In: KELEMEN, JOZEF AND SOSÍK, PETR: *Advances in Artificial Life (Proceedings of the 6th European Conference on Artificial Life)*, 242-251. Springer Verlag, Berlin Heidelberg, 2001.

transsys: A Generic Formalism for Modelling Regulatory Networks in Morphogenesis

Jan T. Kim

Institut für Neuro- und Bioinformatik,
Seelandstraße 1a, 23569 Lübeck, Germany
kim@inb.mu-luebeck.de

Abstract The formal language `transsys` is introduced as a tool for comprehensively representing regulatory gene networks in a way that makes them accessible to ALife modelling. As a first application, Lindenmayer systems are enhanced by integration with `transsys`. The resulting formalism, called `L-transsys`, is used to implement the ABC model of flower morphogenesis. This `transsys` ABC model is extensible and allows dynamical modelling on the molecular and on the morphological level.

1 Introduction

During the last years, regulatory networks consisting of transcription factors and the genes encoding them have received increasing amounts of interest in various biosciences, including bioinformatics and Artificial Life. Regulatory networks are among the key mechanisms of phenotypic realization of genetic information. One of the most prominent phenotypic properties is the morphology of an organism, and consequently, various models and analyses of regulatory networks are concerned with morphogenesis.

Regulatory networks can be modelled by various approaches, e.g. by differential equation systems [10, 16] or rule-based systems [7, 13]. Furthermore, a multitude of systems for modelling morphogenesis have been developed, including cellular automata [1], Lindenmayer systems (L-systems) [2, 5, 6, 8] and others [3, 4, 7, 15], and such models have been combined with regulatory network models in different ways. These models have led to many new results and insights. However, the development of increasingly detailed and specialized models makes it difficult to integrate the findings obtained with different models into a coherent theoretical framework.

In this paper, the `transsys` modelling framework is introduced which is designed to address this problem. The motivation underlying `transsys` is to provide a framework which (1) is sufficiently generic to integrate several methods for modelling gene regulatory systems, (2) facilitates interfacing with models of growth processes and (3) allows formulation of concise and expressive models to facilitate scientific communication.

L-systems (see [17] for a general introduction) are powerful systems for modelling morphogenesis. However, even with parametric L-systems, it is not possible to simulate regulatory networks in a concise and comprehensive way because explicit representations for genes and their products are lacking. Therefore, L-systems were considered to be a suitable target of enhancement by `transsys`. The resulting system allows modelling of wildtype and mutant flower morphogenesis dynamics such that models of single gene mutants can be derived from the wildtype model by single alterations in the `transsys` model specification. This is demonstrated by implementing the ABC model of flower morphogenesis in *Arabidopsis* [14].

2 The Modelling Framework

Conceptually, regulatory networks consist of two types of node elements, transcription factors and genes that encode transcription factors. Edges in a regulatory network are constituted by activating and repressing effects that factors have on the expression of genes. `transsys` is a computer language designed to comprehensively represent regulatory networks such that they can be combined with a wide range of dynamical models. A network description is called a *`transsys` program*. The `transsys` language is outlined in Sect. 2.1, using the `transsys` program shown in Fig. 1 as an example. A full specification will be published in a forthcoming technical report[1]. Sections 2.2 and 2.3 describe simulation of `transsys` network and integration into an L-system framework.

The set of factors in a `transsys` program is denoted by $\mathcal{F} = \{f_1, f_2, \ldots, f_n\}$, the set of genes is denoted by $\mathcal{G} = \{g_1, g_2, \ldots, g_m\}$.

The state of a regulatory network in `transsys` is defined by the concentrations of the factors described in the `transsys` program. The set of factor concentrations is denoted by $\mathcal{C} = \{c_f : f \in \mathcal{F}\}$. Such a set is also referred to as a *`transsys` instance*.

Temporal dynamics within a regulatory networks are constituted by changes in concentrations of transcription factors. These changes are computed by numerically integrating a set of differential equations which are specified and parameterized by the `transsys` program. The change in concentration of factor f is denoted by Δc_f. The amount of factor f synthesized due to expression of gene g is denoted by $\Delta_g c_f$.

2.1 The `transsys` Language

Transcription Factors Transcription factors are described by factor blocks, as shown in Fig. 1. A factor is characterized by its name immediately following the `factor` keyword and its decay rate, specified within the block. The decay rate of a factor f is formally denoted by r_f. In relation to the many biochemical

[1] check `http://www.inb.mu-luebeck.de/transsys/` for software and current documents on `transsys`.

```
transsys example
{
  factor A { decay: 0.05; }
  factor R { decay: 0.05; }

  gene rgene
  {
    promoter
    {
      A: activate(1.0, 10.0);
      R: repress(1.0, 1.0);
    }
    product
    {
      default: R;
    }
  }

  gene agene
  {
    promoter
    {
      constitutive: random(0, 0.1);
      A: activate(0.1, 1.0);
      R: repress(1.0, 1.0);
    }
    product
    {
      default: A;
    }
  }
}
```

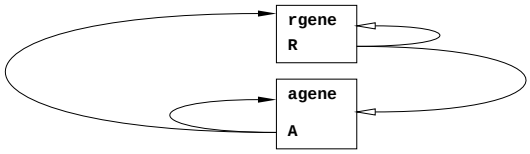

Figure 1. An example `transsys` program with two factors `A` and `R`, and two genes `agene` and `rgene`. The regulatory interactions expressed by `activate` and `repress` promoter statements can be displayed graphically by arrows with filled or open arrow heads, respectively.

and biological properties of a protein, this is a strongly simplified and abstract representation. It is possible, however, to model effects of protein-protein to some extent, as decay rates are specified by expressions (see below), so they can e.g. be made to depend on activities of other factors.

Genes Genes are units of genetic information that can be partitioned into a structural part and a regulatory part. The structural part encodes a biological activity which may be realized RNA or by protein molecules. The regulatory part determines the expression pattern of the gene by cis-acting elements which are recognized by transcription factors.

The regulatory and the structural part of a gene are represented in `transsys` by the `promoter` and `product` blocks within a `gene` block (see Fig. 1). The `product` block specifies the name of the factor encoded by the gene. The set of all genes encoding a factor f is denoted by $\mathcal{E}_f$.

The `promoter` block contains a list of statements describing conditions for activation or repression of the gene. A statement models a transcription factor binding site and the effect which binding has on the activation of the gene by describing how to calculate a contribution to activation (see Sect. 2.2). Let a_i denote the contribution of promoter statement i.

There are three types of statements possible in the `promoter` block. `constitutive` is the most generic type, this statement specifies an expression determining an amount of activation:

$$a_i = \text{ result of evaluating expression} \tag{1}$$

`activate` and `repress` statements are both preceded by a factor name f, and both have a list of two expressions as arguments. The arguments determine the specificity, denoted by a_{spec}, and the maximal rate of activation, denoted by a_{max}. The actual amount of activation is calculated according to the Michaelis-Menten-equation:

$$a_i = \frac{a_{\text{max}} c_f}{a_{\text{spec}} + c_f} \tag{2}$$

Repression is calculated by the same formula with the sign reversed:

$$a_i = -\frac{a_{\text{max}} c_f}{a_{\text{spec}} + c_f} \tag{3}$$

Both parameters a_{spec} and a_{max} are specified by expressions, which allows modelling of modulation of activation by protein-protein interactions. The amount of product p synthesized through expression of gene g in a time step is given by

$$\Delta_g c_p = \begin{cases} a_{\text{total}} := \sum_i a_i & \text{if } a_{\text{total}} > 0 \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

Expressions The form of expressions in `transsys` was designed to match that of expressions in programming languages like C or Java, i.e. the arithmetic and logical operators from these languages were built into `transsys`.

The main difference between expressions in `transsys` and in other languages is the interpretation of identifiers: Identifiers are used exclusively to refer to factor concentrations. Thus, for example, the promoter statement `constitutive: 2 * x` in a gene `g` encoding factor `p` means that the concentration of `f` is increased through expression of `g` by $\Delta_{\mathtt{g}} c_{\mathtt{p}} = 2 \cdot c_{\mathtt{x}}$. This statement could e.g. be used to model a factor `x` in the transcription initiation machinery which determines the basal level of expression.

2.2 Dynamics

Simulation of regulatory network dynamics in `transsys` starts with calculating factor concentration changes according to

$$\Delta c_f(t) = \left(\sum_{g \in \mathcal{E}_f} \Delta_g c_f(t) \right) - r_f(t) c_f(t) \tag{5}$$

where t denotes the time step in which the calculation takes place. Having calculated all concentration changes, $\mathcal{C}(t+1) = \{c_{f_1}(t+1), c_{f_2}(t+1), \ldots, c_{f_n}(t+1)\}$, the set of factor concentrations at time $t+1$, is determined by calculating

$$c_f(t+1) = c_f(t) + \Delta c_f(t) \tag{6}$$

for all factors $f \in \mathcal{F}$. After this derivation of $\mathcal{C}(t+1)$ from $\mathcal{C}(t)$, simulation of growth or diffusion dynamics may be performed, depending on the modelling context, before the next derivation is done.

2.3 `transsys` within an L-system Framework

The formalism designed by combining parametric L-systems with `transsys` is called `L-transsys`. The concept underlying `L-transsys` is a replacement of parameter lists, which are associated with symbols in parametric L-systems, with `transsys` instances.

An `L-transsys` symbol can be associated with one specific `transsys`. In the terms of parametric L-systems, this can approximately be construed as associating the factor concentrations as real-valued parameters with the symbol. However, the factor concentrations in a `transsys` instance are more than an unstructured set of numeric parameters as the instance provides a method for deriving $\mathcal{C}(t+1)$ from $\mathcal{C}(t)$. This allows a straightforward algorithm for interleaved application of `transsys` updates and L-system rule application:

1. Create the axiom (initial string), including initial factor concentrations.
2. For all symbols with a `transsys` instance, derive $\mathcal{C}(t+1)$.
3. From the current string with updated `transsys` instances, derive the next string according to the L-system production rules.
4. Continue with step 2 until desired number of derivations is reached.

3 Results

According to the ABC model of flower morphogenesis [14], flower morphogenesis is controlled by three functions, labelled A, B and C. Expression of only A function results in formation of sepals, joint expression of A and B functions leads to petal formation, stamens are formed upon joint expression of of B and C, and expression of C alone yields carpels.

The implementation of the ABC model with a `transsys` program is shown in Fig. 2, a listing is provided in Appendix A. In addition to the factors `a`, `b` and `c`, encoded by `agene`, `bgene` and `cgene`, respectively, the model comprises three more factors. Factors `bc` and `bc_act` and the corresponding genes form an oscillating system which is used to control the domain of B function expression. Factor `cc` controls the onset of C function expression. These three `transsys` factors are an *ad hoc* design which is intended to be replaced with more realistic models of molecular factors in the future. Of course, `transsys` can also be used to explore various hypotheses about the regulatory network controlling expession of the A, B and C functions. As shown in Fig. 3, the model satisfactorily matches the dynamics expected in the wildtype situation.

Figure 4 shows the graphical result of incorporating the `transsys` ABC program into an `L-transsys` model. The L-system rules in this model implement the growth behaviour outlined above. The morphological whorl structure of an

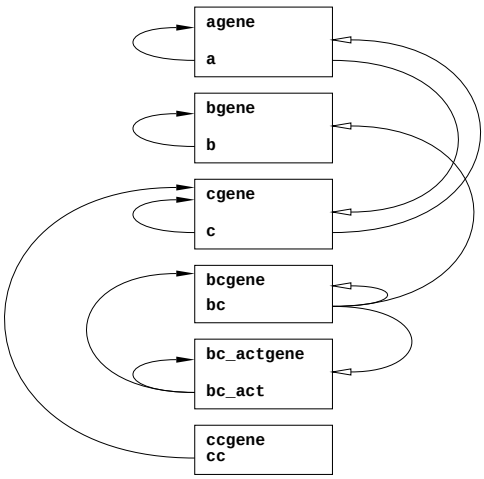

Figure 2. The `transsys` program implementing the ABC model of flower morphogenesis.

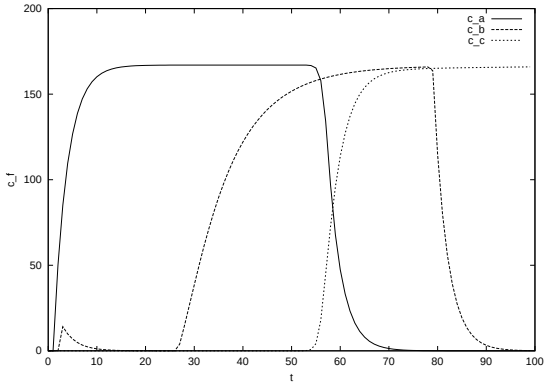

Figure 3. Temporal dynamics of gene expression in the regulatory network depicted in Fig. 2. The `transsys` instance was initialized with all concentrations set to zero.

Arabidopsis flower is correctly rendered by the `L-transsys` version of the ABC model.

Loss of function mutants can be modelled in `transsys` by adding a factor that represents a nonfunctional mutant of a protein and replacing the product of a gene with the nonfunctional factor. Gain of function mutants can be simulated by adding a high level of constitutive expression to the promoter of a gene.

Figure 5 shows the results of simulating both gain and loss of function of A, B and C. All mutations result in phenotype models that are qualitatively correct. The number of whorls is not correctly captured. This is due to the L-

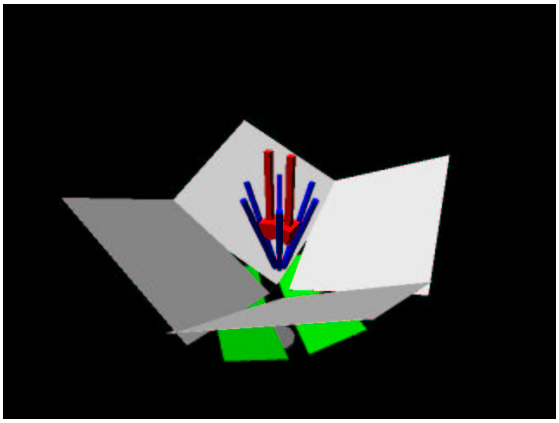

Figure 4. Model of a fully developed wild type flower grown under the control of the ABC `transsys` model depicted in Fig. 2. Sepals are rendered as small green squares, petals as large, white squares, stamens are depicted by blue cylinders and carpels are composed of red boxes. The grey cylinder at the bottom represents the stalk.

system component of the model which uses changes in $c_{\mathtt{a}}$, $c_{\mathtt{b}}$ and $c_{\mathtt{c}}$ to trigger generation of a new whorl, so multiple consecutive whorls with identical organ identities are collapsed into one.

4 Discussion and Outlook

Using the ABC model of flower morphogenesis as an example, it has been demonstrated that concise and expressive models of morphogenetic processes controlled by regulatory networks can be realized with `transsys`. Individual, localized mutations can be modelled by individual, localized alterations in the `L-transsys` model specification. This shows that `transsys` models are not only suitable for representing individual regulatory networks, but beyond that, are also capable of modelling the effects of mutations. Thus, adequation between nature and `transsys` models extends deeply into the genetic level. In this sense, `transsys` is a step towards modelling the fundamental principles underlying morphogenetic phenomena, following a traditional ALife modelling spirit of abstracting fundamental dynamical processes underlying life (see [11, preface]).

While the numerical parameters for activation and repression for the ABC model could simply be chosen *ad hoc*, the dependence on such numerical parameterization may become a problem when dealing with larger networks. This problem is currently being addressed by developing a qualitative or a semiquantitative mode of `transsys` dynamics, and by exploring numerical parameter space with evolutionary approaches, following the approach used in [13].

As concise and comprehensive representations which enable dynamical simulation and incremental modelling, `transsys` programs provide a suitable format

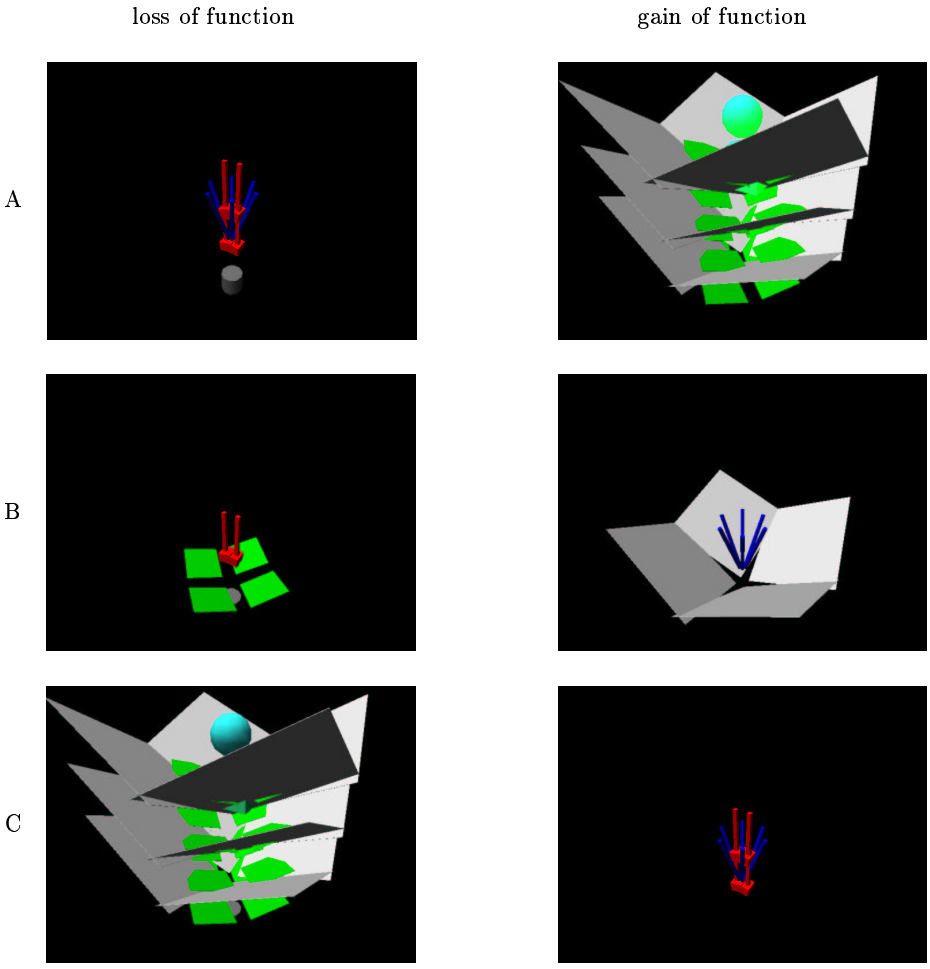

Figure 5. `L-transsys` models of flowers grown with mutant variants of the ABC network shown in Fig. 2. A sphere represents a meristem (where growth continues), other structures are explained in Fig. 4. Top row: mutants in A function, middle row: mutants in B function, bottom row: mutants in C function. Left column: loss of function mutants, right column: gain of function mutants. Loss of function mutants were modelled by replacing the product of the mutated gene with a factor called `nonfunc`, which does not interact with any promoters. Gain of function mutants were modelled by adding the promoter statement `constitutive: 500;` to the promoter of the mutated gene.

for representing and exchanging regulatory networks. As a start for building a collection of `transsys` programs, it is planned to implement some networks published in the literature (e.g. [13, 18]). Such a study will also serve to assess the level of universality attained with `transsys`, analogously to the analysis of universality recently conducted for the GROGRA growth grammar system

[9]. As an upcoming extension of `transsys`, integration of diffusion in a cellular lattice framework is currently being developed. This will allow modelling of reaction-diffusion processes, as exemplified by [10, 12].

5 Acknowledgements

I am grateful to Gerhard Buck-Sorlin, Winfried Kurth and Daniel Polani for critical and inspiring discussions regarding various aspects of `transsys`. I acknowledge financial support from the BMBF (Grant 0311378) during the initial phase of `transsys` development.

References

[1] Andrew Adamatzky. Simulation of inflorescence growth in cellular automata. *Chaos, Solitons & Fractals*, 7:1065–1094, 1996.

[2] Gerhard Buck-Sorlin and Konrad Bachmann. Simulating the morphology of barley spike phenotypes using genotype information. *Agronomie*, 20:691–702, 2000.

[3] Kurt Fleischer and Alan Barr. The multiple mechanisms of morphogenesis: A simulation testbed for the study of multicellular development. In Christopher G. Langton, editor, *Artificial Life III*, Santa Fe Institute Studies in the Sciences of Complexity, pages 379–416, Redwood City, CA, 1994. Addison Wesley Longman.

[4] Christian Fournier and Bruno Andrieu. A 3D architectural and process-based model of maize development. *Annals of Botany*, 81:233–250, 1998.

[5] Christian Jacob. Evolution programs evolved. In H.-M. Voigt, W. Ebeling, I. Rechenberg, and H.-P. Schwefel, editors, *PPSN-IV*, pages 42–51, Berlin, Germany, 1996. Springer Verlag.

[6] Jaap A. Kaandorp. Simulation of environmentally induced growth forms in marine sessile organisms. *Fractals*, 1:375–379, 1993.

[7] Jan T. Kim. LindEvol: Artificial models for natural plant evolution. *Künstliche Intelligenz*, pages 26–32, 2000.

[8] Winfried Kurth. 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. Technical report, Universität Göttingen, 1994. `ftp://www.uni-forst.gwdg.de/pub/biom/gro.ps.gz`.

[9] Winfried Kurth. Towards universality of growth grammars: Models of Bell, Pagés, and Takenaka revisited. *Ann. For. Sci.*, 57:543–554, 2000.

[10] Koji Kyoda and Hiroaki Kitano. A model of axis determination for the *Drosophila* wing disc. In Dario Floreano, Jean-Daniel Nicoud, and Francesco Mondada, editors, *Advances in Artificial Life*, Lecture Notes in Artificial Intelligence, pages 473–476, Berlin Heidelberg, 1999. Springer-Verlag.

[11] C.G. Langton, C. Taylor, J.D. Farmer, and S. Rasmussen, editors. *Artificial Life II*, Redwood City, CA, 1992. Addison-Wesley.

[12] H. Meinhardt. Biological pattern formation: New observations provide support for theoretical predictions. *BioEssays*, 16:627–632, 1994.

[13] Luis Mendoza and Elena R. Alvarez-Buylla. Dynamics of the genetic regulatory network for *Arabidopsis thaliana* flower morphogenesis. *J.theor.Biol.*, 193:307–319, 1998.

[14] Elliot M. Meyerowitz. The genetics of flower development. *Scientific American*, 271(5):56–65, 1994.

[15] Karl J. Niklas. Adaptive walks through fitness landscapes for early vascular land plants. *American Journal of Botany*, 84:16–25, 1997.

[16] S.W. Omholt, E. Plathe, L. Oyehaug, and K.F. Xiang. Gene regulatory networks generating the phenomena of additivity, dominance and epistasis. *Genetics*, 155:969–980, 2000.

[17] P. Prusinkiewicz and A. Lindenmayer. *The algorithmic beauty of plants*. Springer-Verlag, New York, 1990.

[18] Günter Theißen and Heinz Saedler. MADS-box genes in plant ontogeny and phylogeny: Haeckel's 'biogenetic law' revisited. *Current Opinion in Genetics and Development*, 5:628–639, 1995.

A The ABC network in `transsys`

```
transsys abc
{
  factor a { decay: 0.3; }
  factor b { decay: 0.3; }
  factor c { decay: 0.3; }
  factor bc { decay: 0.6; }
  factor bc_act { decay: 0.1; }
  factor cc { decay: 0.0; }

  gene agene
  {
    promoter
    {
      constitutive: 0.1;
      a: activate(0.000000001, 50.0);
      c: repress(50.0, 100.0);
    }
    product { default: a; }
  }

  gene bgene
  {
    promoter
    {
      constitutive: 0.00000001;
      b: activate(0.0003, 50.0);
      bc: repress(80.0, 5000.0);
    }
    product { default: b; }
  }

  gene cgene
  {
    promoter
    {
      cc: activate(0.5, 4.0);
      c: activate(10.0, 50.0);
      a: repress(1500.0, 20.0);
    }
    product { default: c; }
  }

  gene bcgene
  {
    promoter
    {
      bc_act: activate(1.0, 5.0);
      bc: repress(1.0, 1.0);
    }
    product { default: bc; }
  }

  gene bc_actgene
  {
    promoter
    {
      constitutive: 0.1;
      bc_act: activate(0.05, 3.0);
      bc: repress(0.1, 5.0);
    }
    product { default: bc_act; }
  }

  gene ccgene
  {
    promoter
    {
      constitutive: 0.01;
    }
    product { default: cc; }
  }
}
```