Submitted to Physica D, Proceedings of the
International Workshop on Lattice Dynamics.

Santa Fe Institute Working Paper 95–10–095

Computational Mechanics of Cellular Automata: An Example

James E. Hanson
Santa Fe Institute
1399 Hyde Park Rd.
Santa Fe, NM 85701 USA
hanson@santafe.edu
http://www.santafe.edu/~hanson

James P. Crutchfield
Department of Physics
University of California
Berkeley, CA 94720 USA
chaos@gojira.berkeley.edu
http://www.santafe.edu/~jpc

Abstract

We illustrate and extend the techniques of computational mechanics in explicating the structures that emerge in the space-time behavior of elementary one-dimensional cellular automaton rule 54. The CA's dominant regular domain is identified and a domain filter is constructed to locate and classify defects in the domain. The primary particles are identified and a range of interparticle interactions is studied. The deterministic equation of motion of the filtered space-time behavior is derived. Filters of increasing sophistication are constructed for the efficient gathering of particle statistics and for the identification of higher-level defects, particle interactions, and secondary domains. We define the emergence time at which the space-time behavior condenses into configurations consisting only of domains, particles, and particle interactions. Taken together, these techniques serve as the basis for the investigation of pattern evolution and self-organization in this representative system.

1. Patterns and computation in cellular automata

After more than a decade of extensive research, it has become a commonplace observation that one-dimensional cellular automata (CA) exhibit a striking—and often perplexing—diversity of spatio-temporal behavior. The early empirical categorization of space-time patterns into four "classes" [1]—loosely based on an analogy with those found in continuous-state dynamical systems—has resisted numerous attempts at formalization. In many CA, it is immediately evident that the system self-organizes into some type of emergent pattern. In other CA, the structure or even existence of an emergent pattern is less clear. The question that naturally arises therefore is how to characterize the spatio-temporal patterns that emerge during the CA's evolution. If such a characterization is possible, it can be used as a basis for numerical and analytical tools that discover, analyze, and filter patterns that emerge in CA.

In refs. [2–5], such methods were developed under the general name of computational mechanics. Computational mechanics is a synthesis of nonlinear dynamics and computation theory, which characterizes patterns and structure occurring in natural processes by means of formal models of computation. Connections between CA and computation theory have been an active area of research for some time.[6–8] One major theme of this research has centered around the problem of designing a CA to behave in some particular way, such as simulating a universal Turing machine,[9] exhibiting particles and computing with them,[10, 11] performing reliable computations in the presence of noise,[12] or a host of other tasks. Computational mechanics, in contrast, approaches CA more from the perspective of physical science. Rather than trying to engineer a CA rule table and construct an initial condition to satisfy some predefined criterion, computational mechanics attempts to discover and characterize the *typical* patterns occurring in a *given* CA.* In a well-defined sense, each CA specifies the local space-time equations of motion of a model universe and computational mechanics is addressed to the question, What are the "physical laws" and structures that emerge in the space-time behavior?

A distinction must be made here between "patterns" and "configurations". Throughout this article, a *pattern* is considered to be a set or ensemble of configurations sharing some common spatial structure. Pattern dynamics then concerns the evolution of sets of similar configurations, rather than of some particular configuration exemplifying the pattern. If the collection of all patterns is thought of as a *pattern space* in which each distinct pattern is a point, then pattern dynamics is the evolution of points in that space. In many CA, there are a few important patterns that dominate and organize the behavior of the system as a whole. These important patterns form a *pattern basis*, i.e., a set of fundamental components in terms of which all patterns in the system are analyzed.[2]

The purpose of this article is to illustrate the techniques of computational mechanics in a familiar setting. For this reason we have selected for investigation an elementary CA rule that has received attention in the past as one of the most interesting and "complex", and whose phenomenology has been at least partially described already.[14] As we will see, computational mechanics can place those results in a more general and more rigorous context, as well as opening avenues of further investigation. In selecting elementary CA 54 to investigate, therefore, we do not focus primarily on the results themselves, but on the methods of obtaining them. This will facilitate a comparison by the reader of the methods of computational mechanics with its alternatives.

In the next section we define cellular automata and the essential elements of computation theory. Section 3 outlines the central concepts of regular domain, of domain filtering, and of particle identification in arbitrary CA. Following this general review, section 4 describes the fundamental domain in ECA 54, constructs its domain filter, shows characteristic filtered and unfiltered space-time diagrams, and derives the deterministic rule governing the time-evolution filtered space-time behavior. Subsequent sections describe the primary particles observed in the filtered data, investigate the basic particle interactions, present numerical data on particle statistics, and begin the investigation of "higher-order" structures emerging out of the description of the system in terms of the fundamental particles.

*Computational mechanics is useful, however, for determining how computations are implemented in CA; e.g. see the analysis of evolved CA in ref. [13].

2. Cellular Automata and Computation Theory

2.1 Cellular automata

To fix notation, we briefly recall the definition of a cellular automaton. The *configuration* of the CA at time t, denoted $\mathbf{s}_t$, consists of a one-dimensional array or lattice of *sites* $\{i = 0, \ldots, N-1\}$ with values s_t^i chosen from a finite alphabet $\mathcal{A} = \{0, 1, ..., k-1\}$, so $\mathbf{s}_t \in \mathcal{A}^N$. The *local site-update function* ϕ operating on *neighborhoods* $\eta_t^i = s_t^{i-r} ... s_t^{i+r}$ of radius r is written $s_{t+1}^i = \phi\left(\eta_t^i\right)$. The list of all neighborhoods η and their corresponding outputs $\phi(\eta)$ is the CA's *lookup table*. The *global update operator* $\Phi : \mathcal{A}^N \to \mathcal{A}^N$ applies ϕ in parallel to all neighborhoods in the lattice; the CA's equation of motion is denoted $\mathbf{s}_{t+1} = \Phi(\mathbf{s}_t)$. For finite N, it is also necessary to specify the behavior at the edges of the array. In the following, we will always use periodic boundary conditions.

The pattern dynamics of a CA is investigated in terms of the CA *ensemble evolution operator*, denoted by $\mathbf{\Phi}$. For any set $\mathcal{L}_t = \{\mathbf{s}_t\}$ of configurations (or strings embedded in configurations), the iterate $\mathcal{L}_{t+1} = \mathbf{\Phi}\mathcal{L}_t$ is given by

$$\mathcal{L}_{t+1} = \{\Phi(\mathbf{s}_t) : \mathbf{s}_t \in \mathcal{L}_t\} \tag{1}$$

The *elementary* cellular automata (ECA) have $k = 2$, $r = 1$. For reference, we list the lookup table of the ECA 54, the rule under consideration here:

$$\phi(\eta) = \begin{cases} 0, & \eta \in \{111, 110, 011, 000\} \\ 1, & \eta \in \{101, 100, 010, 001\} \end{cases} \tag{2}$$

2.2 Computation theory

Before proceeding to the discussion of domains and domain filtering, it is necessary to review some rudiments of the theory of computation.[15] We will primarily be concerned with two varieties of *finite state machine*: *finite automata* (FAs) and *finite state transducers* (FSTs). We will refer to both FAs and FSTs by the generic term *machine*.

A finite automaton consists of a finite set of *states* and a *transition function* specifying directed paths between states. Any state may be a *start state*, an *accept state*, or both. Initially the machine is "in" one of its start states. During operation, it reads symbols one by one from some data source such as a CA configuration—or alternatively, it may generate symbols internally—and uses those symbols to make transitions from state to state according to the transition function. The structure of an FA is perhaps best understood by reference to its *graph*. The graph of a finite automaton represents its states as nodes and its transitions as edges labelled with symbols in the alphabet $\mathcal{A}$. An allowed path in the graph is a connected sequence of edges that begins at a start state and ends on an accept state. Each such path is associated with a sequence of symbols (a *word*) consisting of the labels on the successive edges traversed. For any given word ω, if there exists an allowed path corresponding to that word, then the FA is said to *accept* ω. If there is no such path, the FA *rejects* ω. The set of all words accepted by a given FA is the *language* accepted by that FA. The class of languages accepted by FAs is called the class of *regular languages*. The FA accepting a regular language $\mathcal{L}$ is denoted by $\mathrm{M}(\mathcal{L})$.

Finite state transducers are a variation of finite automata in which the edge labels consist of two parts: an *input* symbol and an *output* string of some length (including length 0). Each path in the graph of a finite state transducer is associated with a sequence of input symbols (the *input string)* and a sequence of outputs. During operation, the input string is read in the same way as in FAs; but in addition, the outputs are concatenated to construct an *output string*. In this way, an FST implements a particular mapping from strings to strings. The simple summary is that FAs represent sets, while FSTs represent functions.

An important subclass of finite automata is the class of *regular processes*. A regular process accepts a regular language with the following properties: (i) *subword closure:* all subwords of all words accepted by the language are themselves accepted; (ii) *dangler-free:* for every word $\omega \in \mathcal{L}$, there is at least one symbol $s \in \mathcal{A}$ such that ωs is also in $\mathcal{L}$; and (iii) the empty set (i.e., a string of zero length) is accepted. Each of these properties is reflected in the structure of the regular process itself. In particular, subword closure requires that all states be accept states and that from a start state there is a branching structure of states and edges devoted to *synchronization* (or *phase-locking*) of the machine to the underlying pattern. In the *process graph* of a regular process all states are both start and

accept states, and the synchronization states and edges are absorbed into the machine proper.

Finite state machines are appropriate for investigating pattern dynamics of CA for a number of reasons, among which we may note the following: (i) finite automata encompass the full range of behavior types from periodic to complex to random; (ii) characterization of patterns using FAs makes possible a definition of pattern complexity which is both natural and computable in practice; (iii) ensemble evolution in the space of regular languages is closed under the CA rule; (iv) the CA update rule is itself a finite state transducer; and (v) automated inference techniques exist for reconstructing finite automata from experimental data.[16, 17]

Another advantage of using finite automata is that the pattern dynamics of any CA, as represented by a time sequence of finite automata, can be explicitly calculated. If a pattern $\mathcal{L}_t$ is a regular language, then its iterate $\mathcal{L}_{t+1} = \mathbf{\Phi}\mathcal{L}_t$ is a regular language as well, by point (iii) above. More importantly, however, $\mathrm{M}(\mathcal{L}_{t+1})$ can be explicitly constructed by means of the "finite machine evolution" (FME) operator defined in ref. [2]. The FME operator makes it possible to trace the time evolution of any pattern representable by a finite automaton, even when that pattern subsumes an infinite number of words—as is usually the case.

3. Domains, Filters, and Particles

3.1 Domains

Of fundamental importance in the pattern dynamics of CA is the *regular domain*, introduced in ref. [2]. Informally, a regular domain is a spatially and temporally homogeneous pattern describable by a finite automaton—where "homogeneous" is to be understood in the sense of *having the same regularities*. It is a precise formulation of the rather vaguely defined term "domain" in physics that is generally used to denote some spatio-temporal region in which "things are basically the same". Well-known examples include ferromagnetic domains, in which all spins are oriented in the same direction, and convection cells in Rayleigh-Bénard fluid flows at the first onset of instability, where a domain is a region of parallel rolls. Besides characterizing emergent spatial structures, the regular domains of a CA are fixed or periodic points in pattern space, and as such play an important role in the qualitative analysis of the system's dynamics.

Having outlined a few necessary ideas in section 2, we can now quote the precise definition of a regular domain from ref. [2], as follows. A regular domain Λ of a CA Φ is a process language representing a set of configurations, with the following two properties: (i) *temporal invariance or periodicity:* Λ is mapped onto itself by the dynamic, i.e., $\mathbf{\Phi}^p\Lambda = \Lambda$ for some finite period p; and (ii) *spatial homogeneity:* the process graph of each temporal phase of Λ is strongly connected. The latter is a form of spatial translation invariance and is similar to statistical stationarity or ergodicity of the spatial pattern.

In the analysis of CA, the identification of domains begins with inference of finite automata from examples of the CA's behavior—i.e., the discovery of patterns in space-time data. In simple cases this can be done by inspection. More complex patterns, and especially patterns containing embedded disorder, usually require the automatic inference technique of ϵ-machine reconstruction.[16] The inferred finite automata represent hypotheses or approximations of the important emergent patterns in the system. Once such a pattern has been tentatively identified, its temporal behavior can be analytically examined using the FME operator. In particular, it can be tested for invariance or periodicity by direct comparison of $\mathrm{M}(\mathcal{L})$ and $\mathrm{M}(\mathbf{\Phi}\mathcal{L})$, where the latter is constructed using the FME operator. Similarly, to test for temporal period p, the FME operator is applied p times to construct $\mathrm{M}(\mathbf{\Phi}^p\mathcal{L})$, which is then compared with $\mathrm{M}(\mathcal{L})$. If the pattern is temporally invariant or periodic, and if the process graph of its finite automaton is strongly connected (i.e., it is spatially homogeneous), then the pattern is a domain. The FME operator, comparison of finite automata, and testing for connectivity are fully automated. In this way one can rigorously prove whether any given finite automaton is or is not a domain of any given CA.

If the discovered pattern consists of spatially periodic repetitions of a basic local configuration, then temporal invariance or periodicity of the language can be established by following the evolution of a sample configuration. But for patterns containing aperiodic words, which are quite common in CA behavior, no finite set of examples can be used to deduce behavior of the language as a whole. In such a case, the FME

operator is essential.

3.2 Filters

Having established the existence of one or more domains for a given CA, a *domain transducer* (or *domain filter*) is constructed using a procedure outlined in ref. [4]. A domain filter is a finite state transducer that implements a mapping from arbitrary configurations to strings in which sites participating in domains and domain walls are flagged with distinct symbols. A domain wall (or *defect*) is defined as follows. We consider a domain Λ and a finite string $\omega = zs$ such that $z \in \Lambda$, $s \in \mathcal{A}$, and $\omega \notin \Lambda$. Then s is a wall of domain Λ. We will also refer to a group of closely interacting walls as a defect.

The domain filter scans sites in a spatial configuration in a particular direction (by convention, from left to right), reading each successive symbol of the configuration and writing a symbol identifying the site by the type of domain or defect. In this way it classifies each site of a configuration according to which domain or defect it is participating in. In general, the transducer must read a number of input symbols before any output can be written, in order to gather enough information to unambiguously identify the pattern; this is called *synchronization*. When the transducer encounters a defect, it must *resynchronize*. The way in which it resynchronizes is built into the filter by construction, but in effect it is equivalent to backing up a sufficient number of sites and starting over.

3.3 Particles

Even though the domain pattern is temporally invariant or periodic, a pattern consisting of a defect separating two domains need not be. For example, a certain type of defect may spontaneously decay into a number of defects of other types, may spawn an ever-spreading region of interacting defects, or may even disappear completely. If the pattern *domain-defect-domain* is itself temporally invariant or periodic, and if the width of the defect never exceeds some fixed maximum, then the defect is called a *particle*. This last term will be used here in a specific sense to mean a spatially localized, temporally invariant or periodic boundary separating two adjacent regular domains. The domain-particle-domain pattern is itself a regular language. Its finite automaton consists of a copy of the FA for each domain, with additional states and edges connecting the first domain Λ^i to the second, Λ^j. Note also that although the domain-particle-domain pattern is temporally invariant or periodic, it is not itself a domain, since it is not spatially homogeneous: it is not possible to reach any state in Λ^i from any state in Λ^j.

Applying the domain filter to each spatial configuration in the space-time diagram of a CA produces the *filtered space-time diagram*. The filtered space-time diagram presents the temporal evolution of embedded particles against the identified domain "background". From this filtered behavior, the particle equations of motion and the interactions between particles can be formulated. What emerges is a hierarchical description of the system in which the fundamental patterns—the domains—serve as a background upon which particles evolve with their own properties.

4. The Primary Domain of ECA 54†

4.1 The domain

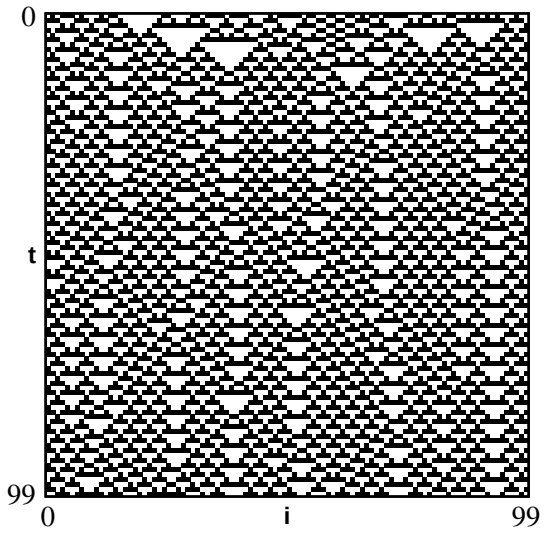

Figure 1. Space-time diagram of ECA 54, starting from an arbitrary initial condition. Boundary conditions are periodic. White squares are cells with value $s_t^i = 0$; black squares are cells with $s_t^i = 1$.

†Some of the material in sections 4 and 5 originally appeared in ref. [5].

Figure 1 shows a typical space-time diagram of ECA 54, starting from an arbitrary initial condition on a lattice of 100 cells and iterating for 100 time steps with periodic boundary conditions. Cells with value $s_t^i = 0$ are printed as white squares; cells with $s_t^i = 1$ are black squares. The question computational mechanics poses is this: Is there some pattern or collection of patterns into which ECA 54 self-organizes? In particular, does ECA 54 have a regular domain?

The answer is yes. We therefore begin the analysis of ECA 54 by describing its most important regular domain. Because it is spatio-temporally periodic (and its period is small), the basic pattern is not difficult to discover by visual inspection. Before showing that this pattern satisfies the requirements of a regular domain, let us first describe it in some detail.

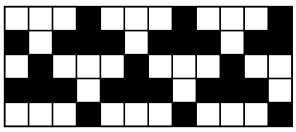

Figure 2. A portion of fig. 1 showing the domain Λ_{54}. Note the spatial phase shift of 2 cells every two iterations.

Figure 2 shows a space-time diagram of ECA 54's primary regular domain Λ_{54}. The domain configurations are spatio-temporally periodic, with periodicity 4 in both space and time. Note, however, that the same *pattern* is repeated after two iterations, spatially shifted by two cells. The pattern of the top row, and every second row following, is given by the expression $(0001)^*$, denoting any number of repetitions of the basic string 0001. Similarly, the second row and every alternate subsequent row are examples of the pattern $(1110)^*$.

We can easily express this domain pattern as a finite automaton $\mathrm{M}(\Lambda_{54})$. The process graph of $\mathrm{M}(\Lambda_{54})$ is shown in fig. 3. The graph consists of two disconnected components, Λ_{A} (at left) and Λ_{B}, which accept the patterns $(0001)^*$ and $(1110)^*$, respectively. Each component consists of four states, represented by circles, and four labelled edges. All states have inscribed circles and squares, indicating start and accept states, respectively. Each state is a distinct spatio-temporal *phase* of the domain and has been assigned a label in $\{a, ..., h\}$ for comparison with other machines below. The spatial periodicity of configurations in Λ_{54} is evident in the fact that each component consists of a single loop of edges without branching. The temporal periodicity of the patterns is schematically indicated by the dotted lines showing the effect of the CA ensemble evolution operator $\mathbf{\Phi}_{54}$. In this representation, we have explicitly discarded all information about the way in which an individual configuration lines up under its parent.

The words in the language Λ_{54} are configurations (or parts of configurations) of the CA. We can think of the machine $\mathrm{M}(\Lambda_{54})$ as scanning the CA lattice from left to right, reading symbols in sequence, and using each symbol to move from state to state by taking the edge with the corresponding label. So long as the input string is in Λ_{54}, it is always possible to make such a transition. When a symbol inconsistent with Λ_{54} is encountered, no corresponding transition exists, so the machine must stop operation; the input string is not recognized by $\mathrm{M}(\Lambda_{54})$. (An alternate version of $\mathrm{M}(\Lambda_{54})$ would have an extra state into which the machine is driven by each input symbol inconsistent with Λ_{54}. This state would have self-loops on both 0 and 1, but would not be an accept state. In this way, all strings would be recognized, but those not in Λ_{54} would be rejected.)

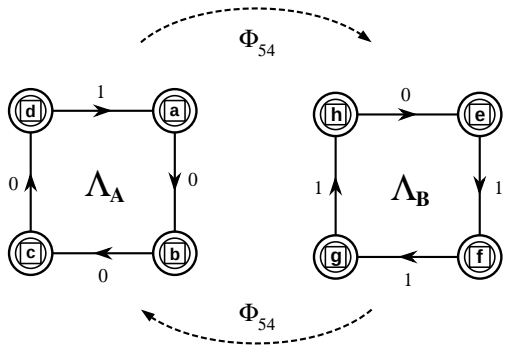

Figure 3. Process graph of ECA 54's domain Λ_{54}. The component on the left is Λ_{A}; on the right is Λ_B. As the dotted lines indicate, they are mapped onto one another by the CA ensemble evolution operator, $\mathbf{\Phi}_{54}$. In this and all following graphs of machines, inscribed circles and squares denote start and accept states, respectively.

Proposition. The regular language Λ_{54} is a domain of ECA 54 with temporal period two.

Proof: The two temporal phases of Λ_{54} satisfy the two conditions of a period-two domain:

1. $\mathbf{\Phi}_{54}\Lambda_{\rm A} = \Lambda_{\rm B}$ and $\mathbf{\Phi}_{54}\Lambda_{\rm A} = \Lambda_{\rm B}$. Recall that $\mathbf{\Phi}_{54}$ is the *ensemble* evolution operator for ECA 54, which iterates each configuration in the language $\Lambda_{\rm A}$ or $\Lambda_{\rm B}$. Explicit construction of $\mathrm{M}(\mathbf{\Phi}_{54}\Lambda_{\rm A})$ and $\mathrm{M}(\mathbf{\Phi}_{54}\Lambda_{\rm B})$ using the FME operator shows that they are identical to $\mathrm{M}(\Lambda_{\rm B})$ and $\mathrm{M}(\Lambda_{\rm A})$, respectively. Alternatively, because all configurations in both $\Lambda_{\rm A}$ and $\Lambda_{\rm B}$ are spatially periodic, their temporal behavior can be inferred from the evolution of a single configuration; see for example fig. 2.
2. The process graph of each temporal phase is strongly connected; this is true by inspection of fig. 3. ∎

Λ_{54} is temporally periodic in two distinct senses. Considered as a *set*, Λ_{54} has temporal period two, as the FME operator verifies, and as indicated by the two disconnected components representing the two temporal phases of the domain. The evolution of a particular domain *configuration*, on the other hand, has temporal period 4, as can be seen in fig. 2. These two distinct types of temporal periodicity give us an opportunity to introduce the *space-time machine* (or *path automaton*) into our set of computational-mechanical tools. A space-time machine is a finite state machine allowed to scan the CA space-time data by means of either spacelike or timelike moves. In the simplest version, spacelike moves are restricted to one spatial direction only; a spacelike move is therefore identical to the move made by the (spatial) domain machine $\mathrm{M}(\Lambda_{54})$ above. Similarly, timelike moves are made only forward in time. Starting from space-time cell (i, t) with value s_t^i, a spacelike move consists of reading the symbol s_t^{i+1} and making the corresponding spacelike transition in the machine, which is indicated by the subscript s. That is, one follows the transition labelled either $0_{\rm s}$ or $1_{\rm s}$, if $s_t^{i+1} = 0$ or $s_t^{i+1} = 1$, respectively. A timelike move consists of reading symbol s_{t+1}^i and making the corresponding timelike transition, labelled with subscript t. In this way, the original binary alphabet $\mathcal{A} = \{0, 1\}$ has been expanded to $\{0_{\rm s}, 0_{\rm t}, 1_{\rm s}, 1_{\rm t}\}$. Thus, the input alphabet for the space-time machine consists of symbols with space-time path information.

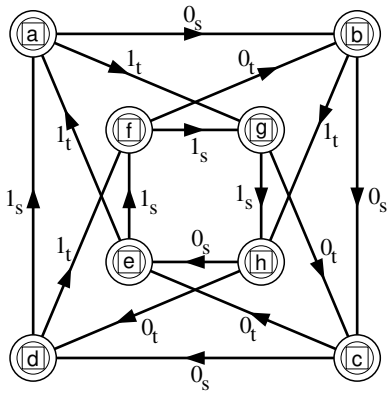

Figure 4. The process graph of $\mathrm{M}_{\rm ST}(\Lambda_{54})$, the space-time machine for Λ_{54}. States are labelled to correspond to those in the (purely spatial) domain machine $\mathrm{M}(\Lambda_{54})$ in fig. 3. Subscripts s and t on the edge labels represent spacelike and timelike moves, respectively. Thus, the label $0_{\rm s}$ represents a symbol 0 encountered on a spacelike move. States and edges devoted to synchronization are not shown.

The process graph of $\mathrm{M}_{\rm ST}(\Lambda_{54})$, the space-time machine for ECA 54's domain, is shown in fig. 4. For clarity, we have omitted the states devoted to synchronization. Symbols with subscripts s and t represent site values encountered on a move to the right (spacelike) or forward in time (timelike), respectively. Note that the two components of Λ_{54}, $\Lambda_{\rm A}$ and $\Lambda_{\rm B}$, are preserved unchanged as square loops of states connected by spacelike edges. The additional information in $\mathrm{M}_{\rm ST}(\Lambda_{54})$ is found in the timelike edges between the two components. $\mathrm{M}_{\rm ST}(\Lambda_{54})$ gives a complete characterization of the spatio-temporal pattern identified by the domain, including both the temporal periodicity of 4 of the configurations and the temporal periodicity 2 of the pattern. A fuller description of the ideas underlying space-time machines, including methods of inferring space-time machines from data and of constructing space-time filters, must wait for presentation elsewhere.

4.2 Domain filtering

According to the sequence outlined in section 3, our next step after finding the domain is to construct the corresponding domain filter. In doing so, we pass on from the spatio-temporal description of the domain (i.e., $\mathrm{M}_{\rm ST}(\Lambda_{54})$ of fig. 4) and return to a purely spatial representation of the domain (i.e., $\mathrm{M}(\Lambda_{54})$ of fig. 3).

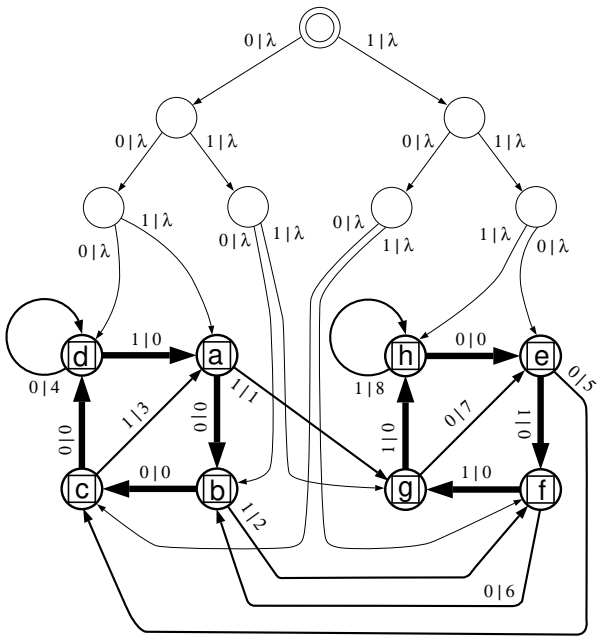

Figure 5. ECA 54's domain filter T^0_{54}*, which maps sites in the domain to 0 and each defect to a unique output in* $\{1, \ldots, 8\}$*. Labelled machine states correspond to the domain states of fig. 3.*

The domain filter T^0_{54}, as constructed from Λ_{54} by the prescription in ref. [4], is shown in fig. 5.

T^0_{54} is a finite state transducer that takes as input any word $\mathbf{s} \in \mathcal{A}^*$ and parses it into a sequence of domain and wall labels. In this way, it recognizes the structures implicit in the spatial configuration $\mathbf{s}$. Two different types of state are shown in fig. 5. States without inscribed labels are synchronization states, devoted to processing the first few symbols in $\mathbf{s}$. The machine starts in the state with the double circle at the top of the figure, then moves along edges according to the input symbols. The first few input symbols serve to synchronize the machine to the underlying pattern, driving it from the start state of total ignorance into one of the recurrent states. The edges are labelled $s_{in}|s_{out}$ for input (raw configuration) and output (filtered) symbols, respectively. Note, however, that during synchronization the machine produces no output, which is signified by putting $s_{out} = \lambda$. During synchronization, the transducer has yet to read in a sufficient amount of information to unambiguously distinguish the domain components and the several wall types. Once the machine has synchronized, every site read is either in a domain or is a wall. At this point, the transducer is in the recurrent part of the graph, where it remains for the rest of its operation. For comparison, asymptotic states in fig. 5 are given labels corresponding to the states of the domain machine $\mathrm{M}(\Lambda_{54})$. Edges corresponding to sites in the domain are printed in bold lines and all have output symbol 0. Walls are printed in lighter lines, and have output symbols in the alphabet $\{1, \ldots, 8\}$. Note that each defect edge has a unique output symbol.

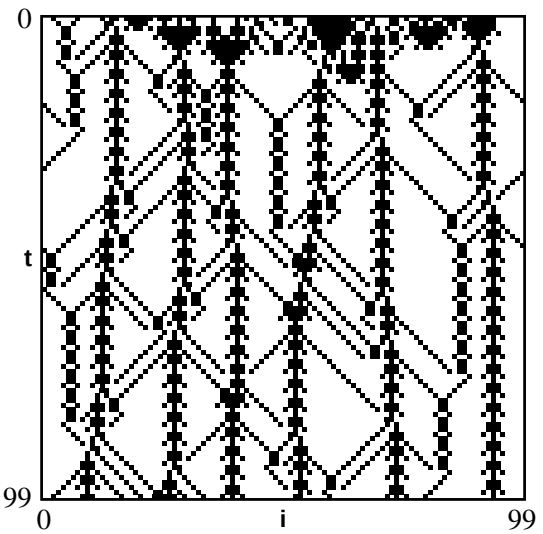

Figure 6. Space-time data of fig.1, filtered with the domain transducer T^0_{54} *of fig. 5. White cells correspond to sites participating in* Λ_{54}*; black cells, to sites with values* $s^i_t \in \{1, \ldots, 8\}$*.*

Applying T^0_{54} to the raw space-time data of fig. 1, we obtained the filtered picture shown in fig. 6. The domain transducer T^0_{54} assigns a unique symbol to each type of wall; in fig. 6, however, we have plotted all eight wall types black squares. In making the filtered plot, the transducer traversed the lattice from left to right, using the periodic boundary conditions to wrap around at the end in order to classify the first few sites, i.e., to classify those used for synchronization.

If we were to change the outputs of domain edges from 0 to 1, and label each defect edge with the symbol 0, then T^0_{54} would implement the same mapping as the transformation function in ref. [14],

$$F\left(s^i_t\right) = \left(\sum_{k=0}^{3} s^{i+k}_t\right) mod\ 2 \qquad (3)$$

As its form indicates, F assigns a 0 or 1 to each length 4 subsequence in the spatial configuration depending on whether the number of 1s in the subsequence is even or odd. Its simplicity takes advantage of the facts that (i) all length 4 words in Λ_{54} have an odd number of 1s; and (ii) all other length 4 words contain defects. Note that F introduces a spatial shift $\Delta i = -4$ in the output by writing its output over the *first* symbol in the word, rather than the last.

Unlike eq. (3) however, T^0_{54} was constructed by a prescription that applies to any domain language—not just that of ECA 54. Additionally, T^0_{54} does more than just flag locations of defects; it also performs important classification of defect type, which we will exploit in the following sections. For example, the pattern of wall symbols $\{1, \ldots, 8\}$ in the filter's output affords sufficient information for automatic identification of the particles visually evident in fig. 6. This is the subject of section 5.

4.3 Equation of motion of the filtered behavior

One question that naturally arises at this point is whether the filtered space-time behavior, like that shown in fig. 6, is itself consistent with some deterministic equation of motion. That is, does there exist a mapping $\widehat{\Phi}_{54}$ such that $\mathrm{T}^0_{54}[\Phi_{54}(\mathbf{s})] = \widehat{\Phi}_{54}\big[\mathrm{T}^0_{54}(\mathbf{s})\big]$ for all CA configurations $\mathbf{s}$? We know that in the case of ECA 18 and in the case of arbitrary filtered CA, this is not true. In the former, the filtered behavior is described by a stochastic equation of motion for diffusive annihilating particles.[3] In the latter, the resulting filtered space-time behavior often can be described only by a strictly more computationally complex class of spatial processes called cellular transducers.[18]

The alphabet of the hypothetical mapping $\widehat{\Phi}_{54}$ is $\mathcal{A}' = \{0, \ldots, 8\}$. The set of initial conditions over which $\widehat{\Phi}_{54}$ is defined is not $(\mathcal{A}')^*$, however; rather, it is the set of possible outputs of T^0_{54}—its *output language*, denoted by $\left[\mathrm{T}^0_{54}\right]_{out}$. Of course, $\left[\mathrm{T}^0_{54}\right]_{out} \subset (\mathcal{A}')^*$. By definition, for all CA configurations $\mathbf{s} \in \mathcal{A}^*$, the filtered string $\mathbf{z} = \mathrm{T}^0_{54}(\mathbf{s})$ is in $\left[\mathrm{T}^0_{54}\right]_{out}$.

To establish the existence of $\widehat{\Phi}_{54}$, we must first prove the following:

Lemma. On CA lattices with periodic boundary conditions, the domain filter T^0_{54} is invertible except on strings without defects. That is, for every filtered string $\mathbf{z} \in \left(\left[\mathrm{T}^0_{54}\right]_{out} - 0^*\right)$, there exists exactly one configuration $\mathbf{s} \in (\mathcal{A}^* - \Lambda_{54})$ such that $\mathbf{z} = \mathrm{T}^0_{54}(\mathbf{s})$. On strings without defects— i.e., for $\mathbf{z} \in 0^*$—T^0_{54} is 8 to 1.

Proof: We show invertibility of T^0_{54} by constructing its inverse, denoted by Q^0_{54}. In essence, this is done by simply reversing the order of the edge labels on the graph of T^0_{54}, i.e., replacing label $x|y$ with $y|x$; this converts input symbols into output symbols, and vice-versa. The states devoted to synchronization are removed and a new synchronization structure, determined by the remaining asymptotic part of the machine, is substituted in their place. Figure 7 shows the resulting transducer. The transducer starts out in its synchronization state (state 0) and remains there until the first nonzero input (i.e., filtered) symbol is encountered. That symbol synchronizes it to the input string, and it begins writing output (i.e., unfiltered) symbols. At the end of the lattice, the filter wraps around the periodic boundary and continues generating output until the first nonzero input is encountered—i.e., it goes back and writes outputs for the portion of the filtered data originally read prior to synchronization. The constraint that at least one defect is present ensures that the transducer will synchronize before wrapping around. Note that Q^0_{54} is *deterministic*; i.e., for each input symbol $z \in \mathcal{A}'$, there is at most one edge with matching input label leaving each state of Q^0_{54}. This means that each filtered string $\mathbf{z}$ containing one or more defects, which by assumption is in the set $\left(\left[\mathrm{T}^0_{54}\right]_{out} - 0^*\right)$, is mapped to exactly one unfiltered configuration $\mathbf{s} \in (\mathcal{A}^* - \Lambda_{54})$. If $\mathbf{z} \in 0^*$, then there is no defect by which Q^0_{54} can synchronize. The unfiltered configuration can be any of the eight length-N domain configurations. ∎

Using the above lemma, we can define the mapping that takes a filtered string $\mathbf{z}_t$ at time t and generates its iterate $\mathbf{z}_{t+1}$, as follows. If there is a nonzero symbol in $\mathbf{z}_t$, we first apply the inverse domain filter to generate $\mathbf{s}_t = \mathrm{Q}^0_{54}(\mathbf{z}_t)$, then apply the CA rule to generate $\mathbf{s}_{t+1} = \Phi_{54}(\mathbf{s}_t)$, then convert back to the filtered string $\mathbf{z}_{t+1} = \mathrm{T}^0_{54}(\mathbf{s}_{t+1})$. If $\mathbf{z}_t$ consists of all 0s, then necessarily the unfiltered string $\mathbf{s}_t$ is in Λ_{54}; since Λ_{54} is a domain, $\mathbf{s}_{t+1} \in \Lambda_{54}$ as well, implying that $\mathbf{z}_{t+1}$ consists of all 0s. Finally, even though $\widehat{\Phi}_{54}$ need

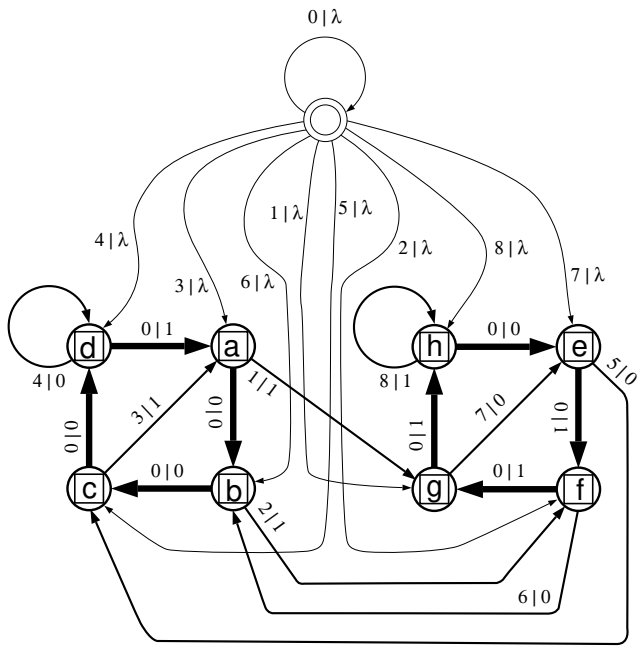

Figure 7. Graph of the finite state transducer Q^0_{54} *implementing the inverse of the domain filter* T^0_{54}.

only be defined over the set $\left[\mathrm{T}^0_{54}\right]_{out}$, we may trivially extend it over all of $(\mathcal{A}')^*$ by mapping every string not in $\left[\mathrm{T}^0_{54}\right]_{out}$ to all 0s. The net result is summarized in the following proposition.

Proposition. Let $\{\mathbf{s}_t : t = 0, \ldots, k-1\}$ be a time sequence of spatial configurations generated by ECA 54 on a finite lattice of N sites with periodic boundary conditions, starting from arbitrary initial condition $\mathbf{s}_0$. Let $\left\{\mathbf{z}_t = \mathrm{T}^0_{54}(\mathbf{s}_t) : t = 0, \ldots, k-1\right\}$ be the time sequence of filtered strings constructed from $\{\mathbf{s}_t\}$. Then the $\mathbf{z}_t$ evolve according to the deterministic equation of motion

$$\widehat{\Phi}_{54}(\mathbf{z}_t) \equiv \begin{cases} \mathrm{T}^0_{54}\left(\Phi_{54}\left[\mathrm{Q}^0_{54}(\mathbf{z}_t)\right]\right) & \mathbf{z}_t \in \left[\mathrm{T}^0_{54}\right]_{out} - 0^* \\ 0^N & otherwise \end{cases} \tag{4}$$

Using a simple variation of *machine composition*,[2] it is straightforward to construct the finite state transducer for $\widehat{\Phi}_{54}$. Since Q^0_{54}, Φ_{54} and T^0_{54} are all deterministic, so is $\widehat{\Phi}_{54}$. This means that the filtered space–time dynamics is governed by an effective equation of motion that is itself deterministic. The FST for $\widehat{\Phi}_{54}$ is given in Appendix A.

In general, it is always possible to construct the inverse of a finite state transducer; however, that transducer may be nondeterministic—i.e., there may be more than one edge with a given input symbol leaving a given state. The inverse of a transducer T is nondeterministic if T has at least one state in which the same output symbol is assigned to more than one edge leaving that state. This can happen, for example, of filters for disordered domains, which have nonzero spatial entropy density. Because the disordered domain from which the filter was constructed contains states with branching edges, the filter itself contains states in which more than one input is mapped to the domain output symbol. Since this occurs in the recurrent portions of the filter, there will be states there with more than one incoming edge. In turn, if the labels of these incoming edges have the same output symbol, then the inverted transducer will be nondeterministic. When the domain configurations are spatially periodic, as is the case in ECA 54, there are no such branches, and the domain filter is invertible.

Since all CA rules are deterministic by definition, and since all domain filters are deterministic by construction, whether $\widehat{\Phi}$ is deterministic depends solely on whether the inverse domain filter is deterministic. As we noted above, this is the case for any spatially periodic domain, but not for disordered domains (see for example the analysis of ECA 18 in ref. [2]). Thus we arrive at the following generalization of the above proposition:

Observation. For any spatially periodic domain in any CA, the space-time dynamics of the domain filter output is governed by a deterministic equation of motion.

We close this section by stating a question that we will not answer here. $\widehat{\Phi}_{54}$ is deterministic, but is it a CA? That is, can $\widehat{\Phi}_{54}$ be expressed in a finite radius lookup table?

5. Particles and Their Interactions

5.1 The fundamental particles

A visual inspection of fig. 6 indicates that there are three particularly common types of wall structure in the filtered data: a stationary pattern with temporal

period 4 and width fluctuating from 1 to 3 to 5 cells; a stationary pattern with temporal period 4 and width 2 or 4 cells; and an isolated defect surrounded by 0s, moving to the left or right with speed $c = \pm 1$ site per iteration. Figure 8 shows raw and filtered plots for each of these three structures, which satisfy the definition of particle given in section 3. They are assigned the names α, β, and γ, respectively. As the figure illustrates, each particle shows temporal as well as spatial structure. We have therefore assigned a temporal phase to each particle, which is printed beside the filtered strings.

Comparison of the filtered space-time signatures of these particles indicates that a single defect edge in the domain filter may participate in the detection of more than one particle. For example, the edges with defect output symbols 8 and 4 are used during both α and β particle detection, defect outputs 5 and 1 are in all three particles, and defect outputs 2 and 6 are in both β and γ particles. Consistent with this observation, we note that one interpretation of the β particle is that phases β_1 and β_2 are a pair of colliding particles ($\gamma^+ + \gamma^-$). In this interpretation, β_0 spontaneously decays into the pair $\left(\gamma_0^+, \gamma_0^-\right)$ separated by two domain sites and directed back toward each other. This is seen as the diagonally placed defect symbols $(5, 2)$ and $(1, 6)$, respectively, in the filtered diagram. These then collide to create a β_3. A similar interpretation can be made for α.

This interpretation illustrates a notable fact about particles as we have defined them: A particle may be composed of interacting subunits that, in isolation, are themselves particles in their own right. The analogy to atomic physics is obvious here. More interestingly, in certain cases particles may line up to form "bound state" patterns that satisfy the definition of a regular domain. We will discuss some examples of this in section 6.

The general motto of filter construction is, Whenever a structure is discovered, filter it out. Once we have identified particles $\{\alpha, \beta, \gamma^+, \gamma^-\}$, therefore, we construct a new filter tuned especially for them. This can be done by direct modification of the domain filter machine, but a more practical and instructive alternative is to construct a *particle filter* T_{54}^1 that operates on the output of the domain filter T_{54}^0, classifying sequences of filtered symbols by particle type. In doing so, we are moving up one level in a hierarchical analysis of the system. At the domain-filter level, all that

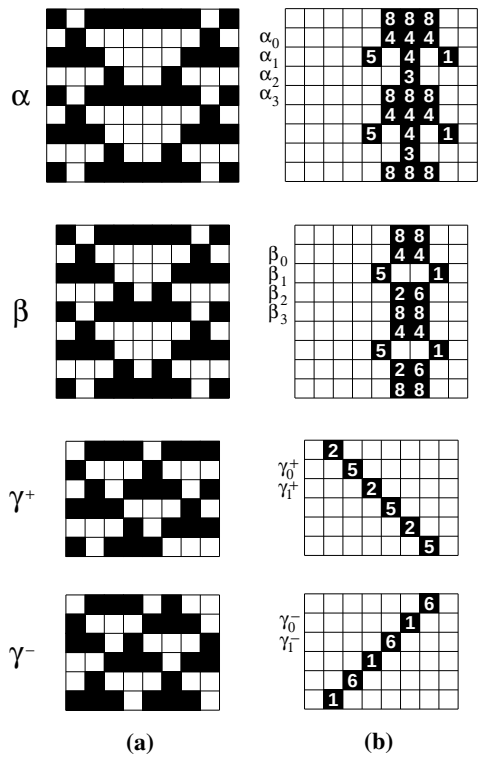

Figure 8. Basic wall structures ("fundamental particles") in space-time patterns of ECA 54. (a) Unfiltered space-time diagrams of the three types of particle α, β, and γ described in the text. (b) Filtered diagrams of the same data, produced by T_{54}^0. Domain symbols are white cells. All defects are shown in black, with the defect symbol inscribed in white. The temporal phases of the particles, chosen by convention, are printed alongside the filtered strings.

was recognized was the domain; defects were classified only as deviations from the domain pattern. The particle filter T_{54}^1 further recognizes some of these deviations as being particles. Each filter performs the same basic type of operation to its input data: it recognizes and filters particular structures, highlighting still-undiscovered patterns as deviations from these structures. In the following discussion, T_{54}^1 is first constructed to recognize the three fundamental particles, and then modified to recognize strings arising in interactions between pairs of particles as well.

In constructing T_{54}^1, we will consider a particle to be well-defined only if it is bounded on both sides by at least one domain cell; this prevents misidentification of very long strings of adjacent defects as sequences of adjacent particles. Each cell identified by T_{54}^0 as

part of the domain is assigned output symbol 0 by T^1_{54} as well, thus preserving the domain filter's recognition of the domain structure. Each defect cell participating in a particle of a given type is mapped to a given output symbol—i.e., to one of the symbols $\{\alpha, \beta, \gamma^+, \gamma^-\}$. Defect cells that do not correspond to any of the fundamental particles are left unchanged; that is, the symbol in $\mathcal{A}' = \{1, \ldots, 8\}$ is copied to the filter output. This will allow us to examine the output of T^1_{54} for new, as yet unrecognized, structure.

ECA 54's particles are made up of a finite number of short strings. Comparison of the particles' filtered strings (fig. 8) shows that, for example, the string 08880 should be classified as an α particle, while the string 0880 corresponds to particle β. On reading the partial input 08, therefore, the particle filter must defer output until the string may be unambiguously classified. This effectively builds a queue of finite depth into the filter. Output is deferred until an isolated particle is recognized (or ruled out), then written as a string in a single transducer transition. There are twelve strings occurring in all phases of the fundamental particles. Each of these is built into the structure of the particle filter in the same way.

5.2 The fundamental particle interactions

Having identified the different particle types, we may now investigate the various interactions among them. For example, near the bottom of fig. 1 one sees that the collision of a γ^+ and a γ^- creates a single β. All told, there are five possible interactions between pairs of particles. Figures 9(a) through 9(e) show the filtered space-time diagrams of all five, which are also summarized in table 1.

Perhaps the most obvious feature of fig. 9 is that all two-particle interactions ultimately result in patterns containing only the domain and the fundamental particles. Interactions (a) and (b) show the effect of a collision of a γ with an α. In both cases, the original particles persist, though with spatial and temporal phase shifts, and a new pair of γs is created. As figs. 9(a) and 9(b) show, new strings not corresponding to any fundamental particle are generated during a few iterations immediately following the collision, but before long these have condensed back into particles. Interaction (a) induces a net spatial shift of $\Delta i = -1$ and a net temporal phase shift of $\Delta t = +1$ in the α particle, and a space-time shift of $(\Delta i, \Delta t) = (+1, 0)$ in the original γ^-. Interaction (b) shifts the α by $(\Delta i, \Delta t) = (+1, +1)$ and the original γ^+ by $(\Delta i, \Delta t) = (-1, 0)$. If we neglect for the moment the different values of the defect cells, we see that interactions (a) and (b) are spatial reflections of each other.

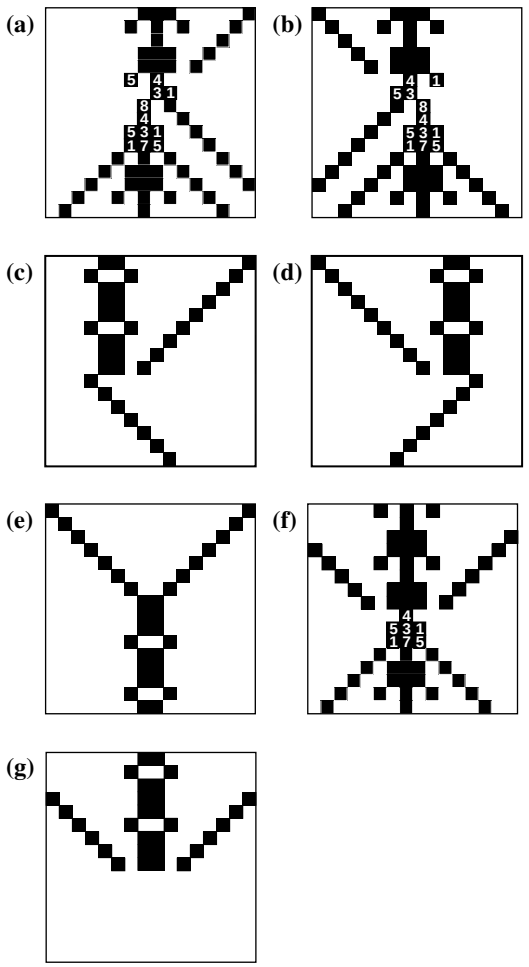

Figure 9. Filtered space-time diagrams of the fundamental interactions among ECA 54's fundamental particles, as listed in table 1. (a)—(e) are two-particle collisions; (f) and (g) are three-particle collisions. Filtering was done with the first version of particle filter T^1_{54} described in the text. The domain is shown as white, the particles $\{\alpha, \beta, \gamma^+, \gamma^-\}$ are shown in black. Defects not corresponding to any of the particles are shown in black and have the corresponding T^0_{54} output symbols inscribed in white.

(a)	$\alpha + \gamma^- \rightarrow \gamma^- + \alpha + 2\gamma^+$
(b)	$\gamma^+ + \alpha \rightarrow 2\gamma^- + \alpha + \gamma^+$
(c)	$\beta + \gamma^- \rightarrow \gamma^+$
(d)	$\gamma^+ + \beta \rightarrow \gamma^-$
(e)	$\gamma^+ + \gamma^- \rightarrow \beta$
(f)	$\gamma^+ + \alpha + \gamma^- \rightarrow \gamma^- + \alpha + \gamma^+$
(g)	$\gamma^+ + \beta + \gamma^- \rightarrow \emptyset$

Table 1. Fundamental interactions among ECA 54's particles. Interactions (a) and (b) induce a spatio-temporal shift in the incident particles, as discussed in the text. Note that the spatial arrangement of input and output particles is respected by the interaction notation. $\emptyset$ denotes no particles.

Interactions (c) and (d) show the results of a collision between a β and a γ. The interaction occurs in a single time step, as indicated by the fact that no new non-particle strings are generated. Note that these two interactions are also spatial reflections of each other. Interaction (e) shows the collision of two γs. Again, the interaction takes place in a single time step. This interaction is itself spatially symmetric. Generally here, and unlike several of the interactions found in the $\phi_{\rm sync}$ CA of ref. [19], there is no dependence of the two-particle interactions on their relative internal phases.

Besides the five two-particle collisions, figs. 9(f) and 9(g) and table 1 show two simple three-particle collisions. Both of these interactions are spatially symmetric. Interaction (f) consists of a pair of incoming γs colliding simultaneously with an α. All three particles survive the collision. The α undergoes a shift of $(\Delta i, \Delta t) = (0, +2)$, the incoming γs are both shifted by $(\Delta i, \Delta t) = (0, -1)$. Note that the new filtered strings occurring during the interaction are also found in the two-particle α-γ collisions (a) and (b). Finally, interaction (g) shows the annihilation of a β by simultaneous collision with two incoming γs.

One advantage to constructing the particle filter T^1_{54} as described in section 5.1 is that it is now easy to modify it to recognize interactions among particles as well. There are eight new strings arising in two-particle interactions, all occurring immediately after an α-γ collision. We can include these strings in T^1_{54} in the same way as the original particles' strings were included, assigning the output symbol δ to all defects in strings unique to α-γ to indicate the intermediate bound state during the particle interaction. Generally these modifications take the form of adding more states and edges to T^1_{54} by a procedure that is readily automated. The net result is our second and final version of T^1_{54}, a portion of which is shown in fig. 10. Output 0 indicates a cell in the domain, outputs $\{\alpha, \beta, \gamma^+, \gamma^-\}$ indicate defect cells making up the fundamental particles, output δ indicates defect cells in α-γ interactions, and outputs $\{1, \ldots, 8\}$ indicate defect cells not corresponding to any recognized structure.

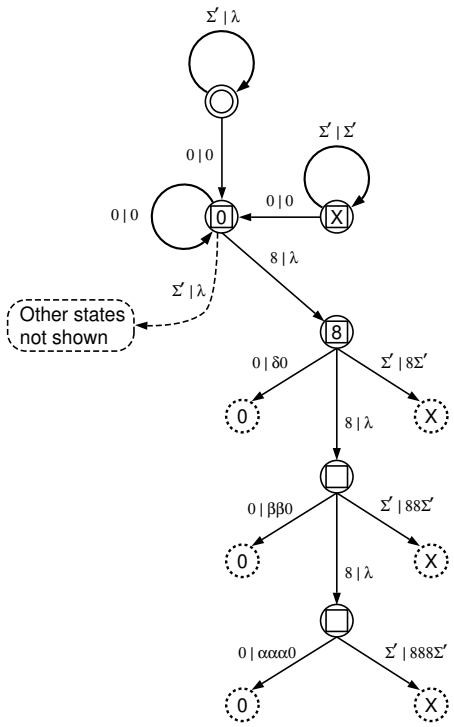

Figure 10. A portion of the graph of particle filter T^1_{54}, which maps domain cells to 0, cells in the fundamental particles to output symbols $\{\alpha, \beta, \gamma^+, \gamma^-\}$ and strings occurring in pairwise α-γ interactions to δ. All other defects are left changed, since they do not correspond to recognized structures. The input label Σ' denotes all input symbols not found on other edges leaving a given state. Edges terminating on broken circles actually reconnect to the indicated state at the top of the machine; they have been displayed in this way for clarity. Note that the number of output symbols in an edge label varies from zero (indicated by λ) to four. Not shown are the other seven edges leaving state 0, corresponding to input symbols $\{1, \ldots, 7\}$. Each of these edges leads to a different group of states and edges similar to the one shown.

T^1_{54} operates on output strings of T^0_{54} in the same way that T^0_{54} operates on raw configurations. The machine is initially in the state with the double circle. Once it reads a domain symbol (0), it synchronizes to state 0, which indicates that the next defect encountered is isolated from any defects to its left. From state 0 lead eight edges, one for each of the outputs of the domain filter. Each of these edges in turn leads to a branching sequence of states and edges that eventually writes as output one of the particles and resets to state 0, or flags an unrecognized defect configuration and resets to state X. In fig. 10, this structure is fully indicated only for the edge with input label 8; each of the other edges leaving state 0 leads to a similar structure.

5.3 Statistics and completeness of the particle-level description

The filters T^0_{54} and T^1_{54} embody a description of ECA 54's information processing structure in terms of the domain Λ_{54}, particles $\{\alpha, \beta, \gamma^+, \gamma^-\}$, and the interactions in table 1. We can now apply this pair of filters to space-time data generated by ECA 54, in order to investigate the extent to which this "particle-level" description is complete. One simple way of doing this is to measure the frequency of occurrence of each of T^1_{54}'s output symbols as a function of time. If the frequency unrecognized defects (T^1_{54} outputs $\{1, \ldots, 8\}$) decays to zero in finite time, or if the ratio of unrecognized defects to particles vanishes asymptotically, then no asymptotic structures not describable in the particle-level vocabulary exist. The conclusion then would be that the particle-level description is complete.

Figure 11 shows the fraction of cells in a CA lattice of size $N = 10^6$ which T^1_{54} classified as particles $\{\alpha, \beta, \gamma^+, \gamma^-\}$, as fundamental interactions $\{\delta\}$, or as unrecognized defects $\{1, \ldots, 8\}$, plotted versus time for the first 50 iterations of a random initial condition. The fraction of cells in the domain, not shown in the figure, increases monotonically from an initial value of 0.4996. The curves for γ^+ and γ^- overlay each other almost exactly. Unrecognized defects 2 and 6 start with the same initial fraction and disappear on the first iteration; defects 1, 3, 5, and 7 all have equal initial fractions that vanish by iteration 10. The fractions of defect types 4 and 8 do not vanish, though they are significantly lower than all of the particle frequencies. Unless they decay to zero faster than the particle frequencies, this is evidence that the "particle-level" description, as captured by T^1_{54}, is incomplete. Unfortunately, the frequencies of defects 4 and 8 remain within a few orders of magnitude of the particle frequencies out to time $t \sim 10^6$, and show no sign of vanishing faster than the particle frequencies. As we will discuss in section 6, these two defect types occur together, and are created by certain types of multiparticle interactions usually involving αs.

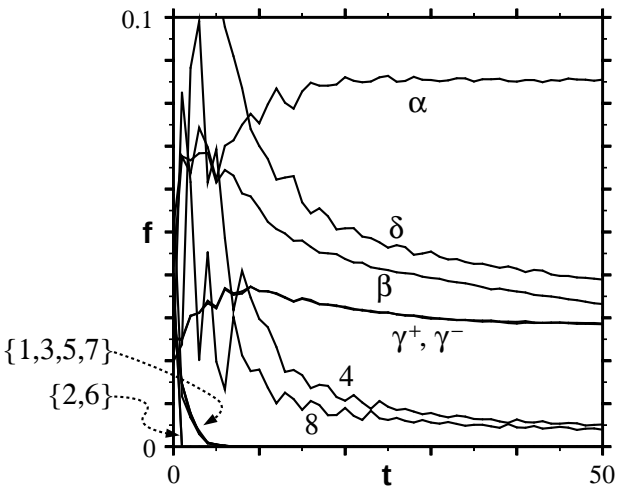

Figure 11. Fraction of the CA lattice devoted to the fundamental particles ($\alpha, \beta, \gamma^+, \gamma^-$), to α-γ interactions (δ), and to unrecognized defects $\{1, \ldots, 8\}$ versus time.

Figure 11 also indicates two epochs of ECA 54's time evolution. For the first few iterations, the system is self-organizing from a random initial configuration into a system of particles (and defects 4 and 8). This process is reflected in the wild fluctuations in particle fractions and in the gradual increase in αs. It is notable that these apparent fluctuations at early times are not due to statistical variation in the random initial condition. We repeated the experiment several times using different random initial configurations and different lattices sizes, and the same "fluctuations" were repeated almost exactly—even down to the size and position of the oscillations in α frequency. The only discernible discrepancies between the experiments were slight differences in the size of the oscillations in δs, 4s, and 8s, starting at about $t = 10$ and gradually disappearing thereafter. As the figure shows, by about $\tau \approx 25$ this initial condensation into domains and particles has been replaced a slow, smooth decrease in all particle frac-

tions, which continues indefinitely. We may therefore consider τ as an *emergence time* for the domain/particle structures.

To measure the asymptotic behavior of the system, we could simply continue the above experiment for much longer. However, to illustrate the flexibility of our filtering scheme, we will instead make a simple modification to the particle filter and count the exact number of each type of particle at each time. This modification consists of changing the output string for each temporal phase of each particle to contain only one nonzero symbol, while maintaining proper spatial locations. Unrecognized defects are handled in the same way as in T^1_{54}. The resulting machine, denoted by T^2_{54}, writes a single nonzero value at the center of each particle, regardless of its width. This makes counting particles just a matter of counting symbols in the filter output.

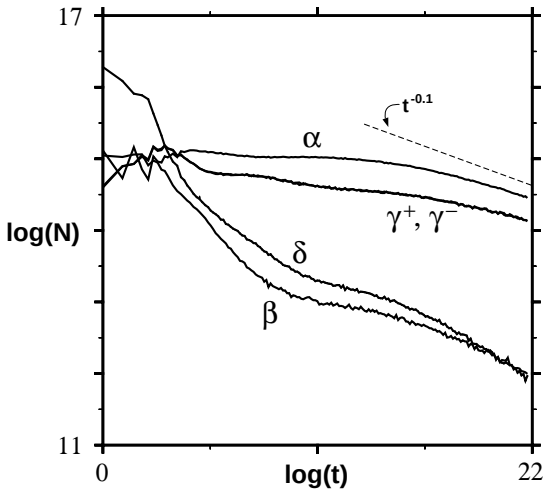

Figure 12. Number of particles $\{\alpha, \beta, \gamma^+, \gamma^-\}$ and α-γ interactions δ versus time. The lattice size is $N = 10^6$. Logarithms are base 2. The dashed line shows the measured power law decay at long times.

Figure 12 shows the number of particles and α-γ interactions versus time plotted on a log-log scale, for the same experiment but longer times as in fig. 11. Numerical studies in ref. [14] have indicated that at very long times the number of αs and δs decreases very slowly according to a power law $[N(\alpha) + N(\delta)] \sim t^{-a}$, with $a = 0.15$. Our experiments on lattices of size $N = 10^4$, 10^5, and 10^6 show the same α-frequency curves in all cases, with a long-time decay $N(\alpha) \sim t^{-a}$ having rate $a = 0.10 \pm 0.01$. As fig. 12 shows, however, this long-time scaling regime sets in extremely gradually with a slow downturn beginning at about $t \sim 16,000$. By about $t \sim 2 \times 10^5$, the downturn is completed and the α-frequency curve follows its long-time power law decay. One experiment on lattice size $N = 10^4$ was allowed to evolve for $t = 2 \times 10^8$ iterations; no systematic deviation from this functional form was observed. The fact that changing the lattice size had no effect on the shape of the α-frequency curve shows that the onset of the scaling regime is not due to finite-size effects.

Another statistical measure of the system's global behavior is given by the fraction $f(\Lambda_{54})$ of sites that are in the domain. After the initial emergence time, $f(\Lambda_{54})$ shows a very slow monotonic increase toward a value of 1.0, which corresponds to a configuration entirely free of defects or particles. In the scaling regime, $f(\Lambda_{54})$ increases as a power law with exponent $a = 0.007 \pm 0.0005$.

A further modification of the particle filter provides us with a valuable tool for data visualization on large space-time scales. Noting that the αs are the longest-lived of the fundamental particles, we construct an "α-tracker," denoted by T^3_{54}, which traces the evolution of αs to the exclusion of all other structures. This is accomplished as follows. First we change the particle filter output to map all βs and γs to the domain symbol 0. Then we modify the output on each input corresponding to an α or an α-γ interaction so that a single nonzero symbol (1) is written, in the same way as in T^2_{54}. Unrecognized defects are all mapped to a third symbol (2). In this way, space-time data filtered with T^0_{54} and T^3_{54} show a single 1 in the central cell of each α, blocks of 2s on the relatively rare unrecognized defects, and 0s in all other cells. This effects a vast reduction in the amount of information in the space-time diagram.

Figure 13 shows two space-time diagrams of a random initial configuration on lattice of 1000 cells, evolving for a total of 10,000 time steps. Each diagram covers a rather large number of space-time cells—10^7. It is interesting to note that even a PostScript rendered $8.5'' \times 11''$ page contains only about 10^7 resolvable dots. At the top of fig. 13 is the unfiltered space-time