L systems, Colonies and Eco-grammars

Ernesto Posse
School of Computer Science, McGill University
Montreal, Quebec, Canada
eposse@cs.mcgill.ca

August 23, 2001

1 Introduction

In recent years there has been a considerably growing interest in the fields of Computational Biology (CB) and Artificial Life (Alife). These are concerned with the study of living organisms and life-like behaviour through the construction of models for biological systems and processes.

Although Formal Language Theory has not been traditionally associated with neither CB nor ALife, its relation and influence on these goes as far back as 1968 when Lindenmayer introduced the notion of L system as a mathematical formalism to model growth in biological systems [5]. Since then, the Theory of L Systems [6] has evolved into a rich mathematical framework, and it has been applied in various disciplines such as Computer Graphics, where variations on L grammars have been used to generate realistic images of trees and plants in general ([3], [2]).

Early developments in the area were almost exclusively concerned with the study and modeling of growth. More recent developments deal with modeling systems constituted by several entities or *agents* "living", and possibly interacting, in an *environment*. The notion of a system composed by several agents is not exclusive of neither CB nor Alife. It is relevant to diverse areas such as distributed databases and operating systems, computer networks, robotics, parallel computing, etc. The study of these *multi-agent* systems in a formal framework promises to provide an important insight in their understanding.

This paper intends to give a brief survey of different variants of L systems, and some of the multi-agent formalisms introduced recently. The paper is organized as follows: Section 2 introduces the basic model of L systems, and several of the most significant variants. Section 3 presents the concept of Colonies, which intend to represent elemental multi-agent systems. Section 4 deals with a more complex approach to multi-agent systems, known as Eco-grammars. In Section 5, some theorems about the comparative generative power of these systems are presented.

2 L systems

Informally speaking, in its simplest form, an L system is a grammar system in which rewriting is performed in parallel at each derivation step, that is, all the production rules that can be applied to the word are applied.

Since their inception as models of growth in biological systems, L systems have been regarded as descriptions of dynamic processes rather than static ones. A derivation is seen as evolution or development of an organism. From the mathematical standpoint, this conception has been emphasized by the absence of distinction between *terminal* and *non-terminal* symbols in the grammar's alphabet. There are, however, some L systems in which this differentiation is present. In Section 5 some results about the difference in expressiveness in such systems are summarized.

Rozenberg and Salomaa provide a comprehensive body of work on the basic models and variations of L systems (see [6]). Colonies and a basic model of Eco-grammar systems have been presented in [4]. Conditional Tabled Eco-grammars are introduced in [1].

How does an L system model growth? Why a derivation is seen as a developmental process? The idea is that a word represents the state of the organism, and each symbol stands for a basic part of the system, for instance, a cell. A rule states how a cell evolves. A derivation step of a word represents a time step in which all the cells evolve. If a symbol is associated to a word with two or more symbols, this represents (asexual) reproduction of the cell. If it is associated to the empty word, it represents its dead. If it is associated with itself, it represents the absence of change.

Following, the formal definition of the models is presented.

2.1 0L Systems

Definition 1 *A 0L system is a triple $G = (N, P, s)$ where N is an alphabet, $P \subseteq N \times N^*$ is a set of production rules, and $s \in N^*$ is the axiom.*

As usual, we denote pairs in P with the notation $a \to w$ for $a \in N$ and $w \in N^*$. We say that P is *complete* iff $\forall a \in N.\ \exists w \in N^*.\ a \to w \in P$, that is, there is at least one rule for every symbol in N.

Given two words $w_1, w_2 \in N^*$, we say that w_1 *yields* w_2, written $w_1 \Rightarrow w_2$ (or $w_1 \Rightarrow_G w_2$ if we need to specify the system we are referring to) iff $w_1 = a_1 a_2 \ldots a_n$, $w_2 = v_1 v_2 \ldots v_n$ where $a_i \to v_i \in P$ for $1 \leq i \leq n$. This embodies the fact that we apply one rule to each of the symbols in the word w_1, and we apply to every symbol in one derivation step. As usual, $\Rightarrow^*$ denotes the reflexive and transitive closure of $\Rightarrow$.

The language generated by G, denoted $L(G)$ is $\{w \in N^* \mid s \Rightarrow^* w\}$.

The simplest variations on 0L systems are D0L and P0L systems.

Definition 2 *A D0L system is a 0L system $G = (N, P, s)$ in which $P \in N \to N^*$.*

D0L systems are *deterministic* 0L systems. The definition states that the set of rewriting rules P is a function instead of a relation, hence there is only one rule per symbol.

Definition 3 *A P0L system is a 0L system $G = (N, P, s)$ in which $\forall a \in N.\ a \to \varepsilon \notin P$ where ε is the empty word.*

P0L systems are *propagating* systems. any word always yields a longer word, since there are no erasing rules in P.

A PD0L system is both D0L and P0L. From now on, any variant named XY0L is assumed to satisfy the conditions of X0L and Y0L systems, for any X and Y.

A *0L scheme* is a tuple $G = (N, P)$ that represents a 0L system without an axiom. D0L, P0L and PD0L schemes are defined analogously.

Extended 0L systems (E0L for short) depart from the original notion from 0L systems of grammars without terminal symbols. An E0L system is a system in which the alphabet has been divided into two separate categories: *variables* or *non-terminals*, and *terminals*.

Definition 4 *A E0L system is a tuple $G = (N, T, P, s)$ where $T \subseteq N$ and $G' = (N, P, s)$ is a 0L system.*

Given a E0L system $G = (N, T, P, s)$, the 0L system $G' = (N, P, s)$ is called the *underlying* system. The language of G is $L(G) = L(G') \cap T^*$, that is, the set of strings containing only terminal symbols.

Although at first this definition does not seem like significantly different, it turns out that E0L systems have more generative power than 0L systems.

2.2 Tabled 0L Systems

Tabled 0L (or T0L) systems consist of several sets of production rules, instead of just one. Each set is referred to as a *table* or *component*.

Definition 5 *A T0L system is a tuple $G = (N, \Pi, s)$ where*

$$\Pi = \{P_1,\ P_2,\ \ldots,\ P_n\}$$

and each $G_i = (N, P_i, s)$ is a 0L system, called an underlying system, for $1 \leq i \leq n$ with $n \geq 1$.

The generated language is $L(G) = \{s\} \cup \{w \in N^* \mid \exists w_1, w_2, \ldots, w_m \in N^*,\ m \geq 1,\ s \Rightarrow_{G_{i_1}} w_1 \Rightarrow_{G_{i_2}} \cdots \Rightarrow_{G_{i_m}} w_m = w,\ for\ 1 \leq i_k \leq n,\ 1 \leq k \leq m\}$.

Informally speaking, a T0L language consists of the words for which there is a sequence of steps according to the tables. The order in which the tables are selected is arbitrary, as well as the number of tables used. Notice that although at each derivation step, rewriting is done in parallel for all the symbols in the alphabet, the overall derivation is sequential, in that at each step only one table is used.

T0L systems can be regarded as a first attempt to model a system comprised by several agents, in an environment. A word represents the state of the environment, and the agents are represented by the tables. In this view, agents are purely reactive, and stateless, because they affect the environment's state based solely on its current state, and do not take into consideration any other information such as internal state of an agent.

It is common to use the notation $\Rightarrow_{P_i}$ instead of $\Rightarrow_{G_i}$ when $G_i = (N_i, P_i, S_i)$ is an underlying 0L system in a T0L system.

As E0L systems, T0L systems are also more powerful than simple 0L systems.

We can define variations such as ET0L, PT0L, DT0L, EDPT0L systems and schemes, etc. There are some variations over T0L systems that are worth taking a look at. They are Programmed 0L Systems (denoted (P)T0L, to avoid confusion with propagating T0L systems), Random Context 0L Systems ((RC)T0L), and Conditional T0L Systems. The main theme in these variations is to add some regulation mechanism to T0L systems.

Definition 6 *A (P)T0L system is a tuple $G = (N, \Pi, s)$ where*

$$\Pi = \ \{(l_1, P_1, E_1),\ (l_2, P_2, E_2),\ \ldots,\ (l_n, P_n, E_n)\}$$

and $G' = (N, \{P_1, P_2, \ldots, P_n\}, s)$ is a T0L system, $Lab = \{l_1, l_2, \ldots, l_n\}$ is a set of labels, and each $E_i \subseteq L$, $1 \leq i \leq n$.

Computation in (P)T0L systems proceeds as follows. We define the "yielding" relation in terms of pairs of labels and words: for (l_i, w_1), $(l_j, w_2) \in Lab \times N^*$, we say $(l_i, w_1) \Rightarrow (l_j, w_2)$ iff $w_1 \Rightarrow_{G_i} w_2$ and $l_j \in E_i$, where $G_i = (N, P_i, s)$ is an underlying 0L system. This means that we use the table P_i, associated with the label l_i, to produce the new word w_2, and select non-deterministically a label l_j from the associated set of labels E_i. This provides a mechanism for controling which tables are to be applied, by contrast with T0L systems, in which the tables are chosen arbitrarily.

The language of the (P)T0L system G is the set of words for which there is a derivation:

$$\begin{aligned} L(G) = \{s\} \cup \{w \in N^* \mid \exists r \geq 0.\ \exists l_{i_0}, l_{i_1}, \ldots, l_{i_r} \in Lab.\ \exists w_1, w_2, \ldots, w_r \in N^*. \\ (l_{i_0}, s) \Rightarrow (l_{i_1}, w_1) \Rightarrow \cdots \Rightarrow (l_{i_r}, w_r),\ \textit{and}\ w_r = w\} \end{aligned}$$

Definition 7 *A (RC)T0L system is a tuple $G = (N, \Pi, s)$ where*

$$\Pi = \ \{(N_1, P_1),\ (N_2, P_2),\ \ldots,\ (N_n, P_n)\}$$

and $G' = (N, \{P_1, P_2, \ldots, P_n\}, s)$ is a T0L system, and for $1 \leq i \leq n$ we have $N_i \subseteq N$.

Random context T0L systems or (RC)T0L, provide another mechanism for controling the application of the tables. We state that $w_1 \Rightarrow_G w_2$ iff $w_1 \Rightarrow_{G_i} w_2$ for some $G_i = (N, P_i, s)$ and $w_1 \in N_i^*$. Therefore, a table table can be used if and only if the word is composed only of symbols from its associated set N_i.

The last variation of tabled 0L systems considered here is the Conditional T0L system.

Definition 8 *A Conditional T0L system with k-ary context conditions is a tuple $G = (N, \Pi)$ where N is an alphabet,*

$$\Pi = \{(c_1, d_1, P_1), (c_2, d_2, P_2), \ldots, (c_n, d_n, P_n)\}$$

is a set of triples with $c_i, d_i \in (N^*)^k$, and $P_i \subseteq N \times N^*$ for $1 \leq i \leq n$.

As in the previous cases, each P_i is a set of production rules or table. The k-tuples c_i and d_i are called *permitting condition parameters* and *forbidding condition parameters* respectively. These determine whether P_i can be applied or not, according to some predicate $\pi : N^* \times N^* \to \{true, false\}$. This predicate takes as parameters one of the conditions (c_i or d_i) and a word.

The derivation step is defined as follows: for $w_1, w_2 \in N^*$ we say that $w_1 \Rightarrow_G w_2$ iff

$$\exists i \in \{1, \ldots, n\}.\ w_1 \Rightarrow_{P_i} w_2 \wedge \\ \forall j \in \{1, \ldots, k\}.\ \pi(c_{ij}, w_1) \wedge \neg\pi(d_{ij}, w_1)$$

Therefore, table can be applied only if the predicate is true for all the components of its permitting vector, and false for all of its forbidding vector.

In [1] the following predicates are defined formally:

- $\pi_b(x, y) = true$ iff x is a substring of y.
- $\pi_s(x, y) = true$ iff x appears scattered in y.
- $\pi_p(x, y) = true$ iff a permutation of x appears scattered in y.

2.3 IL Systems

So far, the systems reviewed are context independent in that each letter in a string is rewritten regardless of its neighbors. The next logical step is to introduce the concept of dependency from the surrounding context. This concept is intended to provide a means of expressing local interaction between different parts of a system. The local interaction is captured by considering the m symbols to the left, and n symbols to the right of any given letter in the string. Such a system is called a (m,n)L system, and is defined as follows:

Definition 9 *A (m,n)L system, is a triple $G = (N, P, s)$ where $s \in N^*$, and $P \subseteq (\bigcup_{i=0}^{m} N^i \times N \times \bigcup_{i=0}^{n} N^i) \times N^*$.*

Rules in P are often written as $(w_1, a, w_2) \to w$ or $w_1 a w_2 \to w$. Such a rule states that the symbol a can be replaced by w, but only if its sourrounding (sub)strings are w_1 (to its left) and w_2 (to its right). Formally, this is stated as follows: given $x = a_1 a_2 \ldots a_r$ and $y = w_1 w_2 \ldots w_k$ for some $k \geq 1$, where $a_i \in N$ and $w_i \in N^*$ for $1 \leq i \leq r$, we have $x \Rightarrow y$ iff $(a_{i-m}, a_{i-m+1}, \ldots, a_{i-1}, a_i, a_{i+1}, \ldots, a_{i+n}) \to w_i$. It is assumed that neighbor symbols "outside" the word are ε (the empty word): if $k < 1$ or $k > r$ then $a_k = \varepsilon$.

This last condition, and the caracterization of P as in the definition, take care of the case in which a symbol is "too close" to the beginning or the end of the word, i.e., when $w = w_1 a w_2$, and $|w_1| \leq m$, or $|w_2| \leq n$.

2.4 Multidimensional L Systems

It is usually necessary to consider the structure of a system as part of its state. This structure might refer to some physical distribution of the parts of the system, or to some logical organization. L systems considered in previous sections deal with linear strings, and although this might be enough for many applications, it may be restrictive for others that require the treatment of a richer structure. Multidimensional L systems address this issue.

Although these are not reviewed here, it is worth to mention them. Some of them consist of *graph* grammars. These systems have rules that associate nodes to graphs, but also need rules that connect the graphs that have replaced the nodes. Other systems perform transformations on maps; they have rules on how to divide "cells" or "regions".

3 Colonies

An approach that resembles T0L systems, but differs in some aspects, is that of *colonies.* In some respects, the concept of a colony is simpler; yet, it has some subtleties not present in T0L grammars. A colony is not an L system, so in its basic model, there is no parallel rewriting, but some of its variants introduce concurrency at the level of components. In these variants more than one table is applied at a single derivation step whereas in a T0L system only one table is used.

Definition 10 *A colony is a triple $G = (N, T, \Gamma, S)$ where*

$$\Gamma = \{G_1, G_2, \ldots, G_n\}$$

is a set of finite regular grammars $G_i = (N_i, T_i, P_i, S_i)$ for $1 \leq i \leq n$, $N = \bigcup_{i=1}^{n}(N_i \cup T_i)$, and $T \subseteq N$, and the axiom $S \in N$.

Each underlying gramar G_i, is compposed, as usual of an alphabet N_i, a set of terminals T_i, a set of production rules P_i, and a start symbol, or axiom $S_i \in N_i$.

At first it appears to be a subtle variation of ET0L systems, since we distinguish between terminals and non-terminals, but small differences give rise to a considerably different model, introducing some issues not present in T0L grammars.

N stands for the alphabet of the system, and T for the subset of terminals. Note that a symbol that is terminal in one of the underlying grammars, might occur as nonterminal in another, or viceversa. Another notorious aspect is that the underlying grammars generate only finite regular languages, a restriction not present in ET0L systems.

Computation in a colony is defined as follows: for two words $u, v \in N^*$ we write $u \stackrel{b}{\Rightarrow} v$ iff for some i such that $1 \leq i \leq n$ we have $u = w_1 S_i w_2,\ v = w_1 w w_2$, and $S_i \Rightarrow^*_{G_i} w$. Thus, the language generated is $L(G) = \{w \in N^* | S \stackrel{b}{\Rightarrow}^* w\} \cap T^*$.

In this first notion of colony, only one ocurrence of a symbol is replaced. A sensible variation is to allow the rewriting of multiple ocurrences of the symbol in one step. We define this for $u, v \in N^*$ as:

$$\begin{aligned} u \stackrel{t}{\Rightarrow} v \text{ iff } & u = u_1 S_i u_2 S_i u_3 \ldots u_k S_i u_{k+1}, \\ & v = u_1 w_1 u_2 w_2 u_3 \ldots u_k w_k u_{k+1}, \\ & u_1 u_2 u_3 \ldots u_{k+1} \in (N - \{S_i\})^*, \\ & \text{for each } j \text{ such that } 1 \leq j \leq k, \\ & \text{there is some } i \text{ such that } 1 \leq i \leq n, \text{ and } S_i \Rightarrow_{G_i} w_j \end{aligned}$$

Yet, a third model of sequential rewriting, intermediate between the two previous (*basic* step $\stackrel{b}{\Rightarrow}$ and *terminal* step $\stackrel{t}{\Rightarrow}$), is to allow a specific number $r \geq 1$ of components (production rule sets, or tables) to be applied:

$$\begin{aligned} u \stackrel{=r}{\Rightarrow} v \text{ iff } & u = u_1 S_i u_2 S_i u_3 \ldots u_r S_i u_{r+1}, \\ & v = u_1 w_1 u_2 w_2 u_3 \ldots u_r w_r u_{r+1}, \\ & u_1 u_2 u_3 \ldots u_{r+1} \in (N - \{S_i\})^*, \\ & \text{for each } j \text{ such that } 1 \leq j \leq r, \\ & \text{there is some } i \text{ such that } 1 \leq i \leq n, \text{ and } S_i \Rightarrow_{G_i} w_j \end{aligned}$$

Some alternatives include "$\leq r$" and "$\geq r$": $u \stackrel{\leq r}{\Rightarrow} w$ iff $\exists r' \leq r.\ u \stackrel{=r'}{\Rightarrow} w$, and similarly $u \stackrel{\geq r}{\Rightarrow} w$ iff $\exists r' \geq r.\ u \stackrel{=r'}{\Rightarrow} w$.

As mentioned before, having separate alphabets for each of the components introduces some complex issues when dealing with shared symbols. Admitting concurrency means that all the tables that can be applied are used. The problem arises when the same non-terminal symbol S_i is in more than one grammar. The solutions to this state that if S_i is shared by $k \geq 2$ grammars and there are k ocurrences of S_i in the word, then each of the ocurrences is rewritten according to each of the tables, i.e. all the tables are used. In case that there are less ocurrences of S_i in the word, then there must be some tie-breaking mechanism. There have been defined two alternatives: *stringly competitive parallel* derivation (denoted by $\stackrel{sp}{\Rightarrow}$) and *weakly competitive parallel* derivation (denoted $\stackrel{wp}{\Rightarrow}$). In the first alternative, only one of the conflicting grammars, chosen nondeterministically, is applied, and rewrites at most one instance of its start symbol (non-conflicting grammars are applied). In the second alternative, if there are $m < k$ ocurrences of S_i, then m grammars, chosen nondeterministically, are applied, and, as in the first alternative, each of them rewrites at most one occurence of the symbol. A formal definition of $\stackrel{sp}{\Rightarrow}$ and $\stackrel{wp}{\Rightarrow}$ can be found in [4]. If all the grammars in the colony have different start symbols, the two kinds of steps are the same, and are denoted by $\stackrel{p}{\Rightarrow}$.

Other variants of the concept of colony depend on how the set of terminals of the overall system is defined. The alternatives defined in [4] are:

- *arb* iff $T \subseteq \bigcup_{i=1}^{n} T_i$
- *one* iff $\exists i.\ 1 \leq i \leq n \ and \ T = T_i$
- *ex* iff $T = \bigcup_{i=1}^{n} T_i$
- *all* iff $T = \bigcap_{i=1}^{n} T_i$
- *dist* iff $T = (\bigcup_{i=1}^{n} T_i) - (\bigcap_{i=1}^{n} N_i)$

Based on these parameters, and the different derivation steps, we define the language of a colony as:

$$L_x^f(G) = \{w \in N^* | S \stackrel{x}{\Rightarrow}^* w, \ w \in T^*, \ and \ f\}$$

where $x \in \{b, t, = r, \leq r, \geq r, sp, wp, p\}$ and $f \in \{arb, one, ex, all, dist\}$.

4 Eco-grammars

XT0L systems and colonies are conceived as models of multi-agent systems, where each production-rule set represents an agent of the system, and derivation is seen as evolution of the state of the overall system. Under this conception only one string is used to describe the state of the system, but there is no notion of internal state for each agent.

Eco-grammar systems (EG) have been introduced in order to provide a more complete picture of complex interaction. Under this view, agents have their own state, and their evolution depends on the environment which in turn depends on the state of all agents.

4.1 A model for families of agents with state

In the first model of EGs discussed, we consider each table associated to a family of agents instead of only one agent. The system is composed of an environment, and a set of grammars, each representing a particular type of agent. Several agents can share the same rule set even though each of them has a separate state.

Definition 11 *An eco(grammar) system is a pair* $G = (E, \Gamma)$ *where* $E = (N_E, P_E)$ *is a 0L scheme, and* $\Gamma = \{G_1, G_2, \ldots, G_n\}$ *is a set of agent schemes* $G_i = (N, P_i, \phi_i, R_i, \psi_i)$ *in which:*

- N *is an alphabet common to all agent types* G_i*, possibly including a special symbol* $\Box$
- $P_i \subseteq (N - \{\Box\}) \times N^*$

- $\phi_i : N_E^* \to 2^{P_i}$
- $R_i \subseteq N_E^* \times N_E^*$
- $\psi_i : N^* \to 2^{R_i}$

In this setting, E represents the environment, which has its own dynamics, and Γ represents the set of families of agents. Each agent family, is represented by a scheme, consisting of some production rules P_i that represent the rules governing the internal state, a set of pure rewriting rules R_i, that represent the influence of the agent over the environment, a function ϕ_i that selects the rules of P_i to be used, thus representing the feedback from the environment to the agent, and a function ψ_i, that regulates which rules in R_i are applied to the environment. The rules P_i are referred to as "evolution" rules, and the rules R_i are called "action" rules.

A snapshot of the state of the system consists of a word for the state of the environment, and a word for each of the agents. We call the overall state a *configuration*. A configuration for G is a tuple $(w_E, W_1, W_2, \dots, W_n)$ where $w_E \in N_E^*$ is the state of the environment, and each $W_i = \{'w_{i1}, w_{i2}, \dots, w_{ik_i}\}'$ is a multiset of words (there might be multiple ocurrences of a word) representing the states of the k_i agents of the i-th family. (We denote a multiset by $\{'\dots\}'$.) The signature of a configuration is $N_E^* \times (2^{N^*})^n$.

Note that P_E, and each P_i are 0L rule-sets, hence their application is in parallel for all the symbols in the respective words, whereas the rules R_i are not 0L, thus realize sequential rewritting.

The special symbol $\Box$ plays a role of separator. When it appears in the right-hand side of an evolution rule, it represents the "birth" of a new agent. This is described formally below. For this symbol, we assume that

$$\forall w.\ w\Box\varepsilon = \varepsilon\Box w = w$$

Before defining formally the derivation step for configurations, it is useful to define the following function, that performs the creation of new agents:

$$s(\varepsilon) = \emptyset$$
$$s(w_1\Box w_2\Box\dots\Box w_r) = \{'w_1, w_2, \dots, w_r\}'$$

A derivation is defined for two configurations $c = (w_E, W_1, W_2, \dots, W_n)$, $c' = (w'_E, W'_1, W'_2, \dots, W'_n)$ where the families states are $W_i = \{'w_{i1}, w_{i2}, \dots, w_{ik_i}\}'$ and $W'_i = \{'w'_{i1}, w'_{i2}, \dots, w'_{ik'_i}\}'$ for $1 \le i \le n$. Let $m = \sum_{i=1}^n k_i$ the total number of agents in the system. We say that $c \Rightarrow_G c'$ iff:

- $w_{ij} \Rightarrow_{\phi_i(w_E)} w'_{ij}$ for $1 \le i \le n, 1 \le j \le k_i$.
- $W'_i = \bigcup_{j=1}^{k_i} s(w'_{ij})$ Note: this union preserves multiple ocurrences.
- $w_E = z_1x_1z_2x_2\dots z_mx_mz_{m+1}$, $w'_E = z'_1x'_1z'_2x'_2\dots z'_mx'_mz'_{m+1}$, where for $1 \le r \le m+1$ we have $z_r \Rightarrow_{P_E} z'_r$, and $x_r \to_{\psi_i(w_{ij})} x'_r$. Note: for each w_{ij} one and only one rule in $\psi_i(w_{ij})$ is used to replace x_r by x'_r.

The first condition, says that the agent state w_{ij} is rewritten according to a subset of rules from its family. This subset is selected by the function ϕ_i and depends on the current environment's state. The application is in parallel since these are 0L rules.

The second condition determines the sets of each family, so in case the state of some agent has a $\Box$ symbol, a new word is added to the family for the new agent, or eliminated, if one word becomes ε.

The third condition determines how the environment is updated. For each agent, one rule from the action rules of its family is applied to one symbol in w_E. The remaining symbols are rewritten, in parallel, according to the 0L rules in P_E.

Since a derivation is defined for configurations, ands not simply for strings, the language of the system is a set of configurations, not words. The language of the EG G with initial configuration c_0 is $L(G, c_0) = \{c \in N_E^* \times (2^{N^*})^n | c_0 \Rightarrow_G c\}$.

In order to compare this models to the rest, we define the language of its environment as $L_E(G, c_0) = \{w_E \in N_E^* \mid (w_E, W_1, W_2, \ldots, W_n) \in L(G, c_0)\}$.

4.2 Conditional Tabled Eco-grammars

The last model reviewed in this paper is that of *CTEG*s or *conditional tabled eco-grammar* systems. In this model the regulatory scheme of conditional T0L systems is brought in to a variation of the model in the last section.

The system described in [1] differs from that in [4] in not only introducing the controlling mechanism of conditional tables, but also simplifies it in that we no longer consider families of agents, but only individual agents associated to each table.

Definition 12 *A conditional tabled eco-grammar system is a pair $G = (E, \Gamma)$ whose components are:*

- $E = (N_E, \Pi_E)$*, a conditional T0L scheme with n-ary context conditions:* $\Pi = \{(c_1, d_1, P_1), (c_2, d_2, P_2), \ldots, (c_m, d_m, P_m)\}$ *in which* $c_i, d_i \in N_1^* \times N_2^* \times \cdots \times N_n^*$
- $\Gamma = \{A_1, A_2, \ldots, A_n\}$*, a set of n conditional T0L schemes with 1-ary context conditions:* $A_i = (N_i, \Gamma_i)$ *with* $\Gamma_i = \{(e_{i1}, f_{i1}, P_{i1}), (e_{i2}, f_{i2}, P_{i2}), \ldots, (e_{ir_i}, f_{ir_i}, P_{ir_i})\}$ *where* $e_{ij}, f_{ij} \in N_E^*$ *for* $1 \leq j \leq r_i$ *and for* $1 \leq i \leq n$

E represents the environment, and each A_i represents an agent. The approach of *CTEG*s is more sofisticated, in that now an agent is not represented by a single table as in XT0Ls and colonies, but by a system that is itself composed of several tables: a conditional T0L scheme. From the definition, each condition of the environment c_i, d_i has n elements, one for each agent. This is intended to model the dependency of the environment in the state of the agents. The agents themselves cave 1-ary conditions (e_{ij} and f_{ij}) which are checked against the environment's state.

As for EGs, computation is defined in terms of configurations, but the concept here is simpler. A configuration is an $(n+1)$-tuple $\kappa = (w_E, w_1, w_2, \ldots, w_n)$ where $w_E \in N_E^*$ is for the state of the environment, and each $w_i \in N_i^*$ is the state of the ith agent.

Derivation depends on the criteria used as predicate to test the conditions. We consider the predicates π_α where $\alpha \in \{b, s, p\}$ if defined as in section 2.2. We state $\kappa \Rightarrow_\alpha \kappa'$ where $\kappa = (w_E, w_1, w_2, \ldots, w_n)$ and $\kappa' = (w'_E, w'_1, w'_2, \ldots, w'_n)$ iff:

- In E there is a table i that satisfies $w_E \Rightarrow_{P_i} w'_E$ and all agents j in {1, ..., n} satisfy $\pi_\alpha(c_{ij}, w_i) \wedge \neg\pi_\alpha(d_{ij}, w_i)$
- And each agent A_i has a table P_{ij} that satisfies $w_i \Rightarrow_{P_{ij}} w'_i$ also satisfies $\pi_\alpha(e_{ij}, w_E) \wedge \neg\pi_\alpha(f_{ij}, w_E)$

The language of the environment given a particular criteria and an initial configuration κ_0 is $L_\alpha(G, \kappa_0) = \{w_E \in N_E^* \mid \kappa_0 \Rightarrow_\alpha^* (w_E, w_1, w_2, \ldots, w_n)\}$.

5 Relation between families of languages

This section is intended to provide a summary of some of the results obtained in relation to the generative power of each of the systems described. Proofs can be found in [6], and [1].

Let us recapitulate by recalling some of the main characteristics of the systems described. 0L systems perform parallel rewriting of all its rules. E0L systems distinguish between terminals and non-terminals. A T0L system is composed of various components. At each step of a derivation, one of the components is selected non-deterministically, and only this component is used. (P)T0L, (RC)T0L and conditional T0L systems introduce mechanisms that control which components can be applied. IL systems consider the context of the symbols to be replaced. In a b-colony one component is selected non-deterministically too, but only one symbol is rewritten, although it can be rewritten by any word in the language of the component, and not necessarily by a direct derivative. In a t-colony, all the occurrences of the symbol are rewritten. $= r$, $\leq r$, and $\geq r$ colonies represent intermediate definitions. The sp, wp, and p variants of colonies introduce parallelism of components. Eco-grammars have the concept of state for each component as well as the environment. Each component is 0L. The choice of which rules are applied to each agent depend on the environment, and viceversa. In $CTEG$s, each agent and the environment are represented by conditional T0L schemes.

In this section we will use the following notation for the families of languages generated by each of the models:

- $\mathcal{L}(0L)$ for the family of all 0L languages.
- $\mathcal{L}(X0L)$ for the family of all X0L languages, where X is E, D, P, T, (P), (RC), Conditional T, or any combination of these.

- $\mathcal{L}(XT0L_n)$ for the family of all XT0L languages with n components.
- $\mathcal{L}(COL_x^f)$ for the family of all colony languages, where $x \in \{b, t, = r, \leq r, \geq r, sp, wp, p\}$ is the type of derivation step, and $f \in \{arb, one, ex, all, dist\}$ is the terminal alphabet criteria.
- $\mathcal{L}(COL_x^f(n))$ for the family of all colony languages, with n components.
- $\mathcal{L}(IL)$ for the family of all IL languages.
- $\mathcal{L}((m,n)L)$ for the family of all IL languages with m left neighbors and n right neighbors.
- $\mathcal{L}(1L)$ for the family of (0,1)L and (1,0)L languages.
- $\mathcal{L}(2L)$ for the family of all (1,1)L languages.
- $\mathcal{L}(EG)$ for the family of all EG languages languages.
- $\mathcal{L}(EG_n)$ for the family of all EG languages languages, with n components.
- $\mathcal{L}(CTEG_n(I, J; \alpha))$ for the family of all CTEG languages with n components, $\alpha \in \{b, s, p\}$ the predicate's criteria, and with permitting and forbidding contexts of maximum length I and J respectively.

The traditional families of languages are denoted:

- $\mathcal{L}(FIN)$ for the family of finite languages.
- $\mathcal{L}(RE)$ for the family of regular languages.
- $\mathcal{L}(CF)$ for the family of context-free languages.
- $\mathcal{L}(CS)$ for the family of context-sensitive languages.
- $\mathcal{L}(RE)$ for the family of recursively enumerable languages.

5.1 Generative power of L systems

Deterministic systems are a specific case of the general model:

$$\mathcal{L}(DX0L) \subset \mathcal{L}(X0L)$$

This deterministic machines are the least powerful. 0L systems have also very limited generative power. It is easy to find some very simple finite languages that are not generated by any 0L system. 0L languages are not closed under union, concatenation, the cross operator ($\sqcup^+$), or intersection with regular languages.

Selecting some symbols as terminals actually increases the generative power:

$$\mathcal{L}(X0L) \subset \mathcal{L}(EX0L)$$

Having several tables also increase expressiveness. Tables act as if they where controlling mechanisms, by limiting the rules available for particular derivations:

$$\mathcal{L}(0L) \subset \mathcal{L}(T0L)$$
$$\mathcal{L}(E0L) \subset \mathcal{L}(ET0L)$$

E0L systems are more powerful that context-free grammars:

$$\mathcal{L}(CF) \subset \mathcal{L}(E0L)$$

$\mathcal{L}(T0L)$ is not comparable with $\mathcal{L}(FIN)$, $\mathcal{L}(REG)$, $\mathcal{L}(CF)$, or $\mathcal{L}(E0L)$. $\mathcal{L}(0L)$ is not comparable with $\mathcal{L}(FIN)$, $\mathcal{L}(REG)$, or $\mathcal{L}(CF)$.

In many cases adding regulation mechanisms increases the power of the grammar, but that is not always the case:

$$\mathcal{L}(T0L) \subset \mathcal{L}((P)T0L) \subset \mathcal{L}(ET0L)$$

and

$$\mathcal{L}(T0L) \subset \mathcal{L}((RC)T0L)$$

but

$$\mathcal{L}(E(P)T0L) = \mathcal{L}(ET0L) \subset \mathcal{L}(E(RC)T0L)$$

while $\mathcal{L}((P)T0L)$ and $\mathcal{L}((RC)T0L)$ are not comparable, as well as $\mathcal{L}(ET0L)$ and $\mathcal{L}((RC)T0L)$ are not comparable.

ET0L languages are considerably powerful. They are closed under the operations mentioned above: union, concatenation, the cross operator ($\sqcup^+$), or intersection with regular languages.

In the case of IL systems there are some interesting theorems:

$$\begin{aligned} \forall m_1, m_2, n_1, m_2 \geq 1.\ \mathcal{L}((m_1,n_1)L) &\subset \mathcal{L}((m_2,n_2)L)\ iff\ m_1+n_1 < m_2+n_2 \\ \mathcal{L}((m_1,n_1)L) &= \mathcal{L}((m_2,n_2)L)\ iff\ m_1+n_1 = m_2+n_2 \end{aligned}$$

Hence, expressiveness depends on the size of the context used by each rule in the system, but it does not matter if we "shift" some of the neighbors from left to right or right to left. This theorem establishes a hierarchy of IL systems with (0,0)L at the bottom.

Another notorious result is that

$$\mathcal{L}(E1L) = \mathcal{L}(E2L)$$

The same is true for the D, P and PD variants. Using only one neigbor is enough to generate as many languages as using two. However, an even more general result is the following:

$$\mathcal{L}(EIL) = \mathcal{L}(E1L) = \mathcal{L}(RE)\ and$$
$$\mathcal{L}(EPIL) = \mathcal{L}(EP1L) = \mathcal{L}(CS)$$

This means that we can reduce any IL system to one that uses only one neighbor, and we can simulate a Turing machine with it. For propagating systems, we obtain the power of context sensitive grammars.

5.2 Generative power of Colonies

Colonies with terminals according to the "ex" criteria (the terminals of the colony are the union of the terminals of all its components), generate less languages than the rest. The "ex" acceptance method represents the largest set of terminals constructed from the terminals of the parts. Therefore, having more restricted sets increments the generative power. This is akin of EX0L being more powerful than X0L systems in that some restrictions are being added. The rest of the acceptance criterions are equivalent. Colonies with a b-derivation step are less powerfull than those with t-derivations.

$$\begin{aligned}
&x \in \{b, t\}, \ and \ f, f' \in \{arb, one, all, dist\} \\
&\mathcal{L}(COL_x^{ex} \subset \mathcal{L}(COL_x^f) \\
&\mathcal{L}(COL_x^f) = \mathcal{L}(COL_x^{f'}) \\
&\mathcal{L}(COL_b^f) \subset \mathcal{L}(COL_t^f)
\end{aligned}$$

However, b-step colonies are as powerful as colonies with parallelism of components, but only when their start symbols are different (p steps).

$$\mathcal{L}(COL_b) = \mathcal{L}(COL_p) \subset \mathcal{L}(COL_sp)$$

For "$= r$", "$\leq r$" and "$\geq r$" colonies we have:

$$\begin{aligned}
&\mathcal{L}(COL_b) = \mathcal{L}(COL_{=1}) = \mathcal{L}(COL_{\leq k}) = \mathcal{L}(COL_{\geq 1}) \\
&\mathcal{L}(COL_{=r}), \ \mathcal{L}(COL_{=s}), \ \mathcal{L}(COL_{\leq t}), \ \mathcal{L}(COL_{\leq u}) \\
&are \ pairwise \ incomparable \ for \ any \ r, s, t, u \geq 1
\end{aligned}$$

The following relates colonies with other systems:

$$\begin{aligned}
&\mathcal{L}(COL_b^f) = \mathcal{L}(CF) \\
&\mathcal{L}(COL_t^f) = \mathcal{L}(ET0L) \\
&\mathcal{L}(COL_{wp}^f) \subset \mathcal{L}(ET0L)
\end{aligned}$$

Of interest are the comparsion between colonies in relation to their number of components. It turns out that colonies with more tables are more powerful:

$$\mathcal{L}(COL_x^f(n)) \subset \mathcal{L}(COL_x^f(n+1))$$

5.3 Generative power of Eco-grammars

For CTEG's the number of components also increases expressiveness, as well as the size of the permitting and forbidding contexts:

$$\mathcal{L}(CTEG_n(I, J; \alpha)) \subseteq \mathcal{L}(CTEG_{n'}(I', J'; \alpha)) \ when \ n \leq n', \ I \leq I', \ and \ J \leq J'$$

Changing the predicate amongst the ones defined (b, s, p) does not have any effect when the contexts have size 0 or 1:

$$\mathcal{L}(CTEG_n(I, J; \alpha)) = \mathcal{L}(CTEG_n(I, J; \alpha'))$$
$$for\ \alpha, \alpha' \in \{b, s, p\}\ and\ I, J \in \{0, 1\}$$

If we do not consider context at all, the number of agents is irrelevant, and the system is reduced to a basic T0L system:

$$\mathcal{L}(CTEG_\infty(0, 0; \alpha)) = \mathcal{L}(CTEG_1(0, 0; \alpha)) = \mathcal{L}(T0L)$$

Systems with only one agent and arbitrarily large permitting contexts are as powerful as systems with any number of agents, and even with limited contexts can generate all finite languages:

$$\mathcal{L}(CTEG_n(\infty, 0; \alpha)) = \mathcal{L}(CTEG_1(\infty, 0; \alpha))\ for\ \alpha \in \{s, p\}$$
$$\mathcal{L}(FIN) \subset \mathcal{L}(CTEG_1(1, 0; \alpha))\ and$$
$$\mathcal{L}(FIN) \subset \mathcal{L}(CTEG_1(0, 1; \alpha))\ for\ \alpha \in \{b, s, p\}$$

The relation with T0L systems and its variants is diverse.

$$\mathcal{L}((RC)T0L) \subset \mathcal{L}(CTEG_\infty(1, 0; \alpha))$$
$$\mathcal{L}((P)T0L) \subset \mathcal{L}(CTEG_1(1, 0; \alpha))$$
$$\mathcal{L}(ET0L)\ is\ incomparable\ with\ \mathcal{L}(CTEG_\infty(I, J; \alpha))$$
$$if\ I \geq 1,\ J \geq 0,\ and\ \alpha \in \{b, s, p\}\ or$$
$$if\ I \geq 0,\ J \geq 2,\ and\ \alpha \in \{s, p\}$$

A particular case of this last result is

$$\mathcal{L}(ET0L)\ is\ incomparable\ with\ \mathcal{L}(CTEG_\infty(1, 0; \alpha))$$

however it is also known that

$$\mathcal{L}(CTEG_\infty(0, 1; \alpha)) \subset \mathcal{L}(ET0L)$$

6 Conclusion

This review has obviously been superficial, and we left aside many relevant issues such as computational complexity or growth functions (functions that represent the length of a word in terms of the number of derivation steps). [6] deal at some extent with these. We also left out many variants and important results. But some of the most significant were overviewed.

It has been shown that formal language theory can be used describe very complex systems. However, some of the constructions tend to become very complex themselves. While some of the systems presented depart significantly from the simplest models of computation, the benefit resides in that they become closer to the description of complex distributed/concurrent/multi-agent systems. This may result in a deeper understanding of such systems.

References

[1] E. Csuhaj-Varju, G. Păun, and A. Salomaa. Conditional tabled eco-grammar systems versus (E)T0L systems. *Journal of Universal Computer Science*, 1(5), 1995.

[2] O. Deussen and B. Lintermann. Interactive modeling of plants. *IEEE Computer Graphics and Applications*, January/February 1999.

[3] J. D. Foley, A. van Dam, S. K. Feiner, and J. F. Hughes. *Computer Graphics: Principles and Practice*. Addison-Wesley, 1997.

[4] A. Kelemenová and J. Kelemen. From colonies to eco(grammar)systems. In *Results and Trends in Theoretical Computer Science*, volume 812 of *Lecture Notes in Computer Science*, pages 213–231, 1994.

[5] A. Lindenmayer. Mathematical models for cellular interactions in development. *Journal of Theoretical Biology*, 18, 1968.

[6] G. Rozenberg and A. Salomaa. *The Mathematical Theory of L systems*. Academic Press, Inc., 1980.