Preprint from Bioinformatics of Genome Regulation and Structure, ed. N. A. Kolchanov, Kluwer Publications. (2005, in press).

A SOFTWARE ARCHITECTURE FOR DEVELOPMENTAL MODELING IN PLANTS: THE COMPUTABLE PLANT PROJECT

Victoria Gor[1], Bruce E. Shapiro[1], Henrik Jönsson[2], Marcus Heisler[3], G. Venugopola Reddy[3], Elliot M. Meyerowitz[3] and Eric Mjolsness[4*]
[1]Jet Propulsion Laboratory, California Institute of Technology, Pasadena, CA 91109, USA; [1]email:victoria.gor@jpl.nasa.gov, bshapiro@caltech.edu;
[2]Department of Theoretical Physics, Complex Systems Division, Lund University, Lund, Sweden; [2]email: henrik@thep.lu.se;
[3]Division of Biology, California Institute of Technology, Pasadena, CA 91125, USA; [3]email mheisler, venu, or meyerow@caltech.edu;
[4]Institute for Genomics and Bioinformatics; [4]Bren School of Information and Computer Sciences, University of California, Irvine CA 92607, USA; [4]email:emj@uci.edu;
**Corresponding author*

Abstract: We present the software architecture of the Computable Plant Project, a multidisciplinary computationally based approach to the study of plant development. *Arabidopsis thaliana* is used as a model organism, and shoot apical meristem (SAM) development as a model process. SAMs are the plant tissues where regulated cell division and differentiation lead to plant parts such as flowers and leaves. We are using green fluorescent proteins to mark specific cell types and acquire time series of three-dimensional images via laser scanning confocal microscopy. To support this we have developed an interoperable architecture for experiment design that involves automated code generation, computational modeling, and image analysis. Automated image analysis, model fitting, and code generation allow us to explore alternative hypothesis *in silico* and guide *in vivo* experimental design. These predictions are tested using standard techniques such as inducible mutants and altered hormone gradients. The present paper focuses on the automated code generation architecture.

Keywords: *Arabidopsis*, Cellerator, correspondence, Delaunay triangulation, meristem, SAM, SBML, softassign, Voronoi diagram

1. Introduction

Scientists who probe the functionality of dynamic developmental systems often express their models mathematically; to make precise system-specific predictions these models are typically encoded with high-level computer languages and standard support libraries and solved numerically. However, high-level languages and libraries typically trade efficiency for generality, and thus may not be appropriate for large hybrid dynamical systems. They also typically lack state-of-the-art technologies in such computationally intensive areas as model optimization and fitting. Finally, custom designed systems are rarely interoperable, making it difficult for researchers to disseminate models.

We have developed an architecture aimed at production-scale model inference. We generate simulation code from models specified in biological

and/or mathematical language. Other computational tools are used to analyze expression imagery and other data sources, and the simulator combined with nonlinear optimization is used to fit the models to the experimental observations. Key elements include: a *mathematical framework* combining transcriptional regulation, signal transduction, and dynamical mechanical models; a *model generation package* (Cellerator) based on a computer algebra representation; *extensions to SBML* (Systems Biology Modeling Language), an exchangeable model representation format, to include dynamic objects and relationships; *a C^{++} code generator* to translate SBML into highly efficient simulation modules; *a simulation engine* including standard numerical solvers and plot capability; *a nonlinear optimize*r; and ad hoc *image processing* and *data mining* tools. This architecture is capable of simulating processes such as intercellular signaling, cell cycling, cell birth and death, dynamic cellular geometry, changing topology of neighborhood relationships, and the interactions of mechanical stresses.

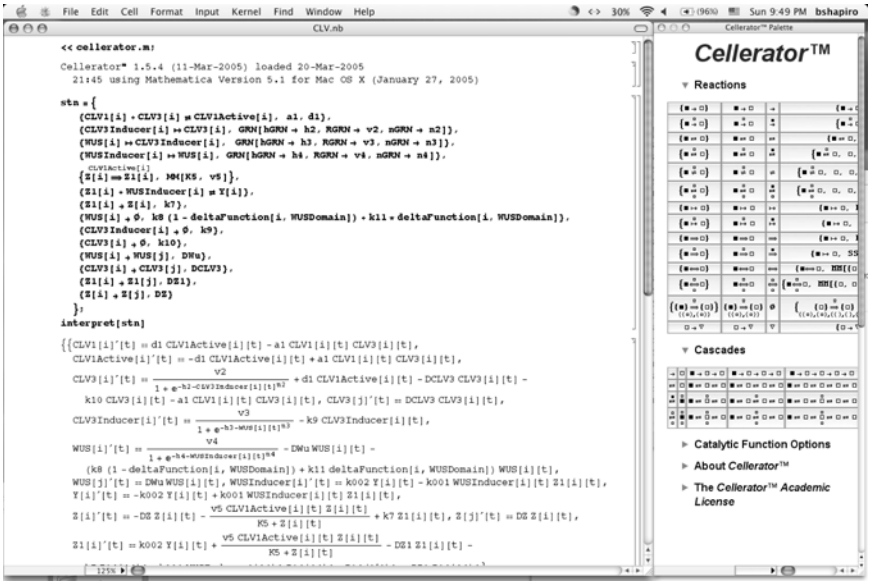

Figure 1. Cellerator screen shot. Signal transduction networks are built in the Mathematica notebook on the left by clicking on the palette on right. The interpret command converts the reactions to differential equations. Array indices are used to indicate different cells.

2. Methods and Algorithms

Models are input in Systems Biology Markup Language (SBML), an XML-based language for exchanging biological models. SBML is currently supported by more than seventy five different software packages used by biological modelers and has become the *de facto* standard for exchanging models among the systems biology community (Hucka et al 2003; Finney and Hucka 2003). The modeling interface is provided by *Cellerator* (Figure 1; Shapiro et al 2003), which allows users to specify models in an arrow-based biochemical notation, and translates them automatically to differential

equations using a variety of different schemes. *Cellerator* produces extended SBML Level 2 code utilizing *MathSBML* (Shapiro et al 2004). SBML encoded models are parsed into internal data structures with a *libSBML*-based parser (Finney et al, 2005).

Several extensions to SBML have been proposed and will likely be adopted in SBML Level 3 (Finney et al 2004). In particular, SBML Level 2 does not support spatially-dependent models where each biological entity is *individually defined and enumerated*, and further, does not provide any easy way to describe dynamic geometry and variable size models resulting from cell birth, death, and differentiation. Therefore we have adopted (Finney et al 2003) to describe dynamic topology and connectivity in terms of arrays, and have extended *Cellerator*, *MathSBML* and *libSBML* accordingly.

The *automatic code generator* is central to the architecture. It consists of an *inferencer*, a *rule segmenter and optimizer*, and *application code writer* modules (Figure 2). It queries the parser for SBML structures and produces efficient C++ application code. The resulting C++ code is then compiled into object code optimized for the desired application. The first two modules of the automatic code generator – the *inferencer* and *rule segmenter* – are pre-processors. They are called once for each SBML model, independent of the application software to be generated. The *inferencer* receives parsed SBML structures from the parser and infers element attributes given the element name. This reflects the inverse relationships between SBML elements and their attributes. For example, the extended SBML has a `parameter` attribute `foreach` that indicates the compartment; the *inferencer* creates a list of inferred elements, such as the list of parameters in each compartment.

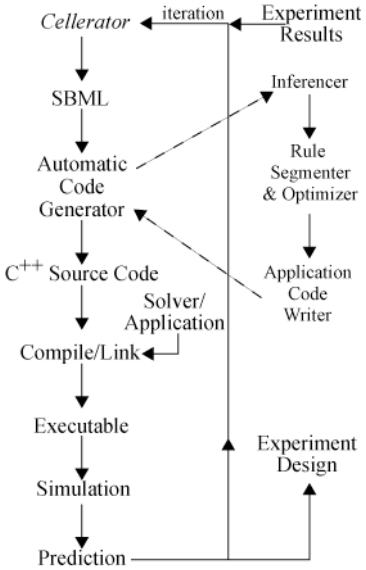

Figure 2. System Architecture.

The *rule segmenter and optimizer* translates SBML rules (which represent mathematical equations using a subset of MATHML) into C++ and performs all necessary renaming of SBML model objects into C variables. Portions of SBML formulas that have no immediate C++ representation, such as the MATHML function sum (which sums a formula over an index) are broken up into sub-rules with intermediate variables; these are later translated into loops or other appropriate control and data structures. Future enhancements will include formula optimization. Identical portions of the formula will be separated into intermediate rules that are only executed once; scalar formulas inside loops will be pre-evaluated outside of the loop. The renaming function completes the work of this module. For example, individual array elements are referenced by index with an SBML model utilizing the MATHML selector operator; this is replaced by the appropriate C array reference such as `name[j]`.

The *application code writer* takes as input the C++ model representation generated by the *rule segmenter and inferencer*, along with an application request, chosen from a menu of available applications. The output is application source code that can be compiled and linked with the chosen application. The *application code writer* consists of a three-level library. The top level contains all of the application-dependent code. This *application level software* is high-level code that is updated as new applications are added. Applications that exist or are being developed include various forward developmental simulators including genetic regulatory network (GRN) temporal synthesis; 4^{th} and 5^{th} order Runge-Kutta differential equation solvers; and optimizers such as Lam-Delosme simulated annealing. In addition, this top level includes overloaded routines that originate at the second level thereby allowing the top level to access this lower level functionality. The second level, *SBML level software*, contains all processes that are not application dependent. This library has entry points for accessing all SBML attributes and elements. The third, and lowest level, is the *utility library*, which contains common operations such as vector algebra and memory maintenance.

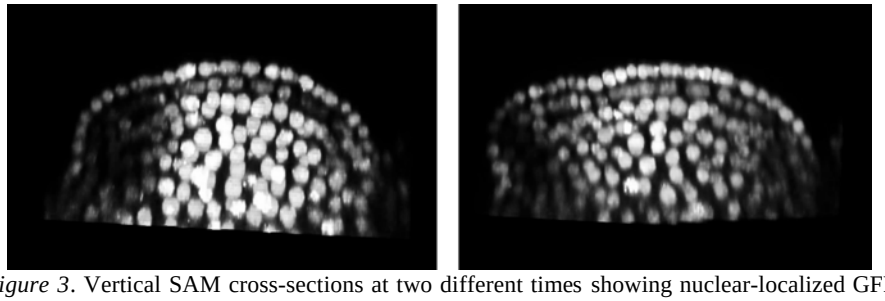

Figure 3. Vertical SAM cross-sections at two different times showing nuclear-localized GFP expressed from a ubiquitous promoter. A flower primordium is emerging at the upper left.

3. Results and Discussion

The SAM (Figure 3) is the plant tissue where regulated cell division and

differentiation lead to plant parts such as flowers and leaves. We are using green fluorescent proteins to mark specific cell types and acquire time series of three-dimensional images via laser scanning confocal microscopy. The three-dimensional reconstruction starts from "stacks" of horizontal sections (Figure 4). Such sections are combined to produce four-dimensional visualizations (3 spatial dimensions plus time) using various programs we have developed.

In any 3D image stack there is a correspondence problem: which cells in one image correspond to which cells in the adjacent cross-section? Cell membranes that are transverse to the image are clearly visible, but it is possible to miss nearly horizontal walls that lie between sections and must be inferred. With a time-course of 3D stacks the formation of floral meristems, cell growth, displacement, and division all complicate the problem. Nuclear locations are determined using a 3D gradient descent algorithm based on image intensity.

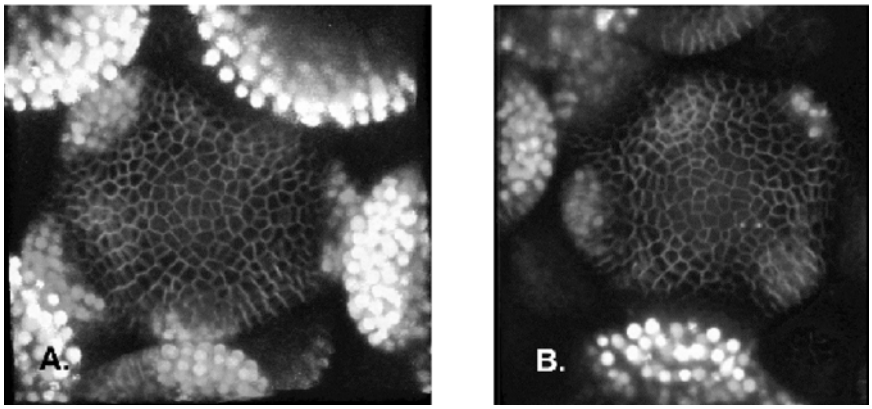

Figure 4. Horizontal SAM cross-sections, showing pPIN1:PIN1GFP (expressed in the cell membranes) in combination with pFIL:dsREDN7 (nuclear, seen here primarily in the primordia) at two time points 33 hours apart; also illustrating the budding of new floral meristems (A: initial view, B: final view).

Automated extraction of cell walls (or cell membranes) is more complicated, as we have discussed. It is possible to estimate their locations using the Voronoi diagram (also called a Dirichlet tessellation) of the nuclear centers; nearest-neighbor links are then given by the corresponding Delaunay triangulation (figure 5). The Voronoi "cell" defined by any nuclear center **p** is the polygon that contains all of the points that are closer to **p** then to any other nuclear center **q**; the Voronoi diagram is the collection of all such Voronoi cells. The Delaunay triangulation is the dual of the Voronoi diagram, namely the collection of lines drawn from each nucleus to all of its nearest neighbors. The walls of the Voronoi diagram are the perpendicular bisectors of the Dalaunay triangulation. This principle that cell walls are equidistant from nuclei is beautifully reflected by the watershed transform (Vincent and Soille, 1991), an image segmentation algorithm based on

mathematical morphology (Sternberg, 1986). The watershed transform is used in combination with Voronoi diagrams for detecting cell walls in images. When the walls are visible, they are detected form the gradient of image data; and when the walls are not visible, they are inferred by the Voronoi diagram (figure 6). Voronoi diagrams and Delaunay triangulations are computed with Qhull (open source software, www.qhull.org).

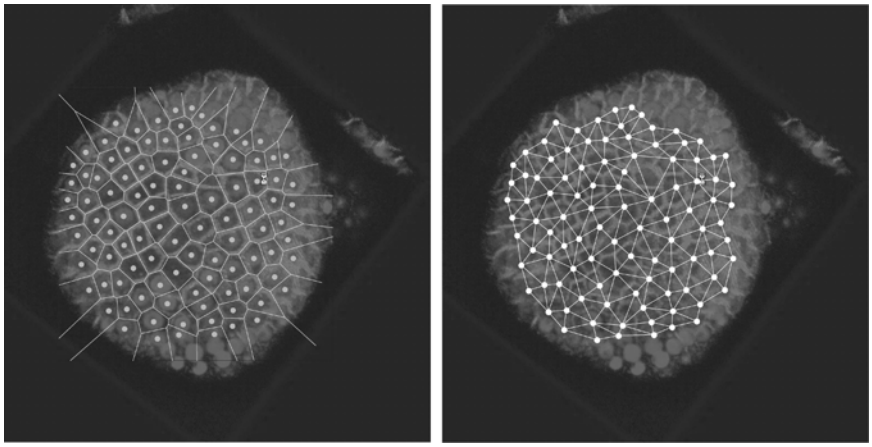

Figure 5. Meristem cross-section stained to show cell membranes and nuclei, superimposed with manually tagged cell-centers and the corresponding Voronoi diagram (left) and Delaunay Triangulation (right). A small number of cell centers were left unmarked; the corresponding Voronoi cells were added to the adjacent cells.

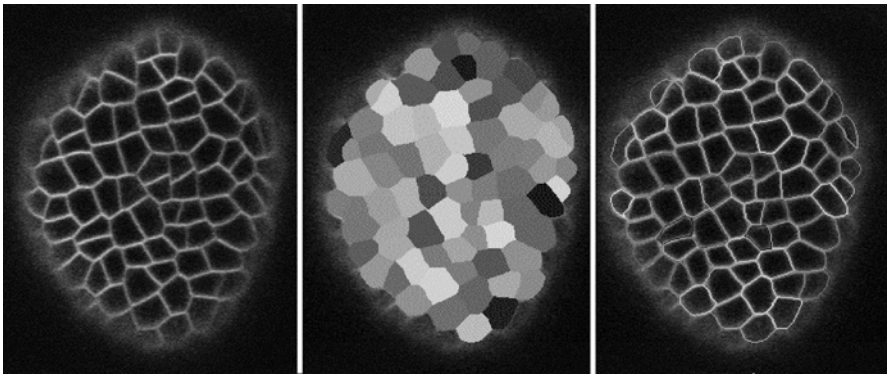

Figure 6. Raw image (left), segmentation of image into cells (center), and cell walls determined with a watershed algorithm (right).

Determining which cells in one 3D image correspond to which cells in the next image is a case of the classic correspondence problem (Post, 1947); many solutions to both point-matching and graph-matching correspondence problems have been published to this extremely difficult problem. Recent work is based on joint estimation of correspondence and spatial mappings via optimization of an energy function (Gold et al 1996). The general framework

uses the method of deterministic annealing in conjunction with the softassign algorithm and clocked objectives to produce an optimizing network and a corresponding energy function (Gold et al 1996; Koslowsky and Yuille, 1994; Mjolsness and Miranker, 1998). The specific energy function used for cell tracking determines cell correspondence, while estimating the mapping functions, such as affine and thin-plate spline transformations (Chui and Rangarajan, 2000) and cell growth and division history.

Table 1. A selection of typical Cellerator arrows.

Arrow	Description
$p_1A_1 + p_2A_2 + \cdots \rightarrow q_1B_1 + q_2B_2 + \cdots$ $p_1A_1 + p_2A_2 + \cdots \rightleftarrows q_1B_1 + q_2B_2 + \cdots$	Law of mass action, one-way and reversible; optional stoichiometry
$S \overset{E}{\rightleftarrows} P$	Enzymatic mass action, same as $S + E \rightleftarrows SE \rightleftarrows P + E$
$S \underset{F}{\overset{E}{\rightleftarrows}} P$	Reversible enzymatic mass action $S + E \rightleftarrows SE \rightleftarrows P + E$ and $P + F \rightleftarrows PF \rightleftarrows S + F$
$S \underset{F}{\overset{E}{\rightleftharpoons}} P$	Reversible enzymatic mass action $S + E \rightleftarrows SE \rightleftarrows PE \rightleftarrows P + E$ $P + F \rightleftarrows PF \rightleftarrows SF \rightleftarrows S + F$
$S \Rightarrow P,\ S \overset{E}{\Rightarrow} P, S \underset{F}{\overset{E}{\Rightarrow}} P$	Michaelis-Menten Kinetics; one-way and reversible
$S \overset{E}{\mapsto} P$	Conversion of *A* to *B*, facilitated by *E*, via Hill function
$A \mapsto B$	Regulation of *B* (*A* unaffected) by Hill function, sigmoid, NHCA, or S-System
$\underset{\{\{A_1,A_2,\ldots\},\{I_1,I_2,\ldots\}\}}{\{S_1,S_2,\ldots\}} \overset{E}{\Rightarrow} \{P_1,P_2,\ldots\}$	Generalized MWC with multiple substrates, products, activators, and inhibitors
$\underset{\{\{A_1,A_2,\ldots\},\{I_1,I_2,\ldots\},\{\{Q_1,Q_2,\ldots\},\{R_1,R_2,\ldots\}}{\{S_1,S_2,\ldots\}} \overset{E}{\Rightarrow} \{P_1,P_2,\ldots\}$	Generalized MWC with competitive inhibition

We are using this simulation environment to extend and enhance our previously reported developmental simulations of the shoot apical meristem (SAM) (Jönsson et al 2003, 2005; Mjolsness et al 2004). Our working hypothesis is that SAM development can be described by the differential expression of key regulatory proteins such as CLV1 (a receptor kinase), CLV3 (thought to be the CLV1 ligand),WUS (a transcription factor negatively regulated by CLV1) and a layer-1 specific protein (L1SP). For example, activation of CLV1 might be described by the reactions

$$CLV1 + CLV3 \rightleftarrows CLV1^*$$
$$CLV3 \rightarrow \varnothing$$

The dependence of CLV1 and CLV3 on WUS, perhaps through a hypothetical diffusible intermediary (CLV3I1), has been inferred from experiments. A second diffusive signal is postulated to originate from L1SP and diffuses into the rest of the meristem via messenger CLV3I2. CLV3 is

turned on only if the sum CLV3I1+CLV3I2 exceeds threshold.

$$WUSI \mapsto WUS$$
$$WUS \mapsto CLV3I1$$
$$L1SP \mapsto CLV3I2$$
$$CLV3I1 + CLV3I2 \mapsto CLV3$$

where WUSI is a hypothetical WUS inducer that originates either in or below the corpus. The expression $A \mapsto B$ is the Cellerator notation for genetic regulation (see Table 1). Inhibitory feedback is provided by a proposed entity Z that sequesters activated CLV1, and when activated, sequesters or removes WUSI:

$$Z + CLV1^* \rightleftarrows Z1$$
$$Z1 + WUSI \rightleftarrows Z2$$

Additionally, an unknown diffusible messenger Y creates a surface specific expression pattern for L1SP, which is itself inhibited by STEM, a hypothetical gene expressed only in the lowest meristem layer:

$$Y \mapsto L1SP$$
$$STEM \mapsto Y$$

Here the first reaction is activating, and the second is inhibitory; both genetic regulation and inhibition are modeled by Hill functions with different parameters. To maintain homeostasis we include the reactions

$$CLV3, CLV3I1, WUS \rightarrow \varnothing$$

and describe diffusion using a simple compartmental approach. A Cellerator model for a single cell that is very similar to this one is illustrated in figure 1. A two-dimensional 133-cell Cellerator implementation has 5422 reactions and 1596 differential equations.

The computable plant architecture provides a systematic, highly automated technique for predictive model generation. The approach combines computer-algebraic representations of biological and mathematical models to produce efficient and problem-specific simulation code. This code can be immediately linked with a menu of external solvers and quantitative predictions generated from the resulting simulations. This architecture is scalable and directly applicable to large-scale developmental systems such as the SAM. The use of extended SBML ensures that models will be interoperable, reusable, and readable by others. Novel to this approach are connections to external solvers by way of automatic code generation and the

ability to interpret and solve any biological developmental or cellular process via automatic generation of mathematical and computational tools. Thus no labor is expended writing and debugging problem-specific code, allowing researchers to spend more time on the wet bench. Further details can be found at the project web-site, http://www.computableplant.org.

Acknowledgments

This work was supported by the United States National Science Foundation (NSF) under Frontiers in Integrative Biological Research (FIBR) grant number EF-0330786. HJ was supported, in part, by the Knut and Alice Wallenberg Foundation through Swegene. Portions of the research described in this paper were performed at the California Institute of Technology.

References

Chui, H. and Rangarajan, A. 2000. A new algorithm for non-rigid point matching. *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, **2**:44-51.

Finney, A. and Hucka, M. (2003) Systems Biology Markup Language: Level 2 and Beyond. *Biochem. Soc. Trans*, **31**: 1472-1473.

Finney, A., Gor, V., Bornstein, B., and Mjolsness, E. 2003. *Systems Biology Markup Language (SBML) Level 3 Proposal: Array Features*, http://www.sbml.org/wiki/arrays.

Finney, A., Hucka. M., Bornstein, B.J., Keating, S., Shapiro, B.E., Matthews, J., Kovitz, B., Funahashi, A., Schilstra, M., Doyle, J.C., and Kitano, H. 2004. Evolving a Lingua Franca and Accompanying Software Infrastructure for Computational Systems Biology: The Systems Biology Markup Language (SBML) Project. *IEE Systems Biology*. **1**(1):41-53.

Finney, A., Hucka, M., Bornstein, B., Keating, S., Shapiro, B.E., Matthews,J., Kovitz, B., Schilstra, M., Funahashi, A., Doyle, J., and Kitano, H. 2005. Software Infrastructure for Effective Communication and Reuse of Computational Models. In System modeling in cellular biology: From concepts to nuts and bolts, ed. Szallasi Zoltan, Vipul Periwal, Joerg Stelling, in press.

Gold, S., Lu, C-P., Rangarajan, A., Pappu, S., and Mjolsness, E. 1995. New algorithms for 2D and 3D point matching: pose estimation and correspondence. *Adv. In Neural Information Processing Systems*, **7**:957-964.

Gold, S. and Rangarajan, A. 1996. Softmax to Softassign: Neural network algorithms for combinatorial optimization. *J. Artificial Neural Networks* **2**(4):384-399.

Koslowsky, J.J and Yuille, A.L. 1994. The invisible hand algorithm: solving the assignment problem with statistics physics. *Neural Networks* **7**:477-490.

Hucka, M., Finney, A., Sauro, H.M., Bolouri, H., Doyle, J.C., Kitano, H., Arkin, A.P., Bornstein, B.J., Bray, D., Cornish-Bowden, A., Cuellar, A.A., Dronov, S., Gilles, E.D., Ginkel, M., Gor, V., Goryanin, II., Hedley, W.J., Hodgman, T.C., Hofmeyr, J.H., Hunter, P.J., Juty, N.S., Kasberger, J.L., Kremling, A., Kummer, U., Le Novere, N., Loew, L.M., Lucio, D., Mendes, P., Mjolsness, E.D., Nakayama, Y., Nelson, M.R., Nielsen, P.F., Sakurada, T., Schaff, J.C., Shapiro, B.E., Shimizu, S., Spence, H.D., Stelling, J., Takahashi, K., Tomita, M., Wagner, J., and Wang, J. 2003. The systems biology markup language (SBML): a medium for representation and exchange of biochemical network models. *Bioinformatics* **19**:513-523

Jönsson, H., Shapiro, B.E., Meyerowitz, E.M., and Mjolsness E. 2003. Signaling in Multicellular Models of Plant Development, in *On Growth, Form, and Computers*, ed. Bentley P and Kumar S, Academic Press, pp.156-161.

Jönsson H, Heisler, M., Reddy, G.V., Agrawal, V., Gor, V., Shapiro, B.E., Mjolsness, E., and Meyerowitz, E.M. 2005. Modeling the organization of the WUSCHEL expression domain in the shoot apical meristem, *Bioinformatics*, in press.

Mjolsness, E., and Miranker, 2. 1998. A Lagrangian Formulation of Neural Networks II: Clocked Objective Functions and Applications. *Neural, Parallel, and Scientific Computations*. **6**(3): 337-372.

Mjolsness, E., Jönnson, H., Shapiro, B.E., and Meyerowitz, E.M. 2004. Modeling plant development with gene regulation networks including signaling and cell division, in *Bioinformatics of Genome Regulation and Structure*, ed. N. A. Kolchanov, Kluwer Publications, pp. 311-318.

Post, E., L., A variant of a recursively unsolvable problem, 1946. *Bull. Am. Math. Soc.*, **52**:264-268.

Shapiro, B.E., Hucka, M., Finney, A., and Doyle, J. 2004. MathSBML: A package for manipulating SBML-based biological models. *Bioinformatics*. **20**(16): 2829-2831.

Shapiro, B.E., Levchenko, A., Wold, B.J., Meyerowitz, E.M., and Mjolsness, E.D. 2003. Cellerator: Extending a computer algebra system to include biochemical arrows for signal transduction modeling. *Bioinformatics* **19**: 677-678.

Sternberg, S.R. 1986. Grayscale Morphology, *Computer Vision ,Graphics, Image Processing.* **35**(3): 333-355.

Vincent, L. and Soille, P. 1991. Watersheds in digital spaces: an efficient algorithm based on immersion simulations. *IEEE Trans. Patt. Anal. Mach. Intel.* **13**(6): 583-591.