, and $1 \leq k \leq m+\ell+2$.

The set of the agent, R_1, contains all the S rewriting rules of P where S is in the form of $[X, i]$ or $[X, k, 1]$ with $X \in \{S_{in}, S_{out}, S^0_{out}, \overline{S_{in}}, \overline{S_{out}}, \overline{S^0_{out}}\}$, $0 \leq i \leq m + \ell + 4$, and $1 \leq k \leq m + \ell + 2$. Moreover, in order to guarantee that the agent could work in the last two steps (if the sentential form is not empty), we also put the following rules into R_1: $[A, m{+}\ell{+}3] \to [\overline{A}, m{+}\ell{+}3]$, $[\overline{A}, m + \ell + 4] \to \lambda$.

It is clear from the construction of Σ_1 that it works in the same way as Σ in Theorem 6.5; the only difference is that the choices are made by the agent, not by the E0L set. Thus $L(\Sigma_1) = L$, which completes the proof. ■

Summary

Eco-grammar systems are formal language theoretical models, introduced to formalise the concept of eco-systems consisting of several agents which share a common environment. In this model, the string describing the environment can be changed by sequential rewritings done by the agents and a parallel evolution performed by the environment itself. The complexity of the system can be described by the generated language, which is the set of all strings representing the environment during the derivations.

In this work, we examine how the changes in the derivation mode, in the nature of the agents, and in other parameters affect the generative capacity of simple eco-grammar systems.

First we investigate team behaviour of the agents and we determine the important parameters of teams both in the original case and also in the case when the agents are represented by sets of nature-motivated rules, insertion and replacement.

Next, we study the generative capacity of two variants of simple eco-grammar systems. We prove that all recursively enumerable languages can be generated in both cases.

Finally, we investigate standard Watson–Crick eco-grammar systems, where Watson–Crick complementarity, one of the crucial features of DNA computing, is introduced into the original concept of eco-grammar systems. We show that these systems also have universal generative power.

Összefoglalás

Az öko-grammatikarendszerek olyan formális nyelvi modellek, melyek a közös környezetben élő, több ágensből álló öko-szisztémák fogalmát formalizálják. Ezekben a modellekben az ágensek szekvenciális módon tudják megváltoztatni a környezetet, azaz a rendszer állapotát leíró szót, míg maga a környezet párhuzamosan fejlődik. A rendszer összetettségét a generált nyelv mutatja. Ez azon szavak összessége, melyek valamely levezetés során, valamely pillanatban a rendszert jellemzik.

A dolgozatban azt vizsgáljuk, hogyan változik az egyszerű öko-grammatikarendszerek generatív ereje, ha megváltoztatjuk a rendszer paramétereit, a levezetési módot, vagy az ágenseket jellemző szabályhalmaz természetét.

Először a csoportok szerepét tanulmányozzuk és meghatározzuk a lényeges csoportparamétereket mind az eredeti modellben, mind abban az esetben, amikor az ágenseket természet által motivált szabályokból álló halmazokkal írjuk le.

Ezután az egyszerű öko-grammatikarendszerek két variánsának generatív erejét vizsgáljuk. Bebizonyítjuk, hogy mindkét variáns képes generálni az összes rekurzívan felsorolható nyelvet.

Végül a standard Watson–Crick öko-grammatikarendszereket tanulmányozzuk. Itt a DNS alapú kiszámítás egyik alapfogalmát, a Watson–Crick komplementaritást ötvözzük az öko-grammatikarendszerek koncepciójával. Megmutatjuk, hogy ezek a rendszerek is univerzális erejűek.

Bibliography

Adleman, L. (1994). Molecular computation of solutions to combinatorial problems. *Science 266*, 1021–1024.

Brooks, R. A. (1999). *Cambrian Intelligence: The Early History of the new AI*. MIT Press, Cambridge, Mass.

Csima, J. (1997). Formal language theoretical models of ecosystems (in Hungarian). Master's thesis, Eötvös Loránd University, Budapest, Hungary.

Csima, J. (1998a). On extended simple eco-grammar systems. *Acta Cybernetica 13*(4), 359–373.

Csima, J. (1998b). On hybrid eco-rewriting systems. In A. Kelemenová (Ed.), *Proceedings of the MFCS'98 Satellite Workshop on Grammar Systems*, pp. 215–234. Silesian University, Opava.

Csima, J. (1999). On hybrid eco-rewriting systems. *Grammars 1*(3), 239–253.

Csima, J. (2000). Two remarks on variants of simple eco-grammar systems. *Acta Cybernetica 14*(4), 569–582.

Csima, J., E. Csuhaj-Varjú, and A. Salomaa (2001). Power and size of extended Watson–Crick L systems. Technical Report 424, Turku Centre for Computer Science, Turku. To appear in Theoretical Computer Science.

Csuhaj-Varjú, E. (1995). Eco-grammar systems: Recent results and perspectives. In Gh. Păun (Ed.), *Artificial Life: Grammatical Models*, pp. 79–103. Black Sea University Press, Bucharest.

Csuhaj-Varjú, E. (1996). Generalized eco-grammar systems: a framework for natural language generation. In C. Martin-Vide (Ed.), *XIIth Congress on Natural and Formal Languages*, pp. 13–27. Barcelona.

Csuhaj-Varjú, E. (1999). Colonies: A multi-agent approach to language generation. In A. Kornai (Ed.), *Extended Finite State Models of Language*, Studies in Natural Language Processing, pp. 208–225. Cambridge University Press, Cambridge, Mass.

Csuhaj-Varjú, E. (1999). Grammar systems with bounded resources: on size complexity of grammar systems. In J. Dassow and D. Wotschke (Eds.), *Proceedings of the International Workshop on Descriptional Complexity of Automata, Grammars and Related Structures*, Magdeburg, 1999, pp. 9–22. Otto-von-Guericke University.

Csuhaj-Varjú, E. (2000). Grammar systems: 12 years, 12 problems (short version). In R. Freund and A. Kelemenová (Eds.), *Proceedings of the International Workshop on Grammar Systems 2000*, pp. 77–92. Silesian University, Opava.

Csuhaj-Varjú, E. and J. Dassow (1990). On cooperating/distributed grammar systems. *Journal of Information Processing and Cybernetics 26*(1-2), 49–63. 4th Workshop on Mathematical Aspects of Computer Science, Magdeburg, 1988.

Csuhaj-Varjú, E., J. Dassow, J. Kelemen, and Gh. Păun (1994). *Grammar Systems: A Grammatical Approach to Distribution and Cooperation*, Volume 5 of *Topics in Computer Mathematics*. Gordon and Breach, London.

Csuhaj-Varjú, E. and M. D. Jiménez-López (1998). Cultural eco-grammar systems: A multi-agent system for cultural change. In A. Kelemenová (Ed.), *Proceedings of the MFCS'98 Satellite Workshop on Grammar Systems*, pp. 165–182. Silesian University, Opava.

Csuhaj-Varjú, E., M. D. Jiménez-López, and C. Martín-Vide (1999). Pragmatics and eco-rewriting systems. In Gh. Păun and A. Salomaa (Eds.), *Grammatical Models of Multi-Agent Systems*, pp. 262–283. Gordon and Breach, London.

Csuhaj-Varjú, E. and J. Kelemen (1989). Cooperating grammar systems: A syntactical framework for the blackboard model of problem solving. In I. Plander (Ed.), *Proceedings of the Conference on Artificial Intelligence and Information Control Systems of Robots, 1989*, pp. 121–127. North-Holland, Amsterdam.

Csuhaj-Varjú, E., J. Kelemen, A. Kelemenová, and Gh. Păun (1994). Eco(grammar) systems. A preview. In R. Trappl (Ed.), *Proceedings of the 12th European Meeting on Cybernetics and Systems Research*, pp. 941–948. World Scientific, Singapore.

Csuhaj-Varjú, E., J. Kelemen, A. Kelemenová, and Gh. Păun (1997). Eco-grammar systems: A grammatical framework for studying lifelike interactions. *Artificial Life 3*, 1–28.

Csuhaj-Varjú, E. and A. Kelemenová (1994). Languages of colonies. *Theoretical Computer Science 134*, 119–130.

Csuhaj-Varjú, E. and A. Kelemenová (1998). Team behaviour in eco-grammar systems. *Theoretical Computer Science 209*(1-2), 213–224.

Csuhaj-Varjú, E., Gh. Păun, and A. Salomaa (1995). Conditional tabled eco-grammar systems versus (E)T0L systems. *Journal of Universal Computer Science 5*(1), 252–268.

Csuhaj-Varjú, E. and A. Salomaa (1998). Networks of language processors: Parallel communicating systems. *EATCS Bulettin 66*, 122–138.

Csuhaj-Varjú, E. and A. Salomaa (2000). Networks of Watson–Crick D0L systems. In M. Ito (Ed.), *Proceedings of the 3rd International Colloquium on Words, Languages and Combinatorics*. World Scientific, Singapore.

Dassow, J., J. Kelemen, and Gh. Păun (1993). On parallelism in colonies. *Cybernetics and Systems 24*, 37–49.

Dassow, J. and V. Mihalache (1995). Eco-grammar systems, matrix grammars and E0L systems. In Gh. Păun (Ed.), *Artificial Life: grammatical models*, pp. 210–226. Black Sea University Press, Bucharest.

Dassow, J. and V. Mitrana (1997). Evolutionary grammars: A grammatical model for genome evolution. In R. Hofestädt, T. Lengauer, M. Löffler, and D. Scomburg (Eds.), *Bioinformatics (Proceedings of the German Conference on Bioinformatics)*, Volume 1278 of *Lecture Notes in Computer Science*, pp. 199–209. Springer–Verlag, Berlin.

Dassow, J. and Gh. Păun (1989). *Regulated Rewriting in Formal Language Theory*. Springer–Verlag, Berlin.

Dassow, J., Gh. Păun, and G. Rozenberg (1997). Grammar systems. In A. Salomaa (Ed.), *Handbook of Formal Languages*, Volume 2, Chapter 4, pp. 155–213. Springer–Verlag, Berlin.

Ehrenfeucht, A. and G. Rozenberg (1976). On Θ-determined E0L languages. In A. Lindenmayer and G. Rozenberg (Eds.), *Automata, Languages, Development*. North-Holland, Amsterdam.

Geffert, V. (1988). Context-free-like forms for the phrase-structure grammars. In *Proceedings of MFCS'88*, Volume 324 of *Lecture Notes in Computer Science*, pp. 309–317. Springer–Verlag, Berlin.

Honkala, J. and A. Salomaa (2001). Watson–Crick D0L systems with regular triggers. *Theoretical Computer Science 259*(1-2), 689–698.

Jiménez-López, M. D. (2000). *Grammar Systems: A Formal-Language-Theoretic Framework for Linguistics and Cultural Evolution*. Ph. D. thesis, Universitat Rovira í Virgili, Tarragona, Spain.

Kari, L. (1991). *On Insertion and Deletion in Formal Languages*. Ph. D. thesis, University of Turku, Finland.

Kari, L., A. Mateescu, Gh. Păun, and A. Salomaa (1995). Teams in cooperating grammar systems. *Journal of Experimental and Theoretical AI 7*, 347–359.

Kari, L. and G. Thierrin (1996). Contextual insertions/deletions and computability. *Information and Computation 131*, 47–61.

Kelemen, J. (1998). Colonies – a theory of reactive agents. In A. Kelemenová (Ed.), *Proceedings of the MFCS'98 Satellite Workshop on Grammar Systems*, pp. 7–38. Silesian University, Opava.

Kelemen, J. (1999). Why grammar systems? *Grammars 1*(3), 271–280.

Kelemen, J. and A. Kelemenová (1992). A subsumption architecture for generative symbol systems. In R. Trappl (Ed.), *Cybernetics and Systems Research*, pp. 1529–1536. World Scientific, Singapore.

Langton, C. G. (1989). Artificial life. In C. G. Langton (Ed.), *Artificial Life*, pp. 1–47. Addison–Wesley, Reading, Mass.

Meersman, R. and G. Rozenberg (1978). Cooperating grammar systems. In *Proceedings of MFCS'78*, Volume 64 of *Lecture Notes in Computer Science*, pp. 364–373. Springer–Verlag, Berlin.

Mihalache, V. and A. Salomaa (1997). Lindemayer and DNA: Watson–Crick D0L systems. *EATCS Bulettin 62*, 160–175.

Mihalache, V. and A. Salomaa (2001). Language theoretic aspects of DNA complementarity. *Theoretical Computer Science 250*(1-2), 163–178.

Nii, P. (1989). Blackboard systems. In A. Barr, P. Cohen, and E. Feigenbaum (Eds.), *The Handbook of AI*, Volume 4. Addison–Wesley, Reading, Mass.

Păun, Gh., G. Rozenberg, and A. Salomaa (1998). *DNA Computing – New Computing Paradigms.* EATCS Text Series. Springer–Verlag, Berlin.

Păun, Gh. and L. Sântean (1989). Parallel communicating grammar systems: The regular case. *Annals of the University of Bucharest 38*(2), 55–63. Mathematics-Informatics Series.

Rozenberg, G. and A. Salomaa (1980). *The Mathematical Theory of L-systems.* Academic Press.

Rozenberg, G. and A. Salomaa (1996). Watson–Crick complementarity, universal computations and genetic engineeering. Technical Report 96-28, Leiden University, Leiden.

Salomaa, A. (1973). *Formal Languages.* Academic Press.

Salomaa, A. (1998). Turing, Watson-Crick and Lindenmayer; aspects of DNA complementarity. In C. Calude, J. Casti, and M. Dinneen (Eds.), *Unconventional Models of Computation*, pp. 94–107. Springer–Verlag, Berlin.

Salomaa, A. (1999). Watson–Crick walks and roads on D0L graphs. *Acta Cybernetica 14*(1), 179–192.

Salomaa, A. (2001a). Iterated morphisms with complementarity on the DNA alphabet. Technical Report 390, Turku Centre for Computer Science, Turku. To appear in Lecture Notes in Computer Science.

Salomaa, A. (2001b). Uni-transitional Watson–Crick D0L systems. Technical Report 389, Turku Centre for Computer Science, Turku. To appear in Theoretical Computer Science.

Sosik, P. (1999). On the hierarchy of extended conditional tabled eco-grammar systems. *Grammars 1*(3), 225–238.

Sosik, P. (2001). D0L system+Watson–Crick complementarity= universal computation. In M. Margenstern and Y. Rogozhin (Eds.), *Machines, Computations and Universality. Proceedings of the Third International Conference MCU 2001*, Volume 2055 of *Lecture Notes in Computer Science*, pp. 308–319. Springer–Verlag, Berlin.

ter Beek, M. H. (1999). Simple eco-grammar systems with prescribed teams. In Gh. Păun and A. Salomaa (Eds.), *Grammatical Models of Multi-Agent Systems*, pp. 113–135. Gordon and Breach, London.

Vaszil, Gy. (2000). *Investigations on Parallel Communicating Grammar Systems*. Ph. D. thesis, Eötvös Loránd University, Budapest, Hungary.

List of Figures

1.1 Original version of eco-grammar systems 8

1.2 Simple eco-grammar system 10

3.1 Hierarchy in team derivation mode 42