Tree and stand architecture and growth described by formal grammars.

I. Non-sensitive trees.

Modelování stromové a porostní architektury
a jejího růstu pomocí formálních gramatik.
I. Pro stromy nesenzitívního typu.

Winfried Kurth, Branislav Sloboda
Universität Göttingen
Institut für forstliche Biometrie und Informatik
Büsgenweg 4, D–37077 Göttingen

wkurth@ufobi4.uni-forst.gwdg.de
bslobod@ufobi6.uni-forst.gwdg.de

Abstract: With the increase in computer performance, 3-dimensional simulations of tree architecture with an unprecedented level of detail became possible. A formal framework for such structural models is provided by parallel string rewriting grammars (Lindenmayer systems). This formalism is presented here, together with several extensions tailored for botanical and forestry applications, comprising interpretative rules, an accelerated expansion of string parts, and object references. These formalisms are demonstrated by models of growing beech shoots and spruce stands, having an empirical basis and made visible by a generic software named GROGRA.

Keywords: tree architecture, virtual tree, crown morphology, crown development, Lindenmayer system, growth grammar, simulation, visualization

Abstrakt: S rostoucím výkonem počítačů je také umožnena 3-dimensionální simulace stromové architektury s doposud nevídaným stupněm její komplexity. Formální technika pro tyto strukturální modely se realizuje pomocí gramatiky, která paralelně interpretuje textové řetězce (Lindenmeyerovy systémy). Tento formalismus je zde podán společně s dalšími obsahovými rozšířeními, která jsou přizpůsobena pro botanické a lesnické aplikace a která také obsahují mj. interpretační pravidla, zrychlené prodlužování částí stringů a objektové informace. Uvedený formalismus je demonstrován na růstovém modelu pro bukové výhony a pro smrkový porost, které jsou empiricky podloženy a byly vizualizovány pomocí generického software GROGRA.

1. Introduction

Modelling can be conceived as a means to organize scientific knowledge and to check hypotheses for consistency. In the last decades, models in biology and forestry have often concentrated on population dynamics or on processes affecting or being part of the metabolism of plants. In these "functional models", the highly heterogeneous structure of forests was often grossly simplified or even neglected. However, knowledge about the structure of plants and plant communities, resp. forest stands, forms an important part of botany and forest science. Only by creating mathematical models which include some structural, morphological features, one can hope to construct a bridge from the descriptive language of classical botany to contemporary quantitative methods (Kurth, 1994). Even more, the very activity of changing the structure of forests is the main way how foresters — traditionally, and still today — control the development of trees and of whole stands. The ecological consequences of different silvicultural techniques, the reactions of trees on thinnings, determinants of wood quality like stem shape or knottiness — all this is intimately related with the 3-dimensional structure of trees and with its control and development during ontogenesis. Since all processes in a forest depend on this structure, even "functional modellers" detect more and more the relevance of structure-function relationships (cf. Fisher, 1984; Küppers, 1989) and include morphological features into their process-oriented models. The term "functional-structural model" was coined for approaches combining both aspects, morphology and process (Sievänen et al., 1997).

But for modelling plant structures today, there is no need to begin with Adam and Eve. More or less independently from the "mainstream" of biological process-oriented modelling, there have been at least six traditions of structural plant studies which led to morphological models and computer simulations:

1. Emerging from the work of Hallé, Oldeman (1970) and Hallé et al. (1978) on the (qualitative) architectural classification of trees, there was a French school on tree morphology, which had — together with practical, agronomical motivations (cf. e.g. Reffye, Snoeck, 1976) — strong influence on the creation of the AMAP simulation system ("Atelier for the modelisation of the architecture of plants") at the CIRAD, Montpellier (see Reffye et al. (1995) for an overview of the current directions of development).

2. Another research direction, traditionally strong in the anglosaxon countries, concentrated on theoretical considerations and on the detection of general morphological rules. The monographies of D'Arcy Thompson (1942) and Stevens (1974) can be seen as forerunners. Interest in the modular construction of plants (White, 1979), in the demography of plant organs (Maillette, 1982), in the fate of buds (with simulation studies based thereupon: Bell, 1976; Bell et al., 1979), in foraging and growth strategies (Bell, 1984; List et al., 1994; Stoll, 1995) and particularly in cost-benefit relationships (Oborny, 1994) and optimality (Fisher, Honda, 1979) can be seen as typical for this rather diverse strand of tradition. Morphology is conceived as process (Sattler, 1992), and it is tried to bring adaptation in a relation to quantitative morphological indices (Borchert, Slade, 1981) — the latter approach, however, is discussed controversially (Waller, Steingraeber, 1983; Fisher, 1984). Paleobotanical analyses of morphology (Niklas, 1982, 1986) and simulations of the evolution of growth rules (Kim, 1996; Jacob, 1997) can also be

subsumed under this "theoretical biology" studies, since form is seen from the perspective of a singular theoretical principle (adaptation).

3. Originating with the work of Lindenmayer (1968) on rewriting systems describing the growth of filamentous organisms, there was a "theoretical computer science" strand of tradition, concerned with the so-called Lindenmayer systems (L-systems; we will give an exact definition later in paragraph 2.3). This mathematical formalism parallels the notion of "phrase-structure grammar" used in linguistics to study the syntax of natural languages and of programming languages. It was an important step by Lindenmayer to apply such a formalism to the shape and growth of organisms and hence to build a bridge between biology and formal language theory. However, most of the work done on L-systems concentrated on mathematical properties of the sets of strings defined by them, not on the morphological structures encoded by the strings. — Another approach, also based on a more or less combinatorial formalism, was the matrix method of tree simulation of Viennot et al. (1989). A revitalization of plant modelling with L-systems came with the monography of Prusinkiewicz and Lindenmayer (1990). New developments are described in Prusinkiewicz et al. (1997). Our approach is also strongly based in this formal foundation of describing plant development by grammars (Kurth, 1994b).

4. For the purposes of visualization and animation of plants, researchers from computer graphics became interested in tree structure. Their algorithms for creating virtual plants were often ad-hoc, and concentrated on the optical impression and on efficiency rather than on botanical validity (Kawaguchi, 1982; Bloomenthal, 1985; Reeves, Blau, 1985; Greene, 1989; Brinkhus et al., 1995; Šoch, Beneš, 1995; Deussen, Lintermann, 1997). However, important ideas concerning discretization and efficient data processing coming from this field were useful for other, more substantial models, too.

5. The discussion on forest decline and acid rain on the one hand, and the demand for new predictive tools and for rationalization in forestry on the other hand initiated a renewed interest in morphological phenomena in frequent forest trees (Gruber, 1987; Roloff, 1989; Schill, 1989; Kellomäki, Kurttio, 1991; Colin, 1992; Tsel'niker, 1994; Deleuze et al., 1996) and in integrated models containing physiological and structural features (Ford et al., 1990; Kellomäki, Strandman, 1995; Pfreundt, Sloboda, 1996; Perttunen et al., 1996) among forest scientists and ecologists. The failure of some process-based models to reproduce the effects of soil pollution on tree growth led to individual-based forest models with more refined structural components but less detailed process descriptions (Hauhs et al., 1995).

6. At last, special-purpose architectural plant models were developed in diverse contexts, particularly in zoology (concerning the interaction of insects with tree crowns: Kaitaniemi, Honkanen, 1996) and in bioclimatology (modelling the radiation regime in plant canopies: Goel et al., 1991; Knyazikhin et al., 1996; Govaerts, 1996).

Today, these strands of tradition are strongly interwoven with each other. Ad-hoc plant simulation models are still constructed and can have some advantages in efficiency and interfacing with special software; however, the formalism of L-systems has meanwhile been extended to a powerful description tool, having the characteristics of a general programming language (Chien, Jürgensen, 1992; Goel, Rozehnal, 1992; Borovikov, 1995) and including specialized features for morphological plasticity and interaction mechanisms in plants (Kurth 1994b, 1997; Prusinkiewicz et al., 1997). The advantage of having such a formal basis lies in the precision and reproducibility of model specification, in the applicability of theoretical results from formal language theory, and in a higher universality and — hopefully — transparency of modelling.

In this paper, we will explain an extended L-system formalism, which was designed for botanical tree simulations with the software GROGRA (Growth Grammar Interpreter, Kurth, 1994b). The classical ingredients of these "growth grammars" are presented in Section 2, and some new extensions in Section 3. The application of these extensions is demonstrated by examples in Section 4. A short description of GROGRA and a critical discussion of the results will finish this paper, which is devoted to "non-sensitive" grammars, i.e. to architectural growth specifications neglecting the influence of the environment on the tree (or accounting for it implicitely by stochastic components). Such models reflect to a great extent the genetical basis of plant form. The sequel of the paper will focus on "sensitive" grammars, i.e. on the representation of causal relationships between the tree and its environment.

2. Fundamental formal tools

2.1. Overview

The specification of structural development by formal grammars can be conceived as being stratified in three levels of formal description, each with its own ideas, mathematical background and symbolization (Figure 1; Kurth, 1994b):

(1) The specification of a structure at a fixed moment in time (statical description). This level of specification can be further differentiated into
(1a) topology,
(1b) geometry,
(1c) additional, non-geometric attributes.
In the growth-grammar approach all these specifications are achieved by the so-called *turtle geometry* (Abelson, diSessa, 1982), an interpretation method relating command strings (input) to geometrical structures (output).

(2) The specification of how these structures develop in time. This is done by *rule systems* operating in a parallel manner on *strings* (which are in turn interpreted in a geometrical manner on level 1). We speak of Lindenmayer systems (*L-systems*) in the narrower sense.

(3) The control which of several applicable rules is actually applied in a given situation. This control is carried out in the framework of *growth grammars* with the help of features like *stochastic rules*, *random variables*, *parameterization*, *conditions*, *two-phase* rule application, *sensitivity* and *local register handling*.

Some of these specification tools and features are only briefly sketched here, some are explained in more detail below.

2.2. Turtle geometry

When the dynamical aspect is neglected, the objects which are created by the GROGRA software are simply *branching structures* in 3-dimensional space which are built up of *cylindrical units* — each with its own length and diameter — which can bear additional, non-geometric *attributes* (standing e.g. for colour or assimilate content). We speak of *(geometrical) structures*. The cylinders are called *elementary units* and normally have the meaning of botanical *growth units* or *shoots* (see Room et al. (1994) for a list of definitions of relevant morphological terms). In some applications with a more detailed level of resolution they can also stand for internodes; examples will follow below. The introduction of *object references*, which will be explained in Section 3.3, enables the inclusion of more complex objects instead of the simple cylinders, but the basic principles remain the same. Each unit has at most one *mother unit* (supporting unit) from which it emerges.

Turtle geometry is a formal tool to describe such structures. Its underlying philosophy is that there is a device, called the *turtle*, which can be told by *commands* to move, to change its orientation in space and other attributes, and *to construct elementary units with given attributes* while moving. The *state* of the turtle is a data record containing all relevant informations on its position, orientation, and all attributes to be used in the next construction step. More precisely, the state variables of the turtle include:

$P = (p_x,\ p_y,\ p_z)$, a vector specifying the *position* of the turtle in space,

$H = (h_x,\ h_y,\ h_z)$ (= *head*), a vector of unit length specifying the direction of the turtle's next move,

$L = (l_x,\ l_y,\ l_z)$ (= *left*) and $U = (u_x,\ u_y,\ u_z)$ (= *up*), two vectors of unit length, orthogonal to each other and to H, specifying directions to be considered as "left" and "upwards", respectively, with respect to the turtle,

ℓ, a real number specifying the *length* of the next move and at the same time the length of the next elementary unit to be constructed,

d, a real number specifying the *diameter* of the next elementary unit to be constructed,

v, a real number specifying a *vertical tendency* which is to be obeyed when a special command, the *RV* command, is given,

n, a real number specifying some non-geometrical parameter of the next elementary unit to be constructed, currently used for *foliage mass* or *surface* supported by the shoot,

c, a real number specifying another non-geometrical parameter, currently used for the quantity of *carbon* or *assimilate* stored in the shoot,

u, an integer specifying a parameter which can be used for various purposes,

p, an integer specifying the *colour* of the next elementary unit (in the current GROGRA version the colour is encoded according to the simple EGA colour table with 16 colours, but this could easily be extended),

q, a real number between 0 and 1 specifying the *relative position* along the mother unit where the next elementary unit will emerge.

There are some more state variables which are of a more technical nature, like the branching order, a reference to the mother unit and the distance to the basic (i.e., motherless) unit; see Kurth (1994b) for details. The default values of most of the state variables can be specified by the user of GROGRA in the declaration part preceding a growth grammar specification. To this purpose, the commands \set or \ask (the latter is for interactive request) are to be used (examples below will demonstrate this possibility). To change the values of the state variables while the turtle is working, several commands are available. Our turtle command language is a systematized and enlarged version of that used by Prusinkiewicz and Lindenmayer (1990); see Kurth (1994b, 1997) for a complete documentation. Here we introduce only the commands which will be used in the subsequent examples.

The *standard assignment commands* all begin with one of the uppercase letters L, D, V, N, C, U or P and control the value of the respective lowercase state variable. Each of these letters can be used without an argument or with one (numerical) argument, and — when used with an argument — eventually together with some modifyers. E.g., the commands influencing length (L commands) have the following rather systematical syntax and semantics:

L *(without argument)*	sets the state variable ℓ back to the default value,
L(x)	gives ℓ the new value x,
L$+(x)$	adds x to ℓ
L$*(x)$	multiplies ℓ by x.

Furthermore, the commands L$\ell(x)$, L$\ell+(x)$ and L$\ell*(x)$ (*local* commands) are possible which have the same effect as their counterparts L(x), L$+(x)$ and L$*(x)$, except that their effect is restricted to the next constructed elementary unit. (The effect of the "normal" commands — without "ℓ" — will last until the next command influencing the same state variable occurs.) The other standard assignment commands are built in an analogous manner, with L replaced by D, V etc.

The *movement* of the turtle in space and the *construction of a new unit* are triggered by the commands f (movement by length ℓ without construction) and F (movement together with the construction of a new unit). Variants like F(x) (enforcing the unit length x without changing the value of ℓ) are also possible, see Kurth (1994b) for full details. A special movement command is @(x), where x is usually a number between 0 and 1. This command is equivalent to f$((x-1)\cdot\ell)$, i.e. the turtle is positioned at a place on the last created unit with the distance x from the basis, expressed in terms of the length of the unit (relative positioning of the turtle). This command is normally used when the position of a lateral bud or a branching node on a shoot is to be specified.

Rotations of the turtle can be specified with respect to the directions H, L and U, which are part of the state of the turtle. E.g., the command RU(x) induces the turtle to turn around the U-axis by x degrees. The position remains unchanged. The commands RH(x) and RL(x) have analogous effects. The whole orthonormal system (H, L, U) is rotated by these commands, but the relation of H, L, U relative to each other remains unchanged.

Another command changing the direction of the turtle is RV. In contrast to RH, RU and RL, this command takes an information of global character into account, namely, the *vertical* (geotropic) *direction*. The inclusion of tropism vectors in L-systems was already described by Prusinkiewicz and Lindenmayer (1990) and can be viewed as a first step towards global sensitivity. Technically, RV calculates the new H direction of the turtle as the normalized sum of H and $(0,\ 0,\ -v)$, the latter vector pointing strictly downwards. A sequence of alternating commands F and RV thus leads to a "bending down" of the generated branch. The greater the value of v, the stronger the bending. Negative values of v are also possible, leading to orthotropic growth.

Two important special turtle commands are the brackets "[" and "]". By their use the occurrence of branching is controlled. When an open bracket "[" is read, the complete actual state of the turtle is stored in a pushdown memory (stack). When the corresponding closed bracket "]" is reached, the turtle stops the construction of the branch which was begun with "[", forgets its current state and adopts again the old state which was actual when the "[" was entered, removing the state from the stack and jumping to the old position. Thus, the commands after "]" will induce the turtle to continue the construction of the "main branch". Another possible interpretation — which leads to the same result — is that the turtle "divides" itself at each open bracket and constructs the side branch (corresponding to the string inside the brackets) and the main branch (string after the closed bracket) in a parallel manner. This interpretation is closer to the reality of plant growth, where meristems can divide itself and work in parallel, but due to the lack of properly parallel computers, the current implementation of GROGRA works according to the first, sequential interpretation.
Two additional commands are useful for constructions of closed polygons (e.g. in leaves):

$$\begin{array}{lcl} \mathsf{S}(n) & = & \text{“save”}, \\ \mathsf{C}(m,\ n) & = & \text{“connect”} \quad \text{(with two arguments).} \end{array}$$

$\mathsf{S}(n)$ (n being an arbitrary integer) saves the current state of the turtle in an internal list under the identificator n. $\mathsf{C}(m,\ n)$ (m and n being integers which were used before in S-commands) puts the turtle at the position which was its position when $\mathsf{S}(m)$ was executed, and moves it afterwards to the position corresponding to the state which was actual when $\mathsf{S}(n)$ was executed, thereby creating a new elementary unit between the two positions. Diameter, colour, N- and V-value are not taken from the saved state, i.e. they can be specified before the $\mathsf{C}(m,\ n)$-command is executed. Length and direction vectors $(H,\ L,\ U)$ are calculated from the vector connecting the two positions, the other state variables of the turtle (including the mother shoot) are taken from the state saved by $\mathsf{S}(m)$.

A quick overview of all commands used in our later examples is provided by the table given in the Appendix.

2.3. Lindenmayer systems

A classical *Lindenmayer system* (0L system; the "0" stands for "zero context") consists of

- an *alphabet* Σ, consisting of a finite number of *symbols* (e.g. a, b, c),
- a *start string* (or *axiom*) α which is made up of symbols from Σ (e.g., aba),

- a set of *replacement rules*, each of the form

$$\textit{symbol} \longrightarrow \textit{string of symbols}$$

(e.g.: $b \longrightarrow cca$), which are to be applied in parallel to all symbols of a string at time t in order to get a new string at time $t + 1$.

We adopt the convention that a symbol for which no rule is applicable remains unchanged.

The rewriting process, i.e. the application of the rules to the given string, will normally be iterated several times. Thus we get a (potentially infinite) *sequence of strings* $\sigma_0, \sigma_1, \sigma_2, \ldots$, where σ_{t+1} is obtained from σ_t by application of the replacement rules, and $\sigma_0 = \alpha$. This rewriting process of strings forms the heart of the L-system modelling approach. We refer to the numbers or timesteps 1, 2, 3... as *generations* and call the above-mentioned replacement rules also *generative rules*. Technically, in the GROGRA syntax the arrow " $\longrightarrow$ " of a generative rule is written as "#". The rules are specified in a file, whereas the start string can be asked interactively from the user.

L-systems are a special case of *grammars* in the sense of formal language theory; see Kari et al. (1997) for an overview on L-systems from the mathematical and linguistic perspective.

But what is the connection between this rewriting system and turtle geometry, i.e. to the specification of 3-D structures representing plants? In the definition of a *growth grammar* we have to include, additionally to the above-mentioned three ingredients,

- a *geometrical interpretation* of the strings (i.e. a *semantics*) translating strings into spatial structures,

and this interpretation is given by turtle geometry. The complete turtle command set T is thus included as a subset in the alphabet Σ. Symbols contained in Σ but not in T are ignored by the turtle. They can have a meaning in the rewriting process generating new generations of strings, but not in the geometrical interpretation.

The interpretation is applied to each string of the generated sequence $\sigma_1, \sigma_2, \sigma_3, \ldots$, yielding a *sequence of geometrical structures* $S_1, S_2, S_3, \ldots$. Hence we arrive at the situation visualized in Figure 2. Here, the horizontal arrows stand for the development governed by the generative rules, whereas the vertical arrows correspond to the geometrical interpretation. Only the structures $S_1, S_2, S_3, \ldots$ are the output of the GROGRA software. The strings $\sigma_1, \sigma_2, \sigma_3, \ldots$ used for their creation are only written temporarily into an auxiliary file and are (in the current GROGRA version) not the object of further analysis once they have been used to construct the structures.

For example, let a stand for a terminal bud of a main axis, b for a bud of a lateral axis, and let F have the already explained turtle command meaning, i.e. F stands for a *shoot* (growth unit). Then the L-system composed of the two generative rules

$$\begin{array}{lll} a & \longrightarrow & \mathsf{F}\ [\ \mathsf{RU45}\ b\ ]\ a, \\ b & \longrightarrow & \mathsf{F}\ b \end{array}$$

gives, when applied to the start word a, the sequence of strings and the corresponding sequence of structures which are depicted in Figure 3 (only the two first developmental steps are sketched).

The buds symbolized by a and b will normally not be visible in the turtle interpretation (if they are not subject to other, interpretative rules; see below), they are depicted as small ovals here to enhance the understanding of the process. Note that F remains unchanged in the second step because no rule is applicable to F.

In a *parametric L-system* (Prusinkiewicz, Lindenmayer, 1990), to each symbol there can be attached a finite list of real-valued parameters, e.g. $a(7,\ 42,\ \text{-}0.5)$ instead of a. We speak of *modules* instead of symbols in this case. These modules form *parametric strings*, i.e. strings of modules. The L-system rules may now contain formal parameters, e.g.

$$a(x,\ y,\ t) \longrightarrow b(2*x+y,\ t)\ a(x,\ y,\ t+1)\ c.$$

If this rule is applied to, let us say, $a(-1,\ 5,\ 4)$, we get the parametric string $b(3,\ 4)$ $a(-1,\ 5,\ 5)\ c$. Furthermore, *conditions* like $(t = 0\ \&\&\ x > y/2)$ containing the formal parameters may restrict the applicability of a rule.

In the arguments attached to the symbols on the right-hand side and in the condition, variables and constants can be combined in *arithmetical expressions*, using the (slightly extended) syntax for such expressions from the programming language C. Analogously, the *condition* is generally a *logical expression*. (Further details can be found in Kurth, 1994b.)

In a *stochastic* L-system, there may exist several rules with the same symbol (or module) on the left-hand side, each of them being attributed a probability. The probabilities of all rules having the same l.h.s. should sum up to 1. In the repeated application of rules, the choice is done randomly with the given probabilities. This implies that the generative process now is no longer necessarily deterministic: The random choice of rules can lead to different sequences $\sigma_1, \sigma_2, \sigma_3, \ldots$ from the same start string α. Syntactically, in GROGRA the probability is written after a "?" at the end of a rule. It can be specified by a floating-point number or by an arithmetical expression. The general form of a growth grammar rule is hence

$$(condition)\ \ l.h.s.\ \ \#\ \ \ r.h.s.\ ?(probability\_expression),$$

where "(*condition*)" and "?(*probability_expression*)" can be omitted.

If several rules apply to a module, and if no conditions and probabilities are involved, the order in which the rules are written down in the grammar file dictates which rule will be applied: It is the first one.

The potential stochasticity of rule applications is further enlarged by the possible *declaration* of variables as "random" (with a specified distribution controlling the outcome of their value) and by their use in expressions occurring in modules on the r.h.s. of rules or in conditions or probability expressions of rules. GROGRA allows the declaration of variables of a multitude of different types in a declaration part preceding the list of rules of a growth grammar. An especially simple form of declaration is that of a *constant*. It has the form

\const *name* *value*,

where *value* is a floating-point number, and the effect is that every appearance of *name* in a position in the rules where a numerical value is permitted is replaced by *value*. The general syntax of a *variable* declaration is

$$\backslash\mathsf{var}\ \ name\ \ specification\ \ parameter\_list,$$

where *specification* must be one of several possible keywords which specifies from where the variable gets its value when it appears in a rule which is to be applied. E.g., "uniform" qualifies a uniformly distributed random variable, "normal" one with a normal distribution, "poisson" one of Poisson type etc.; see Kurth (1994b, 1997) for a complete list of types.

The *parameter_list* consists of one or several numerical values; it can also be empty. How many values are required and what their meaning is depends on the specification. E.g., \var $x1$ binomial 0.3 20 specifies $x1$ as a random variable with binomial distribution, described by the parameters $p = 0.3$ (probability of success of single Bernoulli experiment) and $n = 20$ (number of repetitions).

Attention is to be paid to the situation when a certain random variable appears twice (or even more often) on the r.h.s. of the same rule. E.g., in $a(x1)\ b(x1)$, the argument of a is taken from a random-number generator working according to the specified distribution, and the same holds for the argument of b — with the consequence that both values will normally differ from each other. If b shall have the same value than a, the *memory operator*, symbolized by an underline (_), has to be applied to the second instance of $x1$: $a(x1)\ b(x1\_)$.

It is also possible to declare a variable with specification "function", which makes it refer to one of a predefined set of special functions implemented in GROGRA. The *parameter_list* is $n\ a$, where n is the number (identifier) of the function and a its arity (number of arguments). Only numerical arguments are counted here, not a possible dependence of the outcome of the function from the generated structure (sensitivity; see Kurth, Sloboda, 1998). The arity may be 0. When a variable declared as function appears in an expression to be evaluated, it must be followed by a list of a parameters. The possible functions are listed in Kurth (1994b, 1997).

3. Extensions of the grammar machinery

3.1. Two-phase growth grammars

The structure-generating process can be considerably flexibilized by the introduction of a second set of rules (possibly empty), named *interpretative rules* or *rules of the second phase*. Their application intervenes between the generation of a string σ_k and its interpretation. That means, the creation of the structure S_k is delayed until an eventual application of rules of the second phase has transformed σ_k into a string σ_k'. (If the set of interpretative rules is empty or if no such rule is applicable, $\sigma_k' = \sigma_k$ is assumed.) Note that the interpretative rules — in contrast to the "ordinary", generative rules — have no influence on the next generation string σ_{k+1}. Figure 4 summarizes this modified structure generation process, with the vertical double arrows indicating the application of interpretative rules.

Two-phase grammars have proved very useful for abbreviating complex geometrical constructions which stand in no connection with the developmental process modelled by the generative rules. Biologically, we may think of structure-forming processes running on different hierarchical levels and time-scales, one of them handled as taking place "immediately" in the model and thus requiring no generative step but only rules of the second phase. Syntactically, the interpretative rules in GROGRA are distinguished from the generative rules by the use of two double-crosses (##) instead of one, fulfilling the function of the arrow of the rule.

3.2. Accelerated expansion

Often there arises the necessity to generate a rather complex structure during a single developmental step of generative rule application. Interpretative rules do not provide a solution to this demand, since their application has no influence on the next generation.

A first formal tool to compress long, redundant rule specifications is the *repetition operator* (implicitely used by Goel et al., 1991; "cycle operator": Borovikov, 1995). It can appear as a part of the right-hand side of a rule in the form

$$\&(nb\_iterations) < \sigma >$$

where *nb_iterations* is an arithmetical expression determining the number n of iterations and σ an arbitrary string of symbols from the L-system alphabet which will be iterated n times once the rule is applied. Inside the string σ, a declared variable of type *index* may occur which counts the iterations from 0 to $n-1$. The repetition operator is useful in cases when a number of similar plant organs (internodes, leaves, lateral buds) is expanded during one and the same developmental step. It can be nested, i.e. another repetition operator may occur in σ. Correspondingly, several index variables distinguished by a *depth* of nesting may be declared; see Kurth (1996) for details.

A more general construction, enabling the *nesting of grammars*, is possible with the *expand operator* E, introduced by Kurth (1996) for models of shoots of deciduous trees. The appearance of

$$E(nb\_steps) < \sigma >$$

on the right-hand side of a rule (with *nb_steps* being an arithmetical expression yielding an integer value n) enforces the "instantaneous" application of n steps of the grammar to the string σ and the insertion of the resulting string, all taking place during one developmental step. This is useful when growth processes running on different time scales are to be synchronized. — Again, nested constructions of several E operators are possible. An "expand" is also allowed in interpretative rules.

3.3. Object references

By the use of the expand operator, complex objects like leaves requiring several steps of rule application for their definition can be created during one developmental step. However, this has the disadvantage that every simple elementary unit defining the geometry of the leaf (veins, parts of the periphery...) is created for every leaf again, leading to an inacceptable waste of memory space during simulation. Therefore, GROGRA allows the creation of several complex objects from grammars and their use in other grammars as

single objects. In the consequence, a complex object like a leaf or a whole plant will be represented by a mere reference to the structure created once in the beginning, and each occurrence will require no more memory space than a simple elementary unit, which is created by the F command. (The disadvantage is that these "multiply referenced" objects will — with the exception of a simple scaling — look all alike, i.e. they are really "clones" of the prototype constructed in the beginning.)
By the line

$$\backslash\mathsf{axiom}\quad \alpha \quad \textit{list\_of\_steps},$$

where α is a string and *list_of_steps* a list of strictly monotonically increasing positive integers, GROGRA is told to apply the rules of the current grammar file to the start word α and to create the structures corresponding to the specified steps in memory. (If several objects are required, there is nevertheless no need to write the grammars into distinct files; it is sufficient to specify different start words by a corresponding number of \axiom-statements. The corresponding applicable rules can be scattered through the whole file.) With an additional line "\askaxiom", GROGRA is enforced to ask an additional start word (and number of steps) from the user. The *list_of_steps* may also contain specifications like m–n (m and n being integers, $m < n$), meaning that all developmental steps between m and n are to be realized.
The reference to a created object is specified by the command

$$O\,(\alpha,\ n)$$

— occurring in the right-hand side of a rule —, where α must be a start string used in an \axiom-statement and n an integer which must be part of the *list_of_steps* which was specified for α (GROGRA can establish references only to objects which are really created before). An iteration of object references is possible, i.e. the rules used in the evaluation of α may again contain "O" commands.

The scaling of the object referenced by $O\,(\alpha,\ n)$ is determined by preceding Lℓ and Dℓ commands. While L$\ell(f)$, with f a positive floating point number, causes an enlargement of the whole object by the factor f, the command D$\ell(f)$ changes only the extension of the object perpendicular to the current axial (H-) direction by the factor f, i.e. the whole object becomes "thicker" or "thinner" (useful for crown shapes of trees when working on the stand level).

The examples in the next section will demonstrate the possibilities opened by the O-command in the case of modelling single plants and tree stands.

4. Examples

4.1. Shoot growth of European beech (*Fagus sylvatica* (L.))

The following growth grammar is not based on detailed quantitative investigations, but on well-known characteristics of growth habit and shoot morphology of European beech (see e.g. Roloff, 1994). Table 1 shows a rule system which can directly be interpreted by the GROGRA software and which encodes four years of the development of a lateral shoot system of beech, each year being represented by two steps (winter: leafless, summer: with leaves).

Flowering is not included here. To keep the example simple, branch development is assumed to take place in two dimensions only, which is an approximation coming close to reality for branches growing in the shade. The growth grammar contains a bud and a leaf submodel, both developing from the start word "veg" (line 13). By lines 13, 15 and 16, GROGRA is told to transform this start word twice, yielding the short generative sequence

$$\mathsf{veg} \quad \longrightarrow \quad \mathsf{bud} \quad \longrightarrow \quad \mathsf{leaf}.$$

By the application of the interpretative rules in lines 18 and 19, the symbols "bud" and "leaf" are transformed into sequences of turtle commands specifying an explicit geometrical construction of a bud, resp. a leaf, with the help of line drawing (F), rotation (RU) and save-and-connect (S, C) commands (Fig. 5). Lengths and distances of leaf veins are read from the tables specified in lines 1–4, each leaf-half requiring its own set of parameters due to the slight asymmetry of beech leaves.

The rest of the growth grammar is devoted to branch growth. In line 14, a second start word, "shoot", is specified, and GROGRA is told to iterate rule application 8 times, beginning with this start word. In line 20, the development of the "shoot" begins with an initial length scaling (referring to a start length specified in line 8), with a diameter initialization ("D+1.2") and with the creation of an apical meristem, symbolized by "m". The "meristem" module, m(j, n), has two parameters: j switches between 1 and 2 (lines 21, 22) and specifies whether a bud (1) or a leaf (2) will be the visible production of m in the actual developmental step. n has the meaning of a *growth potential* or vigour (cf. Kurth, Früh, 1998) and specifies the number of internodes which will be produced from m in the next year (repetition operator in line 22).

For the sake of simplicity, this grammar was designed as a completely deterministic one (with the exception of a random deviation of the branching angle $x1$ from the mean value of 40 degrees, line 7). However, it would be possible to feed the internode number n with values from a random distribution (of binomial, negative binomial or other type) estimated from empirical observations at shoot samples, like in the AMAP model (Reffye et al., 1995). Instead, we let n fixed for the apical meristem at the tip of the shoot ("m(1, n)" at the end of line 22), whereas for lateral meristems, we include an acrotonic tendency by letting this parameter depend on the index i of the actual internode.

```
1  \var l1 table 8 9 11 14 17 16 14,    /* leaf description:          */
2  \var l2 table 40 44 44 40 31 20 9,   /* lengths inside leaf blade */
3  \var l3 table 10 11 12 15 17 15 13,
4  \var l4 table 24 35 38 35 27 17 7,
5  \var i index,
6  \angle 5,                 /* zigzag growth */
7  \var x1 normal 40 5,      /* branching angle */
8  \const bl 35,             /* length conversion factor internode/leaf */
9  \const minlf 15,          /* minimal leaf length */
10 \const cf 0.2,            /* coefficient in leaf length estimation */
11 \const secgr 1.6,         /* strength of secondary growth */
12 \var len length,
13 \axiom veg 1 2,           /* vegetative production: bud, leaf */
14 \axiom shoot 1–8,         /* half-annual steps of shoot development */
15 veg  #  bud,
16 bud  #  leaf,
17 leaf  # ,
18 bud ## P12 [ RU-90 F1 S(0) ] RU90 F1 S(1) RU-132 F3.8 S(2) f-0.8 RU88
          F4 S(3) f-0.8 RU-84 F4.7 S(4) RU52 F7.2 S(5) C(0,2) C(2,4)
          C(5,3) C(3,1),
19 leaf ## P2 D1 F12 D0 [ S(0) [ RU58 f(15) S(1) ] [ RU53 f(30) S(2) ]
          &(7) < F(l1(i)) [ RU(55-4*i) F(l2(i)) S(i+3) ] > F11 S(10) ]
          [ RU-45 f(15) S(18) ] &(7) < f(l3(i)) [ RU(4*i-55) F(l4(i)) S(17-i) ] >
          &(18) < C(i, i+1) > C(18, 0),
20 shoot # L(bl) D+1.2 m(1, 9),    /* growth starts with 9 internodes */
21 (j < 2)  m(j, n)  #  m(j+1, n),
22 (j=2 && n>1) m(j, n)  #  L*0.68
      &(n−1) < − a(0) RH180 − [ RU(x1) ang m(1, if(i<(n−1)/2,i/2,i−1)) ] L*1.1 >
      − L*0.5 a(0) RH180 − L*1.7 m(1, n),    /* main rule for shoot growth * /
23 (j=2 && n<=1) m(j, n)  #  L*0.4 a(0) L*2 m(1, 1),
24 ang  #  RU10,
25 a(t)  #  a(t+0.5),
26 a(t)  ##  Dℓ+(secgr * t) F,              /* internode * /
27 m(j, n)  ##  Lℓ(len/bl+1/(cf*len+1/minlf)) Dℓ1 O(veg, j),
```

Table 1: A non-sensitive growth grammar encoding vegetative branch development of European beech (*Fagus sylvatica* (L.)) with a rhythm of two steps per year. (See the Appendix for a short overview of the functionality of all used symbols and keywords.)

By the specification

$$\mathsf{if}(i < (n-1)/2,\ i/2,\ i-1))$$

in line 22, the dependency on i will not just be linear, but the suppression of lateral shoot length will be more pronounced in the lower half ($i < (n-1)/2$) of the annual shoot.

The essential object reference, establishing the link to the bud and leaf submodels, is specified in the interpretative rule for m in line 27. By the "Lℓ" command, a scaling of leaf size controlled by the actual internode length *len* used by the turtle takes place (avoiding pathologically small leaves, therefore not just linear, but respecting the minimal leaf length given in line 9). The length-width ratio of the leaves is not changed (therefore

"Dℓ1"). By "O(veg, j)" a reference to the bud submodel ($j = 1$) or to the leaf submodel ($j = 2$) is included. — The rest of the declarations and rules concerns branching geometry (lines 6, 7, 24) and secondary growth of the internodes (lines 11, 25, 26). Fig. 6 shows the evolution of the resulting branch structure during 5 consecutive semi-annual steps. The artificial branch could be used as a part of a whole-tree model, cf. the modular construction of the spruce growth grammar in Kurth, Früh (1998).

4.2. Tree and stand growth of Norway spruce (*Picea abies* (L.) Karst.)

Based on quantitative, empirical observations at several samples, a growth grammar for crowns of young spruce trees (age 5–15 years) was developed. A detailed description of the model will be published elsewhere (Kurth, Früh, 1998). The model is stochastic and consists of 35 declarations of variables or constants and 33 rules. The branch submodel includes quantifications of morphogenetic gradients (axis trend, acrotony), variations in shoot vigour, clustering of branching node positions along the annual shoot, changement of branching angle, differential cambial growth in the branching orders, and shoot mortality depending on length. Stem growth in height and diameter forms a further submodel. By RV commands, the gravitational impact on branch form is reproduced. (An earlier, similar attempt to simulate the architecture of spruce based on morphological data, which was, however, restricted to branches and did not use the grammar formalism, is given by the work of Pirochtová and Barták, 1991.)

Table 2 shows a grammar describing a sloping spruce stand. It has to be complemented by the grammar for the spruce trees itself from Kurth and Früh (1998), the evaluation of which begins with the symbol "plant" (line 18). The spruce trees become part of the stand in the form of object references "O(plant, 11)" in line 16. The size of the trees varies according to the command "L(*eps*(*epsplant*))" (cf. line 10, where *epsplant* is declared as random variable with normal distribution), but their shapes are exactly equal, since "plant" is evaluated and stored only once. Fig. 7 shows a view of the resulting sloping stand with 10-years old trees (the number 11 specified in line 11 and in the "O" command has to be reduced by 1 since the first step in the spruce grammar (not shown here) is an auxiliary step corresponding to no real annual growth).

For old stands, the assumption that all trees have the same shape is no longer realistic. Dominant trees have other crown ratios than suppressed ones. This can be respected in the model by using several start words for the trees, each corresponding to one tree class. By inserting the appropriate start word in the "O" command, a reference to a tree of a specified class is created. Fig. 9 shows the result of applying this technique to a part of the experimental plot F1 in the Solling (site and stand description in Ellenberg et al., 1986). The 42 spruce trees with an age of 112 years were classified into 3 classes (dominant, subdominant, suppressed), and for each class one "typical" tree was modelled (Fig. 8; Sloboda et al., 1996), using empirical height growth curves. In contrast to the example in Table 2, the positions of the trees are not assumed to be arranged on a rectangular grid here, but GROGRA reads them from a file. The tree coordinates in this file were obtained from an evaluation of remote sensing data. The heigth values in the data file also enable an automatic classification of the trees into the 3 classes. — Of course, mixed-species stands can be modelled using the same method, with object references distinguishing species instead of social classes (or with a combination of both).

```
1   \const rs 2000,       /* row spacing */
2   \const ls 2000,       /* line spacing */
3   \const bs 800,        /* border spacing */
4   \const soilh 1000,    /* heigth of soil above zero level */
5   \const nb_rows 4,
6   \const nb_lines 5,
7   \const slope_x 0.25,
8   \const slope_y 0.07,
9   \var epsilon normal 0 6000,      /* variance of planting position */
10  \var epsplant normal 0 0.015,  /* log-variance of tree size factor */
    ...                /* other declarations, concerning the spruce, follow */
11  \axiom plant 11,
12  \axiom plot 1 2,
13  plot  #  border RL-90 f(bs) RU90 f(bs)
        RU-90 RL90 f(soilh + slope_x * bs + slope_y * bs) E(2) < stand >,
14  border  #  [ P4 D0 [ F(soilh) S(1) ]
            RL-90 F(2*bs + (nb_rows−1)*rs)
            [ RL90 F(soilh + slope_x * (2*bs + (nb_rows−1)*rs)) S(2) ]
            RU90 F(2*bs + (nb_lines−1)*ls)
            [ RL90 F(soilh + slope_x * (2*bs + (nb_rows−1)*rs) +
                slope_y * (2*bs + (nb_lines−1)*ls)) S(3) ]
            RU90 F(2*bs + (nb_rows−1)*rs)
            [ RL90 F(soilh + slope_y * (2*bs + (nb_lines−1)*ls)) S(4) ]
            RU90 F(2*bs + (nb_lines−1)*ls)
            P7 C(1, 2) C(2, 3) C(3, 4) C(4, 1) ],
15  stand  #  &(nb_rows) < [ row ] RL-90 f(rs) RL90 f(slope_x * rs) >,
16  row  #  &(nb_lines) <
            [ disturb RH(ro) L(exp(epsplant)) Dℓ1 O(plant, 11) ]
            RU90 f(ls) RU-90 f(slope_y * ls) >,
17  disturb # RL-90 f(epsilon) RU90 f(epsilon) RU-90 RL90,
18  plant  #  ...        /* spruce system begins here */
```

Table 2: Part of a growth grammar describing a sloping coniferous stand. This grammar has to be complemented by a subsystem describing the growth of a single tree, beginning in line 18.

If only a quick overview of the stand structure without individual tree architecture is required, a schematical representation of the stand like in Fig. 10 can also easily be generated by GROGRA — only the set of rules specifying the tree construction has to be exchanged.

5. Discussion and Conclusions

The software GROGRA, which was used to interprete the grammars described above, offers several options to work with the generated artificial trees and stands:

- In combination with other, specialized software and with statistical analysis packages, a lot of analysis tools are at hand, ranging from pathlength analysis (McMahon, Kronauer, 1976) and determination of fractal dimension (Falconer, 1991) to the statistical analysis of shoot length and internode number distribution (cf. Reffye et al., 1995) and their comparison with intensively measured branches or trees (Kurth, Anzola, 1997), contributing to a better understanding of the botanical basis of crown development.

- The visualization by printable Postscript files or on the screen of a high-resolution graphics-end workstation — with possible growth animation in a special "movie modus" — offers possibilities for educational applications as well as for forest planning and landscape design. With the AMAP software, following a similar modelling philosophy, simulated tree stands in Japan with several thousands of trees have already been visualized (Lecoustre et al., 1997).

- Interfaces to process-oriented simulation software enable the use of GROGRA-generated virtual tree architectures as a structural basis for various other models designed to investigate the ecophysiology of single trees and tree stands: A discretization mechanism was included in GROGRA which makes the tree structures accessible for a radiation transport model (Knyazikhin et al., 1996) allowing the calculation of intercepted solar radiation and of photosynthesis. Another type of (re-)discretization was used for the flow model HYDRA (Früh, 1995) which simulates the transpiration-driven water movement in the branched tree architecture. Future extensions will combine the simulation of radiation and water flow (Lanwert et al., 1997), from which new insight in stomatal control and drought reaction of trees may result.

These different directions of work, and the partial results which are already obtained, cannot be exposed in more detail here. However, one objection should be mentioned which can often be met when high-resolution 3D simulations of trees are presented: It is the argument that these models have an "unreasonably high level of detail" and an unnecessary complexity, even more pronounced: that it is "sheer madness to construct every leaf of a tree". Indeed, for a human being it would be madness, but the computer is perfectly suited to do a mechanical, iterative work like creating thousands of virtual leaves. But what is more important: The underlying grammars are not really "very complex". The beech shoot example had only 13 rules, the rather elaborated grammar for young spruce consists of not more than 33 rules. We can conclude that the encoding of tree architecture in grammars provides us with a powerful method of "database amplification" which makes the seemingly over-complex structure of tree crowns manageable and can lead to botanical insights (on the scale of the single tree) as well as to an unprecedented visual realism in forest planning (on the stand scale).

But one objection will remain in the mind of the sceptical reader: All grammars presented so far were "non-sensitive", i.e. they allowed neither a dynamical interaction of tree growth with the environment, nor an influence of trees on each other while they grow. This is, of course, an earnest shortcoming since the shape of trees is strongly influenced by environmental factors and by competition, and therefore the sequel of this article (Kurth, Sloboda, 1998) will be devoted to sensitive grammars which include such interactions.

Acknowledgements

The authors wish to express their thanks to Dirk Lanwert for his technical skill concerning the XWindows interface of GROGRA and to Thomas Beisch and Silke Beleit (Institute for Yield Science, Göttingen) for their help in evaluating remote sensing data of the Solling research plot. The research reported here was funded by the DFG (German Research Foundation) in the frame of a postdoc grant for the first author, and by the BMBF (German Ministry of Research) in the frame of the Forest Ecosystem Research Centre, University of Göttingen. Facilities of the Faculty of Forest Sciences and Forest Ecology and of the GWDG (computation centre, Göttingen) were also essential. All support is gratefully acknowledged.

References

Abelson, H. – diSessa, A. A., 1982. Turtle Geometry. M.I.T. Press, Cambridge: 477 p.

Bell, A., 1976. Computerized vegetative mobility in rhizomatous plants. – In: Lindenmayer, A.; Rozenberg, G. (eds.): Automata, Languages, Development. Amsterdam: 3–14.

Bell, A. D., 1984. Dynamic morphology: A contribution to plant population ecology. – In: Dirzo, R.; Sarukhán, J. (eds.): Perspectives on plant population ecology. Sinauer, Sunderland, MA.: 48–65.

Bell, A. D. – Roberts, D. – Smith, A., 1979. Branching patterns: The simulation of plant architecture. Journal of Theoretical Biology, 81: 351–375.

Bloomenthal, J., 1985. Modeling the mighty maple. Computer Graphics (ACM/SIGGRAPH), 19: 305–311.

Borchert, R. – Slade, N. A., 1981. Bifurcation ratios and the adaptive geometry of trees. Botanical Gazette, 142: 394–401.

Borovikov, I. A., 1995. L-systems with inheritance: An object-oriented extension of L-systems. ACM SIGPLAN Notices, 30: 43–60.

Brinkhus, L. – Drewes, F. – Kreowski, H.-J. – Wirdemann, R. (Hg.), 1995. Benutzung des Bizarr2-Studios. Universität Bremen, Studiengang Informatik, Projekt-Abschlußbericht.

Chien, T. W. – Jürgensen, H., 1992. Parameterized L systems for modelling: Potential and limitations. – In: Rozenberg, G.; Salomaa, A. (eds.): Lindenmayer Systems. Springer, Berlin: 213–229.

Colin, F., 1992. Analyse et modélisation de la branchaison de l'épicéa commun en vue de predire la qualité de la ressource. Thèse, ENGREF, Nancy: 278 p.

D'Arcy Thompson, W., 1942. On growth and form. Cambridge University Press, Cambridge.

Deleuze, C. – Hervé, J.-C. – Colin, F. – Ribeyrolles, L., 1996. Modelling crown shape of Picea abies: Spacing effects. Can. J. For. Res., 26: 1957–1966.

Deussen, O. – Lintermann, B., 1997. A modelling method and user interface for creating plants. Graphics Interface '97 (May 1997), Postscript preprint: http://simsrv.cs.uni-magdeburg.de/~deussen/GI97.ps.gz.

Ellenberg, H. – Mayer, R. – Schauermann, J. (Hg.), 1986. Ökosystemforschung. Ergebnisse des Sollingprojekts 1966–1986. Ulmer, Stuttgart: 507 p.

Falconer, K. J., 1991. Dimensions — their determination and properties. – In: Bélair, J.; Dubuc, S. (eds.): Fractal Geometry and Analysis. Kluwer, Amsterdam: 221–254.

Fisher, J. B., 1984. Tree architecture: Relationships between structure and function. – In: White, R. A.; Dickison, W. C. (eds.): Contemporary Problems in Plant Anatomy. Academic Press, Orlando: 541–589.

Fisher, J. B. – Honda, H., 1979. Branch geometry and effective leaf area: a study of Terminalia-branching pattern. 1. Theoretical trees. Amer. J. Botany, 66: 633–644.

Ford, E. D. – Avery, A. – Ford, R., 1990. Simulation of branch growth in the *Pinaceae*: Interactions of morphology, phenology, foliage productivity, and the requirement for structural support, on the export of carbon. J. Theor. Biol., 146: 15–36.

Früh, Th., 1995. Entwicklung eines Simulationsmodells zur Untersuchung des Wasserflusses in der verzweigten Baumarchitektur. Berichte des Forschungszentrums Waldökosysteme Göttingen, Ser. A, 131: 192 p.

Goel, N. S. – Rozehnal, I., 1992. A high-level language for L-systems and its applications. – In: Rozenberg, G.; Salomaa, A. (eds.): Lindenmayer Systems. Springer, Berlin: 231–251.

Goel, N. S. – Knox, L. B. – Norman, J. M., 1991. From artificial life to real life: Computer simulation of plant growth. International Journal of General Systems, 18: 291–319.

Govaerts, Y. M., 1996. A model of light scattering in three-dimensional plant canopies: A Monte Carlo ray tracing approach. Ph. D. Dissertation, Université Catholique de Louvain (Belgium), Joint Research Centre of the European Commission, Doc. EUR16394EN: 165 p.

Greene, N., 1989. Voxel space automata: Modeling with stochastic growth processes in voxel space. Computer Graphics (ACM/SIGGRAPH), 23: 175–184.

Gruber, F., 1987. Das Verzweigungssystem und der Nadelfall der Fichte (*Picea abies* (L.) Karst.) als Grundlage zur Beurteilung von Waldschäden. Ber. Forschungszentr. Waldökosys./Waldsterben Göttingen, Ser. A, 26: 264 p.

Hallé, F. – Oldeman, R. A. A., 1970. Essai sur l'architecture et la dynamique de croissance des arbres tropicaux. Masson & Cie, Paris.

Hallé, F. – Oldeman, R. A. A. – Tomlinson, P. B., 1978. Tropical trees and forests — an architectural analysis. Springer, Berlin.

Hauhs, M. – Kastner-Maresch, A. – Rost-Siebert, K., 1995. A model relating forest growth to ecosystem-scale budgets of energy and nutrients. Ecological Modelling, 83: 229–243.

Jacob, C., 1997. Principia Evolvica. dpunkt Verlag, Heidelberg.

Kaitaniemi, P. – Honkanen, T., 1996. Simulating source-sink control of carbon and nutrient translocation in a modular plant. Ecological Modelling, 88: 227–240.

Kari, L. – Rozenberg, G. – Salomaa, A., 1997. L systems. – In: Rozenberg, G.; Salomaa, A. (eds.): Handbook of Formal Languages, Vol. 1. Springer, Berlin: 253–328.

Kawaguchi, Y., 1982. A morphological study of the form of nature. Computer Graphics (ACM/SIGGRAPH), 16: 223–232.

Kellomäki, S. – Kurttio, O., 1991. A model for the structural development of a Scots pine crown based on modular growth. For. Ecol. Manag., 43: 103–123.

Kellomäki, S. – Strandman, H., 1995. A model for the structural growth of young Scots pine crowns based on light interception by shoots. Ecological Modelling, 80: 237–250.

Kim, J. T. T., 1996. Untersuchungen zur Evolution von morphologischer und taxonomischer Diversität und Komplexität anhand von Computermodellen. Dissertation, Universität Köln, Math.-naturwiss. Fak.

Knyazikhin, Yu. – Kranigk, J. – Miessen, G. – Panfyorov, O. – Vygodskaya, N. – Gravenhorst, G., 1996. Modelling three-dimensional distribution of photosynthetically active radiation in sloping coniferous stands. Biomass and Bioenergy, 11: 189–200.

Küppers, M., 1989. Ecological significance of above-ground architectural patterns in woody plants: A question of cost-benefit relationships. Trends in Ecology and Evolution, 4: 375–379.

Kurth, W., 1994. Morphological models of plant growth: Possibilities and ecological relevance. Ecological Modelling, 75/76: 299–308.

Kurth, W., 1994b. Growth Grammar Interpreter GROGRA 2.4. A software tool for the 3-dimensional interpretation of stochastic, sensitive growth grammars in the context of plant modelling. Introduction and Reference Manual. Ber. Forschungszentr. Waldökosys. Göttingen, Ser. B, 38: 192 p.

Kurth, W., 1996. Elemente einer Regelsprache zur dreidimensionalen Modellierung des Triebwachstums von Laubbäumen. – In: Hempel, G. (Hg.): 8. Tagung Sekt. Forstl. Biometrie u. Informatik DVFF (Tharandt / Grillenburg, 25.–28. 9. 1995), Ljubljana: 174–187.

Kurth, W., 1998. Some new formalisms for modelling the interactions between plant architecture, competition and carbon allocation. Bayreuther Forum Ökologie, 52: 53–97.

Kurth, W. – Anzola, G., 1997. Triebwachstum und Verzweigung junger Fichten in Abhängigkeit von den beiden Einflußgrößen "Beschattung" und "Wuchsdichte": Datenaufbereitung und -analyse mit GROGRA. – In: Pelz, D. (Hg.): 10. Tagung Sekt. Forstl. Biometrie u. Informatik DVFF (Freiburg i. Br., 24.–26. 9. 1997), Ljubljana: 89–108.

Kurth, W. – Früh, Th., 1998. Modelling crown structure in conifers: A stochastic growth grammar for the developing branching system of spruce. (Submitted)

Kurth, W. – Sloboda, B., 1998. Tree and stand architecture and growth described by formal grammars. II. Sensitive trees and competition. Lesnictví-Forestry (to appear).

Lanwert, D. – Dauzat, J. – Früh, Th., 1997. Water use of forest trees: About a possibility of combining structures and physical process models. Bayreuther Forum Ökologie, 52: 117–128.

Lecoustre, R. – Saito, K. – Kumagai, Y. – Honjo, T. – Reffye, Ph. de – Ishida, Y., 1997. Simulation d'un paysage forestier à partir d'un SIG et de la technique de modélisation des plantes. – In: Bouchon, J.; Reffye, Ph. de; Barthélémy, D. (eds.): Modélisation et simulation de l'architecture des végétaux. INRA, Paris: 425–434.

Lindenmayer, A., 1968. Mathematical models for cellular interactions in development. J. Theor. Biol., 18: 280–299, 300–315.

List, R. – Küppers, M. – Schröder, F., 1994. Madeira — ein Programm zur Simulation des Wachstums (Biomasse und Architektur) von Holzgewächsen. Verhandlungen der Gesellschaft für Ökologie, 23: 39–48.

Maillette, L., 1982. Structural dynamics of silver birch. I. The fates of buds. Journal of Applied Ecology, 19: 203–218.

McMahon, Th. A. – Kronauer, R. E., 1976. Tree structures: Deducing the principle of mechanical design. J. Theor. Biol., 59: 443–466.

Niklas, K. J., 1982. Computer simulations of early land plant branching morphologies: Canalization of patterns during evolution? Paleobiology, 8: 196–210.

Niklas, K. J., 1986. Computer simulations of branching patterns and their implications on the evolution of plants. Lect. Math. Life Sci., 18: 1–50.

Oborny, B., 1994. Growth rules in clonal plants and environmental predictability — a simulation study. J. Ecology, 82: 341–351.

Perttunen, J. – Sievänen, R. – Nikinmaa, E. – Salminen, H. – Saarenmaa, H. – Väkevä, J., 1996. LIGNUM: A tree model based on simple structural units. Annals of Botany, 77: 87–98.

Pfreundt, J. – Sloboda, B., 1996. The relation of local stand structure to photosynthetic capacity in a spruce stand: A model calculation. Lesnictví-Forestry, 42: 149–160.

Pirochtová, M. – Barták, M., 1991. Formalizace architektury koruny smrku ztepilého (Picea abies (L.) Karst.): Model větvení vyšších řádů. Lesnictví-Forestry, 37: 233–243.

Prusinkiewicz, P. – Lindenmayer, A., 1990. The Algorithmic Beauty of Plants. Springer, New York: 228 p.

Prusinkiewicz, P. – Hammel, M. – Hanan, J. – Měch, R., 1997. Visual models of plant development. – In: Rozenberg, G.; Salomaa, A. (eds.): Handbook of Formal Languages, Vol. 3. Springer, Berlin: 535–597.

Reeves, W. T. – Blau, R., 1985. Approximate and probabilistic algorithms for shading and rendering structural particle systems. Computer Graphics (ACM/SIGGRAPH), 19: 313–322.

Reffye, Ph. de – Snoeck, J., 1976. Modèle mathématique de base pour l'étude et la simulation de la croissance et de l'architecture du Coffea robusta. Café Cacao Thé, 20: 11–32.

Reffye, Ph. de – Houllier, F. – Blaise, F. – Barthélémy, D. – Dauzat, J. – Auclair, D., 1995. A model simulating above- and below-ground tree architecture with agroforestry applications. Agroforestry Systems, 30: 175–197.

Roloff, A., 1989. Kronenentwicklung und Vitalitätsbeurteilung ausgewählter Baumarten der gemäßigten Breiten. J. D. Sauerländer, Frankfurt.

Roloff, A., 1994. Effects of pollutants and environmental stress on the morphology of deciduous trees and on the ground vegetation. – In: Godbold, D. L.; Hüttermann, A. (eds.): Effects of acid rain on forest processes. Wiley-Liss, New York: 325–351.

Room, P. M. – Maillette, L. – Hanan, J. S., 1994. Module and metamer dynamics and virtual plants. Adv. Ecol. Res., 25: 105–157.

Sattler, R., 1992. Process morphology: Structural dynamics in development and evolution. Canad. J. Botany, 70: 708–714.

Schill, H., 1989. Triebbildung, Verzweigungsverhalten und Kronenentwicklung junger Fichten und Lärchen. Birkhäuser, Basel: 157 p.

Sievänen, R. – Mäkelä, A. – Nikinmaa, E. – Korpilahti, E., 1997. Silva Fennica special issue on functional-structural tree models, Preface. Silva Fennica, 31: 237–238.

Sloboda, B. – Kurth, W. – Früh, Th., 1996. Morphologisches Modell des Baumwachstums. – In: Beese, F. (Hg.): Verbundprojekt "Veränderungsdynamik von Waldökosystemen", Zwischenbericht 1994/1995, Teil II. Ber. Forschungszentr. Waldökosys. Göttingen, Ser. B, 52: 185–193.

Šoch, M. – Beneš, B., 1995. Simulation of plant development using strands model. – In: T. Plachetka (ed.): Proc. 11th spring school on computer graphics, Bratislava, May 1995: AP.30–AP.35.

Stevens, P. S., 1974. Patterns in nature. Little, Brown and Co., Boston.

Stoll, P., 1995. Modular growth and foraging strategies in rhizome systems of *Solidago altissima* L. and branches of *Pinus sylvestris* L. Dissertation, Universität Zürich: 97 p.

Tsel'niker, Yu. L., 1994. Structure of spruce crown. Russian For. Sci., 1994-4: 29–37.

Viennot, X. G. – Eyrolles, G. – Janey, N. – Arquès, D., 1989. Combinatorial analysis of ramified patterns and computer imagery of trees. Computer Graphics (ACM/SIGGRAPH), 23: 31–40.

Waller, D. M. – Steingraeber, D. A., 1983. Branching and modular growth: Theoretical models and empirical patterns. – In: Buss, L.; Jackson, J. (eds.): Symposium on the Biology of Clonal Organisms. Yale University Press, Yale: 85–108.

White, J., 1979. The plant as a metapopulation. Annual Review of Ecology and Systematics, 10: 109–145.

Appendix: Overview of used grammar symbolism

Keywords	
\angle	sets the angle α for subsequent $+$ and $-$ operations
\ask I1	asks a value for global register 1 from the user
\axiom	sets a start word
\const	declaration of a constant
function	associates a variable with one of several predefined functions
if	in arithmetical expressions: if(*condition*, *v1*, *v2*) gives value *v1*, if *condition* is true, and *v2* otherwise.
index	declares a variable to act as a counter in repeated parts, counting from 0 to $n-1$, inside $\&(n) < \ldots >$
length	declares a variable to stand for the actual length
normal	declaration of a normally distributed (Gaussian) random variable
register	declaration of a global register variable
\set V	sets the strength of vertical tendency for subsequent *RV*
table	declaration of a table variable with predefined, subsequent values
uniform	declaration of a uniformly distributed random variable
\var	keyword preceding each declaration of a variable
zcoordinate	declares a variable to stand for the actual height
Symbols	
#	separates left and right side of a generative rule
##	separates left and right side of an interpretative rule
[ ]	branch delimiters
$+$	rotation, equivalent to $\mathsf{RU}(\alpha)$
$-$	rotation, equivalent to $\mathsf{RU}(-\alpha)$
@	relative positioning; @(x) is equivalent to $\mathsf{f}((x-1)*$ *length*$)$
%	cut operator
$*$	default start word
&	repetition operator
&&	logical "and"
$<\ >$	delimiters for part of string subject to "&" or "E"
/* */	delimiters for comments
In arithmetical, resp. logical expressions, the usual operators $+$, $-$, $*$, /, $<$, $>$, $<=$, $>=$, $=$ and != ("not equal") are allowed.	
Letter combinations with fixed meaning	
C	connection operator (with two arguments; cf. "S")
D	sets the diameter for subsequent F commands
Dℓ	sets the local diameter, affecting only the next F
D+	increments the diameter
Dℓ+	increments the diameter locally
E	expansion operator
F	movement and shoot generation
F$*$	like F, but with length multiplied by a given factor
f	movement without shoot generation
L	sets the length for subsequent F and f commands
Lℓ	sets the local length, affecting only the next F
L$*$	multiplies the length with a given factor
N	sets the foliage parameter for subsequent F commands
O	inclusion of a shared object: $\mathsf{O}(w, n)$ refers to the result of the nth developmental step obtained from start word w
P	sets the colour for subsequent F commands
Pℓ	sets the local colour, affecting only the next F
RH	rotation around the H (head) axis
RL	rotation around the L (left) axis
RU	rotation around the U (up) axis
RV	rotation downward, according to current vertical tendency
S	saves current position for later connection by C

Figure captions

Fig. 1: The three levels of formal description of plant form.
(1) Statical description, (2) development, (3) control of development.

Fig. 2: Structure generation by a simple growth grammar (L-system with interpretation).

Fig. 3: Structure generation from the growth grammar
$a \longrightarrow \mathsf{F}\ [\ \mathsf{RU45}\ b\ ]\ a, \quad b \longrightarrow \mathsf{F}\ b.$

Fig. 4: Structure generation by a two-phase growth grammar.

Fig. 5: Result of the submodels "bud" and "leaf" of the beech shoot growth grammar in Table 1, lines 1–5, 13 and 15–19.

Fig. 6: Developmental steps 3–7 of the virtual beech branch specified by the growth grammar from Table 1, generated with GROGRA.

Fig. 7: Sloping spruce stand, generated by GROGRA from the grammar of Table 2 together with the spruce grammar from Kurth and Früh (1998), corresponding to age 10.

Fig. 8: 112 years-old spruce trees of 3 different social classes, generated by GROGRA.

Fig. 9: Part of the experimental spruce stand Solling F1, generated by GROGRA from a growth grammar for aged spruce trees of 3 different social classes (cf. Fig. 8), together with a list of tree coordinates and heights.

Fig. 10: The same spruce stand as in Fig. 8. Here, a simplified tree description was inserted into the growth grammar instead of the detailed tree model of Fig. 8.