Available online at www.sciencedirect.com
ScienceDirect
Developmental Biology 310 (2007) 187–195
DEVELOPMENTAL BIOLOGY
www.elsevier.com/developmentalbiology

Review

The regulatory genome and the computer

Sorin Istrail [a], Smadar Ben-Tabou De-Leon [b], Eric H. Davidson [b,*]

[a] *Center for Computational Molecular Biology and Department of Computer Science, Brown University, 115 Waterman Street, Box 1910, Providence, RI 02912, USA*
[b] *Division of Biology 156-29, California Institute of Technology, Pasadena, CA 91125, USA*

Received for publication 10 May 2007; revised 31 July 2007; accepted 4 August 2007
Available online 10 August 2007

Abstract

The definitive feature of the many thousand *cis*-regulatory control modules in an animal genome is their information processing capability. These modules are "wired" together in large networks that control major processes such as development; they constitute "genomic computers." Each control module receives multiple inputs in the form of the incident transcription factors which bind to them. The functions they execute upon these inputs can be reduced to basic AND, OR and NOT logic functions, which are also the unit logic functions of electronic computers. Here we consider the operating principles of the genomic computer, the product of evolution, in comparison to those of electronic computers. For example, in the genomic computer intra-machine communication occurs by means of diffusion (of transcription factors), while in electronic computers it occurs by electron transit along pre-organized wires. There follow fundamental differences in design principle in respect to the meaning of time, speed, multiplicity of processors, memory, robustness of computation and hardware and software. The genomic computer controls spatial gene expression in the development of the body plan, and its appearance in remote evolutionary time must be considered to have been a founding requirement for animal grade life.

Keywords: Regulatory information processing; Gene regulatory networks; Biological logic computations

Introduction

The genomic regulatory code that controls animal development is resident in the sequences of the several hundred thousand *cis*-regulatory modules (CRMs) that the genomes of animals such as us contain. Regulatory functions of the control modules are genetically hardwired since their target site sequences determine the identities of the particular transcription factor inputs to which they respond. In development the most important *cis*-regulatory modules are those controlling (that is, turning on or off) expression of genes encoding transcription factors since it is these molecules which cause or prevent expression of all genes, i.e., other regulatory genes, signaling genes, differentiation genes and morphogenesis genes. Individual transcription factors are expressed in cell type specific, and temporally specific ways, and the ensemble of transcription factors that defines the regulatory state of each cell type differs from that of any other cell type and specifies its functional possibilities (1).

The essential functional character of developmentally active CRMs is that they execute information processing of their multiple inputs. That is, they generate new functional outputs by combining these inputs to produce switches, gates or quantitative activity controllers. Their regulatory outputs, the "instructions" they transmit to the basal transcription apparatus of the gene they control, are always the result of processing their several inputs so that it is usually impossible to predict from any one input how the regulatory system will behave. However, the actual instantaneous regulatory function of a CRM is not a constant of its structure: it depends on the current regulatory states to which it is exposed at each given time and place in the developmental history of the organism. Thus CRMs are the conditional information processing elements of the overall genomic computational system.

Information processing in development occurs at the next level of regulatory organization as well, that is, at the level of the networks of CRMs and regulatory genes that do the biological

* Corresponding author. Fax: +1 626 793 3047.
E-mail address: davidson@caltech.edu (E.H. Davidson).

0012-1606/$ - see front matter © 2007 Elsevier Inc. All rights reserved.
doi:10.1016/j.ydbio.2007.08.009

"jobs" of development, such as specification of territorial fate, interpretation of signals, installation of self-limiting or periodic functions, and so forth. Development is intrinsically a process of change in regulatory state in space and time, every step of which devolves from and involves regulatory information processing; the very condition for the existence of development is the genomically encoded information processing system.

The comprehensive analysis of the logic functions of *cis*-regulatory modules on the one hand (see for example, Yuh et al., 2001, 2004, 2005) and of the topology of gene regulatory networks on the other hand (Davidson, 2006) brings us to the era where we can examine the intrinsic nature of the genomic information processing system. Gene regulatory networks display the exact interactions among regulatory and signaling genes which are responsible for establishing the regulatory state in each cell type, at the developmental times to which the networks pertain. Thus they provide us with exact pathways of causality between the A's, C's, G's and T's of the static regulatory genome and the generation of novel regulatory states as development progresses. These pathways all include *cis*-regulatory and network information processing, which can be considered as computational logic operations. The genomic computer has been assembled step-wise during evolution and in any given modern animal is a complex of older and more recently altered regulatory structures (Davidson and Erwin, 2006). It is illuminating to consider the biological computational machine in the terms usually applied to manmade computation devices. Here we first examine briefly the nature of some basic information processing functions that both our computers and the genomic computer carry out and then compare diverse aspects of the means of computation.

Fundamentals of information processing

We begin with elementary logic functions which CRMs execute and which computers also execute. A CRM can be decomposed into elementary subcomponents each of which carries out some specific basic function, and many *cis*-regulatory functions can be reduced to Boolean truth tables (Istrail and Davidson, 2005). A CRM is thus a conditional information processing device that can be compared to a typical computer circuit. Among the elementary CRM subcomponents are those which execute the basic logic functions. A common and basic example is the AND function. If a part of a CRM has sites for two transcription factors, and its output is conditional on both being present, then this part of the CRM in fact operates as an "AND" gate: it performs the logical operation of "conjunction" since it responds only when both of its inputs are active. This condition might obtain where two non-coincident spatial domains of transcription factor expression overlap or when a temporal and a spatial domains overlap (for many examples see Davidson, 2006; Istrail and Davidson, 2005). For instance several *Drosophila* CRM that respond both to anterior/posterior stripes of gene expression and to dorsal/ventral transcriptional cues and operate where they intersect are of this class (op. cit.). The consequence is definition of new spatial subdomains. If on the other hand a part of a CRM requires for activity that at least one of two inputs be present, then it acts as an "OR" gate; that is, it performs the logical operation of "disjunction" since it responds to some extent when either of its two inputs is present. In development many OR gates function additively in that the output is the sum of the two inputs, but either one suffices for some activity. These gates are very commonly seen in differentiation genes where diverse transcriptional drivers each contribute to the output level of expression, as for instance in muscle contractile genes or pancreatic endocrine genes (op. cit.). AND and OR are two of the basic operations of mathematical logic. The third basic logic operation is the negation, "NOT," which acts as a switch: when the repressor is present and the input is therefore "on," it outputs "off," transcriptional silence, and vice versa: when the input is "off" it outputs "on." This formalization applies to CRM functions that include active transcriptional repressors as inputs. These three logic operations are universal: all other logical operations, viewed as mathematical functions, can be obtained by suitable combinations of AND, OR and NOT, a principle adapted to computational devices by von Neumann (1958).

Fig. 1 describes several elemental logic computations in the different languages of electronics, logics, *cis*-regulatory function and biochemical assembly. These four representations of the AND logic operation are given in Fig. 1, Part 1. The electronic diagram is represented on the left, i.e., the customary diagram of an AND gate in electronic computers. A Boolean truth table of the AND function is shown next to it. It has two inputs (*I*) represented in the first two columns, and the third is the output column; each row shows the output value for the corresponding values of the inputs. A CRM diagram of such an AND logic processor is then presented: it has two transcription factor inputs binding to the CRM, and the output is active only when both inputs are present; otherwise the output is just "background." A biochemical way of looking at the same processor is shown in the diagrams on the right. First is a "device" part, which is a physical representation of how such an AND operator might work in a CRM. In this particular case the biochemical reason for this mode of function is that the cofactor which actually activates the basal transcription apparatus requires both DNA-binding transcription factors to be present in order for it to bind in the CRM. Of course, the same input–output behavior might be implemented with various different biochemistries. Possible outputs of a *cis*-regulatory AND processor are shown in two ways: as a spatial specification of expression only where two non-coincident inputs overlap (as above); second, as a quantitatively measured output, as would be seen in a graphical representation of expression measurements in which removal of either input decreases output to background (for a real life example, see Yuh et al., 2002).

In Part 2 of Fig. 1 we present four equivalent representations of an additive OR logic operation. Electronic and mathematical logic diagrams are given on the left followed by the CRM diagram for such an OR logic function, indicating two transcription factors binding to the CRM. There is an output when either of these inputs is present (valued at "1" in the table), additive when both are present (valued at "2" in the table). The

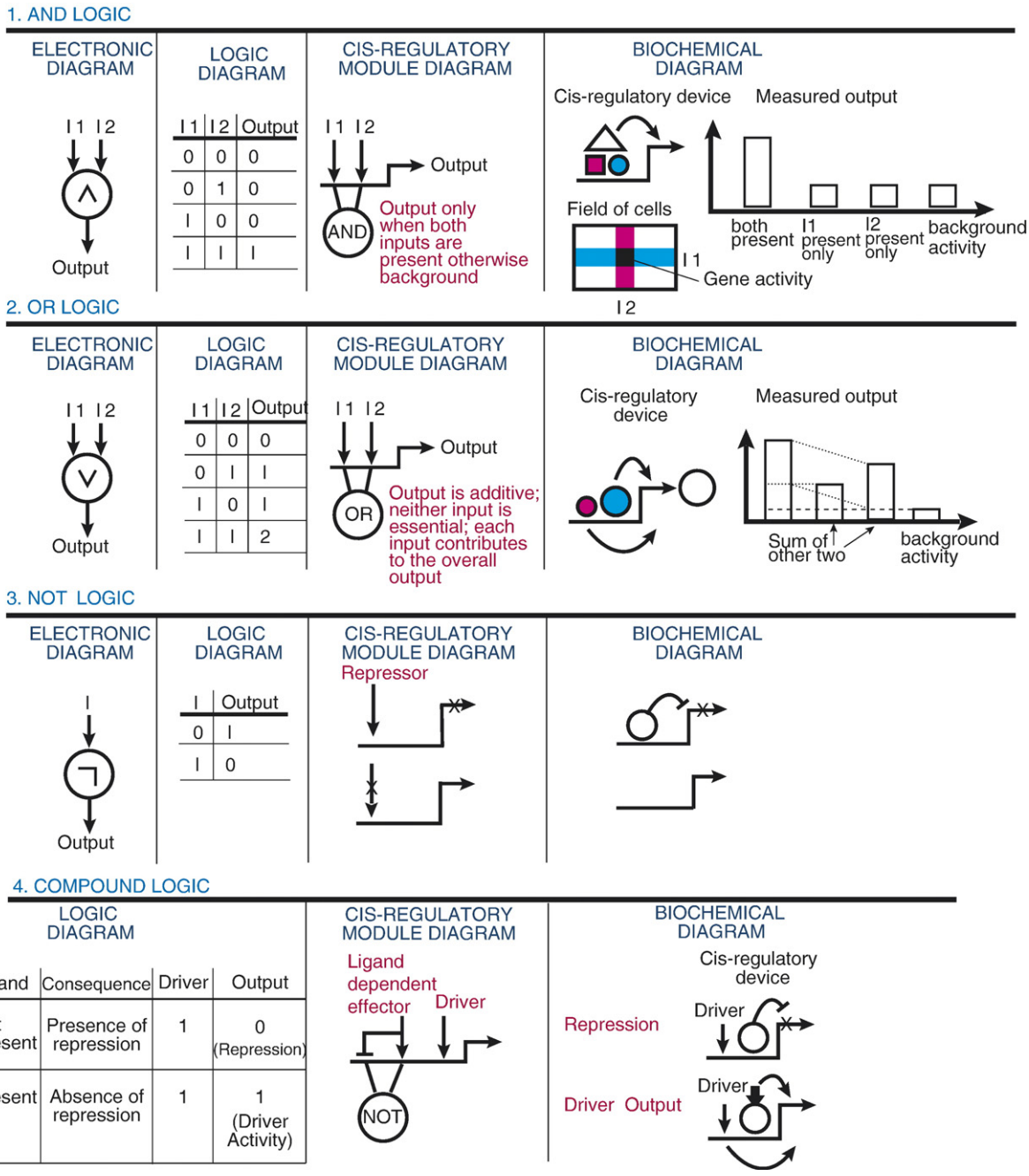

Fig. 1. Basic logic operations executed by computers and *cis*-regulatory modules. Part 1, AND logic; Part 2, OR logic; Part 3, NOT logic; Part 4, compound logic. See text for explanation and discussion.

"device" part of the diagram at the right provides a physical representation of how such an OR device might work in a CRM: here, as symbolized by the curved arrows, one of the factors stimulates transcription by activating an element of the basal transcription apparatus, the other by affecting chromatin state (e.g., see Stathopoulos and Levine, 2002). The "measured output" might appear as shown on the right; compare the AND processor in Part 1.

In Part 3 of Fig. 1 are equivalent representations of a NOT logic operation, the customary diagram for NOT processors in electronic computers on the left, followed by the mathematical logic table of the NOT function. In gene regulation this is the function of transcriptional repressors, which act as NOT operators: when repressors are present, output is absent; when repressors are absent output is not absent. In Part 4 is shown a CRM function which is a combination of a NOT processor and an activating input; as shown in an earlier study (Istrail and Davidson, 2005), CRMs in general execute compound operations, which are composed of multiple logic operations. The case considered in Fig. 1, Part 4 is commonly observed in signal-mediated transcriptional control. In a cell receiving the signal an "effector" transcription factor transduces the signal and acts as a gate, allowing a positively acting transcription factor binding in the same CRM (driver) to turn on the basic

transcriptional apparatus. But in a cell not receiving the signal, the same effector is an obligate, dominant repressor. The signal response system thus acts as a toggle switch (Barolo and Posakony, 2002; Istrail and Davidson, 2005). On the left we deconstruct this type of response element and show that it includes a NOT logic operator (black box).

This view of CRM as a combination of multiple logical gates defines what we mean in stating that CRMs function as information processing devices. Gene regulatory networks consist of assemblies of CRMs and the regulatory and other genes they control. Since such networks are assemblies of primary information processing devices, their operation can be compared to that of manmade computing machines. In principle, the genomic computers that control development could carry out all computations because they execute combinations of AND, OR and NOT computations.

Properties of the genomic computational system

Considering the genome as a whole and the complete life cycle, the developmental information processing complexity of the genomic computer can be crudely approximated as follows. We take the sum total of the regulatory linkages in all the gene networks which control cell function, integrating over the whole life cycle and all cell types which arise therein, multiplied by the number of cells per type. For each cell type there are at least 10^4 genes that must be regulated. If there are 10^{12} cells in a human, there are 10^{16} regulatory performances required on *cis*-regulatory modules that on the average each may utilize 4–8 diverse inputs. It is interesting to consider the design principles of this enormous device, with reference to those of the large computers that we ourselves build.

Diffusion vs. wires: the electronic computer and the regulatory genome

In the genomic computer processors of any given gene regulatory networks are linked by diffusion. Here is one of the most profound differences between the electronic computer and the genomic regulatory system. Compared to the one way, directed transfer of information in metal wires, diffusion of transcription factors is slow and probabilistic, and its rate is dependent on concentration, temperature, shape and mass. A typical rate constant for transcription factor diffusion is $\sim$100 $\mu m^2/s$, orders of magnitude slower than the rate of transit of electrons in a metallic conductor. Successful transcription factor interactions with their target site sequences depend on stereochemical intercalation into the DNA double helix and formation of chemical bonds between the protein side chains and bases in the target site DNA. But this occurs only after probabilistic tests of many alternative possibilities as the factor diffuses along the DNA molecule, not on pre-wired linkages. In computer science the least efficient, “brute force” programs are those that do all possible combinations and ex post facto select what gives the desired answer. But in that diffusion is a random walk process, it has exactly this property. Nonetheless there are no systems more fail-safe than the developmental systems that operate by means of diffusion: the “safety net” in animal cell nuclei devolves from the large number of transcription factor molecules of each molecular species in a relatively small volume, and from virtual insensitivity to the exact time it takes for any one of these molecules to dock at its target *cis*-regulatory sequence.

Time and synchrony: the electronic computer and the regulatory genome

The basic design principle of electronic computers is the synchronization of every elementary operation to a single pulsing clock. This clock operates at very high rates, and the efficacy of the computer depends on its precision. This principle is entirely irrelevant to the genomic information processing system. The genomic computer has no rigid temporal synchrony, in that it consists of a large number of independent *cis*-regulatory information processors, each with its own temporal relations which depend on dynamics of input (change in transcription factor concentration with time) and output (rates of RNA production). The Draconian requirements for exact synchrony in electronic computers, and their failure to operate if deviations occur, are not properties of living regulatory systems, nor, obviously, could they be. The genomic regulatory system is, in contrast, tolerant of temporal variation and local asynchronies. Causal coordination between different networks and network elements replaces imposed temporal synchrony: if given genes turn on, in consequence their target genes presently turn on (or off, if a transcriptional repressor is among the given genes). That is, in each cell at any given time, the current regulatory state and the signaling cues are processed to produce the next regulatory state. A relaxed form of synchrony between the different embryo domains and also between the cells of the same domain is achieved by signaling, again a diffusion mediated process, at both inter- and intra-cellular levels. In marine embryos, for example, the whole process works perfectly well at different rates over a range of ambient temperatures.

The complexity of development requires an enormous number of information processing steps as regulatory states are established in every cell over time and space. In electronic computers the requirement of large computations is met by increasing speed. This is done in many ways: parallel processing, advanced software and faster basic clocks. None of these strategies for increasing computational power by increasing speed have been deployed during evolution in answer to the need for large gene regulatory computations. In contrast, genomic computational systems have evolved that deploy very large numbers of slowly working information processing modules and that arrange them in shifting sets of subcircuits.

Multiplicity of processors: the electronic computer and the regulatory genome

Ordinary desktop electronic computers have one or two processors and ordinary parallel computers may have up to a few

thousand. The basic limitation in the number of independent processors in such computers derives from the difficulties of synchronization in fixed communication architecture. The "mean time failure," the interval between successive failures, decreases with the number of processors (Hennessy and Patterson, 2003). In contrast, the genomic computation apparatus includes at least several hundred thousand small independent processors (i.e., the number of CRMs in a typical animal genome; Davidson, 2006). Perhaps 10^4 of these are directly and indirectly linked together functionally in a given cell at any given time, if we take this as a rough estimate of the number of genes in a typical genome, each of which is regulated either positively or negatively, by CRMs.

The large number of processors operative at any one time in the genomic computer illuminates one of the major payoffs of a temporally tolerant system of intra-machine communication that runs on diffusion, rather than on communication through inflexibly designed wires. Fixed communication architecture limits not only the number of independent processors that can be effectively coordinated and programmed, but also the variety of modes of operation. The changeable architecture and temporal freedom of the genomic regulatory system in principle allow a large and flexible number of processors and a great variety in modes of problem solving. Evolutionary change in the animal body plan occurs by alterations in the gene regulatory networks that control its development. The flexibility of these networks in respect to computational design thus fundamentally underlies the processes leading to evolutionary change in major aspects of morphology.

Distributed or parallel computer clusters utilize asynchronous processors, and they share this design principle with genomic regulatory systems. However, they require a dense and tightly regulated traffic of messages passing between processors along the fixed matrix of channels. In contrast, in genomic regulatory networks the "communication channels" are formulated on an instantaneous need basis as the developmental computation proceeds: if a subcircuit of regulatory genes causes the regional activation of other regulatory genes in given cells, the products of these genes will find their downstream targets in the genome, thus creating a new subcircuit in those cells. The active genomic computer continually redesigns itself during development, within the range of the hard-wired template of potentialities resident in the DNA sequence.

Memory: the electronic computer and the regulatory genome

Electronic computers are equipped with various forms of memory, but most are passive in that information is stored and remains there in the absence of further computation, until it is erased or overwritten. DRAM (dynamic random access memory) does require periodic refreshment of charge, but this is essentially only to preclude gradual loss of information by charge leakage. The genomic regulatory system also utilizes various forms of passive memory. The major passive mechanisms result in installation of chromatin states which "lock down" conditions of activity or inactivity that were initially mandated by CRM interactions, but which are retained long after the relevant transcription factors have disappeared from the system. Among phenomena of this type are histone acetylation and deacetylation, histone methylation, DNA methylation, installation of polycomb or trithorax group complexes (see Davidson, 2006 for review). As is also true of the memories of electronic computers, the information storage i.e., of a state of gene expression, is initially put in place by a computational interaction (on a CRM), and it can also be erased, by restoration of the initial state. However, a basic difference between the passive memories of electronic and genomic computers is that any place in the genome may be utilized for installation of a state of memory, while in electronic computers information is stored only in a priori dedicated, rigidly defined subdomains of the machine.

The genomic regulatory system utilizes another, extremely important kind of memory altogether. This is a dynamic or active rather than passive memory, in which transcriptional subcircuits maintain given regulatory states, as directed by inputs to continuously functional CRMs. Intra-cellular positive feedback subcircuits linking regulatory genes are an example (Fig. 2A). Once they are set up they force continuing transcriptional expression of the participant regulatory genes and thus of all genes which are downstream targets of these regulatory genes. Hence the regulatory state generated by the locked in set of regulatory genes is "remembered," that is, it continues to be presented. A second very common kind of active, transcriptionally driven memory subcircuit is the "community effect" (Fig. 2B) (Gurdon, 1988; Davidson, 2006). Here all cells of a territory signal to one another mutually, and this signaling is essential to maintenance (memory) of the territorial regulatory state. The mechanism is that in each cell the CRM of the gene encoding the signal ligand responds to the transcriptional effector of the same signal from the adjacent cell. This produces an intercellular positive feedback circuit. The unique feature is thus essentially the use of actively driven memory of state produced by inter- and intra-cellular transcriptional feedback subcircuits, a feature that is almost universally found in those gene regulatory networks so far analyzed (Davidson, 2006).

In the electronic computer the central processing unit (CPU) and the working memory used for computation are separate parts of the device. They are connected by hard wires. Due to the separation of the CPU and memory, all data and operations that need to be performed must travel from the memory to the CPU or between CPU and memory. This gives rise to what is called the "von Neumann bottleneck" (von Neumann, 1945). Wrote Backus, "Surely there must be a less primitive way of making big changes in the store than by pushing vast numbers of words through the von Neumann bottleneck …much of that traffic concerns not significant data itself but where to find it." (Backus, 1978). But such bottlenecks are irrelevant to the genomic computer, in which the *cis*-regulatory information processing nodes may be an intrinsic part of the dynamic memory circuitry. In the genomic computer, when passive memory is employed it does not directly provide input into the nodes where information processing is taking place; passive

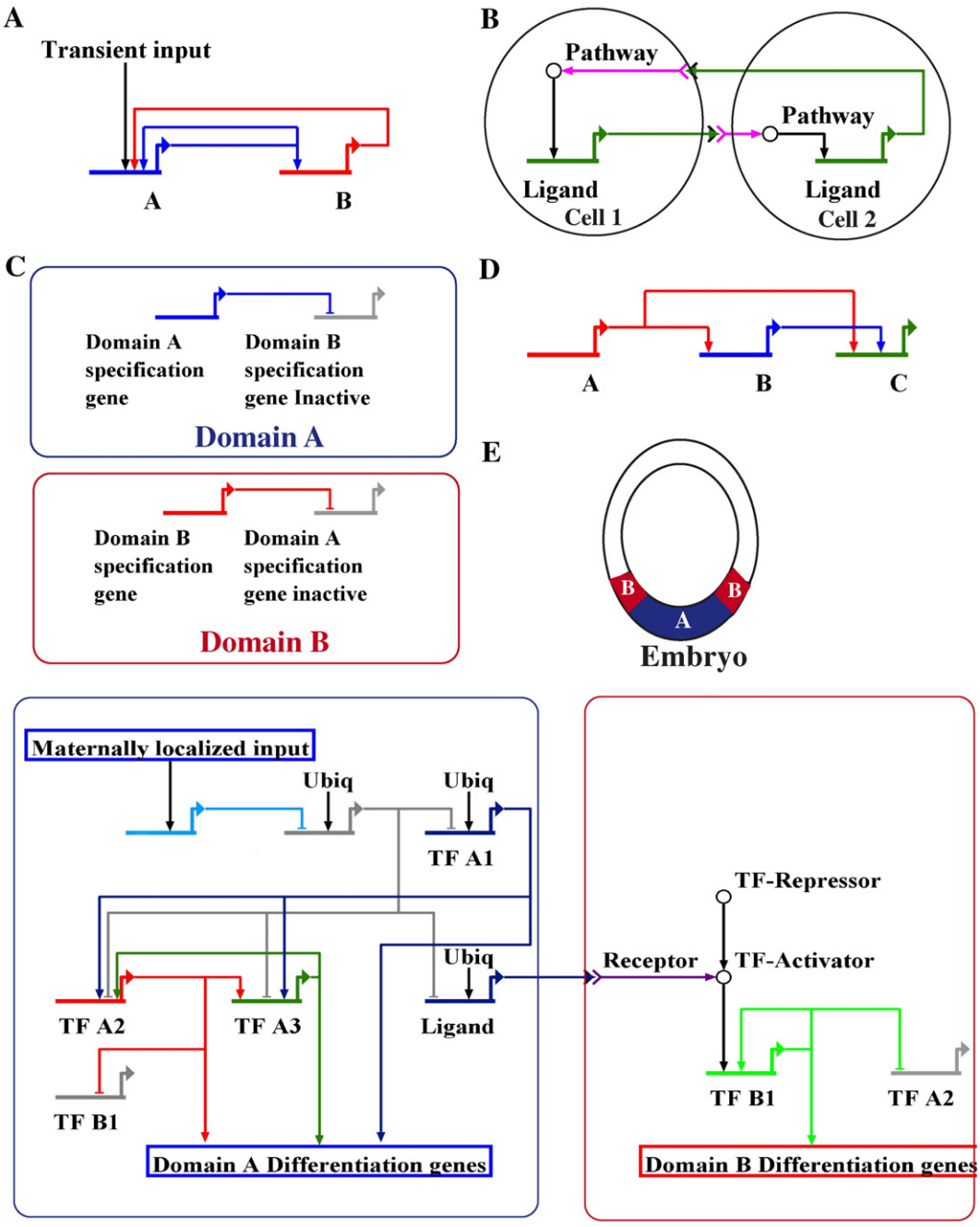

Fig. 2. Properties of the genomic computational system, dynamic memory and robustness. (A) An example of an intra-cellular positive feedback subcircuit. Gene A product auto-activates its own *cis*-regulatory element and also activates gene B expression. Gene B product feeds back to provide a necessary input into gene A *cis*-regulatory element. (B) Intercellular positive feedback circuit, a community effect. A ligand is secreted and activates a signaling pathway in the neighboring cell, which has a direct activating input into the ligand *cis*-regulatory element. Thus, all the neighboring cells in this territory signal to one another so the cells are locked in a specific regulatory state. (C) Exclusion of alternative fate by repression. One of the key specification genes of domain A represses a key specification gene of domain B, and vice versa. This mutual exclusion prevents specification state ambiguity that could result from an erroneous activation of both genes in the same territory. (D) Feed forward loop. Gene A activates the expression of gene B and together they activate the expression of gene C. (E) This synthetic network describes the specification of two neighboring domains in the embryo, A and B, as illustrated in the embryo diagram on the upper panel. The *cis*-regulatory element of "local repressor" (light blue line) responds to a maternally localized input so that local repressor is expressed only in domain A. Local repressor represses an otherwise ubiquitous repressor, "global repressor (gray line)," that keeps all the genes of domain A subcircuit off except from the cells where local repressor is expressed. Transcription factor A1 turns on in domain A due to local repressor activity and a ubiquitous activator. Transcription factor A1 provides a positive input into the *cis*-regulatory elements of two other targets of global repressor, transcription factors A2 and A3. Transcription factor A2 input is also necessary for the activation of transcription factor A3, so the three genes A1, A2 and A3 form a feed forward loop. Transcription factors A2 and A3 form a positive feedback loop that locks domain A in this regulatory state. The three genes, A1, A2 and A3, activate the expression of a differentiation gene battery that is specific to domain A. Another target gene of global repressor is ligand, a signaling molecule that is transcriptionally activated in domain A. Reception of ligand by the neighboring tier of cells activates the transcription factor B1 there. Transcription factor B1 locks itself on to fix domain B specification state and also activates domain B differentiation gene battery. In order to exclude alternative fate, transcription factor B1 represses the expression of transcription factor A2 in domain B and transcription factor A2 represses the expression of transcription factor B1 in domain A.

memory devices are downstream of the *cis*-regulatory responses to transcriptional inputs.

Robustness: the electronic computer and the regulatory genome

Robustness, the quality of being "strongly built" (Webster's English dictionary) is an urgent requirement because this quality is inversely related to the failure rate. Here we consider the kinds of strategies by which failure is minimized in genomic regulatory systems, compared to those of electronic computational systems. In electronic computers hardware solutions include features such as majority gates, that is, engagement of several identical copies of the same circuit with selection of the majority output; and inbuilt error detection circuits that check for glitches after the fact, e.g., by monitoring conservation of bit number in the results of a computation. Software may also incorporate many different strategies for diagnosis of error cause and location.

The possibility of failure is minimized in a completely different way in the genomic regulatory systems that control animal development (to which these considerations are confined): essentially, they utilize multiple design devices to doubly and triply ensure every regulatory state they set up. One strategy is use of diverse kinds of subcircuit which prohibit alternative states. A second is to lock down by an active transcription subcircuit the state required in a given spatial domain and at a given time. Commonly these are combined: a regulatory state is set up in a given domain in response to an initial transient input; the consequence of this is installation of a feedback circuit, sometimes multiple feedback subcircuits, within or among the cells of this domain which ensures the continued active maintenance of this state (Figs. 2A, B; see Memory: the electronic computer and the regulatory genome). But also, within the domain, repressors of possible alternative states are transcribed (Fig. 2C); and at the same time in cells outside the domain, the regulatory state of the cells within the domain considered is specifically repressed at the transcriptional level. Furthermore, in addition to feedback subcircuits, there are other positive circuit devices that reinforce given states beyond their initial specification. These include feed forward subcircuits (Fig. 2D) and more complex cross-regulatory subcircuits in which multiple inputs impinge on each gene, and these inputs themselves derive from genes which respond to one another's outputs. A synthetic example meant to illustrate the way such circuit design elements are combined to produce robust regulatory states in two adjacent territories is shown in Fig. 2E (see legend). The example is imaginary, but not too imaginary: all of the design features in Figs. 2A–D and all the combinations in Fig. 2E are indeed to be found in the sea urchin endomesoderm gene regulatory network (for current version see http://sugp.caltech.edu/endomes/), and they occur in various other such networks as well (reviewed in Davidson, 2006).

Genomic circuit designs of this nature may seem unnecessarily redundant, compared to a minimum streamlined engineering design. Their character emerges from two fundamental attributes. First, they have been selected in evolution to provide maximum insurance against the possibility of catastrophic developmental failure, even under rare circumstances. In their architecture resides the "strongly built" quality of cellular regulatory state specification. Second, the subcircuit elements of any given developmental system may be of different ages and origins in evolution (Davidson and Erwin, 2006); some pieces of the system have been co-opted from other functions, others assembled de novo and added in on top of pre-existing subcircuits.

In addition there is an underlying layer of robust regulatory design not considered here. This is in the internal structure of the individual *cis*-regulatory modules which constitute the nodes of gene regulatory networks (exemplified by the *endo16* gene; Yuh et al., 2001, 2005). For example such modules often contain multiple target sites for given factors and utilize multiple different factors to accomplish given tasks, such as promoting gene expression.

Hardware and software: the electronic computer and the regulatory genome

Computational paradigms are based on the concepts of hardware and software. However, nothing shows more clearly the distinctions between manmade computers and the genomic information processing system than the clumsiness of the concepts of hardware and software when these concepts are applied to biological regulatory systems. Genomic DNA is both the essential physical component of the regulatory apparatus (hardware) and the digital regulatory code (software); these properties are physically inseparable. The transcription factors are physical entities which execute regulatory functions (hardware) but their summed identity in any given nucleus at any time constitutes the regulatory input program for that moment (software); these properties too are physically inseparable. The concept of computational hardware is particularly inadequate for gene regulatory networks, which are continuously reorganizing themselves as new regulatory genes come into play and engage new targets to deploy new subcircuits. Readout of the DNA produces new transcription factor combinations and thus new regulatory states, and these cause further new readout of the DNA. This is the essence of development, i.e., change in regulatory state: but conventional computers do not develop as they work.

So how should we think of the overall genomic regulatory code for the body plan of each species that is inherited with its genomic sequence? The regulatory machinery is a real physical entity consisting of those DNA sequences constituting its *cis*-regulatory units. Its operation is mandated by its hardwired *cis*-regulatory sequence, but what it does depends entirely on the regulatory state these sequences see in each given cell at each given time. If we add together all the conditional gene network operations, integrating over all cells and the whole life cycle, then we arrive at the total regulatory potentiality of the organism. But the operative words here are "potential" and "conditional": genes and their regulatory modules are used in overlapping combinations to produce diverse functional connections among regulatory genes in different developmental contexts. We can

imagine a static genomic map that includes all transcription factor target sites for all genes, but this does not in itself tell us how the regulatory system works in development since we can only describe the regulatory code in terms of the operative gene regulatory network architecture for all times and cells. The conditional structure of the overall developmental regulatory system of an animal is the basic reason "hardware" and "software" as distinct aspects of the computational system are not useable concepts for the genomic regulatory computer.

Evolvability and the genomic computer

Evolutionary change in the body plans of animals is driven by reorganization of given aspects of developmental gene regulatory networks (Davidson and Erwin, 2006). What this means is altering network architecture by altering the regulatory linkages, particularly in CRMs that control regulatory gene expression. This is the basic and general mechanism of body plan evolution in animals, generally thought of as "co-option" of regulatory gene expression to new functions (for a review see Davidson, 2001). Mechanistically, co-option depends on *cis*-regulatory changes at the DNA level: either by the transposition of a pre-existing CRM to the vicinity of a different gene, causing the gene to fall under the different regulatory controls ordained by the DNA sequence of the incident CRM, or by the mutational acquisition in pre-existent CRMs of target sites for transcription factors that did not previously affect them. If the gene controlled by the CRM encodes a transcription factor, the result is potential expression of a novel transcriptional regulatory state with respect to time and space in the developing organism.

The regulatory evolution of the animal body plan by institution of novel linkages among regulatory genes, the most potent of all evolutionary mechanisms, depends directly on a definitive physical feature of the genomic computer. Thus the possibility that novel regulatory gene expression in a given context will affect downstream events follows immediately from the property that internal communication in the genomic regulatory computer works by means of transcription factor diffusion. For example, consider a regulatory gene that comes to be expressed in a new spatial domain due to appearance of novel activator sites in its CRM. In the diffusion-mediated interaction system the factor this gene encodes will automatically find its target sites in other genes since all potential DNA target sites are searched. Thereby CRM mutations may cause effective change in cellular regulatory state and hence in the functional character of cells in a new spatial domain. The diffusion-mediated communication mechanism of the genomic computational system can be said to provide the essential mechanistic basis of regulatory evolvability. This is of course a macroevolutionary mechanism. Alterations in regulatory state may only affect peripheral genes and produce small morphological changes, for example, by addition of a given keratin genes to an integument gene battery. But they may also produce much larger changes, as for example if they affect expression of inductive signaling molecules in space, or cause reassignment of gene network subcircuits which encode aspects of the body plan (cf. Davidson and Erwin, 2006).

Some implications

Abstractions focus the mind on essential features easily occluded by masses of details. Viewing the genomic regulatory system in terms of its logic processing functions is an abstract approach that will ultimately have direct practical import in the area of synthetic regulatory biology. At the *cis*-regulatory level, the first implication is that given kinds of logic gate and combinations thereof are functional features that are independent of specific transcription factor identities and biochemistries. That is, there are many ways to make an AND gate or a NOT processor. This in turn invites the approach of parsing known CRMs so as to deconstruct them into their component elemental logic functions (as in the start made by Istrail and Davidson, 2005). What will follow is the redesign of synthetic CRMs that are functionally equivalent to those which evolution has produced, in that they execute the same logic, though using different biochemistries and factors. By such means will we achieve a final level of demonstration that we understand what developmental CRMs are really doing. The same arguments pertain to synthetic redesign of information processing gene regulatory network subcircuits. At this level as well, multiple combinations of diverse regulatory genes are linked in different systems to generate the same abstract outputs (e.g., Oliveri and Davidson, 2007), which can be expressed in terms of computational logic. This is the pathway to constructing new biologically functional regulatory components, an inevitable, inescapable challenge that lies ahead. Constructing these will require understanding of the ways and means of the genomic computer, just as redesign of electronic computers requires comprehension of their principles.

A different kind of implication lies in the application of the concept of the genomic computer to early animal evolution. In all modern bilaterian animals information processing CRMs are utilized for development (Davidson, 2006). Such CRMs might likely exist in cnidarian and ctenophoran metazoans as well. Development of all animals that display tissue grade morphological organization and fixed body plans requires that spatial gene expression be exquisitely regulated. The fundamental CRM logic gates, particularly AND and NOT gates, are utilized in organizing spatial states of gene expression, specifically in mandating domains of expression as opposed to domains of non-expression. CRMs which operate in development according to basically similar principles of combinatorial logic have been found from *Drosophila* to sea urchins and mice (Davidson, 2006). Since such CRMs are a common shared feature of the genomic regulatory systems of complex animals, they must have been present in their common ancestors, the bodies of which, like those of their descendants, were formed by complex developmental processes.

The CRMs that process spatial inputs are themselves functionally linked into developmental gene regulatory networks in all types of animal embryo so far studied at this level. These networks are also a shared property of complex animals and therefore an ancestral character of complex animals. In considering the more remote origins of animal life

forms, we have to imagine how the earliest gene regulatory networks that controlled spatial gene expression in development might have been assembled. Network assembly basically requires that CRMs of given regulatory genes come to include target sites for the products of other regulatory genes located elsewhere in the enormous genome. The argument above shows that evolutionary change in pre-existing gene regulatory networks by co-option requires the existence of the global target search mechanism, i.e., diffusion of transcription factors, which the genomic computer utilizes for intra-system communication. Even the earliest steps of assembly of gene regulatory networks in remote evolutionary time must have depended from the outset on this same fundamental feature. Put another way, regulatory communication by non-directed diffusion underlies not only the evolvability of animal regulatory systems once they appeared, but even the initial existence of developmental processes of the nature of those utilized in modern animals.

In summary, a view of the evolutionary process leading to complex animals is that the essential properties of the genomic computer discussed in this essay were the condition for, and predate, complex animal forms: first came the properties of the genomic computer, including logic processing CRMs and regulatory network subcircuits, and then came programs for development built on these properties, and hence the animals.

Acknowledgments

Smadar Ben-Tabou de-Leon was supported by the Human Frontier Science Program Organization. Research was supported by NIH grant GM-61005.

References

Backus, J., 1978. Can programming be liberated from the von Neumann style? Commun. ACM 21, 613–641.

Barolo, S., Posakony, J.W., 2002. Three habits of highly effective signaling pathways: principles of transcriptional control by developmental cell signaling. Genes Dev. 16, 1167–1181.

Davidson, E.H., 2001. Genomic Regulatory Systems. Development and Evolution. Academic Press, San Diego.

Davidson, E.H., 2006. The Regulatory Genome. Gene regulatory networks in development and evolution. Academic Press/Elsevier, San Diego, CA.

Davidson, E.H., Erwin, D.H., 2006. Gene regulatory networks and the evolution of animal body plans. Science 311, 796–800.

Gurdon, J., 1988. A community effect in animal development. Nature 336, 772–774.

Hennessy, J.L., Patterson, D.A., 2003. Computer Architecture, a Quantitative Approach. Morgan Kaufmann Publishers.

Istrail, S., Davidson, E.H., 2005. Logic functions of the genomic *cis*-regulatory code. Proc. Natl. Acad. Sci. U. S. A. 102, 4954–4959.

Oliveri, P., Davidson, E.H., 2007. Built to run, not fail. Science 315, 1510–1511.

Stathopoulos, A., Levine, M., 2002. Dorsal gradient networks in the *Drosophila* embryo. Dev. Biol. 246, 57–67.

von Neumann, J., 1945. First Draft of Report on EDVAC, Technical Report. Moore School of Electrical Engineering, University of Pennsylvania.

von Neumann, J., 1958. The Computer and the Brain. Yale Univ. Press, New Haven.

Yuh, C.-H., Bolouri, H., Davidson, E.H., 2001. *cis*-Regulatory logic in the *endo16* gene: switching from a specification to a differentiation mode of control. Development 128, 617–628.

Yuh, C.-H., Brown, C.T., Livi, C.B., Rowen, L., Clarke, P.J.C., Davidson, E.H., 2002. Patchy interspecific sequence similarities efficiently identify positive *cis*-regulatory elements in the sea urchin. Dev. Biol. 246, 148–161.

Yuh, C.-H., Dorman, E.R., Howard, M.L., Davidson, E.H., 2004. An *otx* *cis*-regulatory module: a key node in the sea urchin endomesoderm gene regulatory network. Dev. Biol. 269, 536–551.

Yuh, C.-H., Dorman, E.R., Davidson, E.H., 2005. Brn1/2/4, the predicted midgut regulator of the *endo16* gene of the sea urchin embryo. Dev. Biol. 281, 286–298.