Chapter 7

Applications and Implementation

Although this book primarily represents a theoretically oriented treatment, most grammars discussed in the previous chapters have quite realistic applications. Indeed, these grammars are useful to every scientific field that formalizes its results by some strings and studies how these strings are produced from one another under some permitting or, in contrast, forbidding conditions. As numerous areas of science formalize and study their results in this way, any description of applications that cover more than one of these areas would be unbearably sketchy, if not impossible. Therefore, we concentrate our attention on a single application area—*microbiology*, which appears of great interest at present. In this intensively investigated scientific field, we give three case studies that make use of L grammars with context conditions (see Chapter 4.2). Section 7.1 presents two case studies of biological organisms whose development is affected by some abnormal conditions, such as some virus infection. From even more practical point of view, Section 7.2 discusses parametric 0L grammars (see [151]), which represent a powerful and elegant implementation tool in the area of biological simulation and modelling today. More specifically, we extend parametric 0L grammars by context conditions and demonstrate their use on models of growing plants.

7.1 Applications

Case Study 1. Consider a cellular organism in which every cell divides itself into two cells during every single step of a healthy development. However, when a virus infects some cells, all the organism stagnates until it is cured again. During the stagnating period, all the cells just reproduce themselves without producing any new cells. To formalize this development by a suitable simple semi-conditional L grammar (see Section 4.2.3), we denote a healthy cell and a virus-infected cell by A and B, respectively, and introduce the simple semi-conditional 0L grammar, $G = (\{A, B\}, P, A)$, where P contains the following productions:

$$\begin{array}{ll} (A \to AA, 0, B), & (B \to B, 0, 0), \\ (A \to A, B, 0), & (B \to A, 0, 0), \\ (A \to B, 0, 0). & \end{array}$$

Figure 7.1 describes G simulating a healthy development while Figure 7.2 gives a development with a stagnating period caused by the virus. □

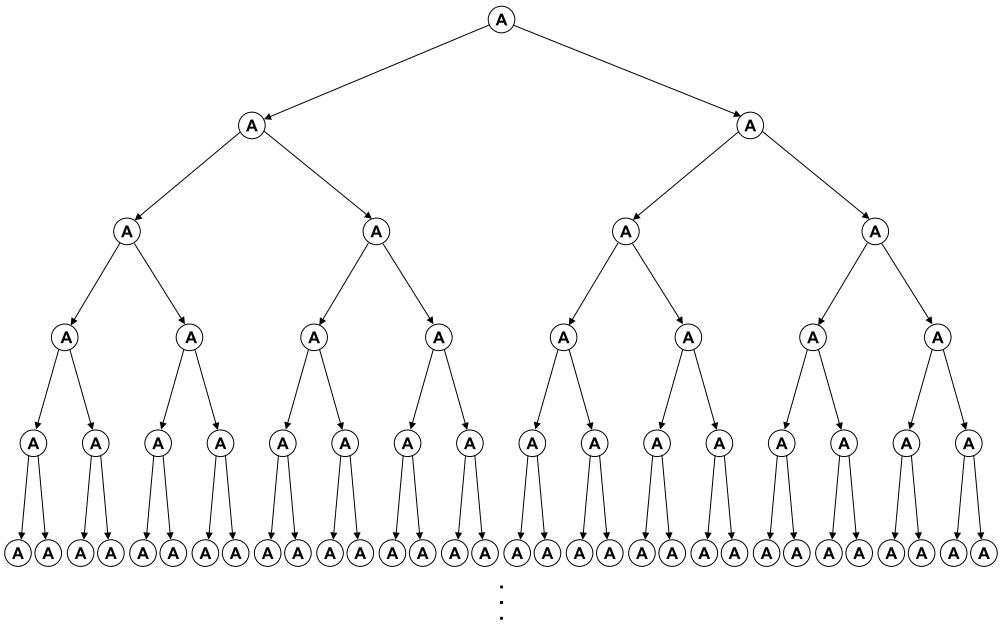

Figure 7.1: Healthy development.

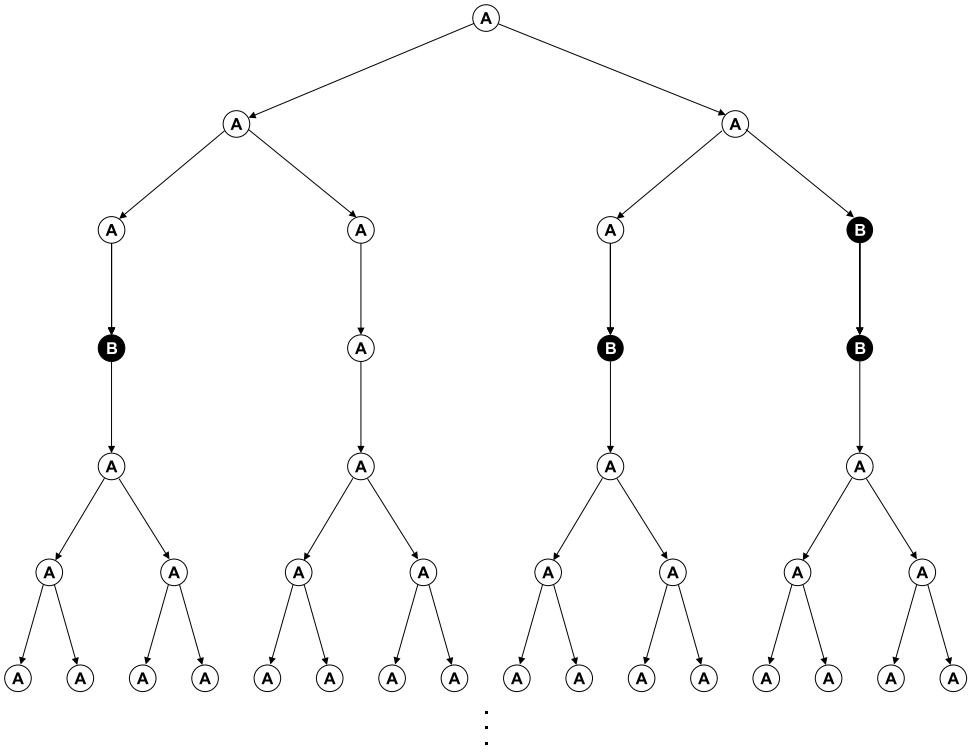

Figure 7.2: Development with a stagnating period.

In the next case study, we reconsider the well known 0L grammar that simulate the developmental stages of a red alga (see [162], [167]). By using context conditions, we modify this system so it describes some unhealthy development of this alga, which leads to its partial death or degeneration.

Case Study 2. Consider an 0L grammar, $G = (V, P, 1)$, where $V = \{1, 2, 3, 4, 5, 6, 7, 8, [,]\}$ and the set of productions P contains

$$\begin{array}{lllll} 1 \to 23, & 2 \to 2, & 3 \to 24, & 4 \to 54, & [\to [, \\ 5 \to 6, & 6 \to 7, & 7 \to 8[1], & 8 \to 8, &] \to]. \end{array}$$

From a *biological viewpoint*, parenthesized expressions represent branches whose position is indicated by 8s. These branches are shown as attached on alternate sides of the branch on which they are born. Figure 7.3 gives a biological interpretation of the developmental stages formally specified by the next derivation, which contain thirteen strings corresponding to stages (a) through (m) in the figure.

$$\begin{array}{lll} 1 & \Rightarrow_G & 23 \\ & \Rightarrow_G & 224 \\ & \Rightarrow_G & 2254 \\ & \Rightarrow_G & 22654 \\ & \Rightarrow_G & 227654 \\ & \Rightarrow_G & 228[1]7654 \\ & \Rightarrow_G & 228[23]8[1]7654 \\ & \Rightarrow_G & 228[224]8[23]8[1]7654 \\ & \Rightarrow_G & 228[2254]8[224]8[23]8[1]7654 \\ & \Rightarrow_G & 228[22654]8[2254]8[224]8[23]8[1]7654 \\ & \Rightarrow_G & 228[227654]8[22654]8[2254]8[224]8[23]8[1]7654 \\ & \Rightarrow_G & 228[228[1]7654]8[227654]8[22654]8[2254]8[224]8[23]8[1]7654. \end{array}$$

Death. Let us assume that the red alga occurs in some unhealthy conditions under which only some of its parts survive while the rest dies. This dying process starts from the newly born, marginal parts of branches, which are too young and weak to survive, and proceeds towards the older parts, which are strong enough to live under these conditions. To be quite specific, all the red alga parts become gradually dead except for the parts denoted by 2s and 8s. This process is specified by the following 0L grammar, G, with forbidding conditions. Let $W = \{a' : \ a \in V\}$. Then, $G = (V \cup W, P, 1)$, where the set of productions, P, contains:

$$\begin{array}{ll} (1 \to 23, W), & (1' \to 2', \{3', 4', 5', 6', 7'\}), \\ (2 \to 2, W), & (2' \to 2', \emptyset), \\ (3 \to 24, W), & (3' \to \varepsilon, \{4', 5', 6', 7'\}), \\ (4 \to 54, W), & (4' \to \varepsilon, \emptyset), \\ (5 \to 6, W), & (5' \to \varepsilon, \{4'\}), \\ (6 \to 7, W), & (6' \to \varepsilon, \{4', 5'\}), \\ (7 \to 8[1], W), & (7' \to \varepsilon, \{4', 5', 6'\}), \\ (8 \to 8, W), & \\ ([\to [, \emptyset), & \\ (] \to], \emptyset), & \end{array}$$

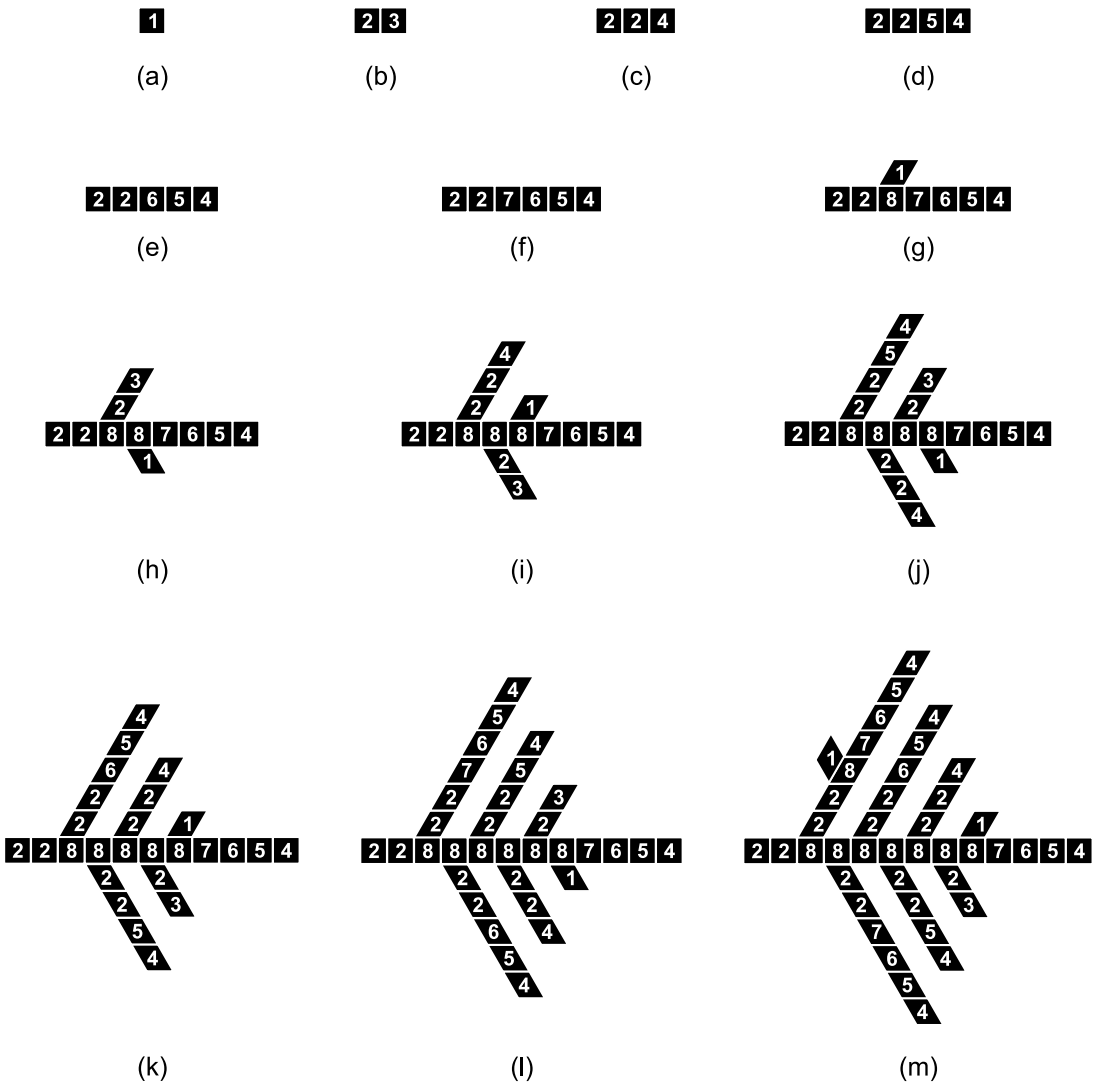

Figure 7.3: Healthy development.

and for every $a \in V$,

$$(a \to a', \emptyset), \qquad (a' \to a', \emptyset).$$

Figure 7.4 pictures the dying process corresponding to the next derivation, whose last eight strings correspond to stages (a) through (h) in the figure.

$$\begin{aligned}
1 &\Rightarrow_G^* 228[228[1]7654]8[227654]8[22654]8[2254]8[224]8[23]8[1]7654\\
&\Rightarrow_G 2'2'8'[2'2'8'[1']7'6'5'4']8'[2'2'7'6'5'4']8'[2'2'6'5'4']8'[2'2'5'4']8'[2'2'4']8'[2'3']8'[1']7'6'5'4'\\
&\Rightarrow_G 2'2'8'[2'2'8'[1']7'6'5']8'[2'2'7'6'5']8'[2'2'6'5']8'[2'2'5']8'[2'2']8'[2'3']8'[1']7'6'5'\\
&\Rightarrow_G 2'2'8'[2'2'8'[1']7'6']8'[2'2'7'6']8'[2'2'6']8'[2'2']8'[2'2']8'[2'3']8'[1']7'6'\\
&\Rightarrow_G 2'2'8'[2'2'8'[1']7']8'[2'2'7']8'[2'2']8'[2'2']8'[2'2']8'[2'3']8'[1']7'\\
&\Rightarrow_G 2'2'8'[2'2'8'[1']]8'[2'2']8'[2'2']8'[2'2']8'[2'2']8'[2'3']8'[1']\\
&\Rightarrow_G 2'2'8'[2'2'8'[1']]8'[2'2']8'[2'2']8'[2'2']8'[2'2']8'[2']8'[1']\\
&\Rightarrow_G 2'2'8'[2'2'8'[2']]8'[2'2']8'[2'2']8'[2'2']8'[2'2']8'[2']8'[2'].
\end{aligned}$$

Degeneration. Imagine a situation in which the red alga is degenerated. During this degeneration, only the main stem is able to give a birth to new branches while all the other branches lengthen themselves without any branching out. This degeneration is specified by forbidding 0L grammar $G = (V \cup \{D, E\}, P, 1)$ with P containing

$$\begin{array}{llll}
(1 \to 23, \emptyset) & (2 \to 2, \emptyset) & (3 \to 24, \emptyset) & (4 \to 54, \emptyset)\\
(5 \to 6, \emptyset) & (6 \to 7, \emptyset) & (7 \to 8[1], \{D\}) & (8 \to 8, \emptyset)\\
([\to [, \emptyset) & (] \to], \emptyset) & (7 \to 8[D], \emptyset) &\\
(D \to ED, \emptyset) & (E \to E, \emptyset). &&
\end{array}$$

Figure 7.5 pictures the degeneration specified by the following derivation, in which the last ten strings correspond to stages (a) through (j) in the figure.

$$\begin{aligned}
1 &\Rightarrow_G^* 227654\\
&\Rightarrow_G 228[D]7654\\
&\Rightarrow_G 228[ED]8[D]7654\\
&\Rightarrow_G 228[E^2D]8[ED]8[D]7654\\
&\Rightarrow_G 228[E^3D]8[E^2D]8[ED]8[D]7654\\
&\Rightarrow_G 228[E^4D]8[E^3D]8[E^2D]8[ED]8[D]7654\\
&\Rightarrow_G 228[E^5D]8[E^4D]8[E^3D]8[E^2D]8[ED]8[D]7654\\
&\Rightarrow_G 228[E^6D]8[E^5D]8[E^4D]8[E^3D]8[E^2D]8[ED]8[D]7654\\
&\Rightarrow_G 228[E^7D]8[E^6D]8[E^5D]8[E^4D]8[E^3D]8[E^2D]8[ED]8[D]7654\\
&\Rightarrow_G 228[E^8D]8[E^7D]8[E^6D]8[E^5D]8[E^4D]8[E^3D]8[E^2D]8[ED]8[D]7654.
\end{aligned}$$

□

7.2 Implementation

In this section, we describe *parametric 0L grammars* (see [151]) and their extension by context conditions. We make this description from a purely practical point of view to clearly demonstrate how these grammars are implemented and used.

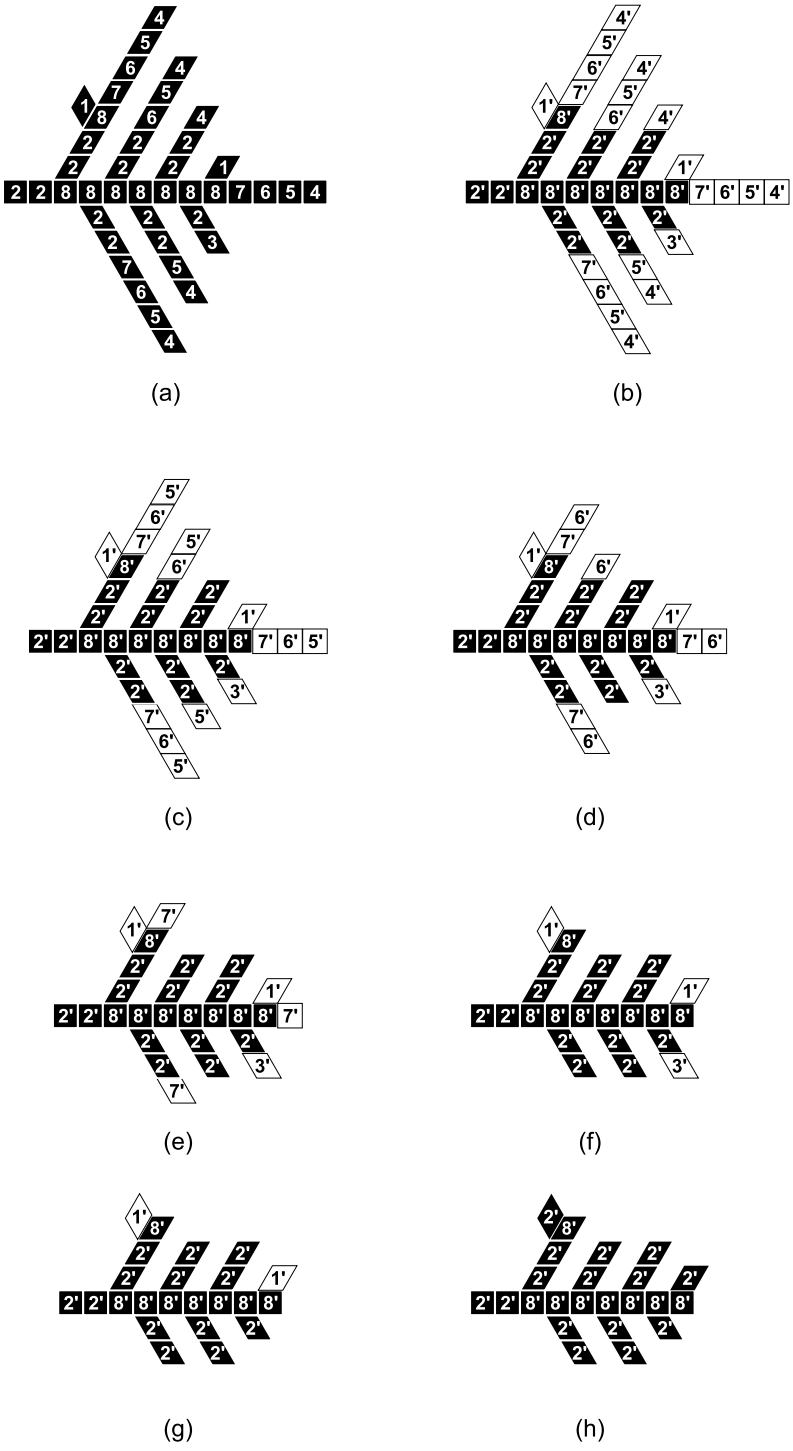

Figure 7.4: Death of marginal branch parts.

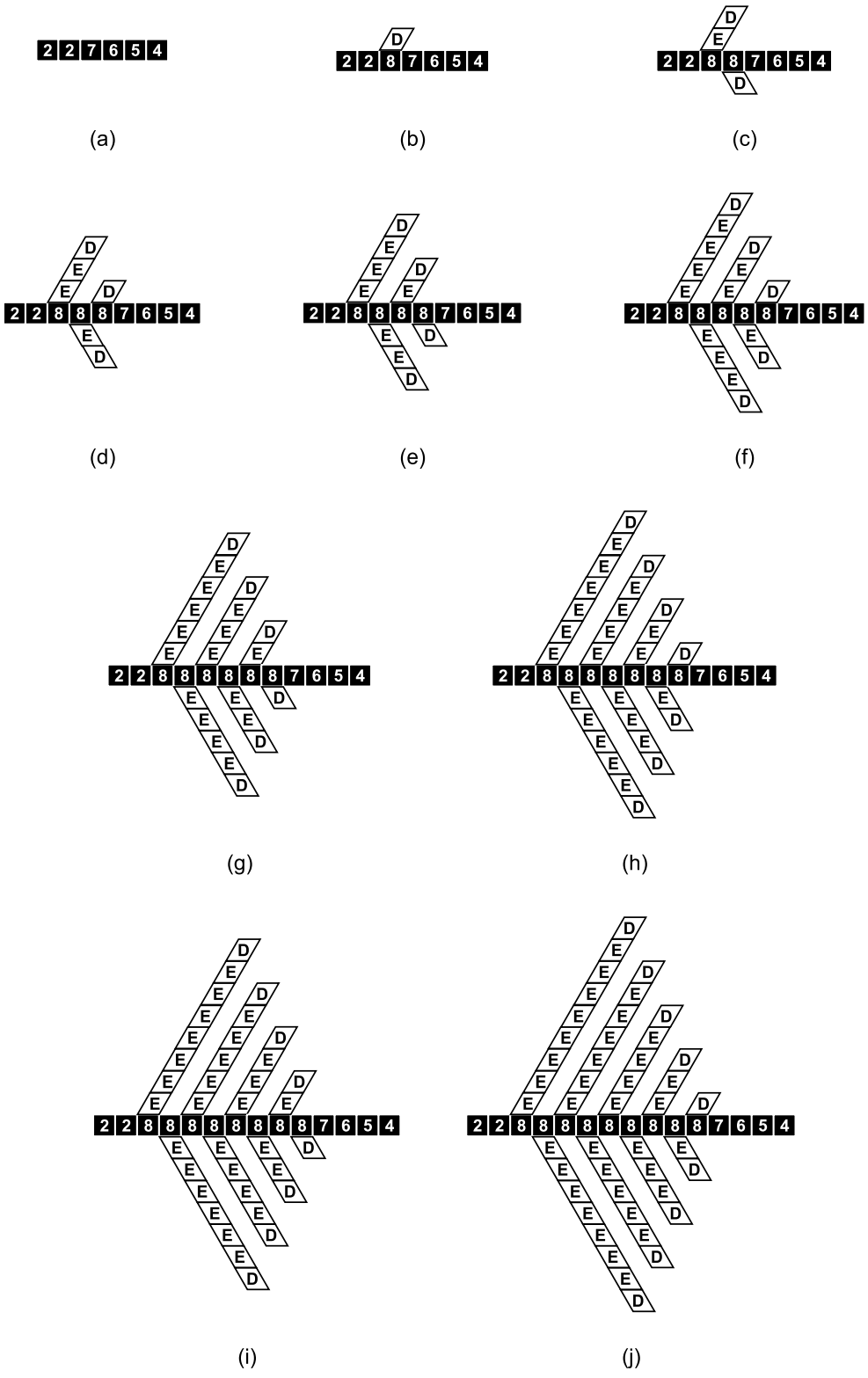

Figure 7.5: Degeneration.

Case Study 3. *Parametric 0L grammars* (see [151], [150]) operate on strings of modules called *parametric words*. A *module* is a symbol from an alphabet with an associated sequence of *parameters* belonging to the set of real numbers. Productions of parametric 0L grammars are of the form

$$predecessor\ [\ :\ logical\ expression\ ]\ \rightarrow\ successor.$$

The *predecessor* is a module having a sequence of formal parameters instead of real numbers. The *logical expression* is any expression over predecessor's parameters and real numbers. If the logical expression is missing, the logical truth is assumed. The *successor* is a string of modules containing expressions as parameters; for example,

$$A(x)\ :\ x < 7\ \rightarrow\ A(x+1)D(1)B(3-x).$$

Such a production *matches* a module in a parametric word provided that the symbol of the rewritten module is the same as the symbol of the predecessor module, both modules have the same number of parameters, and the value for the logical expression is true. Then, the module can be rewritten by the given production. For instance, consider $A(4)$. This module matches the above production since A is the symbol of production's predecessor, there is one actual parameter, 4, in $A(4)$, which corresponds to the formal parameter x in $A(x)$, and the value for the logical expression $x < 7$ with $x = 4$ is true. Thus, $A(4)$ can be rewritten to $A(5)D(1)B(-1)$.

As usual, a parametric 0L grammar can rewrite a parametric word provided that there exists a matching production for every module that occurs in it. Then, all modules are simultaneously rewritten, and we obtain a new parametric word.

Parametric 0L grammars with context conditions. Next, we extend the parametric 0L grammars by permitting context conditions. Each production of a *parametric 0L grammar with permitting conditions* has the form

$$predecessor\ [\ ?\ context\ conditions]\ [\ :logical\ expression]\ \rightarrow\ successor.$$

where the *predecessor*, the *logical expression*, and the *successor* have the same meaning as in parametric 0L grammars, and *context conditions* are some permitting context conditions separated by commas. Each condition is a string of modules with formal parameters. For example, consider

$$A(x)\ ?\ B(y),\ C(r,z)\ :\ x < y + r\ \rightarrow\ D(x)E(y+r).$$

This production matches a module in a parametric word w provided that the predecessor $A(x)$ matches the rewritten module with respect to the symbol and the number of parameters and there exist modules matching to $B(y)$ and $C(r,z)$ in w such that the value for logical expression $x < y + r$ is true. For example, this production matches $A(1)$ in $C(3,8)D(-1)B(5)H(0,0)A(1)F(3)$ because there are $C(3,8)$ and $B(5)$ such that $1 < 5+3$ is true. If there are more substrings matching the context condition, any of them can be used.

Having described the parametric 0L grammars with permitting conditions, we next show how to simulate the development of some plants by using them.

In the nature, developmental processes of multicellular structures are controlled by the quantity of substances exchanged between the modules. In case of plants, the growth

depends on the amount of water and minerals absorbed by the roots and carried upwards to the branches. The model of branching structures making use of the resource flow was proposed by Borchert and Honda in [24]. The model is controlled by a *flux* of resources, that starts at the base of the plant and propagates the substances towards the apices. An apex accepts the substances and when the quantity of accumulated resources exceeds a predefined threshold value, the apex bifurcates and initiates a new lateral branch. The distribution of the flux depends on the number of apices that the given branch supports and on the type of the branch—plants usually carry greater amount of resources to straight branches than to lateral branches (see [24] and [150]).

The following two examples illustrate the idea of plants simulated by parametric 0L grammars with permitting conditions.

(I) Consider the following model:

$$
\begin{array}{lcl}
axiom & : & I(1,1,e_{root})\,A(1) \\
p_1 & : & A(id)\ ?\ I(id_p,c,e)\ :\ id == id_p\ \wedge\ e \geq e_{th} \\
 & & \rightarrow\ [+(\alpha)\,I(2*id+1,\gamma,0)\,A(2*id+1)]/(\pi)\,I(2*id,1-\gamma,0)\,A(2*id) \\
p_2 & : & I(id,c,e)\ ?\ I(id_p,c_p,e_p)\ :\ id_p == \lfloor id/2 \rfloor \\
 & & \rightarrow\ I(id,c,c*e_p)
\end{array}
$$

This L grammar describes a simple plant with a constant resource flow from its roots and with a fixed distribution of the stream between lateral and straight branches. It operates on the following types of modules:

- $I(id,c,e)$ represents an internode with a unique identification number id, a distribution coeficient c, and a flux value e;
- $A(id)$ is an apex growing from the internode with identification number equal to id;
- $+(\phi)$ and $/(\phi)$ rotate the segment orientation by angle ϕ (for more information, consult [150]);
- [and] enclose the sequence of modules describing a lateral branch.

Standardly, we assume that if no production matches a given module $X(x_1,\ldots,x_n)$, the module is rewritten by an implicit production of the form

$$X(x_1,\ldots,x_n) \rightarrow X(x_1,\ldots,x_n);$$

that is, it remains unchanged.

At the beginning, the plant consists of one internode $I(1,1,e_{root})$ with apex $A(1)$, where e_{root} is a constant flux value provided by roots. The first production, p_1, simulates the bifurcation of an apex. If an internode preceding the apex $A(id)$ reaches a sufficient flux $e \geq e_{th}$, the apex creates two new internodes I terminated by apices A. The lateral internode is of the form $I(2{*}id{+}1,\gamma,0)$ and the straight internode is of the form $I(2{*}id,1-\gamma,0)$. Clearly, identification numbers of these internodes are unique. Moreover, every child internode can easily calculate the identification number of its parent internode; the parent internode has $id_p = \lfloor id/2 \rfloor$. The coeficient, γ, is a fraction of the parent flux to be directed to the lateral internode. The second production, p_2, controls the resource flow of a given

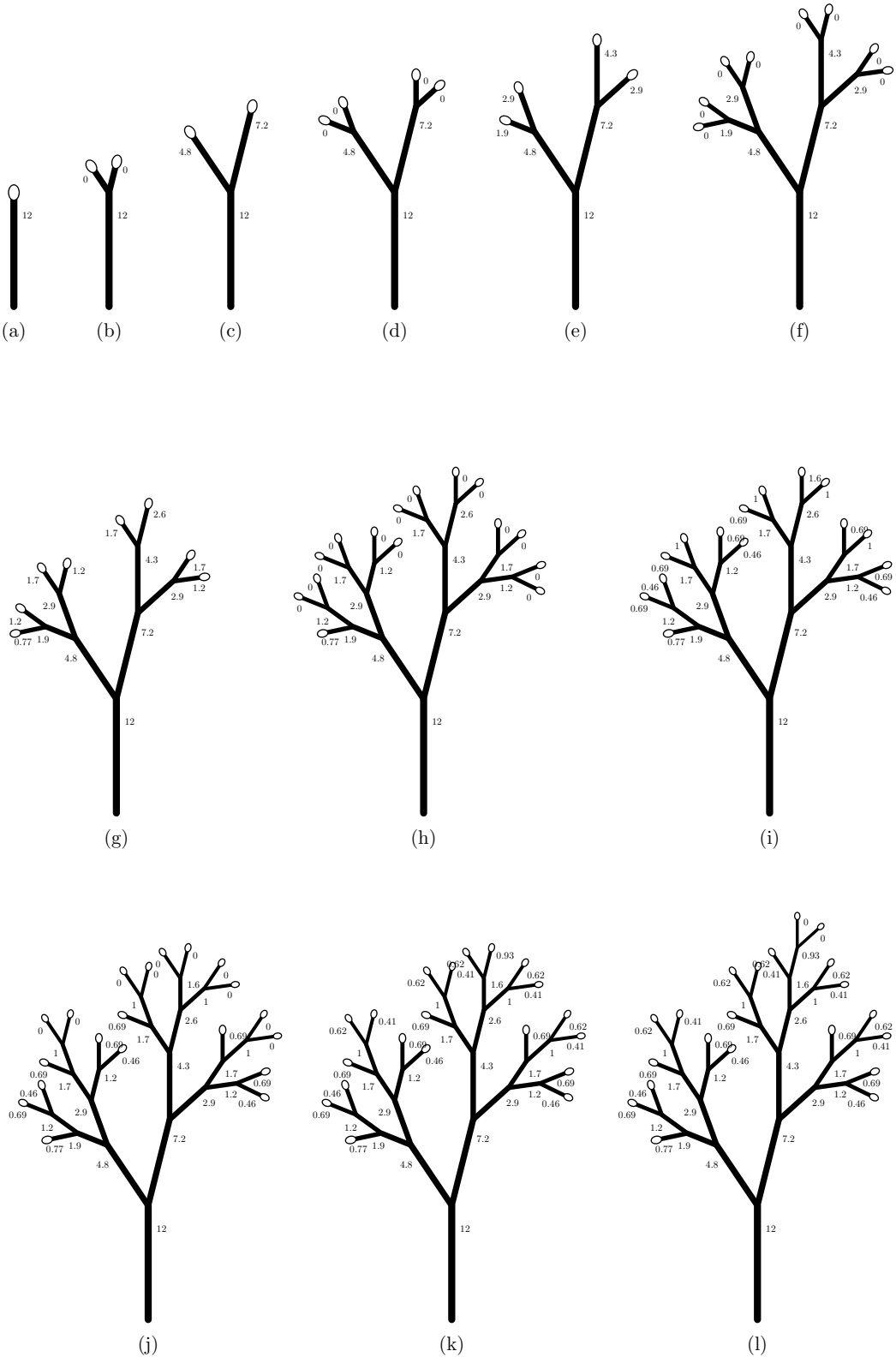

Figure 7.6: Developmental stages of the plant generated by (I).

internode. Observe that the permitting condition $I(id_p, c_p, e_p)$ with $id_p = \lfloor id/2 \rfloor$ matches only the parent internode. Thus, p_2 changes the flux value e of $I(id, c, e)$ to $c * e_p$, where e_p is the flux of the parent internode, and c is either γ for lateral internodes or $1 - \gamma$ for straight internodes. Therefore, p_2 simulates the transfer of a given amount of parent's flux into the internode. Figure 7.6 pictures twelve developmental stages of this plant, with e_{root}, e_{th}, and γ set to 12, 0.9, and 0.4, respectively. The numbers indicate the flow values of internodes.

It is easy to see that this model is unrealistically simple. Indeed, the model ignores the number of apices, its flow distribution does not depend on the size of branches, and the basal flow is set to a constant value. However, it sufficiently illustrates the technique of a communication between adjacent internodes. Thus, it is intended to be a template for more sophisticated models of plants, such as the following model.

(II) We discuss a plant development with a resource flow controlled by the number of apices. This example is based on Example 17 in [150].

$$
\begin{array}{lcl}
axiom & : & N(1)\, I(1, straight, 0, 1)\, A(1) \\
p_1 & : & N(k) \;\rightarrow\; N(k+1) \\
p_2 & : & I(id, t, e, c) \;?\; N(k),\; A(id) \\
 & & :\; id == 1 \\
 & & \rightarrow\; I(id, t, \sigma_0 2^{(k-1)\eta^k}, 1) \\
p_3 & : & I(id, t, e, c) \;?\; N(k),\; I(id_s, t_s, e_s, c_s),\; I(id_l, t_l, e_l, c_l) \\
 & & :\; id == 1 \;\wedge\; id_s == 2 * id \;\wedge\; id_l == 2 * id + 1 \\
 & & \rightarrow\; I(id, t, \sigma_0 2^{(k-1)\eta^k}, c_s + c_l) \\
p_4 & : & I(id, t, e, c) \;?\; I(id_p, t_p, e_p, c_p),\; I(id_s, t_s, e_s, c_s),\; I(id_l, t_l, e_l, c_l) \\
 & & :\; id_p == \lfloor id/2 \rfloor \;\wedge\; id_s == 2 * id \;\wedge\; id_l == 2 * id + 1 \\
 & & \rightarrow\; I(id, t, \delta(t, e_p, c_p, c), c_s + c_l) \\
p_5 & : & Id(id, t, e, c) \;?\; I(id_p, t_p, e_p, c_p),\; A(id_a) \\
 & & :\; id_p == \lfloor id/2 \rfloor \;\wedge\; id_a == id \\
 & & \rightarrow\; I(id, t, \delta(t, e_p, c_p, c), 1) \\
p_6 & : & A(id) \;?\; I(id_p, t_p, e_p, c_p) \\
 & & :\; id == id_p \;\wedge\; e_p \geq e_{th} \\
 & & \rightarrow\; [+(\alpha)\, I(2 * id + 1, lateral, e_p * (1 - \lambda), 1)\, A(2 * id + 1)] \\
 & & \quad /(\pi)\, I(2 * id, straight, e_p * \lambda, 1)\, A(2 * id)
\end{array}
$$

This L grammar uses the following types of modules:

- $I(id, t, e, c)$ is an internode with a unique identification number id, where t is a type of this internode, $t \in \{straight, lateral\}$, e is a flux value, and c is a number of apices the internode supports;
- $A(id)$ is an apex terminating the internode id;
- $N(k)$ is an auxiliary module, where k is the number of a developmental cycle to be done by the next derivation;
- $+(\phi)$, $/(\phi)$, [and] have the same meaning as in the previous example.

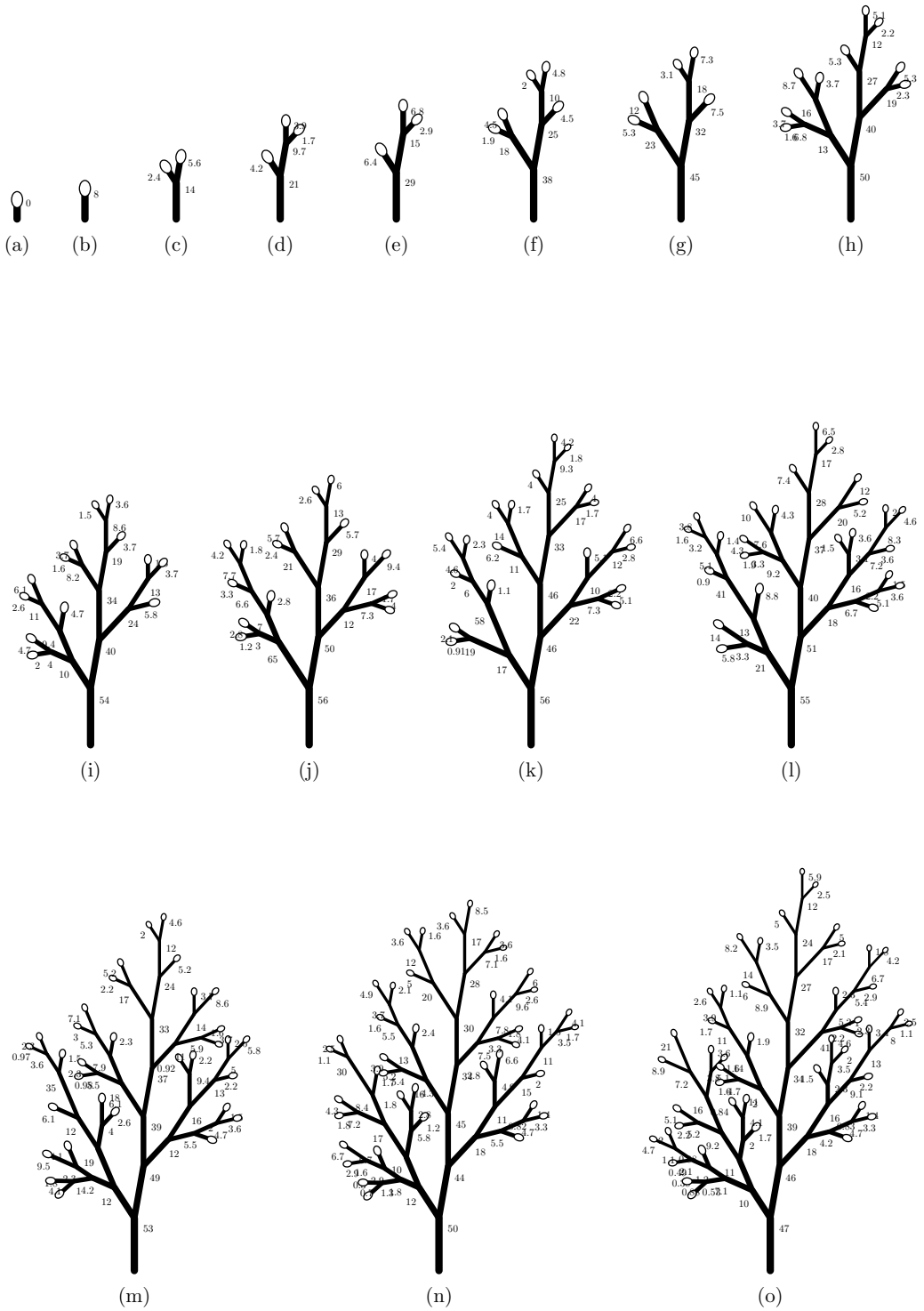

Figure 7.7: Developmental stages of the plant generated by (II).

The flux distribution function, δ, is defined as

$$\delta(t, e_p, c_p, c) = \begin{cases} e_p - e_p(1-\lambda)((c_p - c)/c) & \text{if } t = straight, \\ e_p(1-\lambda)(c/(c_p - c)) & \text{if } t = lateral. \end{cases}$$

The development starts from the axiom $N(1)\, I(1, straight, 0, 1)\, A(1)$ containing one straight internode with one apex. In each derivation step, by application of p_4, every inner internode $I(id, t, e, c)$ gets the number of apices of its straight ($I(id_s, t_s, e_s, c_s)$) and lateral ($I(id_l, t_l, e_l, c_l)$) descendant. Then, this number is stored in c. Simultaneously, it accepts a given part of the flux e_p provided by its parent internode $I(id_p, t_p, e_p, c_p)$. The distribution function δ depends on the number of apices in the given branch and in the sibling branch, and on the type of this branch (straight or lateral). The distribution factor, λ, determines the amount of the flux that reaches the straight branch in case that both branches support the same number of apices. Otherwise, the fraction is also affected by the ratio of apex counts. Productions p_2 and p_3 rewrite the basal internode, calculating its input flux value. The expression used for this purpose, $\sigma_0 2^{(k-1)\eta^k}$, was introduced by Borchert and Honda to simulate a sigmoid increase of the input flux; σ_0 is an initial flux, k is a developmental cycle and η is a constant value scaling the flux change. Production p_5 rewrites internodes terminated by apices. It keeps the number of apices set to 1 and, by analogy with p_4, it loads a fraction of parent's flux by using the δ function. The last production, p_6, controls the addition of new segments. By analogy with p_1 in the previous example, it erases the apex and generates two new internodes terminated by apices. Figure 7.7 shows fifteen developmental stages of a plant simulation based on this model.

Obviously, there are two concurrent streams of information in this model. The bottom-up (acropetal) stream carries and distributes the substances required for the growth. The top-down (basipetal) flow propagates the number of apices which is then used for the flux distribution. A remarkable feature of this model is the response of a plant to a pruning. Indeed, after a branch removal, the model redirects the flux to the remaining branches and accelerates their growth.

Let us note that this model is a simplified version of the model described in [150], which is very complex. Under this simplification, however, $c_p - c$ may be equal to the zero as the denominator in the distribution function δ. If this happens, we change this zero value to the proper non-zero value so the number of apices supported by the parent internode corresponds to the number of apices on the straight and lateral branches growing from the parent internode. Consult [150] for a more appropriate, but also complicated solution of this problem.

From the presented examples, we see that parametric 0L grammars with permitting conditions can describe sophisticated models of plants in a very natural way. Particularly, compared to the context-sensitive L grammars, they allow to refer to modules that are not adjacent to the rewritten module, and this property makes them more adequate, succint and elegant from a practical point of view. □