JOURNAL OF EXPERIMENTAL ZOOLOGY (MOL DEV EVOL) 294:216–251 (2002)

Ingeneue: A Versatile Tool for Reconstituting Genetic Networks, With Examples From the Segment Polarity Network

ELI MEIR,[1†] EDWIN M. MUNRO,[1] GARRETT M. ODELL,[1] AND GEORGE VON DASSOW[1,2*†]
[1]*Department of Zoology, University of Washington, Seattle, Washington 98105*
[2]*Friday Harbor Laboratories, Friday Harbor, Washington 98250*

ABSTRACT Here we describe a software tool for synthesizing molecular genetic data into models of genetic networks. Our software program Ingeneue, written in Java, lets the user quickly turn a map of a genetic network into a dynamical model consisting of a set of ordinary differential equations. We developed Ingeneue as part of an ongoing effort to explore the design and evolvability of genetic networks. Ingeneue has three principal advantages over other available mathematical software: it automates instantiation of the same network model in each cell in a 2-D sheet of cells; it constructs model equations from pre-made building blocks corresponding to common biochemical processes; and it automates searches through parameter space, sensitivity analyses, and other common tasks. Here we discuss the structure of the software and some of the issues we have dealt with. We conclude with some examples of results we have achieved with Ingeneue for the *Drosophila* segment polarity network. *J. Exp. Zool, (Mol. Dev. Evol.) 294:216–251, 2002.*

The advent of whole-genome sequencing, DNA microarrays, and proteomics promises an imminent embarrassment of riches. Biologists are accumulating a phenomenal amount of information about genes, their functions, and their interactions. Soon, if not already, the available maps of known genetic interactions for any particularly well-studied cell physiological or developmental process will be so complex as to defy the ability of human brains to understand and manipulate those maps without help from computers. Most genes can be said to have a ''function'' only as constituent parts of networks of cross-regulatory and biochemical interactions with other genes and their products. Increasingly, biologists think in terms of whole networks, the biological analogue of the integrated circuit. The network, rather than any individual gene, is the causal unit that does useful things such as transduce, transmit, or transmute signals, stabilize cell states, form expression patterns in groups of cells, etc. This is particularly evident for many paradigmatic developmental mechanisms in which genetic networks produce patterns in space (e.g., the segmentation cascade in *Drosophila*) or time (e.g., cell cycle oscillator or the circadian clock).

There is growing interest in using computers to synthesize genetic data into mechanistic models at the network level. Several inter-related goals motivate that interest:

- For some biological process of interest (e.g., bacterial chemotaxis, *Drosophila* segment boundary formation, the cell cycle, etc.), is the known map of interactions between molecular components complete enough to actually explain that phenomenon?
- If so, what systems-level properties, unanticipated from the nature of the parts themselves, emerge in the whole network?
- What rules, if any, govern ''design'' of genetic networks, and which details are crucial to making mechanisms that work?
- How does network architecture constrain or facilitate the interaction between evolutionary and developmental processes?

A natural approach for making computer models is to represent a network of interacting genes as a set of coupled ordinary differential equations

Grant sponsor: National Science Foundation; Grant numbers: MCB-9732702, MCB-9817081, MCB-0090835.
†Equal contributors.
*Correspondence to: George von Dassow, Friday Harbor Laboratories, Friday Harbor, WA 98250. E-mail: dassow@u.washington.edu
Received 19 September 2001; Accepted 10 January 2002
Published online in Wiley InterScience (www.interscience.wiley.com). DOI: 10.1002/jez.10145

(ODEs) and then integrate the system of equations over time to determine how the network behaves (Edgar et al., '89; Slack, '83; Tyson et al., '96; Barkai and Leibler, '97; Bray et al., '98; Laub and Loomis, '98). (For a selection of alternate approaches see Kauffman, '93; McAdams and Shapiro, '95; Reinitz and Sharp, '95; Thieffry et al., '98; Barkai and Leibler, 2000.) Although excellent general-purpose computer programs exist for solving ODEs (e.g., Mathematica, Maple, or Matlab) we find them unwieldy for genetic networks. For instance, a simple representation of the segment polarity network in *Drosophila* (see below) involves 13 different components, operating across at least 4 cells, with the dynamics of the network governed by 48 free parameters and including ≈ 140 coupled equations. Using a standard mathematics package, the task of constructing the model is slow and error-prone and requires a degree of mathematical and programming sophistication which most lab-bench biologists do not possess.

This paper describes the computer program Ingeneue, which we wrote specifically to construct models of gene networks and explore their pattern formation repertoire. Ingeneue uses a library of stereotyped, biologically meaningful building blocks to assemble models. This makes it straightforward to translate network diagrams, such as those that often appear as the last figure in molecular genetics articles, into systems of ODEs. Having assembled a model, Ingeneue imposes a user-specified initial pattern and then integrates the ODEs to find the temporal and spatial patterns it produces over time. Ingeneue can search for combinations of parameter values that confer on a network the ability to make particular target patterns. Once sets of ''working'' parameter values are discovered, Ingeneue helps one test how sensitive the model is to changing those values. Ingeneue is highly modular and is designed to be extended with a minimum of effort. Most features can be accessed through a point-and-click interface. Software like Ingeneue allows biologists with minimal mathematical training to make and explore models of their own networks and, if widely adopted, will foster a kind of standardization that will make the results of such studies mutually intelligible.

We have used Ingeneue to explore two patterning networks in *Drosophila*: the segment polarity network (von Dassow et al., 2000; this paper; and companion paper by von Dassow and Odell, 2002, this issue) and the neurogenic/proneural network (Meir et al., 2002). Our results confirm the usefulness of modeling at the network level, both as a tool for testing the plausibility of proposed mechanisms and as a way of revealing network-level properties that would not otherwise come to light. Our initial model of the segment polarity network attempted to reconstitute the real network's behavior using only the best-understood players and the best-documented connections among them. Using Ingeneue we found this model was completely incapable of mimicking the known expression patterns of segment polarity genes. In order to make our model work we needed to add two more interactions, one of which was well documented but whose importance was not fully clear beforehand, and another, which, frankly, was a guess, suggested only by circumstantial evidence. This result highlights the need for a way to check, formally, whether the current understanding of a genetic network is complete, and if not, to develop hypotheses for what pieces may be missing. Once patched up, we discovered that our model was astonishingly robust to changes in both the parameters that govern the kinetics of component molecules and the initial pre-pattern (von Dassow et al., 2000). This is an empirically testable, network-level property that we doubt could have been intuited from the known facts about the individual segment polarity genes and their interactions but which has very interesting theoretical and practical implications.

Here we describe both the design of Ingeneue as well as its capabilities and interface. Ingeneue is a work-in-progress that we, its original users, extend as we encounter new needs. Our goal in writing this paper is not only to introduce Ingeneue but also to summarize the lessons we have learned in developing it and offer our ideas to others developing similar software. Detailed information, including source code and tutorials, is available with the program online at http://www.ingeneue.org. In order to exemplify the use of Ingeneue we will follow a model of the segment polarity network in *Drosophila* throughout this paper. The segment polarity network in reality consists of dozens of genes and their products whose earliest, fundamental function is to maintain parasegmental boundaries and provide positional information within each segment during embryogenesis in *Drosophila*. Our initial minimal model explicitly employs just five of those genes and proposes to account only for how this network stably maintains a boundary, as shown in Figure 1 (fully described in von Dassow et al., 2000). Here

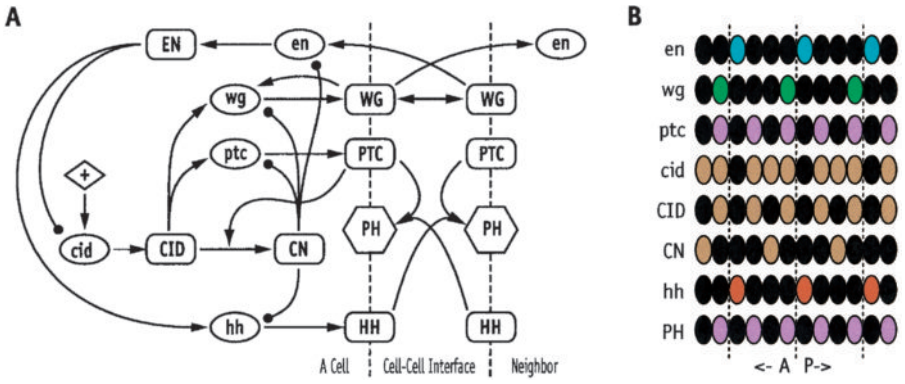


Fig. 1. Segment polarity network and the target pattern. (**A**) Core of the segment polarity network in *Drosophila*; this is a ''minimal subset'' that we used in all work described in this paper and in von Dassow et al. (2000). (**B**) The pattern we asked the network to achieve and to maintain stably. Both the network diagram and pattern are based on our synthesis of the work of hundreds of researchers on these genes, and while greatly simplified relative to the real network, they are adequate to capture the most essential function of the segment polarity network. A description of **A** with references to the primary work is contained in the Supplement to von Dassow et al. (2000), along with a description of how we translated the network map into mathematical expressions.

we illustrate Ingeneue's ability to explore parameter spaces by providing additional results on the effects of diffusion and cooperativity on this model.

OVERVIEW OF INGENEUE

Ingeneue is written using the object-oriented language Java. Object-oriented programming means dividing a program into classes of ''objects,'' where each object knows its own state, how to perform its own behaviors, and how to interact with other objects. This paradigm suits many biological problems because biological systems tend to divide naturally into distinct objects (e.g., species, individual organisms, genes, neurons, etc.). Thus, Ingeneue consists of several interdependent, extensible libraries of object types. Some of them encapsulate representations of biological entities or their properties, whereas others are tools for manipulating those objects or conducting numerical analyses. Java is rapidly gaining favor as a language for constructing scientific software, both because it is well-designed and also because a program (including graphical interface) written in Java runs on all kinds of computer hardware and operating systems without any changes. We find this to be true in practice as well as in theory. Early in the evolution of the Java language and platform, Java programs suffered a serious performance deficit compared to programs written using more traditional languages such as C/C++ or FORTRAN. However, by now very efficient Java runtime environments are available for the most commonly used computer platforms, and our experience is that numerical routines written in Java compete with similar routines coded in C or C++.

Ingeneue's core construction and analysis module creates a system of ordinary differential equations (and their initial conditions) from a textual description of the network that the user writes, and integrates those equations over time. A graphical interface then allows the user to modify quantitative (but not yet topological) properties of the network and view its behavior. The core can run without the interface so one can run the program remotely, e.g., as a batch job on a Unix server. Ingeneue represents a genetic network using three types of objects: Nodes, Cells, and Affectors (Fig. 2). Nodes are the network components, such as mRNAs, proteins, and protein complexes; these are the dependent variables in the model's ODE system. Each Node tracks temporal change in the concentration of a network component within a single cell or cell compartment. Since gene copy number does not change over time we do not include a Node to track DNA. For reference, the minimal segment polarity network (Fig. 1) requires 13 Nodes per Cell: five mRNAs, three directly transcribed intracellular proteins, one protein fragment produced by cleavage of the Cubitus interruptus protein, three cell-surface proteins and one cell-surface protein complex between HH and PTC (cell-surface Nodes track six concentrations, one for each Cell face). A 14th ''dummy'' Node provides a basal

transcriptional input to the Node ci (which represents the *cubitus interruptus* mRNA). In Figure 1, Nodes are represented by oval icons for mRNAs, rectangular icons for directly transcribed proteins and their cleavage products, and hexagonal icons for dimers/polymers of those proteins.

Each cell in an epithelial sheet corresponds to one of Ingeneue's Cell objects, which stores a complete set of that Cell's Nodes in the network along with the identities of each neighboring Cell. Epithelial layers in developing *Drosophila* embryos (and many other species as well) generally consist of polygonal cells with roughly six sides. The issue of how cell shape affects patterning is worth exploring, but currently, for simplicity, we represent Cells in Ingeneue as two-dimensional regular hexagons that do not move. No constraint inherent in Ingeneue's design prevents us from eventually adding a more sophisticated representation of cells, sheets of cells, movement of cells within sheets, and mitoses. Each Ingeneue Cell has seven compartments: a ''cytoplasmic/nuclear'' compartment, where all intracellular Nodes reside, and six ''membrane'' compartments, one for each side of the cell. Ingeneue tracks concentrations of membrane-bound components (for instance trans-membrane ligands and receptors such as WG, PTC, HH, and PH in Fig. 1) separately for each cell face, and each pair of neighboring cell faces interact independently. Ingeneue allows these membrane-bound components to flow from each side of a cell to the two neighboring sides of the same cell. Ingeneue also allows exchange between apposite faces of neighboring cells. Together, these two features allow molecules to ''diffuse''[1] across the sheet of cells (Fig. 2, middle panel). Ingeneue can easily accommodate other notions of cell compartmentalization which certain other applications may require.

Ingeneue uses arrays of hexagonally packed cells (Fig. 2). In the current version, boundaries wrap around both horizontally and vertically so, for instance, the left-hand neighbor of a cell

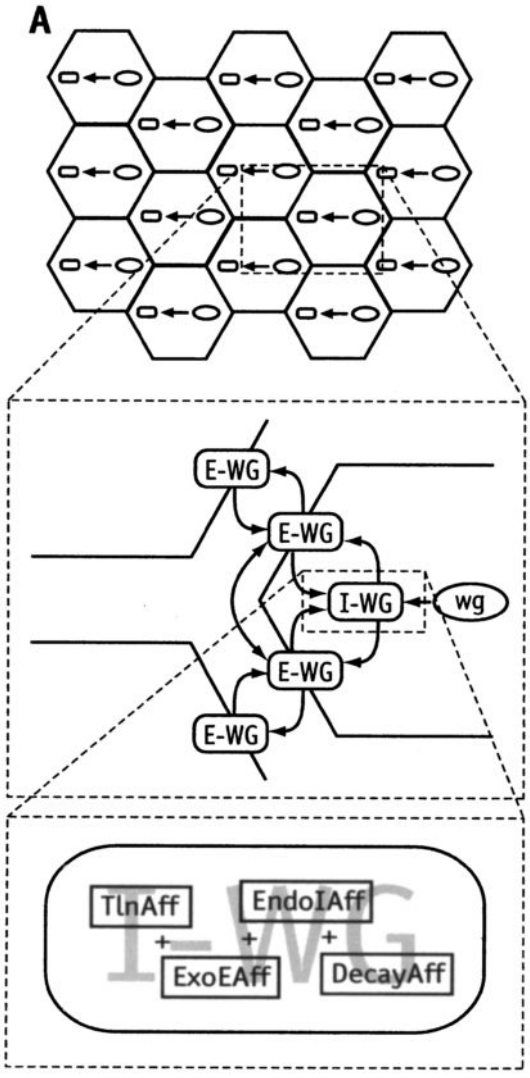


Fig. 2.

[1]Ingeneue does not do ''real'' diffusion, that is, concentration-dependent flux governed by a partial differential equation parameterized by D, hence the quotation marks. However the face-to-face exchange mechanism corresponds to a coarse discretization of a flux equation, and serves adequately for any purpose we have yet encountered. In any case, we doubt that free diffusion, per se, is realistic for most molecules expressed in embryonic tissues of animals; rather, most secreted proteins probably associate with the cell surface and extracellular matrix to greater or lesser degrees, and thus Ingeneue's mechanism seems to us more satisfying. It is also more useful because it subsumes all the buzzwords: autocrine, paracrine, and juxtacrine signaling, as well as free diffusion, can be achieved within different parameter regimes.

on the left edge is a cell on the right edge. Our wrap-around boundaries, which give our cell sheets the topology of a torus, remove boundary effects. This is an advantage for spatially periodic patterns such as those the segment polarity network typically makes. In cases where wrap-around boundaries are inappropriate, we add an extra strip of ''dead'' cells around the edge of the sheet with initial conditions that cause them not to participate in the patterning process being modeled. The initial segment polarity pattern is composed of stripes, perpendicular to the anterior–posterior axis, that repeat every four cells, so for the crudest applications using this network we can get away with a 4×1 rectangle of cells.

A major advantage of using Ingeneue for genetic network modeling is that the user need specify the network in a typical cell only once. Ingeneue does the drudgery of instantiating the network into each Cell in the model and setting up the appropriate relations between neighboring cells. Thus in Figure 2, there are only three components we need to specify: wg (mRNA) and I-WG (protein) in the intracellular compartment; and E-WG (protein) in the cell surface compartment. There are four interactions: wg$\rightarrow$I-WG (translation); I-WG$\leftrightarrow$E-WG (bi-directional exchange between compartments); the flux rate of E-WG around the cell periphery; and the flux rate of E-WG from cell to cell. This is much faster and more reliable than enumerating the ODE system by hand, as one would have to do absent a tool like Ingeneue: imagine typing out all 120 ODEs (with all connections represented by 870 additive terms, carefully indexed to appropriate Cell and Cell face) required for the 15-cell grid at the top of Figure 2! For a realistic problem, such as the minimal segment polarity model in a mere 4×1 cell grid, the use of a standard symbolic mathematics package would require the user to type out a system of nearly 140 coupled ODEs. Ingeneue reduces this to 13, uses stereotyped building blocks, Affectors, to construct the differential equations, and can expand that 4×1 grid to arbitrary size by editing just two numbers in the input file.

Affectors represent the interactions between Nodes. Each Affector encapsulates a formula corresponding to a physical process involving one or more Nodes. Each Node computes the rate of change in its concentration (its time derivative) by combining the values all its Affectors contribute. Table 1 illustrates the four conceptual groups that most of Ingeneue's Affectors fall into. Some Affectors govern synthetic processes such as transcription and translation. A second group governs non-specific first-order decay of each molecule. A third group represents transformations of proteins from one form to another (via reactions such as cleavage and hetero-dimerization), some of which may be reversible. The fourth group mediates exchange of molecules between different compartments within and among cells. Ingeneue currently includes approximately 60 Affectors (all in dimensionless form, as described in the supplement to von Dassow et al., 2000; future editions will include complementary dimensional-form building blocks) that we developed to construct our segment polarity and neurogenic network models. These provide a versatile, expanding basis for constructing models of similar ''resolution'' and complexity to our segment polarity model. Furthermore, it is a simple Java programming task to create a new Affector type, especially given the existing library as exemplars. This is a key feature of our design goals for Ingeneue: it should be easy for a biologist possessing an acquaintance with the mathematics and only a little familiarity with

Fig. 2 is on page 219

Fig. 2. Pieces of an Ingeneue model. (**A**) Ingeneue models are made of Cells, Nodes, and Affectors. Cells are hexagonal with one cytoplasmic compartment and six membrane compartments. Cells are arranged in a grid, with each face of a Cell in contact with a face of one of its neighbors. Cell faces on the edges of the grid are wrapped around as if on a torus to be in contact with ''neighbors'' on the opposite side of the grid. All Cells contain identical copies of a network, where the network is composed of Nodes (that is, molecular species; ovals and polygons in middle panel) and Affectors (arrows in middle panel, boxes in lower panel). The middle panel shows a subset of the Nodes and Affectors from the segment polarity network. Transcription of the wg mRNA produces an internal WG protein pool (I-WG). This internal WG pool exocytoses onto each face of the Cell (E-WG), from whence it can exchange to the apposite faces of neighboring cells or to neighboring faces of the same cell. Endocytosis transfers E-WG, at a rate proportional to its concentration on each Cell face, to I-WG. Combined, these exchange processes subsume, in different regions of parameters space, autocrine, paracrine and juxtacrine signaling, transcytosis, and even ''free diffusion.'' The bottom panel shows the four Affectors that are summed together to compute the time rate of change in I-WG concentration. They represent, from left to right, translation of I-WG from wg mRNA, exocytosis to the membrane, endocytosis from the membrane, and non-specific decay. (**B**) The ''sheet-of-hexagons'' approximation is not so unrealistic; in many developing embryos, such as the neurula-stage ascidian embryo shown here in a scanning electron micrograph provided by G. von Dassow, pattern formation takes place in epithelial sheets in which the cells are roughly hexagonally packed.

TABLE 1. Ingeneue's classes of affectors[1]

Affector type	Description	Examples
Synthesis	Transcription of mRNAs and translation of mRNAs into proteins	$\frac{T_o}{H_x}\left(\frac{Y^{\nu_{Yx}}}{K_{Yx}^{\nu_{Yx}}+Y^{\nu_{Yx}}}\right)$ $\frac{T_o}{H_x}\left(\frac{A^{\nu_{Ax}}}{K_{Ax}^{\nu_{Ax}}+A^{\nu_{Ax}}}\right)\left(1-\frac{I^{\nu_{Ix}}}{K_{Ix}^{\nu_{Ix}}+I^{\nu_{Ix}}}\right)$
Decay	First-order generic, nonspecific decay that (usually) affects all Nodes.	$-\frac{X}{H_X}$
Transformation	Changes of one Node Into another, e.g., cleavage, phosphorylation, dimerization, etc.	$T_oY_o(-k_{X+Y\rightarrow XY}XY)$ $-C_{YX}X\frac{T_o}{H_X}\left(\frac{Y^{\nu_{YX}}}{K_{YX}^{\nu_{YX}}+Y^{\nu_{YX}}}\right)$
Transfer	Various exchanges: of membrane-bound Nodes among cell faces, between cells, endo- and exocytosis, etc.	$r_{\text{flux of }X}\left(X_{\text{opposite face}}-X_{\text{this face}}\right)$

[1]We combine each Affector's formula into the right hand side of the ODE that specifies the time rate of change of a Node's concentration. Under "Synthesis" the exhibited formulae confer transcription regulated by a single activator, and transcription regulated by a single activator and single *global* repressor. An affector from the "Decay" category is usually added to every component of the model. Although Ingeneue never checks, it is usually important to include a non-specific decay term for every Node. Since this term is linearly dependent on the concentration of the Node, and all synthetic processes should saturate to be biologically realistic, the decay term ensures a maximal steady state level for every Node. Under "Transformation" the first formula represents hetero-dimerization between X and Y, while the second gives the rate of cleavage (or some other transformation) of X regulated by Y. Under "Transfer" the exhibited formula confers exchange between the apposite faces of two neighboring cells. In fact this formula is represented by an obligate pair of Affectors, thus maintaining a one-to-one relationship between Affectors and additive terms in the ODE. See the Supplement to von Dassow et al. (2000) for a rationalization of the formulas used and the non-dimensionalization scheme.

Java programming to extend Ingeneue to accommodate a wide variety of similar gene network modeling tasks.

For some processes we use Affector formulae that represent the exact chemistry involved; examples include first-order reactions like decay and second-order reactions such as ligand binding. In other instances we only approximate the literal kinetic process. The most common approximation we adopt is to use generic Hill-function-like sigmoid curves to model how transcription factors work. We digress a moment to explain that we use approximations because we prefer formulae that correspond to a biologist's description of a gene network quantified by parameters which, in principle, biologists could measure experimentally. For instance, in most contexts a biologist explaining how a group of genes regulate each other would not detail all the steps of transcription factor binding and sequential assembly of the generic transcriptional machinery. Instead she would say things like, ''Engrailed represses *cubitus interruptus* transcription.'' The details of how it does so, even though they might be very interesting, often remain unknown. What one *does* know is that every target gene has *some* maximal rate of transcription. That maximal rate, determined by enzymatic properties of RNA polymerase and by how suitable a template the gene in question makes, provides one parameter governing the synthesis rate function for that gene. Saturation at a maximal rate implies that, for every regulator, there is some regulator level at which the target achieves one-half its maximal synthesis rate. That half-maximal level provides another parameter.

The functional form quantifying the dose–response relation between a transcription factor and the transcription rate it modulates *must* be nonlinear if that rate saturates. The next practical issue is, how sharply nonlinear is the dose–response curve? Thus a third parameter determines the slope of the dose–response curve at the half-maximal point. An inflected dose–response curve could arise, for example, from cooperative binding effects, with steeper curves resulting from higher-order complex formation. This reasoning led us to adopt the kind of curve and parameters shown in Figure 3, combinations of which Ingeneue uses to model transcriptional regulatory interactions and other dose–response relationships. By neglecting to explicitly represent the assembly of the generic transcriptional machinery we assume that this process is not a rate-limiting

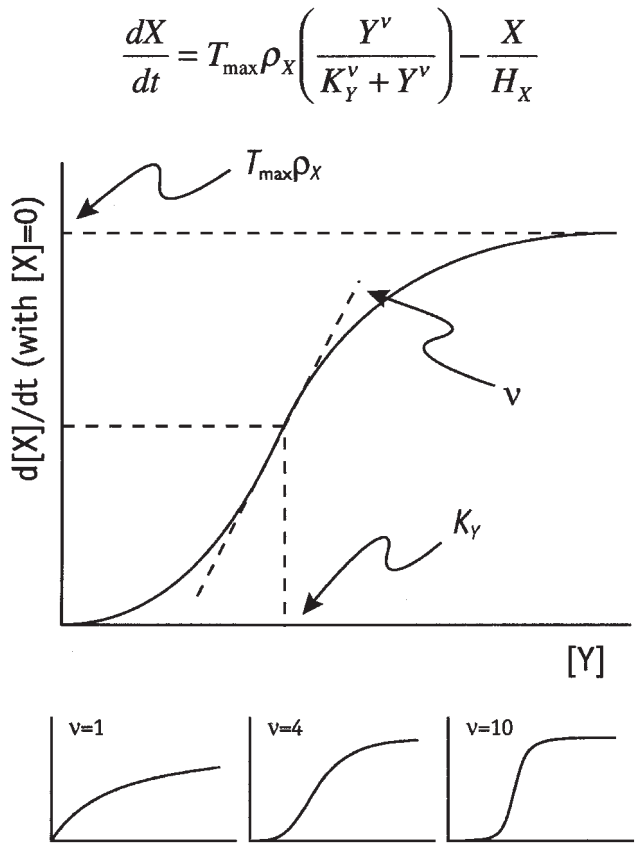


Fig. 3. Standard dose–response curve. This S-shaped curve, a graph of the key term in the differential equation shown above, is a fundamental approximation for regulatory relationships in Ingeneue. The equation in this figure is a simple case in which synthesis of X is promoted by Y, and X degrades non-specifically. The curve represents the rate of synthesis of X as a function of Y, absent decay. A salient feature is that at high activator concentration, the response saturates at some maximal value, determined by the properties of the biosynthetic machinery (e.g., RNA polymerase). It follows that at some intermediate activator concentration (K_Y) the response reaches half its maximal value. Throughout, when we use the word ''cooperativity'' we are really referring to the steepness of the curve at the inflection point (since such an inflection could be produced by cooperative interactions among activator molecules). The thumbnail curves show identically scaled versions with the cooperativity equal to 1.0, 4.0, and 10.0. With a cooperativity of 1.0 (that is, no cooperativity) the dose–response curve is nearly linear at low regulator concentrations, has no inflection, and saturates more slowly than corresponding curves with higher ν. With a cooperativity above 10.0 or so, the curve is nearly a step function. In Ingeneue models the half-maximal activation value and the cooperativity become free parameters of this formula, but the equations are normalized and rendered dimensionless such that, for processes like transcription, the maximal rate is 1.0. An inhibitory dose–response curve is thus obtained by subtracting a formula for the curve shown from 1.0. Complicated terms, such as would govern transcriptional regulation by several factors, are obtained by nesting, adding, and multiplying these curves in various combinations determined by the mechanism in question.

step. The major constraint we maintain is that all transcription must saturate at some maximal level; thus we never use linear or strictly additive formulas. But we emphasize that Ingeneue itself enforces no constraints whatever on an Affector's formula, so others could make different choices.[2]

The most mathematically complex Affectors are those governing transcription. This reflects the inherent complexity of the transcription process. Multiple transcription factors activate/inhibit most enhancer regions through detailed interactions we do not know. As explored further in Appendix A, various nested or multiplicative combinations of sigmoid dose–response curves like Figure 3 accommodate most of the possibilities for simple relationships between regulators and targets. In cases where only a single activator acts on a target gene, we use the simple parametrized sigmoid function in Figure 3 to quantify the transcription rate. To add a single inhibitor we might choose to multiply the activation function by one minus a similar function (Table 1). Even in such a simple one activator/one inhibitor case we necessarily make assumptions about the physical mechanism of inhibition. With multiple activators and inhibitors, the potential combinations get even more complicated. Ingeneue presently includes the rudiments of a flexible system using ''meta-affectors'' that add and multiply together simple activation and inhibition formulas while still preserving an overall maximum transcription rate for the target gene.

Partitioning the equations for a network model into stereotyped, reusable function fragments is another major advantage of using Ingeneue for modeling genetic networks. The Affectors constitute a parts kit for translating the cross-regulatory connections in a typical network diagram, such as Figure 1, into a set of ODEs representing those connections mathematically. This tactic enables users to build and modify a network model without needing to derive equations themselves for each case. (Nevertheless, it is vital that users understand the generic nature of the kinds of ODEs that Ingeneue uses and understand how each choice differs mechanistically.) This tactic also greatly speeds up the process of changing network topologies, even for mathematically sophisticated users.

[2]Since there is an infinitude of monotone-increasing functions having a given maximum, a given half-maximal point, and a given slope there, we do not believe that it is worthwhile to distinguish them. Convenience and tradition dictated the choice of a Hill function, but Ingeneue would just as happily integrate anything else.

Creating a model of a gene network using Ingeneue thus consists of specifying the Nodes, Affectors, and the size and geometry of the Cell array. In order to do anything with that topology one must assign values to *all* the Affectors' parameters and to the initial concentrations of each Node in each Cell. Then Ingeneue ''runs'' the model by integrating the equations over a user-specified time interval. Running the model produces dynamic behavior, visible on the computer screen as well as measurable within the program's guts, which we generally want to compare to gene expression patterns in the real biological system. The network's behavior, and the final state that the pattern may approach after long times (if such a state exists), depend on all the model's ingredients and initial conditions. This behavior may be sensitive to any particular ingredient, or may hardly be sensitive to any of them, and clearly could depend a great deal on governing parameter values. We now describe some of Ingeneue's tools for exploring those dependencies.

WANDERING THROUGH HIGH-DIMENSIONAL PARAMETER SPACE SEEKING ZONES IN WHICH MODEL NETWORKS EMULATE REAL NETWORKS

In the classical context of genetics, each gene's effects on phenotype characterize its *function*; one *function* of *wingless,* for instance, is to specify regions of naked cuticle in each segment during embryogenesis. In the context of molecular genetics, gene function means how the gene product interacts with other genes and their products within some pathway or program. That is, the *function* of *wingless* is to respond to certain intra- and inter-cellular signals and transmit them to certain targets. How does one ascribe a function to a network of genes? Going from individual genes to networks, it seems sensible to try abstracting functional ''behaviors,'' much as one would do to comprehend integrated circuit chips. One *function* of the segment polarity network is *to do a certain task*, and the question is, what task?[3]

A simplistic description of the segment polarity network's task in early *Drosophila* development is to sharpen initial condition cues, conferred by the transiently-expressed pair rule genes, into the parasegmental boundaries, and then maintain these boundaries (in a subset of the animal) throughout development. The boundaries are defined by expression of the network's constituent genes in a particular stable spatial pattern. Another way of saying this is that if somehow we could make a naïve field of cells capable of expressing only the segment polarity genes (and all the generic machinery for basic cell function), and if somehow we could provide an initial pre-pattern equivalent to the input they usually get from pair-rule genes, we would expect this network to stably maintain an asymmetric spatial regime of gene expression. That, we propose, is the *function* of this network. Ingeneue allows one to do this experiment in silice.[4] One can reconstitute a working circuit from the parts list deciphered by molecular geneticists and inquire whether that circuit indeed does the task it is supposed to do. Incidentally, we find the abstraction of a network's functional task one of the most difficult (and most critically vulnerable) parts of the entire modeling process. Given a group of genes that interact, how are we to characterize what it is that they *do*, and which aspects of what they are *observed* to do, are the important aspects?

No matter how difficult it is to deduce the function of a specific network, that function usually abstracts to some dynamical behavior, such as producing a pattern of gene expression in time, space, or both. The spatiotemporal pattern that emerges depends not only on the network's topology but also on the values of parameters in the model and on the initial conditions. These parameters quantify the shapes and strengths of the network's connections by specifying biochemical reaction rates for translation, degradation, dimerization, and so on. Exploring how a network's behavior may (or may not) change as parameter values change is the central, inescapable task of all gene network modelers (until actual values for all the parameters have been measured—thus, realistically, forever or at least until all authors of this paper have passed away). Even the simplest realistic networks involve extravagant numbers of parameters (48 in Fig. 1). This is not a calamity caused by mathematical

[3]Of course the same network may have other functions at different developmental stages, or in a different organism.

[4]The term ''in silico'' has become commonplace in the last several years, but after we used it in von Dassow et al. (2000), Reed A. Cartwright wrote us as follows:

''I would like to point out an error in your paper. You used the term ''*in silico*'' to compare computational simulations to that of life (*in vivo*) and laboratory (*in vitro*). However, you have made a mistake in your Latin. Silicon comes from the Latin word *silex* which means ''hard stone.'' Unlike *vivus* and *vitrum* which are second declention nouns, *silex* is a third declention noun, and thus its ablative singular is *silice*. And since ''in'' takes the ablative, the correct phrase for ''in stone'' would be *in silice* and not *in silico*.''

modeling, *but a fact of nature*, a fact which mathematicians cannot but sin to gloss over. For much simpler networks than Figure 1 it is possible to exhaustively catalog a network's behavior across all biologically realistic parameter combinations, but this is out of the question for almost anything interesting. If we assign each parameter either a high, medium, or low value, it would require roughly 3^{48}, or $\approx 8 \times 10^{22}$, samples to try every possible combination in the segment polarity network. Waiting (e.g., for AMD to release the Athlon XII) will *never* make it realistic to do a *tenth of a mole* of simulation runs. Our laboratory's existing computers could perform the necessary calculations in 10^{14} years. So, since exhaustive exploration is right out, one has roughly four choices:

- Use mathematical strategies for dealing with large parameter spaces, such as non-linear optimization techniques that start from a guessed set of parameter values and search iteratively for nearby parameter values that better match the target behavior;
- Constrain the problem using empirical information, such as the knowledge that transcription factor X is a potent activator of gene y but a poor activator of gene z, or that protein A is much more rapidly degraded than protein B;
- Intuit what values will make the network perform its ''function'';
- Depend upon the luck to model a network whose connection topology, we presume, natural selection has optimized over deep time to have the mysterious property that it is trivially easy to find, by accident, sets of parameter values that confer the desired behavior on the network.

We confess to relying primarily, though not exclusively, on the fourth option, after discovering this unforeseen but happy possibility. As mentioned above, for the segment polarity network model we were able to find ''good'' parameter sets that caused the network to make a reasonable match to Figure 1B merely by randomly choosing values for each parameter. Naming mathematical parameters that confer shape and strength on network connections, then seeking ''good'' values for them, may seem artificial mumbo jumbo with which mathematical modelers merely confuse an already difficult problem. But it is not so: regardless of what jargon one uses to describe the formidable task of navigating high-dimensional parameter spaces, performing that task is *the* likely operation of evolution by natural selection.

The classes of parameters in a network model depend on the types of equations used. Table 2 shows the classes of parameters (all dimensionless) appearing in our segment polarity models. We have tried to ensure all parameters in our models are, at least in principle, measurable quantities. It follows that the parameters are thus physically intuitive quantities, for instance the rate at which each component decays over time or the concentration at which a transcriptional activator half-maximally turns on transcription. In each model, we set bounds on the realistic range of each parameter. Where available, we used

TABLE 2. Common classes of parameters[1]

Parameter Type	Symbol	Description
Half-maximal activity	K_{Yx}	Dimensionless concentration of a Node at which it half-maximally activates or inhibits some process.
Cooperativity (or Hill) coefficient	ν_{Yx}	Exponent that determines how separately the rate of some process changes as some regulator Node increases (i.e., how S-shaped the dose–response curve is).
Half-life	H_X	Time constant for nonspecific degradation of each Node (half-life=ln(2)*time constant).
Maximal rate	C_X, V_{YX}, etc.	Miscellaneous rates, e.g., for a cleavage reaction, equivalent to V_{max} for an enzyme.
Maximal concentration	X_o	The maximum dimensional concentration that a Node can achieve at a steady state. This parameter is required chiefly in heterodimerization reactions.
Exchange rate	r	Rates at which transfer process equilibrate Node concentrations between various compartments.
Activation strength	α	The relative strengths of different activators and inhibitors in complex enhancer regions.

[1]See the Supplement to von Dassow et al. (2000) for a more extensive discussion.

published values as guides to the general range and then expanded bounds to span the smallest and largest biologically reasonable values. As an example, in exploring the segment polarity network we let what we call ''half-lives'' (the degradation time constants, actually) vary between 5 and 100 min. We considered it unlikely that these molecules were degraded much more rapidly than with a 5-min time constant (corresponding to a half-life of about 3 min, about what has been measured for *ftz* mRNA; Edgar et al., '86). At the upper end, the segment polarity pattern forms over the course of hours or less, so the gene products involved must degrade fast enough to change on that time scale. We thus set the upper limit on degradation time constants at 100 min. Published measurements of half-lives in the segmentation network all fall within this range: Engrailed protein has a half-life of <15 min (Weir et al., '88); Armadillo protein (involved in segmentation though not explicitly represented in the model discussed here) has a half-life of around 12 min (Pai et al., '97); Cubitus interruptus protein, in cultured cells, has a half-life of 75 min (Aza-Blanc et al., '97).

Automating the search for ''working'' sets of parameter values requires a goodness-of-fit function which we craft to assess how close the network, with each trial parameter set, comes to matching the target behavior. Pattern matching is currently the weakest part of Ingeneue in terms of making a program that can be used by biologists without additional customization, and we do not yet have a full solution. Presently the user must custom-code their own pattern recognizer if those we supply do not suit the task. It is much less clear in the case of pattern recognition than it is for the Affectors what the primitive unit for general applications should be, so to date we have supplied very few. The general strategy in Ingeneue is to allow the user to assemble, from a library of primitive parts or by custom coding, so-called ''StoppingCondition'' objects. StoppingCondition objects monitor the progress of the integration run and return a scalar score measuring how well the run conforms to some ideal behavior, and they can be set up to suspend the run if it is pointless to continue further (i.e., if either the network has already conformed well enough to the ideal behavior, or if it is clear that it will never conform, say, because it has achieved some alternate stable state). Our current StoppingConditions can look for primitive behaviors such as ''Node X on in Cell 1,'' ''Node X off in Cell 2,'' and ''Node Y oscillating in Cell 3.'' We then combine several StoppingConditions together to recognize more complex patterns.

For example, the target pattern for the segment polarity network is a set of vertical stripes: a stripe of en in the first column of each parasegmental repeat, a stripe of wg in the fourth column, a stripe of hh where en is on, and so on. Our pattern recognizer function for the segment polarity network is thus a group of several StoppingConditions that recognize stripes. Together these assess whether the model achieves the correct on/off pattern of wg, en, and hh, in the desired positions and in a sufficiently short time (see Supplement to von Dassow et al., 2000). The stripe recognizer returns a better (lower) score as the difference between the concentrations within that column and in neighboring columns grows larger, and its score also improves if the concentration within the column is stable over time. We laboriously hand-tuned the exact recipe for the segment polarity model's pattern recognizer function until it caught only the parameter sets we wanted. We doubt that we will ever be able to eliminate the need for such hand-tuning, but the challenge for future development of this aspect of Ingeneue is to incorporate a versatile library of pattern recognizer modules that allow one to avoid custom coding for most applications.

Given a function that scores patterns, Ingeneue can automatically search parameter space for sets of parameter values that produce that pattern (i.e., that achieve a sufficiently low value of the ''objective function'' which measures distance away from the ideal pattern). This is another major advantage of using Ingeneue to build and analyze gene network models. The basic framework for automated searching is a collection of ''Iterator'' objects. We have been rather surprised to find that so far, for several different cases, the most useful algorithm has been random sampling of parameters from within biologically realistic ranges of parameter values. Other Iterators implement various algorithms for navigating the landscape in parameter space, including a variety of standard and custom-made nonlinear optimization schemes. Because most of our work to date has, fortuitously or otherwise, allowed us to take the most simple-minded of approaches, we have yet to explore thoroughly the effectiveness of various sophisticated strategies for searching parameter space. We expect, however, that since these network models all use similar equations, it should be possible to identify which search

algorithms work best on this whole class of models. Once (or if) one finds sets of parameter values for which the model works, the Iterator framework enables automated sensitivity tests. For example, Figure 3 of von Dassow et al. (2000) was made using the ''TransectIterator'' object, which, starting with an input parameter set, simply scans along the entire range of one or more parameters while keeping all other parameter's values fixed and records the score at each point. Again, Ingeneue's design allows one to add, as a drop-in module, any stratagem one wants to try.

USING INGENEUE WITH A MOUSE

Ingeneue records its results automatically in a text file, and the Ingeneue core described above can run on its own, without a graphical display, simply based on text file input specified in a Unix command line. This is useful for running Ingeneue remotely, once one has developed a reliable pattern recognition function and devised a strategy for searching parameter space. When starting a new task with the program, though, it is vital to see the dynamics of the model as it runs. This develops the user's intuition about the model, lets the user fine-tune automated strategies, and helps to catch mistakes. Guided by these needs, Ingeneue provides a simple graphical interface for making certain kinds of changes to the model (for instance, changing parameters and initial conditions, but not yet the topology of network connections), and viewing the concentrations of any Node in any Cell as a model runs.

Ingeneue has four main windows in addition to the console (Fig. 4). The Cell View (upper middle) shows color-coded concentrations of different Nodes within each Cell in the model. Clicking on one of the hexagons representing an individual Cell brings up an Inspector window that shows a numerical display of the concentration of that Node in that Cell. The user can change the current and initial concentrations of any Node in any Cell. On the upper left is the Network View, which displays the topology of the currently loaded network. Different shapes correspond to different types of Nodes (mRNA, protein, protein complex, etc.), and the lines indicate which Nodes influence each other through Affector objects. For instance, the line from en to EN represents the EN translation Affector. Clicking on the circle within this line causes the Inspector window to show a list of the parameters that appear in that Affector's formula and allows the user to change the parameter values arbitrarily.

The segment polarity model has too many parameters to view entire sets using standard graphs, yet it is very useful to compare different parameter sets visually. To display multiple parameter sets, each containing dozens of parameters, we use ''wheel'' plots where each parameter set is displayed as an irregular polygon intersecting the many spokes of a wheel (Fig. 5). Each spoke represents the axis along which we allow an individual parameter to vary. The inner circle defines the minimum and the outside circle the maximum possible value. Thus a single polygon, intersecting one point on each spoke, represents a set of parameter values. Figure 5 shows an example of a wheel plot of seven different parameter sets, each of which confers upon the segment polarity network the ability to pass our most basic functional test. Buttons and menu choices in the wheel plot window (top right of Fig. 4) allow the user to flip quickly through many parameter sets and impose any of them on the currently loaded model. This window also allows one to calculate simple statistics (standard deviations, cross-correlation coefficients, etc.) from a group of parameter sets. We continue to develop methods to make sense of the parameter space; for instance, an ancillary program (''Gatherer''; Odell, unpublished observations) allows us to look for ''clusters'' of related parameter sets within the larger group found in a random sample.

EXAMPLE EXPLORATIONS USING INGENEUE

Above we highlighted three major advantages of using Ingeneue to model gene networks instead of general-purpose mathematics software: 1) Ingeneue handles the busy-work of stamping out copies of the gene network template into each cell in an arbitrary-sized field of cells; 2) Ingeneue allows one to construct equations from stereotyped building blocks; 3) Ingeneue facilitates automated searches through parameter space, sensitivity analyses, and similar tasks. All these combine to make possible a fourth major advantage, which is that *Ingeneue makes it easy for the user to systematically test the effect of changing the parameters, architecture, and components of gene network models.* To illustrate these advantages we give three short examples below. In the first example we look at how the rate of diffusion of secreted signals affects the ability of the

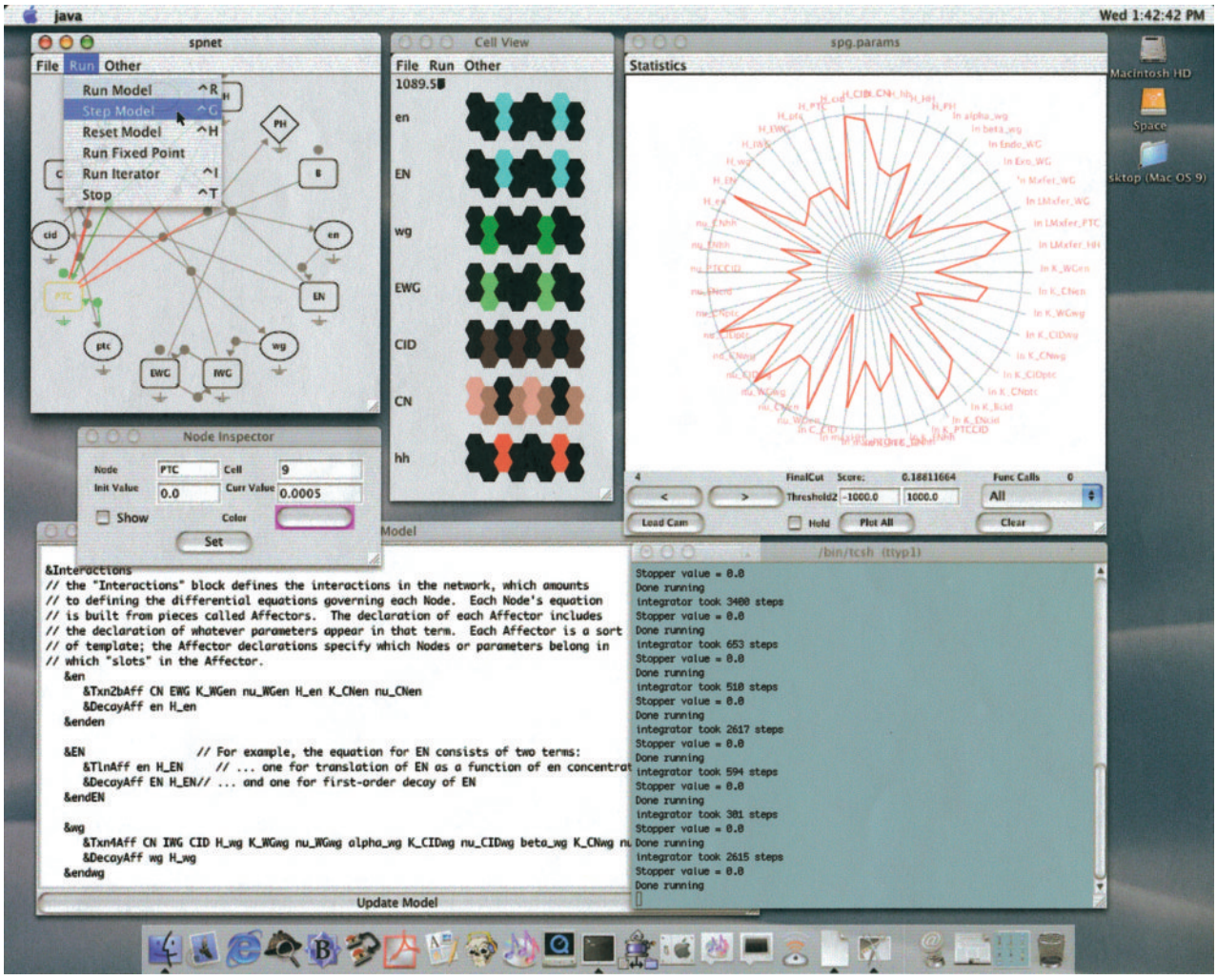


Fig. 4. Ingeneue interface. Screen snapshot showing the main windows in Ingeneue's interface. The top left window shows the Nodes and Affectors in the segment polarity network. Clicking on one of these nodes brings up the Inspector window below, which allows the user to inspect and change concentrations of any Node in any Cell and to change parameter values for any Affector. In the middle is the Cell View window, which graphically shows the concentrations of a user-selected set of Nodes in each of the Cells in the model. The top right window displays a battery of parameter sets on a wheel plot and allows the user to impose one set at a time on the currently-loaded network (see Fig. 5).

segment polarity network to produce the pattern in Fig 1B. In the second example we ask the same question but with respect to the degree of ''cooperativity'' in regulatory interactions in the network. Finally we illustrate how the segment polarity module really consists of two sub-modules.

Sensitivity of the segment polarity model to diffusion rates

There is a long-standing controversy concerning the role diffusion plays in developmental pattern formation. It has proven difficult to demonstrate conclusively that diffusion gradients of proteins actually exist, or that these gradients affect patterning. In part, this experimental quest to find diffusing molecules was driven by a body of mathematical theory that assumed diffusion of molecules was centrally important in patterning (Slack, '83; summarized in Murray, '93). The theory is based on Turing's idea of reactions among competing proteins that diffuse at different rates. Depending on the types and rates of the reactions and the rates of diffusion, various simple models can exhibit interesting pattern-forming behaviors. One of the primary difficulties with these so-called reaction–diffusion mechanisms, however, is that they generally require careful tuning of relative reaction rates versus the

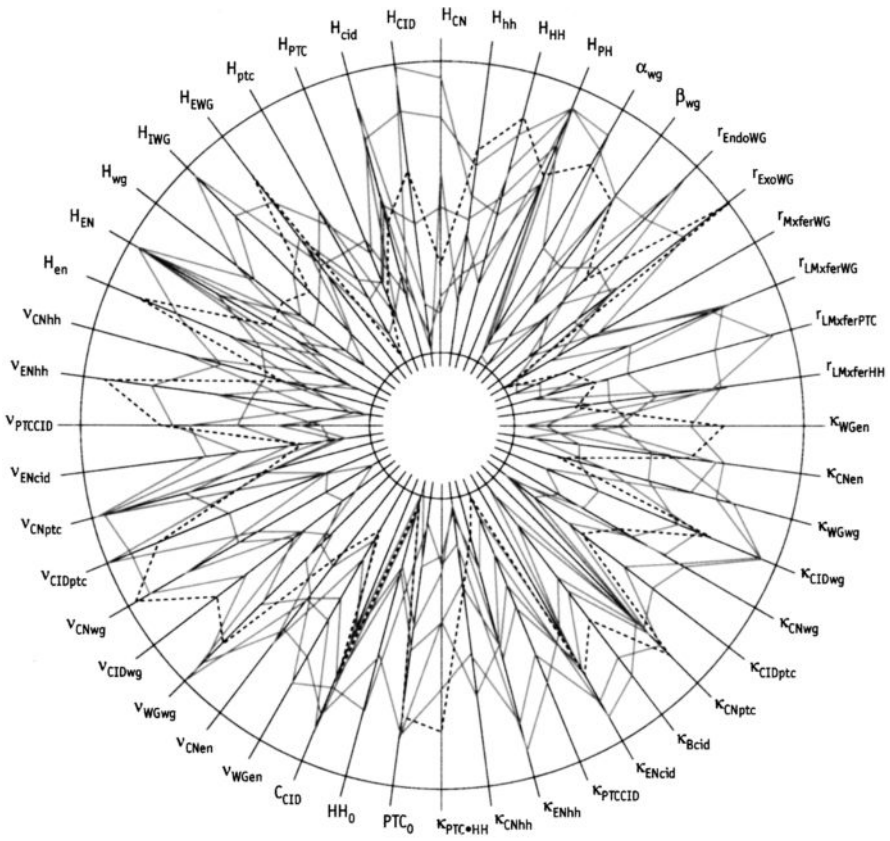


Fig. 5. Wheel plot. Wheel plots exhibit multiple parameter sets, each comprising dozens of parameters. Each spoke is the axis (log or linear scale, as chosen by the user) for one of the 48 parameters in the model. The inner circle represents the minimum value in each parameter's range, and the outer circle represents the maximum value. Thus one parameter set is represented by a single irregular polygon with a vertex on each spoke of the wheel. The dashed polygon is one parameter set that was successful at forming the segment polarity pattern. The other six polygons show other successful parameter sets. Notice that each of these has very different parameter values from the others. By plotting many parameter sets on top of each other, one can get an idea of whether one or another parameter's values are clustered in one part of its range.

diffusion rates. In particular, the ''wavelength'' of the patterns made by reaction–diffusion mechanisms is highly sensitive to the rates at which various factors diffuse across the simulated field.

Several recent studies demonstrate that developmentally important molecules do indeed diffuse from their site of synthesis, and that the concentration gradients thus established play a role in patterning (see for example Gurdon et al., '94, and Nellen et al., '96). Two of the segment polarity network components, Wingless and Hedgehog, are secreted proteins that potentially spread through the embryonic epidermis by diffusion (or even by convection or active transport of some kind). However, we do not know if diffusion of these signals influences the segment polarity network's function in vivo. Both Wg and Hh transport are carefully regulated in fly embryos. There is substantial evidence that during embryogenesis the primary means by which Wg travels from cell to cell is an active transport mechanism involving endocytic uptake (Dierick and Bejsovec, '98; Moline et al., '99; Pfeiffer and Vincent, '99), and it remains unclear to what extent (if any) the appearance of Wg protein far from its site of synthesis is due to genuine extracellular diffusion. Hh is synthesized as a transmembrane precursor and undergoes autolytic processing to a cholesterol-tethered form whose cell-to-cell transport may be mediated by other regulators, such as Patched and Dispatched (Burke et al., '99). This suggests that Hh transport might be under tight control in vivo, but introduction of a freely diffusible Hh has only subtle effects on embryogenesis (Porter et al., '96).

During our initial work with the segment polarity model we allowed the Wingless ''diffusion'' rate to vary over a wide range, from values

TABLE 3. Frequency of solutions as a function of various exchange/flux processes[1]

Constraint	No. of hits	No. tries	Avg. score	Hit rate
Standard ranges	1,120	235,875	0.077	1 in 211
WG diffusion fast: r_{MxferWG} [0.1–1.0]	1,033	405,394	0.077	1 in 392
WG diffusion moderate: r_{MxferWG} [0.01–0.1]	1,811	416,824	0.075	1 in 230
WG diffusion slow: r_{MxferWG} [0.001–0.01]	1,804	237,877	0.075	1 in 132
WG intramembrane diffusion, endocytosis slow: r_{LMxferWG} r_{EndoWG} [0.001–0.01]	1,618	243,875	0.073	1 in 151
HH allowed to diffuse r_{MxferHH} [0.001–1.0]	1,197	237,572	0.075	1 in 198

[1]The proportion of randomly chosen parameter sets with which our segment polarity model (Fig. 1A) produced the correct segment polarity pattern (Fig. 1B) when the diffusion rates of WG and HH were constrained. Each try started from an initial condition of a stripe of wg and a stripe of en (von Dassow et al., 2000) and had to acquire the target pattern within 200 min and be stable over 1,000 min in order to count as a success. The standard ranges we use for comparisons allow the WG diffusion and WG endocytosis parameters to vary between 0.001 and 1.0 and do not allow any diffusion of HH.

that allow essentially no transport (on the time scale of the simulation) to values at which extracellular Wingless concentrations rapidly equilibrate across the cell field. In addition, exo- and endocytosis together mediate ''transcytosis'' of Wg. In our original model we did not allow Hedgehog to move from cell to cell at all. In light of recent reports on Wg and Hh traffic, we tested whether cell-to-cell transport of either Wg and/or Hh affects the basic pattern-maintaining function of the segment polarity model. In the original model, with a wide range for the Wg diffusion rate, we found that ≈ 1 in 200 randomly chosen parameter sets conferred on the network the ability to make and hold the pattern in Figure 1B. We now restrict the range of the Wg diffusion rate to slow, intermediate, or fast sub-ranges, while still picking all other parameters from the same ranges as before. The overall result is that neither fast nor intermediate nor slow Wg diffusion greatly affects the ability of the network to pass the test (Table 3). For all Wg diffusion rates, random sampling yields one ''working'' parameter set for every few hundred sampled sets. There is, however, a definite increase in the hit rate when we constrain the Wg diffusion rate to the slow end of its range. Thus, the less freely Wg diffuses the more tolerant the model is to variations of other parameters. To look for mutual constraints among parameters we can examine how a constraint enforced on one parameter affects the distribution of values for others. In this case we found that the rates of Wg exo- and endocytosis exhibit tradeoffs relative to Wg cell-surface-to-surface flux rate (see companion paper by von Dassow and Odell, 2002, this issue). In contrast, allowing Hh to travel from cell to cell, quickly or slowly, seems to make almost no difference at all to the ability of the network to achieve the target pattern (Table 3). The important point is that, despite the subtle sensitivity we show to Wg flux rates, vast changes in those rates have *no* effect on the ''wavelength'' of the pattern formed. Indeed, unlike reaction–diffusion mechanisms, the segment polarity model has no intrinsic wavelength at all, and the periodicity of its pattern (when it makes the right pattern, that is) is controlled solely by initial conditions; adjacent boundary regions simply do not depend on one another.

Cooperativity as a determinant for robustness in switching networks

Although diffusion rates seem to have only a small effect on the basic function of the segment polarity model, the same cannot be said for the steepness of dose–response curves (see Fig. 3), which we call ''cooperativity,'' in regulatory interactions. We can get a rough idea for the role of cooperativity in this model by fixing every cooperativity coefficient in the model at a single value, either all together or in turn, as shown in Tables 4 and 5. If all regulatory interactions are constrained to be non-cooperative, we can find *no* parameter sets for which the model works. As the uniform cooperativity level increases (Table 4), the hit rate increases but plateaus above about 5.0. However, when we hold each individual cooperativity coefficient to 1.0 in turn and allow the others to vary freely, only one of these parameters (the cooperativity coefficient for *wingless* autoactivation) is absolutely required to be greater than 1.0 (Table 5).[5] All the other cooperativity coefficients

[5]Note that 48-dimensional parameter space is so vast that our failure to find any working parameter sets does not mean no such set exists. Far from it. Even if one in a billion random sets succeeded, one would conclude the network was far more robust than the best electronic circuits. However, considering the role that *wingless* autoregulation plays in the segment polarity network, and given the formula we used for wg transcription (see von Dassow et al., 2000, and the companion paper by von Dassow and Odell, 2002, this issue), it is clear that this interaction *must* be cooperative; otherwise it would be impossible for there to exist two stable levels of wg expression.

TABLE 4. Frequency of solutions as a function of global level of non-linearity[1]

All ν constrained to	No. of hits	No. of tries	Hit rate
ν range 2–10	1,192	240,000	1 in 201
ν range 1–10	1,316	320,806	1 in 244
1	0	40,467	Never
1.3	0	38,559	Never
1.7	4	37,180	1 in 9,295
2	23	37,424	1 in 1,627
3	82	36,545	1 in 446
4	91	24,684	1 in 271
5	137	32,775	1 in 239
6	195	32,908	1 in 169
7	351	57,111	1 in 163
8	199	29,054	1 in 146
9	287	37,587	1 in 131
10	140	17,721	1 in 127

[1]The proportion of randomly chosen parameter sets where the segment polarity model (Fig. 1A) produced the segment polarity pattern (Fig. 1B) when all cooperativity parameters were constrained to a single value. Tests were done as in Table 3. Separately we sampled over 400,000 parameter sets with all cooperativities set to 1 and found no sets that worked.

can be constrained, individually, to 1.0 without greatly affecting the hit rate, and indeed the hit rate actually improves when certain interactions are forced to be non-cooperative. In this context we note that the most intensively studied enhancer regions tend to have multiple binding sites for important activators and inhibitors (e.g., Arnosti et al., '96; La Rosee et al., '97). These sites interact in various ways, perhaps leading to the phenomenon that we call ''cooperativity,'' that is, the sigmoid shape of the dose–response curve. In some cases, such as the targets of Bicoid (Driever et al., '89; Struhl et al., '89) and the phage lambda repressor (Ptashne, '92), this effect has been confirmed to be due directly to the nature and number of binding sites.

TABLE 5. Frequency of solutions as a function of level of non-linearity in particular regulatory interactions[1]

	No. of hits	No. of tries	Hit rate
ν 2–10	1,192	240,000	1 in 201
ν constrained to 1	0	49,250	Never
ν_{WGen}	173	37,734	1 in 218
ν_{CNen}	73	41,236	1 in 565
ν_{WGwg}	0	42,708	Never
ν_{CIDwg}	80	28,416	1 in 355
ν_{CNwg}	138	40,351	1 in 292
ν_{CIDptc}	228	40,143	1 in 176
ν_{CNptc}	265	39,079	1 in 147
ν_{ENcid}	32	45,612	1 in 1,425
ν_{ENhh}	185	42,667	1 in 230
ν_{CNhh}	141	39,755	1 in 282
ν_{PTCCID}	271	39,748	1 in 147

[1]All tests as described in Tables 3 and 4.

The effect of cooperativity on the robustness of the segment polarity network is particularly interesting from an evolutionary point of view. As Gibson has noted elsewhere, cooperativity could be a generic means of achieving canalization (Gibson, '96; see Gibson and Wagner, 2000, for an informative review of canalization). Robustness to parameter variation is, more or less, canalization against genetic variation, albeit from a different point of view. How hard can it be, evolutionarily, to modulate the steepness of cooperative transcriptional regulation? We do not know for sure, but taking the work on Bicoid as an example (Driever et al., '89; Struhl et al., '89; reviewed in Driever, '93), it seems not hard at all. Point mutations in Bicoid binding sites in the *hunchback* enhancer region, by changing the binding affinity at that site, alter the dose–response profile as visualized by levels of *hunchback* expression along the anterior–posterior axis of the egg. Adding or removing binding sites has a more dramatic effect, and the natural targets of Bicoid differ in the number and affinity of binding sites in a way that corresponds to their expression profile. Assuming this case is paradigmatic, the effect we describe in the previous paragraph means that the segment polarity network *incorporates tuning dials for robustness.* Should the segment polarity network have been ''invented'' lacking cooperativity, and therefore lacking robustness as well, it may have been a relatively trivial path for the evolutionary process to tune it up. Should cooperativity prove a generic determinant of robustness, as suggested by Gibson and supported here, the implications would be profound.

Segment polarity network consists of two sub-modules

We have often wondered where the segment polarity network came from; that is, how did natural selection hit upon such a remarkably robust design? As described in the companion paper, when we try to concoct networks that can perform the same task, we have a hard time of it. What could nature have started with? The segment polarity network consists of two cell–cell signals: Wg activates itself via a poorly understood mechanism and also activates *en* in neighboring cells; Hh regulates the relative abundance of

activator and repressor forms of Ci, which in turn regulates various target genes, in particular the Hh-binding component of the Hh receptor complex, *ptc*. These two signals are sewn together into a larger mechanism by the facts that En regulates *hh* expression, and Ci regulates *wg* (see Fig. 1). The larger mechanism, with a few more connections, is capable of maintaining asymmetric boundaries with three cell states: a ground state in which neither signal is expressed and two co-dependent cell states in which either *wg* or *en*/*hh* is expressed. This is the function we test the segment polarity models for. The *hh-ptc-ci* sub-network, by itself, cannot do this: the closest it comes is to make ''center-surround'' patterns in which a central cell expresses *hh*, nearby cells express Ci targets such as *ptc*, and distant cells express only the repressor form of Ci, CN. In order for this sub-network to do this, there must be 1) a constant input to *hh* in some cells and 2) constitutive expression of *ci* in all cells. Indeed, the *hh-ci-ptc* network is put to exactly this use in a bewildering diversity of vertebrate tissues, including hair follicles, limb buds, and tooth primordia, to name only a few (Goodrich et al., '96; Marigo et al., '96).

The other sub-network, consisting of an auto-regulating *wg* gene and Wg-dependent *en*, also can make center-surround patterns under certain conditions but cannot, as represented in our original model, maintain asymmetric boundaries. Thus we concluded that the network in Figure 1 was a ''minimal'' reconstitution of the segment polarity module. However, we continued to wonder if there might be a way to get a simpler subset of the known interactions among segment polarity genes to do the job. The *sloppy-paired* gene is a good candidate for an intermediary in *wg* auto-regulation (Grossniklaus et al., '92; Cadigan et al., '94a,b; Bhat et al., 2000; Lee and Frasch, 2000; see companion paper by von Dassow and Odell, 2002, this issue). *slp* encodes a transcriptional regulator that activates *wg* and represses *en*. At the same time, Wg promotes *slp* expression. It seemed to us that, if *en* also inhibits *slp,* an *slp-wg* positive regulatory loop, coupled to a *slp-en* mutually negative loop, as shown in Figure 6A, *ought* to be able to do the same task that the larger network does. We challenged such a model, and, after much trial and error, found that, under finely tuned conditions, it could do so if the initial pre-pattern is rigidly specified, as in Figure 6B (without initial high-level slp and SLP in the wg-expressing cells and the blip of initial slp and SLP in the first cell in each repeat, not too much and not too little, solutions are too scarce to find in a random sample). Even then, however, this network is much more sensitive than the network in Figure 1 to many of the governing parameters. Figure 6C shows that some parameters are tightly constrained to certain ranges of values. Furthermore, not apparent from the drunken-spider plot but obvious from histograms of the same values (Fig. 6D), about one-half of the parameters are biased toward some cluster of values. Also, compared to the larger network, the *wg-slp-en* network is more sensitive to the choice of initial conditions, and to details of how the component genes interact; in comparison to the larger structure, this little fragment is a pale wisp of a network indeed.

Thus, the *wg-slp-en* sub-network *can do the same task as the larger network in which is it embedded, but it does so without nearly the robustness characteristic of the larger design*. We indulge in the following speculation: perhaps this tiny, (relatively) fragile protomodule might have been an evolutionary precursor to the segment polarity network we know today in *Drosophila*. Indeed, perhaps there are unlucky arthropods out there today struggling to get by with such a primitive boundary maker. Perhaps the *hh-ci-ptc* circuit was originally a subsidiary process in segmentation, with *hh* a downstream target of En. If some enterprising Ur-arthropod, maybe even an insect, happened to manage to invent a connection between Ci and *wg*, the device shown in Figure 1 would have been born and might have loosened constraints on the variation of certain segmentation genes. If some descendent managed, further, to connect CN to *en*, the robustness of the network (to variation in parameters, initial conditions, and even structure) would have increased further still. We know, as yet, very little about the conservation of the segment polarity network among arthropods. It would be fascinating to know if such a scenario might even be traced among the arthropods living today!

DISCUSSION

Ingeneue is an early entrant in what is sure to become a whole breed of software for dealing with large tangles of molecular genetic data as networks, circuits, and systems. Because our knowledge of how genetic regulatory systems operate is advancing so rapidly, computer tools that help integrate and interpret these data will soon

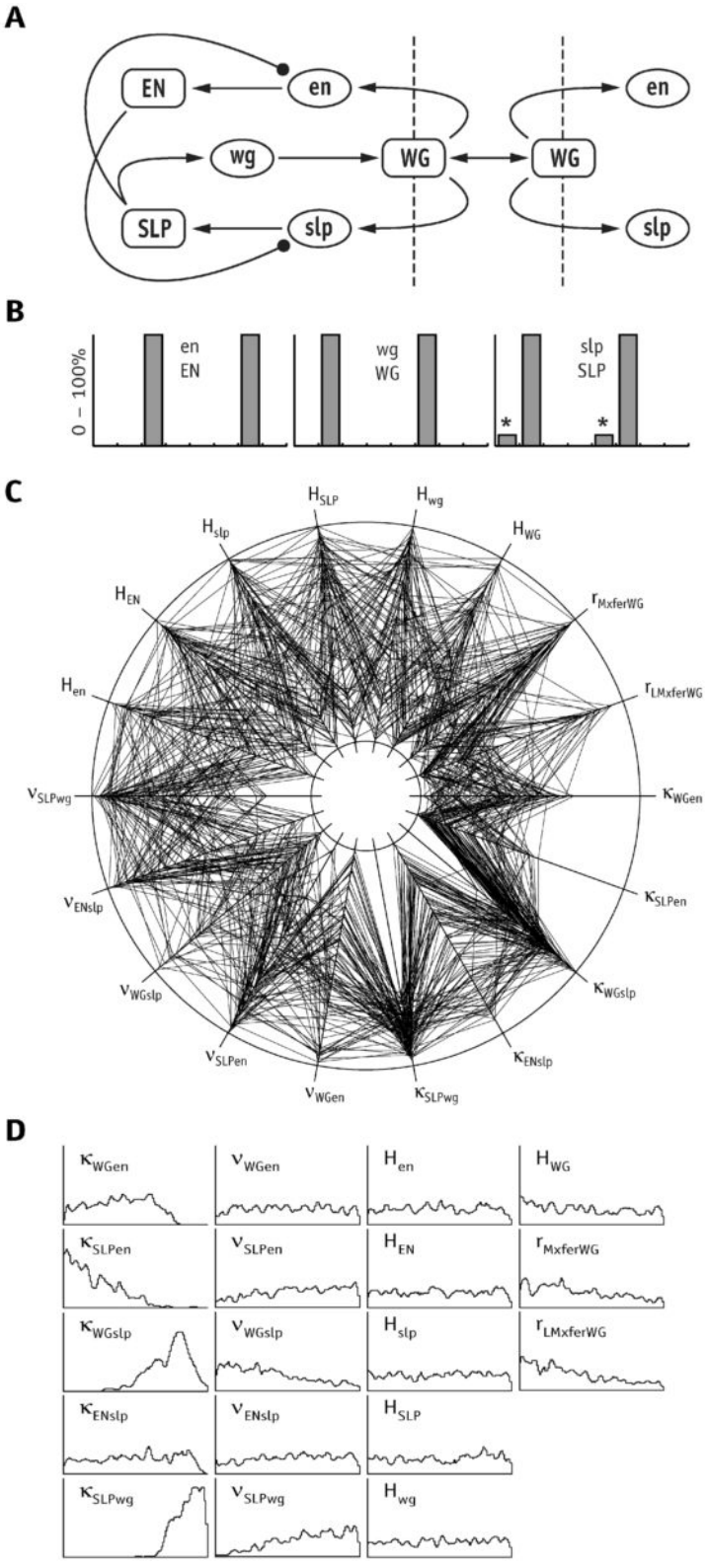


Fig. 6. Wg-Slp-En sub-module. (**A**) Network diagram summarizing known facts about the interconnections of *wg*, *slp*, and *en*. Slp represses *en* transcription and promotes *wg* transcription; Wg promotes both *en* and *slp* transcription; En represses *slp* transcription. The last link we have seen described on a poster (M. Kobayashi et al., '98, 40th Annual *Drosophila* Research Conference, Abstract 184C) but not in final publication. In our model of this circuit we have made the assumption that in each Cell the Nodes en and slp are sensitive to WG bound to that Cell's surface; that is, WG must move from one Cell to another to affect its targets in Cells where WG is not produced. This kind of detail is relatively insignificant to the standard segment polarity network model (Fig. 1A; see companion paper by von Dassow and Odell, 2002, this issue, for analysis), but significantly affects the behavior of this sub-circuit. (**B**) Initial conditions required by the Wg-Slp-En sub-module. Each plot shows eight Cells' worth, with the height of the bar corresponding to the initial level of each Node. Without the low-level initial expression of slp and SLP in positions 1 and 5, marked by asterisks, solutions are about five times scarcer, but the same restrictions on parameter values appear (not shown), albeit more extreme. Note that this initial pattern is also almost exactly the final pattern we expect. (**C**) Drunken-spider plot of 100 parameter sets that enable the Wg-Slp-En sub-module to hold the same pattern as we require of the standard segment polarity model; strong restrictions are evident on several parameters. κ_{Xy} refers to the half-maximal coefficient for action of regulator X on target y; ν_{Xy} is the cooperativity exponent for that interaction; H_X refers to the time constant for X degradation; $r_{MxferWG}$ and $r_{LMxferWG}$ are the rates at which WG equilibrates from one cell face to the apposed face of the neighbor, or to the neighboring faces on the same Cell, respectively. Compare this plot to Figure 2a in von Dassow et al. (2000). There are stronger cross-correlations among parameters in this subcircuit than for the whole network (not shown). (**D**) Histograms compiled from 617 parameter sets that make the subcircuit ''work,'' illustrating that there are many modest biases in addition to the strong restrictions evident in the wheel plot. For instance, note that the cooperativity for WG activation of slp should be low, but for SLP activation of wg should be high. Compare to Figure 6 in the companion paper. Ranges along the horizontal axis are 1,000-fold (log scale) for values of κ and r, 10-fold for values of ν, and 100-fold for values of H, as for the segment polarity network. Thus the restriction on κ_{SLPwg}, for example, is about 1 order of magnitude wide.

become critical adjuncts to lab-bench molecular biology, just as they have become for various fields from enzymology to ecology. Yet most biologists do not presently receive or seek the mathematical and computer science training they would need to develop such tools on their own or even to use general-purpose mathematical software. A program like Ingeneue can build mathematically rigorous models using a syntax that biologists with brief training in differential equations can learn easily. This biologist can then explore his or her favorite networks through the graphical interface and gain an intuitive understanding of its dynamical behavior as a whole mechanism. This kind of computer-assisted synthesis, if made accessible to the scientists that actually confront the biological subject every day, will help us all understand gene network mechanics better.

We designed Ingeneue to facilitate its use as a fairly general tool. Because the code is object-oriented and separated into distinct pieces that do not rely explicitly upon each other's details, Ingeneue can be easily extended to add features, methods, and facilities. This is particularly evident in the Affectors, where it takes a trivial amount of code to add a new formula. The same thing is true for pattern-matching algorithms, search strategies, initial conditions, and the graphical interface. In addition to ease of modification, the program's structure has made it easy for us to replace mathematical formulas with names (of Affectors, for instance), and in the future with graphical symbols, that can be combined together to make models. Ultimately this architecture will become a grammar and syntax for translating a diagram of a gene network to a set of mathematical equations and back again. Using this syntax we can construct new networks almost entirely from the existing Affectors in relatively short times. Even those of us who have some mathematical training find this tremendously helpful! Perhaps even more important, the stereotyping of building blocks makes it easier to compare one model to another; if two models have been built from the same parts and analyzed the same way, the results should be more readily comparable than if two different artisans crafted them in their own styles. Just as helpful is Ingeneue's ability to instantiate a network over a field of cells, correctly hooking up exchanges of all the transmembrane components and eliminating many of the small mistakes that would inevitably creep in when trying to keep track of so many equations by hand.

We do not think of Ingeneue as the ultimate in pattern formation simulators. Ingeneue is still very much a work in progress, and serious conceptual and technical weaknesses remain. One deficiency is Ingeneue's current inability to deal with morphogenesis or cell movement. In many cases, developmentally interesting patterns are formed at the same time that cells are dividing, moving around, and changing neighbors. These movements may be intimately coupled to the pattern formation process. Furthermore, Ingeneue assumes cells to be well-stirred reaction beakers (although we impose compartmentalization, e.g., separating cell faces), an approximation that no one even remotely believes. Indeed, almost every well-understood developmental mechanism includes some fascinating and functionally important instance in which the structure of cells plays a crucial role: apical–basal sorting of receptors and ligands; the flexibility of stretches of chromatin; clustering of receptors; etc. Models of all kinds must balance tractability with realism. In Ingeneue we have chosen a certain level of realism which may limit its usefulness but which allows us to use it to explore parameter space in a way that a more realistic modeling framework would preclude because of the computational burden.

Another weak spot in our approach is the problem of recognizing patterns. Although we have a mechanism in place to build complex pattern recognizers from smaller pieces, pattern recognition is a hard problem, and it is not clear whether our mechanism will work in general without imposing an onerous coding and training burden on the user. Thus some uses of Ingeneue are still likely to involve clever custom programming, although this will hopefully decrease as the program matures. No matter what, we doubt very much that any computer recipe can, without extensive training, replace the intuition of the human biologist when it comes to pattern recognition.

We have been using Ingeneue to address a wide variety of questions in developmental and evolutionary biology (von Dassow et al., 2000; Meir et al., 2002; Odell et al., unpublished observations; von Dassow and Odell, 2002). These include simply asking, how complete is our current understanding of a gene network? Given the known facts, is the proposed mechanism plausible? And, how do the different components of a network contribute to its behavior? We have also used Ingeneue to ask how important particular classes of interactions are to the functioning of a network, and to explore how the structure of a

network affects its function. The examples above illustrate some of these results. Most of all, we feel that through model-building we can translate what the community of developmental biologists knows about developmental mechanics into the theoretical framework of evolutionary biology. For example, our model of the segment polarity network unexpectedly yielded a putative example of the mechanistic origins of canalization. We hope that these beginnings and the free availability of this program will inspire other biologists to try similar explorations.

ACKNOWLEDGMENTS

Thanks to Dara Lehman for comments on early drafts. Funding for the development of Ingeneue and for other work mentioned here came from the National Science Foundation (MCB-9732702, MCB-9817081, MCB-0090835). Much of the work described here was done at Friday Harbor Laboratories, and we thank Dennis Willows and the FHL staff for space and support. Appendix B was previously published as part of GvD's doctoral thesis (University of Washington Department of Zoology, 2000). We dedicate this paper to the memory of Dr. DeLill Nasser, program director in genetics at NSF. DeLill encouraged us even in the early exploratory stages of this work, before there was any buzz about bio-informatics, when even we ourselves weren't sure this project would amount to much. We are deeply grateful for her open-minded vision.

LITERATURE CITED

Arnosti DN, Barolo S, Levine M, Small S. 1996. The eve stripe 2 enhancer employs multiple modes of transcriptional synergy. Development 122:205–214.

Aza-Blanc P, Ramirez-Weber FA, Laget MP, Schwartz C, Kornberg TB. 1997. Proteolysis that is inhibited by hedgehog targets Cubitus interruptus protein to the nucleus and converts it to a repressor. Cell 89:1043–1053.

Barkai N, Leibler S. 1997. Robustness in simple biochemical networks. Nature 387:913–917.

Barkai N, Leibler S. 2000. Circadian clocks limited by noise. Nature 403:267–268.

Bhat KM, van Beers EH, Bhat P. 2000. Sloppy paired acts as the downstream target of wingless in the *Drosophila* CNS and interaction between sloppy paired and gooseberry inhibits sloppy paired during neurogenesis. Development 127:655–665.

Bray D, Levin MD, Morton-Firth CJ. 1998. Receptor clustering as a cellular mechanism to control sensitivity. Nature 393:85–88.

Burden RL, Faires JD, Reynolds AC. 1978. Numerical analysis. Boston: Prindle, Weber and Schmidt.

Burke R, Nellen D, Bellotto M, Hafen E, Senti KA, Dickson BJ, Basler K. 1999. Dispatched, a novel sterol-sensing domain protein dedicated to the release of cholesterol-modified hedgehog from signaling cells. Cell 99:803–815.

Cadigan KM, Grossniklaus U, Gehring WJ. 1994a. Functional redundancy: the respective roles of the two sloppy paired genes in *Drosophila* segmentation. Proc Natl Acad Sci U S A 91:6324–6328.

Cadigan KM, Grossniklaus U, Gehring WJ. 1994b. Localized expression of sloppy paired protein maintains the polarity of *Drosophila* parasegments. Genes Dev 8:899–913.

Cash JR, Karp AH. 1990. A variable order Runge–Kutta method for initial value problems with rapidly varying right-hand sides. ACM Trans Math Software 16:201–222.

Dierick HA, Bejsovec A. 1998. Functional analysis of Wingless reveals a link between intercellular ligand transport and dorsal-cell-specific signaling. Development 125:4729–4738.

Driever W. 1993. Maternal control of anterior development in the *Drosophila* embryo. In: Bate M, Martinez-Arias A, editors. The development of *Drosophila melanogaster*. Plainview, NY: Cold Spring Harbor Laboratory Press. p 301–324.

Driever W, Thoma G, Nusslein-Volhard C. 1989. Determination of spatial domains of zygotic gene expression in the *Drosophila* embryo by the affinity of binding sites for the bicoid morphogen. Nature 340:363–367.

Edgar BA, Weir MP, Schubiger G, Kornberg T. 1986. Repression and turnover pattern fushi tarazu RNA in the early *Drosophila* embryo. Cell 47:747–754.

Edgar BA, Odell GM, Schubiger G. 1989. A genetic switch, based on negative regulation, sharpens stripes in *Drosophila* embryos. Dev Genet 10:124–142.

Gibson G. 1996. Epistasis and pleiotropy as natural properties of transcriptional regulation. Theor Popul Biol 49:58–89.

Gibson G, Wagner G. 2000. Canalization in evolutionary genetics: a stabilizing theory? BioEssays 22:372–380.

Goodrich LV, Johnson RL, Milenkovic L, McMahon JA, Scott MP. 1996. Conservation of the hedgehog/patched signaling pathway from flies to mice: induction of a mouse patched gene by Hedgehog. Genes Dev 10:301–312.

Grossniklaus U, Pearson RK, Gehring WJ. 1992. The *Drosophila* sloppy paired locus encodes two proteins involved in segmentation that show homology to mammalian transcription factors. Genes Dev 6:1030–1051.

Gurdon JB, Harger P, Mitchell A, Lemaire P. 1994. Activin signalling and response to a morphogen gradient. Nature 371:487–492.

Johnson LW, Reiss RD. 1982. Numerical analysis. Reading, MA: Addison-Wesley.

Kauffman SA. 1993. The origins of order: self-organization and selection in evolution. New York: Oxford University Press. xviii, 709 pp.

La Rosee A, Hader T, Taubert H, Rivera-Pomar R, Jackle H. 1997. Mechanism and Bicoid-dependent control of hairy stripe 7 expression in the posterior region of the *Drosophila* embryo. EMBO J 16:4403–4411.

Laub MT, Loomis WF. 1998. A molecular network that produces spontaneous oscillations in excitable cells of *Dictyostelium*. Mol Biol Cell 9:3521–3532.

Lee HH, Frasch M. 2000. Wingless effects mesoderm patterning and ectoderm segmentation events via induction of its downstream target sloppy paired. Development 127: 5497–5508.

Marigo V, Scott MP, Johnson RL, Goodrich LV, Tabin CJ. 1996. Conservation in hedgehog signaling: induction of a

chicken patched homolog by Sonic hedgehog in the developing limb. Development 122:1225–1233.
McAdams HH, Shapiro L. 1995. Circuit simulation of genetic networks. Science 269:650–656.
Meir E, von Dassow G, Munro E, Odell GM. 2002. Robustness, flexibility, and the role of lateral inhibition in the neurogenic network. Curr Biol 12:778–786.
Moline MM, Southern C, Bejsovec A. 1999. Directionality of wingless protein transport influences epidermal patterning in the *Drosophila* embryo. Development 126:4375–4384.
Murray JD. 1993. Mathematical biology. Berlin: Springer-Verlag. xiv, 767 pp.
Nellen D, Burke R, Struhl G, Basler K. 1996. Direct and long-range action of a DPP morphogen gradient. Cell 85:357–368.
Pai LM, Orsulic S, Bejsovec A, Peifer M. 1997. Negative regulation of Armadillo, a Wingless effector in *Drosophila*. Development 124:2255–2266.
Pfeiffer S, Vincent JP. 1999. Signalling at a distance: transport of Wingless in the embryonic epidermis of *Drosophila*. Semin Cell Dev Biol 10:303–309.
Porter JA, Ekker SC, Park WJ, von Kessler DP, Young KE, Chen CH, Ma Y, Woods AS, Cotter RJ, Koonin EV, Beachy PA. 1996. Hedgehog patterning activity: role of a lipophilic modification mediated by the carboxy-terminal autoprocessing domain. Cell 86:21–34.
Press WH. 1992. Numerical recipes in C: the art of scientific computing. New York: Cambridge University Press. xxvi, 994 pp.
Ptashne M. 1992. A genetic switch: phage l and higher organisms. Cambridge, MA: Cell Press, Blackwell Scientific Publications. ix, 192 pp.
Reinitz J, Sharp DH. 1995. Mechanism of eve stripe formation. Mech Dev 49:133–158.
Slack JMW. 1983. From egg to embryo: determinative events in early development. New York: Cambridge University Press. 241 pp.
Struhl G, Struhl K, Macdonald PM. 1989. The gradient morphogen bicoid is a concentration-dependent transcriptional activator. Cell 57:1259–1273.
Thieffry D, Huerta AM, Perez-Rueda E, Collado-Vides J. 1998. From specific gene regulation to genomic networks: a global analysis of transcriptional regulation in *Escherichia coli*. BioEssays 20:433–440.
Tyson JJ, Novak B, Odell GM, Chen K, Thron CD. 1996. Chemical kinetic theory: understanding cell-cycle regulation. Trends Biochem Sci 21:89–96.
von Dassow G, Meir E, Munro EM, Odell GM. 2000. The segment polarity network is a robust developmental module. Nature 406:188–192.
von Dassow G, Odell GM. 2002. Design and constraints of the Drosophila segment polarity module: robust spatial patterning emerges from intertwined cell state switches. J Exp Zool (Mol Dev Evol) 294:179–213.
Weir MP, Edgar BA, Kornberg T, Schubiger G. 1988. Spatial regulation of engrailed expression in the *Drosophila* embryo. Genes Dev 2:1194–1203.

APPENDIX A: MATHEMATIZING GENE REGULATORY INTERACTIONS

We outline a general approach for converting empirical descriptions of network interactions into the mathematical formulae that Affector objects encapsulate. We focus on transcription as an example because it is a form of network interaction that involves a wide range of regulatory phenomena, but the methods we describe can be readily applied to the other types of Affectors.

General form of transcriptional Affector equations

A transcriptional Affector determines the transcription rate for a Node as a function of all regulatory influences impinging on that Node (each Node has at most one primary synthesis Affector). In dimensional terms, the rate equation for a transcriptional Affector has the general form

$$\frac{d[x]}{dt} = T_{\max}\rho_x F([R_1],[R_2],\ldots,[R_N]). \qquad \text{(A1)}$$

$T_{\max}$ is the maximal possible transcription rate for any gene, determined by the enzymology of RNA polymerase, and ρ_x is an efficiency characteristic of gene x, reflecting how good a template x is. $F:\Re_N \rightarrow [0,1]$, a function of the regulator concentrations $([R_1],[R_2],\ldots,[R_N])$, determines at what fraction of full efficiency x is transcribed. By definition, values of F must lie between 0 and 1 inclusive. Building an Affector formula for a given regulation mechanism amounts to determining an appropriate functional form for F, and parametrizing it.

A simple example: transcriptional activation

We start with a very simple example which forms the basis for all the more complicated regulatory schemes described below. Consider a single regulatory factor A, which activates transcription of gene x. For mathematical clarity, we focus attention on one copy of x (unless transcriptional regulators are present in very low numbers, the overall transcription rate is simply twice the rate from one copy). Without any significant loss of generality, we can think of the underlying mechanism as involving two steps (Fig. A1): in the first step, A interacts with a specific DNA sequence in a concentration-dependent way to form an ''active complex'' A*. For example, A might bind directly to DNA, or bind as part of a larger complex involving other molecules, or catalyze the formation of a complex which binds to DNA without itself doing so. In the second step, the active complex A* interacts with a proximal promoter region to boost the rate at which new

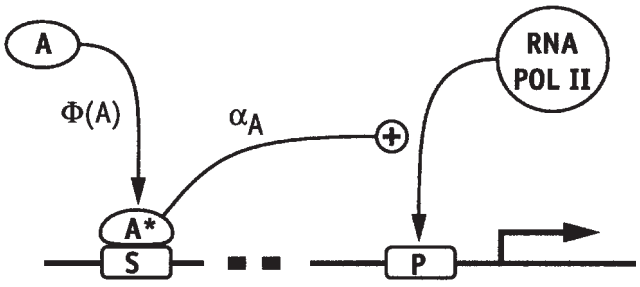


Fig. A1. Schematic view of transcriptional regulation as a two-step process. In step 1, regulator A forms an active complex A* around sequence S in a concentration-dependent manner. The fraction of bound complex is given by Ψ(A). In step 2, the active complex A* regulates the rate at which transcription is initiated at the proximal promoter with efficiency $0 \leq \alpha_A \leq 1$ (see text for details).

transcripts are initiated. Again, there are many possible ways this could occur: A* could modify the local structure of chromatin to allow greater access to other transcription factors, or associate more directly with proximal transcriptional machinery or the proximal promoter to recruit one to the other, etc.

With this general mechanism in mind, we can write $F(A)$ as

$$F(A) = \alpha_A \Phi(A), \tag{A2}$$

where $\Phi(A)$ is the fraction of time that the active complex A* exists as a function of A's concentration, and α_A is the efficiency with which A* boosts transcription of x. Alternatively, we can think of $F(A)$ as representing the probability that x is transcriptionally active at a given point in time, where transcriptionally active means that x is transcribed at its maximum rate $T_{\max}\rho_x$. In this case, $\Phi(A)$ represents the probability that an active complex exists at a given point in time, and α_A represents the conditional probability that x is transcriptionally active, given that an active complex exists, or simply the probability that A* activates x. This probabalistic interpretation turns out to be very useful in building representations of more complex forms of regulation, as we will return to below.

In the case of simple activation, and in the absence of any detailed knowledge about how A interacts with gene x, we use the simple generic form for $\Phi(A)$:

$$\Phi(A) = \frac{\left(\frac{[A]}{K_A}\right)^{\nu_A}}{1 + \left(\frac{[A]}{K_A}\right)^{\nu_A}} = \left(\frac{[A]^{\nu_A}}{K_A^{\nu_A} + [A]^{\nu_A}}\right), \tag{A3}$$

where K_A is the value at which $\Phi(A) = 0.5$ and ν_A is the degree of cooperativity. As discussed in the text, we can think of $\Phi(A)$ as a measurable dose–response curve, expressing in the most general terms what must be the case for an interaction between A and x: that the magnitude of the interaction must increase and then saturate with increasing concentrations of A, that it does so more or less steeply, and that A has a maximal effectiveness as an activator of x.

To represent more complicated forms of transcriptional regulation involving multiple regulators, we replace $\Phi(A)$ and $_A$ with more complicated expressions, which we must deduce from what we know about those additional regulators and their interactions. We have taken two general approaches to doing so. The first generalizes from simple kinetics to produce replacements for $\Phi(A)$. The second represents interactions as logical conditions and uses simple probability theory.

Deducing regulatory forms from steady-state kinetics

Many transcriptional regulators are known to interact either before or during complex formation. For example, different activators bind to DNA as heterodimers, or otherwise assemble active complexes around the same or nearby DNA sequences. Many inhibitors act by competing with an activator for its binding site or by binding to and titrating away an activator before it binds. In these cases, we can deduce suitable regulatory forms by 1) writing down the simplest kinetic scheme that summarizes (or caricatures) what we know about the binding interactions among regulators and DNA sequences; 2) solving the resulting kinetic rate equations to find the steady state concentration of whatever complex we are interested in; 3) where necessary or convenient, approximating the resulting steady state expression with an appropriate simpler form; and 4) generalizing the resulting form in an appropriate way to allow for cooperative effects. We will illustrate this approach with several examples.

Example 1: A single activator binds a target DNA sequence

Consider first the simple kinetic scheme in which activator A binds to a specific sequence S within the regulatory region of gene x to form the bound complex AS (Fig. A2A):

$$A + S \underset{k^-}{\overset{k^+}{\Leftrightarrow}} AS. \tag{A4}$$

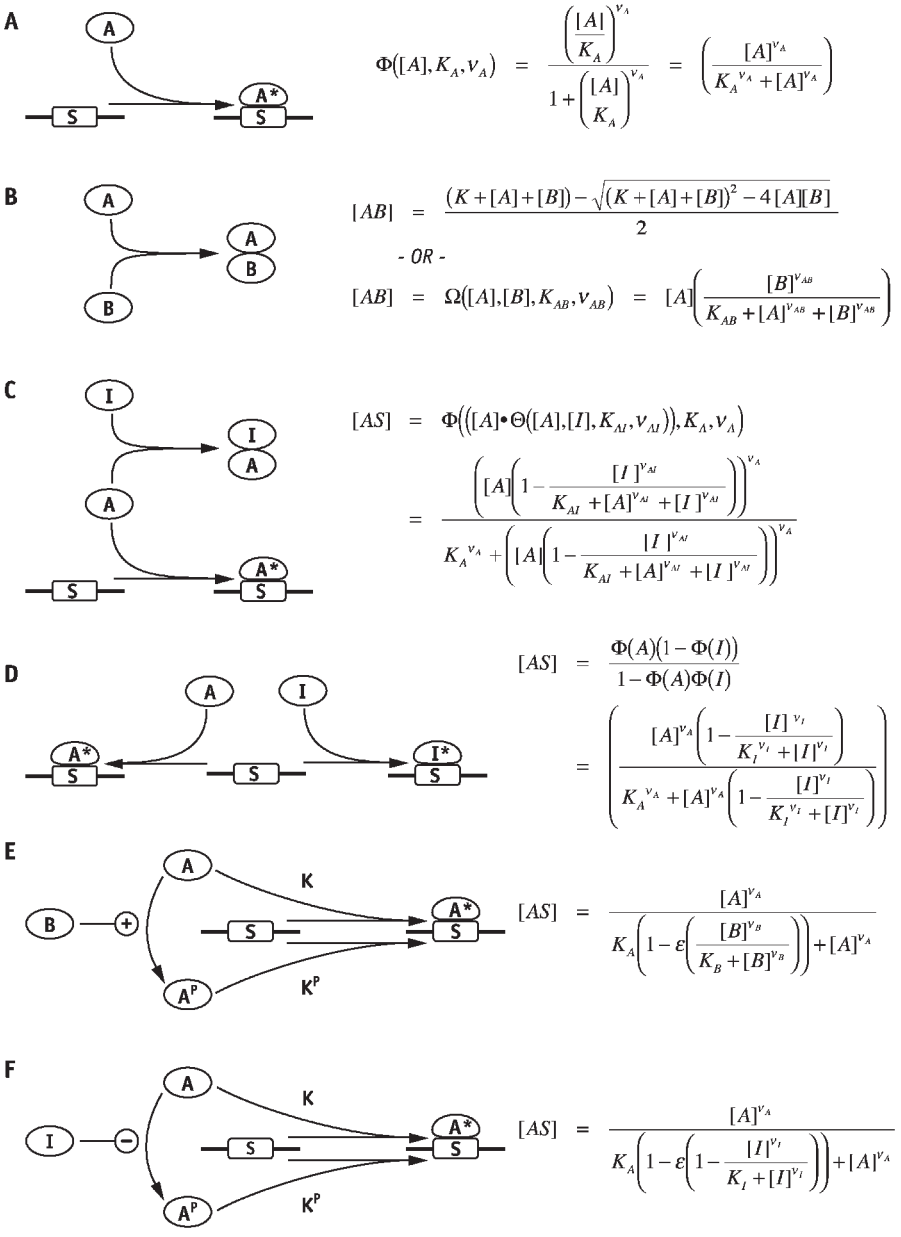
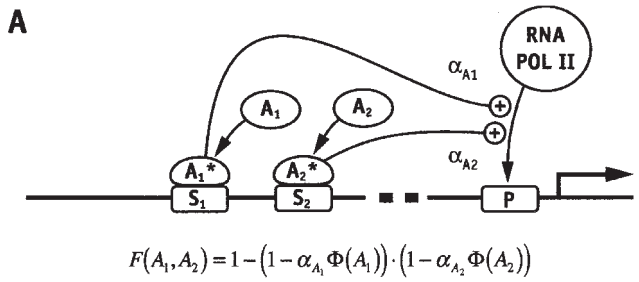
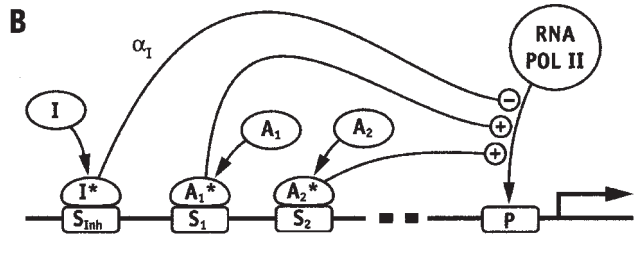


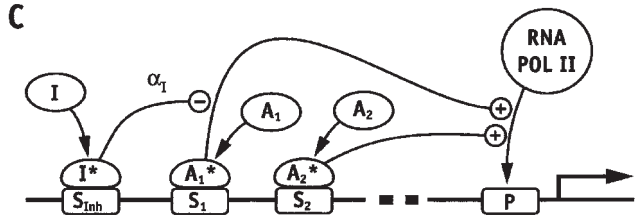
Fig. A2. Deducing pseudo-kinetic representations of regulatory interactions from kinetic interaction diagrams. Left column shows interaction diagrams. Right column shows the algebraic forms we deduce from those diagrams by assuming all binding reactions are at steady state, making use of simple approximations where necessary, and generalizing to include cooperative effects. S indicates a specific DNA sequence. A and B represent transcriptional activators, and I represents a transcriptional inhibitor. Arrows indicate reversible reactions (only the forward half of each reaction is shown for clarity). Plus sign (or minus sign) within a circle indicates a positive (or negative) regulation of the indicated reaction. (**A**) An activator binds a target DNA sequence (present at two copies per cell) to form an active complex. The fraction of bound complex is given by the generalized function $\Psi(A)$ shown to the right. (**B**) Two molecules (each present at arbitrary concentrations) bind to form a dimer complex. To the right are the exact concentration of dimer, and the approximate form $\Omega(A,B)$ generalized to arbitrary cooperativity. (**C**) An inhibitor I binds to and titrates away an activator A before it binds to a target DNA sequence S to form an active complex. The fraction of active complex is given by the same function Ψ shown in (**A**), but in this case, the amount of A available to bind its target is the total amount of A minus the amount of bound activator–inhibitor complex $\Omega(A,I)$ as shown to the right. (**D**) An inhibitor acts to competitively exclude A from binding to its target sequence S. Here we show I binding to S itself, but it could also bind to an adjacent sequence. To the right is the generalized steady-state form, obtained by assuming that A and I bind with entirely independent affinities and cooperativities. (**E**) Regulator B boosts A's efficacy as a transcriptional activator by converting it from a low-affinity form (A) to a high-affinity form (A^P). In this case, several successive approximations are required to obtain the simple form shown to the right. (**F**) Similar to **E**, but in this case I converts A from a high-affinity to low-affinity form.



$$F(A_1,A_2)=1-\left(1-\alpha_{A_1}\Phi(A_1)\right)\cdot\left(1-\alpha_{A_2}\Phi(A_2)\right)$$



$$F(A_1,A_2,I) = \left(1-\alpha_I\cdot\Phi(I)\right)\cdot\left(1-\left(1-\alpha_{A_1}\Phi(A_1)\right)\cdot\left(1-\alpha_{A_2}\Phi(A_2)\right)\right)$$



$$F(I,A_1,A_2)=1-\left(1-\alpha_{A_1}\cdot\left(1-\alpha_I\cdot\Phi(I)\right)\cdot\Phi(A_1)\right)\cdot\left(1-\alpha_{A_2}\Phi(A_2)\right)$$

Fig. A3. Deducing regulatory forms using simple probabilities. (**A**) Two activators, A_1 and A_2, bind to separate enhancer elements, S_1 and S_2, and independently activate transcription from the proximal promotor with probabilities (efficiencies) α_1 and α_2. (**B**) A single inhibitor I binds to its enhancer element S_I and globally inhibits transcription as activated by either A_1 or A_2 with probability (efficiency) α_I. (**C**) Inhibitor I binds to enhancer element S_I and inhibits A_1's activation of transcription with probability (efficiency) α_I but has no effect on transcription as activated by A_2.

The resulting equations are:

$$\frac{d[AS]}{dt}=k^+[A][S]-k^-[AS],\ [AS]+[S]=1, \quad \text{(A5)}$$

where the second equation expresses the stoichiometric fact that there is a single sequence which must be in either a bound or unbound state. We then set $d[AS]/dt=0$ and solve for the steady-state fraction of bound complex AS to obtain

$$[AS]=\frac{\frac{[A]}{K}}{1+\frac{[A]}{K}}=\frac{[A]}{K+[A]}, \quad \text{(A6)}$$

where $K=k^-/k^+$ becomes the half-maximal activation coefficient. To obtain the generic form $\Phi([A])$, we simply apply a cooperativity exponent ν_A to each appearance of [A] and K in Eq. (A6).

Example 2: Protein dimerization and its generalization

Now consider a similar reaction in which A and B are two proteins that bind to form the heterodimer AB (Fig. A2B):

$$A+B\underset{k^-}{\overset{k^+}{\Leftrightarrow}}AB. \quad \text{(A7)}$$

Unlike Example 1 where [A] changes negligibly as a consequence of the reaction, in general, both [A] and [B] might change as a consequence of dimerization, and indeed we may often be interested in the fraction remaining unbound. Thus we need to be more explicit in the associated kinetic equations:

$$\begin{aligned}
&\frac{d[A_{\text{free}}]}{dt}=k^-\cdot[AB]-k^+\cdot[A_{\text{free}}][B_{\text{free}}],\\
&\frac{d[B_{\text{free}}]}{dt}=k^-\cdot[AB]-k^+\cdot[A_{\text{free}}][B_{\text{free}}],\\
&\frac{d[AB]}{dt}=k^+\cdot[A_{\text{free}}][B_{\text{free}}]-k^-\cdot[AB],\\
&[A]=[A_{\text{free}}]+[AB],\\
&[B]=[B_{\text{free}}]+[AB].
\end{aligned} \quad \text{(A8)}$$

Here [A] and [B] represent the total concentrations of reactants, $[A_{\text{free}}]$ and $[B_{\text{free}}]$ represent the concentrations of free (unbound) reactants, and [AB] is the total concentration of bound complex. Solving Eq. (A8) at steady state (or just using the mass action law's formula for the equilibrium condition, $K_{\text{eq}}=[AB]/[A_{\text{free}}][B_{\text{free}}]$), substituting for $[A_{\text{free}}]$ and $[B_{\text{free}}]$ from the mass balance equations, and using $K=k^-/k^+=1/K_{\text{eq}}$ we obtain the quadratic equation

$$\frac{([A]-[AB])([B]-[AB])}{K}-[AB]=0 \quad \text{(A9)}$$

with a single positive root given by

$$[AB]=\frac{(K+[A]+[B])-\sqrt{(K+[A]+[B])^2-4\cdot[A][B]}}{2}. \quad \text{(A10)}$$

Equation (A10) is a perfectly good way to estimate the steady state of a dimerization reaction exactly, but it is non-intuitive and difficult to generalize (i.e., to add cooperativity). To this end, we introduce a simpler function:

$$[AB] = \Omega([A],[B],K_{AB}) = \frac{\frac{[A][B]}{K_{AB}}}{1+\frac{[A]}{K_{AB}}+\frac{[B]}{K_{AB}}} = \frac{[A][B]}{K_{AB}+[A]+[B]}. \quad \text{(A11a)}$$

If we want the fraction of [A] (or [B]) unbound we use the relation $[A_{\text{free}}] = [A] - [AB]$:

$$[A_{\text{free}}] = \Theta([A],[B],K_{AB}) = [A] - \frac{[A][B]}{K_{AB}+[A]+[B]} = [A]\left(1-\frac{[B]}{K_{AB}+[A]+[B]}\right). \quad \text{(A11b)}$$

Equation (A11) has the same qualitative properties as Eq. (A10): it is symmetric to exchange of A and B; for fixed [A], it saturates to the value of [A] with increasing [B] (and vice versa); it increases without bound as both [A] and [B] increase. It is a good approximation to Eq. (A10) in the sense that, for any value of K in Eq. (A10), we can find a value of K_{AB} in Eq. (A11) such that the difference between Eqs. (A10) and (A11) is small for all values of [A] and [B] (we formalize this notion below). Finally, we can generalize Eq. (A11) to add a ''cooperativity'' effect[6] as follows:

$$\begin{aligned}[AB] &= \Omega([A],[B],K_{AB},\nu_{AB}) = [A]\left(\frac{[B]^{\nu_{AB}}}{K_{AB}^{\nu_{AB}}+[A]^{\nu_{AB}}+[B]^{\nu_{AB}}}\right),\\ [A_{\text{free}}] &= \Theta([A],[B],K_{AB},\nu_{AB}) = [A]\left(1-\frac{[B]^{\nu_{AB}}}{K_{AB}^{\nu_{AB}}+[A]^{\nu_{AB}}+[B]^{\nu_{AB}}}\right).\end{aligned} \quad \text{(A12)}$$

Example 3: A form of competitive inhibition

Here is how we would use the same approach as in Example 1 to model competitive inhibition in which activator A and inhibitor I bind to the same sequence S (Fig. A2D). In this case, the kinetic scheme is

$$\begin{aligned}A + S &\underset{k_A^-}{\overset{k_A^+}{\Leftrightarrow}} AS,\\ I + S &\underset{k_I^-}{\overset{k_I^+}{\Leftrightarrow}} IS.\end{aligned} \quad \text{(A13)}$$

The resulting equations are

$$\begin{aligned}\frac{d[AS]}{dt} &= k_A^+[A][S] - k_A^-[AS],\\ \frac{d[IS]}{dt} &= k_I^+[I][S] - k_I^-[IS],\\ [AS]+[IS]+[S] &= 1,\end{aligned} \quad \text{(A14)}$$

and solving them at steady state yields

$$[AS] = \frac{F(A)(1-F(I))}{1-F(A)F(I)}, \quad \text{(A15)}$$

where

$$F(A) = \frac{\frac{[A]}{K_A}}{1+\frac{[A]}{K_A}}, \quad K_A = \frac{k_A^-}{k_A^+}, \quad \text{(A16)}$$

as for Example 1 above. We can generalize this form exactly as described for Example 1 above by replacing the functions *F(A) and F(I)* with the generic forms $\Phi(A)$ and $\Phi(I)$ to obtain:

$$\begin{aligned}[AS] &= \frac{\Phi(A)(1-\Phi(I))}{1-\Phi(A)\Phi(I)} = \left(\frac{[A]^{\nu_A}\left(1-\frac{[I]^{\nu_I}}{K_I^{\nu_I}+[I]^{\nu_I}}\right)}{K_A^{\nu_A}+[A]^{\nu_A}\left(1-\frac{[I]^{\nu_I}}{K_I^{\nu_I}+[I]^{\nu_I}}\right)}\right)\\ &= \left(\frac{\left(\frac{A}{K_A}\right)^{\nu_A}}{1+\left(\frac{A}{K_A}\right)^{\nu_A}+\left(\frac{I}{K_I}\right)^{\nu_I}}\right).\end{aligned} \quad \text{(A17)}$$

However, Eq. (A15) could be generalized differently, such that I affects the concentration of A *before* application of the exponent in Φ. This corresponds to a choice that is simple to explain mathematically but rather complicated to unwind mechanistically: does the inhibitor affect the effective concentration of A directly, or does it affect A activity ''after'' A has assembled into whatever bound complex confers the cooperativity exponent? Figure A5B and C contrast these two choices. A fourth choice, not plotted in Fig. A5 but diagrammed as Fig. A2C, would multiply [A]

[6]This introduces an asymmetry between A and B. However, this is realistic. Cooperative binding phenomena usually involve a single macromolecule (say, a single molecule of A, even if A is in fact a protein complex) that binds several ligands. Examples include hemoglobin binding to oxygen, and IPTG binding to the *lac* repressor; both macromolecules are tetramers. A typical treatment of such a case assumes the ligand is present in vast excess, and thus results in Eq. (A3). However, if the ligand could be present at similar concentration to its partner, or if what we know is the *total* ligand concentration rather than the concentration of unbound ligand, then Eq. (A3) is inadequate and potentially misleading. The rigorous treatment of such a case is complicated, and we emphasize that Eqs. (A11) and (A12) are only offered as convenient approximations whose qualitative behavior matches our expectations of the behavior of such a system.

by $\Theta([A], [I], K_{IA}, \nu_{IA})$ prior to the computation of $\Phi([A_{\text{free}}], K_A, \nu_A)$; this approximates a mechanism in which the inhibitor titrates away free A before A binds its target sequence.

Figure A2 shows some of the other forms one can readily deduce to describe different types of regulatory interaction. The forms fall into two general classes. Forms in the first class (Fig. A2A and A2C–F) are dimensionless and have values that lie strictly within the interval [0,1]. Each of these forms represents the fraction of time that some active complex exists and therefore ''fits'' into the general form of F(A) given as Eq. (A2) above. Forms in the second class (Fig. A2B) have dimensions of concentration and therefore fit as arguments to the general form F(A). All reduce to the appropriate simpler forms when one or more of the regulators are absent.

We want to stress that these forms are not intended to be literal kinetic representations of some detailed regulatory mechanism, which will usually involve, in reality, a great deal more algebraic gore. If we knew all of the details of such a mechanism, then we could deduce such a literal representation and incorporate it into the framework described here. But even if we could, there would be little advantage in using a detailed representation of one interaction within a network if we are only using cruder approximations somewhere else. However, it is instructive to think of the equations we deduce as approximations to the literal kinetic forms and then ask whether they are ''good'' approximations.

To do so, we must define what we mean by ''good''. Suppose $F_{\text{approx}}(R_1, R_2, p_1, p_2, p_3)$ is an approximate form deduced as above for two regulators R_1 and R_2 of gene x, and parameterized by p_1, p_2, and p_3. Suppose further that we actually knew all of the underlying molecular players involved in mediating interactions of R_1 and R_2 with x. Then we could write down a complex set of kinetic equations expressing these interactions and solve them at steady state to deduce an ''exact'' (not really) function $F_{\text{exact}}(R_1, R_2, C_1, ..., C_N, k_1, ...k_M)$ where the C's and k's are the concentrations[7] of those underlying players and kinetic parameters governing their interactions. The set of all possible values for the parameters $(C_1, \ldots, C_N, k_1, \ldots, k_M)$ determines a range of shapes of F_{exact} as a function of R_1 and R_2. We say that $F_{\text{approx}}(R_1, R_2)$ is a good approximation of $F_{\text{exact}}(R_1, R_2)$ if it manifests a quantitatively similar set of shapes when we vary the parameters (p_1, p_2, p_3). Another way to say this is that $F_{\text{approx}}(R_1, R_2)$ is a good approximation of $F_{\text{exact}}(R_1, R_2)$ if for any choice of parameters for F_{exact} $(C'_1, \ldots, C'_N, k'_1, \ldots, k'_N)$, there exists a choice of values (p'_1, p'_2, p'_3), such that $F_{\text{exact}}(R_1, R_2, C'_1, \ldots, C'_N, k'_1, \ldots, k'_N)$ and $F_{\text{approx}}(p'_1, p'_2, p'_3)$ have similar *shapes*.

Of course, we cannot compare our approximate representations to the literal ones because we do not know what the literal ones are. But we can compare two different approximate schemes that represent the same regulatory scheme at different levels of resolution (where one might include additional intermediates or auxiliary proteins present at constant concentrations). Where we have done so, we find that the simpler schemes are surprisingly good at approximating the more complicated ones, good enough that we can find no reason to use the more complicated ones in our network models.

We can use this same notion to compare different ways of representing the same set of empirically observed regulatory interactions. For example, we deduced many of the regulatory forms we used in our early models by intuition, seeking formulae whose properties expressed adequately what we took to be the basic empirical facts. In many cases, we find that different forms are equivalently good representations of a particular regulatory interaction in the sense that each approximates the others as defined above (see Figs. A4 and A5). In this case, we are free to apply other criteria to choose the ''best'' one, e.g., which is most easily parametrized, or makes the most biological sense, or combines most easily with other forms.

We emphasize that the important distinction between these different formula options is the distinction between their graphs (Figs. A4 and A5) and the qualitative properties thereof. For example, compare the shapes in Fig. A5A,D,G,J with the shapes in Fig. A5C,F,I,L; in Fig. A5D, a particular concentration of the inhibitor (moving to the left) squelches activation by a certain fraction no matter what the level of activator; in Fig. A5F, this is not so. The difference is that in the formula governing the left column the inhibitor regulates the maximal activity level, whereas in the right column the inhibitor regulates how much activator is required to achieve that level, and in the middle column inhibitor

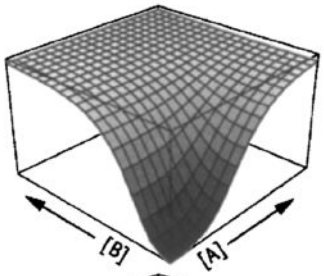
[7]We assume that the concentrations of the underlying players are constant on the timescale of interest. Otherwise, we would have to include them in our pseudo-kinetic representation.

$$\frac{d[x]}{dt} = T_{max}\rho_x \cdot F([A],[B]) - \frac{[x]}{\tau_x}$$

$\kappa_{Ax} = 0.9$; $\kappa_{Bx} = 0.5$; $\nu_{Ax} = 4$; $\nu_{Bx} = 3$

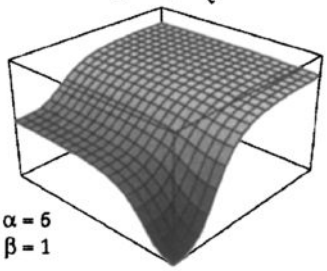
A

$$\left(\frac{\left(\frac{[A]^{\nu_{Ax}}}{\kappa_{Ax}{}^{\nu_{Ax}}}\right)+\left(\frac{[B]^{\nu_{Bx}}}{\kappa_{Bx}{}^{\nu_{Bx}}}\right)}{1+\left(\frac{[A]^{\nu_{Ax}}}{\kappa_{Ax}{}^{\nu_{Ax}}}\right)+\left(\frac{[B]^{\nu_{Bx}}}{\kappa_{Bx}{}^{\nu_{Bx}}}\right)}\right)$$



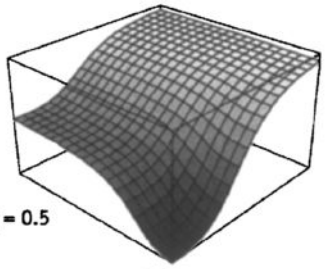
B

$$\left(\frac{\alpha\left(\frac{[A]^{\nu_{Ax}}}{\kappa_{Bx}{}^{\nu_{Ax}}+[A]^{\nu_{Ax}}}\right)+\beta\left(\frac{[B]^{\nu_{Bx}}}{\kappa_{Bx}{}^{\nu_{Bx}}+[B]^{\nu_{Bx}}}\right)}{1+\alpha\left(\frac{[A]^{\nu_{Ax}}}{\kappa_{Bx}{}^{\nu_{Ax}}+[A]^{\nu_{Ax}}}\right)+\beta\left(\frac{[B]^{\nu_{Bx}}}{\kappa_{Bx}{}^{\nu_{Bx}}+[B]^{\nu_{Bx}}}\right)}\right)$$



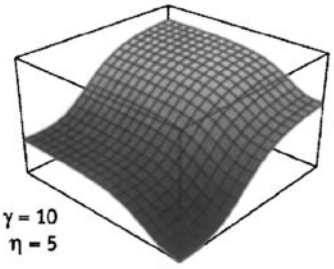
C

$$\left(1-\left(1-\left(\frac{[A]^{\nu_{Ax}}}{\kappa_{Bx}{}^{\nu_{Ax}}+[A]^{\nu_{Ax}}}\right)\right)\left(1-\chi\left(\frac{[B]^{\nu_{Bx}}}{\kappa_{Bx}{}^{\nu_{Bx}}+[B]^{\nu_{Bx}}}\right)\right)\right)$$



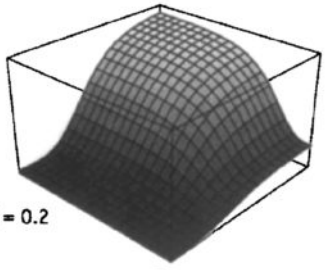
D

$$\left(\frac{Max[\gamma,\eta]}{\gamma+\eta}\right)\left(\frac{\gamma\left(\frac{[A]^{\nu_{Ax}}}{\kappa_{Bx}{}^{\nu_{Ax}}+[A]^{\nu_{Ax}}}\right)+\eta\left(\frac{[B]^{\nu_{Bx}}}{\kappa_{Bx}{}^{\nu_{Bx}}+[B]^{\nu_{Bx}}}\right)}{1+Max[\gamma,\eta]}\right)$$



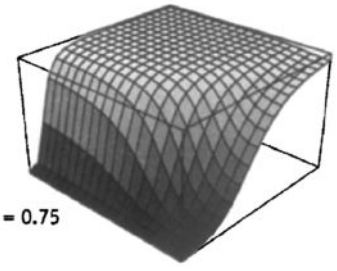
E

$$\left(\frac{[A]^{\nu_{Ax}}}{\kappa_{Bx}{}^{\nu_{Ax}}+[A]^{\nu_{Ax}}}\right)\left(\delta+(1-\delta)\frac{[B]^{\nu_{Bx}}}{\kappa_{Bx}{}^{\nu_{Bx}}+[B]^{\nu_{Bx}}}\right)$$



F

$$\left(\frac{[A]^{\nu_{Ax}}}{\left(\kappa_{Bx}\left(1-\varepsilon\left(\frac{[B]^{\nu_{Bx}}}{\kappa_{Bx}{}^{\nu_{Bx}}+[B]^{\nu_{Bx}}}\right)\right)\right)^{\nu_{Ax}}+[A]^{\nu_{Ax}}}\right)$$



regulates availability of the acivator (that is, inhibitor modulates T_{max} versus κ versus [A]). Other scenarios (especially mixes of these) are biologically plausible. Often the model-maker simply will not have the information to choose, and the fact is that in certain circumstances, some shapes work, and others don't (see example of *wg* autoregulation in the companion paper by von Dassow and Odell, '02, this issue).

Using logical descriptions and probabilities to deduce regulatory forms

As mentioned above, we can think of binding fractions and interaction efficiencies in probabilistic terms. Doing so provides a powerful way to deduce representations for more complicated types of regulation. To see how, we must introduce more notation. First, we must define the conditions whose probabilities we would like to represent. Let X be a gene and R be one of its regulators, and let

$\bar{x}$ be the condition : "x is transcriptionally active".

R^* be the condition : "An active complex, representing the interaction between R and x, exists".

$R^* \rightarrow x$ be the condition : "The active complex regulates x".

More generally, we use $A \rightarrow B$ to mean "A has its characteristic effect on B," and we use this notation wherever A and B are such that the condition $A \rightarrow B$ makes sense. Finally, we let P(C) be the probability that condition C holds at a given point in time. For convenience we will use &&, ||, and ! interchangeably with logical AND, OR, and negation, respectively. Returning to the simple case of a single activator A and gene x, the probability that x is transcriptionally active is the probability that A activates x, which is the

Fig. A4. Formulas for transcriptional control by two activators. The differential equation at the top of the figure is typical for mRNAs: the concentration of x increases at a rate proportional to some non-linear function F of various regulators, in this case two activators (A and B), and decreases in proportion to the concentration of x. (**A**–**D**) Various forms of f([A],[B]) with accompanying graphs. All graphs use the parameter values shown at the top, and plot normalized values of [A] and [B] from 0.0 to 2.0. The simple formula in (**A**) has the advantage of reducing to a single Hill function in the absence of either activator. It corresponds to the assumption that A and B bind the same site with different affinities and cooperativities, that each forms an active complex at that site which activates transcription with maximal efficiency α=1. In other words, it's what one obtains from computing the bound fraction of A in the presence of competitor B (Eq. (A17), rightmost coequal form) and adding it to the bound fraction of B in the presence of competitor A. However, using this formula both activators reach the same saturating level, and the formula expresses neither additive nor synergistic effects, thus we have not much used it in practice. (**B**) is an alternative which, through the weights α and β, allows one to adjust the relative ability of each activator to saturate the target promoter, and causes the activators to have an additive, but saturating, effect on transcription rate. However, among other oddities, this formula does *not* reduce to the simple single-activator case in the absence of the other, and furthermore, the weight parameters change the meaning of the half-maximal coefficients κ_{Ax} and κ_{Bx}. Nevertheless this formula may be a reasonable approximation to certain two-step enhancer-binding phenomena, and we have used it in various forms of the segment polarity network model (however, see the companion paper by von Dassow and Odell, '02, this issue). (**C**) is an improvement over (**B**) that, as described in the Appendix text, multiplies the probabilities that either activator will *not* be bound and then takes the complement of this quantity to determine the overall transcriptional activity. This is the same as Eq. (A22) but with a slight change in how the weights are treated (χ is the ratio α_{A1}/α_{A2}; since in this formula *one* term must be able to effect a maximal response, it makes sense to normalize all weights to the largest of them). Formula C/Eq. (A22) *does* reduce to the simple case, *does* preserve the meaning of the half-maximal coefficients, allows for weighting and easy incorporation of inhibitors, and furthermore, readily lends itself to modularization in software such that one can string together an arbitrary number of regulators into a single additive term. We prefer this form for additive activation. **D** represents another possibility for two activators that has many of the virtues of **C** but with different operation of the weights. **E** and **F** represent synergistic, rather than additive, activators. In **E**, one activator, B, has no effect in the absence of the other, A, and on the other hand, A by itself is only a poor activator, but both together fully saturate the promoter. In **F** the second activator, B, functions to lower the half-maximal concentration of A, or in other words it makes A more potent but has no effect itself, but nevertheless A can still achieve full promoter saturation in the absence of the cofactor. **E** suggests an equivalent scheme in which B catalyzes the transformation of A (by phosphorylation or cleavage or what have you) from a low-affinity form to a high-affinity form. The same exact scheme could make B an inhibitor (if it promoted formation of the low-affinity form), as shown in **F**). These six formulas doubtless do not cover all possibilities, but it is apparent that the choice of **B**, **C**, or **D** leaves one with qualitatively similar dose-response behaviors, even though they may represent different kinetic mechanisms and may affect the parameterization of the model differently. The same cannot be said of **E** and **F**, however.

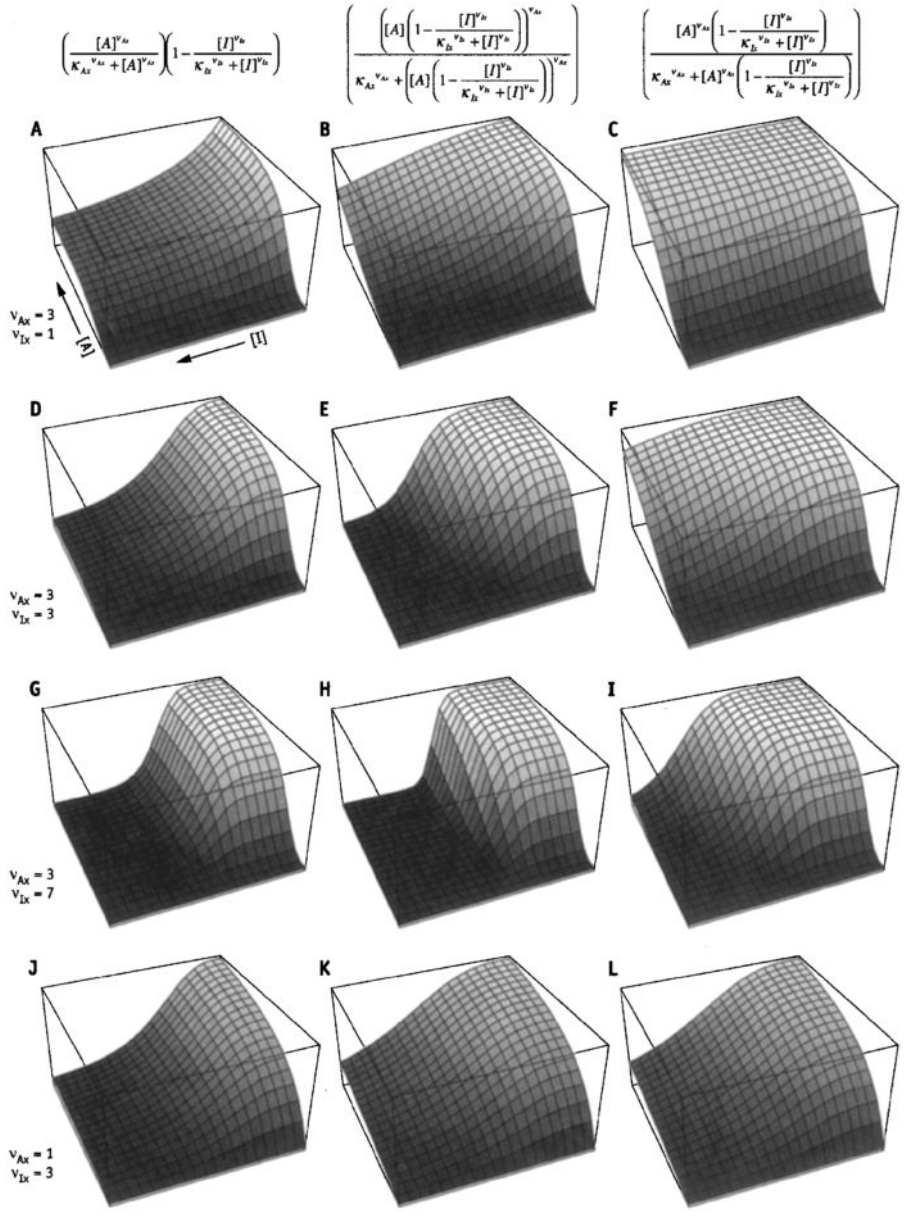


Fig. A5. Formulas for transcriptional control by an activator and inhibitor. All three formulas shown at the top are derived in the Appendix text; the individual graphs, which plot normalized values of [A] and [I] from 0.0 to 2.0, represent each of the formulas with different choices of ν_{Ax} and ν_{Ix}, as shown to the left of each row. In all graphs $\kappa_{Ax}=0.5$, $\kappa_{Ix}=0.8$. For (**A–I**) $\nu_{Ax}=3$, representing modest cooperativity for the activator; **A–C** impose a non-cooperative inhibitor; **D–F**, a moderately cooperative inhibitor; **G–I**, a highly cooperative inhibitor; and **J–L** combine a non-cooperative activator with a moderately cooperative inhibitor. **A**, **D**, **G**, and **J** show variations of the formula derived in Eq. (A27), but without the α's. These parameters are left out; α_A is redundant if there is only a single activator, and α_I merely prevents the inhibitor from fully squelching the activation term so we do not usually use it. **B**, **E**, **H**, and **K** exhibit an alternate generalization of Eq. (A16), whereas **C**, **F**, **I**, and **L** exhibit the formula in Eq. (A17). The main conclusion we draw from this comparison is that although the shape of the curve is broadly similar for each formula, different choices dramatically affect how the parameter ranges limit gene action. For example, the formula in the middle column enables the strongest inhibition at high cooperativity, but at low cooperativity it resembles the formula in the right-most column (compare **B** and **C** and compare **K** and **L**). Meanwhile, the formula in the right-most column is overall the ''weakest'' inhibition mechanism.

probability that A makes an active complex AND that active complex activates x. We can write

$$P(\bar{x}) = P(A \to x) = P((A^*)\&\&\ (A^* \to x)) \\ = P(A^*) \cdot P(A^* \to x). \quad \text{(A18)}$$

The last equality above comes from the fact that A^* and $A^* \to x$ are independent conditions, and thus their joint probability is the product of their individual probabilities. This is simply Eq. (A3) in disguise, with $P(\bar{x}) = P(A \to x) = F(A)$, $P(A^*) = \Psi(A)$, and $P(A^* \to x) = \alpha_A$. To see how we can extend this notation to describe new situations, we must look at some examples.

Example 4: Independent activators

Suppose that we want to represent activation of X by two independent activators A_1 and A_2, each of which alone acts like A above (see Fig. A3A). In this case, the probability that x is transcriptionally active is the probability that A_1 activates x OR A_2 activates x:

$$F(A_1, A_2) = P(\bar{x}) = P(A_1 \to x || A_2 \to x). \quad \text{(A19)}$$

Using simple probability relations, we can quickly transform the expression on the right-hand side into something useful;

$$P(A_1 \to x || A_2 \to x) = \\ P(!(!(A_1 \to x)\&\&!(A_2 \to x))) = , \\ 1 - P(!(A_1 \to x)\&\&!(A_2 \to x)) \quad \text{(A20)}$$

where we have used the relations

$$P(A||B) = P(!(!A)\&\&(!B)) \text{ and } P(!A) = 1 - P(A).$$

Since we assume A_1 and A_2 act independently, we can reduce the joint probability on the right hand side to the product of individual probabilites:

$$1 - P(!(A_1 \to x)\&\&!(A_2 \to x)) = \\ 1 - P(!(A_1 \to x)) \cdot P(!(A_2 \to x)) = \\ 1 - (1 - P(A_1 \to x)) \cdot (1 - P(A_2 \to x)). \quad \text{(A21)}$$

Finally, putting Eqs. (A19), (A20), and (A21) together, and replacing the final probabilities in Eq. (A21) from Eq. (A3), we obtain

$$F(A_1, A_2) = 1 - (1 - \alpha_{A_1}\Phi(A_1)) \cdot (1 - \alpha_{A_2}\Phi(A_2)). \quad \text{(A22)}$$

Note that this reduces to Eq. (A3) if either A_1 or A_2 is absent. Note also that we can immediately generalize this expression to any number of independent activators:

$$F(A_1, \ldots, A_N) = 1 - \prod_{i=1->N} \left(1 - \alpha_{A_i}\Phi(A_i)\right). \quad \text{(A23)}$$

This is the virtue of this way of formulating transcriptional control Affectors: the program code can interpret an arbitrary list of regulators, trivially building collections like Eq. (A23) as arrays of a few simple building blocks. This is the basis for "meta-affectors" in the current version of Ingeneue.

Example 5: Inhibiting activation

Suppose now that factor A activates x and factor I inhibits this activation. We have already dealt with the case in which I interferes with formation of the active complex A*. Here we assume that I acts downstream of complex formation. We can express the action of I the logical condition $I \to (A^* \to x)$ meaning ''I prevents A* from activating x''. In this case, the probability that A activates x is the joint probability that A makes an active complex AND that active complex is not inhibited by I AND the uninhibited complex succeeds in activating x:

$$F(A, I) = P(\bar{x}) = P(A \to x) \\ = P(A^*\&\&\ (!I \to (A^* \to x))\&\&\ (A^* \to x)). \quad \text{(A24)}$$

If the individual conditions are independent, the joint probability is the product of their individual probabilities, or

$$F(A, I) = P(A^*) \cdot P(!I \to (A^* \to x)) \cdot P(A^* \to x) \\ = \alpha_A\Phi(A) \cdot (1 - P(I \to (A^* \to x))). \quad \text{(A25)}$$

For generality, we assume that I binds to DNA to form an active complex I*, which inhibits A*'s activation of x with some probability (or efficiency). We can then resolve the inhibition probability exactly as we did above for the activator A:

$$P(I \to (A^* \to x)) = P(I^*) \cdot P(I^* \to (A^* \to x)) = \alpha_I\Phi(I), \quad \text{(A26)}$$

and we can put Eqs. (A25) and (A26) together to obtain

$$F(A, I) = \alpha_A\Phi(A) \cdot (1 - \alpha_I\Phi(I)). \quad \text{(A27)}$$

We can easily extend the same approach to represent a ''global'' inhibitor that interacts with the basal transcription machinery to shut down activation, regardless of activator concentration, by whatever mechanism. To do so, we simply replace the condition $I \to (A^* \to x)$ with the more global condition $I \to (\bar{x})$. In this case, the probability that x is transcriptionally active is simply the probability that I does not inhibit the basal

machinery AND the uninhibited x is active:

$$P(\bar{x}) = P(!I \rightarrow \bar{x})\&\&(\bar{x}_U)) = (1 - \alpha_I \Phi(I)) \cdot P(\bar{x}_U). \quad \text{(A28)}$$

For example, if x is activated by N independent activators, and globally inhibited by I (see Fig. A3B), then $P(\bar{x}_U)$ is given by Eq. (A23) above and we have

$$F(I, A_1, \ldots, A_N) = (1 - \alpha_I \cdot \Phi(I)) \left(1 - \prod_{i=1->N} (1 - \alpha_{A_i} \cdot \Phi(A_i)) \right). \quad \text{(A29)}$$

By this point, it should be apparent to the reader without going through all the tedious probabilities how we could begin to generalize this process. To make an inhibitor target a particular part of a transcription regulatory mechanism, we simply multiply the probability term representing that part by the generic inhibitory term $\Psi(I) = (1 - \alpha_I \Phi(I))$. For example, if factors A_1 and A_2 activate x independently as in Example 4 and we want factor I selectively to inhibit A_1 (see Fig. A3C), we multiply the efficiency α_{A_1} in Eq. (A22) by $\Psi(I)$ to obtain

$$F(I, A_1, A_2) = 1 - (1 - \alpha_{A_1} \cdot \Psi(I) \cdot \Phi(A_1)) \cdot (1 - \alpha_{A_2} \Phi(A_2)). \quad \text{(A30)}$$

There is of course no reason why we could not choose a slightly different mechanism for I, in which $\Psi(I)$ nestles within $\Phi(A_1)$ somehow, as in Eq. (A17).

This trick is not limited to inhibitory influences. For example, we might know that factor B is "required" for A to do its job and that it, too, must bind to x to fulfill this requirement. If B is constantly present, it becomes one of the many shadowy co-conspirators that lurk beneath $A \rightarrow A^*$ or $A^* \rightarrow x$. However, if B is a dynamic part of our network, we want to include that requirement. To do so, we could simply multiply the efficiency term for A, *wherever it appears*, by an activation term for B:

$$\alpha_A \rightarrow \alpha_A \Phi(B). \quad \text{(A31)}$$

The reader will no doubt think of other examples.

SUMMARY

The approach above has emerged from an ongoing effort to encapsulate, in mathematical equations, what physical intuition and empirical facts tell us about various forms of regulation. Given our current state of knowledge, these formulations are necessarily approximate. There are many ways to invent equations that furnish a reasonable representation of any particular form of regulation. Many of these are equivalent in the sense described above, namely, that one can be used to approximate the other with appropriate choices of parameter (see Figs. A4 and A5). We do not claim that the approach described here will produce the "right" equations in every case. But it has a number of virtues that lend themselves to the type of modeling we do. Chief among these is that it provides a general way to "map" biologists' non-mathematical descriptions of network interactions to specific mathematical equations whose parameters make intuitive biological sense and are at least in principle empirically measurable. The conceptual framework in which it does so is general enough to encompass many different proposed mechanisms of transcriptional regulation. It is both modular and hierarchical, allowing one easily to represent ever more complicated types of regulation as combinations of simpler forms. This means that it is also self-consistent in the sense that complicated forms involving multiple regulators reduce to appropriate simpler forms when one or more of the regulators is absent. Together, these properties mean that a biologist using Ingenue could (in the future) construct the types of diagrams shown in Figs. A1–A3 to represent a specific type of regulation mechanism, and Ingenue could then convert these diagrams into the appropriate Affector formulae.

APPENDIX B: EXPLOITING THE STRUCTURE OF ODES CHARACTERIZING GENE NETWORK DYNAMICS TO DESIGN A FAST NUMERICAL SOLVER

Ingeneue implements a stereotyped formulation for mathematizing maps of epigenetic networks, as described here and in the Supplement to von Dassow et al. ('00). Applying this formulation to any real case typically yields a large system of many non-linear ordinary differential equations (ODEs), which we need to solve numerically for hundreds of thousands, if not hundreds of millions, of different parameter sets. Each parameter set requires the numerical solver to cross a large time interval, which, depending both on the numerical algorithm used and on the stiffness of the equations with that parameter set, may be broken into hundreds, thousands, or even hundreds of thousands of individual time steps. Even

if we knew measured values of all the free parameters in some model, we would still wish to explore the sensitivity to parameters, initial conditions, and structure. Thus the time needed to solve the system of equations that constitutes the model is the primary determinant of how thoroughly we can explore a particular model. If we could shorten this procedure without sacrificing accuracy, we would significantly improve our ability to model complex networks. However, we are constrained in two important, inviolable ways. First, we are unwilling to simplify further the formulation for the sake of computational tractability. Second, we are committed to developing tools that rely on standardized, transparent formulas so as to make it as easy as reasonably possible for users of our tools to extend them without expecting those users to acquire (and get reliably good at) sophisticated mathematical procedures.

This appendix is addressed to biologists who use methods like ours to model epigenetic phenomena and to those who have found themselves entangled in problems that require them to solve ODEs with a similar structure. Many biologists' only familiarity with numerical ODE solvers comes from the canned routines provided by generic mathematical software. Those routines, while excellent general-purpose tools, may not allow customization for a particular application, as we have begun to do with Ingeneue. Thus we first outline the basics of solving ODE systems, then describe our experience with two commonplace methods, and finally describe two techniques which exploit specific features of our modeling framework to accomplish a great improvement in the speed at which we can solve the models' differential equations. We fully expect these methods to be of wider use than for our own problems.

Brief introduction to solving ODE initial-value problems

This section is intended for readers who are unfamiliar with numerical ODE solution methods; those more familiar with these methods should skip most of it and go on to the next section. Our numerical solution task in the abstract is to integrate this autonomous ODE initial-value problem:

$$\frac{dy}{dt} = f(y),\ y(0) = y_{t=0},\ y(t) = \{y_1(t), y_2(t), \ldots y_i(t) \ldots y_n(t)\},\ 0 \le t \le \mathrm{T}. \tag{B1}$$

In words, y is a vector and f is a vector function that specifies how y changes over time as a function of y itself. We want to know, given the formula for f and some initial position y_0, what unfolds as time proceeds.[8] Depending on the problem, we may be interested in a specific time frame, or on limiting behavior; we may be interested in just the quantities y at some time t, or we may want to know how it got there as well, and so forth. In the case of a gene network model y_1, y_2, y_3,... are the concentrations of individual molecular species in the model, indexed by cell or cell face, and f_1, f_2, f_3,... represent the kinetic formulae for each molecular species. Each has the format

$$\frac{dy_i}{dt} = \text{synthesis} - \text{decay} \pm \text{transformations} \pm \text{fluxes} = f_i. \tag{B2}$$

These four terms correspond to the four classes of Affectors shown in Table 1. The simplest way to get from one timepoint to another, given the derivative formula f and the value of y at the initial point, is according to the forward Euler formula:

$$y_{t+h} = y_t + hf(y_t). \tag{B3}$$

The Euler formula is the prototype for all numerical strategies for solving differential equations. Of course the Euler formula is only accurate to the extent that the derivative does not change across the interval h. So one solves a system of differential equations across a large interval by taking a series of time steps of size h across the interval; consider

$$y_{t=nh} = y_{t=0} + h\sum_{i=0}^{n-1} f(y_{ih}), \tag{B4}$$

where nh is the interval we want to cross; as h gets infinitesimal Eq. (B4) above approximates better and better the integral

$$y_{nh} = y_0 + \int_0^{nh} f(y)dt. \tag{B5}$$

There are not many problems for which Euler's method approximates this formula for h large enough to be efficient, because the derivative usually *does* change as we cross h. At the least, too great a time step results in ''hopping'' onto trajectories further and further removed from the

[8]The time, t, does not usually appear explicitly in f, but if it did y would include t as another state variable whose derivative formula appears in f as the constant 1.0.

''true'' one. Worse, if the system of ODEs is stiff, with steps too large, wild oscillations may dominate the time course of the solution. Of course the extent to which these problems arise depends on the system in question; even Euler's method is exact for a certain (uninteresting) class of equations. Every numerical analysis text (see Burden et al., '78; Johnson and Reiss, '82) discusses the following improvements to Euler's method, none of which one would likely choose for the kind of problem we discuss here, but which easily convey the gist of the problem and its solution:

$$y_{t+h} = y_{t-h} + hf(y_t), \tag{B6}$$

$$y_{t+h} = y_t + hf(y_{t+h}), \tag{B7}$$

$$y_{t+h} = y_t + hf(y_t + \tfrac{1}{2}hf(y_t)), \tag{B8}$$

$$y_{t+h} = y_t + \tfrac{1}{2}h(f(y_t) + f(y_t + hf(y_t))). \tag{B9}$$

The first is the midpoint method; the second is the backward Euler formula; the third is the ''modified Euler'' method; the last is Heun's method. Note that the backward Euler formula involves evaluating the value of f at y_{t+h}, although we do not yet know y_{t+h}; thus it is an implicit formula that requires solving a non-linear algebraic equation system starting with some initial estimate of y_{t+h}. Also, Heun's method averages the forward Euler formula with an estimate of the backward Euler formula, the latter computed using the result of the forward formula as the estimate of y_{t+h}. Heun's method, however, is explicit.

Each of these improvements to the prototype tries to accomplish more or less the same thing: to account for the fact that the derivative changes as we cross h, or, in other words, to account for the *second* and higher derivatives of y. This reminds us that the Euler method corresponds to only the first two terms of the Taylor series expansion around the initial point y_t:

$$y_{t+h} = y_t + hy'_t + \frac{h^2}{2}y''_t + \cdots + \frac{h^n}{n!}y_t^{(n)} + \cdots. \tag{B10}$$

In order to use more than just the first two terms one would have to know not just a formula for the first derivative of y with respect to time (f, that is) but also the second partials with respect to each component of y (the Jacobian matrix). This is not suitable for a general ODE cookbook, especially as the Jacobian may be very difficult to compute if f is even moderately complicated. However, the stereotyping of Ingeneue's formulation means that, since the formulas are all fairly straightforward, we could, for once and for all, compute all the second partials. We have not pursued this approach because we place a high premium on the extensibility of our methods to accommodate new cases. Requiring users of the method to write formulas for the second partials accurately not only would introduce an unsavory chore but would also introduce an error-prone step sure to reduce the reliability of the method.

Instead, the most straightforward strategy is to use numerical methods that approximate the higher-order terms of the Taylor series by generalizations of the modified Euler, midpoint, or Heun methods. So-called Runge–Kutta methods do just that. These methods are wonderfully elegant; in effect, they ''feel'' the local flow field around the initial point for each step and then use that information to build up a complete step by linear combination of the local estimates. The standard textbook fare is the fourth-order Runge–Kutta method:

$$\begin{aligned} y_{t+h} &= y_t + \frac{h}{6}(m_1 + 2m_2 + 2m_3 + m_4), \\ m_1 &= f(y_t), \\ m_2 &= f(y_t + \tfrac{1}{2}hm_1), \\ m_3 &= f(y_t + \tfrac{1}{2}hm_2), \\ m_4 &= f(y_t + hm_3). \end{aligned} \tag{B11}$$

This explicit recipe is trivially easy to program, takes as its starting information only y at t, and costs four derivative evaluations per step, while making an error proportional to the fourth power of h. In practice Eq. (B11) by itself is not suitable for most purposes because there is no mechanism expressed for truncation error estimation and adjustment of the timestep h to keep truncation error below some tolerance the user specifies. However, there are straightforward strategies to ''embed'' a lower-order formula in a higher-order one (say a fourth-order estimate in a fifth-order one), so that a simple change in coefficients leads to an estimate of the local truncation error. Such methods allow one to automate changing h (at every step if necessary) according to the magnitude of the local error, resulting in much better efficiency for problems in which the scale of the curvature in the solution changes over the course of the trajectory toward the stopping point. These methods provide both convenience and accuracy, and the efficiency to be expected from what

Numerical Recipes in C calls a ''workhorse'' (Press, '92). Accordingly we have relied heavily on a particular embedded scheme due to Cash and Karp ('90). Our software Ingeneue includes a re-implementation of the Cash–Karp method, in a slightly simpler form than originally suggested by Cash and Karp themselves, following code given in *Numerical Recipes in C*. Indeed, as described below, this scheme proved tough to beat for our particular problems.

The only drawback to the Cash–Karp scheme (and other embedded Runge–Kutta methods of order greater than 4) is that six or more values of the derivative function must be computed at each step. This does not afflict another large class of methods, called ''multi-step methods,'' because they use the history of prior integration steps to extrapolate estimates for the next value of y. Of course this means one has to accumulate a little history before one can use a multi-step formula. Also, the easy-to-program multi-step methods depend on having the history data evenly spaced; thus one cannot easily change h at each step without the cumbersome burden of reworking the history list, either by interpolation or by directly recomputing it (i.e., by using a one-step method to execute a few steps). The potential payoff is that each step with a multi-step method may require only two or three evaluations of f per step. Thus *if a multi-step method can take steps at least half as large as the steps taken by a Runge–Kutta method of equivalent order* it is perhaps to be preferred. As we discovered from experience, that ''if'' clause can become quite important.

The typical use of multi-step formulas requires a pair, one of which is explicit and the other of which is implicit. The explicit formula is used as a ''predictor'' of y_{t+h}, and then this predicted value is used as an initial estimate on the right-hand-side of the implicit "corrector" formula. The fourth-order Adams–Bashforth–Moulton method looks like

$$\begin{aligned} y_{p,t+h} &= y_t + \frac{h}{24}(-9f_{t-3h} + 37f_{t-2h} - 59f_{t-h} + 55f_t), \\ y_{c,t+h} &= y_t + \frac{h}{24}(f_{t-2h} - 5f_{t-h} + 19f_t + 9f_{t+h}), \\ &\quad \text{where } f_{t+h} = f(y_{t+h}). \end{aligned} \tag{B12}$$

Although the predictor formula requires only one evaluation of the derivative, the corrector is an implicit formula that must be solved iteratively for the unknown y_{t+h}; each estimate of $y_{c,t+h}$ is substituted to calculate f for another pass:

$$\begin{aligned} y^{k+1}_{c,t+h} &= y_t + \frac{h}{24}\left(f_{t-2h} - 5f_{t-h} + 19f_t + 9f(y^k_{c,t+h})\right), \\ y^1_{c,t+h} &= y_{p,t+h}; \quad \text{iterate until } y^k_{c,t+h} - y^{k-1}_{c,t+h} < \varepsilon. \end{aligned} \tag{B13}$$

The textbook advice goes that as long as this iteration process is converging, more than two iterations of the corrector formula are a waste of derivative calculations, and one should take a smaller time step instead (Burden et al., '78; Johnson and Reiss, '82; Press, '92). The difference between successive results is used to judge convergence, and the difference between the predictor's estimate and the converged corrector's estimate is used to judge whether the accuracy is adequate. Many implementations of this recipe will accept early corrector estimates in the event that the corrector iteration diverges but some value along the way is within the error tolerance of the predictor estimate. This may save some function calls if we are not truly at risk from instability, but if we are, then this strategy can result (as we discovered from experience) in a local bubble of instability that ends up demanding smaller time steps than if we had done the extra work to achieve convergence.

Experience with the Ingeneue framework, and a way to exploit the stereotype

We coded and tested the fourth-order Adams–Bashforth–Moulton predictor–corrector (APC) method using the core segment polarity network model as a test problem and compared its performance relative to the Cash–Karp scheme that we used to obtain all the results reported here and in von Dassow et al. (2000). We were disappointed to find that, no matter how we manipulated the parameters governing the integrator heuristics (e.g., the maximum number of iterates allowed per step), there was no way to make this method faster than the Cash–Karp scheme. We realized why when we tried to relax the requested error tolerance completely (i.e., allowing absolute errors of the same order as the maximum values of the state variables) and found that even then the APC method could not take fewer time steps than when the required accuracy was more stringent. In fact, this proved more or less true of the Cash–Karp method as well. These findings indicated that stability of the numerical solution, not accuracy, is the main constraint on the time step used in solving gene network models.

While the Cash–Karp method fails to achieve accuracy if the time step is too large, even when the error tolerance is relaxed, the APC method fails to achieve convergence on the corrector formula. The reason has to do with the nature of our equations and the resulting terms in the Jacobian of f. The corrector formula will converge if

$$\frac{9h}{24}\left\|\frac{\partial f}{\partial y}\right\| < 1. \tag{B14}$$

We have a case in which, depending on the parameters of the model, this inequality may hold only for a time step much smaller than could be taken by some type of Runge–Kutta method that does not involve fixed-point iteration. The primary reason for this is sigmoid dose–response curves. If the half-maximal activity coefficient is small (e.g., 10^{-2}) then the partial derivative of f with respect to the regulator will be large. That is, there will be a large change in the value of the sigmoid term for a correspondingly small change in the regulator concentration. This will be exacerbated with larger values of the cooperativity coefficient.

When faced with the considerations above, numerical analysts might advise turning to more sophisticated methods like the Gear formulas, which employ Newton's method to solve the corrector formula instead of fixed-point iteration. However, Newton iteration requires that one know the Jacobian of f, so we are back to the problem that if we are to preserve extensibility then we can not expect users of our methods to derive reliable formulae for all the second partials and write the code to sort them out properly. We could estimate the Jacobian numerically, but this, too, is an error-prone process. Finally, it is not at all clear that Gear's method or any other stiff solver would improve matters, because our equations are only mildly stiff, and the burden of computing and inverting the Jacobian may vastly outweigh any benefit gained from taking a larger step size.

It should be kept in mind that in our application we are obliged to vary essentially all the parameters of the model over very wide ranges. Given certain parameter sets, straightforward integration methods like the Cash–Karp scheme allow us to take giant leaps along the trajectory. The trouble is that with another choice of parameters the same method might bog down with tiny time steps. The Cash–Karp method, for instance, is not fond of second-order reactions (hetero-dimerizations) with high rate constants. We could, and may in the future, try to develop run-time heuristics that switch among different methods according to how difficult the model is expected to be with the present parameter choices, attempting to choose the best performer for each case. The drawback is that this approach, again, would likely require users to become skilled in an onerous task to ''train'' those heuristics with each model.

However, we are free of generalist considerations; we do not need a solver that works well for any foreseeable system, we need one that solves our problem with the greatest efficiency. Thus we sought ways to exploit the specific nature of our modeling framework to see if we could improve the performance of simple solvers like the ABM method. In our problem it turns out that each and every equation follows a stereotypical, generic formula that, within the context of the choices we made about how to model molecular interactions, encompasses almost every imaginable kind of intermolecular reaction. The generic formula (in dimensionless form) can be written as

$$\frac{dy_i}{d\tau} = F(\bar{y}_{j\neq i}) - G(\bar{y}_{j\neq i})y_i - Hy_i + Ky_n. \tag{B15}$$

This looks rather similar to Eq. (B2), but in fact there is not a one-to-one map between the terms. Equation (B2) groups terms according to the kind of reaction; Eq. (B15) groups terms according to how they depend on the components of y, the vector of concentrations of individual molecular species, indexed by cell or cell face. y_i is thus an individual species in a particular compartment. y_n represents the same species in adjacent compartments (e.g., a neighboring cell).

As far as we have been able to foresee, the four types of terms in Eq. (B15) cover every kind of reaction that would happen to a molecule *and that we would be disposed to model as an elementary process*, with one exception that will be mentioned below. $F(\,)$ represents non-linear terms that are functions of other species' concentrations, but not of the species governed by that term. Such terms include transcription rate as a function of the concentration of regulator species. $G(\,)$ represents terms in which a non-linear function of other species multiplies the concentration of the species governed by the term. Examples include regulated cleavage and heterodimerization; these terms always subtract from the concentration of the species in question because they are converting it to some other form. The term involving the constant H represents first-order reactions that

remove the species in question; decay, flux to neighboring compartments, dissociation, or unregulated processes like constitutive conversion to another form (e.g., dephosphorylation). Again, such terms are always subtracted. Finally, the constant K represents the reverse of H, and typically includes flux from another compartment. Strictly speaking, there is no reason K is not encompassed by $F(\,)$, but we break it out because of the conceptual distinction.

We noticed that *every term involving y_i is linear in y_i*. (Again, there is an exception to be mentioned below, and there would be no such opportunity if we collapsed mRNAs and proteins into a single molecular species, as do many modelers.) Using the backward Euler formula for simplicity of illustration, this special structure allows the following rearrangement:

$$y_{i,t+h} = y_{i,t} + hf(y_{t+h}) = y_{i,t} + h\big(F(y_{t+h,j\neq i}) - G(y_{t+h,j\neq i})y_{i,t+h} - Hy_{i,t+h} + Ky_{j\neq i,t+h}\big) \quad \text{(B16)}$$

becomes

$$y_{i,t+h} = \frac{y_{i,t} + h\big(F(y_{t+h,j\neq i}) + Ky_{j\neq i,t+h}\big)}{1 + h\big(G(y_{t+h,j\neq i}) + H\big)}. \quad \text{(B17)}$$

For lack of a better term we call this a ''semi-explicit'' formula because it is explicit for $y_{i,t+h}$ but in terms of unknown $y_{j\neq i,t+h}$. Here, the coefficient in front of the norm of the Jacobian in Eq. (B14) will be smaller in proportion to the size of $G(\,)+H$. These can be very large, and in fact since these are (depending on the parameter set) often the largest terms in the derivative, it is these terms which make the equations hard for the Cash–Karp method to solve. Here, instead, they will *help, rather than hinder,* us to get convergence in a corrector iteration. The equations may still be stiff, but we should now be able to outrun the adaptive-stepsize Runge–Kutta method *when the parameters are such as to render the equations stiff*.

Practical tests using various versions of the segment polarity model show that an APC implementation using the fourth-order equivalent of Eq. (B17) (which we refer to as SEAPC for ''Semi-Explicit Adams Predictor–Corrector'') does indeed outrun the Cash–Karp method. Our standard practice has been to impose an absolute error tolerance of 10^{-4} (on dependent variables which, rendered dimensionless, range from 0.0 to 1.0). For the simple segment polarity model (''spg1'') discussed here and in von Dassow et al. ('00), SEAPC solves a suite of test parameter sets in approximately 70% of the time required by the Cash–Karp method. However, for a more complex model the difference is greater; for the model ''spg4'' described in the companion paper by von Dassow and Odell ('02, this issue), the new method is at least 3-fold faster than the Cash–Karp method. A similar speed-up was obtained with a fairly complex model of the neurogenic network (Meir et al., unpublished observations). We believe, but have not rigorously verified, that the difference in the speed-up achieved has to do with the relative proportion of $F(\,)$ and K versus $G(\,)$ and H terms in the model. Both spg4 and the neurogenic network model involve more heterodimerization reactions than spg1.

It occurred to us that if stability were the primary issue, then we ought to be able to achieve similar or greater gains using the backward Euler formula, rearranged as in Eq. (B17), instead of the fourth-order SEAPC method. With the error tolerance set to 10^{-4} the semi-explicit backward Euler (SEBE for ''Semi-Explicit Backward Euler'') method is a poor competitor. However, with error tolerance of 10^{-2} or 10^{-1}, we achieve a 10- or 20-fold speed-up, respectively, on spg4. Of course in practice one would be uncomfortable using such lenient error tolerances; however, recall that in our application we are typically looking for attractors, and must search iteratively through hundreds of thousands of parameter sets. Thus, we can use a lenient error tolerance and the SEBE method to screen the random samples, then test any ''good'' sets found by the low-accuracy method with a high-accuracy SEAPC or Cash–Karp method. We have verified that the low-accuracy SEBE method has the following properties:

(1) Any parameter set found using SEBE would have been found by the Cash–Karp method too;

(2) SEBE correctly solves the vast majority of parameter sets found by the Cash–Karp method;

(3) When SEBE is used, the time required to solve the model is far less sensitive to the choice of parameters than the Cash–Karp method.

We cannot overstate the potential impact of this discovery on our work; this means that we can now accomplish in a day a search that would have required several weeks of computer time using the Cash–Karp method, without sacrificing much accuracy. In dollar terms, it means we can do with a $10,000 computer network what would have required $200,000 worth of computers absent this invention. These comparisons apply to using a search strategy as described above, in which a

rough cut with SEBE is used to ''filter'' the primary samples and the few that get through are verified with SEAPC or Cash–Karp. Our software, Ingeneue, incorporates the fourth-order APC, Cash–Karp, SEBE, and SEAPC methods, and it is an easy matter of inserting in an input script a flag instructing the program which integrator to use.

The one exception we have encountered is cases in which molecular species form homo-dimers. Then there is a term equivalent to $G(\,)$ in Eqs. (B15–B17) in which the derivative of y_i depends non-linearly on y_i itself (the term is a second-order rate constant times the square of y_i). One can still conduct the rearrangement shown from Eqs. (B16) and (B17), but no longer is Eq. (B17) explicit in y_i. The algebra still comes out, and Eq. (B17) still converges better than Eq. (B16). As far as we can tell, the rearrangement can only help (as long as the terms moving around are, first of all, divisible by y_i, and second, negative-valued in the original form). In fact, one could choose to leave terms non-linear in y_i alone. This is good: to be used in this method, Affectors, if they depend on y_i, must have coded into them a function that returns $G(\,)$ or H. Affectors that supply this function set a flag telling the integrator that they may be rearranged. Should the user, when coding a new Affector, neglect or prefer not to supply such a function, the integrator simply leaves the Affector on the right-hand side as is. This does no violence to the SEBE or SEAPC methods, but it may render them not quite as efficient as they could be.

Many applications may benefit from a similar approach to solving large ODE systems. Any system, (1) the component equations of which break down into additive terms, (2) in which there are negative-valued terms involving products of the dependent variable (even if there are other terms involving the dependent variable), (3) in which those terms are potentially large compared to the other terms, and (4) for which the problem involves screening vast numbers of integration runs, can probably benefit from a similar scheme. Possibilities include simulations of chemical reaction systems, population ecology models, non-linear neural network-type models, and probably others as well.