Physiologie und Morphologie der Pappel (*Populus sp.*) modelliert mit Relationalen Wachstumsgrammatiken

Gerhard Buck-Sorlin[1,2], Ole Kniemeyer[2], Winfried Kurth[2]

[1]Institut für Pflanzengenetik und Kulturpflanzenforschung (IPK),
Corrensstraße 3, 06466 Gatersleben
[2]Brandenburgische Technische Universität Cottbus, Institut für Informatik,
Lehrstuhl Praktische Informatik/Grafische Systeme, Postfach 101344, 03013 Cottbus

Summary

Models which combine the abstract (e.g., physiological) function of a plant with its architecture have more and more come into the focus of international research in recent years. In the wake of this development, an entirely new class of models has arisen, generally known as Functional Structural Plant Models (FSPM). The precise and transparent specification of models of this type poses new challenges to the design and implementation of programming languages.

Here we introduce, exemplarily at a biological model of young poplar trees, our new formalism of Relational Growth Grammars (RGG), which are a further development and amalgamation of the well-known graph grammar concept with Lindenmayer (L-) systems. The model has been written using our modelling language XL, which combines the conciseness of RGG and the power of the programming language Java, whilst its interpretation is being done using the software system GroIMP (Growth Grammar Related Interactive Modelling Platform). GroIMP is a new open-source project (available at www.grogra.de) offering unique possibilities for plant modelling and Artificial Life projects. RGG offer all the advantages known from L-systems, and, in addition, they overcome many of their disadvantages and can form a bridge between different scales: In the model proposed here, morphogenetic node replacement rules are combined with graph replacement rules describing a regulatory network of hormone biosynthesis, and rules updating nutrient concentrations of shoot modules, all in one and the same formalism.

Zusammenfassung

Modelle, welche die abstrakte (z.B. physiologische) Funktion eines Baumes mit seiner Architektur verknüpfen, sind in den letzten Jahren verstärkt in den Fokus der internationalen Forschung gerückt. Hierdurch ist eine völlig neue Klasse von Modellen entstanden, die sogenannten Functional Structural Plant Models (FSPM).

In diesem Beitrag stellen wir ein biologisches Modell junger unverzweigter Pappeln vor, das allgemeine morphogenetische Regeln für die Sprossarchitektur mit einer Untermenge physiologischer Prozesse kombiniert, wobei mit letzterem im wesentlichen die Bildung und der Transport des Pflanzenhormons Gibberellinsäure sowie einige vereinfachte Photosynthesefunktionen gemeint sind.

Das Modell wurde unter Verwendung unseres neuen Formalismus der Relationalen Wachstums-grammatiken (RGG) konzipiert, welche eine Weiterentwicklung und Verschmelzung zweier wohl-bekannter Konzepte, Graphgrammatiken und L(indenmayer)-Systeme, darstellen. Das Modell wurde in XL geschrieben, einer von uns entwickelten Modellsprache, die die Prägnanz der RGG mit der Mächtigkeit der Programmiersprache Java kombiniert. Das XL-Modell wird interpretiert unter Verwendung des Softwaresystems GroIMP (Growth Grammar Related Interactive Modelling Platform). GroIMP ist ein neues Open-Source-Projekt (verfügbar unter www.grogra.de), welches einzigartige Möglichkeiten für Projekte in den Bereichen Pflanzenmodellierung und Artificial Life bietet.

RGG/XL verfügen über alle von L-Systemen bekannten Vorteile und überwinden zusätzlich viele ihrer Nachteile: In dem hier vorgestellten Modell werden morphogenetische Ersetzungsregeln mit Graphen-ersetzungsregeln (letztere beschreiben ein regulatorisches Netzwerk der Hormonbiosynthese) und Regeln für die Aktualisierung der Konzentration von Photosyntheseprodukten in den Sprossmodulen kombiniert.

Einleitung

Typische Modellansätze in der Pflanzenphysiologie lassen sich meist einer der folgenden drei Klassen zuordnen: Erstens den statistischen Modellen, welche (nicht)lineare Abhängigkeiten zwischen Parametern, z.B. durch Korrelations- oder Regressionsanalyse, darstellen. Ein Nachteil dieses Ansatzes ist die Vernachlässigung kausaler Beziehungen. Die zweite Klasse sind prozessorientierte Reaktions- und Transportmodelle, die zwar wohldefiniert und weit verbreitet sind, deren simulierte Prozesse jedoch meist isoliert, unter Vernachlässigung räumlicher Strukturen, vonstatten gehen (oder bestenfalls eine stark vereinfachte räumliche Kompartimentierung anbieten). Als dritte Klasse haben sich reine Strukturmodelle etabliert, welche eine feste raumzeitliche Anordnung von Pflanzenorganen oder Zellen imitieren. Der Nachteil dieses Ansatzes liegt darin, dass er es nicht gestattet, die der simulierten Struktur zugrundeliegenden physiologischen und genetischen Wechselbeziehungen darzustellen. Der epistemologische Nutzen solcher isolierten Modelle ist oft gering, da in ihnen der wahren Natur des *Systems Pflanze* mit seinen Wechselwirkungen zwischen Prozessen, die auf verschiedenen Hierarchieebenen stattfinden, und emergenten Eigenschaften, die aus der Kopplung dieser Skalen hervorgehen, nicht in ausreichendem Maße Rechnung getragen wird.
Eine vielversprechende Perspektive zur Überwindung dieser Beschränkungen bieten in jüngster Zeit die sogenannten Funktions-Struktur-Pflanzenmodelle (FSPM; Godin et al. 2004). Andererseits ist eine maßgeschneiderte Modellspezifikationssprache eine unbedingte Voraussetzung, um die Komplexität der resultierenden Modelle auch in Zukunft meistern zu können. Die meisten gegenwärtigen FSPM haben nämlich den entscheidenden Nachteil, dass ihre Modellspezifikation nur Programmierern und Informatikern verständlich ist. Dieser Umstand verlangsamt die Schlüsselprozesse der Modellierung (Konzeption und Strukturierung in Programmmodule, Parametrisierung, Kalibrierung und Validierung) und schreckt potenzielle Nutzer des Modells ab.
Der derzeit am weitesten verbreitete Formalismus zur Spezifikation der architekturalen Entwicklung von Pflanzen sind L-Systeme. Sie wurden 1968 von dem Biologen Aristid Lindenmayer (Lindenmayer 1968) zur Modellierung fädiger Blaualgen eingeführt und erlangten danach zunächst nur unter Theoretikern im Bereich der Automaten und formalen Sprachen Popularität. Später wurden sie dann mit einer graphischen 3D-Interpretation der Strings (zugrundeliegende Datenstruktur) kombiniert und damit für die Modellierung der Struktur höherer Pflanzen nutzbar gemacht (z.B. Prusinkiewicz et al. 1994). Aus der Sicht eines Programmierers ist ein L-System im wesentlichen ein regelbasiertes System, ähnlich denen, die auf dem Gebiet der "Künstlichen Intelligenz" seit langem verwendet werden. L-Systemen fehlen jedoch einige der Merkmale, die heutzutage von einer Modellspezifikationssprache erwartet werden – insbesondere die Unterstützung objektorientierter Konzepte, ein hohes Maß an Interaktivität, sowie die Möglichkeit zum Schreiben multiskalierter Modelle. Darüber hinaus ist die Topologie der Strukturen, die sich aus der standardmäßigen graphischen Interpretation von L-Systemen ergibt, beschränkt auf lokal eindimensionale, baumartige Verzweigung, was nicht ausreicht, um komplexere Strukturen wie z.B. Netzwerke zu produzieren.

Wir haben daher einen neuen Modellformalismus entwickelt, "Relationale Wachstumsgrammatiken" (Relational Growth Grammars, RGG). Dies sind Graphgrammatiken, welche den L-System-Ansatz verallgemeinern, indem sie das Transformieren knotengelabelter Graphen mit einer beliebigen Anzahl von Kantentypen (Relationen) ermöglichen. Zusätzlich stellt die Einbettung der Programmiersprache Java prozedurale und objektorientierte Konstrukte für die Modellierung mit RGG zur Verfügung (Programmiersprache "XL", Kniemeyer et al. 2004). Ein ähnlicher Ansatz – jedoch auf C++ statt auf Java und auf herkömmlichen L-Systemen statt auf Graphgrammatiken basierend – wird derzeit von Karwowski und Prusinkiewicz (2003) in Form der Sprache "L+C" verfolgt.
Im Rahmen der von der DFG geförderten Forschergruppe "Virtual Crops" wurde die Eignung von RGG für die Modellierung von Getreide (Gerste: Buck-Sorlin et al. 2005) aufgezeigt. In diesem Modell konnten die äußere Form der Pflanze und ihre zeitliche Dynamik (Morphologie), die Wirkung einzelner Majorgene, sowie die Dynamik metabolischer regulatorischer Netzwerke, welche die Morphogenese (d.h. die Formentwicklung) kontrollieren, alle in ein und demselben formalen Rahmen repräsentiert werden.
Im folgenden sollen zunächst RGG etwas genauer definiert werden. Um ihre Leistungsfähigkeit zu demonstrieren, werden wir danach kurz ein RGG-basiertes FSPM der Pappel (*Populus spec*.) beschreiben. Die Pappel ist ein schnellwüchsiger, kommerziell wertvoller Baum der gemäßigten Breiten und gilt als Modellart der Forstgenetik und -physiologie.

Relationale Wachstumsgrammatiken

Relationale Wachstumsgrammatiken wurden als spezielle Art der Graphgrammatiken eingeführt (Kurth et al. 2005). Eine RGG-Regel ist formal ein Quintupel *(L,C,E,R,P)*, wobei $L \cup C \neq \emptyset$. L, die *eigentliche linke Seite* der Regel, ist eine Menge von *Graphen* mit Knoten- und Kantenlabeln. Ein *Ableitungsschritt* einer RGG umfasst die Entfernung einer Kopie ("Match") des L einer Regel aus einem (gewöhnlich) größeren Graphen und die Einsetzung des entsprechenden R, der *eigentlichen rechten Seite* der Regel, welche ebenfalls eine Menge von Graphen ist (wobei die zugrundeliegende Knotenmenge nicht notwendigerweise mit der von L disjunkt sein muss). C ist wiederum eine Menge von Graphen (wobei die Knotenmenge möglicherweise, aber nicht notwendig, mit der von L überlappt) und heißt *Kontext* der Regel. Damit eine Regel anwendbar ist, muss die Menge C mit einer Menge von Subgraphen des gegebenen Graphen übereinstimmen, und zwar in einer Weise, die mit dem Match von L konsistent ist; allerdings wird der Kontext im Ableitungsschritt nicht entfernt (außer den Teilen, die auch in L enthalten sind). Dieser Kontextbegriff verallgemeinert die "linken" und "rechten Kontexte" kontextsensitiver L-Systeme (Prusinkiewicz und Lindenmayer 1990) und ermöglicht eine flexible Kontrolle der Subgraphenersetzung. E ist eine Menge logischer Ausdrücke in Java-Syntax, die für gewöhnlich einige auf Knotenlabels von $L \cup C$ verweisende Parameter enthalten und als *Bedingungen* interpretiert werden, die erfüllt sein müssen, bevor die Regel angewendet werden kann. P schließlich ist eine (u.U. leere) Liste mit Java-Befehlen, welche eventuell auf Knotenlabels von $L \cup C \cup R$

verweisende Parameter sowie Parameter von E enthält. P spezifiziert ein Stück imperativen Code, das nach der Regelanwendung ausgeführt wird. Wir schreiben RGG-Regeln in der Form $(*C*), L, (E) \Rightarrow R\{P\}$, wobei die Reihenfolge der Teile C, L und E unbestimmt ist. Eine Kante mit dem Label a zwischen den Knoten x und y wird geschrieben als $x - a \rightarrow y$.

Eine RGG ist eine Menge von RGG-Regeln. In der Sprache XL können RGG-Regeln in Blöcken zusammengefasst werden, was die Schaffung einer zusätzlichen Regelhierarchie und eine explizite Kontrolle ihrer Anwendungsreihenfolge ermöglicht, wie in Table-L-Systemen (Rozenberg 1973). Eine RGG-basierte *Ableitung* ist eine Abfolge diskreter, sukzessiver Ableitungsschritte, die von einem gegebenen Anfangsgraphen (Axiom) ausgeht. In jedem Schritt werden – in Abhängigkeit vom gewählten Regelanwendungsmodus – eine oder alle passenden Regel(n) angewendet. *Sequenzielle* bzw. *parallele* Anwendungsmodi sind von Chomsky-Grammatiken bzw. L-Systemen her wohlbekannt. In den meisten biologischen Anwendungen ist der parallele Modus geeigneter. Parallelismus erfordert jedoch die Spezifikation einer *Konfliktlösungsstrategie* für überlappende Übereinstimmungen mit linken Regelseiten. Künftige Erweiterungen des RGG-Formalismus werden explizite Unterstützung für die geläufigsten Konfliktlösungsschemata anbieten. Bisher haben wir nur einen speziellen Fall in Betracht gezogen: die mehrfache Übereinstimmung von L mit einer und derselben Kopie von L selbst im Graphen (tritt auf, wenn L interne Symmetrien hat). Der Standardmodus der Regelanwendung, wie er in XL umgesetzt worden ist – im wesentlichen der bekannte Single-Pushout-Ansatz der Graphgrammatiktheorie – versucht, die Regel auf jede Übereinstimmung anzuwenden.

Unser Mechanismus zur *Einbettung* der rechten Regelseite in den umgebenden Graphen übergibt einfach einlaufende (bzw. ausgehende) Kanten der in der Textdarstellung am weitesten links (bzw. rechts) stehenden Knoten von L an die in der Textdarstellung am weitesten links (bzw. rechts) befindlichen Knoten von R. Künftige Versionen von XL werden andere Einbettungsstrategien zulassen.

Wir haben anderswo (Kniemeyer et al. 2004) gezeigt, dass es einfach ist, typische Standarddatenstrukturen wie Mengen, Multimengen, Listen oder mehrskalige Bäume (vgl. Godin und Caraglio 1998) als gelabelte Graphen darzustellen. Der RGG-Formalismus stellt Standardkantentypen (d.h. spezielle Kantenlabels) zur Darstellung der in diesen Datenstrukturen verbreiteten Relationen zur Verfügung, wie z.B. die Nachfolgerrelation in Listen oder das Enthaltensein in Mengen. Insbesondere sind *Zeichenketten* (Strings) spezielle Graphen, in denen alle Knoten in linearer Reihenfolge durch Nachfolgerkanten verbunden sind (daher sind L-Systeme ein Spezialfall der RGG). Da die Nachfolgerrelation so häufig verwendet wird, wird sie in unserer Notation durch ein Leerzeichen bezeichnet, d.h. $a \;\; b$ ist äquivalent zu $a - Nachfolger \rightarrow b$. Zusätzlich kann der Nutzer neue Relationen definieren, die in RGG in gleicher Weise wie Kanten verwendet werden können, indem er z.B. algebraische Operatoren wie die transitive Hülle verwendet.

Die RGG wurden zum Teil durch das PROGRES-System inspiriert (Schürr et al. 1999). Weiterhin wurden Eigenschaften parametrischer L-Systeme in den RGG-Formalismus einbezogen, insbesondere Befehle aus der Turtlegeometrie (vgl. Prusinkiewicz und

Lindenmayer 1990), welche als Knoten erlaubt sind und zur Interpretation abgeleiteter Graphen im 3D-Raum genutzt werden können.
Die Berücksichtigung eines imperativen Teils (*P* in unserer Definition) erlaubt die Ausführung von Code aus einer konventionellen objektorientierten Sprache. Zusätzlich dient in XL eine solche Sprache (Java) als Rahmen für die gesamte RGG und erlaubt dem Nutzer die Definition von Konstanten, Variablen, Klassen und Methoden. Weiterhin sind Graphknoten in XL Java-Objekte, die Träger beliebiger zusätzlicher Informationen und Funktionalitäten sein können, z.B. für Geometrie, visuelle Erscheinung oder Animationsverhalten.

Das Pappelmodel

An dieser Stelle kann aus Platzgründen keine vollständige Beschreibung des Pappelmodells erfolgen. Eine vollständige kommentierte Fassung des Quelltextes sowie weitere Informationen sind jedoch von den Autoren erhältlich. Weitere ausgewählte kommentierte Abschnitte des Quelltextes finden sich im Anhang zu diesem Artikel. Wir geben hier eine Zusammenfassung der Grundzüge: Das hier vorgestellte RGG-Modell basiert teilweise auf ECOPHYS (Host et al. 1990), einem Prozessmodell, das die Entwicklung einjähriger unverzweigter Pappelsämlinge während des ersten Jahres simuliert. Das RGG-Modell läuft mit einer Schrittweite von einer Stunde; Eingabeparameter sind die täglichen Verläufe der Lichtstärke (photosynthetisch aktive Strahlung, PAR) und Temperatur. Diese Eingabevariablen, sowie die aktuelle Blattfläche jedes simulierten Blattes, dienen der Berechnung der stündlichen Produktion von Photosynthaten (PS, als primäre Kohlenhydrate), und zwar in jedem zweiten Programmschritt. Diese PS sind die Bausteine für das Wachstum und die Streckung aller simulierten Organe. Eine einfache Beschattungsfunktion erlaubt die Unterscheidung zwischen beschatteten und unbeschatteten Blättern – letztere produzieren weniger PS. In unserer RGG-Implementierung wächst die simulierte Pappel aus einem Samen (der Samen enthält einen bestimmten Vorrat an Photosynthaten, um das Wachstum anzustoßen). Der Transport der PS entlang der simulierten Strukturen (Internodien, Blattspreiten und -stiel) findet im Wechsel mit dezidierten Wachstumsschritten statt: während des Transportes pausiert also das Wachstum, und umgekehrt. Dieser künstliche Wechsel zwischen Wachstums- und Transportschritten vermeidet mögliche Konflikte bei der Aktualisierung der Parameter und sichert so den reibungslosen Ablauf des Modells.
Wachstum besteht aus der Produktion neuer Blätter und Internodien aus einem Meristem in bestimmten Zeitintervallen (Plastochron), sowie der Streckung/Entfaltung bereits bestehender Organe in Abhängigkeit von den lokal verfügbaren bzw. aus benachbarten Organ-Modulen importierten PS. Die nachfolgenden beiden XL-Regeln legen das Verhalten des Meristems während und nach dem Plastochron fest:

```
  Meristem(r, p), (p>0)  ==> if (r <= 55) ( Meristem(r, p-1) );

x:Meristem(r, p), (p==0) ==> if (r <= 55) (

Internode(0,0,r+1,0) [ wj(0, pa[r]) Petiole(0.001, 0.01, 0, 0) RH(la)

Leaf(1, 2880, 3.0, 5.0, 0.1, 0.01, 0.01, r) w(1,ma[r]) RL(1)

blade.{color=0x008060} { numLeaves++; ori = irandom(0, 5);} ]

RH(azi[x.r]) RU(ori) Meristem(r+1,plast) );
```

Das Symbol **`Meristem`**, das auf der linken Seite beider Regeln erscheint, verweist auf eine Instanz eines zuvor definierten Objektmoduls. Biologisch repräsentiert es einen Wachstumskegel (Meristem). Beide Regeln spezifizieren Knotenersetzungen wie in konventionellen L-Systemen. In der ersten Regel wird der interne Parameter **`p`** mit einer Schrittweite von eins heruntergezählt. Diese Variable ist mit der deklarierten Variable *plast* initialisiert (Länge des Plastochrons in Stunden, siehe letzte Codezeile). Die erste Regel stellt somit sicher, dass ein neues Metamer (d.h. Internodium, Blattspreite und -stiel) nur nach Ablauf eines Plastochrons (normalerweise 48 Stunden, aber dieser Wert kann je nach Datenlage variieren) hergestellt wird. Die Bedingung **`r <= 55`** begrenzt die Gesamtzahl an Metameren auf 55. Sobald ein Plastochron abgelaufen ist (**`p == 0`**, Regel 2), wird ein **`Meristem`** in ein Internodium (**`Internode`**), einen Blattstiel (**`Petiole`**), ein Blatt (wobei die Spreite (**`Blade`**) durch eine externe Bézier-Oberflächendatei repräsentiert wird), und schließlich ein neues **`Meristem`** verwandelt, letzteres initiiert mit einem neuen Parameter **`p = plast`** sowie einem um eins inkrementierten Blattrang, **`r`**. Die parametrisierten Knoten **`RL`**, **`RU`** und **`RH`** sind Turtlebefehle zur relativen räumlichen Orientierung der neu gebildeten Organe. **`wj`** und **`w`** sind Hilfssymbole, welche die Dynamik (Null bis Maximalwert) des Divergenzwinkels zwischen dem Blattstiel und einerseits dem Internodium sowie andererseits der Spreite festlegen, wobei jedes Symbol mit seinem eigenen, blattrang-spezifischen (**`r`**) Maximalwert (**`pa[r]`** bzw. **`ma[r]`**) initialisiert wird. Man beachte den prozeduralen Java-Code in den geschweiften Klammern, der überall auf der rechten Seite einer XL-Regel eingestreut werden kann. Im oben angegebenen Fall dient er der Definition eines neuen Parameters, der für einen Turtlebefehl benötigt wird, sowie für die Inkrementierung eines Blattzählers.

In Übereinstimmung mit Host et al. (1990) durchläuft jedes neu gebildete Blatt vier Entwicklungsschritte (unentwickelt, neu gebildet, reif, überreif), welche unterschiedliche photosynthetische Charakteristiken aufweisen. Letztere dienen wiederum in Form von Koeffizienten als Eingabeparameter für die PS-Funktionen. Die Reifeklassen sind eine Funktion der Blattbildungsrate.

Das Modell wurde weiterhin mit einem metabolischen Regelungsnetz ausgestattet, welches die Biosynthese des Pflanzenhormons Gibberellinsäure (GA_1) sowie zweier seiner metabolischen Vorstufen beschreibt. Die Regeln, welche die Topologie und Dynamik dieses Netzwerks spezifizieren, werden ausführlich in Buck-Sorlin et al. (2005) beschrieben.

Ergebnisse und Diskussion

Um die Eignung von RGG zur Repräsentation unterschiedlicher hierarchischer und prozeduraler Skalenebenen innerhalb eines Formalismus zu demonstrieren, wurde das Pappelmodell derart konzipiert, dass es sowohl morphogenetische Regeln als auch ein damit verbundenes metabolisches Regulationsnetzwerk enthält. Die zugrundeliegende Idee war es, durch die Modifikation bestimmter Netzwerkparameter ein neues metabolisches Gleichgewicht herbeizuführen, welches wiederum Auswirkungen auf bestimmte morphogenetische Regeln haben und wodurch sich letztendlich die resultierende sichtbare Morphologie verändern sollte. Auf diese Weise werden übrigens moderne Genexpressionsversuche *in vivo* ausgeführt, nämlich durch die Anzucht von normalen Pflanzen (Wildtyp) neben solchen mit einer genetischen Mutation an einer wohldefinierten Position im Genom (Mutante), worauf die Beobachtung der resultierenden Morphologie folgt. Hieraus lassen sich meist Schlussfolgerungen über den zugrundeliegenden (molekularen) Mechanismus des untersuchten Gens ziehen.
Für unsere Zwecke wählten wir ein Beispiel aus Busov et al. (2003): Diese Untersuchung beschreibt eine transgene Hybridpappelmutante, in der das Gen für das Enzym GA 2-oxidase durch einen Transkriptionsenhancer überaktiviert wird. GA 2-oxidase ist ein katabolisches Enzym, das die bioaktive Form der Gibberellinsäure (GA_1) in seine inaktive Form, GA_8, überführt und dadurch abbaut. Die Internodienstreckung (nach der Internodienbildung durch das Meristem) hängt von der GA_1-Konzentration ab. Die graphische Ausgabe des Pappelmodells ist in Abb. 1 zu sehen. Auf der rechten Seite des Bildes sieht man eine Simulation der Mutante, die den Phänotyp der echten Mutante reproduziert: Aufgrund der niedrigen GA_1-Konzentration strecken sich die Internodien kaum, was zu einer gedrungenen oder plumpen (engl. "stumpy", der Name der Mutante) Erscheinung führt, während die Anzahl der Blätter nicht modifiziert wird.
Durch Hinzufügen bzw. Modifizieren nur weniger XL-Regeln wurde das Einzelpflanzenmodell zu einem (provisorischen) Bestandesmodell erweitert, welches zwar noch keine Umweltsensitivität aufweist, jedoch als Fallstudie über die Auswirkungen unterschiedlicher Anfangsparameter (in diesem Fall Länge des Plastochrons) auf die morphologische Entwicklung dienen kann (Abb. 2).
Die gegenwärtige Version des hier vorgestellten Modells ist nicht dazu konzipiert worden, um bestimmte wissenschaftliche Fragestellungen hinsichtlich der Physiologie bzw. Genetik der Pappel zu lösen, obwohl sich spätere Modellvarianten durchaus dazu eignen sollten. Unsere Absicht war es zum jetzigen Zeitpunkt vielmehr aufzuzeigen, dass der hier vorgestellte Formalismus prinzipiell geeignet ist, eine weite Palette biologischer Prozesse zu repräsentieren und sich dabei gleichzeitig strikt an ein klares und knappes Format zu halten. Es bleibt abzuwarten, ob er wirklich – wiederum prinzipiell – auf *jeden beliebigen* biologischen Prozess (wobei Struktur als das Ergebnis einer Prozesskette aufzufassen ist) angewendet werden kann: Wir haben bereits selbst Situationen identifiziert, welche die Erweiterung unseres Formalismus erfordern, besonders auf der zellulären Ebene, bei der Prozesse, welche die Zellteilung und -streckung betreffen, nicht durch traditionelle Knotenersetzungsregeln erfasst werden können, sondern bei der wir es mit einer Menge eher komplexer Kanten- bzw. Zellwand-

Ersetzungsregeln zu tun bekommen (vgl. Prusinkiewicz und Lindenmayer 1990). Da jedoch andererseits RGG echte Graphen verarbeiten (und nicht eindimensionale Zeichenketten wie in konventionellen L-Systemen), würde die dynamische Darstellung eines aus Zellen aufgebauten Gewebes als 3D-Graph mit unserem Formalismus bedeutend einfacher sein. Weiterhin findet der Transport von Substanzen entlang bzw. in simulierten Strukturen derzeit in einer diskretisierten Weise sowie auf einer fixierten Meso-Ebene statt (d.h. von einem Internodium zum nächsten): Die Simulation eines kontinuierlichen Transports sowie skalierbarer Konzentrationsgradienten erfordert daher eine Erweiterung der RGG in Richtung einer optimierten Lösung gewöhnlicher und partieller Differentialgleichungen.

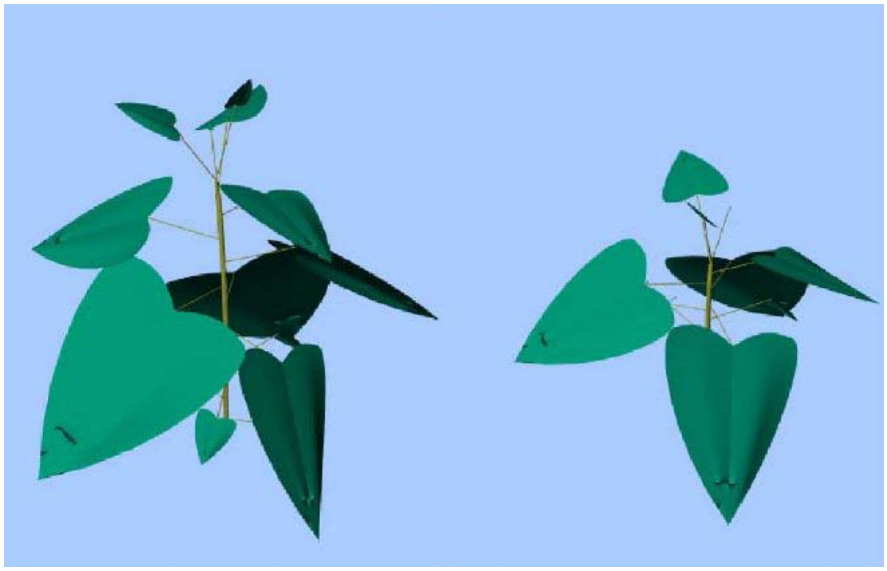

Abb. 1: Zwei Simulationsläufe des RGG-Pappelmodells. Der Wildtyp (links) zeigt normal gestreckte Internodien, die "stumpy"-Mutante (rechts) gestauchte Internodien. Siehe "Ergebnisse und Diskussion" für weitere Informationen.

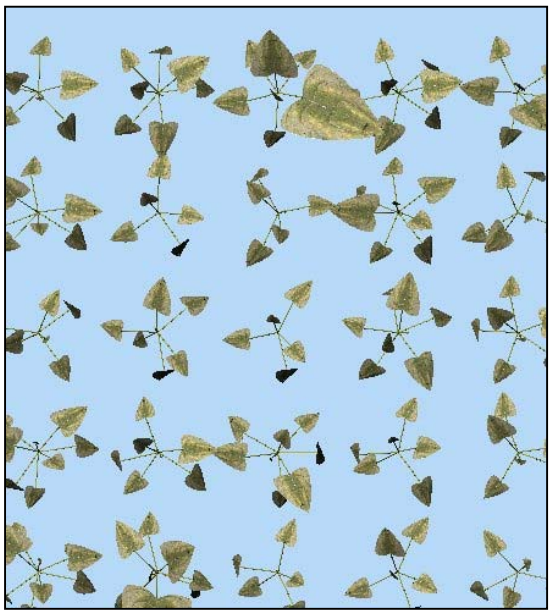

Abb. 2: Kleiner Bestand simulierter Jungpappeln (Alter bis ca. 1 Monat), mit POV-Ray gerendert (Draufsicht). Heterogenität zwischen den einzelnen Pflanzen wurde durch unterschiedliche Plastochrone sowie durch Variation der Grundausrichtung der Pflanzen erzielt.

Literatur

BUCK-SORLIN, G.H., KNIEMEYER, O., KURTH, W. (2005) Barley morphology, genetics and hormonal regulation of internode elongation modelled by a relational growth grammar. *New Phytol.* **166**, 859-867

BUSOV, V.B., MEILAN, R., PEARCE, D.W., MA, C., ROOD, S.B., STRAUSS, S.H. (2003) Activation tagging of a dominant gibberellin catabolism *GA 2-oxidase* from poplar that regulates tree stature. *Plant Physiol.* **132**, 1283-1291

GODIN, C., CARAGLIO, Y. (1998) A multiscale model of plant topological structure. *J. Theor. Biol.* **191**, 1-46

GODIN, C., HANAN, J.S., KURTH, W., LACOINTE, A., TAKENAKA, A., PRUSINKIEWICZ, P., DEJONG, T., BEVERIDGE, C., ANDRIEU, B. (eds.) (2004) 4th International Workshop on Functional-Structural Plant Models. Abstract Volume. Montpellier France: UMR AMAP.

HOST, G.E., RAUSCHER, H.M., ISEBRANDS, J.G., DICKMANN, D.I., DICKSON, T.R., CROW, T. R., MICHAEL, D.A. (1990) The ECOPHYS User's Manual. Rhinelander, WI, North Central Forest Experiment Station: 50 pp.

KARWOWSKI, R., PRUSINKIEWICZ, P. (2003) Design and implementation of the L+C modeling language. *El. Notes Theor. Comp. Sci.* **86**: 19 pp.

KNIEMEYER, O., BUCK-SORLIN, G., KURTH, W. (2004) A graph grammar approach to Artificial Life. *Artif. Life* **10**, 413-431.

KURTH, W., KNIEMEYER, O., BUCK-SORLIN, G. (2005) Relational growth grammars – a graph rewriting approach to dynamical systems with a dynamical structure. In: Unconventional Programming Paradigms. International Workshop UPP2004, Le Mont Saint Michel, France, September 15-17, 2004, Revised Selected and Invited Papers. Eds.: Banâtre, J.-P.; Fradet, P.; Giavitto, J.-L.; Michel, O. *Lect. Notes Comp. Sc.* **3566**, 56-72. DOI: 10.1007/11527800_5

LINDENMAYER, A. (1968) Mathematical models for cellular interactions in development. *J. Theor. Biol.* **18**, 280-315.

PRUSINKIEWICZ, P., LINDENMAYER, A. (1990) The Algorithmic Beauty of Plants. Springer, New York.

PRUSINKIEWICZ, P.W., REMPHREY, W.R., DAVIDSON, C.G., HAMMEL, M.S. (1994) Modeling the architecture of expanding *Fraxinus pennsylvanica* shoots using L-Systems. *Can. J. Bot.* **72**, 701-714.

ROZENBERG, G. (1973) T0L systems and languages. *Information and Control* **23**, 357-381.

SCHÜRR, A., WINTER, A.J., ZÜNDORF, A. (1999) The PROGRES approach: Language and environment. In: G. Rozenberg (ed.): Handbook of Graph Grammars and Computing by Graph Transformation, Vol. 2, Applications, Languages and Tools. World Scientific, Singapore, 487-550.

Anhang: Quelltext des Pappelmodells (Auszüge):

Produktion von GA_{19} (Vorstufe des bioaktiven Hormons GA_1) zur Zeit *t* in einem Meristem:

```
double ga19Prod (double t)
{
    return ((t > 0) && (t < 400)) ? 0.2 : 0; //wild type
}
```

Aktivität der 2β-Hydroxylase zur Zeit *t* (in allen "Zellen", d.h. aktiven Modulen):

```
double ga2bOH (double t)
{
    return (t < 1000) ? 3 : 0;              //wild type: 0.8, "stumpy": 3
}
```

Methode **`init`**, erzeugt aus dem Startwort **`Axiom`** ein Objekt **`Tree`**, sowie – darin gekapselt – ein Meristem **X** mit einem einfachen GA-Netzwerk:

```
void init ()
[
    Axiom ==> Tree(0, 1.5f) X(0, 0.1f, plast, 0f, 1) [GA19(3.577)] [GA20(0.719)]
              [GA1(0.226)];
]
```

Öffentliche Methode **`grow`** (kann vom Nutzer direkt aufgerufen werden). Hier wird Wachstum als eine Abfolge zwischen einem metabolischen Prozess (hauptsächlich Hormonbiosynthese), zusammengefasst in der Methode **`metabolism`**, und einer sich anschliessenden morphogenetischen bzw. Organstreckungsreaktion aufgefasst (Methode **`morphology`**):

```
public void grow ()
{
    metabolism ();
    morphology ();
    time += DELTA_T;
}
```

Methode **`metabolism`** (kann nicht direkt vom Nutzer aufgerufen werden). Hier wird in einer Serie von Aktualisierungsregeln das Hormon GA_1 aus seinen beiden Vorstufen GA_{19} und GA_{20} hergestellt; GA_1 selbst zerfällt und inhibiert außerdem seine eigene Produktion (die verwendeten Funktionen **`michaelisMenten`**, **`catabolism`** und **`competitiveInhibition`** sind nicht aufgeführt):

```
void metabolism ()
[
/* Produktion von GA19 in einem Meristem (0, Cell) */
    c:X [ga:GA19] ::>
        {
        /* Die Produktion von GA19 hängt vom Blattrang ab (in il hinterlegt): */
        ga[concentration] :+= ga19Prod (time) * DELTA_T * il[c.r];
        }
    Cell [s:GA19] [p:GA20] [f:GA1] ::>
        competitiveInhibition (s, p, f, 0.1 * DELTA_T, 2, 0.4);
    Cell [s:GA20] [p:GA1] ::>
        michaelisMenten (s, p, 0.2 * DELTA_T, 1); //letzter Wert: 1
        /* Abbau von GA1 durch 2β-Hydroxylase: */
    Cell [s:GA1] ::>
        catabolism (s, ga2bOH (time) * DELTA_T, 1);
        }
]
```

Methode **morphology**. Sie stellt die richtige Abfolge der Methoden **transport**, **grow** und **metabolism** sicher:

```
void morphology ()
{
     ((Node)this)[name] := getName ();   /* hindert "Run run" am Stehenbleiben */
     time += DELTA_T;
     if (time > transportTime + DELTA_T_TRANSPORT)
        {
        apply ("transport", 1);
        transportTime += DELTA_T_TRANSPORT;
       }
     if (time > growTime + DELTA_T_GROW)
        {
        apply ("grow", 1);
        growTime += DELTA_T_GROW;
        }
     if (time > metabTime + DELTA_T_METABOLISM)
        {
        apply ("metabolism", 1);
        metabTime += DELTA_T_METABOLISM;
        }
}
```

Methode **grow** : Zusammenstellung der eigentlichen morphogenetischen (Wachstums- und Verzweigungs-) Regeln:

```
void grow ()
[
     t:Tree ::> t.age++;
     X(r, l, p, ps, o), (p>0)  ==> if (r <= 55 && o<=2) ( X(r, l, p-1, ps, o) );
     X(r, l, p, ps, o), (p==0) ==> if (r <= 55 && o==1) ( Y(0f,0,r+1, 0f, 0)
          [GA19(0.01)] [ GA20(0.01)] [GA1(0.001)] [ w(0, pa[r])
             P(0.001f, 0.01f, 0f, 0f) RH(la)
             B(1f, 2880, 3.0f, 5.0f, 0.1f, 0.01f, 0.01f, r) wj(1,ma[r]) RL(1)
             blade.{material=leafmat} { numLeaves++;
                 ori = irandom(0, 5); azi = irandom(120, 150);} ]
          RH(azi) RU(ori) X(r+1, l, plast, ps, o) );
     x:w(,wmax), (x[angle]<=wmax) ::> x[angle] :+= 1; /* Inkrementierung des Winkels
                                                  zwischen Blattstiel und Spreite */
     x:wj ::> x[angle] :+= 10; /* Inkrementierung des Winkels zwischen Blattstiel und
                                 Internodium */
     y:Y [ga19:GA19][ga20:GA20][ga1:GA1] ::> /* Internodienentwicklung */
          if (numLeaves - y.lnfb <= 5 && y.age <= 48)
               {
               ga19[concentration] :+= ga19Prod (time) * DELTA_T;
               y[length] :+= (float) ga1[concentration]; /* direkte Abhängigkeit der
                             Internodienstreckung von der GA1-Konzentration */
               y[ps] :+= -y[ps]/24;
               y[volume]:= PI*(y[diameter]/2)*(y[diameter]/2)*y.length;
               y[mass]:=y[volume]*0.0045f;
               y[age] :+= 1;
               if (y.diameter <= mdm[y.lnfb]) { y[diameter]:+= 0.01f;}
               }
          else if (numLeaves - y.lnfb > 5 && y.age <= 48)
               {
               ga19[concentration] :+= ga19Prod (time) * DELTA_T;
               y[length] :+= (float) ga1[concentration];
               y[volume]:= (PI*(y[diameter]/2)*(y[diameter]/2)*y.length);
               y[mass]:=y[volume]*0.0045f;
               if (y.diameter <= mdm[y.lnfb]) { y[diameter]:+= 0.005f;}
               y[age] :+= 1;
               }
     b:B, root(b) [ga1:GA1] ::> /* Blattentwicklung */
          if ((numLeaves - b.lnfb - 1 < 5) && (!isShaded(b)))
             {
             b.ps += PS((float) b.area, ConvertMC(numLeaves-b.lnfb-1)) * 0.1f;
             if (ga1[concentration] > 0.01) { if (b.area <= ar[b.lnfb])
                 { b.area += 1;}}
```

```
           else { if (b.area <= ar[b.lnfb]*0.67) b.area += 0.67;}}
           b.ps = b.ps - 1.0f;
           b.setScale (5*(float) b.area);
           }
        else
           {
           b.ps += PS((float) b.area, ConvertMC(numLeaves-b.lnfb-1)) * 0.1f;
           if (ga1[concentration] > 0.01) { if (b.area <= ar[b.lnfb])
               { b.area += 1;}}
           else { if (b.area <= ar[b.lnfb]*0.67) { b.area += 0.67;}}
           b.ps = b.ps - 1.0f;
           b.setScale (5*(float)  b.area);
           }
    p:P, (* b: B *), root(p) [ga1:GA1] ::> /* Entwicklung des Blattstiels */
            if (ga1[concentration] > 0.01)
        { if (p.length <= mpl[b.lnfb])
            {
            p.length += 0.1f;
            }
        }
        else
        { if (p.length <= mpl[b.lnfb] * 0.67)
            {
            p.length += 0.067f;
            }
        }
    b:B(m,z,d,l,lsc,a,ps,lnfb), (z>1) ==> B(b.getScale(),z-1,d,l,lsc,a,ps,lnfb);
    B(m,z,d,l,lsc,a,ps,lnfb), (z==1) ==>  ; /* Blattabwurf */
]
```