Developing Multiscale Simulation Models using the Software GroIMP

Yongzhi Ong, Winfried Kurth

Department Ecoinformatics, Biometrics & Forest Growth
University of Göttingen
Büsgenweg 4, 37077 Göttingen, Germany
yong@gwdg.de

Abstract:
GroIMP (Growth grammar based Interactive Modelling Platform) is an open-source software tool focused on the development of functional-structural plant and forest stand models. It encompasses a domain-specific programming language, Extended L-System language (XL), which provides standard Java language features and additional rule-based graph rewriting mechanisms. A model with meaningful 3D representation and visualization, such as a forest plot, can be constructed by interpreting each project graph geometrically. Additionally, embedded imperative code allows to run process-based simulations (e.g., light interception) on these 3D structures.
Recent extensions to the graph model and XL have allowed convenient representations of hierarchical or multi-scale objects in GroIMP. Using these features, models at different spatial scales are coupled in a common data structure, raising further interests towards specific methods in up- and down-scaling model dynamics.
In this paper, we show the graph- and rule-based programming approach as a viable method to developing multi-scale functional-structural models. Examples include stand dynamics and morphological developments of individual conifers as well as reactions to ozone exposure of beech trees at the metabolic, organ and whole-tree scale. Distinct scales can be visualized and observed for analysis and comparisons.

1 Introduction

1.1 Rewriting Systems

Architecture and structure play an important role in statistical and functional models of individual plants (Godin et al. 1999). The L-systems (Lindenmayer 1968), initially used to model development of multicellular organisms, were later extended to model these physical aspects of plants (Smith 1984; Prusinkiewicz 1986). Variants of L-systems emerged, catering to increasing requirements of structural plant models. Parametric L-systems (Hanan 1992), for example, allow the association of numerical parameters with L-system symbols, enabling the quantification of geometric attributes of a model. Other examples of L-system variants include stochastic L-systems (Prusinkiewicz & Lindenmayer 1990), context-sensitive L-systems, table L-systems (Kari et al. 1997), pseudo L-systems (Prusinkiewicz 1986), differential L-systems (Prusinkiewicz et al. 1993), sensitive growth grammars (Kurth 1999; Kurth & Sloboda 1999), etc. The concision or brevity of rewriting rules is one reason that L-systems remain one of the dominant formalisms for plant modelling today (Hanan 2013).

Along the same line of research, development of several software tools with 3-dimensional interpretation of (extended) L-systems, e.g., L-Studio (Karwowski & Prusinkiewicz 2004), GroIMP (Kniemeyer 2008), and OpenAlea/LPy (Boudon et al. 2012) aim to mitigate the process of creating realistic virtual plant architectures and models.

1.2 Stand and Tree Modelling

At a coarser scale, i.e. higher level in the organizational hierarchy, models of tree and stand development are required for assessment and management of forests (Burkhart & Tomé 2012). Different approaches, such as empirical, process-based, and hybrid, have been taken to construct these models. While empirical models, e.g. conventional growth and yield models, essentially describe observed data at a given scale (e.g. tree or stand) in terms of attributes at the same scale, empiricism for process-based models is usually at finer scales than individual trees or stands (Thornley & Johnson 1990). Process-based models deal, primarily, with dry mass production processes at the leaf or canopy scale and its allocation to plant parts, resulting in tree or stand development. A combination of process-based and empirical elements at the same scale gives a hybrid model (Mäkela et al. 2000).

Many approaches to empirical modelling of forest stand development exist. One way to classify the approaches is by the modelling entity (i.e. stand, size class, tree) utilized (cf. table 10.1, page 235, Burkhart & Tomé 2012). A separate, commonly-used approach involves the disaggregation of whole-stand models by a continuous dbh distribution function, as opposed to the discrete disaggregation to size classes in size class models.

In empirical growth and yield models, field measurements of variables such as age, diameter at breast height (dbh), height, site quality are taken and fitted with regression models that allow reproduction of the characteristics in the data set. With more available data, e.g. whole stand, diameter distribution, size-class, and individual tree models, improved yield models using advanced analytical methods and computing technology have been developed (Burkhart & Tomé 2012). In later models, crown and branching traits have been included (Mohren & Burkhart 1994). A variety of approaches have been used to quantify tree crowns. Straightforward approaches use crown length and width to estimate crown surface area and volume, assuming that the crown is some regular geometric shape. Alternatively, regression equations have been used to estimate crown volume from dbh, height, and live-crown ratio (Biging & Wensel 1990). Later research resulted in crown profile models with finer detail by dividing each crown into segments (Hann 1999; Marshall et al. 2003). Other techniques for modelling tree crowns include stochastic frontier models (Nepal et al. 1996), non-parametric models (Doruska & Mays 1998), and use of fractal geometry (Zeide & Gresham 1991; Zeide & Pfeifer 1991). A comprehensive list of approaches for describing crown profiles can be found in Pretzsch (2009) and Burkhart & Tomé (2012).

In order to derive the dynamics of stand structure, competition, and interaction of trees with other organisms from simple rules acting at a local scale, individual-based models have been developed (Grimm & Railsback 2005). It has been demonstrated that the above-mentioned rewriting-system approach is also applicable for a concise specification of such models at stand scale (Kurth 2002, Kurth et al. 2012).

1.3 Multiple Scales

Apparent in various domains of research, attempts have been made to integrate scales in plant, tree, stand models, etc. In computer graphics, Deussen et al. (1998) and Lane & Prusinkiewicz (2002) generated plant ecosystems for visualization. In process-based stand and tree modelling, Mäkela (2003) proposed a hierarchical treatment of multiscale processes. Seidl et al. (2012) presented a hierarchical multiscale framework in ecology that constructs ecosystems at landscape level from the scale of individual trees. In functional-structural plant modelling, Godin and Caraglio (1998) proposed the multiscale tree graph in order to formalize the hierarchical representation of individual plant structures. Cournède et al. (2008) have combined a competition model based on point patterns in a plane with an individual-tree FSPM to simulate 3D forest structures. By taking into account individual 3D tree structures and competition in a mixed stand, Lintunen et al. (2011) have successfully simulated the crown structures of trees. Last but not least, scale-integration, otherwise known as *linkages* in forest stand modelling (cf. chapter 10.2, Burkhart & Tomé 2012), is commonly used for efficiency and consistency in providing different levels of detail on stand structure.

Continued efforts to integrate scales in models saw the coupling of L-system-based models with objects not easily accessible from the rule-based realm, giving rise to lengthy model definitions, contradicting the original advantage of concise rule-based formalisms. To illustrate this problem, consider a plant organ modelled as a symbol in an L-system rule and an object representing a forest stand that is not recognized in rules. When computations require an interface between the two, either (index-based) referencing or lengthy graph queries need to be engaged. This problem was addressed separately by Boudon et al. (2012) using strings that encode multiscale structures in rules, and by Ong & Kurth (2012) using multiscale graph grammars.

1.4 Multiscale Programming with Rules and a Unified Data Structure

In this paper, we present the applicability of multiscale graph grammars (Ong & Kurth 2012) in the context of its implementation – the programming language XL and the software platform GroIMP (Kniemeyer 2008). We showcase the usage of rule-based programming on a single unified data structure for multiscale models, as an alternative to coupling scattered software packages. Drawing from the wide range of stand and tree models in the aforementioned categories, the use of rules is not only

demonstrated for the structural growth of individual trees, but also for a selected empirical stand model and crown profile model.

In section 2, we revisit main concepts of earlier work in multiscale graph grammar and show the graph construct for containing atmospheric effects on metabolic networks. Section 3 describes, using an example, the step-by-step development of a multiscale model using the programming language XL in the software GroIMP. Lastly, in section 4, we summarize and discuss upcoming and potential future work.

2 Previous Work

2.1 Multiscale Graph Model and Grammar

For the purpose of embedding and accessing objects at different scales via rules in a unified data structure, a three-part graph model and grammar was proposed (Ong & Kurth 2012). The first part, a structure-of-scales, is a finite partially ordered set that establishes the scales and the ordering of refinement relationships between them. A scale is either a refinement of, an encoarsement of, or incomparable with another scale. These inter-scale relationships are (super)imposed on the second part, a type graph. The type graph is a labelled, directed graph that consists of nodes, each representing an entity type in the model. Refinement relationships are represented by edges labelled by the unique *refinement edge* label. The third part is an instance graph that comprises node or object instances (in terms of object-oriented programming) corresponding to entity types in the type graph. Figure 1 depicts the three-part multiscale graph model. The structure-of-scales, although a fundamental component of the graph model, is exempted from source code. In practice, it can be deduced from the type graph. Typically, a rule in the initial procedure – init() of projects in GroIMP is used to specify the type graph. Other rule statements leverage on the type graph for semantic interpretation of intended graph queries (on the left-hand side) and productions (on the right-hand side).

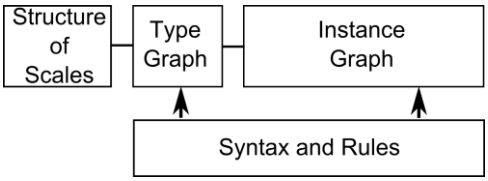

Figure 1. Three-part graph model and grammar. Refinement relationships are superimposed from the structure-of-scales to the type graph, and from the type graph to the instance graph. Syntax interpretation relies on the type graph and rules operate on the instance graph.

2.2 Example: Rules for Atmospheric Gas, Trees, and Biological Networks

The three-part graph model was used to represent tropospheric ozone conditions, European beech (*Fagus sylvatica*) trees (whole tree, axes, growth units, organs,

crown layers), and the Shikimate pathway network (Ong et al. 2014, submitted). Figure 2 illustrates the structure-of-scales and type graph used.

Some rules used in the model highlight the advantages of this three-part graph data structure. For example, to look up current ozone AOT40 (accumulated ozone exposure over a threshold of 40 parts per billion) conditions in order to initialize enzyme concentrations of the Shikimate pathway networks in the instance graph, we used the query (left-hand side of rule)

```
s:Stand (/>)* CrownLayer [dahps:DAHPS][dhqd:DHQD][sd:SD][epsps:EPSPS]
```

In this query, `(/>)*` represents a path of refinement edges with an arbitrary length from the `Stand` node to the `CrownLayer` node. The brackets between the `CrownLayer` node and nodes representing metabolic species represent refinement edges instead of branching edges in conventional bracketed L-systems. This interpretation occurs as a result of the refinement relationships in the type graph. In a single query, the ozone conditions stored in `Stand` nodes and the related nodes for metabolic species (e.g. `DAHPS`) are found in the instance graph. The colon-separated, lower-case prefixes (`s`, `dahps`, ...) of the node type names are labels of individual instances, used to retrieve them in subsequent code (not shown).

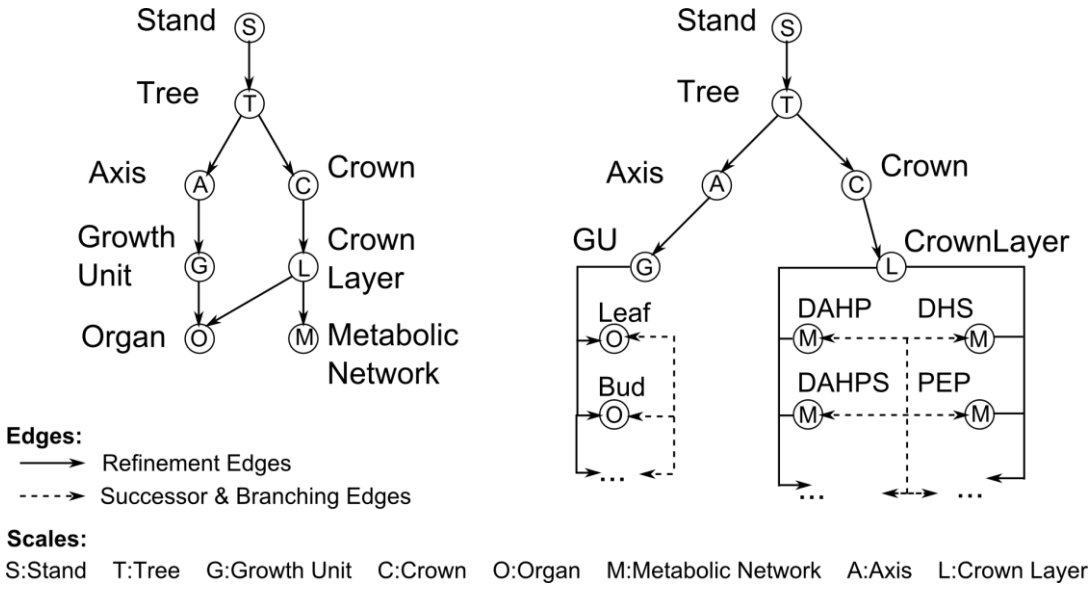

Figure 2. Structure-of-scales and type graph of model simulating ozone trends, beech trees and the Shikimate pathway.

On the other hand, conventional successor and branching edge relationships are expected for two consecutive nodes with entity types at the same scale. To illustrate this, consider the rule

```
t:Tree a:Axis g:GU Bud ==> t a g
for(1:numInternode) (Internode [Axis GU Bud]) GU Bud;
```

that specifies the structural development of beech trees at various scales (tree, axes, growth units, and internodes). The left-hand side of the rule, i.e. `t:Tree a:Axis g:GU`

`Bud`, is replaced by the production graph specified after the symbol `==>`. A branching edge, for example, is expected between `Internode` and the `Bud` node within brackets although they are separated by coarser scale nodes, `Axis` and `GU`.

In addition, the type graph is used to determine edge connections necessary to embed production graphs in the instance graph during graph rewriting operations (cf. Ong & Kurth 2012 for details of the rewriting mechanisms).

3 Multiscale Model Development

This section describes the development of a multiscale model that consists of stand dynamics, crown profile development, height and dbh growth, and branching structure development of individual trees. Emphasis is placed on the rule-based (graph) data structure-oriented programming approach, and the integration of quantitative values at different scales for a consistent multiscale model. For this paper, empirical models are adopted from literature and combined. The rest of this section is divided into sub-sections describing the multiscale graph structure, model initialization, the germination model, the growth model, and the mortality model.

3.1 Multiscale Graph Structure and Model Initialization

An overview of the scales and entity types required in the model is first constructed using the structure-of-scales and type graph (Figure 3).

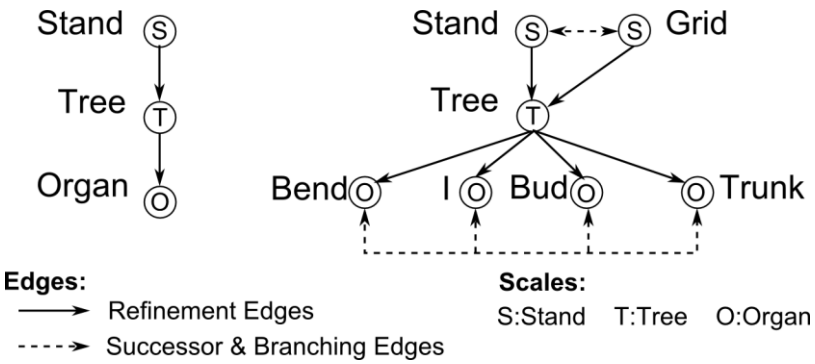

Figure 3. Graph construct for model simulating stand dynamics and developments of crown profile, height, dbh, and branching structure of virtual fir trees. Left: structure-of-scales. Right: type graph.

Three scales make up the model. The Stand scale is the coarsest scale and comprises the entity types *Stand*, which represents a collective object for a forest stand, and *Grid*, which represents a unit of the regularly divided stand area. The Tree scale comprises an entity type *Tree* that represents a single fir tree. The finest scale Organ has entity types *Bend*, *I*, *Bud*, and *Trunk*. *Bud* represents a bud, *I* represents an internode, *Trunk* represents an internode along the trunk or main stem, and *Bend* represents a connection between a lateral branch and the trunk. The entity types, otherwise known as *modules* in the programming language XL, are declared in code.

In the initial procedure `init()`, a forest stand with an evenly divided grid floor is created:

```
{grids = createGridFloor();}//grids is a double array of Grid objects
Axiom ==> Stand for(int i:(1: S_GRID_COUNT_X))
            (for(int j:(1: S_GRID_COUNT_Y))([grids[i][j]]));
```

The procedure `createGridFloor()` initializes a double array (rows and columns) of *Grid* nodes. To initialize the instance graph, a rule with query for `Axiom` creates a *Stand* node that is connected to each *Grid* node via a branching edge. The `init()` procedure additionally contains the rule for creating the type graph, as illustrated in Figure 3, and connecting it to the root node `^`:

```
==>> ^ {# Stand Grid #} /> Tree /> {# Bend I Trunk Bud #};
```

The symbols `{#` and `#}` enclose nodes in a clique, connecting each pair within bi-directionally by successor and branching edges.

Upon completion of `init()`, a procedure `run()` is invoked once for every simulation year. Three major simulation sub-steps, described by the next three sub-sections, are included in `run()`.

3.2 Germination

A constant pool of germinating seeds is assumed to reside in the virtual forest stand each year. The procedure for the first sub-step of `run()`, `germinate()`, contains the following rule that appends graph nodes representing seedlings to the instance graph:

```
s:Stand ==> s for(int i:(1:S_SEED_POOL))(
      {x = random(0,S_DIM_X); y = random(0,S_DIM_Y);}
      [createTree(x,y) createTrunk Bud(0)]);
```

In the second line of the rule, a Java code block (between curly brackets) is embedded to generate random 2-d coordinates for the position of a new tree. `createTree(x,y)` and `createTrunk` are procedures that return new graph nodes with entity types *Tree* and *Trunk* respectively. All in all, a total of `S_SEED_POOL` nodes with entity type *Tree* are newly refined from each *Stand* node. Each new *Tree* node is refined to a new *Trunk* node succeeded by a new *Bud* node.

Internally, the procedure `createTree(x,y)` establishes a refinement edge from a *Grid* node to the new *Tree* node it creates based on the position given by `x` and `y`:

```
{Tree t = new Tree(x,y,T_INIT_HT,T_INIT_CI,T_INIT_DBH,T_INIT_AGE,
T_CR,treeCwmax(0),T_INIT_HCBASE,T_INIT_HD,T_INIT_NUMINT,T_CROWN_SHADE
T_RPMAX);}
==>> grids[x/S_DIM_X][y/S_DIM_Y] t <+ ^;
```

The *Grid* node representing the grid unit in which the tree resides is identified from the double array using a simple division of the coordinates by the grid dimensions, `S_DIM_X` and `S_DIM_Y`. The graph root node `^` is connected to the new *Tree* node via a

branching edge for visualization purposes, since our tool GroIMP currently shows in its 3D view only nodes accessible from the root by a path composed exclusively of successor and branching edges. Parameters for a new *Tree* node are mostly constants except for maximum crown width, which is computed by the procedure `treeCwmax`. Empirical data for maximum crown width of fir trees was extracted from Schober (1995). The data was fitted and computed as

$$cw = 8.331566 / (1 + 5.011992 * age^{-1.011805})^2 + 0.194259$$

where *cw* is the maximum crown width for a tree with age *age*.

Similar to `createTree`, `createTrunk` consists of a rule to connect the graph's root node to new *Trunk* nodes.

3.3 Growth

The second sub-step of `run()`, `grow()`, contains procedures and rules to compute competition among trees, growth of individual trees, and development of organs and finer structures for each tree.

3.3.1 Stand Competition Index

An empirical model of forest stand development based on Burkhart et al. (1987) is utilized. Dbh of trees in each grid unit is reset and summed using the rules:

```
g:Grid ::> {g.dbhSum = 0; g.dbhSumInRange = 0;}
g:Grid t:Tree ::> {g.dbhSum += t.dbh;}
```

The first rule resets the value of parameters `dbhSum` and `dbhSumInRange` for each *Grid* node. The second rule aggregates the dbh of each tree, `t.dbh`, residing in each grid unit into `g.dbhSum`.

The effects of competition between trees occur within a specified distance range `S_COMPETE_RANGE`, which is resolved into a number of grid units depending on the dimensions of a grid unit. The sum of dbh in grid units within the range `S_COMPETE_RANGE`, `dbhSumInRange`, is aggregated for each *Grid* node using conventional Java iterations on the double array `grids`. Correction is performed for grid units along the marginal areas of the virtual stand by compensating for out-of-range areas using the average `dbhSum` of within-range grid units.

With the accumulation of dbh in grid units, competition indices of trees are set using the following rule:

```
g:Grid t:Tree ::> {t.ci = g.dbhSumInRange/t.dbh;}
```

where `t.ci` is the competition index for tree `t` residing in the grid unit represented by *Grid* node `g`.

3.3.2 Individual Tree Growth

Empirical data for height and dbh of fir trees is extracted from Schober (1995). The data was fitted and computation of height and dbh are as follows:

$$ht = 0.023777 / (0.000559 + age^{-1.896382}) + 0.400637$$

$$dbh = 0.225899 * age^{1.142437 + (-0.000378 * age)} + 1.012812$$

where *ht*, *dbh*, and *age* are the height, dbh, and age of a fir tree respectively. Given height, dbh, and age, crown ratio (vertical proportion of height that the crown occupies) is computed based on the model by Dyer & Burkhart (1987):

$$cr = 1 - e^{-(0.55243 + 5.026/age) * dbh / ht}$$

where *cr* is the crown ratio of the tree.

A rule in `grow()` utilizes the empirical fittings and pre-computed competition indices to determine the development of each tree:

```
t:Tree ::> {
    float htPotDelta = treeHt(t.age + 1) - t.ht; //poten. ht inc.
    t.hd = cDelta(htPotDelta, t.ci, t.cr); //ht inc. w. competition
    t.ht += t.hd;
    float dbhPotDelta = treeDbh(t.age + 1) - t.dbh; //poten. dbh inc.
    t.dbh += cDelta(dbhPotDelta, t.ci, t.cr);//dbh inc. w. competi.
    t.age ++; //tree age
    t.cr = crownRatio(t.dbh,t.ht,t.age); //crown ratio
    t.cwmax = treeCwmax(t.age); //maximum crown width
    t.hcbase = (1-t.cr) * t.ht; //height to base of crown
    }
```

The procedures `treeHt`, `treeDbh`, and `crownRatio` contain the aforementioned empirically fitted formulas for height, dbh, and crown ratio of fir trees. Given the succeeding age of a tree (`t.age + 1`) , the potential height increment, `htPotDelta`, and potential dbh increment, `dbhPotDelta`, are computed. These potential increments are provided as inputs to the competition model (based on Burkhart et al. 1987) specified in the procedure `cDelta` to obtain actual increments in the competitive environment. The competition model is computed as follows:

$$d = d_{pot} * (0.26325 + (2.11119 * cr^{0.56188} * e^{-0.26375 * ci - 1.03076 * cr}))$$

where *d* is the actual increment, d_{pot} is the potential increment, *cr* is the crown ratio, and *ci* is the competition index of the tree. Lastly, age, crown ratio (`cr`), and maximum crown width (`cwmax`) are updated for each tree.

3.3.3 Structural and Architectural Development

In this section, we first describe apical growth of a tree's trunk, followed by elongation of lateral first order branches. Bending and senescence of lateral branches are described at the end.

Development of tree trunks is specified by the rule

```
t:Tree b:Bud, (b.order == 0) ==> { int numInt = (t.hd / I_ELONG0);}
    t for(int i: (1:numInt)) (
        Trunk(I_ELONG0, I_INIT_DBH) RH(I_PHYLLO)
        [Bend(I_ANGLE, 0) I(I_ELONG1) Bud(1)] )
    Trunk(t.hd%I_ELONG0, I_INIT_DBH) RH(I_PHYLLO) Bud(0);
```

where `numInt` is the number of internodes to be created along the trunk and `t.hd` is the precomputed actual height increment for the tree. The for-loop inserts *Trunk* nodes according to a phyllotaxy constant `I_PHYLLO`, each with a lateral branching angle `Bend(I_ANGLE, 0)`, internode `I(I_ELONG1)`, and bud `Bud(1)`. The final apical internode is represented by a *Trunk* node with length `t.hd%I_ELONG0`, remainder of the division of actual height increment by internodal length. This rule operates only for the bud with order zero, i.e. apical bud for the main stem or trunk, as indicated by the query condition `(b.order == 0)`.

The development of lateral first order branches is specified by the rule

```
t:Tree b:Bud, (b.order == 1) ==> { Point3d locB = location(b);
    float dist = distToTrunk(t.x,t.y,locB.x,locB.y);
    if(locBud.z > t.hcbase) {
        float rp = 1-((locBud.z - t.hcbase)/(t.ht - t.hcbase));
        float cwah = (cwah(t.cwmax, rp, t.ht, t.dbh)/2);
    }
    float elong = cwah - dist; if(elong < 0) elong = 0;}
    t if(elong > 0)(RU(I_TROPISM) I(elong) Bud(1))
      else(b);
```

The 3D position, `locB`, of the first order bud is first computed. With its position, the minimum (perpendicular) distance, `dist`, of the bud from the trunk is computed. If the bud is not below `t.hcbase`, the height to the base of the crown, we compute the vertical relative position, `rp`, of the bud in the crown. `rp` is provided as input to a crown profile model based on Hann (1999) to compute `cwah`, the crown width at a specific height as follows:

$$f = 0.929973 - 0.135212 * rp^{0.5} - 0.131316 * (ht / dbh), \quad cwah = cwmax * rp^{f}$$

where *f* is a coefficient computed using the vertical relative position in crown, height, and dbh of the tree. *cwmax* is the pre-computed maximum crown width of the tree. Elongation, `elong`, of the lateral branch is computed as the difference between `cwah` and `dist`. A rotational node `RU(I_TROPISM)` with tropism angle `I_TROPISM`, and an *I* node representing an internode with length `elong` are appended to the graph if elong is positive. Figure 4 illustrates the various parameters associated with the crown profile graphically.

Mechanical bending of the branches is simulated by modifying the angles of *Bend* nodes. Branches originating below the pre-computed minimum height to crown base are removed. These two operations are specified with the following rules:

```
t:Tree b:Bend ::> {if(location(b).z > t.hcbase) b.angle+=I_BEND;
    else cutBranch(b);}
b (-->)+ Node ==>> ;
```

The first rule checks if the height `location(b).z` of the *Bend* node `b` is higher than the minimum height to crown base `t.hcbase` of the tree. If so, the angle of the *Bend* node `b` is incremented by `I_BEND` degrees. If not, the procedure `cutBranch` is invoked with `b` as input parameter to remove nodes representing the branch. The second rule is specified in the `cutBranch` procedure. It queries for the node `b` as well as all nodes `Node` with a path from `b` and replaces them with an empty production graph, effectively deleting these nodes from the instance graph.

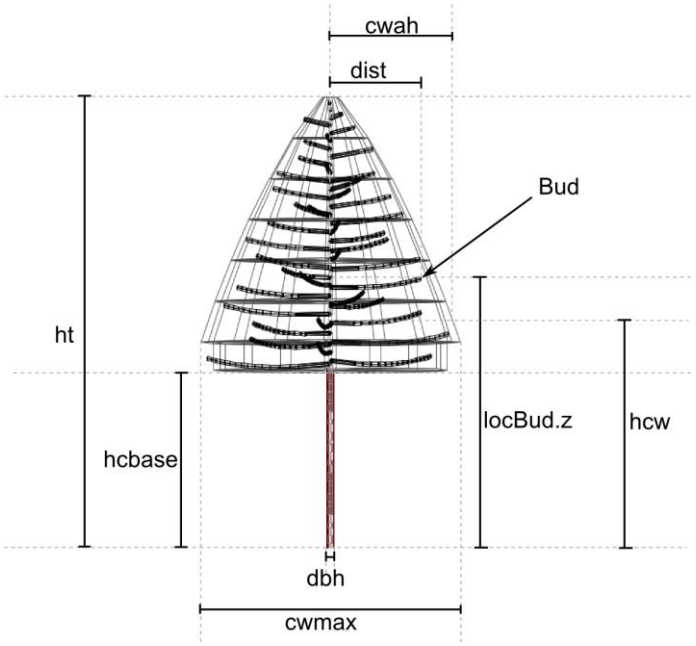

Figure 4. Illustration of variables – height (ht), diameter at breast height (dbh), maximum crown width (cwmax), bud's minimum distance to trunk (dist), crown width at height (cwah) for vertical height (hcw), height to base of crown (hcbase), and bud's vertical position (locBud.z). Relative position in crown (rp) is 1 – ((locBud.z – hcbase) / (ht – hcbase)).

3.4 Mortality

The third and last sub-step of `run()`, `mortality()`, contains procedures and rules that simulate the death of trees in the virtual forest stand. Mortality is based on the model by Burkhart et al. (1987) and is specified by the rule

```
t:Tree /> n:Node ==> {float prob = S_LIVE_JUVENILE;
    if (t.age >= S_AGE_COMPETE) prob = probLive(t.cr, t.ci);}
    if (probability(prob)) (t /> n);
```

A *Tree* node and all nodes with a refinement edge connection from it are queried from the instance graph. If the queried tree is younger than `S_AGE_COMPETE`, its probability of survival, `prob`, is `S_LIVE_JUVENILE`. If it is of age `S_AGE_COMPETE` or older,

its probability of survival is computed from its crown ratio (`t.cr`) and competition index (`t.ci`) in the procedure `probLive` as

$$k = -0.0023 * ci^{0.65206}, \quad p = 1.02797 * cr^{0.0379} * e^{k}$$

where k is a cofficient computed using the competition index ci, and p is the probability of survival, computed from the tree's crown ratio cr and k. The tree's survivial is determined by the procedure `probability` with `prob` as input parameter. If the tree survives, the nodes are specified as they were in the production statement on the right-hand side, leaving them intact in the instance graph. Otherwise, the production statement is left empty, effectively removing the nodes representing a tree from the instance graph.

A list of parameter values (capitals in code statements) is provided in Table 1. Figure 5 shows screenshots of the model.

S_AGE_COMPETE	8	T_INIT_HCBASE	0
S_DIM_X	16	T_INIT_HD	0
S_DIM_Y	16	T_INIT_HT	0.1
S_GRID_COUNT_X	4	T_INIT_NUMINT	0
S_GRID_COUNT_Y	4	T_RPMAX	0.1
S_LIVE_JUVENILE	0.8	I_ANGLE	80
S_SEED_POOL	26	I_BEND	1
T_CR	0.3	I_ELONG0	0.2
T_CROWN_SHADE	0.9	I_ELONG1	0.1
T_INIT_AGE	0	I_INIT_DBH	0.0575
T_INIT_CI	0	I_PHYLLO	132
T_INIT_DBH	1	I_TROPISM	-1.3

Table 1. Constant parameter values. Prefix S, T, and I, for stand, tree, and internode (organ) scales respectively.

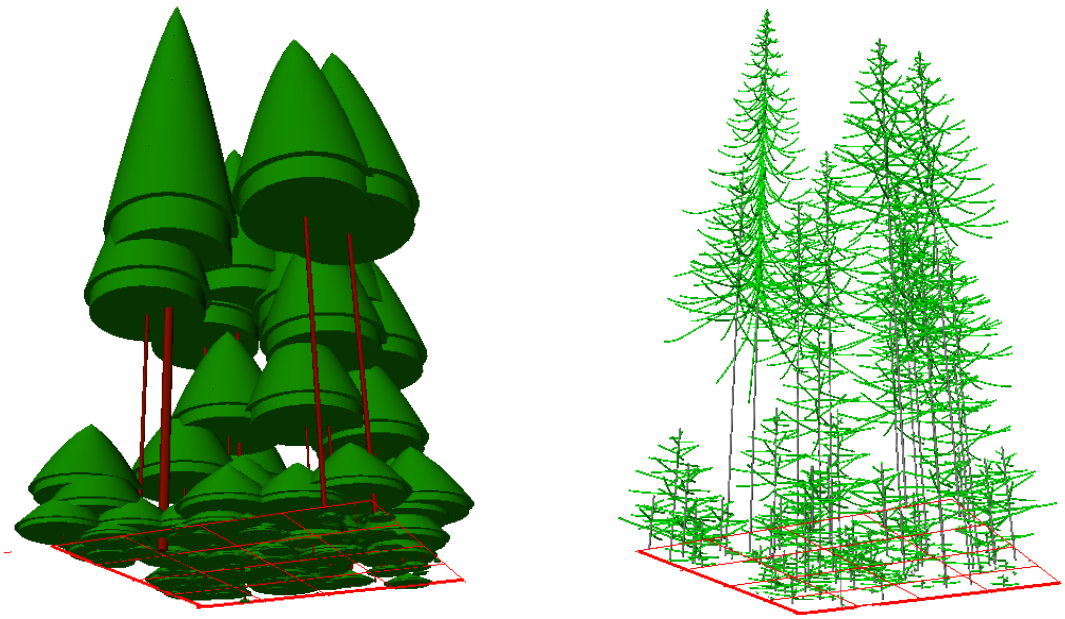

Figure 5. Illustration of 154 trees in a 265 year old stand. Left: Visualization of the crown profiles. Right: Visualization of organ scale with internodes.

4 Summary and Outlook

We presented the usage of rule-based programming on a single unified data structure for multiscale models as an alternative to merging scattered software packages. As shown in the demonstrative model, empirical forest stand and tree models can be integrated with rule-based modelling of tree architectures. With pre-defined scale relationships, entities in conventional forest stand and tree models are readily accessible in the rule-based realm. Potential work in the future is the integration of process-based or hybrid category models with significantly larger amounts of data at finer scales. Specific multiscale modelling techniques may be required to overcome computational barriers associated with large data quantity.

References

Biging, G.S.; Wensel, L.C. (1990): Estimation of crown form for six conifer species of northern California. Canadian Journal of Forest Research **20** (1990) 1137–1142.

Boudon, F.; Pradal, C.; Cokelaer, T.; Prusinkiewicz, P.; Godin, C. (2012): L-Py: An L-system simulation framework for modeling plant architecture development based on a dynamic language. Frontiers in Plant Science **3**, Article 76 (20 p.).

Burkhart, H.E.; Tomé, M. (2012): Modeling Forest Trees and Stands. Springer, Dordrecht Heidelberg New York London.

Burkhart, H.E.; Farrar, K.D.; Amateis, R.L.; Daniels R.F. (1987): Simulation of individual tree growth and stand development in loblolly pine plantations on cutover, site-prepared areas. Virginia Polytechnic Institute and State University, Blacksburg, Pub. FWS-1–87.

Cournède, P-H.; Mathieu, A.; Houllier, F.; Barthélémy, D.; de Reffye, P. (2008): Computing competition for light in the GREENLAB model of plant growth: A contribution to the study of the effects of density on resource acquisition and architectural development. Annals of Botany **101** (2008) 1207-1219.

Deussen, O.; Hanrahan, P.; Lintermann, B.; Mech, R.; Pharr, M.; Prusinkiewicz, P. (1998): Realistic modeling and rendering of plant ecosystems. Proceedings SIGGRAPH98, Computer Graphics Proceedings, Annual Conference Series, 1998, 275–286.

Doruska, P.F.; Mays, J.E. (1998): Crown profile modeling of loblolly pine by nonparametric regression analysis. Forest Science **44** (1998) 445–453.

Dyer, M.E.; Burkhart, H.E. (1987): Compatible crown ratio and crown height models. Canadian Journal of Forest Research **17** (1987) 572–574.

Godin, C.; Caraglio, Y. (1998): A multiscale model of plant topological structures. Journal of Theoretical Biology **191** (1998) 1-46.

Godin, C.; Costes, E.; Sinoquet, H. (1999): A method for describing plant architecture which integrates topology and geometry. Annals of Botany **84** (1999) 343-357.

Grimm, V.; Railsback, S.F. (2005): Individual-based Modeling and Ecology. Princeton University Press, Princeton Oxford.

Hanan, J. (1992): Parametric L-systems and their application to the modelling and visualization of plants. PhD Thesis, University of Regina.

Hanan, J. (2013): Functional-structural modelling with L-systems: Where from and where to. Proceedings of the 7th International Conference on Functional-Structural Plant Models FSPM13, Saariselkä, Finland, 2013, 1-3.

Hann, D.W. (1999): An adjustable predictor of crown profile for stand-grown Douglas-fir trees. Forest Science **45** (1999) 217–225.

Kari, L.; Rozenberg, G.; Salomaa, A. (1997): L systems. Handbook of Formal Languages, volume 1, chapter 5, 253-328. Springer, Berlin Heidelberg.

Karwowski, R.; Prusinkiewicz, P. (2004): The L-system-based plant-modeling environment L-studio 4.0. Proceedings of the 4th International Workshop on Functional-Structural Plant Models FSPM04, Montpellier, France, 403-405.

Kniemeyer, O. (2008): Design and Implementation of a Graph Grammar Based Language for Functional-Structural Plant Modelling. Doctoral dissertation, University of Technology at Cottbus, Fakultät für Mathematik, Naturwissenschaften und Informatik.

Kurth, W. (1999): Die Simulation der Baumarchitektur mit Wachstumsgrammatiken. Wissenschaftlicher Verlag Berlin, Berlin.

Kurth, W. (2002): Spezifikation der Simulation der Struktur und Dynamik von Pflanzenbeständen und Tierpopulationen mit sensitiven Wachstumsgrammatiken. In: Wittmann, J.; Gnauck, A. (eds.), Simulation in Umwelt- und Geowissenschaften. Workshop Cottbus, 7.-8. 3. 2002. Shaker, Aachen, 37-51.

Kurth, W.; Sloboda, B. (1999): Tree and stand architecture and growth described by formal grammars. II. Sensitive trees and competition. Journal of Forest Science **45** (1999) 53-63.

Kurth, W.; Kniemeyer, O.; Sloboda, B. (2012): Forest structure, competition and plant-herbivore interaction modelled with relational growth grammars. Lesnícky časopis - Forestry Journal, **58** (2012), 75-91.

Lane, B.; Prusinkiewicz, P. (2002): Generating spatial distributions for multilevel models of plant communities. Proceedings of Graphics Interface 2002, Calgary, Alberta. Canadian Human-Computer Communications Society, 2002, 69-80.

Lindenmayer, A. (1968): Mathematical models for cellular interactions in development. Journal of Theoretical Biology **18** (1968) 280–299, 300–315.

Lintunen, A.; Sievänen, R.; Kaitaniemi, P.; Perttunen, J. (2011): Models of 3D crown structure for Scots pine (Pinus sylvestris) and silver birch (Betula pendula) grown in mixed forest. Canadian Journal of Forest Research **41** (2011) 1779-1794.

Marshall, D.D.; Johnson, G.P.; Hann, D.W. (2003): Crown profile equations for stand-grown western hemlock trees in northwestern Oregon. Canadian Journal of Forest Research **33** (2003) 2059–2066.

Mäkela, A.; Landsberg, J.; Ek, A.R.; Burk, T.E.; Ter-Mikaelian, M.; Agren, G.I.; Oliver, C.D.; Puttonen, P. (2000): Process-based models for forest ecosystem management: current state of the art and challenges for practical implementation. Tree Physiology **20** (2000) 289–298.

Mäkela, A. (2003): Process-based modelling of tree and stand growth: towards a hierarchical treatment of multiscale processes. Canadian Journal of Forest Research **33** (2003) 398-409.

Mohren, G.M.J.; Burkhart, H.E. (1994): Contrasts between biologically-based process models and management-oriented growth and yield models. Forest Ecology and Management **69** (1994) 1–5.

Nepal, S.K.; Somers, G.L.; Caudill, S.B. (1996): A stochastic frontier model for fitting tree crown shape in loblolly pine (Pinus taeda L.). Journal of Agricultural, Biological, and Environmental Statistics **1** (1996) 336–353.

Ong, Y.; Kurth, W. (2012): A graph model and grammar for multi-scale modelling using XL. Proceedings 2012 IEEE International Conference on Bioinformatics and Biomedicine Workshops, Philadelphia, USA, 2012, 1-8.

Ong, Y.; Streit, K.; Henke, M.; Kurth, W. (2014): An approach to multiscale modelling with graph grammars. Submitted to Annals of Botany.

Pretzsch, H. (2009): Forest Dynamics, Growth and Yield. Springer, Berlin.

Prusinkiewicz, P. (1986): Graphical applications of L-systems. Proceedings of Graphics Interface 86, 1986, 247-253.

Prusinkiewicz, P.; Lindenmayer, A. (1990): The Algorithmic Beauty of Plants. Springer, New York.

Prusinkiewicz, P.; Hammel, M.; Mjolsness, E. (1993): Animation of plant development. Proceedings SIGGRAPH93, 1993, 351-360.

Schober, R. (1995): Ertragstafeln wichtiger Baumarten. Frankfurt am Main: J. D. Sauerländer's Verlag.

Seidl, R.; Rammer, W.; Scheller, R.M.; Spies, T.A. (2012): An individual-based process model to simulate landscape-scale forest ecosystem dynamics. Ecological Modelling **231** (2012) 87-100.

Smith, A.R. (1984): Plants, fractals, and formal languages. Computer Graphics (ACM/SIGGRAPH), **18** (1984), 1-10.

Thornley, J.H.M.; Johnson, I.R. (1990): Plant and Crop Modeling. A Mathematical Approach to Plant and Crop Physiology. Clarendon, Oxford.

Zeide, B.; Gresham, C.A. (1991): Fractal dimensions of tree crowns in three loblolly pine plantations of coastal South Carolina. Canadian Journal of Forest Research **21** (1991) 1208–1212.

Zeide, B.; Pfeifer, P. (1991): A method for estimation of fractal dimension of tree crowns. Forest Science **37** (1991) 1253–1265.