PHILOSOPHICAL TRANSACTIONS A

rsta.royalsocietypublishing.org

Research

Cite this article: Nehaniv CL, Rhodes J, Egri-Nagy A, Dini P, Rothstein Morris E, Horváth G, Karimi F, Schreckling D, Schilstra MJ. 2015 Symmetry structure in discrete models of biochemical systems: natural subsystems and the weak control hierarchy in a new model of computation driven by interactions. *Phil. Trans. R. Soc. A* **373**: 20140223.
http://dx.doi.org/10.1098/rsta.2014.0223

Accepted: 27 April 2015

One contribution of 13 to a Theo Murphy meeting issue 'Heterotic computing: exploiting hybrid computational devices'.

Subject Areas:
theory of computing, computational biology, computational chemistry, computer modelling and simulation, systems theory

Keywords:
interaction computing, algebraic automata theory, systems biology, automata networks

Author for correspondence:
Paolo Dini
e-mail: p.dini@herts.ac.uk

THE ROYAL SOCIETY PUBLISHING

Symmetry structure in discrete models of biochemical systems: natural subsystems and the weak control hierarchy in a new model of computation driven by interactions

Chrystopher L. Nehaniv[1], John Rhodes[2], Attila Egri-Nagy[1,3], Paolo Dini[1], Eric Rothstein Morris[4], Gábor Horváth[5], Fariba Karimi[1], Daniel Schreckling[4] and Maria J. Schilstra[1]

[1]Royal Society Wolfson Biocomputation Research Laboratory, University of Hertfordshire, Hatfield AL10 9AB, UK
[2]Department of Mathematics, University of California Berkeley, Berkeley, CA 94720, USA
[3]Centre for Research in Mathematics, University of Western Sydney, Locked Bag 1797, Penrith, New South Wales 2751, Australia
[4]Institute of IT Security and Security Law, University of Passau, Passau 94030, Germany
[5]Institute of Mathematics, University of Debrecen, Pf. 12. Debrecen, 4010 Hungary

Interaction computing is inspired by the observation that cell metabolic/regulatory systems construct order dynamically, through constrained interactions between their components and based on a wide range of possible inputs and environmental conditions. The goals of this work are to (i) identify and understand mathematically the natural subsystems and hierarchical relations in natural systems enabling this and (ii) use the resulting insights to define a new model of computation based on interactions that is useful for both biology and computation. The dynamical characteristics of the cellular pathways studied in systems biology relate,

mathematically, to the computational characteristics of automata derived from them, and their internal symmetry structures to computational power. Finite discrete automata models of biological systems such as the lac operon, the Krebs cycle and p53–mdm2 genetic regulation constructed from systems biology models have canonically associated algebraic structures (their transformation semigroups). These contain permutation groups (local substructures exhibiting symmetry) that correspond to 'pools of reversibility'. These *natural subsystems* are related to one another in a hierarchical manner by the notion of '*weak control*'. We present natural subsystems arising from several biological examples and their weak control hierarchies in detail. Finite simple non-Abelian groups are found in biological examples and can be harnessed to realize *finitary universal computation*. This allows ensembles of cells to achieve any desired finitary computational transformation, depending on external inputs, via suitably constrained interactions. Based on this, *interaction machines* that grow and change their structure recursively are introduced and applied, providing a natural model of computation driven by interactions.

> In the 21st century, algebra will be as important for biology as chemistry was in the 20th century.
> —Günter P. Wagner

1. Introduction

(a) Motivation and overview

Natural biological and biochemical systems are able to construct order dynamically through constrained interactions between their constituents based on a wide range of possible inputs and environmental conditions. What role does symmetry in biological systems play in this capacity to self-organize, and, in another direction, could similar mechanisms be exploited in computation driven by interaction? By examining natural symmetry structures that arise from the algebraic analysis of discrete models of biological and related biochemical systems, we aim to begin to answer these questions.

Our main goals are thus twofold: (i) to identify and understand mathematically the natural subsystems and hierarchical relations inside discrete models of natural systems and (ii) to use insights we find there to define a new model of computation driven by interactions that is useful for both biology and computation. Despite the fact that symmetry structures in biological systems continually degenerate (but are also renewed and replaced), we demonstrate here that nature may be using symmetry computationally in hidden ways. This leads us to consider dynamic ensembles (such as cell assemblies) whose constituents and interconnections change based on interaction and whose constituents have a rich structure of natural subsystems internally.

The remainder of §1 reviews necessary background on algebraic theory of automata, automata networks and algebraic structures associated with them, while motivating the *ensemble viewpoint* on multiple interacting systems that is developed further throughout the article. The article then proceeds as follows: §2 develops novel algebraic tools, including the concepts of *natural subsystems* (which are permutation groups in the algebraic structures associated to discrete dynamical systems) and their *weak control hierarchy*, for the analysis of discrete models of biological systems. Section 3 applies these tools to give detailed analyses of several examples of discrete biochemical and biological models (a Boolean network model of the *Escherichia coli* lac operon, reaction graph models of the Krebs cycle in metabolism, and a Petri net model of the p53–mdm2 genetic regulation system). We observe that simple non-Abelian groups (SNAGs) can occur as the symmetry groups acting in the natural subsystems. Section 4 explains how SNAGs can work as a basis for computation as they are functionally complete (like the two-element Boolean algebra in logic), reviews the background and recently demonstrated bounds on the realization of arbitrary finitary functions, and proves new results showing how discrete systems with SNAGs such as the

p53–mdm2 genetic regulatory system or ensembles of such systems could exploit this to achieve *finitary universal computation*. In §5, we introduce *interaction machines* as ensemble structures that dynamically grow or change their topologies based on internal and external interaction, and outline how these discrete dynamical systems can implement novel dynamic computational structures (e.g. dynamic cascades for positional number systems). Section 6 applies the interaction machine framework to show how it can naturally model biological systems that change their own structure (e.g. multicellular differentiation with cell division), discusses the interpretation of permutation groups in temporally changing interacting ensembles, and proves that ensembles of the natural subsystems can be arranged according to the weak control hierarchy to emulate a given discrete finite automaton. Moreover, we prove that the last can be achieved with a dynamic cascade, generalizing the holonomy method in algebraic automata theory for finite automata to decomposition using interaction machines. This gives the basis for further work on interaction computing.

(b) Automata and their associated algebra structures

Numerous biologists and their collaborators now model biological, biochemical or genetic regulatory systems using discrete models such as Petri nets, Boolean networks or multi-valued logical networks [1–6], and the use of annotated directed graphs to encode biological networks is even more prevalent in numerous scientific repositories. This entails that these biological models are finite automata or well approximated by finite automata,[1] or can be discretized to yield finite automata models. Automata are discrete dynamical system structures whose state-space trajectories are influenced by events or external inputs. They have associated algebraic structures (their transformation semigroups) and these include dynamical building-block components as substructures that can be expressed in terms of mathematical symmetries (permutation groups). An example of automata from biochemical models appears in §3.

A (complete, deterministic) *automaton* $\mathcal{A}$ is a tuple (Q, A, δ), where A is the set of *input symbols* (the *input alphabet*—also called *inputs* or *basic events*), $Q = States(\mathcal{A})$ is the set of *states*, the *state space*, and δ is the (fully defined) *state transition function* $\delta\colon Q \times A \to Q$. Note that use of deterministic automata does not preclude stochasticity in their input/event streams, i.e. which event occurs or which transition fires may well be determined probabilistically depending on current configuration, environment factors, etc.

One could also include in the tuple an *output function* $\epsilon\colon Q \to B$ to an *output alphabet* B. As output can be recovered from state, we shall not use an explicit output function, but assume state information is available where needed (equivalently, the output function is the identity function on Q and the output alphabet is Q).[2] Generally, we shall consider *finite-state automata*, i.e. all the sets involved are finite.

A *semigroup* S is a set with an associative multiplication defined on it, i.e. $(st)u = s(tu)$ for all $s, t, u \in S$. Let A^+ be the *free semigroup generated by* A, i.e. all the finite words with positive length constructed from the alphabet A with concatenation as the associative binary operation. Similarly, $A^* = A^+ \cup \{\lambda\}$ is the *free monoid on* A consisting of all finite words and the empty word λ which acts as the multiplicative identity for concatenation. For any $t = a_1 \cdots a_n \in A^+$ let us define $\delta^+(t)\colon Q \to Q$ inductively: $\delta^+(a_1)(q) = \delta(q, a_1)$ for $a_1 \in A$ and $q \in Q$. Let $\delta^+(a_1 \cdots a_k)(q) = \delta^+(a_k)(\delta^+(a_1 \cdots a_{k-1})(q))$ for $k > 1$, $a_1 \cdots a_k \in A^+$ and $q \in Q$. Let $F(Q) = Q^Q$ denote the semigroup of all transformations of Q into itself under function composition $\circ$, where, for $f, g \in F(Q)$, we have $(f \circ g)(q) = g(f(q))$. Then $\delta^+\colon A^+ \to F(Q)$ is a homomorphism. Let us denote $\delta^+(A^+)$ by $S(\mathcal{A})$, and call it the *characteristic semigroup of* $\mathcal{A}$. Thus, a word in A^+ gives a member of the semigroup of $\mathcal{A}$ whose elements are transformation of Q, and concatenation of words corresponds to the function

[1] As in the case of Petri nets with no *a priori* bound on the number of tokens, see [7] for details on constructing finite automata from Petri nets.

[2] The formulation with output as a function of state is sometimes called a Moore automaton. Alternatively, the output function may be allowed to depend on input, $\epsilon\colon Q \times A \to B$, giving an alternative definition (Mealy automaton) with similar properties for which one can develop essentially the same theory and generalizations as we do in the article with interaction machines.

composition for these transformations. We can extend δ^+ to A^* by mapping the empty word to the identity function on Q giving a monoid homomorphism $\delta^*: A^* \to F(Q)$. The pair $(Q, S(\mathcal{A}))$ is the *transformation semigroup of the automaton* $\mathcal{A}$. Similarly, $M(\mathcal{A}) = \delta^*(A^*)$ is the *characteristic monoid of* $\mathcal{A}$ and $(Q, M(\mathcal{A}))$ is its *transformation monoid*.[3]

Notation: we write $q \cdot a$ for $\delta(q, a) = \delta^+(a)(q) = \delta^*(a)(q)$, where q is a state and a is a basic event. This extends to all finite words by letting $q \cdot w = \delta^*(w)(q)$. We have $q \cdot \lambda = q$ and $(q \cdot w) \cdot v = q \cdot wv$, for all words $w, v \in A^*$ and for all states $q \in Q$, as δ^* is a homomorphism. Simply put, $q \cdot w$ denotes the new state if, starting with state q, the sequence of events w occurs, so $\cdot w = \delta^*(w) \in S(\mathcal{A})$ is the mapping on states given by the input word $w \in A^*$. If $X \subseteq Q$ and $w \in A^*$, we write $X \cdot w$ for $\{x \cdot w \mid x \in X\}$.

A *group* G is a semigroup with unique idempotent element $e = e^2$ that acts as a right and left *identity* on G: $ge = eg = g$ for all $g \in G$, such that for each $g \in G$ there is an *inverse* $h \in G$ with $hg = e = gh$. Given any subgroup G of $S(\mathcal{A})$, that is, a subset of $S(\mathcal{A})$ which is a group, with identity element $e \in G$, define $X(G) = \{q \in Q : q \cdot e = q\}$. Note that e is an idempotent $e^2 = e$ but will generally not be the identity map on Q. Then it is easy to check that $(X(G), G)$ is a *permutation group*. That is, each of its elements $g \in G$ acts as a permutation of $X(G)$, e acts as the identity on $X(G)$, and the (unique) inverse $h \in G$ of g acts as the inverse permutation to g.

Note: if a transformation semigroup (X, S) is given and G is a subgroup of S, then (X, G) may not be a group of permutations (e.g. consider a constant map on n points giving $(X_n, \{1\})$). However, it is easy to prove that each member of G has the same range $X(G)$ on X, i.e. for all $g \in G$, $X \cdot g = \{x \cdot g : x \in X\} = X \cdot e = \{x : x \cdot e = x\} \equiv X(G)$, where $e^2 = e$ is the identity of G. Thus $G \mapsto X(G)$ maps subgroups of S into subsets of X. As they are closed under the action of the group G by permutations, the transitive components of $X(G)$ (in $(X(G), G)$) can be considered as 'pools of feedback' or 'reaction chains' or 'natural subsystems', etc.

Transformation semigroups and permutation groups are key concepts in algebraic automata theory, where the Krohn–Rhodes prime decomposition theory guarantees that any given finite automaton can be emulated by a cascade (automata network with fixed linear topology) whose component automata are basic atomic building blocks (finite simple group automata (permutation groups) and substructures of the flip-flop, a two-state identity-reset automaton) [8–12]. The computationally most important proof of the Krohn–Rhodes theory, the holonomy theorem [9,13,14], proceeds by finding representative permutation groups for the automaton. These permutation groups are of the form $(X(G), G)$, where G is a maximal subgroup of the semigroup $S(\mathcal{A})$. We shall study them and their relationships for a number of biological systems below (§2) on natural subsystems of p53, the lac operon, the Krebs cycle and their weak control relations. Section 1d states and explains a useful elementary version of the Krohn–Rhodes decomposition (theorem 1.1). For notation and extensive background, the reader may refer to [8,9,11–13].

(c) Ensemble approach

Within a body, there may be many nearly identical cells (or cells in various states of differentiation descended from a common progenitor cell), and within a cell there are many instances of particular reaction networks (such as the Krebs citric acid cycle of intermediary metabolism, etc.). These biological examples suggest another interpretation for automata and their hierarchical (Krohn–Rhodes) decompositions, based on ideas borrowed from statistical mechanics. In this *ensemble approach*,[4] we consider the state transitions of many copies of the same automaton.

Let n be a positive integer. Then from an automaton $\mathcal{A} = (Q, A, \delta)$, we get a direct product power $\mathcal{A}^n$ consisting of n copies of $\mathcal{A}$ running in parallel and each accepting the same input $a \in A$

[3]We remark that it is also common to write (Q, S) for an *action of a semigroup* S on a set Q: that is, each $s \in S$ gives function $q \mapsto q \cdot s$ on Q satisfying $(q \cdot s) \cdot s' = q \cdot (ss')$ for all $q \in Q, s, s' \in S$. An action is *faithful* if $q \cdot s = q \cdot s'$ for all $q \in Q$ implies $s = s'$. In particular, the transformation semigroup or monoid of an automaton is obviously faithful as its semigroup's elements lie in $F(Q)$.

[4]This idea is due to John L. Rhodes [11] following physicists Josiah Willard Gibbs and Erwin Schrödinger [15].

Table 1. States in a Boolean network model of the lac operon. The states are defined by Boolean combinations of the presence/activity versus absence/inactivity of biochemical components: **L**, lactose; **A**, allolactose is an isomer of lactose; **Op**, the repressor molecule; **ZYA**, the structural genes for the enzymes needed for lactose metabolism. In the Boolean network, 1 means that the molecule is present/active or the gene is expressed; 0 is for the absence/inactivity. States are given by binary column vectors, e.g. 1100 corresponds to the state L A, where **L** and **A** have value 1 and **Op** and **ZYA** have value 0. States to the left of the vertical separating line correspond to metabolic states in which lactose is present.

Boolean variable	Boolean state vector															
L	1	1	1	1	1	1	1	1	0	0	0	0	0	0	0	0
A	1	1	1	1	0	0	0	0	1	1	1	1	0	0	0	0
Op	1	1	0	0	1	1	0	0	1	1	0	0	1	1	0	0
ZYA	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0

from a state $q_i \in Q$ $(1 \leq i \leq n)$:

$$(q_1, \ldots, q_n) \cdot a = (q_1 \cdot a, \ldots, q_n \cdot a).$$

This can be construed mathematically or physically (for the latter, there will be distinct physical copies of the system present at the same time, e.g. cells in a multicellular organism). In particular, if we consider $n = |Q|$ and enumerate the elements of Q as $q_1, \ldots, q_n$, then each elementary event a transforms the ensemble in a way that concretely and completely describes a as a transformation. Gibbs essentially did this to consider statistical mechanics of physical systems [11,15]. Alternatively, we might consider the dynamically stable states (i.e. orbits[5]) of a discrete dynamical system and take a smaller $m < |Q|$ and start with a tuple $(q'_1, \ldots, q'_m)$ with the $q'_i \in Q$ representing all the distinct dynamically stable states of the system, and study how events transform ensembles of dynamically stable states to other dynamically stable states (cf. [4]).

We shall speak of ensembles when we have many copies of the same or different automata running in parallel as simple *direct-product ensembles*, but we shall also consider more general networks of automata updated synchronously (possibly with dependencies between them) as *ensembles*. For example, the state of a lac operon in each cell in a large population of *E. coli* cells growing in a Petri dish may be modelled as an ensemble of copies of the automaton in table 1.

Krohn–Rhodes decompositions (such as in theorem 1.1 and the holonomy decomposition [9, 13], which we have computationally implemented [16–18]) give a 'hierarchical coordinate system' on the semigroups of transformations acting on the state set of an automaton.[6] Here higher level coordinates correspond to a coarse-graining of the system, i.e. to *subsets* of the state set. Thus, we can regard any level of the decomposition as an ensemble being transformed in parallel as in a direct product. One can therefore use the decomposition as an atlas with arbitrary scale at different hierarchical levels, from coarse to fine, for the global states of the ensemble of cells. This approach is appealing biologically and also removes some of the difficulties originating from a discrete modelling approach.

Clearly, the global state of an ensemble can be defined in many different ways (e.g. the set of observed states, or their frequency distribution). The choice of an 'update rule' (how the transformations are applied to the individual automata, the members of the ensemble), if not synchronous, may potentially change the behaviour of the ensemble [19], but we shall restrict ourselves to synchronous update here.

For direct-product ensembles of a single system, a given transformation $s \in S(\mathcal{A})$, resulting from a sequence of basic events or inputs, is interpreted mathematically or physically as simultaneously being applied to each copy as they all change state in parallel and independently.

[5]Similar to continuous systems, in discrete dynamical systems, an *orbit* comprises a set of system states closed under the action of time, i.e. under the occurrence of sequence events $a \in A$ (or transformations s in some group or semigroup).

[6]See [11] for details and examples of the important and deep notion of how to interpret cascade decompositions as coordinate systems.

For more complex ensembles, the effect of this transitioning occurs synchronously but not necessarily independently. In particular, this is the case for networks of automata, Krohn–Rhodes decompositions, and, as we shall see, for interaction machines whose topology and constituent components may change dynamically. The ensemble approach can be used as a guiding metaphor for building and understanding complex systems.

(d) Automata networks

We consider not only strictly parallel ensembles but also more complex networks of automata. A *directed graph* (or *digraph*) $\Gamma = (V, E)$ is a set of nodes V and directed edges $E \subseteq V \times V$. If v is a node in a digraph Γ, then its *neighbourhood* $N(v)$ is the set of nodes with incoming edges to v, i.e. $N(v) = \{v' \in V : (v', v) \in E\}$. An *automata network* $\mathfrak{A} = (\Gamma, \{\mathcal{A}_v\}_{v\in V}, \{\varphi_v\}_{v\in V}, A)$ consists of a collection of automata $\mathcal{A}_v = (Q_v, A_v, \delta_v)$ associated with the vertices $v \in V$ of a digraph $\Gamma = (V, E)$, a global input alphabet A and *interaction functions*

$$\varphi_v : \prod_{w\in N(v)} Q_w \times A \to A_v,$$

for each node v of Γ. A state of the network is a choice of state for each component automaton, i.e. $States(\mathfrak{A}) = \prod_{v\in V} Q_v$. Given a global input $a \in A$ and a state of the automata network, the next input to the automaton at node v is a function φ_v of the states of the automata in the neighbourhood of v and the input a. Thus, the new state of the automaton at node v may be written as

$$q'_v = \delta_v(q_v, \varphi_v(q_{N(v)}, a)),$$

where $q_v \in Q_v$ and $q_{N(v)} := \{q_w\}_{w\in N(v)}$ are, respectively, the current state at v and the states $q_w \in Q_w$ of all nodes w in the neighbourhood of v.[7]

Special cases.

(1) An automata network is called a 'general product' of automata if Γ is a complete graph, i.e. $E = V \times V$. Without loss of generality, we may take $V = \{1, \ldots, n\}$ in the case of finite V. Then for each $i \in V$, there is an interaction function $\varphi_i : Q_1 \times \cdots \times Q_n \times A \to A_i$. The *general product* (or *Gluškov product*) of the automata $\mathcal{A}_i = (Q_i, A_i, \delta_i)$ with respect to the interaction functions $\varphi_i (i \in \{1, \ldots, n\})$ is then the automaton $\mathcal{A}$ with state set $Q = Q_1 \times \cdots \times Q_n$, input alphabet A, transition function δ given by $\delta((q_1, \ldots, q_n), a) = (\delta_1(q_1, \varphi_1(q_1, \ldots, q_n, a)), \ldots, \delta_n(q_n, \varphi_n(q_1, \ldots, q_n, a)))$, for all $(q_1, \ldots, q_n) \in Q$ and $a \in A$.
Note that automata networks can be defined in terms of a general product of their components where the interaction functions depend just on the neighbourhood states $q_{N(v)}$ and the global input.

(2) In the network $\mathfrak{A}$, if all automata are copies of the same automaton $\mathcal{A}_v = \mathcal{A}$ (for all $v \in V$), then we say that $\mathcal{A}$ is a *(general) power* of $\mathcal{A}$, or an *ensemble* of copies of $\mathcal{A}$.

(3) Another special case is a *direct product of automata* where $E = \emptyset$, i.e. the graph with disconnected nodes. This is still taken with respect to interaction functions $\varphi_v : A \to A_v$ $(v \in V)$, depending only on the global input $a \in A$, so

$$q'_v = \delta_v(q_v, \varphi_v(a)).$$

(4) If the underlying graph of the network is totally ordered (or more generally directed acyclic), then $\mathfrak{A}$ is said to be a *cascade* of its component automata. In the linear case with nodes $V = \{1, \ldots, n\}$ ordered as usual, each interaction function φ_i depends only on the global input a and the states q_j with $j < i$, and not on $q_i, \ldots, q_n$. That is, $\varphi_i : \prod_{j<i} Q_j \times A \to A_i$, so that the new state of $\mathfrak{A}$ is given locally as $q'_i = \delta_i(q_i, \varphi_i(q_1, \ldots, q_{i-1}, a))$, where $q_j \in Q_j$ and $a \in A$.

[7] In the literature, what we call 'interaction functions' have been referred to as 'feedback functions' [9,20].

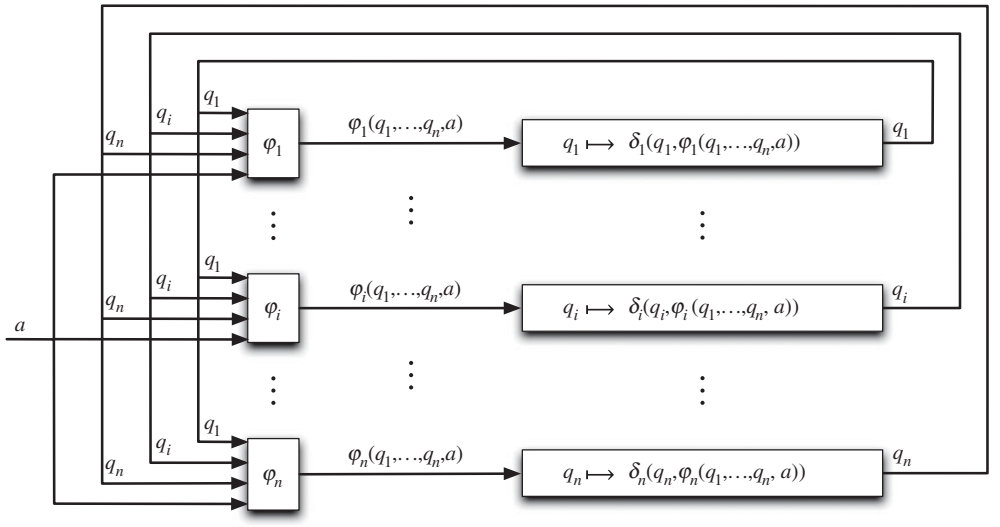

Figure 1. General product. (Adapted from [9, p. 59].)

(5) We shall use the interaction functions $\varphi_i, i \in V = \{1,\ldots,n\}$ in an extended sense as mappings $\varphi_i^* : Q_1 \times \cdots \times Q_n \times A^* \to A_i^*$, where $\varphi_i^*(q_1,\ldots,q_n,\lambda) = \lambda$ and

$$\varphi_i^*(q_1,\ldots,q_n,pa) = \varphi_i^*(q_1,\ldots,q_n,p)\varphi_i(\delta_1(q_1,\varphi_1^*(q_1,\ldots,q_n,p)),\ldots,\delta_n(q_n,\varphi_n^*(q_1,\ldots,q_n,p)),a),$$

where $q_i \in Q_i$, $i = 1,\ldots,n, p \in A^*, a \in A$. In the sequel, φ_i^* will also be denoted by φ_i. This is equivalent to allowing words rather than just letters in the inputs to each component automaton $\mathcal{A}_i$. Or, again equivalently, we may regard φ_i as taking values in the characteristic monoid $M(\mathcal{A}_i)$ of the transformations acting naturally on $\mathcal{A}_i$'s states Q_i that are induced by words over A_i. (This too holds of course if φ_i only depends on some of its arguments, i.e. those with indices in the neighbourhood of $i \in V$, where without loss of generality $V = \{1,\ldots,n\}$.)

We can imagine the structure of the general product $\mathfrak{A}$ as a working model in the following way:[8] the product is a collection of automata such that every member $\mathcal{A}_i$ of this collection is supplied with a transformer which is a special type of an automaton (figure 1). The transformers, realizing the interaction functions φ_i mentioned above, are able to get an input vector containing a common external input sign and the state of all component automata in their neighbourhood of $\mathcal{A}_v$ according to the directed graph V. They can each transform this input vector into an appropriate input sign for their component automaton. The automata network $\mathfrak{A}$ is at work along a discrete time scale in the following way: all transformers of the product get a common external input event a, and, simultaneously, get the value of the instantaneous states $q_1,\ldots,q_n$ of all component automata as input information. By the effect of this input vector $(q_1,\ldots,q_n,a)$, the transformers produce an input sign $a_i = \varphi_i(q_1,\ldots,q_n,a) \in A_i, i = 1,\ldots,n$ for their component automata. Then, by the effect of these (transformed) input signs, every component automaton goes into a new (not necessarily different) state $\delta_i(q_i,a_i) = \delta_i(q_i,\varphi_i(q_1,\ldots,q_n,a))$ and then, in the next time period, this process all happens again.

The transformation semigroup $(Q, S(\mathcal{A}))$ of a finite automaton $\mathcal{A} = (Q, A, \delta)$ can be regarded as an automaton itself; its basic event or input alphabet is the collection of all transformations

[8]The definition of automata network and the explanation are adapted from [9, p. 59] by P. Dömösi and C. L. Nehaniv.

$t\colon Q \to Q$ given by some word $w \in A^*$ in the inputs of $\mathcal{A}$. We may then write $t = [w]$ or $t = \cdot w$, for this transformation $t \in S(\mathcal{A}) \subseteq Q^Q$, in the dynamical space of $\mathcal{A}$.

An automaton $\mathcal{A}' = (Q', A', \delta')$ *emulates* automaton $\mathcal{A}$ if $\mathcal{A}'$ has a subautomaton mapping onto $\mathcal{A}$. That is, there exist $\tilde{Q} \subseteq Q'$, $\tilde{A} \subseteq A'$ with $\delta'(\tilde{q}, \tilde{a}) \in \tilde{Q}$ for all $\tilde{q} \in \tilde{Q}, \tilde{a} \in \tilde{A}$, and surjective functions $\varphi_1 : \tilde{Q} \to Q$ and $\varphi_2 : \tilde{A} \to A$ satisfying $\varphi_1(\delta'(\tilde{q}, \tilde{a})) = \delta(\varphi_1(\tilde{q}), \varphi_2(\tilde{a}))$.

Very important for understanding the fundamental building blocks of finite discrete dynamical systems are the cascade products of transformation semigroups; in particular, one can emulate any finite automaton $\mathcal{A}$ by cascading representative symmetry structures (permutation groups) arising from $(Q, S(\mathcal{A}))$ and copies of two-state identity reset automata (so-called 'flip-flops').

Theorem 1.1 (Krohn–Rhodes [8,11,13,21].). *Let $\mathcal{A}$ be a finite automaton. Then there is a cascade $\mathfrak{A} = (\Gamma, \{\mathcal{A}_v\}_{v\in V}, \{\varphi_v\}_{v\in V}, A')$, with Γ linearly ordered, such that $\mathfrak{A}$ emulates $\mathcal{A}$ and each $S(\mathcal{A}_v)$ is either a group or contains no non-trivial group.*

Let $c(\mathfrak{A})$ be the number of $v \in V$ for which $S(\mathcal{A}_v)$ is a non-trivial group. The *(Krohn–Rhodes) complexity* of $\mathcal{A}$ is the least $c(\mathfrak{A})$ over all cascades $\mathfrak{A}$ emulating $\mathcal{A}$ as in theorem 1.1.

The Krohn–Rhodes prime decomposition theorem refines theorem 1.1 to guarantee that $S(\mathcal{A}_v)$ can be chosen to be finite simple groups and flip-flops and describes which irreducible building blocks must occur in every decomposition of a given automaton $\mathcal{A}$ [8,11,13,21].

In §5, we define an *interaction machine* (of level 1) as an automata network that can dynamically change its structure and topology in the course of an update (following [19,22–24]). This points to a method for keeping computational activity far from equilibrium by continually regenerating structure. Interaction machines may themselves be recursively combined into networks that dynamically respond when interacting with each other and the environment. An interaction machine (of level $n+1$) will comprise a dynamic network of interaction machines of level n and below. More generally, interaction machines consist of dynamic networks of interaction machines without regard to level (as, for example, these levels may be changing or unknown).

2. Weak control

(a) $\mathcal{R}$-equivalence, permutator and stabilizer of states

In the following, X, Y, Z, etc., with various subscripts denote subsets of finite set $Q = States(\mathcal{A})$.

The set of *stabilizer strings* of a subset X of states is

$$(X)\mathbf{stab}_{str} = \{w \in A^* \mid X \cdot w \subseteq X\}.$$

It consists of all input strings w that map each state in X to some state in X. It has a subset, the *permutator strings* of X,

$$(X)\mathbf{per}_{str} = \{w \in A^* \mid X \cdot w = X\},$$

which since X is finite consists of the strings that permute the members of X.

In particular, $(X)\mathbf{stab}_{str}$ and $(X)\mathbf{per}_{str}$ are subsets of A^* containing the empty word $\lambda \in A^*$, and each is closed under concatenation of its members. So $(X)\mathbf{per}_{str}$ is a subsemigroup of $(X)\mathbf{stab}_{str}$.

Define the *stabilizer* $(X)\mathbf{stab}$ of X to be the set of equivalence classes of words in $(X)\mathbf{stab}_{str}$ that realize the same transformation of X. Thus, two stabilizer strings w_1 and w_2 are identified in $(X)\mathbf{stab}$ when they agree on all $x \in X$, i.e. each $w \in (X)\mathbf{stab}_{str}$ is considered as the function $x \mapsto x \cdot w$ on X. Similarly, $(X)\mathbf{stab}_{sgp}$ denotes the equivalence classes of words w in $(X)\mathbf{stab}_{str}$ yielding the same transformation $q \mapsto q \cdot w$, for all $q \in Q$, i.e. as members of the characteristic monoid $M(\mathcal{A})$.

The *permutator group* $(X)\mathbf{per}$ of a subset X consists of, by definition, those members of $(X)\mathbf{stab}$ which are permutations of X (i.e. 1 : 1 and onto, but of course 1 : 1 is equivalent to onto because X is finite). Then $(X)\mathbf{per}$ is a subgroup of $(X)\mathbf{stab}$ and $(X, (X)\mathbf{per})$ is a permutation group.

The *permutator semigroup* $(X)\mathbf{per}_{sgp}$ is the set of mappings on all states Q given by members of $(X)\mathbf{per}_{str}$. Then restriction to $X \subseteq Q$ is a homomorphism from $(X)\mathbf{per}_{sgp}$ onto $(X)\mathbf{per}$.

We have the following commutative diagram relating these semigroups, including the group $(X)\mathbf{per}$, via natural homomorphisms and inclusions:

$$\begin{array}{ccccc} A^* & & Q^Q & & X^X \\ \cup\!| & & \cup\!| & & \cup\!| \\ (X)\mathbf{stab}_{str} & \overset{\eta}{\twoheadrightarrow} & (X)\mathbf{stab}_{sgp} & \overset{\rho}{\twoheadrightarrow} & (X)\mathbf{stab} \\ \cup\!| & & \cup\!| & & \cup\!| \\ (X)\mathbf{per}_{str} & \overset{\eta}{\twoheadrightarrow} & (X)\mathbf{per}_{sgp} & \overset{\rho}{\twoheadrightarrow} & (X)\mathbf{per} \end{array}$$

Here, for $w \in A^*$, $\eta(w)$ is the mapping $Q \to Q$ given by $\eta\colon w \mapsto \cdot w$, and ρ is restriction to X. Thus, $(X, (X)\mathbf{stab}_{str})$ is a not necessarily faithful (right) transformation monoid, $(X, (X)\mathbf{stab})$ is a faithful transformation monoid and $(X)\mathbf{stab}_{sgp}$ is a submonoid of the characteristic monoid $M(\mathcal{A})$.

Theorem 2.1. *The restriction homomorphism from* $(X)\mathbf{per}_{sgp}$ *to* $(X)\mathbf{per}$ *is injective on any subgroup* G *of* $(X)\mathbf{per}_{sgp}$ *whose identity element* e *has image* X.

Proof. Let G be a subgroup of $(X)\mathbf{per}_{sgp}$ with identity element $e^2 = e$ and $Q \cdot e = X$, where Q is the full state set. Consider $g_1, g_2 \in G$ and $q \in Q$. Suppose the restrictions of g_1 and g_2 to X agree, then $q \cdot g_1 = q \cdot eg_1 = (q \cdot e) \cdot g_1 = (q \cdot e) \cdot g_2 = q \cdot eg_2 = q \cdot g_2$. So g_1 and g_2 also agree on the full state set, hence $g_1 = g_2$. ∎

For subsets $X_1, X_2 \subseteq Q$, we define the reflexive transitive relation $X_1 \geq_{\mathcal{R}} X_2$, if there exists $t \in A^*$ such that $X_1 \cdot t = X_2$. We define $X_1 \mathcal{R} X_2$ if and only if there exists $t_{12}, t_{21} \in A^*$ such that $X_1 \cdot t_{12} = X_2$ and $X_2 \cdot t_{21} = X_1$. In this case $x_1 \mapsto x_1 \cdot t_{12}$ and $x_2 \mapsto x_2 \cdot t_{21}$, for $x_j \in X_j$ are bijections. Then for some positive integer w, $(t_{12}t_{21})^w$ is the identity map on X_1, so, by replacing t_{12} by $(t_{12}t_{21})^{w-1}t_{12}$, we can assume the two bijections above are inverses of each other.

Theorem 2.2. *If* X_1 *is* $\mathcal{R}$*-equivalent to* X_2*, then we have an isomorphism of permutation groups*

$$(X_1, (X_1)\mathbf{per}) \cong (X_2, (X_2)\mathbf{per}).$$

Proof. Using the notation above, it is immediate to check that the bijective mappings $x_1 \mapsto x_1 \cdot t_{12}$ (for states $x_1 \in X_1$) and $s \mapsto t_{21} s t_{12}$ (for transformations $s \in (X_1)\mathbf{per}$) respect the group action giving an isomorphism of permutation groups. ∎

An element e of a semigroup is *idempotent* if $e^2 = e$. There is an important transitive, reflexive *natural partial order* on idempotents: $e_1 \leq e_2$ if and only if $e_1 e_2 = e_2 = e_2 e_1$.

(b) Natural subsystems and weak control of one permutator group over another

We shall only be concerned with (non-trivial) permutators of subsets $X_w = Q \cdot w$ that are reachable and stable under some sequence of events $w \in A^*$, where $X_w \cdot w = X_w$. In the ensemble viewpoint, starting the system in every possible state and feeding in the input event sequence w results in the set of possible states X_w. For $X = X_w$, we call the permutation group $(X, (X)\mathbf{per})$ a *natural subsystem* for the automaton $\mathcal{A}$.

The transformation on Q due to w has a unique idempotent power $e = e^2$. We assume G to be the (unique[9]) maximal subgroup of $S(\mathcal{A})$ with e as the identity, and let $X(G) = X = Q \cdot e = X \cdot e$, which we may also denote $X(e)$. We have $Q \cdot e = X \cdot e \subseteq X \cdot G = \{x \cdot g : x \in X, g \in G\} \subseteq (X \cdot G) \cdot e \subseteq Q \cdot e$, so $X(G) = X \cdot G$. By theorem 2.1, we have that $(X(G), G)$ is a faithful permutation group. We also call $(X(G), G)$ a natural subsystem, although unlike $(X, (X)\mathbf{per})$ the elements of its group are defined on all states of Q, not just on $X \subseteq Q$. Of course each element of G will determine a unique member of $(X)\mathbf{per}$, but many different maximal subgroups G of $(X)\mathbf{per}_{sgp}$ may act by permutations on $X(G)$, and each such G embeds in $(X)\mathbf{per}$ by theorem 2.1.[10]

[9]Standard results of semigroup theory (e.g. [8,25–27]) show that every idempotent is contained in a unique maximal subgroup G_e, i.e. if H is a subgroup containing e then $H \subseteq G_e$.

[10]Thus, there frequently may exist different w with different idempotent powers e stabilizing the same image $X = X \cdot w = X(e) = X(G_e)$. In that case, it follows from footnote 9 that the different groups G_e are disjoint.

We say $(X_1,(X_1)\mathbf{per})$ exercises *weak control* over $(Y_1,(Y_1)\mathbf{per})$ if and only if there exists X_2 with $X_1\,\mathcal{R}\,X_2$ so $(X_1,(X_1)\mathbf{per})\cong(X_2,(X_2)\mathbf{per})$ and $X_2\geq_{\mathcal{R}}Y_1$, $Y_1\subseteq X_2$ (therefore there exists $r\in A^*$ so that $X_2\cdot r=Y_1\subsetneq X_2$, but no $r_0\in A^*$ so that $Y_1\cdot r_0=X_2$). So $r\in(X_2)\mathbf{stab}_{str}$ and r is not a permutation when restricted to X_2. We write $(X_1,(X_1)\mathbf{per})\succ_{wc}(X_2,(X_2)\mathbf{per})$ in this case.

In studying automata models of biological (metabolic, genetic regulatory, etc.) systems, we want to return for biochemical and biological inspection strings (i) which generate $(X_2)\mathbf{per}$, (ii) string r with $X_2\cdot r=Y_1$, and (iii) strings which generate $(Y_1)\mathbf{per}$.

Lemma 2.3. *For natural subsystems $(X(G_1),G_1)$ and $(X(G_2),G_2)$ where $X(G_2)\subsetneq X(G_1)$, with identity elements $e_i=e_i^2\in G_i$ (for $1\leq i\leq 2$), there exists a group G_2' with idempotent e_2' such that $e_1>e_2'$ and $X(G_2')=X(G_2)$, and $(X(G_2),G_2)\cong(X(G_2'),G_2')$.*

Proof. Let $e_2'=(e_1e_2e_1)^k$ the unique idempotent power of $e_1e_2e_1$, with $k>1$. Then $e_2'e_1=(e_1e_2e_1)^ke_1=(e_1e_2e_1)^k=e_2'$, and similarly $e_1e_2'=e_2'$, so $e_1>e_2'$. Then $Q\cdot e_2'=Q\cdot(e_1e_2e_1)^k=X(G_2)$ as e_1 maps Q onto $X(G_1)$ and e_2 fixes every element of $X(G_2)\subsetneq X(G_1)$ but maps Q (hence $X(G_1)$) to $X(G_2)$. Obviously, e_2' acts as the identity on $X(G_2)$. Let $G_2'=e_2'G_2e_2'$. Then $g\mapsto e_2'ge_2'$ is a surjective function from G_2 to G_2'. Clearly, for all $x\in X(G_2)$, $g\in G_2$, $x\cdot e_2'ge_2'=x\cdot g$. For $g,h\in G_2$, if $e_2'ge_2'$ and $e_2'he_2'$ agree on Q, then they agree on $X(G_2)$, so then g and h do, from where $g=h$ by theorem 2.1. Thus, $g\mapsto e_2'ge_2'$ is an injective. Now G_2' has identity e_2'. As e_2' is the identity of the image of g, $(e_2'ge_2')(e_2'he_2')=e_2'ge_2'he_2'=e_2'ghe_2'$. Thus, the mapping $g\mapsto e_2'ge_2'$ is a homomorphism, and hence an isomorphism (in particular G_2' is a group). This gives an isomorphism of the permutation groups $(X(G_2),G_2)\cong(X(G_2'),G_2')$, which is the identity map on states. ■

Theorem 2.4. *Let $X(G_1)\supsetneq\ldots\supsetneq X(G_k)$ be the state sets of natural subsystems $(X(G_i),G_i)$ with idempotents $e_i^2=e_i$. Then there exist $G_i'\cong G_i$ with $X(G_i)=X(G_i')$ having identity elements e_i' for $1\leq i\leq n$, with $(X(G_i),G_i)\cong(X(G_i'),G_i')$, and, moreover, $e_1'>\cdots>e_n'$.*

Proof. Let $(X(G_1'),G_1')=(X(G_1),G_1)$ so $e_1'=e_1$. Then, inductively for $1\leq i<n$ using lemma 2.3, given $X(G_i')\supsetneq X(G_{i+1}')$, one can replace $(X(G_{i+1}),G_i)$ by isomorphic $(X(G_{i+1}'),G_{i+1}')$ with $X(G_{i+1}')=X(G_{i+1})$ and $e_i'>e_{i+1}'$ without changing e_i'. After $n-1$ steps, we obtain idempotents $e_1'>\cdots>e_n'$ and groups G_i' with the required properties. ■

Simple counterexamples show that weak control need *not* be transitive. However, we have the following general result.

Theorem 2.5. *Natural subsystems $(X_i,G_i)(1\leq i\leq k)$ whose state sets form a chain $X_1\supsetneq\ldots\supsetneq X_k$ are in weak control hierarchy where the larger sets' permutation groups (X_i,G_i) weakly control each of the smaller ones (X_j,G_j) with $j>i$.*

Proof. By theorem 2.4, we can replace the natural subsystems with equivalent ones having the same state sets and the same permutator group so that their identity elements are related in the natural partial order. As the image of e_j is $X_j=X(G_j)$ and e_j is the identity on X_j, we have that $X_i\cdot e_j=X_j$ for $j>i$ shows that the empty word and the e_j establishes weak control by $(X(G_i),G_i)$ over $(X(G_j),G_j)$. ■

Let $\mathcal{A}=(Q,\delta,A)$ be a finite automaton and consider the digraph Ξ of natural subsystems (X,G), where $X\subseteq Q$ and G is a subgroup of the characteristic monoid $M(\mathcal{A})$, with edges given by the weak control relation, i.e. there is a directed edge from (X_1,G_2) to (X_2,G_2) if and only if $(X_1,G_1)\succ_{wc}(X_2,G_2)$. We call $\Xi=\Xi(\mathcal{A})$ the *weak control hierarchy* of the automaton $\mathcal{A}$.

Lemma 2.6.

(*a*) *The weak control relation for an automaton is irreflexive and antisymmetric.*
(*b*) *The weak control hierarchy Ξ is an acyclic directed graph.*
(*c*) *The weak control relation is transitive if and only if, for all natural subsystems where (X_1,G_1) weakly controls (X_2,G_2) and (X_2,G_2) weakly controls (X_3,G_3), X_1 is $\mathcal{R}$-equivalent to some X_1' containing X_3.*

Proof. (a) By the definition of weak control, $(X_1, G_1) \succ_{wc} (X_2, G_2)$ implies $|X_1| > |X_2|$, from where irreflexivity and anti-symmetry follow. (b) A cycle in the directed graph starting and ending with (X, G) would entail $|X| > |X|$, a contradiction. (c) The condition on the existence of $X_1' \supseteq X_3$ with $X_1 \, R \, X_1'$ is obviously necessary as it must hold for (X_1, G_1) to weakly control (X_3, G_3). It is also sufficient: for if it holds then by definition of weak control there exist subsets $X, X_2' \subseteq Q$ and words $w_1, \bar{w}_1, w_2, w_3, \bar{w}_3, w_4 \in A^*$, such that $X_1 \cdot w_1 = X$, $X \cdot \bar{w}_1 = X_1$, $X \cdot w_2 = X_2$, $X_2 \cdot w_3 = X_2'$, $X_2' \cdot \bar{w}_3 = X_2$ and $X_2' \cdot w_4 = X_3$. Then by hypothesis there exists $X_1' \subseteq Q$ containing X_3 and words $w, \bar{w} \in A^*$ with $X_1 \cdot w = X_1'$ and $X_1' \cdot \bar{w} = X_1$. Hence, $X_1' \cdot \bar{w} w_1 w_2 w_3 w_4 = X_1 \cdot w_1 w_2 w_3 w_4 = X \cdot w_2 w_3 w_4 = X_2 \cdot w_3 w_4 = X_2' \cdot w_4 = X_3$, so (X_1, G_1) weakly controls (X_3, G_3). ■

Lemma 2.7. *Weak control is transitive up to $\mathcal{R}$-equivalence.*

Proof. Suppose $(X_1, G_1) \succ_{wc} (X_2, G_2) \succ_{wc} (X_3, G_3)$, then there exist X and X_2 and words $w_i, \bar{w}_i$ as in the proof of lemma 2.6(c). Without loss of generality, we may assume that $\cdot w_3 \bar{w}_3'$ and $\cdot \bar{w}_3' w_3$ are idempotent and restrict to the identity on X_2 and X_2', respectively (as in theorem 2.2), and that the image of w_3 is X_2' and the image of $\bar{w}_3$ is X_2. Let $X_3' = X_3 \cdot \bar{w}_3$. Then $X_3' \cdot w_3 = X_3$, as X_2 and X_2' are in one-to-one correspondence via the bijections $\cdot w_3$ and $\cdot \bar{w}_3'$. Clearly $X_3' \mathcal{R} X_3$. For $g \in G_3$, let $\psi(g) = \cdot \bar{w}_3' g w_3 \in (X_3')\mathbf{per}_{sgp}$. Let $G_3' = \psi(G_3)$ with identity $w_3 \bar{w}_3$. Then $(X_3, G_3) \mathcal{R} (X_3', G_3')$ and are isomorphic. As $(X_1, G_1) \mathcal{R} (X, G_1')$ with $X \cdot w_2 = X_2$, the assertion follows: $X \cdot w_2 w_3 w_4 \bar{w}_3 = X_2 \cdot w_3 w_4 \bar{w}_3 = X_2' w_4 \bar{w}_3 = X_3 \bar{w}_3 = X_3'$, and $X_3' \subsetneq X_2 \subsetneq X$, showing $(X, G_1') \succ_{wc} (X_3', G_3')$. ■

Corollary 2.8. *The weak control hierarchy modulo $\mathcal{R}$-equivalence is a partial order $\Xi/\mathcal{R}$.*

(c) Interpretations of groups, control and essential dependencies

For a given subset X of the states of a biological system, $(X, (X)\mathbf{per})$ is a permutation group. Thus, the members of $(X)\mathbf{per}$ are symmetries of the subset X. They leave X invariant. Any string of events $t \in A^*$ representing one of the members of $(X)\mathbf{per}$ will preserve X. That is, t gives a bijection of the states in X to themselves, and any sequence of members of $(X)\mathbf{per}$ will also map X to itself in a one-to-one onto manner. The structure $(X, (X)\mathbf{per})$ is thus a 'natural system' or 'pool of reversibility'. Non-trivial $(X)\mathbf{per}$ groups appear in the algebraic decomposition of the automata into simpler components (Krohn–Rhodes theorem), which shows how to synthesize the given automaton using a hierarchical arrangement of components—in the holonomy theorem [9, ch. 3] these can be obtained from the weak control hierarchy (theorem 6.1).[11] The irreducible components of these groups are the 'atomic pieces' or elementary building blocks constituting the entire structure. The Krohn–Rhodes decomposition can be viewed as providing a generalized coordinate system (analogous to the decimal or binary expansion) that models how computation in the biological system occurs).[12]

Now suppose a subset of $Q = States(\mathcal{A})$ is partitioned into distinct configuration classes with representatives $c_1, \ldots, c_m$ $(m \leq |Q|)$. For example, we may take the c_i to be all the states of Q (so $m = |Q|$), or, alternatively, the c_i may be representatives of orbit trajectories in Q under subgroups of $S(\mathcal{A})$, i.e. dynamically stable orbits in different natural subsystems. Suppose a word $t \in A^*$ maps each c_i to $c_i \cdot t$ in the partition class of some $c_{t(i)}$ and that this map is well defined on partition classes. Then $\mathcal{A}$ has *absolutely complete control* or *total control* over the configuration classes if input sequences can perform arbitrary operations on the classes. That is, every function $f : \{1, \ldots, m\} \to \{1, \ldots, m\}$ is realized by some input sequence $t \in A^*$. The collection of all mappings on the partition classes has maximal complexity with this property.

Note: mere transitivity on states is *not* an interesting alternative. That is, if for every c_i and c_j in Q there exists a $t_{i,j} \in A^*$ with $c_i \cdot t_{i,j} = c_j$. For example, if the characteristic monoid of $\mathcal{A}$ contains just all constants then this condition is satisfied, but then there is no dependency of the resulting state on the current state. So such systems have zero complexity.

[11] The groups occurring are non-trivial quotient groups of the $(X)\mathbf{per}$ groups for X a subset of the form $X = Q \cdot t$, $t \in A^*$.

[12] The remainder of this section on different notions of control and essential dependencies is adapted from [11, pp. 49–50, 188–191] by John Rhodes; however, the notion of weak control and its application is new.

Let G_1 and G_2 be two subgroups of S, with $e_k^2 = e_k$ the identity of G_k, for $k = 1, 2$. We define $e_1 > e_2$ if and only if $e_1 \neq e_2$ and $e_1 e_2 = e_2 e_1 = e_2$. It is easy to show that $>$ is a transitive relation on the collection of idempotents of S. Also denote $X(G_k)$ by X_k. Then it is not difficult to show that $e_1 > e_2$ implies $X_1 \supsetneq X_2$. Now a chain of *internal dependencies* is given by $e_1^2 = e_1 > e_2^2 = e_2 > \cdots > e_n^2 = e_n$ and non-trivial subgroups G_k with identities e_k so that

$$X_1 \supsetneq X_2 \supsetneq \cdots \supsetneq X_n \neq \emptyset.$$

But we must also require that each step $X_k \supsetneq X_{k+1}$ be 'essential', so we precisely define this next.

Given subgroups G_k, G_{k+1} of S with identities $e_k^2 = e_k$ and $e_{k+1}^2 = e_{k+1}$, respectively, and $X_k = X(G_k)$, $X_{k+1} = X(G_{k+1})$ and $e_k > e_{k+1}$ so $X_k \supsetneq X_{k+1} \neq \emptyset$, define $\mathcal{X}_{k+1} = \{X_{k+1} \cdot g_k : g_k \in G_k\}$. Thus, $\mathcal{X}_{k+1}$ is all those subsets of X_k given by letting (X_k, G_k) act or 'knock about' $X_{k+1} \subseteq X_k$. Now let $\mathcal{I}_{k+1} = \{f^2 = f \in S : X \cdot f = X_k \cdot f \in \mathcal{X}_{k+1}\}$. Thus, $\mathcal{I}_{k+1}$ is all those idempotents of S whose range is some translation of X_{k+1} under some member of G_k. Let $\langle \mathcal{I}_{k+1} \rangle$ be the subsemigroup of S generated by $\mathcal{I}_{k+1}$. Then by definition $X_k \supsetneq X_{k+1}$ is an *essential dependency* if and only if

$$\langle \mathcal{I}_{k+1} \rangle \cap (X_{k+1})\mathbf{per} \supseteq G_{k+1} \neq \{e_{k+1}\}.$$

Then we have the following.

Theorem 2.9 (complexity lower bounds (Rhodes–Tilson [11])). *Let k be the length of a longest chain of essential dependencies of an experimental system E described by an automaton $\mathcal{A}$ with characteristic semigroup $S(\mathcal{A})$. Then the number of non-trivial groups which must occur at different levels of any linear cascade decomposition for $\mathcal{A}$ (i.e. the Krohn–Rhodes complexity of this model of E) is greater than or equal to k.*

Let the *spectrum* of $\mathcal{A} = (Q, A, \delta)$ be defined as $\mathbf{spectrum}(Q, A) = \{k > 1 : k = |Q \cdot t|, t \in A^+\}$, i.e. the set of cardinalities of images of Q under some input sequence.

The control of $\mathcal{A}$ is *above the bare minimum* if there is a sequence of input words $t_1, \ldots t_{n-2} \in A^*$ such that $|A \cdot t_1 \ldots t_i| = n - i$. Thus, each successive t_i removes one state from the possible configurations. This property implies that spectrum($\mathcal{A}$) is then the same as the spectrum of all mappings on Q. Rhodes [11, ch. 6, part II] gives a mathematical argument that evolved systems such as the cell must have control above the bare minimum. (It turns out that simpler systems such as the Krebs cycle do too.)

One idea of *locally good control* of a collection of states $\{q_1, \ldots, q_p\} = Q$ is that a group of permutations of Q acts transitively on Q (i.e. given $q_1, q_2 \in Q$, there exists an input $t \in A^*$ yielding $\pi : Q \to Q$ so that $Q \cdot \pi = Q$ and $q_1 \cdot \pi = q_2$) and also for each $q \in Q$ there exists an input t_q such that $q_j \cdot t_q = q$ for all $j = 1, \ldots, p$. The semigroup $S = S(\mathcal{A})$ under locally good control contains a transitive permutation group G on Q together with all the constant maps (or resets) on Q. The constant map control is only *locally* good since whenever a constant map is used (a reset) there is no way for any map but a constant map to occur again (as constants$\cdot S \subseteq S \cdot$constants $\subseteq$ constants). Clearly, how effective a good local control is is inversely proportional to the number of states. Of course, if $\mathcal{A}$ is in a cascade with other locally good control units, then if each unit just controls a small number of states, the global control can be quite good, so heuristically high complexity, relative to the total states, means many locally good control units in a cascade, while low complexity relative to the total states implies few locally good control units in cascade. In the first case, each local unit has few states to manage so the local control is good everywhere, so the global control is good. Similarly in the second case, the global control is bad, as locally good control units have so many states that their global control is bad.

Using the precise definition of essential dependencies, we can make the above heuristic argument precise.

Let $X_1 \supsetneq X_2 \supsetneq \cdots \supsetneq X_n$ be a chain of essential internal dependencies. Consider the step from k to $k+1$, $X_k \supsetneq X_{k+1}$. Using the notation in the definition of essential dependency, we have G_k a transitive group on $\mathcal{X}_{k+1} = \{X_{k+1} \cdot g_k : g_k \in G_k\}$. Using some standard semigroup theory (details of which we omit), $\mathcal{I}_{k+1}$ can be shown to have the following property. For each $X'_{k+1} \in \mathcal{X}_{k+1}$, there exists $f \in \mathcal{I}_{k+1}$ so that for *all* $X^*_{k+1} \in \mathcal{I}_{k+1}$ either $X^*_{k+1} \cdot f \equiv f(X^*_{k+1}) = X'_{k+1}$ or $|X^*_{k+1} \cdot f| \equiv |f(X^*_{k+1})| <$

$|X'_{k+1}| = q_{k+1}$, where q_{k+1} is the number of elements in each member of $\mathcal{X}_{k+1}$. Also, for at least one element of $\mathcal{I}_{k+1}$, the first condition must hold. In other words, f acts like the constant map sending every element of $\mathcal{X}_{k+1}$ to X'_{k+1} or to 'zero' and it sends at least one element to X'_{k+1}. Thus, $G_k \cup \mathcal{I}_{k+1}$ acting on $\mathcal{X}_{k+1}$ has locally good control (where our previous definition has been slightly generalized to allow 0-constant maps).[13]

Thus, each essential dependency $X_k \supsetneq X_{k+1}$ leads to locally good control. If n is nearly as large as the number of states $|Q|$, by a strengthened definition of essential dependency giving tighter bounds (similar to the above but going into precise details about how many zeros are allowed in the 0-constant maps and not allowing for many zeros), the number of locally good control units is large, so each has only a few states to govern, so the global control is good.

Based on the weak control hierarchy, natural subsystems give rise in theorem 6.1 to control which is locally good and yield a coordinate system for the model being analysed.

3. Natural subsystems and weak control by permutation groups in biochemical and biological examples: models of the lac operon in *Escherichia coli*, p53–mdm2 and the Krebs cycle

Weak control hierarchies and natural subsystems are presented here for finite automata models of various biochemical and biological systems. These illustrate the concepts and definitions presented in §§1 and 2 in concrete systems biology models. Transformation semigroups, natural subsystems and weak control structures were computed directly from these definitions for the presented automata models using our open-source computer algebra system SGPDEC for analysis and hierarchical synthesis and decomposition of finite automata, transformation semigroups and permutation groups [16,17].

In all p53–mdm2 biological examples in this article and for the lac operon automaton, weak control is transitive.[14] The Krebs cyclic example has non-transitive occurrences of weak control, but, by corollary 2.8, $\Xi/\mathcal{R}$ is a transitive hierarchy (i.e. partial order) in every case.

We show that permutation groups in the algebraic structures associated to biological models can be non-trivial, and moreover that finite SNAGs occur. The latter is of special interest as they have been shown to be capable of finitary universal computation (§4).

(a) Permutation groups in the lac operon in *Escherichia coli*

Now we study permutation groups and relations among them in a biological example. *Escherichia coli*, the 'workhorse' bacterium of microbiology, can metabolize glucose and lactose also, but will facultatively switch to glucose when it is available: lactose is metabolized only when glucose is absent and lactose is available. In all other cases, the expression of the structural genes for the enzymes of lactose metabolism is suppressed. This genetic regulatory mechanism, the lac operon, is a canonical example of prokaryotic gene regulation (e.g. [28,29]). We use a finite-state automaton description of this gene regulation mechanism in order to examine natural subsystems and algebraic structure associated to the lac operon mechanism. This is a very simple Boolean network model of the lac operon mechanism in *E. coli*, originally suggested by Stuart Kauffman [4].[15] Its states are configurations given by four Boolean variables, as shown in table 1. Time evolution of the lac operon model depends on three kinds of basic event: t, the passage of time (a discrete 'clock tick'); $L0$, allowing lactose, if present, to deplete; $L1$, the introduction or

[13] A function is said to be a *0-constant map* if it takes a constant value except possibly at some points where its value is undefined or outside the range of interest and so is reported as 'zero'. In the example here, the application of f induces a 0-constant map from $\mathcal{X}_{k+1}$ to itself. Such maps arise frequently in semigroup theory (e.g. in relation to the wreath product of permutation groups augmented with constant maps—implicitly related to the case discussed here—or in the global action of a semigroup on the local Rees–Sushkevych coordinatizations).

[14] This was checked exhaustively by direct computation for all the examples on a laptop running SGPDEC. This took over one week for the larger p53–mdm2 network to do directly, but less than one minute to check using the condition of lemma 2.6.

[15] We are grateful to George F. Estabrook for providing us with the details of Kauffman's model.

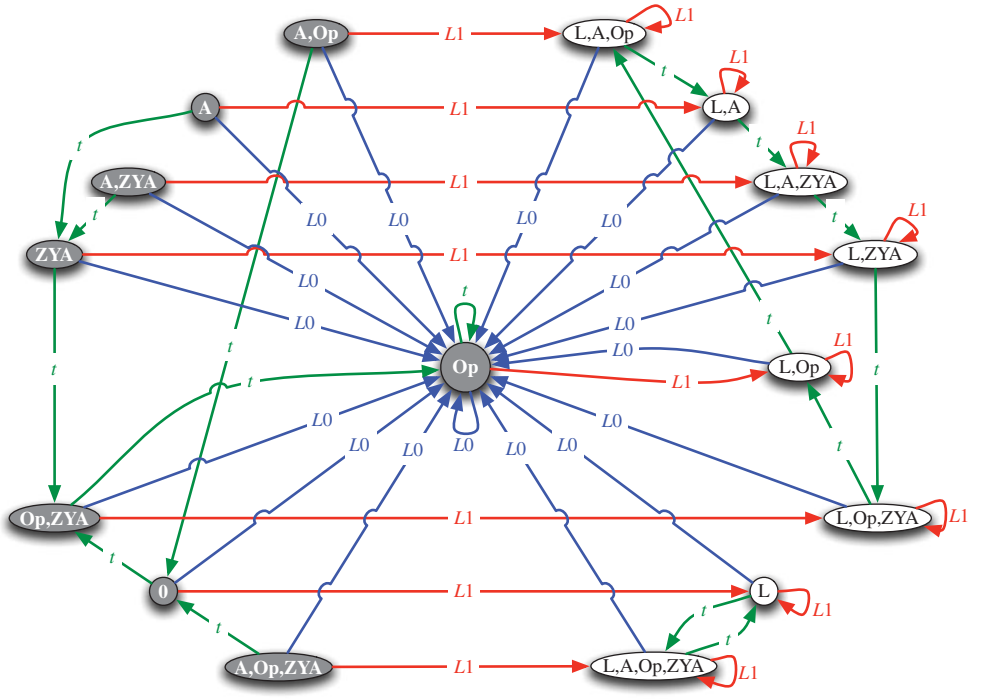

Figure 2. Automaton derived from Boolean network model of the lac operon mechanism in *E. coli*. Transition *L*1 denotes the introduction of lactose, *L*0 the depletion of lactose and *t* labels transitions due to the passage of time. Nodes in the diagram are states of the network and are thus given by subsets of {**L**, **A**, **Op**, **ZYA**}, consisting of the active/present biochemical components. Here state 0 denotes the empty set where none of these components is present/active. The transformations of the system are given by all finite sequences of transitions from *L*1, *L*0 and *t*. Light nodes denote states where lactose is present, whereas dark nodes denote states without lactose. (Based on Kauffman [4]; adapted from Egri-Nagy & Nehaniv [30].)

maintenance of lactose. The clock tick is assumed to occur on a short but unspecified time scale, where there is no change in whether lactose is present or not above a threshold value. These three basic events thus comprise an external input alphabet to the 'event-driven' finite-state automaton model, whose state-transition diagram is given in figure 2.

Natural subsystems are generally not immediately evident from either Boolean networks or finite-state automaton descriptions. Knowing a model's generators or transition diagram fully describes it, but this gives almost no insight into its structure and no higher level insight into how it functions. This is analogous to the fact that writing down a system of differential equations does not by itself yield understanding of a system's orbit and stability structure, local and global dynamics, etc.

Studying the transformation semigroup of this lac operon model reveals exactly two non-trivial natural subsystems. These structures were calculated in a few seconds using our computational algebra system SGPDEC [16–18] together with the weak control relationship between the two. Figure 3 shows the weak control structure of this model of the lac operon with two permutator groups inside the semigroup of the automaton for this model. Both consist of cyclic group C_6 of order 6 acting on subsets of the state space: there is a natural subsystem (X, C_6) with X consisting of the nine states Op, L, L ZYA, L Op, L Op ZYA, L A, L A ZYA, L A Op and L A Op ZYA. These are Op and the states with lactose. The set X is permuted in three orbits by the clock tick t: one orbit consists of two states L and L A Op ZYA that are transposed by t, the second of six states of the Boolean network L Op, L A Op, L A, L A ZYA, L ZYA and L A Op ZYA cyclically permuted in that order, while a final orbit consists of Op alone as the repressor molecule is continually produced but activates nothing else in the absence of lactose. There is a

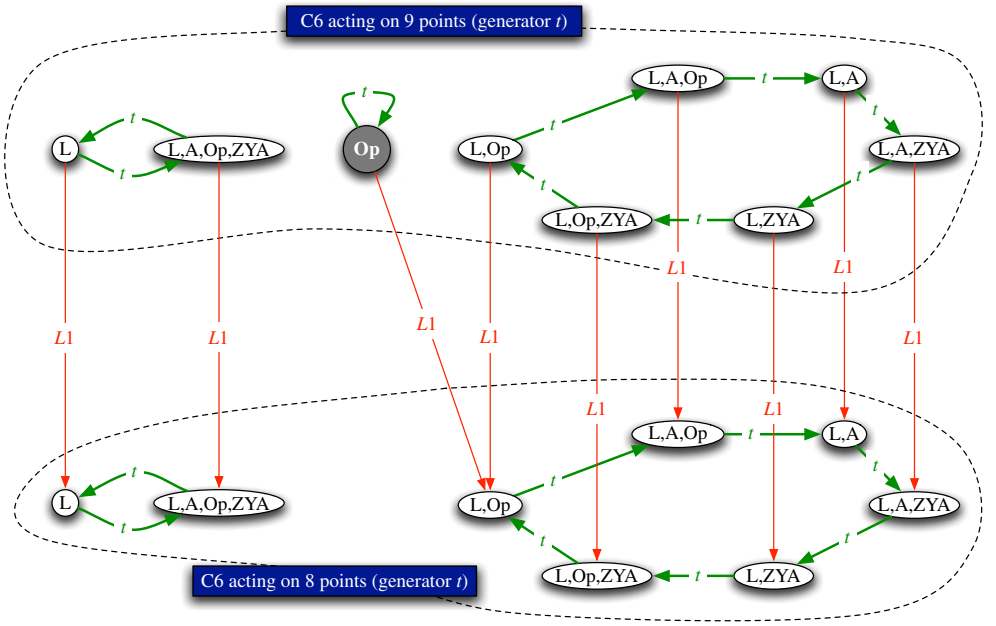

Figure 3. Weak control hierarchy for the lac operon Boolean network model.

natural subsystem (Y, C_6) where Y consists of the eight states L, L ZYA, L Op, L Op ZYA, L A, L A ZYA, L A Op and L A Op ZYA, where lactose is present (states on the left in table 1 and also appearing as the white nodes in figure 2). The permutation group with nine states $X = Y \cup \{Op\}$ includes these eight plus the Op state, where the operon repressor molecule is present but no lactose or allolactose is present and also protein biosynthesis according to genes ZYA are not active. This permutation group weakly controls and is a superstructure of the permutation group with the eight states: adding lactose (input $L1$) maps the Op state to L Op, where both lactose and the repressor molecule are present (one of the eight states), but fixes the remaining eight states. These eight states are the image of the idempotent $L1$, and the group has identity $L1$. Formally, by weak control words λ and $L1$: $X \equiv X$ via the empty word λ and $Y = X \cdot L1 \subsetneq X$. The permutations of the lower group are generated by the event sequence $L1\ t\ L1$ (or $L1\ t$).

In the ensemble viewpoint, given a direct product of lac automata each in one of the states in X, then applying an element of the group $(X)\mathbf{per}$ leaves the set of states fixed although it permutes them non-trivially pointwise. From the entire possible set of states of the Boolean network after the occurrence of three or more clock ticks, the possible states are exactly X, the lactose-containing states and Op. This is why X and not some other subset occurs—it is the first natural system reached after a transient. If lactose is then added, $L1$, the possible set of states is Y, as lactose takes Op to Op L, and the lactose-containing states are not affected. These are the only non-trivial natural subsystems in this example and they are defined algebraically by the notion of the permutator group. No other subset of possible states that can be reached by any event sequence starting from the full set of possible states is non-trivially permuted by any event sequence. Note that every sequence consisting of only $L1$'s and t's will permute Y, and $(Y)\mathbf{per}_{str} = \{L1, t\}^*$. Every sequence consisting only of t's permutes X, and $(X)\mathbf{per}_{str} = t^*$. These two infinite regular languages describe exactly six cyclic permutations of Y and X, respectively, where the former permutations are restrictions of the latter $C_6 \cong (X)\mathbf{per} \cong (Y)\mathbf{per}$.[16] In this case, the 'weak control'

[16]We remark that, while the event sequences t and $L1\ t$ each generate the permutator group $(Y)\mathbf{per}$, the subsemigroups of the semigroup of the lac operon automaton that they generate contain different subgroups isomorphic to C_6. This can be seen from the fact that elements of the former fix the state Op while the latter do not. Generally, as it is defined by restriction, the permutator group may be realized by many different subgroups of the semigroup of the automaton, as is the case also for the other examples in this paper (see also [31]). Here the monogenic semigroups $\langle t \rangle$ and $\langle L1\ t \rangle$ inside $(X)\mathbf{per}_{sgp}$ contain disjoint subgroups, each isomorphic to C_6.

is rather 'strong', in that every permutator element of the lower set Y is realized as the restriction of a permutator element of the larger natural system (X, C_6).[17] This will not in general be the case, as we shall see for the p53–mdm2 genetic regulatory and Krebs cycle models.

(b) Krebs cycle

The major portion of a eukaryotic cell's energy requirements are met by the breakdown of sugars, with glucose as the principal source. The complete burning of 1 mole of glucose is given by

$$C_6H_{12}O_6 + 6O_2 \rightarrow 6CO_2 + 6H_2O + 673\,\text{kcal}.$$

Glucose is converted to CO_2 and water by a process involving nearly 30 different successive steps that gradually release energy. This reaction network was a big evolutionary leap, as anaerobic glycolysis is much less efficient at extracting energy from sugar [32]. Any standard textbook in biochemistry contains the details (e.g. [33]). First, a glucose molecule is broken down into two pyruvic acid molecules with the Embden–Meyerhof pathway (*fermentation*). Second, pyruvic acid is completely oxidized to CO_2 and water via a series of acidic intermediates called the *citric acid cycle* or the *Krebs cycle*. In the course of the cycle, oxaloacetic acid acquires an acetyl group from pyruvic acid to form citric acid, which then undergoes a number of transformations that eventually yield oxaloacetic acid that is ready to take up a new acetyl group from pyruvate. The Krebs cycle releases a great amount of energy (in the form of ATP), requires oxygen and is sometimes called *respiration* or *glucose oxidation*.[18] Now that we have said what the Krebs cycle does, we describe how this is achieved (figure 4*a*).[19]

In figure 4, NAD, CoASH, NADP, GDP and FAD are the coenzymes; pyruvic acid, acetyl CoA, citric acid, etc., are the substrates. To obtain a discrete dynamical system (finite-state automaton) model of the Krebs cycle, we must define the state space, the inputs and the action. With reference to figure 4*a*, we define the set of inputs to be {NAD + CoASH, CoASH, NADP, GDP, FAD, NAD}. For ease in defining the states, we enumerate the substrates: 1_1 for pyruvic acid, 2_1 for acetyl CoA, 3_1 for citric acid, 4_1 for isocitric acid, 5_1 for oxalosuccinic acid, 6_1 for α-ketoglutaric acid, 7_1 for succinyl CoA, 8_1 for succinic acid, 9_1 for fumaric acid, 10_1 for L-malic acid, 11_1 for oxaloacetic acid. We let all the relevant enzymes (listed in figure 4*a*), the ADP, P_i, O_2, etc., i.e. everything required for the reactions but not listed in the inputs or states, to be considered in the 'soup' and present in the amount required and at the time and place required.

Figure 4*b* shows the reaction graph of the Krebs cycle in which substrates appear as states and arrows are labelled by coenzymes required for the transition. In cases where the transition from one substrate to another runs to completion given the constitution of the soup, the substrates in question are shown in the same state as in the model (e.g. citric and isocitric acid). We denote this model of the Krebs cycle as K_1. Now by reaction ③ of figure 4*a*, citric acid is converted to its isomer (chemical composition the same but structure or shape differing) isocitric acid by way of the enzyme aconitase, which functions by removing a molecule of water from the citric

[17] A Krohn–Rhodes hierarchical decomposition using a cascade of transients as well as the subgroup structure of this model using a different analysis is presented in [30]. As the relationship between the two permutator groups is so close, the holonomy decomposition can be carried out using only one copy of C_6, showing that the model has complexity 1, i.e. it requires only one level of (generalized) arithmetic given by a non-trivial group.

[18] The explanation of the Krebs cycle and its discrete dynamical modelling using the reaction graph of figure 4*b* closely follows our previous work [11,34]. The extended Krebs cycle two-molecule model (figure 5) and analyses of its natural subsystem permutator group structure and weak control hierarchy are new.

[19] We make some very brief explanatory comments. *Enzymes* are biological catalysts. Catalysts modify the speed of a chemical reaction without being used up or appearing as one of the reaction products. Certain enzymes are solely proteins (e.g. the digestive enzymes pepsin and trypsin). Other enzymes consist of a non-protein part plus the protein and are therefore conjugated proteins. If the non-protein portion is an organic part and is readily separated from the enzyme, this fragment is called a *coenzyme*. It is called a *prosthetic group* if it is firmly attached to the protein portion of the enzyme. The amount (by weight) of coenzymes in cells is very small. It is very important that the cell requires a different enzyme for practically every reaction it carries out. No enzyme, no reaction. This specificity is not entirely absolute (there are enzymes which will catalyse the hydrolysis of an ester R–COO–R′ + $H_2O \rightleftarrows$ R–COOH + R′–OH without much regard for R, R′ if there is an ester linkage between them) but for the majority of enzymes absolute specificity is the rule.

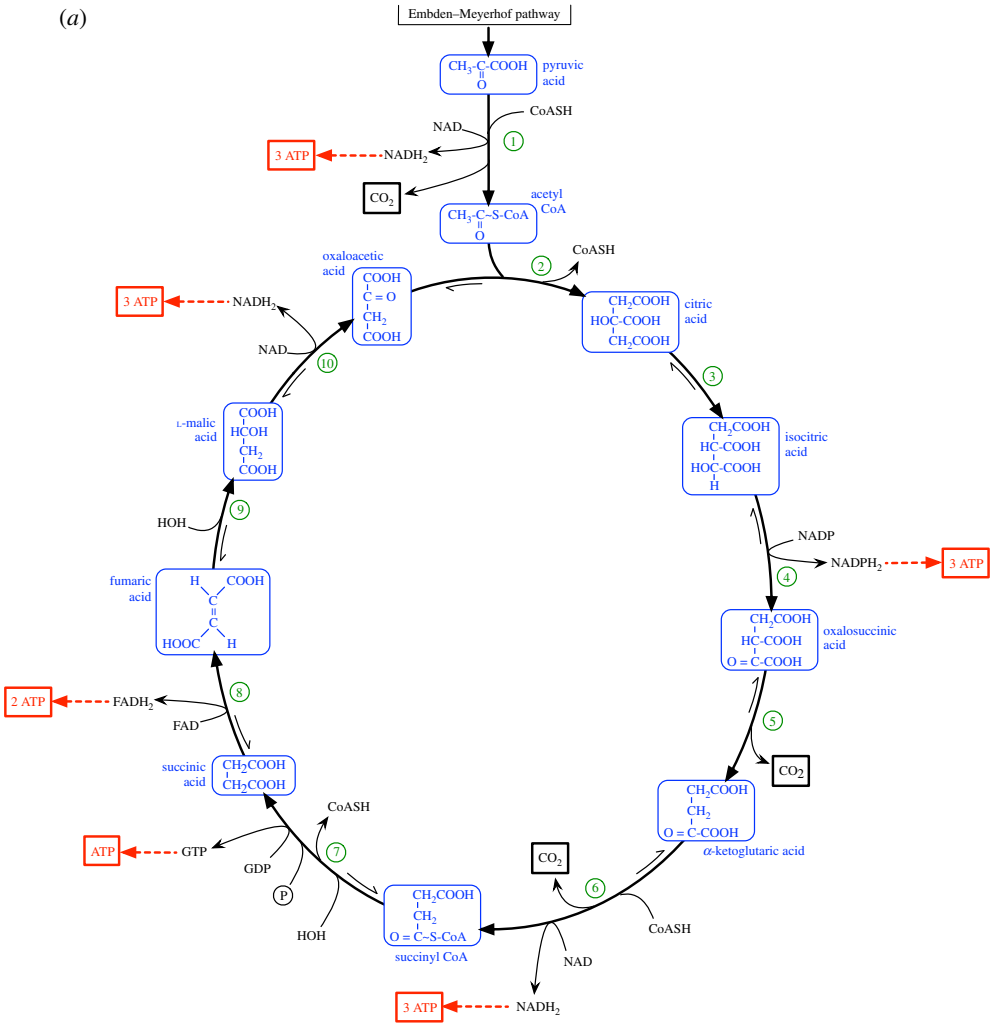

Figure 4. Embden–Meyerhof pathway and Krebs cycle. (*a*) Enzymes involved in the reactions as numbered in the diagram are: (1) pyruvate oxidase, (2) citrate synthase, (3) aconitase, (4) and (5) isocitrate dehydrogenase, (6) α-ketoglutarate dehydrogenase, (7) succinyl thiokinase, (8) succinate dehydrogenase, (9) fumarase and (10) malate dehydrogenase. (*b*) Reaction graph and finite-state automaton model K_1 of the Krebs cycle (see text). Substrates correspond to states of the automaton and if acted on by the relevant coenzyme are transformed, e.g. pyruvic acid (x_1) interacting with NAD and CoASH (a_1) is transformed into acetyl CoA (x_2). This is denoted $\delta(x_1, a_1) = x_2$. Note that if the coenzymes have no effect on a substrate then no arrow is shown, e.g. $\delta(x_8, a_4) = x_8$ as oxaloacetic acid (x_8) is not transformed by interaction with GDP (a_4). The presentation is 'event-based' or 'perturbation-based' and time does not appear explicitly in the model. Reactions shown run to completion, and thus all basic transitions are idempotent (i.e. $\delta(x, a) = \delta(\delta(x, a), a)$ always holds in a reaction graph). (Models from Rhodes [11, ch. 6, Part I].)

acid and then replacing the water molecule isomerically. However, as no cofactor (i.e. input) is required for this (just members of the 'soup'), we identify 3_1 and 4_1. Similarly in reaction ⑤ of figure 4*a*, oxalosuccinic acid is decarboxylated to give α-ketoglutaric acid, so we identify 5_1 and 6_1. Similarly, we also identify 9_1 and 10_1. Thus, the states of our model K_1 are formally $X_1 = \{1_1, 2_1, \{3_1, 4_1\}, \{5_1, 6_1\}, 7_1, 8_1, \{9_1, 10_1\}, 11_1\}$.

Now to be completely formal, the first model K_1 of the Krebs cycle is the automaton derived from the reaction graph in figure 4*b*. The transition function of the automaton of K_1 is defined

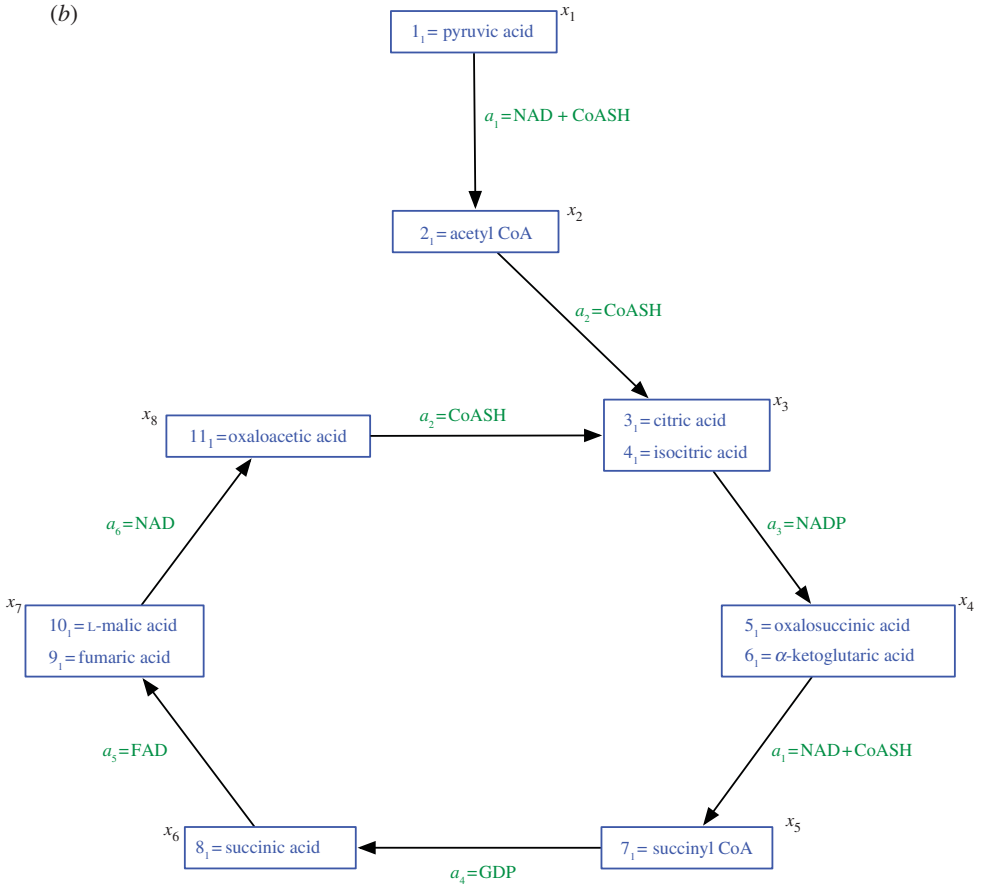

Figure 4. (*Continued.*)

in a natural way by letting, for example, $\delta_1(10_1, \mathrm{NAD}) = 11_1$. The reaction network shows how the encounters with coenzymes in the presence of appropriate enzymes yields chemical transformation. For example, $\delta_1(x_5, a_4) = x_6$ as the reaction graph shows

$$x_5 = \text{succinyl CoA} \overset{a_4=\mathrm{GDP}}{\longrightarrow} \text{succinic acid} = x_6,$$

whereas

$$\delta_1(x_1, a_4) = x_1$$

since, according to the model, there is no arrow labelled GDP (a_4) leaving state x_1. So, if GDP does come into contact with pyruvic acid (x_1), it leaves this substrate unchanged. As the state transition function δ_1 is fully and uniquely defined for all possible state and input symbol combinations, we have a complete knowledge of the automaton. However, this knowledge does not imply immediate understanding of the automaton, as it is given in a *recursive form* (just as in the case of differential equations), i.e. in order to gain insight about the computations carried out by the automaton, we need to start from an initial state, apply an input symbol, arrive in another state, then apply another transformation, and so on. Following the definitions of §1, the transition function extends to input words, i.e. sequences of input symbols: for the empty word $\delta(x, \lambda) = x$, and for arbitrary words $u, v \in A^*$ (finite input sequences from the input alphabet A), we have $\delta(x, uv) = \delta(\delta(x, u), v)$. This justifies the practice that, for notational simplicity, we sometimes write $x \cdot u$ instead of $\delta(x, u)$ for any $x \in Q$ and $u \in A^*$.

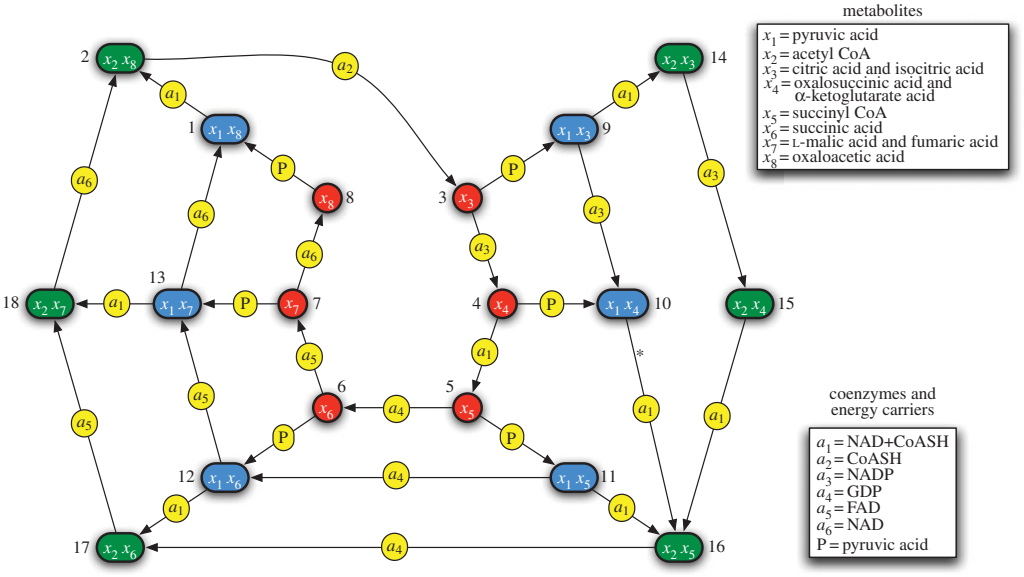

Figure 5. Krebs cycle two-molecule automaton model. States are given by one or two metabolites, while transitions are given by coenzymes and energy carriers (including input of pyruvic acid (P)) using the same notation as in figure 4*b*. Only transitions which change the state are shown, e.g. $(x_2\ x_8)\cdot a_2 = x_3$, as acetyl CoA and oxaloacetic acid in the presence of CoASH (with appropriate enzymes) are transformed to citric acid (then isocitric acid), whereas $(x_2\ x_8)\cdot a_3 = (x_2\ x_8)$ as acetyl CoA and oxaloacetic acid are unchanged in the presence of NADP (and ambient enzymes). This version of the Krebs cycle shows explicitly how its end product oxaloacetic acid together with a new pyruvic acid that has been fermented (by splitting glucose) to acetyl CoA then combines to form citric acid, thus closing the cycle. The state transition marked $*$ involves two steps as $(x_1\ x_4)$ acted on by a_1 becomes $(x_2\ x_5)$ as the presence of a_1 transforms x_1 to x_2 and x_4 to x_5 (either simultaneously or in either possible order).

For example, in figure 4*b*, $\delta(x_1, a_6a_1a_2) = x_1\cdot a_6a_1a_2 = ((x_1\cdot a_6)\cdot a_1)\cdot a_2 = x_3$ as the reader can immediately verify.

For mathematical convenience, given $x, y \in Q = States(\mathcal{A})$ for any automaton $\mathcal{A}$, we define an *elementary collapsing* $T_{xy}\colon Q \to Q$, by $(x)T_{xy} = y$ and $(z)T_{xy} = z$ for all $z \in Q$ with $z \neq x$.

In the model here, states $X_1 = \{1_1 = x_1,\ 2_1 = x_2,\ \{3_1, 4_1\} = x_3,\ \{5_1, 6_1\} = x_4,\ 7_1 = x_5,\ 8_1 = x_6,\ \{9_1, 10_1\} = x_7,\ 11_1 = x_8\}$; inputs $A_1 = \{\text{NAD} + \text{CoASH} = a_1, \text{CoASH} = a_2, \text{NADP} = a_3, \text{GDP} = a_4, \text{FAD} = a_5, \text{NAD} = a_6\}$ and the action $\delta_1\colon X_1 \times A_1 \to X_1$ is defined by $\delta_1(x, a_1) = (x)T_{x_1x_2}T_{x_4x_5}$, $\delta_1(x, a_2) = (x)T_{x_2x_3}T_{x_8x_3}$, $\delta_1(x, a_3) = (x)T_{x_3x_4}$, $\delta_1(x, a_4) = (x)T_{x_5x_6}$, $\delta_1(x, a_5) = (x)T_{x_6x_7}$, and $\delta_1(x, a_6) = (x)T_{x_7x_8}$.[20] Thus, our first model of the Krebs cycle yields $K_1 = (X_1, A_1, \delta_1)$ with characteristic semigroup $S(K_1)$ and transformation semigroup $(X_1, S(K_1))$. In [34], we described a hierarchical decomposition of this automaton having cyclic permutation groups as non-trivial computing levels.

In order to model that the cycle closes with oxaloacetic acid acquiring an acetyl group from pyruvic acid to form citric acid, we derive from model K_1 another reaction graph whose states may consist of more than one molecule at a time. Truncating this automaton at two molecules to allow tracking of an additional molecule of pyruvic acid or acetyl CoA (which results from under-fermentation) results in the finite automaton shown in figure 5.

Computing the natural subsystems shows that all of them are cyclic groups, namely the cyclic modulo n groups for $2 \leq n \leq 6$ of the form $(X, (X)\mathbf{per})$ where X is a set of n states and $(X)\mathbf{per} \cong C_n$. There are 51 equivalent, hence isomorphic, permutation C_6 groups cyclically permuting on different sets of six states in the transformation semigroup of the

[20]When we compose elementary collapsings, $(x)T_{x_1x_2}T_{x_4x_5}$ equals x if $x \neq x_1, x_4$ and $(x_1)T_{x_1x_2}T_{x_4x_5} = x_2$ and $(x_4)T_{x_1x_2}T_{x_4x_5} = x_5$. $T_{x_1x_2}$ and $T_{x_4x_5}$ both occur because a_1 occurs from x_1 to x_2 and from x_4 to x_5.

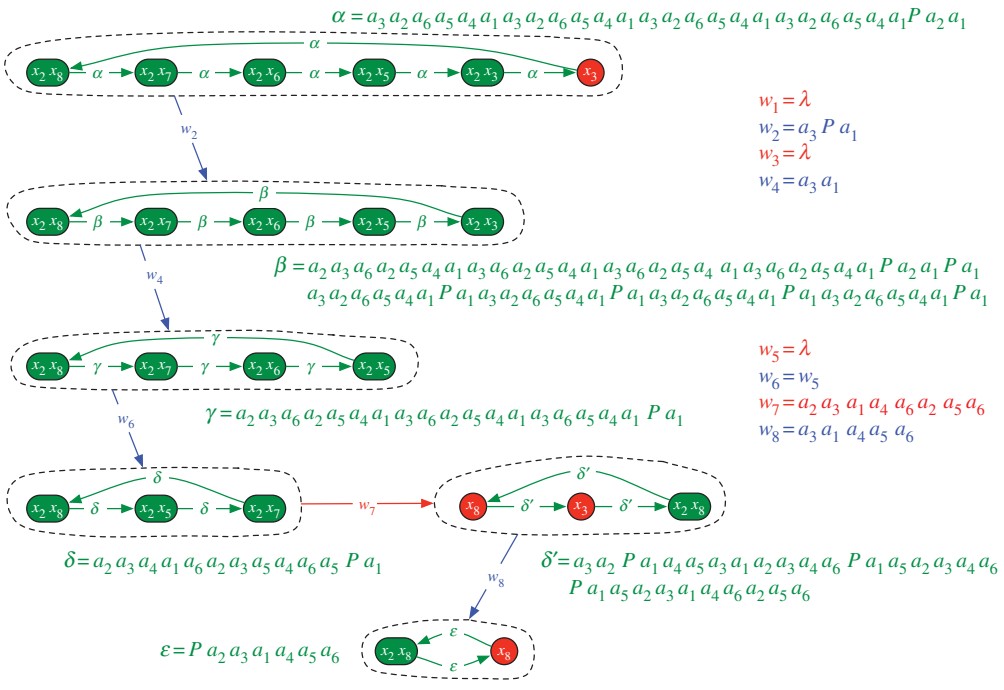

Figure 6. Weak control hierarchy for Krebs cycle two-molecule automaton model.

automaton, 274 equivalent permutation C_5 groups on five states, 375 equivalent permutation C_4 groups on four states, 248 equivalent permutation C_3 groups on three states and 95 equivalent permutation C_2 groups on two states.[21,22] Weak control is transitive in this model and illustrated in figure 6. For instance, the set with three states $x_2\ x_8$, $x_2\ x_5$ and $x_2\ x_7$ is cyclically permuted by the word $\delta = a_2a_3a_4a_1a_6a_2a_3a_5a_4a_6a_5Pa_1$, i.e. a sequence of encounters with the corresponding chemical species (and infinitely many other words). This three-state set consists of acetyl CoA together with exactly one of succinyl CoA, oxaloacetic acid or fumaric/L-malic acid. This natural subsystem is isomorphic to another three-state set consisting of oxaloacetic acid, citric/isocitric acid and pyruvic acid with oxaloacetic acid, which is cyclically permuted by the word $\delta' = a_3a_2Pa_1a_4a_5a_3a_1a_2a_3a_4a_6Pa_1a_5a_2a_3a_4a_6$. The isomorphism is determined by words $w_7 = a_2a_3a_1a_4a_6a_2a_5a_6a_3a_1a_2a_3a_4a_5Pa_1$ mapping the first three-state set onto the second one, and $\bar{w}_7 = a_3a_1a_2a_3a_4a_5Pa_1$ taking the second set back onto the first. Applying $a_3a_1a_4a_5a_6$ takes the second three-state set to its two-state subset oxaloacetic acid and pyruvic acid with oxaloacetic acid on which the word $\epsilon = Pa_2a_3a_1a_4a_5a_6$ acts as a transposition: $x_8 \cdot Pa_2a_3a_1a_4a_5a_6 = x_1x_8 \cdot a_2a_3a_1a_4a_5a_6 = x_1x_8 \cdot a_1a_4a_5a_6 = x_2x_8 \cdot a_4a_5a_6 = x_2x_8$, and also $x_2\ x_8 \cdot Pa_2a_3a_1a_4a_5a_6 = x_3 \cdot a_3a_1a_4a_5a_6 = x_4 \cdot a_1a_4a_5a_6 = x_5 \cdot a_4a_5a_6 = x_6 \cdot a_5a_6 = x_7 \cdot a_6 = x_8$. By definition then, the representative order 3 cyclic permutation group on $x_2\ x_8$, $x_2\ x_5$, $x_2\ x_7$ weakly controls the order 2 cyclic group on $x_2\ x_8$ and x_8. In this case, the first three-state subset acetyl CoA together with exactly one of citric/isocitric acid, oxaloacetic acid or fumaric/L-malic acid does in fact weakly control each of its two-element subsets, e.g. via the action of a_6 this three-state subset maps onto the pair $x_2\ x_8$ of acetyl CoA and oxaloacetic acid together and $x_2\ x_5$ of acetyl CoA and succinyl CoA. While no element of the three-element group transposes the latter pair $\{x_2x_8, x_2x_5\}$,

[21] The list of member states of each of these natural subsystems is given in supplementary material available from the authors on request.

[22] It follows by the holonomy theorem (e.g. [9, ch. 3]) or theorem 6.1 below that the automaton can be decomposed by a cascade alternating one copy of each cyclic permutation group C_n $(6 \geq n \geq 2)$ with series-parallel banks of flip-flops (which have no non-trivial subgroups).

the orbit of this pair under the three-element cycles through two-element subsets of the three-state natural subsystem. Similarly, each n-element natural subsystem on the left-hand side of figure 6 cycles the proper subset below it through $n-1$ of its proper subsets. But no element of the group of the n-state natural subsystem gives a non-trivial permutation of the $n-1$-element, . . ., or two-element natural subsystem. (Thus, the control is in this case properly 'weak'—no non-trivial permutation in the weakly controlled group is realized by the weakly controlling groups. This is in contrast to the lac operon example where every permutation of the lower subsystem is realized by the weakly controlling group.)

Thus, in this Krebs cycle model, there are dozens or hundreds each of natural subsystems of each of the five cardinalities 6, 5, 4, 3 and 2 for their state sets with cyclic groups acting on them transitively, as described above. Moreover, any other natural subsystem is equivalent to and isomorphic to one of these five. Among these we have identified at least one subset chain of subsets of these cardinalities whose permutation group each weakly controls the next smaller one. Thus, by theorem 2.5, weak control by these natural subsystems (permutator groups) by each other is a transitive total order of natural subsystems.

Remark. Generally if (X_1, G_1) weakly controls (X_2, G_2) with $X_2 \subsetneq X_1$, the subgroup of G_1 stabilizing X_2 restricts to a subgroup of G_2. The examples of the lac operon and Krebs cycle show that this may be all of G_2 or be the trivial group, but it may also be a non-trivial proper subgroup of G_2 as in the example for a symmetric group on three letters weakly controlling a SNAG arising in the p53–mdm2 pathway.

(c) p53–mdm2 genetic regulatory pathway

We examine natural subsystems and algebraic structure in a model of the p53–mdm2 system [35,36]. Figure 7 shows such a model of the p53–mdm2 regulatory pathway, which is important in the cellular response to ionizing radiation and can trigger self-repair or, in extreme cases, the onset of programmed cell death (apoptosis). This pathway is involved in ameliorating DNA damage and preventing cancer [38]. The p53 protein is linked to many other proteins and processes in the cell [38], but its coupling to the mdm2 protein appears to be particularly important for understanding cancer. Depending on the concentration level of p53, the cell can (in order of increasing concentration): (i) operate normally; (ii) stop all functions to allow DNA repair to take place; (iii) induce reproductive senescence (disable cellular reproduction); and (iv) induce apoptosis instantly (cell 'suicide'). Therefore, p53 is a very powerful and potentially very dangerous chemical that humans (and most other animals) carry around in each cell, whose control must be tuned very finely indeed. Approximately 50% of all cancers result from the malfunction of the p53–mdm2 regulatory pathway in damaged cells that should have killed themselves.

Levels of p53 are controlled by a fast-feedback mechanism in the form of the mdm2 protein. Synthesis of p53 occurs all the time, at a fairly fast rate; but the presence of p53 induces the synthesis of mdm2, which binds to p53 and causes it to be disintegrated. When the DNA is damaged (for instance, by radiation in radiotherapy) the cell responds by binding an ATP molecule to each p53, bringing it to a higher energy level that prevents its destruction and causes its concentration to rise. Thus in the simplest possible model [36], there are in all four biochemical species: p53 (**P**), mdm2 (**M**), p53–mdm2 (**C**) and p53* (**R**), whose concentrations are modelled by four coupled and nonlinear ordinary differential equations (ODEs). In [37], we derived a Petri net model for p53–mdm2 gene regulation shown in figure 7 and studied algebraically in [31,35,37,39]. It is increasingly common to model biological networks as Petri nets. Petri nets [40,41] can easily be transformed into finite automata, though this operation should be carried out with some care (for a detailed explanation, see [7]). A Petri net's state ('configuration') consists of assigning an amount of tokens (e.g. corresponding to molecules or moles) to each of several 'places' (e.g. corresponding to molecular species). A basic event or input for a Petri net ('transition') is a rule for consuming a fixed number of tokens from certain places and producing new tokens

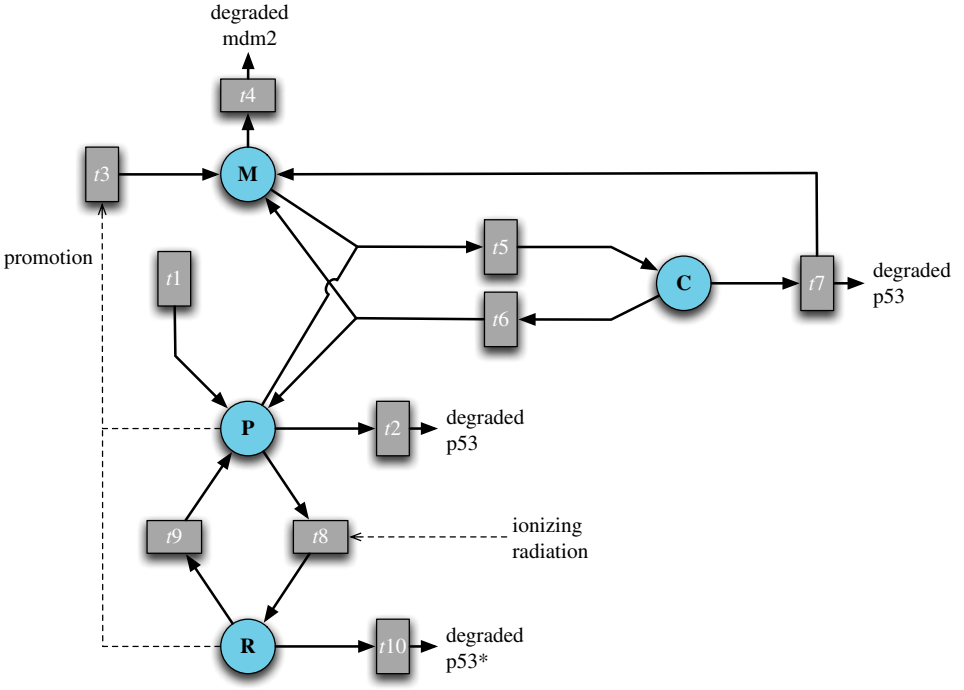

Figure 7. Petri net for the p53–mdm2 regulatory pathway following [31,35,37] with places **P** = p53, **M** = mdm2, **C** = p53–mdm2, **R** = p53* and transitions modelling genetic regulatory dynamics.

at other places. If the rule does not apply due to lack of tokens, no change occurs. Different transitions may occur at different rates, and these might vary or become zero ('inhibition') if the state satisfies certain conditions [42]. Depending on the capacity allowed in the places of the Petri net, the automaton's state set can be different in size, thus we can get different resolutions during discretization. Here if a transition occurs that would cause a place to exceed its capacity, its capacity is reached [7].

When we distinguish only between the presence and absence (in sufficient concentration above a threshold) of the molecules, the derived automaton has only 16 states (see figure 7).[23] This may be considered as a very rough approximation of the original process, but still natural subsystems given by permutation groups do appear in this simple model: S_3, the symmetric group on three points is among the components. The existence of these group components is clearly connected to oscillatory behaviours, however the exact nature of this relation still needs to be investigated. Natural subsystems representing all possible ones for this model are visualized in figure 8.

If we in addition distinguish three levels of concentration of the molecular species, i.e. absence (or very low, sub-threshold concentration), low and high levels concentrations, then the corresponding automaton has 81 states ($3^4 = 81$) and the algebraic decomposition reveals that it contains S_5 among its components [35]. The appearance of S_5 is particularly interesting as it contains A_5, the alternating group on five points, which is the smallest SNAG. If we let the place **M** for mdm2 have three levels (capacity 2), but the others **C**, **P**, **R** have only two (capacity 1), then A_5 already appears. Weak control relations among representative natural subsystems for this model of p53–mdm gene regulation appear in figure 9.

Weak control hierarchy representatives for the p53–mdm2 model using the Petri net capacity 1 model has 16 states, denoted by which of molecular species **M**, **C**, **P**, **R** are present (above some threshold levels). States give the set of active species, thus ∅ denotes none of the species

[23]Four places each taking on two possible values (0 or 1) makes $2^4 = 16$ states.

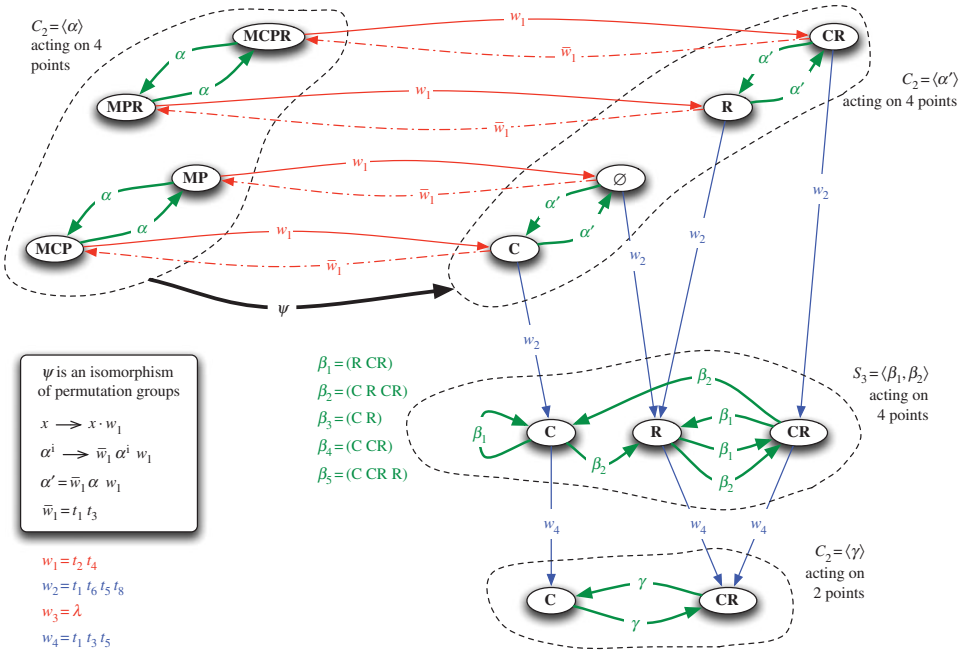

Figure 8. Weak control hierarchy for the p53–mdm2 case with binary values for the levels of **M** (mdm2), **C** (p53–mdm2 complex), **P** (p53) and **R** (p53*).

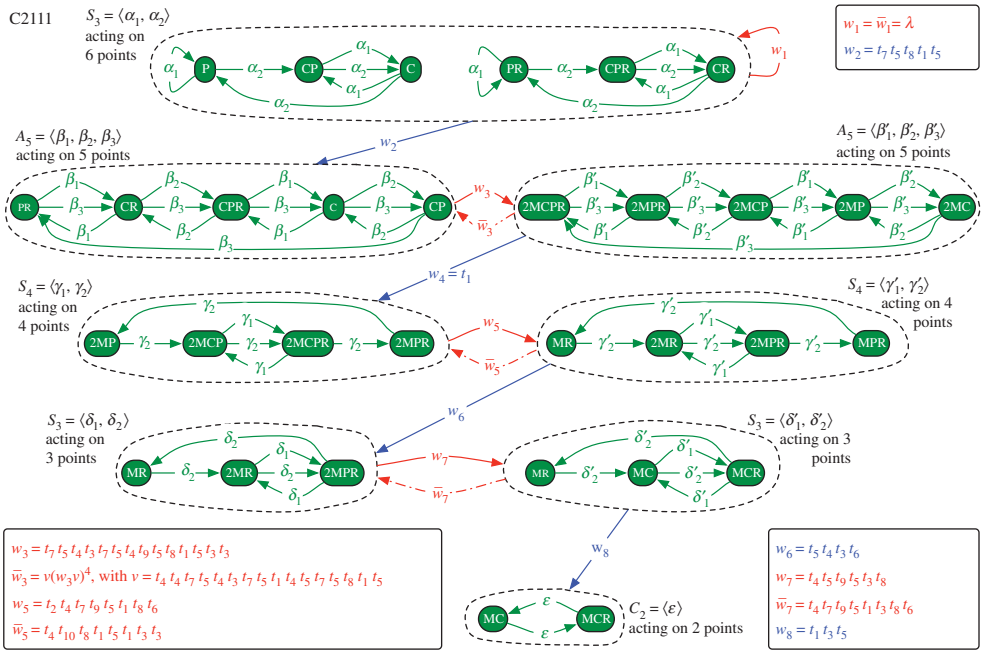

Figure 9. Weak control hierarchy for the p53–mdm2 petri net with capacities for $\mathbf{M} = 2$, $\mathbf{C} = 1$, $\mathbf{P} = 1$ and $\mathbf{R} = 1$.

above threshold, while **MP** denotes *mdm*2 and *p*53* above threshold but **C** and **R** below their respective thresholds. The natural subsystem $X_1 = \{\mathbf{MP}, \mathbf{MCP}, \mathbf{MCR}, \mathbf{MCPR}\}$ with cyclic group $C_2 = \langle \alpha \rangle$ acting on permutations is a transformation group with two orbits. The permutation α swaps states **MP** with **MCP**, and swaps **MPR** with **MCPR**. This permutation is realized by

infinitely many sequences of basic events in the Petri net, for example α is realized by infinitely many different words, including $t_4t_6t_5t_3t_7t_5t_3t_1$ and $t_2t_6t_5t_1t_5t_3t_5t_1t_5t_7t_5t_3t_1t_4t_6t_5t_3t_5t_1t_5t_7t_5t_3t_1$. Note that these words do *not* have the same effect on states outside of X_1, although they restrict to the same permutation α of X_1. In this case, the group G_1 of permutations of X_1 given by permutator words has exactly two distinct elements, α and the identity function α^2.

The permutator group (X_1, G_1) for this natural subsystem is, as visualized in figure 8, isomorphic to the natural subsystem with state set $X_1' = \{\emptyset, \mathbf{C}, \mathbf{R}, \mathbf{CR}\}$ and order 2 cyclic group G_1' via the isomorphism $\psi_{States}(q) := q \cdot w_1$, where w_1 is given by the word t_2t_4, the degradation of p53 and of mdm2. Its inverse is given by $\psi^{-1}_{States}(q) := q \cdot \bar{w}_1$ with $\bar{w}_1 = t_1t_3$ by the synthesis of p53 and of mdm2. These functions give a one-to-one correspondence between the two state sets X_1 and X_1'. Moreover, $\psi_{Group}(g) := \bar{w}_1 g w_1$ is a homomorphism of groups ($g \in G_1$) as, for all $g_1, g_2 \in G_1$ giving permutations of $\{\mathbf{MP}, \mathbf{MCP}, \mathbf{MCR}, \mathbf{MCPR}\}$, then $\psi_{Group}(g_1g_2) = \bar{w}_1g_1g_2w_1 = \bar{w}_1g_1w_1\bar{w}_1g_2w_1 = \psi_{Group}(g_1)\psi_{Group}(g_2)$ and is in fact an isomorphism with inverse homomorphism: $\psi^{-1}_{Group}(g) = w_1g\bar{w}_1$. We have $\psi_{States}(q) \cdot \psi_{Group}(g) = (q \cdot w_1) \cdot \bar{w}_1 g w_1 = q \cdot (w_1\bar{w}_1)gw_1 = q \cdot w_1 = \psi_{States}(q \cdot g)$, for all $g \in G_1$ and $q \in X_1$. (Note: this computation is essentially the proof of theorem 2.2 that $\mathcal{R}$-equivalent permutation groups are isomorphic using appropriate w_1 and $\bar{w}_1$.)

The right-hand side of figure 8 shows three natural subsystems, where each successive one's state set is a proper superset of the one below, in a transitive weak control hierarchy with three levels.[24] All natural subsystems in the semigroup of the p53–mdm2 capacity 1 automaton are equivalent to one of these: $G_1' = \langle \alpha' \rangle \cong C_2$ cyclic permutation group (X_1', G_1') acting on four states (with two disjoint orbits) representing 31 isomorphic natural subsystems, $G_2 \cong S_3$ the symmetric permutation group (i.e. all possible permutations) on three states $X_2 = \{\mathbf{C}, \mathbf{R}, \mathbf{CR}\}$ representing 120 isomorphic natural subsystems, and $G_2 \cong C_2$ cyclic permutation group on two states $X_3 = \{\mathbf{C}, \mathbf{CR}\}$, representing 86 isomorphic natural subsystems.[25] The sequence of transitions given by the word $w_2 = t_1t_6t_5t_8$ collapses the state set of the top natural subsystem onto the X_2, and event sequence $w_4 = t_1t_3t_5$ collapses X_2 onto X_3.[26]

Note that the top $C_2 = G_1'$ does not have any non-trivial elements that permute X_2 or X_3, while G_2 has all the permutations in G_3. Unlike in the previous examples, here we encounter a *non-Abelian* group S_3, i.e. which does not satisfy the commutative law $xy = yx$.

(i) Weak control in a larger p53–mdm2 model

With increased capacity of mdm2 so that three levels of concentration are distinguished, there are 24 possible configurations (states) in the p53–mdm2 Petri net, with the other species still limited only to being distinguished in a binary way as above a threshold or not. Writing **M** for concentration above the first threshold of mdm2, and 2**M** for concentrations above a second threshold, we extend the notation for states of the Petri net so that 2**MCP** denotes that **C** (the p53–mdm2 complex), and $\mathbf{P} = p53$ are both above threshold and mdm2 is above its second threshold. Similarly, **MCP** denotes mdm2 above the first threshold (but not the second), with both **C** and **P** above threshold, etc. Figure 9 gives the natural subsystems and words consisting of basic transitions for representative members of the weak control hierarchy.

The weak control relation is again transitive so we have a hierarchy of permutator groups with five levels: cyclic permutation group C_3 on six states with two disjoint orbits (representing 39 isomorphic natural subsystems), alternating group A_5 acting on five states (representing 234 isomorphic natural subsystems), symmetric group S_4 on four states (representing 531 isomorphic natural subsystems), symmetric group S_3 on three states (representing 476 isomorphic natural

[24]Words in the figure $w_1 = \lambda$ and w_2 show that (X_1', G_1') weakly controls (X_2, G_2), and $w_3 = \lambda$ and w_4 show that (X_2, G_3) weakly controls (X_3, G_3), while (X_1, G_1) weakly controls (X_3, G_3) by the empty word and w_2w_4.

[25]Generators of permutator groups are given in the figure, while generator words in basic events of the Petri net are given in the electronic supplementary material available on request for this and other examples in this article. Also exhaustive lists of state subsets for all equivalent natural subsystems are in the electronic supplementary material available on request.

[26]Weak control does entail that the controlled system is a homomorphic image. The reader should note that *none* of the state-mapping functions given by w_2, w_4 or w_2w_4 in this weak control hierarchy can be the state map for homomorphism of transformation groups.

subsystems) and cyclic group C_2 on two states (representing 173 isomorphic natural subsystems). Every natural subsystem is equivalent and isomorphic to one of these. In figure 9, the words of the form w_{2i} give maps a subset at level i to the level $i+1$, and words w_{2i-1} and $\bar{w}_{2i-1}$ yield isomorphisms between permutation groups at level i ($1 \leq i \leq 4$), establishing weak control.

Here is a nested collection of sets with each being an image of the next larger one, giving equivalent and isomorphic natural subsystems to the permutator groups in figure 9:

$$Y_1 = \{\mathbf{P}, \mathbf{PR}, \mathbf{C}, \mathbf{CP}, \mathbf{CR}, \mathbf{CPR}\} \supsetneq Y_2 = \{\mathbf{PR}, \mathbf{C}, \mathbf{CP}, \mathbf{CR}, \mathbf{CPR}\} \supsetneq Y_3 = \{\mathbf{C}, \mathbf{CP}, \mathbf{CR}, \mathbf{CPR}\}$$
$$\supsetneq Y_4 = \{\mathbf{CP}, \mathbf{CR}, \mathbf{CPR}\} \supsetneq Y_5 = \{\mathbf{CR}, \mathbf{CPR}\}$$

with permutator groups and transitive weak control relations

$$(Y_1, S_3) \succ_{wc} (Y_2, A_5) \prec_{wc} (Y_3, S_4) \succ_{wc} (Y_4, S_3) \succ_{wc} (Y_5, C_2),$$

where transitive weak control follows from the subset relation and from $Y_1 \cdot t_7t_5t_8t_1t_5 = Y_2$, $Y_2 \cdot t_3t_5t_1t_5 = Y_3$, $Y_3 \cdot t_7t_5t_4t_3t_6t_8t_1t_5t_1t_5 = Y_4$ and $Y_4 \cdot t_3t_5t_1t_8t_1t_5 = Y_5$.

Here, at the second level of a weak control hierarchy, is the alternating group A_5, a simple[27] non-Abelian group that acts on the collection of configurations $Y_2 = \{\mathbf{PR}, \mathbf{C}, \mathbf{CP}, \mathbf{CR}, \mathbf{CPR}\}$ by all possible even permutations. It is generated by permutations given by infinitely many words, e.g. by three permutations given by words $t_7t_5t_4t_3t_7t_5t_1t_4t_5t_3t_7t_5t_1t_5t_7t_5t_1t_4t_5$ yielding a product of disjoint transpositions $\beta_1 = (\mathbf{C}, \mathbf{CP})(\mathbf{CR}, \mathbf{CPR})$, $t_7t_5t_4t_3t_7t_5t_4t_9t_5ut_1t_4t_5t_7t_5(ut_4t_9t_5v)^2t_8t_1t_5$ (with $u = t_8t_1t_5t_7t_5t_4t_3t_7t_5$, $v = t_8t_1t_5t_3t_7t_5t_1t_5t_7wt_4t_7t_3w$, where $w = t_5t_4t_1t_5t_7t_5$) yielding a product of transpositions $\beta_2 = (\mathbf{PR}, \mathbf{CR})(\mathbf{C}, \mathbf{CPR})$, and $t_7t_5t_4t_3t_7t_5t_4t_9t_5ut_1t_4t_5t_7t_5t_9t_8t_1t_5$ giving the 5-cycle $\beta_3 = (\mathbf{PR}, \mathbf{CR}, \mathbf{CPR}, \mathbf{C}, \mathbf{CP})$ as visualized in figure 9.

4. Computing with simple non-Abelian groups: unconventional computation based on functionally complete alternatives to the two-element Boolean algebra in biological systems

Finite SNAGs are not only fundamentally important to group theory but are also very interesting computationally. In the p53–mdm2 model, we see both non-Abelian groups and even non-solvable groups arising. These are more complex types of symmetry than the cyclic groups we saw in the lac operon and Krebs cycle examples. In particular, we have seen various symmetric groups and have demonstrated the presence of the SNAG A_5 in multiple natural subsystems. The smallest SNAG is the alternating group A_5 comprising the 60 even permutations of five elements. For the p53–mdm2 genetic regulatory control network in the semigroup of the automaton of the second model examined in §3c(i), the SNAG A_5 appears in 234 isomorphic natural subsystems.[28]

SNAGs have been hypothesized to be related to biological error-correction [11]. They have the property of functional completeness [43–45] (see §4b), making them a natural candidate for realizing an analogue of 'universal computation' within the finite realm. The presence of a SNAG in the decomposition of the p53–mdm2 pathway suggests that the cell could potentially be performing arbitrarily complex finitary computation using this pathway.

[27] A group is *simple* if it has only trivial quotient groups. Finite simple groups are the atomic building blocks of symmetry. All finite groups may be reduced to a uniquely determined collection of such atomic components, and may be reconstructed from finite simple groups by cascading these irreducible simple components. With 60 elements, A_5 is the smallest SNAG.

[28] In general, and in the semigroup of this model in particular, the permutator group $(X, (X)\mathbf{per})$ on the subset X can be isomorphic to permutators of subsets other than X as we see from the example of 234 $\mathcal{R}$-equivalent natural subsystems each with a different state set on which an A_5 permutator group acts. But isomorphic copies of the group can also be present in multiple, disjoint isomorphic copies as subgroups of $S(\mathcal{A})$ inside $(X)\mathbf{per}_{sgp}$, with each acting on the *same* subset X in exactly the same way. These groups' actions will differ, however, on members of the rest of the state space $Q \setminus X$. So there are even more isomorphic copies than suggested by looking at the permutator groups.

(a) Unconventional computing with simple non-Abelian groups

Limits for current and foreseeable computer technology in terms of computational speed and power present a pressing challenge for science and mathematics to transcend using new and unconventional ideas. Present-day computation is based on two-valued Boolean logic as a two-state logic is the simplest to conceive of. However, other algebras—including finite SNAGs—with similar functional completeness properties are known but have not yet been deeply investigated as alternative foundations for possible future and emerging devices. Owing to the prevalence of symmetry groups in nature, especially physics [46] (including elementary particles [47], quantum dots [48,49], etc.), chemistry (e.g. in many quasi-crystals [50]) and quantum chemistry (e.g. with alternating group A_5 multiply and irreducibly represented in $SO(4)$, see [51]), many naturally occurring physical systems could potentially realize SNAG-based computation. Indeed, SNAGs such as the group A_5 of icosahedral/dodecahedral symmetry occur (in continuum-many instances) in the classical physical groups $SU(2)$ and $SO(3)$, and the SNAG A_{n+1} ($n \geq 4$) embeds in $SO(n)$ as oriented symmetries of the n-dimensional simplex [51], also. SNAG symmetry in such physical systems could potentially be manipulated at high speeds or in high densities (cf. [52]) in future and emerging devices. This potential could lead in the next decades to a shift from the use of bits, Boolean synthesis and logical techniques to computing based on non-conventional algebraic structures. Thus, as a complement to the current pursuit of new technologies to overcome speed and density barriers just to keep up with Moore's law, it makes sense to investigate the computational potential of alternative kinds of algebras as a broader, unconventional basis of computation superseding the Boolean logic of the bit.

(b) Finitary universal computation by simple non-Abelian groups

Computers up to the present day have almost exclusively been based on the two-element Boolean algebra $\boldsymbol{B}$. It has a special property which makes the computers computationally universal, namely every arbitrary function from $\{0,1\}^n$ to $\{0,1\}$ can be expressed by the basic operations of $\boldsymbol{B}$. This property is called *functional completeness*. However, not only $\boldsymbol{B}$ has this property. Formally, by a *functionally complete* algebra we mean an algebra with underlying set A and with basic operations $f_1, \ldots, f_k$ such that for every non-negative integer n and for every function $f\colon A^n \to A$ there is a polynomial expression $p(x_1, \ldots, x_n)$ over $\boldsymbol{A}$ such that for every n-tuple $(a_1, \ldots, a_n) \in A^n$ we have $f(a_1, \ldots, a_n) = p(a_1, \ldots, a_n)$. (*Polynomial expressions* are expressions built up from variables, constants from $\boldsymbol{A}$ and the basic operations of $\boldsymbol{A}$ using composition.) The two-element Boolean algebra, matrix rings over finite fields or the finite SNAGs are examples of functionally complete algebras [45,53,54]. Any computational device based on any of these algebras offers an alternative paradigm for computation. Horváth [43] gives a basic overview characterizing functionally complete algebras. There are several equivalent characterizations of functional completeness (some are theoretical and some operationally useful, e.g. the expressibility of certain basic functions which generate a functionally complete algebra). The functionally complete rings are matrix rings over finite fields, the functionally complete groups are the finite SNAGs. Moreover, every functionally complete semigroup is a group and every functionally complete semiring is a ring [43].

A natural question to ask is how to represent a function as a polynomial. Intuitively, these polynomials correspond to directed acyclic 'programmes' or 'circuits' for computing the desired function using basic operations of the algebra, so those of minimal length or depth are of special interest. One would like to know whether there is a short way of representing and a fast way of computing an arbitrary or a specific function over a given functionally complete algebra. The length of the polynomials needed to realize a given (Boolean or more general) function has been investigated for several different algebras. Not surprisingly, most of these results concern the two-element Boolean algebra (e.g. [53]). There are investigations also for rings [55], including various sporadic results, e.g. short representing polynomials were given for the square-root function in

[56]. Recently, a computational search has been applied to find particular polynomials for certain special functions over certain functionally complete algebras [57].[29]

The original paper of Maurer & Rhodes [45] characterizing the functionally complete groups is not algorithmic, but Horváth [43] gave an algorithm to calculate a representing polynomial for a given function and gave several upper and lower bounds for minimal length and for minimal depth of these polynomials. While there exist some results on the length of unary polynomials over finite groups [58], no estimates could be found in the literature for the n-ary case until recently [43,59]. This seems surprising as groups algebraically capture the notion of symmetry and play a fundamental role in physics, mathematics and numerous areas of computer science.

Finitary universal computation is defined here as the set of computational properties that follow from functional completeness. It includes the capacity to implement arbitrary mappings between finite sets of arbitrarily large size using polynomial functions, a familiar property of Boolean functions. This follows from the fact that functionally complete algebras A can implement any $f\colon A^n \to A$ with a polynomial for all $n \geq 0$, and hence can implement any function $f'\colon A^n \to A^m$ component-wise for all $m, n \geq 0$.

In the presence of SNAGs, there are many ways to implement finitary analogues of universal computation, as we now explore, with new results in §4c. Fröhlich's theorem [60] states that every function $f\colon G \to G$ with $f(1) = 1$ is a product of $\varphi_a(x) := axa^{-1}$ (inner automorphisms, $a \in G$) under pointwise multiplication if and only if G is a SNAG or has at most two elements. For SNAGs, there is the following analogue of the Stone–Weierstrass theorem significantly generalizing Fröhlich's result.

Theorem 4.1 (Stone–Weierstrass property of SNAGs (Maurer–Rhodes [45])). *Let G be a finite group and X be any finite set, both with three or more elements. Consider the set of functions G^X from X to G under pointwise multiplication $(f_1 * f_2)(x) = f_1(x) * f_2(x)$. Then let F be a subset of G^X such that*

(*a*) *F contains all constant functions: for all $g \in G$, there exists $f \in F$, such that $f(x) = g$ for all $x \in X$.*
(*b*) *F separates points: for all $x_1, x_2 \in X$, there exists $f \in F$ with $f(x_1) \neq f(x_2)$.*

Let $\langle F \rangle$ be the subsemigroup of G^X generated by F. Then, for all such F, $\langle F \rangle = G^X$ if and only if G is a SNAG.

It is easy to see that G^X is a group, and so, due to finiteness, $\langle F \rangle$ is a group too. Observe that condition (a) may be weakened to requiring F to contain constant functions for a set of generators of G. By the classification of finite SNAGs (see, for example, [61] for an overview), SNAGs can all be generated by just two elements, so it suffices to have two particular constant functions in F as the others are products of these.

Taking $X = G^n$ and considering all functions $G^n \to G$ under pointwise multiplication, the conditions of theorem 4.1 are satisfied if we take F to be the projections $\pi_i(g_1, \ldots, g_n) = g_i$ $(1 \leq i \leq n)$ and constant maps. Products of these projections and constant maps are the same as polynomials over G. Here, projections correspond to the variables. So we have that SNAGs can compute any function $f\colon G^n \to G$ using a polynomial. This remains true if we include more functions in F using inverses as well: a *polynomial function* over a group G is a function of the form $G^n \to G$ given by $(x_1, \ldots, x_n) \mapsto g_1 x_{i_1}^{e_1} g_2 \ldots g_k x_{i_k}^{e_k} g_{k+1}$, for some $k \geq 0$, $1 \leq i_1, \ldots, i_k \leq n$, exponents $e_1, \ldots, e_k \in \{-1, 1\}$, and constants $g_1, \ldots g_{k+1} \in G$.

Corollary 4.2 (functional completeness of SNAGs). *Let G be a SNAG. For every function $f\colon G^n \to G$ there is a polynomial p over G such that $f(g_1, \ldots, g_n) = p(g_1, \ldots, g_n)$ holds for all n-tuples of elements of G.*

[29]The authors of [57] used a computational search method (genetic programming) to search for discriminator polynomials, Mal'cev polynomials and majority polynomials for particular three- and four-element functionally complete algebras. It turns out that even for such small algebras it is quite difficult to find these polynomials. For example, the exhaustive search to compute a short discriminator polynomial over a particular four-element functionally complete algebra would take about 10^{38} years by their estimation. After a week of running time, their genetic programming method was not able to provide a discriminator polynomial for the algebra either (see [57] for further details).

Thus polynomials over SNAGs can realize any function, and no other non-trivial finite groups have this property. This implies that SNAGs are capable of finitary universal computation. The minimal length $\|f\|$ of a polynomial p needed to compute f in this manner is the *length* of f.

A completely different way to represent functions is via finite state sequential circuits. Krohn *et al.* [44] used this to show that a finite automaton whose semigroup is a SNAG can implement any Boolean function. It follows that so can any finite automata which have a SNAG subgroup divisor. Barrington [62] discovered the computational power of SNAGs independently of [44,45] and has shown, as a direct consequence of this result, that a language can be recognized by an $O(\log n)$ depth, polynomial size Boolean circuit if and only if it can be recognized by a polynomial length branching programme over a SNAG. Explicit upper and lower bounds on the lengths of polynomials needed to define functions over SNAGs, including the implementation of Boolean computation and permutation branching programmes, are given by Horváth & Nehaniv [43,59].

Theorem 4.3 (length of logical functions over SNAGs [59]). *Let $\mathbf{G}$ be a SNAG. Then there exists $g(\neq 1) \in \mathbf{G}$ such that, for every n-ary function $f\colon \{1,g\}^n \to \{1,g\}$ over $\mathbf{G}$, we have*

$$\|f\| \leq 448 \cdot n^8 \cdot e,$$

where $e = \left|f^{-1}(g)\right|$, ($e \leq 2^n$). If $\mathbf{G}$ is the alternating group $\mathbf{A}_m$ for some $m \geq 5$, then

$$\|f\| \leq (3n^2 - 3n + 2) \cdot e + 1.$$

Theorem 4.4 (length of functions over SNAGs [59]). *Let $\mathbf{G}$ be a SNAG, $N = |\mathbf{G}|$. Let f be an n-ary function over $\mathbf{G}$ taking non-identity values e times ($e \leq N^n$). Then the following inequality holds*:

$$\|f\| \leq 100352 \cdot K^2 \cdot N^8 \cdot n^8 \cdot e,$$

where $K \leq \min(c_0 \log N,$ number of conjugacy classes of $\mathbf{G})$ with c_0 a universal constant not depending on $\mathbf{G}$ or f. If $\mathbf{G} = \mathbf{A}_m$ for some $m \geq 5$, then

$$\|f\| \leq 9 \cdot m \cdot N^2 \cdot n^2 \cdot e.$$

If we use additional operations such as the *commutator* $[x,y] = x^{-1}y^{-1}xy$ in building up polynomials, then of course we can still realize all functions. Let $\|f\|_{\mathbf{C}}$ denote the minimal length of a polynomial p needed to compute f if commutators may be used. Clearly $\|f\|_{\mathbf{C}} \leq \|f\|$, but, intriguingly, Horváth [43] observed that expressions for a function may be decreased substantially if commutator is used. For example, the n-ary Boolean logic AND function can be represented in $O(n^k)$ length (where $k \leq 8$ depends only on the SNAG) using a recursive divide-and-conquer logarithmic depth idea [59,63], while with commutator its length is linear. There are also situations where the difference in the length of expressions is exponential with commutator, e.g. for $f(x_1,\ldots,x_n) = [x_1,[x_2,[x_3,[\ldots.[x_{n-1},x_n]]]]]$. However, for many solvable, non-nilpotent groups problems related to the length of the polynomials (the so-called equivalence and equation solvability problems) are in P without the commutator, but are (co)NP-complete if one is allowed to use the commutator [64,65]. This suggests again that there may be an exponential shortening for particular polynomials (but does not prove there is, unless $P \neq NP$). Generally, there are still very few results about minimal realizing polynomials as mostly there are upper bound results only but very few lower bounds.

If the commutator is considered as a basic operation, we have the following.

Theorem 4.5 (length of logical functions over SNAGs with a commutator [43]). *Let $\mathbf{G}$ be a SNAG with an added commutator operation $[,]$. Let $1 \neq u \in G$, and let f be an arbitrary n-ary function $f\colon \{1,u\}^n \to \{1,u\}$ with at most e-many non-identity values. Then*

$$\|f\|_{\mathbf{C}} \leq K \cdot ((10 + 3(n - \log e)) \cdot e - 5) + 1,$$

where K is a constant depending on the group $\mathbf{G}$ and the element u is bounded above by the number of conjugacy classes of $\mathbf{G}$.

When $\boldsymbol{G} = \boldsymbol{A}_m$ ($m \geq 5$) *and* u *is a three-cycle, then*

$$\|f\|_{\mathbf{C}} \leq 4 \cdot ((10 + 3(n - \log e)) \cdot e - 5) + 1.$$

If $4 \nmid m$, *then we can replace the constant factor* 4 *by* 2.

Theorem 4.6 (length of functions over SNAGs with commutator [43]). *Let* $\boldsymbol{G}$ *be a SNAG with an added commutator operation* $[,]$. *Let* f *be an arbitrary* n*-ary (possibly partial) function over* $\boldsymbol{G}$ *with* e*-many non-identity values. Let* $N = |\boldsymbol{G}|$. *Then the following inequality holds*:

$$\|f\|_{\mathbf{C}} \leq 3 \cdot K^4 \cdot N \cdot n \cdot e,$$

where K *is a constant depending on the group* $\boldsymbol{G}$ *and is bounded by the number of conjugacy classes of* $\boldsymbol{G}$. *If* $\boldsymbol{G} = \boldsymbol{A}_m$ ($m \geq 5$), *then*

$$\|f\|_{\mathbf{C}} \leq 176 \cdot \left\lfloor \frac{m}{2} \right\rfloor \cdot (N - 1) \cdot n \cdot e.$$

If $4 \nmid m$, *then we can replace the constant* 176 *by* 28.

By way of comparison, we note that, for the two-element Boolean algebra $\boldsymbol{B}$, the well-known conjunctive normal form gives an upper bound on the length on the order of $n \cdot e$ to realize any n-ary logical function taking the value true e times ($1 \leq e \leq 2^n$).

(c) Biological systems can compute any finitary function: simple non-Abelian groups and genetic regulatory control

SNAGs allow one to realize any mapping of a set they act transitively on using 'primitives' from an F as in theorem 4.1.

Proposition 4.7 (total control by SNAGs with feedback). *Let* (Y, G) *be a transitive permutation group where* G *is a SNAG. Let* $F \subseteq G^Y$ *satisfy the conditions of the Maurer–Rhodes theorem. Let* $h\colon Y \to Y$ *be any function. Then there is an* $f \in \langle F \rangle$ *with* $y \cdot f(y) = h(y)$ *for all* $y \in Y$.

Proof. As (Y, G) is transitive for all $y_1, y_2 \in Y$ there exists $g_{y_1,y_2} \in G$ with $y_1 \cdot g_{y_1,y_2} = y_2$. Defining $f\colon Y \to G$ by $f(y) = g_{y,h(y)}$ yields the desired f as $y \cdot f(y) = y \cdot g_{y,h(y)} = h(y)$. By the Maurer–Rhodes theorem, we can write f as a product of members of F. ∎

Recent work on the evolution of gene expression in living organisms has indicated that much of the evolutionary activity involves not only purifying selection and gene product evolution, but also regulatory evolution [66]. Indeed, the majority, perhaps as much as 83%, of genome-wide changes at adaptive loci in sticklebacks are regulatory [67]. Genetic regulatory evolution has been important in speciation and developmental changes [68,69] as well as in human evolution [70]. Moreover, conserved non-coding elements account for most of the genome that is conserved, whereas conserved coding DNA is an order of magnitude less [67].

The computational power of SNAGs suggests how they might be involved in regulatory control. Let Z be any collection of cell and environmental states, or any set of partition classes on a subset of a collection of such states in an automaton model $\mathcal{A}$ of a cell, or other biological system. Let (X, G) be a natural subsystem of $\mathcal{A}$. Suppose the cell's activity (e.g. via genetic regulatory control) can realize $\rho\colon Z \to A^*$, where $\rho(z)$ is a sequence of events in $\mathcal{A}$, giving a transformation of (X)**per**. That is, $\rho(z)$ acts as an element of G when restricted to X. Let F be a collection of such ρ that the cell can realize.

We say Z is G-distinguishable if, for all distinct $z, z' \in Z$, there is a $\rho \in F$, with $\rho(z)$ and $\rho(z')$ giving distinct permutations of X. For example, Z is G-distinguishable if the cell can detect each state z as opposed to any other state by a kind of generalized characteristic function $\chi_z \in F$ with $\chi_z(z)$ never equal to $\chi_z(z')$ as transformations of X for $z' \neq z$, Then χ_z is a kind of 'sensor' for state z. We say F contains a *generalized constant function* $c_g\colon Z \to A^*$, with $c_g(z)$ always giving the same permutation $g \in G$ of X.

Proposition 4.8 (genetic regulation with SNAGs). *Let (X, G) be a natural subsystem of automaton model $\mathcal{A}$ of some part of a cell, where G is a SNAG. Let Z be a collection of cell and environment states (or equivalence classes of these). Assume the cell can generate regulatory mappings of Z to event sequences in A^* by a collection F of regulatory transformations $\rho: Z \to A^*$ giving elements of G. If F contains generalized constant functions for generators of G and Z is G-distinguishable, then, for every $f: Z \to G$, there is a sequence $p = \rho_1 \cdots \rho_k$ of members of F such that $f(z) \in G$ is realized by the permutation given by sequence $p(z) = \rho_1(z) \cdots \rho_k(z) \in A^*$.*

Proof. This follows immediately from the Stone–Weierstrass property of SNAGs (theorem 4.1). In detail, write $[\rho]$ for the function from Z to G given by letting, for $z \in Z$, $[\rho](z) = [\rho(z)] = \cdot\rho(z)$ be the permutation of X given by the event sequence $\rho(z)$. Let $[F] = \{[\rho] \in G^Z \mid \rho \in F\}$. By hypothesis, $[F]$ separates points of Z and generates all constant maps in G, thus satisfying the conditions of theorem 4.1. Therefore, every function $f \in G^Z$ can be written as a product $[\rho_1] \cdots [\rho_k]$ of members of $[F]$ under pointwise multiplication, but $[\rho_1(z) \cdots \rho_k(z)] = [\rho_1(z)] \cdots [\rho_k(z)]$. ■

Thus, cells can use a natural subsystem (X, G) with G a SNAG to compute an arbitrary function from such a collection of cell and environment states Z to manipulations of X by any desired elements of the SNAG (proposition 4.8), and might also perform arbitrary mappings on X (proposition 4.7).

Thus, as suggested by Rhodes [11], SNAGs could be important in error correction and other forms of regulation in biological systems. Propositions 4.7 and 4.8, and the other results here, suggest numerous ways in which this might occur.

Suppose a SNAG (Y, G) occurs for the model of a cell. In terms of genetic regulatory control, suppose the cell can produce distinct members of G, e.g. as products of elementary events, in response to its current state y (separating points in state space), as well as produce outputs in G that do not depend on y (but at least a generating set for G, which might consist of as few as two elements), i.e. constant maps. Then the above results show that every function $Y \to Y$ can be realized by a sequence of these genetic regulatory outputs if they can be produced fast enough so that state may be regarded as unchanging during the production of the sequence.[30] *Thus cells (e.g. a cell with the p53–mdm2 genetic regulatory system modelled in §3c) are capable of finitary universal computation.* Indeed given any cell state $y \in Y$, such a system can steer y to a suitable new state (proposition 4.7). In this way, a cell with such genetic regulatory control could implement error correction, recovery, homeostasis and dynamic equilibrium, responding to perturbations of its state so as to return the cell to desirable dynamic regimes in its trajectory through state space. Similarly, proposition 4.8 suggests how genetic regulatory control could implement any mapping from cell and environment states to the action of a SNAG.

The results of this section have natural analogues for the more general case where an automaton $\mathcal{A}$ has a SNAG G as a divisor (homomorphic image of subgroup $\tilde{G}$ of $S(\mathcal{A})$), then $\tilde{G}/N \cong G$ for a normal subgroup N in $\tilde{G}$, then we can use $\tilde{G}$ instead of G to compute any function $f: X \to \tilde{G}$ up to a coset using the same methods as above, that is, there is h in the $\tilde{G}^X$ generated by suitable[31] subset F, with $h(x) \in f(x)N$, i.e. $h(x)f(x)^{-1} \in N$. This also applies to computing partial functions $f: X \to \tilde{G}$, which are less constrained.

Moreover, there are also many ways to employ direct product ensembles, independent copies of the cell or system, by using multiple independent copies of (X, S) for any semigroup S acting faithfully on a set X. In particular, this is true when $X \subseteq Q$ and $S = (X)\mathbf{per}$ divides $S(\mathcal{A})$, with S a SNAG. A useful fact via 'computing in parallel' is related to the interpretation of groups in Gibbs's viewpoint on ensembles. For any faithful permutation group (X, G), the right regular

[30] This restriction might be removable also if the sequences produced would vary in time in response to state that changes more quickly, but details of conditions that make this possible have not been characterized yet.

[31] F must separate points of X up to cosets and contain enough constants up to cosets: for the first, distinct points should map to different cosets of N and, for the second, there must exist functions approximating constants for generators of G, i.e. values must be in the same coset of $\tilde{G}/N$ rather than strictly constant.

representation (G, G) with action $g_1 \cdot g = g_1 g$ can be realized as a substructure of the direct product of X-many copies of (X, G). That is, $|X|$ instances of a regulatory network within a cell, or $|X|$ instances of the cell, allow one to embed (G, G) into the ensemble: let $X = \{x_1, \ldots, x_m\}$ denote the m distinct elements of X, then a state $g_1 \in G$ can be uniquely encoded as $(x_1 \cdot g_1, \ldots, x_m \cdot g_1)$ as the action is faithful. Then the action of $g \in G$ is given by letting g act in parallel on each component. The result, $(x_1 \cdot g_1, \ldots, x_m \cdot g_1) \cdot g = (x_1 \cdot g_1 g, \ldots, x_m \cdot g_1 g)$, corresponds to state $g_1 g$ of the right regular representation. (This works even if G is not a group as long as it acts faithfully on X). However, many other different kinds of networks of copies of (X, G) could be employed in different ways to yield circuits computing over the SNAG (e.g. implementing any function or polynomial). Combining multiple copies of a system with SNAG (X, G) in a way that emulates multiplication in G gives another way to compute any finitary function $G^n \to G$, in particular $n|X|$ copies suffice.

5. Interaction machines

(a) Dynamically growing and changing automata networks

Nehaniv & Wagner [71] wrote that appropriate mathematics is lacking for developmental and evolutionary biology; in particular, there is still no developed mathematical theory of *constructive dynamical systems* whose state space and dimensionality may change, growing and shrinking over time, as a function of—or at least constrained by—their interaction with other such networks and the external environment. Moreover, mathematical and computational biology still continue, almost completely, to lack adequately developed formal tools ('the right stuff' [71]) to begin the serious study of biological complexity. Early insightful efforts by Rosen on constructive sequential functions [72–74] in this direction continue to arouse scientific interest but reportedly suffer from mathematical errors that remain to be corrected [75]. Constructive models of self-maintaining, self-producing systems have been studied computationally by several researchers (e.g. [76,77]), but not mathematically. Methods to address the kind of structures that appear in biological and biochemical systems, and artificial systems with similar characteristics, include those of Dittrich and co-workers [78–83], who introduced a general theory of chemical organization, while Paun and colleagues [84–86]; developed a theory of membrane computing; in addition, there are attempts to apply Milner's π-calculus [87] to systems biology [88–90]. Other useful methods in biological and adaptive complex systems include Petri net modelling [1,40,41] with hierarchical extensions [5], multi-valued logical modelling (generalizing Kauffman's Boolean network approach [4]) combined with constraint analysis [1,3,6,91,92].

Overall, such efforts, while often constructive and even useful for simulation modelling, identifying constraints or to generate testable hypotheses on biological structures, are still largely at the level of formal descriptions and lack the kind of mathematical power one would like to see: we still lack the analytic mathematical tools for providing deeper understanding and for proving powerful theorems to yield insight into higher level structure. This is most severe for constructive dynamical systems which interact and change the essentials of their structure (e.g. the dimensionality of their phase spaces).

Krohn–Rhodes theory [21] provides a powerful framework for discrete dynamical systems, and theorems for the synthesis and decomposition of finite automata models using algebra applied to biological complexity [11,30,34,93]. As illustrated here algebraic methods allow one *to discover and interpret higher level structure including natural systems and their relations* (in this article), and they also allow one *to give hierarchical coordinate systems and coarse-to-fine graining of dynamical structure* ([11,34] and theorems 6.1 and 6.2). However, these methods were developed for systems that do not fundamentally change their structure as living systems do. We can begin to address this by introducing dynamical automata networks ([19], §5d) and interaction machines (§5e).

(b) Example 1. Modelling cell division

The biological cell is an entity with variable size, variable numbers and types of internal structures—such as organelles, vesicles, ribosomes—as well as variable numbers and concentrations of metabolites, enzymes and structural proteins. Dynamical systems with a fixed number of coordinates or dimensionality, whose space of possible system trajectories is fixed and unvarying from the outset of analysis, are adequate primarily only to model such entities as cells on short time scales, over which the structure and topology of constituent components' interactions do not change.

A traditional approach thus fails to capture and adequately represent the possibilities inherent in natural systems. For example, even for something relatively 'simple' like cell division, the most natural dynamical model would require doubling the number of variables somehow in the course of continuous dynamics, with the number of chromosomes in their respective states of methylation, and so on, doubling. Furthermore, with one cell dividing into two, it would be natural for the model to double the number of variables describing the internal milieu, concentrations of metabolites, enzymes, co-factors, and as well as variables describing the states of internal structures (organelles such as mitochondria, vesicles, ribosomes, Golgi bodies, etc.), which themselves consist of complex structures. We outline how to do this using interaction machines in §6b.

(c) Example 2. Finite dynamically expansive modulo *n* counter cascade

A familiar example of changing the number of states and components in computation arises in our daily practice of employing the decimal or binary expansion of the natural numbers. It is well known that the cascade of modulo n counters ($n \in \mathbb{N}^+$) encodes our usual base n positional notation and basic arithmetic operations. That is, with k positions one can denote any integer v from 0 to $n^k - 1$ as $\sum_{i=0}^{k-1} d_i n^i$, where the ith digit $d_i \in \{0, \ldots n-1\}$ gives the number of n^i in v to encode v as a sequence $(d_{k-1}, \ldots, d_0)$. For example, the three-digit sequence 431 encodes $4 \times 7^2 + 3 \times 7^1 + 1 \times 7^0$ in base 7 but denotes $4 \times 16^2 + 3 \times 16 + 1 \times 16^0$ in hexadecimal. Moreover, the usual carry rules guarantee that, in adding or multiplying numbers, information flows only from right to left. That is, the computation of the ith digit never depends on digits to the left of the ith position. Such base n expansions with their carry rules, which are taught and used worldwide for representing and manipulating numbers, are just linear cascade automata networks of modulo n counters [11,94] with very useful properties (e.g. [95]). Generalizations to such coordinatizations for arbitrary finitary systems are detailed in [11] and, for example, §6c.

Now in practice the number k of digits needed to represent numbers is not fixed for all time and all applications. We ignore leading zeros, but also add new digits to the left as needed to represent non-zero place values arising from carries; for example, in adding in base 10: $17 + 99 = 116$, a third position on the left *appears* whereas the summands had only two digits. Or in subtracting: $1003 - 6$, the leftmost position *disappears*, when we report the answer 997 as a sequence of fewer digits. A cascade of some *fixed* number k of modulo n-counters can represent what is going on, as long as k is some integer greater than the number of digits one will ever need for representing place values. Conventionally, all leftmost zero values are tacitly ignored although they are implicitly present in this representation. Alternatively, an *infinite* number of counters can be used to guarantee never running out of positions [9,94].

However, a novel minimal extension of the idea of cascade of automata gives a more natural formalization for how we do arithmetic in practice: to add two k-digit numbers, only k counters are required before the addition, but a new component counter appears ('grows') at the left of the k component cascade after the operation if a carry rule applies in the leftmost position. The number of automata may change based on upstream conditions given an input to be added or subtracted (figure 10). The distinction is subtle, but important. At any given moment we have only a finite number of positions (counters in the cascade), but this number may change *dynamically*. Formally, the current state is a k-component network of automata, each one carrying the value

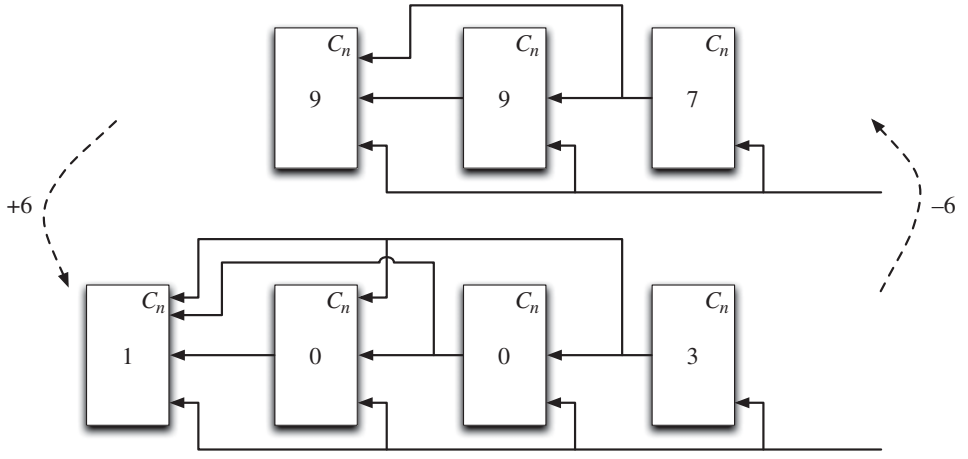

Figure 10. *Formulation of the base n expansion of natural numbers as a dynamic cascade automata network.* A cascade of modulo n counters C_n gives the usual base n expansion of integers with a dynamically changing number of digits. Taking $n = 10$, the top cascade holds three-digit base n numerical representations, while the bottom cascade holds four-digit base n numerical representations. The number of digits may increase if a carry results in a non-zero value beyond the leftmost digit; while if subtraction would result in the leftmost digits becoming zero then at the same time the leftmost automata for these digits are removed, resulting in a cascade of fewer automata. (Example due to C. L. Nehaniv and E. Rothstein Morris.)

$d_i \times n^i$ if it is in the ith position. *During the transition* with the carry rule a $k + 1$st position, i.e. a new modulo n counter, is created and put in a state giving the value for the new position. Similarly, for subtraction, the size of the cascade (i.e. the number of positions) may change, with all components with leading zero values disappearing. Thus, while four counters are needed prior to computing $1003 - 996$, afterwards, rather than four counters with states 0007, only one modulo 10 counter in state seven remains.

In a transformation semigroup (Q, S) or automaton $\mathcal{A}$, if X is a set of states and $t \in A^*$ is an input word giving a transformation, we always have $|X \cdot t| \leq |X|$. Now from the ensemble point of view, it is worth noticing that the number of states represented may *increase* in a dynamic ensemble of copies of $\mathcal{A}$. For example, in figure 10 with its cascade of modulo n counters, the size of the ensemble may grow under the action of the input +6. Moreover, in a direct product ensemble of fixed size, the number of distinct states X occurring in the ensemble cannot increase under the action of t, but with a dynamic automata network that creates a new parallel component this can easily occur, so it is possible for $|X \cdot t| \leq |X|$ to fail.

(d) Interaction machines (level 1): dynamic generalization of automata networks

Interaction machines of level 1 are a generalization of automata networks (with static topologies—whose study goes back at least to Gluškov & Letichevsky in the early 1960s [9,20]). As we saw in §1d, an automata network is an ensemble of automata connected by interaction functions, and the next state of each automaton is computed synchronously as a function of an external input (basic event) and the current state of all the automata in the network according to the interaction function for the given component automaton. The so-called *general product* of automata allows arbitrary feedback between components, and the overall network is a new automaton. Other types of products restrict the topology and/or number of components that the ith interaction function may depend on, giving rise, for example, to bounded-feedback products, products with bounded number of dependencies, direct product, and cascade products.[32] In cascade products,

[32]The latter correspond in algebra to substructures of the wreath product. These are also called *feedback-free* or *loop-free products*.

the interconnection topology is given by an acyclic directed graph, typically a linear arrangement as in the example of decimal and base n expansions used in arithmetic.

The notion of a (level 1) interaction machine generalizes the notion of an automata network by allowing the removal or addition of some component automata and changes to the interconnection topology. Any new components also go into a well-defined state at the end of a transition. Variants of such growing and dynamically changing automata networks have been introduced and studied by Nehaniv & Quick [19] as *dynamic automata networks* and independently by Sayama and colleagues [22–24] as *generative network automata*.

A **level 1 interaction machine** is a (synchronous) *dynamic automata network* $\mathfrak{A}$ (generalizing the usual definition of automata network) comprising a time-varying structure

$$(\Gamma^{(t)}, \{\mathcal{A}_v^{(t)}\}_{v\in V^{(t)}}, \{\varphi_v^{(t)}\}_{v\in V^{(t)}}, A)$$

that has interconnection digraph $\Gamma^{(t)} = (V^{(t)}, E^{(t)})$ with vertices $V^{(t)}$ and edges $E^{(t)} \subseteq V^{(t)} \times V^{(t)}$ which, starting from an initial state, both may change as a function of discrete time $t \in A^*$ and local conditions. Note that time is modelled by the free monoid A^* (or free semigroup A^+) on the global input event alphabet A.[33] (Notice that the length of an input word t gives the number of discrete external events driving the system. Thus length can be thought of as a clock time index in the natural numbers $\mathbb{N}$. That is, different interactions with the external environment will give different trajectories through the space of the possible configurations of the interaction machine, depending on its initial configuration). Each node $v \in V^{(t)}$ indexes an automaton $\mathcal{A}_v^{(t)}$ in some state $q_v^{(t)}$. The tuple $\mathfrak{A}^{(t)} = (\Gamma^{(t)}, (\mathcal{A}_v^{(t)})_{v\in V^{(t)}}, \{\varphi_v^{(t)}\}_{v\in V^{(t)}}, (q_v^{(t)})_{v\in V^{(t)}})$ is called the *total configuration of the dynamic automata network at time t*. To aid the intuition, by abuse of notation we write '$t+1$' for the concatenation ta of the word t and new external input $a \in A$ here, as ta has length ('clocktime index' in $\mathbb{N}$) one more than t. Then, depending on the global input $a \in A$ in the next step we will have $\mathfrak{A}^{(t+1)} = (\Gamma^{(t+1)}, (\mathcal{A}'_v)_{v\in V^{(t+1)}}, \{\varphi'_v\}_{v\in V^{(t+1)}}, (q'_v)_{v\in V^{(t+1)}})$. Denote the interconnection digraph of the automata network at time $t+1$ by $\Gamma' := \Gamma^{(t+1)}$, the automaton at time $t+1$ (that is, $ta \in A^*$) in node v by $\mathcal{A}'_v := \mathcal{A}_v^{(t+1)}$ and its next state at node v by $q'_v := q_v^{(t+1)}$. If $v \in V^{(t)} \cap V^{(t+1)}$, then $\mathcal{A}'_v = \mathcal{A}_v$ and is in the state $q'_v = \delta_v(q_v^{(t)}, \varphi_v^{(t)}(q_{N(v)}^{(t)}, a))$. Otherwise, if $v \in V^{(t+1)} \setminus V^{(t)}$, then $\mathcal{A}'_v$ is a new automaton in state q'_v which is created as a function of $\mathfrak{A}^{(t)}$ and a. Also φ'_v for each $v \in V^{(t+1)}$ is given by a function of $\mathfrak{A}^{(t)}$ and $a \in A$. Generally, such new interaction functions φ'_v, automata $\mathcal{A}'_v$ and their states q'_v, as well as changes to the interconnection digraph, will be locally determined. For our purposes here we shall allow arbitrary changes to the topology, components, and state as a function of the current configuration of the network and $a \in A$. These updates can be constrained in various ways to guarantee various special properties. See §5c for an important novel application, [19] for constraints guaranteeing synchronization properties, as well as [19,22–24] for various graph-rewriting constraints.[34] All this can be viewed as a higher level transition function $\mathfrak{A}^{(ta)} = \mathfrak{A}^{(t+1)} = \delta(\mathfrak{A}^{(t)}, a)$. Sections 5b and 6b describe more general (level $n+1$ interaction machine) applications to multicellularity.

(e) Interaction machines (level $n+1$): dynamically deployable computational structure

These are the same as level 1 interaction machines except that the component automata of the network may themselves be interaction machines. Thus one has a 'recursive' structure, where constituent components themselves may be dynamically changing networks of automata (or interaction machines). For simplicity, we shall suppose that the component automata of an

[33] Writing earlier events to the left and later events to the right, the associative law $\alpha(\beta\gamma) = (\alpha\beta)\gamma$ holds for (non-overlapping) events α, β, γ in time. Semigroups (which are defined as those structures satisfying the associative law) can be thus considered as *models of time* (following [11, Prologue; 96]).

[34] In that work, constraints are stated on topology changes which guarantee that such a synchronously updated network can be emulated asynchronously. For example, this is guaranteed for certain basic changes analogous to biological growth in multicellular organisms where cells correspond to nodes and edges to contact between cells: nodes may split into two connected to their parent's neighbourhood, new nodes appear connected to at most one current node, connections may disappear, and changes in topology depend only on local conditions.

interaction machine of level $n+1$ are interaction machines of level at most n, where level 1 interaction machines were defined in the previous section.[35] Indeed, in modelling, the degree of recursiveness might not be known, or fixed in advance. Moreover, in behavioural specifications of the interaction machines details such as the number of levels need not be constrained. Interaction machines may thus also be construed as 'fractals', i.e. with structure at every level.

Iterating the idea of a level 1 interaction machine, a **level $n+1$ interaction machine** is a dynamic interaction machine network $\mathfrak{A}$ with time-varying structure

$$(\Gamma^{(t)}, \{\mathfrak{A}_v^{(t)}\}_{v\in V^{(t)}}, \{\varphi_v^{(t)}\}_{v\in V^{(t)}}, A)$$

that has interconnection digraph $\Gamma^{(t)} = (V^{(t)}, E^{(t)})$ with vertices $V^{(t)}$ and edges $E^{(t)} \subseteq V^{(t)} \times V^{(t)}$ which, starting from an initial configuration, both may change as a function of discrete time $t \in A^*$ and local conditions. Each node $v \in V^{(t)}$ indexes an interaction machine of level n or less in some total configuration $\mathfrak{A}_v^{(t)}$. The tuple $\mathfrak{A}^{(t)} = (\Gamma^{(t)}, (\mathfrak{A}_v^{(t)})_{v\in V^{(t)}}, \{\varphi_v^{(t)}\}_{v\in V^{(t)}})$ is called the *total configuration of the interaction machine at time t*. Note that $\mathfrak{A}_v^{(t)}$ includes information on the state of all of its constituent automata. Again, by abuse of notation, we write '$t+1$' for the concatenation ta of the word t and new external input $a \in A$ here, as ta has length ('clocktime index' in $\mathbb{N}$) one more than t. Then, depending on the global input $a \in A$ in the next step we will have $\mathfrak{A}^{(t+1)} = (\Gamma^{(t+1)}, (\mathfrak{A}'_v)_{v\in V^{(t+1)}}, \{\varphi'_v\}_{v\in V^{(t+1)}})$. Denote the interconnection digraph of the interaction network at time $t+1$ by $\Gamma' := \Gamma^{(t+1)}$, the component interaction machine at time ta in node v by $\mathfrak{A}'_v := \mathfrak{A}_v^{(t+1)}$. If $v \in V^{(t)} \cap V^{(t+1)}$, then $\mathfrak{A}'_v = \delta_v(\mathfrak{A}_v^{(t)}, \varphi_v^{(t)}(\mathfrak{A}_{N(v)}^{(t)}, a))$. Otherwise, if $v \in V^{(t+1)} \setminus V^{(t)}$, then $\mathfrak{A}'_v$ is a new interaction network of level n or less in configuration $\mathfrak{A}'_v$ which is created as a function of $\mathfrak{A}^{(t)}$ and a. Similarly, φ'_v for each $v \in V$ is determined by the total configuration $\mathfrak{A}^{(t)}$ and $a \in A$. Again, generally, the configurations of such constituent interaction machines $\mathfrak{A}'_v$ of lower level, and interaction functions φ'_v as well as changes to the interconnection digraph, will be locally determined. Like before, for the $n+1$ level interaction machine, this defines a higher level transition function $\mathfrak{A}^{(ta)} = \mathfrak{A}^{(t+1)} = \delta(\mathfrak{A}^{(t)}, a)$.

For our purposes here we shall allow arbitrary changes to the topology, interaction machine components, and their configurations as a function of the current configuration of the network and $a \in A$, but in practice these must be constrained to reflect the feasibility and locality of computation in any given model (figure 11).

We remark that, for the purpose of specification of behaviours and studying multiple realizations, one could reformulate our definitions of automata network, level 1, and level $n+1$ interaction machines co-algebraically (cf. [98,99]).

6. Permutation groups in recurring transient structures in dynamic ensembles

> Beyond the instability of the thermodynamic branch (i.e. equilibrium) we may have a new type of organization relating the coherent space–time behaviour to the dynamical processes (e.g. chemical kinetics and convection) inside the system. Only if appropriate feedback conditions are satisfied can the thermodynamic branch become unstable at a sufficient distance from equilibrium. The new structures that appear in this way are radically different from the 'equilibrium structures' studied in classical thermodynamics, such as crystals and liquids. They can be maintained in far-from-equilibrium conditions only through a sufficient flow of energy and matter.
>
> —Nicolis & Prigogine [100]

We'll have to do it again then, won't we? — British pantomime

[35] However, it may be useful to allow 'non-well-founded' induction (cf. [97]), where the level of constituent components is not bounded below, or a temporarily changing partial order may index the level of machines.

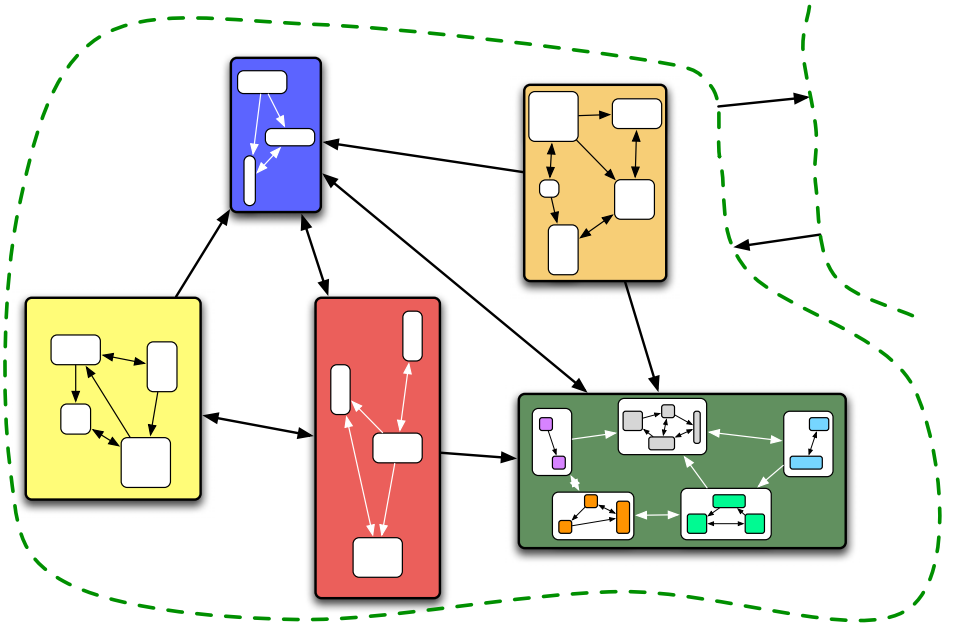

Figure 11. A level $n+1$ interaction machine is a dynamic automata network of interaction machines of level n or below. The entire structure and each constituent component updates its state based on local conditions and environmental input. Based on interactions and local state the network may change its interconnection topology and how components affect each other (i.e. the functions that locally determine inputs to constituent components), as well as changing the number and type of components dynamically. Interaction machines and their constituents of level 1 or above may grow or remove components as needed in response to interactions.

(a) The ensemble viewpoint for interaction machines

Typically, physicists and engineers studying a complex system may discard features of a model that represent its initial 'transient' behaviour, and focus on long-term properties that persist in the limit, e.g. using ergodic methods that apply to a behavioural regime that obtains after some initial transient behaviour which lacks properties such as ergodicity. The long-term behaviour of living systems, however, is *death*, which, while thermodynamically simple, is not biologically interesting. Indeed, biological systems continually regenerate structure and even reproduce themselves. Individuals and also evolving populations continually engage with and re-engage with the same contexts. While any given cell or individual will eventually die, similar entities are generated which re-enact the same kinds of interactions with entities and events in their environments.

In automata models, when non-trivial sequences of inputs (or events) permute some subset Y of states, the set of sequences permuting Y will be infinite,[36] and each such sequence will determine a member of permutation group $(Y)\mathbf{per}$. The pair $(Y, \mathbf{per}(Y))$ can be considered a 'natural subsystem', 'reaction chain', 'pool of feedback' or 'pool of local reversibility' (§2b). These groups are significant for instance in characterizing the atomic components (finite simple groups) needed to build a Krohn–Rhodes decomposition of an automaton in the form of either a cascade of automata or a series–parallel composition of sequential machines (see, for example, [8,11] for details). These groups and the relations between them are also important in computing the Krohn–Rhodes complexity of these automata (minimal number of group levels needed in a decomposition (theorem 1.1). However, in the course of running such an automaton, if inputs are stochastic, say, then there is a probability $\epsilon > 0$ that a given input symbol or event takes us out

[36] In the case of finite automata, the sequences permuting the members of Y will be a regular language $L(Y)$.

of the language of words that can be extended to permutations of the local state space Y, so that after m events the probability of remaining in the 'pool of reversibility' is $(1-\epsilon)^m$ if the events are independent. This probability converges to zero with more events. The natural subsystem is thus 'transient'. Indeed all reversible behaviour is transient in this sense, except for a final lowest level where the system is 'dead'. This lowest level may be ergodic and characterize what the system does as time goes to infinity, but, for living systems, this ignores all their most interesting structure. (And in contrast to Krohn–Rhodes analysis says next to nothing about their complexity or how to synthesize them from simple components.) Long-term behaviour in this sense reflects only death.

However, living systems such as cells and differentiated multicellular organisms, and also their components such as constituent instances of the Krebs citric acid cycle, organelles or particular genetic regulatory subnetworks (like the p53–mdm2 cellular response to damage and cancer), exist in varying amounts and in varying topologies. In particular, they can be modelled using interaction machines that dynamically grow and change their interconnectivity while undergoing temporal evolution in response to interactions with the environment.

That the long-term behaviour of a fixed automaton will converge out of any transient regime to a final subset of possible states is very simple to see, but the long-term behaviour of a dynamically renewed ensemble of such automata can continue (physically and mathematically) to re-visit any given natural subsystem $(Y, (Y)\mathbf{per})$, albeit with newly generated components.

Gibbs's ensemble interpretation applied to automata whose semigroups contain permutation groups is made concrete by interaction machines where natural subsystems (which are permutation groups) repeatedly recur. The multiple copies or components are simultaneously present in the ensemble defined by the interaction machine for such an interaction network of copies of the automata (even without any interaction between the components, i.e. parallel composition as in the simplest Gibbs interpretation with one copy of the system in each of its possible states). At any given moment, new copies of component automata may be generated. In such a setting, which applies for modelling multicellular growth or protein bio-synthesis of the constituents needed for multiple instances in a cell of a given metabolic pathway, etc., the notion of transient behaviour of any given copy of the system fails to capture the global recurring instantiation of the full dynamics of the system arising in the behaviour of the ensemble. Indeed, the same principle of adding copies of the system applied to a transformation semigroup in each of its possible states allows all the possibilities and mappings in the semigroup to be realized recurrently, without permanent loss in a transient. For every mapping s of the semigroup, any set X stabilized by s (e.g. any natural subsystem where s acts) recurs in the ensemble, so even after arbitrary time $\cdot s\colon X \to X$ will realized. Thus, the entire semigroup S with its space of dynamical possibilities $(Q, S(\mathcal{A}))$ recurs indefinitely often. If one has a good model for understanding its natural subsystems and the transitions between them,[37] this can be applied repeatedly, in spite of the fact that they may only occur transiently in a particular component.

(b) Example 1 continued: cell cycle and multicellular differentiation

Let $\mathcal{C}$ be any finite automaton cell model including the cell cycle which results at some point in division, for example using the simple cell-cycle model of [101] modelled as a Petri net in figure 12.[38] Here phosphorylated cdc-2 (cdc2-P) and cyclin oscillate in synchronization with the cell cycle; in sufficient concentration cdc2-P sets off a cascade of events that results in the cell

[37]This refers not only to scientific or experiential understanding, but also to hierarchical coordinate systems obtained by Krohn–Rhodes decompositions, using say the partial order $\Xi/\mathcal{R}$ of weak control hierarchy (as in theorems 6.1 and 6.2).

[38]We remark that in this Petri net for the cell cycle there are non-trivial automorphisms (symmetries). One gets an automorphism of the Petri net via the (external) symmetry of order 2: on places, (cdc2 amino acids) (cdc2-P cyclin), i.e. swap cdc2 and amino acids and simultaneously swap cdc2-P and cyclin; whereas, on transitions, swap as follows ($t5t7$) ($t4t6$). This induces an (external) automorphism of order 2 on the six-state automaton, as evident from figure 12*b*.

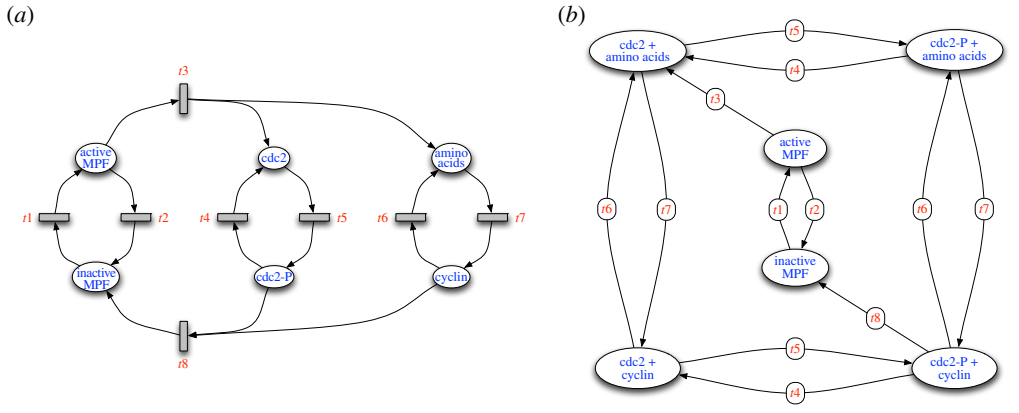

Figure 12. (*a*) Petri net model of the cell cycle based on a differential equation model of reactions as described by Tyson [101]. (*b*) Finite six-state automaton associated to this Petri net whose state set consists of all configurations reachable from the initial configuration with one token at the place representing MPF (active or inactive).

division. Mitosis-promoting factor (MPF) consists of cdc2-P bound to cyclin, and when activated breaks down into cdc2 and amino acids, mediating this process.

The natural subsystems and their weak-control hierarchy in the transformation semigroup of the six-state automata (found using an analysis with SGPDEC [17] like before, modulo $\mathcal{R}$-equivalence) comprise three permutation groups in this order with cyclic groups C_2, C_3, C_2 acting on 4-, 3- and 2-macrostate sets. Thus, Abelian subgroups C_2, C_3, C_2 correspond to natural subsystems that exist in multiple copies in the transformation semigroup (which has 1309 elements) of the automaton. Larger starting configurations lead to even more and larger groups, including non-Abelian ones already when starting with two tokens of MPF.

We might also add constituent finite automata into the network modelling the Krebs cycle or p53–mdm2 genetic regulatory control to obtain an automata network with fixed topology.[39] If the constituents have natural subsystems with functionally complete groups (SNAGs), they could in single or multiple copies be computing any finitary functions via local interaction functions (e.g. as in propositions 4.7 and 4.8) to control the growth, change and development of the network.

Now we select a state q of the finite cell-cycle automaton $\mathcal{C}$ corresponding to cell division (for the above Petri net a many-token starting state would be needed, and division would be triggered by the amount of cdc2-P tokens exceeding a threshold). A level 1 interaction machine consists of just the automaton $\mathcal{C}$ at time zero (corresponding to the empty word), and is defined by adding a new copy of $\mathcal{C}$ connected to the original one (its 'mother') whenever the cdc2-P threshold is reached, giving rise to cell division with two automata in appropriate states. We might suppose that the state of this new automaton either is determined by external inputs or is the same as that of its mother, or for mathematical convenience that the state is given by some function of a clock so that all states can occur as initial states for some division.

Now we have a very different situation regarding transients. Each cell $\mathcal{C}$ has a transformation semigroup with transient properties. Long sequences of inputs, unless they correspond exactly to words in a permutator semigroup $(X)\mathbf{per}_{str}$, will 'knock the cell out of' any given pool of reversibility (natural subsystem with state set X), until $\mathcal{C}$ is at a lowest level in the weak control hierarchy after all transient behaviour has finished. However, the interaction machine $\mathfrak{A}$ (which we may suppose has inputs modelling nutrients reaching the cell $\mathcal{C}$ triggering transition sequences according to some interaction functions φ_v that eventually cause the cell to divide) will repeatedly give rise to copies of $\mathcal{C}$ in any ensemble state. Thus, while the behaviour of any given constituent cell is transient, the dynamically growing ensemble will repeatedly visit all the natural subsystems with their associated permutation groups $(X)\mathbf{per}$.

[39] A dynamic variant (level 1 interaction machine) would allow the creation of new instances of the Krebs cycle or other finite automata components depending on the metabolic and genetic regulatory activity modelled in the network.

It is easy to construct elaborations of this model with many other constituent automata within each cell $\mathcal{C}$. For example, we may expand $\mathcal{C}$ to include metabolism and p53 genetic regulatory control in a more sophisticated model of the cell. Moreover, based on local conditions, external input and intercellular signalling, the model cells of the ensemble exhibit differentiation into cell types in different ways, e.g. activating different parts of their genome corresponding to different cell types (e.g. neurons, muscle cells, etc.) in a division of labour.

Now changes in the number of cells occur during multicellular differentiation and development, and discrete automata models of this process within cells are increasingly used (e.g. [102,103]) to understand morphogenesis of living organisms. It is clear that, within the framework of interaction machines, cell number increases and decreases, as well as changes in topology and differentiation of cells and ensembles of cells into different types, can be modelled naturally. A higher level interaction machine elaboration could organize cells into constituents that are themselves *organs*, modelled as constituent interaction machines in a multicellular body that interact with each other and the environment to constitute a higher level individual. In such a setting, considerations of robustness and self-reproduction [104,105], self-maintenance and self-repair (more generally, *autopoiesis* [106], i.e. processes of 'self-production' including constructing the machinery for producing the cell's components) can naturally be modelled for natural and robust, artificial systems.

(c) Layers of time: recurrence and dynamic coordinates with natural systems as building blocks

> One generation passes away, and another generation comes [...]. The sun also arises, and the sun goes down, and hastens to his place where he arose.
>
> —Ecclesiastes

We now examine how any discrete dynamical system given by an automaton $\mathcal{A}$ can be synthesized using ensembles of natural subsystems according to the weak control hierarchy as an interaction machine, as either a static or dynamic cascade (theorems 6.1 and 6.2). For any permutation group (X, G), such as a natural subsystem $(X, (X)\mathbf{per})$ of $\mathcal{A}$, let $(X, \bar{G})$ denote the transformation semigroup obtained from (X, G) by including all constant maps on X, i.e. all $c\colon X \to X$ of the form $c(x) = x_0 \in X$ for all $x \in X$. Each element of $\bar{G}$ is either a permutation or constant. Here we allow trivial permutator groups $|(X)\mathbf{per}| = 1$. For $X \subseteq Q = States(\mathcal{A})$, let $Tiles(X)$ be the maximal (under inclusion) subsets of X which either are singletons or occur as *images* of Q, i.e. $X = Q \cdot w$ for some $w \in A^*$. Subduction for images is defined by: $Y \leq_S X$ if there exists $s \in M(\mathcal{A})$ with $Y \subseteq X \cdot s$. Subduction equivalence is the same as $\mathcal{R}$-equivalence. It is clear that the partial order $(X, (X)\mathbf{per}) \prec_{wc} (Y, (Y)\mathbf{per})$ implies the same ordering in subduction. Note that subduction dominance may be strict even if X and Y have the same cardinality, which is not possible for weak control. Now $\bigcup Tiles(X) = X$ and it is easy to show that each element of $(X)\mathbf{per}$ permutes $Tiles(X)$.[40] Moreover, each $s \in S(\mathcal{A})$ either permutes $Tiles(X)$ or there exists $T \in Tiles(X)$, with $X \cdot s \subseteq T$. Consider $\mathcal{A}$'s weak control hierarchy $\Xi/\mathcal{R}$ giving a partial order $\prec_{wc}$ on equivalence classes of natural subsystems $(X, (X)\mathbf{per})$. Expand this to include $\mathcal{R}$-equivalence classes of $(X, (X)\mathbf{per})$ for all X images of Q, even if X is not the image of an idempotent in $S(\mathcal{A})$. Add new inequalities for subduction modulo $\mathcal{R}$-equivalence. Thus, subduction modulo $\mathcal{R}$ is a partial order that includes weak control modulo $\mathcal{R}$. Finally, remove all the classes for $(X, (X)\mathbf{per})$ if X is a singleton, to obtain a partial order $\prec_{wc^+}$ on non-singleton image sets X modulo $\mathcal{R}$-equivalence.

Theorem 6.1 (weak control coordinatization by ensemble of natural subsystems). *Let $\mathcal{A}$ be a finite automaton with characteristic monoid $M(\mathcal{A})$. Then the partial order $\prec_{wc^+}$ indexes an ensemble, i.e. a cascade, of $(Tiles(X), \overline{(X)\mathbf{per}})$ that emulates $\mathcal{A}$. Moreover, each constituent $(Tiles(X), \overline{(X)\mathbf{per}})$ either (1) comes from a natural subsystem $(X, (X)\mathbf{per})$ or (2) has a trivial group $(X)\mathbf{per}$; in the latter case*

[40] This action might not be faithful. The group $(X)\mathbf{per}$ modulo the kernel of the action is called the *holonomy group* $(X)\mathbf{hol}$ of X, and one can replace $(X)\mathbf{per}$ in the components in the argument by the smaller $(X)\mathbf{hol}$.

(Tiles(X), $\overline{(X)\mathbf{per}}$) contains only the identity and constant maps. Finally, each of these components can be replaced by parallel copies of $(X, \overline{(X)\mathbf{per}})$.

Proof. (Outline) This is based on a modification of the holonomy theorem. (See [9, ch. 3] or [13] for the holonomy group and decomposition theorem.) By simple modifications of standard proofs of the holonomy theorem [9, ch. 3], consider all X that are non-singleton images X of Q under some $a \in A^*$. Then subduction modulo $\mathcal{R}$ includes $\Xi/\mathcal{R}$, the weak control partial order on $\mathcal{R}$-equivalence classes of image sets. This gives the partial order $\prec_{wc^+}$ of these equivalence classes, ordered according to the subduction order that includes all weak control relations. The directed network with this ordering of constituents $(Tiles(X), \overline{(X)\mathbf{per}})$ is then a cascade network that emulates the automaton $\mathcal{A}$ that we started with, as its constituents map onto the holonomy permutation groups with constants $(Tiles(X), \overline{(X)\mathbf{hol}})$. But each $(Tiles(X), (X)\mathbf{per})$ can be replaced by a parallel ensemble of $|Tiles(X)|$ copies of $(X, (X)\mathbf{per})$ as the former embeds in the latter. ■

This shows how to reconstruct $\mathcal{A}$ from its component natural subsystems $(X, (X)\mathbf{per})$, using copies of only one per equivalence class, augmented with constants for locally good control. The network is a cascade (and can be made linear if desired) as we started with a partial order.

Now if the automaton $\mathcal{A}$ can occur in a recurring way in some interaction machine, theorem 6.1 gives it a hierarchical coordinate system in which all the natural subsystem representatives may occur arbitrarily often (with changing, i.e. non-constant, values). That is, transience recurs and may be usefully coordinatized via computation in natural subsystems.

We also have an interaction machine result (a new version of the holonomy decomposition): theorem 6.2 shows that hierarchical coordinatization (i.e. cascade decomposition) can be done with a growing changing interaction machine that is a dynamic cascade of natural subsystems based on the weak control hierarchy. Just like our decimal example (§5c), this interaction machine gives a coordinate system that grows as needed.

Theorem 6.2 (emulation via a growing dynamic cascade of natural subsystems). *Let $\mathcal{A}$ be a finite automaton. Then $\mathcal{A}$ is emulated by an interaction machine $\mathfrak{A}$ whose constituents at every given time $t \in A^*$ are a collection of representatives $(X, \overline{(X)\mathbf{per}})$ that are either natural subsystems of $\mathcal{A}$ or, if not, $(X)\mathbf{per}$ is a trivial group. The digraph of the interaction machine at time t is a suborder of the partial order $\prec_{wc^+}$ (but with some nodes expanded in parallel for multiple copies of the same component) changing as a function of time and configuration, so $\mathfrak{A}$ comprises a dynamic growing cascade.*

Proof. (Outline) From the proof of the holonomy theorem in [9, ch. 3] and so also for the emulating ensemble in theorem 6.1, it is clear that, in lifting a state for the emulation, only certain components are used (i.e. have non-trivial values). Let these components be present with the topology given as a suborder of $\prec_{wc+}$ but expanding some nodes by parallel copies as in the proof of theorem 6.1. Then using the interaction machine notion, when a lifted input symbol $\tilde{a}$ corresponding to $a \in A$ acts on the machine, all present components are updated according to the usual cascade formulation, but those constituents where a value should fall that have not been set before come into existence at that moment with the appropriate state value. This continues as each input symbol acts, resulting in a growing cascade. ■

Remarks on recurrence. Transience of the natural subsystems $(X, (X)\mathbf{per})$ of $\mathcal{A}$ is a natural consequence of the fact that semigroups allow for irreversible change. However, with the notion of an interaction machine as formulated here and an ensemble approach in which new instances of $\mathcal{A}$ may continually arise, this transience itself becomes a recurring phenomenon, as do instances of the natural subsystems given by its permutation groups $(X, (X)\mathbf{per})$. Semigroups such as the natural numbers $\mathbb{N}$ or real numbers $\mathbb{R}$ are taken in physics as models of time. But more generally (see footnote 33), we can consider A^*, the free semigroup over an alphabet A of basic events $a \in A$, or the semigroup $S(\mathcal{A})$ of a discrete dynamical system $\mathcal{A}$, as its appropriate model of time. In an interaction machine $\mathfrak{A}$, for different times $t \in A^*$, a new copy of $\mathcal{A}$ may repeatedly arise as a component of $\mathfrak{A}^{(t)}$. The semigroup $S(\mathcal{A})$ and the natural subsystems $(X, (X)\mathbf{per})$ of $\mathcal{A}$ may be repeatedly encountered. Recurrence of full possibilities of the full state space of $\mathcal{A}$ (due to

growing and changing dynamics of $\mathfrak{A}$ allowing the reproduction of more copies of the system) must now be considered for a full understanding of systems maintained far from thermodynamic equilibrium. Similarly, the *weak control hierarchy is repeatedly encountered*. In the same way, in an interaction machine of level greater than 1, its constituent interaction machines may repeatedly arise in this way too, so that this kind of recurrent structure can be layered or 'nested'. ■

(d) Discussion

Symmetry appears in the notion of natural subsystems as a particular kind of permutation group in the transformation semigroup of an automaton. These symmetry structures are related by hierarchical relations of weak control and essential dependencies between natural subsystems as introduced here. We have explored the algebraic and symmetry structures for various discrete biological models in detail and described their weak control hierarchies in detail. In particular, biological systems (e.g. the p53–mdm2 genetic regulatory pathway) may have symmetry structures that are capable of a finitary analogue of universal computation (SNAGs). The natural symmetry structures within these discrete dynamical systems are generally present in very many copies, and are related to each other via a notion of weak control and $\mathcal{R}$-equivalence, which allows us to see hidden hierarchical structure from both global and local perspectives. These relations can be used to reconstruct the system in an ensemble using just representative copies of its natural subsystems, in a static or growing dynamic cascade.

Dynamic cascades are a case of level 1 interaction machines, introduced here, which generalize static automata networks and can be used to naturally model systems that we see all around us, e.g. in our capacity to use growing or shrinking decimal representation of numbers. Iterating this insight, we introduced general interaction machines as dynamically changing networks of interaction machines whose components are themselves interaction machines. These allow the natural formulation of a new class of discrete biological models, e.g. for growing, differentiating multicellular systems with recurring division of cells based on the cell cycle. Interaction machines give a model of computation close to what biological systems do based on interactions. Furthermore, this work lays the foundation for us to apply mathematical tools to understand and design dynamically changing computational structures that respond to each other and the environment.

In natural examples, multiple instances of these natural subsystems may continually arise and later be destroyed or renewed, giving them a recurrent presence in ensembles. Considering networks of automata, including dynamic automata networks and more general interaction machines, that can grow or change their components and topologies like living things do, or that can deploy multiple copies of components in suitable configurations (as in differentiated multicellular development of living things), suggests that the powerful computational capacity of symmetry structures that we find within models of biological systems may drive, control and shape their activity in response to the environment and each other, in a manner where their activity keeps them far from thermodynamic equilibrium ('death'). Indeed, we obtained results showing how genetic regulatory control (or other methods) could harness the finitary universal computational power of SNAGs.

Competing interests. We declare we have no competing interests.

Funding. The research by the authors leading to these results was funded in part by the European Union's Seventh Framework Programme (FP7/2007-2013) under the BIOMICS project, grant agreement no. 318202. G.H. was partially supported by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences, and by the Hungarian Scientific Research Fund (OTKA) grant no. K109185.

References

1. Chaouiya C, Remy E, Thieffry D. 2008 Petri net modelling of biological regulatory networks. *J. Discrete Algorithms* **6**, 165–177. (doi:10.1016/j.jda.2007.06.003)

2. Chaouiya C *et al.* 2013 SBML qualitative models: a model representation format and infrastructure to foster interactions between qualitative modelling formalisms and tools. *BMC Syst. Biol.* **7**, 135. (doi:10.1186/1752-0509-7-135)
3. Fauré A, Naldi A, Chaouiya C, Thieffry D. 2006 Dynamical analysis of a generic Boolean model for the control of the mammalian cell cycle. *Bioinformatics* **22**, e124–e131. (doi:10.1093/bioinformatics/btl210)
4. Kauffman SA. 1969 Metabolic stability and epigenesis in randomly connected genetic nets. *J. Theoret. Biol.* **22**, 437–467. (doi:10.1016/0022-5193(69)90015-0)
5. Marwan W, Sujatha A, Starostzik C. 2005 Reconstructing the regulatory network controlling commitment and sporulation in *Physarum polycephalum* based on hierarchical Petri net modelling and simulation. *J. Theoret. Biol.* **236**, 349–365. (doi:10.1016/j.jtbi.2005.03.018)
6. Thomas R, d'Ari. R 1990 *Biological feedback*. Boca Raton, FL: CRC Press.
7. Egri-Nagy A, Nehaniv CL. 2008 Algebraic properties of automata associated to Petri nets and applications to computation in biological systems. *BioSystems* **94**, 135–144. (doi:10.1016/j.biosystems.2008.05.019)
8. Arbib MA (ed.). 1968 *Algebraic theory of machines, languages, and semigroups*. New York, NY: Academic Press.
9. Dömösi P, Nehaniv CL. 2005 *Algebraic theory of finite automata networks: an introduction*. SIAM Series on Discrete Mathematics and Applications, vol. 11. Philadelphia, PA: Society for Industrial and Applied Mathematics.
10. Holcombe W. 1982 *Algebraic automata theory*. Cambridge, UK: Cambridge University Press.
11. Rhodes J. 2010 *Applications of automata theory and algebra via the mathematical theory of complexity to biology, physics, psychology, philosophy, and games*. Singapore: World Scientific Press. Foreword by MW Hirsch, edited by CL Nehaniv. (Original version: University of California at Berkeley, Mathematics Library, 1971.)
12. Rhodes J, Steinberg B. 2008 *The q-theory of finite semigroups*. Berlin, Germany: Springer.
13. Eilenberg S. 1976 *Automata, languages, and machines, vol. B*. New York, NY: Academic Press.
14. Paul Zeiger H. 1967 Cascade synthesis of finite state machines. *Information and Control* **10**, 419–433. (Erratum: 11:471, 1967.) (doi:10.1016/S0019-9958(67)90228-8)
15. Schrödinger E. 1967 *Statistical thermodynamics*. Cambridge, UK: Cambridge University Press.
16. Egri-Nagy A, Nehaniv CL. 2010 `SgpDec`—software package for hierarchical coordinatization of groups and semigroups, implemented in the `GAP` computer algebra system, v. 0.7.29. See http://sgpdec.sf.net.
17. Egri-Nagy A, Mitchell JD, Nehaniv CL. 2014 SgpDec: cascade (de)compositions of finite transformation semigroups and permutation groups. In *Mathematical Software—ICMS 2014*. Lecture Notes in Computer Science, vol. 8592, pp. 75–82. Berlin, Germany: Springer.
18. Egri-Nagy A, Nehaniv CL. 2004 Algebraic hierarchical decomposition of finite state automata: comparison of implementations for Krohn–Rhodes theory. In *Implementation and Application of Automata: 9th Int. Conf., CIAA 2004, Kingston, Canada, 22–24 July 2004*. Lecture Notes in Computer Science, vol. 3317, pp. 315–316. Berlin, Germany: Springer Verlag.
19. Nehaniv CL, Quick P. 2007 Emulation of synchronous automata networks with dynamically changing topologies by asynchronous automata networks. In *Proc. 7th International Workshop on Information Processing in Cells and Tissues* (*IPCAT 2007*), *Oxford, UK, 29–31st August 2007*, pp. 20–29. Brno, Czech Republic: Tribun EU.
20. Gécseg F. 1986 *Products of automata.* EATCS Monographs in Theoretical Computer Science 7. Berlin, Germany: Springer.
21. Krohn K, Rhodes J. 1965 Algebraic theory of machines. I. Prime decomposition theorem for finite semigroups and machines. *Trans. Am. Math. Soc.* **116**, 450–464. (doi:10.1090/S0002-9947-1965-0188316-1)
22. Gross T, Sayama H. 2009 *Adaptive networks: theory, models and applications*. New England Complex Systems Institute Studies on Complexity. Berlin, Germany: Springer.
23. Sayama H. 2007 Generative network automata: a generalized framework for modeling complex dynamical systems with autonomously varying topologies. In *Proc. 1st IEEE Symp. on Artificial Life* (*IEEE ALIFE 2007*), *Honolulu, HI, 1–5 April 2007*, pp. 214–221. Piscataway, NJ: IEEE.
24. Sayama H, Laramee C. 2009 Generative network automata: a generalized framework for modeling adaptive network dynamics using graph rewritings. In *Adaptive networks: theory, models and applications* (eds T Gross, H Sayama), pp. 311–332. Berlin, Germany: Springer.

25. Clifford AH, Preston GB. 1967 *The algebraic theory of semigroups*, vol. 1, 2nd edn. Mathematical Surveys, no. 7. Providence, RI: American Mathematical Society.
26. Howie JM. 1995 *Fundamentals of semigroup theory*. London Mathematical Society Monographs New Series, vol. 12. Oxford, UK: Oxford University Press.
27. Lallement G. 1979 *Semigroups and combinatorial applications*. New York, NY: Wiley.
28. Ptashne M. 1992 *A genetic switch: λ and higher organisms*, 2nd edn. Oxford, UK: Blackwell Publishers.
29. Ptashne M, Gann A. 2001 *Genes and signals*. Cold Spring Harbour, NY: Cold Spring Harbor Laboratory Press.
30. Egri-Nagy A, Nehaniv CL. 2008 Hierarchical coordinate systems for understanding complexity and its evolution with applications to genetic regulatory networks. *Artif. Life* **14**, 299–312. (doi:10.1162/artl.2008.14.3.14305)
31. Egri-Nagy A, Nehaniv CL. 2011 On straight words and minimal permutators in finite transformation semigroups. In *Implementation and Application of Automata: 15th Int. Conf., CIAA 2010, Manitoba, Canada, 12–15 August 2010* (eds M Domaratzki, K Salomaa). Lecture Notes in Computer Science, vol. 6482, pp. 115–124, Berlin, Germany: Springer.
32. Margulis L, Sagan D. 2002 *Early life: evolution on the precambrian Earth*, 2nd edn. Sudbury, MA: Jones and Bartlett.
33. Berg JM, Tymoczko JL, Stryer L. 2006 *Biochemistry, 6th edn, international version*. New York, NY: W. H. Freeman.
34. Egri-Nagy A, Nehaniv CL, Rhodes JL, Schilstra MJ. 2008 Automatic analysis of computation in biochemical reactions. *BioSystems* **94**, 126–134. (doi:10.1016/j.biosystems.2008.05.018)
35. Egri-Nagy A, Dini P, Nehaniv CL, Schilstra MJ. 2010 Transformation semigroups as constructive dynamical spaces. In *Digital ecosystems*. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 67, pp. 245–265. Berlin, Germany: Springer.
36. Van Leeuwen I, Munro AJ, Sanders I, Staples O, Lain S. 2010 Numerical and experimental analysis of the p53–mdm2 regulatory pathway. In *Digital Ecosystems: Third International Conference, OPAALS 2010, Aracuju, Sergipe, Brazil, 22–23 March 2010* (eds FAB Colugnati, LCR Lopes, SFA Barretto), pp. 266–284. Lecture Notes of the Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering, vol. 67. Aracaju, Sergipe, Brazil: Springer.
37. Egri-Nagy A, Nehaniv CL, Schilstra MJ. 2009 Symmetry groups in biological networks. *8th Information Processing in Cells and Tissues Conf.* (*IPCAT09*), *Ascona, Switzerland, 5–9 April 2009*. Journal preprint.
38. Kastan MB, Kuerbitz SJ. 1993 Control of G1 arrest after DNA damage. *Environ. Health Perspect.* **101**, 55–58.
39. Dini P, Egri-Nagy A, Nehaniv CL, Schilstra MJ, VanLeeuwen I, Munro AJ, Lain S. 2010 *D1.4: Mathematical models of gene expression computing*. OPAALS Deliverable, see http://www.lse.ac.uk/media@lse/research/OPAALS/D01.4_Mathematical_Models_of_Gene_Expression_Computing.pdf.
40. Petri CA. 1962 *Kommunikation mit Automaten*. Schriften des IIM 2. Bonn, Germany: Institut für Instrumentelle Mathematik.
41. Reisig. W 2011 *Petri nets: an introduction*. Berlin, Germany: Springer.
42. Brown CT *et al.* 2002 New computational approaches for analysis of cis-regulatory networks. *Dev. Biol.* **246**, 86–102. (doi:10.1006/dbio.2002.0619)
43. Horváth G. 2008 Functions and polynomials over finite groups from the computational perspective. PhD dissertation, University of Hertfordshire, Hatfield, UK.
44. Krohn K, Maurer WD, Rhodes J. 1966 Realizing complex Boolean functions with simple groups. *Inform. Control* **9**, 190–195. (doi:10.1016/S0019-9958(66)90229-4)
45. Maurer WD, Rhodes JL. 1965 A property of finite simple non-abelian groups. *Proc. Am. Math. Soc.* **16**, 552–554. (doi:10.1090/S0002-9939-1965-0175971-0)
46. Jones HF. 1998 *Groups, representations and physics*, 2nd edn. Amsterdam, The Netherlands: IOP Publishing.
47. Gell-Mann M, Ne'eman Y. 1964 *The eightfold way*. New York, NY: Benjamin.
48. Reboredo FA, Galli G. 2005 Theory of alkyl-terminated silicon quantum dots. *J. Phys. Chem. B* **109**, 1072–1078. (doi:10.1021/jp0462254)

49. Szuchmacher Blum A *et al.* 2006 Templated self-assembly of quantum dots from aqueous solution using protein scaffolds. *Nanotechnology* **17**, 5073–5079. (doi:10.1088/0957-4484/17/20/006)
50. Raman KV. 2004 *Group theory and its applications to chemistry*. New Delhi, India: Tata McGraw-Hill.
51. Backhouse NB, Gard P. 1974 The representation theory of the icosahedral group. *J. Phys. A: Math. Nucl. Gen.* **7**, 2101–2108. (doi:10.1088/0305-4470/7/17/004)
52. Plakhutin BN, Carbó-Dorca R. 2000 Icosahedral symmetry structures with open-shell electronic configuration h^N $(N = 1 - 9)$. *Phys. Lett. A* **267**, 370–378. (Erratum: 279:102–103, 2001.) (doi:10.1016/S0375-9601(00)00142-0)
53. Wegener I 1987 *The complexity of Boolean functions*. Stuttgart, Germany: John Wiley & Sons Ltd, B. G. Teubner.
54. Werner H. 1978 *Einführung in die allgemeine algebra*. Mannheim, Germany: Bibliographisches Institut.
55. Nipkow T. 1990 Unification in primal algebras, their powers and their varieties. *J. Assoc. Comput. Machinery* **37**, 742–776. (doi:10.1145/96559.96569)
56. Agou SJ, Deléglise M, Nicolas J-L. 2003 Short polynomial representations for square roots modulo *p*. *Des. Codes Cryptogr.* **28**, 33–44. (doi:10.1023/A:1021819602497)
57. Spector L, Clark DM, Lindsay I, Barr B, Klein J. 2008 Genetic programming for finite algebras. In *Proc. Genetic and Evolutionary Computation Conference* (*GECCO-2008*), pp. 1291–1298. New York, NY: ACM Press.
58. Scott SD. 1969 The arithmetic of polynomial maps over a group and the structure of certain permutational polynomial groups. I. *Monatsh. Math.* **73**, 250–267. (doi:10.1007/BF01300543)
59. Horváth G, Nehaniv CL. In press. Length of polynomials over finite groups. *J. Comp. Syst. Sci.*
60. Fröhlich A. 1958 The near-ring generated by the inner automorphisms of a finite simple group. *J. London Math. Soc.* **33**, 95–107.
61. Michler G. 2006 *Theory of finite simple groups*, vol. 1. Cambridge, UK: Cambridge University Press.
62. Barrington DA. 1989 Bounded-width polynomial-size branching programs recognize exactly those languages in NC^1. *J. Comput. System Sci.* **38**, 150–164. (doi:10.1016/0022-0000(89)90037-8)
63. Horváth G, Nehaniv CL, Szabó Cs. 2008 An assertion concerning functionally complete algebras and NP-completeness. *Theoret. Comput. Sci.* **407**, 591–595. (doi:10.1016/j.tcs.2008.08.028)
64. Horváth G, Szabó Cs. 2011 The extended equivalence and equation solvability problems for groups. *Discrete Math. Theor. Comput. Sci.* **13**, 23–32.
65. Horváth G, Szabó Cs. 2012 Equivalence and equation solvability problems for the group A_4. *J. Pure Appl. Algebra* **216**, 2170–2176. (doi:10.1016/j.jpaa.2012.02.007)
66. Brawand D *et al.* 2011 The evolution of gene expression levels in mammalian organs. *Nature* **478**, 343–348. (doi:10.1038/nature10532)
67. Indjeian V. 2014 Gene regulatory changes in fish and human skeletal evolution. Centre for Ecology and Evolution (CEE) Autumn Symposium on the Evolution of Gene Regulation, 3 November 2014.
68. Gaunt SJ, Paul Y-L. 2012 Changes in cis-regulatory elements during morphological evolution. *Biology* **1**, 557–574. (doi:10.3390/biology1030557)
69. McLean CYF. 2011 Computational analysis of the mammalian cis-regulatory landscape. PhD thesis, Stanford University, USA.
70. McLean CY *et al.* 2011 Human-specific loss of regulatory DNA and the evolution of human-specific traits. *Nature* **471**, 216–219. (doi:10.1038/nature09774)
71. Nehaniv CL, Wagner GP. 2000 The right stuff: appropriate mathematics for evolutionary and developmental biology. *Artif. Life* **6**, 1–2. (doi:10.1162/106454600568285)
72. Rosen R. 1958 A relational theory of biological systems. *Bull. Math. Biophys.* **20**, 245–260. (doi:10.1007/BF02478302)
73. Rosen R. 1958 The representation of biological systems from the standpoint of the theory of categories. *Bull. Math. Biophys.* **20**, 317–341. (doi:10.1007/BF02477890)
74. Rosen R. 1959 A relational theory of biological systems II. *Bull. Math. Biophys.* **21**, 109–128. (doi:10.1007/BF02476354)
75. Landauer C, Bellman KL. 2002 Theoretical biology: organisms and mechanisms. In *Proc. 5th Int. Conf. on Computing Anticipatory Systems: CASYS 2001*, vol. 627 (ed. DM Dubois). AIP Conference Proceedings, pp. 59–70. Melville, NY: American Institute of Physics.

76. Ikegami T. 1999 Evolvability of machines and tapes. *Artif. Life Robot.* **3**, 242–245. (doi:10.1007/BF02481188)
77. Sirmai J. 2013 Autopoiesis facilitates self-reproduction. *Adv. Artif. Life* **12**, 417–424.
78. Centler F, Kaleta C, di Fenizio PS, Dittrich P. 2008 Computing chemical organizations in biological networks. *Bioinformatics* **24**, 1611–1618. (doi:10.1093/bioinformatics/btn228)
79. Dittrich P. 2009 Artificial chemistry. In *Encyclopedia of complexity and system science* (ed. B Meyers), pp. 113–136. Berlin, Germany: Springer.
80. Dittrich P, Speroni di Fenizio P. 2007 Chemical organization theory. *Bull. Math. Biol.* **69**, 1199–1231. (doi:10.1007/s11538-006-9130-8)
81. Dittrich P. 2005 Chemical computing. In *Unconventional programming paradigms: International Workshop UPP 2004* (eds J-P Banâtre, J-L Giavitto, P Fradet, O Michel), pp. 19–32. Lecture Notes in Computer Science 3566. Berlin, Germany: Springer.
82. Dittrich P, Ziegler J, Banzhaf W. 2001 Artificial chemistries—a review. *Artif. Life* **7**, 225–275. (doi:10.1162/106454601753238636)
83. Matsumaru N, Centler F, Speroni di Fenizio P, Dittrich P. 2007 Chemical organization theory as a theoretical base for chemical computing. *Int. J. Unconventional Comput.* **3**, 285–309.
84. Calude CS, Paun G. 2001 *Computing with cells and atoms: an introduction to quantum, DNA and membrane computing*. London, UK: Taylor & Francis.
85. Paun G. 2003 Membrane computing. In *Fundamentals of computation theory* (eds A Lingas, BJ Nilsson), pp. 284–295. Lecture Notes in Computer Science, vol. 2751. Berlin, Germany: Springer.
86. Paun G, Rozenberg G, Salomaa A. 2009 *The Oxford handbook of membrane computing*. Oxford, UK: Oxford University Press.
87. Milner R. 1999 *Communicating and mobile systems: the π-calculus*. Cambridge, UK: Cambridge University Press.
88. Phillips A, Cardelli L. 2007 Efficient, correct simulation of biological processes in the stochastic pi-calculus. In *Computational methods in systems biology* (eds M Calder, S Gilmore) pp. 184–199. Berlin, Germany: Springer.
89. Phillips A, Cardelli L, Castagna G. 2006 A graphical representation for biological processes in the stochastic pi-calculus. In *Transactions on computational systems biology VII* (ed. C Priami), pp. 123–152. Berlin, Germany: Springer.
90. Regev A, Panina EM, Silverman W, Cardelli L, Shapiro E. 2004 BioAmbients: an abstraction for biological compartments. *Theoret. Comp. Sci.* **325**, 141–167. (doi:10.1016/j.tcs.2004.03.061)
91. Fauré A, Naldi A, Lopez F, Chaouiya C, Ciliberto A, Thieffry D. 2009 Modular logical modelling of the budding yeast cell cycle. *Mol. Biosyst.* **5**, 1787–1796. (doi:10.1039/b910101m)
92. Naldi A, Bérenguier D, Fauré A, Lopez F, Thieffry D, Chaouiya C. 2009 Logical modelling of regulatory networks with GINsim 2.3. *BioSystems* **97**, 134–139. (doi:10.1016/j.biosystems.2009.04.008)
93. Nehaniv CL, Rhodes JL. 2000 The evolution and understanding of hierarchical complexity in biology from an algebraic perspective. *Artif. Life* **6**, 45–67. (doi:10.1162/106454600568311)
94. Faltin F, Metropolis N, Ross B, Rota G-C. 1975 The real numbers as a wreath product. *Adv. Math.* **16**, 278–304. (doi:10.1016/0001-8708(75)90115-2)
95. Zhang J, Norman DA. 1995 A representational analysis of numeration systems. *Cognition* **57**, 271–295. (doi:10.1016/0010-0277(95)00674-3)
96. Nehaniv CL. 1993 The algebra of time. In *Proc. National Conf. of the Japan Society for Industrial and Applied Mathematics*, pp. 127–128. Tokyo, Japan: Japan Society for Industrial and Applied Mathematics.
97. Aczel P. 1988 *Non-well-founded sets*. CSLI Lecture Notes 14. Chicago, IL: University of Chicago Press.
98. Goguen JA. 1972 Realization is universal. *Math. Syst. Theory* **6**, 359–374. (doi:10.1007/BF01843493)
99. Rutten JJMM. 2000 Universal coalgebra: a theory of systems. *Theor. Comput. Sci.* **249**, 3-–80. (doi:10.1016/S0304-3975(00)00056-6)
100. Nicolis G, Prigogine I. 1977 *Self-organization in non-equilibrium systems*. New York, NY: John Wiley.
101. Tyson JJ. 1991 Modelling the cell division cycle: cdc2 and cyclin interactions. *Proc. Natl Acad. Sci. USA* **88**, 7328–7332. (doi:10.1073/pnas.88.16.7328)

102. Akam M. 2014 The evolution of the segmentation gene network in arthropods. Centre for Ecology and Evolution (CEE) Autumn Symposium on the Evolution of Gene Regulation, 3 November 2014. (Modelling work of Erik Clark.)
103. González A, Chaouiya C, Thieffry D. 2008 Logical modelling of the role of the Hh pathway in the patterning of the *Drosophila* wing disc. *Bioinformatics* **24**, i234–i240. (doi:10.1093/bioinformatics/btn266)
104. von Neumann J. 1956 Probabilistic logics and the synthesis of reliable organisms from unreliable components. In *Automata studies* (eds CE Shannon, J McCarthy), pp. 43–98. Annals of Mathematical Studies, vol 34. Princeton, NJ: Princeton University Press.
105. von Neumann J 1966 *Theory of self-reproducing automata*. Champaign, IL: University of Illinois Press. Edited and completed by Arthur W. Burks.
106. Varela FG, Maturana HR, Uribe R. 1974 Autopoiesis: the organization of living systems, its characterization and a model. *BioSystems* **5**, 187–196. (doi:10.1016/0303-2647(74)90031-8)