to appear in *Multiple-Valued Logic*, Special Issue
on Collision-Based Computing, early 2001

Computing with Solitons: A Review and Prospectus

Mariusz H. Jakubowski *and* Ken Steiglitz
Computer Science Department
Princeton University
Princeton NJ 08544

Richard Squier
Computer Science Department
Georgetown University
Washington DC 20057

August 27, 1999

Abstract

We review work on computing with solitons, from the discovery of solitons in cellular automata, to an abstract model for particle computation (particle machines), to information transfer in collisions of (continuum) optical solitons, to state transformations in collisions of Manakov (vector) solitons. We conclude by discussing open problems and the prospects for practical applications using optical solitons in photorefractive crystals and other materials.

1 Introduction

In most present-day conceptions of a "computer," information travels between logical elements fixed in space. This is true for abstract models like Turing machines, as well as real silicon-chip-based electronic computers. This paper will review work over the past few years that views computation in an entirely different way: information is carried through space by particles, computation occurs when these particles collide.

Much of this review will focus on our own work, but of course will necessarily touch on related work by many others. Our intent is to describe the progression of ideas which has brought the authors to a study of the computational power of the Manakov system and its possible practical implementation, and in no way do we intend to minimize important contributions by others to the growing field of embedded computation, and nonstandard computation in general. We will cite such related work as we are aware of, and will appreciate readers bringing omissions to our attention.

This paper is written more or less historically, and we will trace the development in the following stages:

- solitons in automata,
- the particle machine model and embedded arithmetic,
- information transfer in collisions of continuum solitons,
- the Manakov system.

We conclude with a discussion of open problems and a brief appraisal of the prospects for the practical application of these ideas.

2 Computation in cellular automata

Our own story begins in a sense with influential work of S. Wolfram in the mid-1980s [1], and in particular with the seminal study [2], where he observed that the behavior of a simple but representative type of cellular automata (CA) can be partitioned into four classes, the most

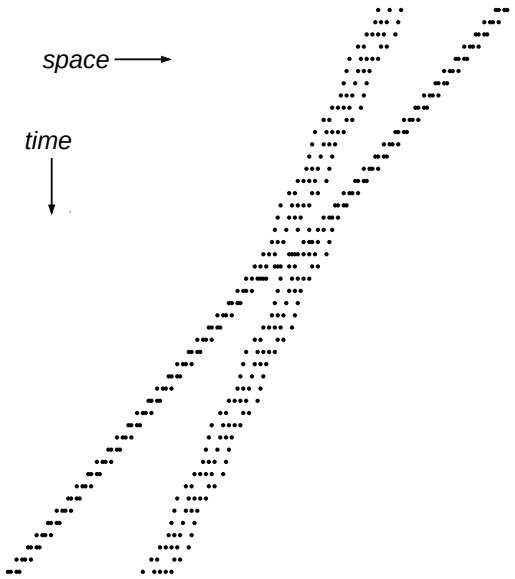

Figure 1: Typical soliton-like collision in a PRFA. The dots are ones against a field of zeros and the time evolution progresses downward. Notice the displacements caused by the collision, quite similar to what happens in continuum systems. This example uses a window radius of five and is from [3].

interesting of which he conjectured to be Turing universal. At this time S. Wolfram was in Princeton, at the Institute for Advanced Study, and his work led one of us (KS) to begin experimentation with CA. It was subsequently discovered that a new class of CA, the *parity rule filter automata* (PRFA) [3], support particles which behave remarkably like solitons in continuum systems. (We review the essential features of continuum solitons in physical systems in Section 4.)

The PRFA differ from conventional CA in that the update rule uses new values as soon as they become available, scanning to update from left to right. They are thus analogous to so-called infinite impulse response (IIR) digital filters, while ordinary CA correspond to finite impulse response (FIR) digital filters [4]. We note that although the operations of FA and CA are different, the two classes of automata are equivalent, as shown in [3]. However, to our knowledge it is an open question whether the subclass of PRFA are computationally universal. The term "parity rule" refers to the algorithm for refreshing site values: to update at a particular site, the total number of ones in a window centered at that site is found (using new values to the left), and the new site value is set to one if that sum is even but not zero, and to zero otherwise. Figure 1 shows a typical soliton-like collision in a PRFA. Notice that the bit pattern of both particles is preserved in the collision, and that both particles are displaced from their pre-collision paths.

The subsequent paper [5] showed that a ripple-carry adder can be realized in a PRFA. This construction was not simple, and we were not able to "program" more complex computation in these one-dimensional structures. However, several researchers have been contributing to a growing literature on embedded computation in automata using particles, and recent results are quite interesting. The literature is large enough, in fact, that it is not possible to review it here. But before leaving PRFA we cite (without claim to completeness) some representative work by important researchers: Goldberg [6], Fokas, Papadopoulou, and Saridakis [7, 8], Ablowitz, Keiser, and Takhtajian [9], Bruschi, Santini and O. Ragnisco [10], Adamatzky [11, 12, 13], and Siwak [14, 15, 16].

3 Particle Machines

The *particle machine* (PM) model was introduced and studied in [5, 17] and is intended to capture the notion of computation by propagating and colliding particles. It was a natural outgrowth of the study of the computational power of the PRFA. Like the Turing machine (TM), the PM can do general computation [17], and operates in discrete time and space. However, while the TM's tape, read-write head, and uniprocessing operations hint at mechanical, man-made origins, the PM's particle interactions and fine-grain parallelism are reminiscent of natural physical systems.

We will review the PM model and mention several efficient algorithms that have been encoded in the model. We define a PM as a cellular automaton (CA) with states that represent idealized particles, and with an update rule that encodes the propagation and collisions of such particles. While PMs can have any number of dimensions, we concentrate here on one-dimensional PMs, which are nevertheless powerful enough to support efficient implementations of arithmetic and convolution.

3.1 Characteristics of PMs

Quite apart from their use as an abstract model of computation, PMs can be viewed as a way to incorporate the parallelism of systolic arrays [18] in hardware that is not application-specific and is easy to fabricate. A PM can be realized easily in VLSI and the resultant chips are locally connected, very regular (being CA), and can be concatenated with a minimum of glue logic. Thus, many identical VLSI chips can be strung together to provide a very long PM, which can then support many computations in parallel. What computation takes place is determined entirely by the stream of injected particles: There are no multipliers or other fixed arithmetic units in the machine, and the logic supports only particle propagation and collisions. While many algorithms for a PM mimic systolic arrays and achieve their parallelism, these algorithms are not hard-wired, but are "soft," or "floating," in the sense that they do not determine any fixed hardware structures.

An interesting consequence of this flexibility is that the precision of fixed-point arithmetic is completely arbitrary and determined at run time by the user. In [17] the authors show that FIR filtering (convolution) of a continuous input stream, and arbitrarily nested combinations of fixed-point addition, subtraction, and multiplication, can all be performed in one fixed CA-based PM in time linear in the number of input bits, all with arbitrary precision. Later in this section we complete this suite of parallel arithmetic operations with a linear-time implementation of division that exploits the PM's flexibility by changing precision during computation.

3.2 The PM model

We define the PM formally as follows:

Definition 1 A *Particle Machine (PM)* is a CA with an update rule designed to support the propagation and collision of logical *particles* in a one-dimensional homogeneous medium. Each particle has a distinct identity, which includes the particle's velocity. We think of each cell's state in a PM as a *binary occupancy vector*, in which each bit represents the presence or absence of one of n particle types (the same idea is used in lattice gasses; see, for example, [19]). The state of cell i at time $t+1$ is determined by the states of cells in the *neighborhood* of cell i, where the neighborhood includes the $2r+1$ cells within a distance, or *radius*, r of cell i, including cell i. In a PM, the radius is equal to the maximum velocity of any particle, plus the maximum displacement that any particle can undergo during collision.

Although this definition is explicitly in one-dimension, it can be generalized easily to higher dimensions.

In summary, a PM is a CA with an update rule modeling propagation and collision of logical particles that are encoded by the state values in one cell (or in a number of adjacent

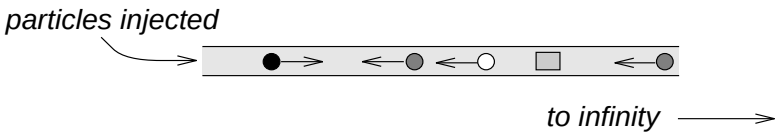

Figure 2: The basic conception of a particle machine.

cells). Particles propagate with constant velocities. Two or more particles may collide; a set of *collision rules*, which are encoded by the CA update rule, specifies which particles are created, which are destroyed, and which are unaffected in collisions. A PM begins with a finite *initial configuration* of particles and evolves in discrete time steps.

3.3 Simple computation with PMs

Figure 2 shows the general arrangement of a 1-d PM. Particles are injected at one end of the one-dimensional CA, and these particles move through the medium provided by the cells. When two or more particles collide, new particles may be created, existing particles may be annihilated, or no interaction may occur, depending on the types of particles involved in the collision.

Figure 3 illustrates some typical collisions when binary addition is implemented by particle collisions. This particular method of addition is only one of many possibilities. The basic idea here is that each addend is represented by a stream of particles containing one particle for each bit in the addend, one stream moving left and the other moving right. The two addend streams collide with a ripple-carry adder particle where the addition operation takes place. The ripple-carry particle keeps track of the current value of the carry between collisions of subsequent addend-bit particles as the streams collide least-significant-bit first. As each collision occurs, a new right-moving result-bit particle is created and the two addend particles are annihilated. Finally, a trailing "reset" particle moving right resets the ripple-carry to zero and creates an additional result-bit particle moving right.

3.4 Algorithms

3.4.1 Arithmetic

Addition and subtraction on a PM are relatively straightforward to implement, and both can operate in linear time and with arbitrary precision. A multiplication algorithm with similar properties is not much more difficult to obtain, but a linear-time, arbitrary-precision division algorithm is somewhat more involved. We briefly describe some arithmetic algorithms, and we refer the reader to [17, 20, 21] for details.

Figure 4 shows the particle arrangement for fixed-point multiplication. This mirrors a well known systolic array for the same purpose [18], but of course the structure is "soft" in the sense that it represents only the input stream of the PM that accomplishes the operation. Figure 5 shows a simulation of this multiplication scheme for the product $11_2 * 11_2$. In that figure, the particles depicted by r and R represent right-moving 0 and 1 operand bits, respectively, and L and l similarly represent left-moving operand bits; p represents stationary "processor" particles in the computation region where the product is formed; c represents "carry" particles propagated during computation; and 0 and 1 represent stationary bits of the product. The top of the figure shows two operands (11_2 and 11_2) on their way towards collisions in the central computation region containing stationary "processor" particles; the bottom of the figure shows the same operands emerging unchanged from the computation, with the answer (1001_2) remaining in the central computation region.

For division, a PM can implement a linear-time, arbitrary-precision algorithm based on Newtonian iteration and described by Leighton [22]. Figure 6 shows a simulation of such an algorithm running on a PM. Details of this algorithm can be found in [20, 21].

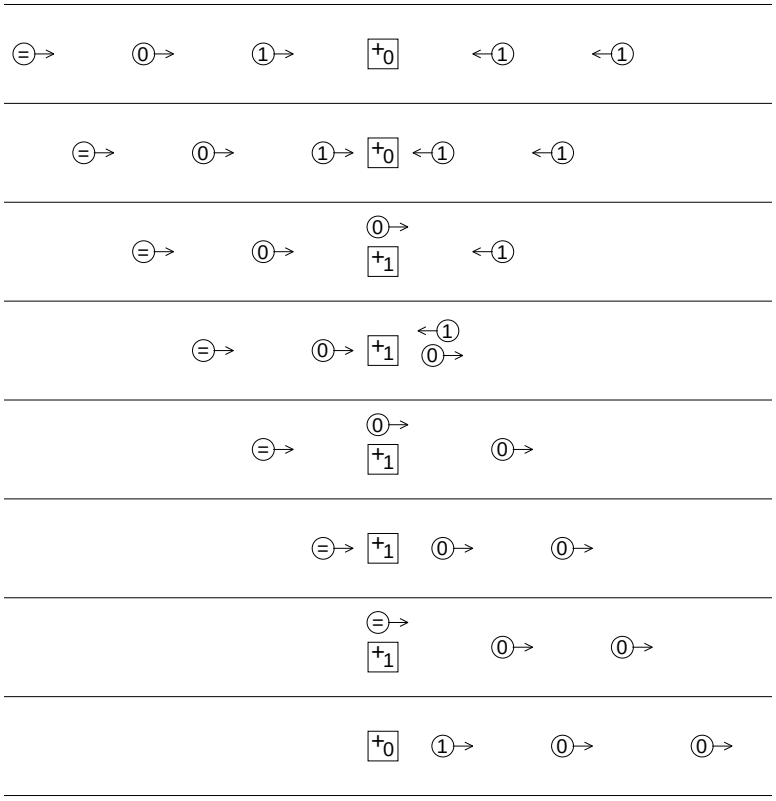

Figure 3: An example illustrating some typical particle collisions, and one way to perform addition in a particle machine. What is shown is actually the calculation $01_2 + 11_2 = 100_2$, implemented by having the two operands, one moving left and the other moving right, collide at a stationary "ripple-carry" particle. When the leading, least-significant bits collide (in the third row from the top of the figure), the ripple-carry particle changes its identity so that it encodes a carry bit of 1, and a right-moving sum particle representing a bit value of 0 is created. The final answer emerges as the right-moving stream 100_2, and the ripple-carry particle is reset by the "equals" particle to encode a carry of 0. The bits of the two addends are annihilated when the sum and carry bits are formed. Notice that the particles are originally separated by empty cells, and that all operations can be effected by a CA with a neighborhood size of 3 (a radius of 1).

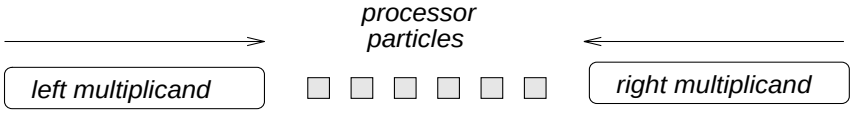

Figure 4: Multiplication scheme, based on a systolic array. The processor particles are stationary and the data particles collide. Product bits are stored in the identity of the processor particles, and carry bits are stored in the identity of the data particles, and thereby transported to neighbor bits.

```
.R   .   .R   .   .p   .p   .p   .p   .   .   .L   .   .L   .
.   .R   .   .R  .p   .p   .p   .p   .   .L   .   .L   .   .
.   .R   .   .R  .p   .p   .p   .p   .   .L   .   .L   .   .
.   .   .R   .   .Rp  .p   .p   .p   .L   .   .L   .   .   .
.   .   .R   .   .Rp  .p   .p   .p   .L   .   .L   .   .   .
.   .   .   .R  .p   .Rp  .p   .Lp  .   .L   .   .   .   .
.   .   .   .R  .p   .Rp  .p   .Lp  .   .L   .   .   .   .
.   .   .   .   .Rp  .p   .RLp .p   .L   .   .   .   .   .
.   .   .   .   .Rp  .p   .RL1 .p   .L   .   .   .   .   .
.   .   .   .   .p   .RLp .1   .RLp .   .   .   .   .   .
.   .   .   .   .p   .RL1 .1   .RL1 .   .   .   .   .   .
.   .   .   .   .Lp  .1   .RL1 .1   .R   .   .   .   .   .
.   .   .   .   .Lp  .1   .RL0c.1   .R   .   .   .   .   .
.   .   .   .L  .p   .L1c .0   .R1  .   .R   .   .   .   .
.   .   .   .L  .p   .L0c .0   .R1  .   .R   .   .   .   .
.   .   .L   .   .Lpc .0   .0   .1   .R   .   .R   .   .   .
.   .   .L   .   .L1  .0   .0   .1   .R   .   .R   .   .   .
.   .L   .   .L  .1   .0   .0   .1   .   .R   .   .R   .   .
.   .L   .   .L  .1   .0   .0   .1   .   .R   .   .R   .   .
.L   .   .L   .   .1   .0   .0   .1   .   .   .R   .   .R   .
.L   .   .L   .   .1   .0   .0   .1   .   .   .R   .   .R   .
```

Figure 5: Simulator output for PM multiplication. The top line in the figure gives the initial state of the PM's medium, representing the multiplication problem $11_2 * 11_2$, as described in the text. Each successive pair of lines depicts the state of the medium after the propagation and collision phases of each time step. The bottom line in the picture shows the stationary answer, 1001_2, in the central computation region, along with the unchanged operands moving away from the region.

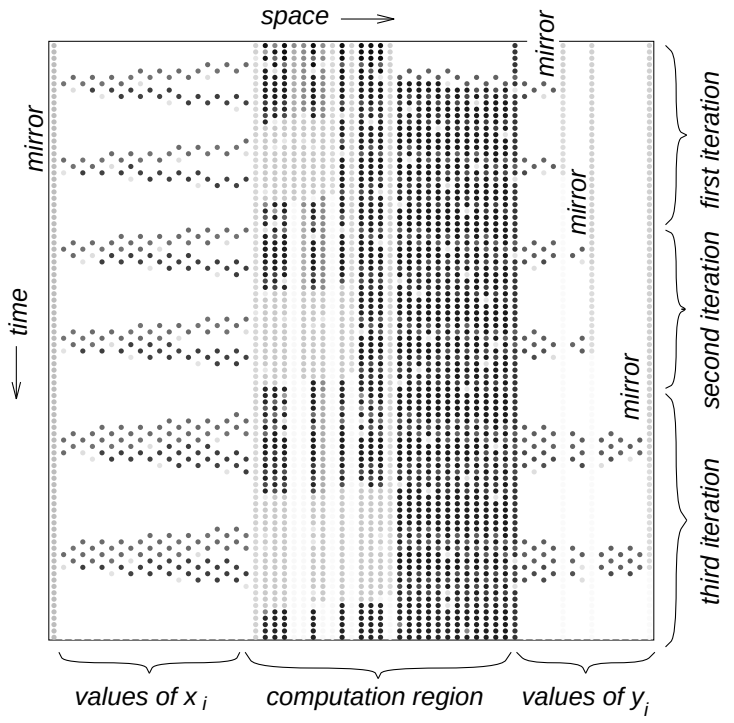

Figure 6: Output generated by a simulation of the division implementation. Each cell is represented by a small circle whose shading depends on which particles are present in that cell. For clarity, only every seventh generation is shown. The example shown is actually the division 1/7.

3.4.2 Convolution, filtering, and other systolic-array algorithms

As mentioned above, arithmetic operations can be nested to achieve the pipelined parallelism inherent in systolic arrays [17]. This leads to highly parallel, pipelined particle machine implementations of convolution, filtering, and other common digital signal processing algorithms, such as the FFT. Similar non-numerical algorithms with systolic implementations also fit in this category (see, for example, [23]) and are amenable to the same soft realizations.

3.5 Comment on VLSI Implementation

Whether it is actually advantageous to implement applications this way in VLSI is an interesting question. The tradeoff is clearly between the efficiencies of using fixed, modular, concatenated chips in a very long pipeline on the one hand, and the inefficiencies in transferring all the problem coding to a very low-level "program" in terms of particles. Where that tradeoff ends up depends a great deal on technology and economies of production scale.

We will not pursue this question of VLSI implementation further in this paper, but rather follow the road that leads to particles as solitons in nonlinear optical materials.

3.6 Particles in other automata

Before we leave CA, we mention some related work on CA with particles. Crutchfield, Das, D'haeseleer, Hanson, Hordijk, Mitchell, Nimwegen, and others report intriguing work in which CA are evolved to perform some simple computational tasks (see [24], for example, and the web site www.santafe.edu/~evca). Particles appear in these evolved CAs quite spontaneously, suggesting that they may be a very natural way to embed computation in regular structures and materials. Boccara, Nasser, and Roger [25] describe a wide variety of particles observed in a conventional CA. Takahashi [26] presents an appealingly simple box-and-ball model for a CA that supports solitons. Santini [27] extends the concept of integrability to algebraic and functional equations, as well as CA, including the joint work with Bruschi and Ragnisco [10].

Finally, we also mention some additional, historically significant work: the very simple universal model using ideal elastically colliding billiard balls in the plane [28, 29]; the exhaustive study of universal dynamic computations by Adamatzky [30]; the very early example of pipelined computation in a one-dimensional CA by Atrubin [31]; Conway's universal game of Life in 2+1 dimensions [32], and perhaps the simplest known universal CA in 2+1 dimensions [33].

4 Solitons and computation

4.1 Scalar envelope solitons

It is a natural step from trying to use solitons in CA to trying to use "real" solitons in physical systems, and the most promising candidates appear to be optical solitons in nonlinear media such as optical fibers and photorefractive crystals. We can envision such computation as taking place via collisions inside a completely uniform medium and aside from its inherent theoretical interest might ultimately offer the advantages of high speed, high parallelism, and low power dissipation. We cannot review here in any detail the fascinating development of soliton theory and its application to optical solitons, but the reader is referred to [34], still a classic beginning reference, and the general book [35] for further reading. We will however review the essential features of *envelope* solitons, which are most relevant to our work.

For our purposes a soliton can be defined as in [36]:

Definition 2 *A soliton is a solitary wave which asymptotically preserves its shape and velocity upon nonlinear interaction with other solitary waves, or, more generally, with another (arbitrary) localized disturbance.*

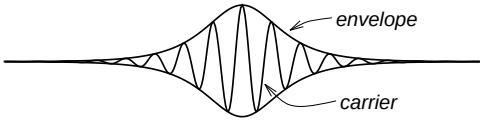

Figure 7: An envelope soliton.

We focus on solitons that arise in systems described by the nonlinear Schrödinger (NLS) equation

$$iu_t + Du_{xx} + N(|u|)u = 0, \tag{1}$$

where D is a real number, and N an arbitrary operator on $|u|$, and the subscripts denote partial differentiation. This describes nonlinear wave propagation in a variety of physical systems, most notably certain optical fibers, where to first order $N(|u|) = |u|^2$, and in certain (so-called saturable) photorefractive crystals, where $N(|u|) = m + k|u|^2/(1 + |u|^2)$, where k and m are real constants. In the former instance we will call the equation the *cubic NLS* (3-NLS), and in the latter, the *saturable NLS* (sat-NLS). The solitons that result in these systems are called *envelope solitons*, and as shown in Figure 7, they are complex-valued wave packets, consisting of a *carrier wave* moving at a characteristic *phase velocity*, modulated by an *envelope*, moving at a characteristic *group velocity*.

4.2 Integrable and nonintegrable systems

There is a crucial difference in behavior between *integrable* and *nonintegrable* systems. Omitting technical details, the integrable systems we consider are analytically solvable, and collisions between solitons are perfectly *elastic*. That is, solitons emerge from collisions with all their original energy. Collisions in nonintegrable systems are characterized by *radiation*—energy that is lost from the solitons in the form of waves radiating away from the collision site. Such unavoidable energy loss means that collisions cannot be cascaded in many stages, and that any useful computational system must entail restoration of full-energy solitons.

Clearly, some particle-like behavior is sacrificed in nonintegrable systems, and in fact purists (generally the mathematical physicists), reserve the term *soliton* for integrable systems only. Particle physicists on the other hand are more forgiving, and we will follow their lead in using the term *soliton* more loosely [37, 38].

4.3 The cubic NLS

The most obvious candidate for a useful soliton system is the integrable equation, 3-NLS. This is one of the two or three best-studied soliton equations, and the resultant sech-shaped solitons have been observed experimentally in real optical fibers for many years. To proceed, we need to identify some soliton parameters as *state variables* that can be used to carry information. Of the possible parameters, the amplitude and velocity can be ruled out because they are unaffected by collisions. The remaining parameters are the carrier phase and positional phase (location). Now what happens in 3-NLS collisions is very disappointing from the point of view of computation: the values of the state variables that can change do not have any effect on the results of subsequent collisions. This rules out communication of information from soliton to soliton and effectively rules out useful computation in 3-NLS.

4.4 Oblivious and transactive collisions

We next introduce two definitions that allow us to state the preceding argument somewhat more precisely.

Definition 3 For a given system define the *state* of a soliton to be a set of selected parameters that can change during collisions.

Definition 4 Collisions of solitons in a given system are termed *transactive* if some changes in the state of one colliding soliton depend on the state of the other. If collisions are not transactive, they are termed *oblivious*.

We also call systems themselves transactive or oblivious. We see therefore that 3-NLS is oblivious. The key problem then becomes finding a transactive system.

4.5 The saturable NLS

At the time this obstacle was encountered it seemed to us that all integrable systems are oblivious, and we began looking at some nonintegrable systems, which strictly speaking do not support solitons, but which in fact support near-solitons [39]. At this point M. Segev brought sat-NLS to our attention, an equation that describes the recently discovered 1+1-dimension (one space and one time dimension) photorefractive optical spatial solitons in steady state [40, 41, 42], and optical spatial solitons in atomic media in the proximity of an electronic resonance [43]. A numerical study revealed definite transactivity [44]. But the observed effect is not dramatic, and it comes at the cost of unavoidable radiation.

At this point it appeared that transactivity and elastic collisions were somehow antagonistic properties, and that integrable systems were doomed to be oblivious. A pleasant surprise awaited us.

5 Computation in the Manakov system

The surprise came in the form of the paper by R. Radhakrishnan, M. Lakshmanan and J. Hietarinta [45], which gave a new bright two-soliton solution for the *Manakov* system [46], and derived explicit asymptotic results for collisions. The solutions were more general than any given previously, and were remarkable in demonstrating what amounts to pronounced transactivity in perfectly integrable equations. The Manakov system consists of two coupled 3-NLS equations, and models propagation of light in certain materials under certain circumstances. The two coupled components can be thought of as orthogonally polarized. Manakov solitons have been recently observed experimentally in [47].

This brings our review to very recent work. The Manakov system is less well known than 3-NLS or sat-NLS, so we will describe it in some detail, following [48].

5.1 The Manakov system and its solutions

As mentioned, the Manakov system consists of two coupled 3-NLS equations,

$$\begin{aligned} iq_{1t} + q_{1xx} + 2\mu(|q_1|^2 + |q_2|^2)q_1 &= 0, \\ iq_{2t} + q_{2xx} + 2\mu(|q_1|^2 + |q_2|^2)q_2 &= 0, \end{aligned} \tag{2}$$

where $q_1 = q_1(x,t)$ and $q_2 = q_2(x,t)$ are two interacting optical components, μ is a positive parameter, and x and t are normalized space and time. Note that in order for t to represent the propagation variable, as in Manakov's original paper [46], our variables x and t are interchanged with those of [45]. The system admits single-soliton solutions consisting of two components,

$$\begin{aligned} q_1 &= \frac{\alpha}{2} e^{-\frac{R}{2}+i\eta_I} \mathrm{sech}(\eta_R + \frac{R}{2}), \\ q_2 &= \frac{\beta}{2} e^{-\frac{R}{2}+i\eta_I} \mathrm{sech}(\eta_R + \frac{R}{2}), \end{aligned} \tag{3}$$

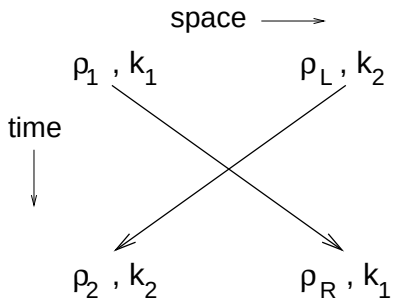

Figure 8: A general two-soliton collision in the Manakov system. The complex numbers ρ_1, ρ_L, ρ_2, and ρ_R indicate the variable soliton states; k_1 and k_2 indicate the constant soliton parameters.

where

$$\eta = k(x + ikt), \tag{4}$$

$$e^R = \frac{\mu(|\alpha|^2 + |\beta^2|)}{k + k^*}, \tag{5}$$

and α, β, and k are arbitrary complex parameters. Subscripts R and I on η and k indicate real and imaginary parts. Note that $k_R \neq 0$. Solitons with more than one component are called *vector* solitons.

5.2 State in the Manakov system

The three complex numbers α, β, and k (with six degrees of freedom) in eq. 3 characterize bright solitons in the Manakov system. The complex parameter k is unchanged by collisions, so two degrees of freedom can be removed immediately from an informational state characterization. We note that Manakov [46] removed an additional degree of freedom by normalizing the polarization vector determined by α and β by the total magnitude $(\alpha^2 + \beta^2)^{1/2}$. However, it is a remarkable fact that the single complex-valued polarization state $\rho = \alpha/\beta$, with only two degrees of freedom [49], suffices to characterize two-soliton collisions when the constants k of both solitons are given [48].

We use the tuple (ρ, k) to refer to a soliton with variable state ρ and constant parameter k:

- $\rho = q_1(x,t)/q_2(x,t) = \alpha/\beta$: a complex number, constant between collisions;
- $k = k_R + ik_I$: a complex number, with $k_R \neq 0$.

We use the complex plane extended to include the point at infinity.

Consider a two-soliton collision, and let k_1 and k_2 represent the constant soliton parameters. Let ρ_1 and ρ_L denote the respective soliton states before impact. Suppose the collision transforms ρ_1 into ρ_R, and ρ_L into ρ_2 (see fig. 8). We will always associate k_1 and ρ_1 with the right-moving particle, and k_2 and ρ_L with the left-moving particle. To specify these state transformations, we write

$$T_{\rho_1,k_1}(\rho_L, k_2) = \rho_2, \tag{6}$$

$$T_{\rho_L,k_2}(\rho_1, k_1) = \rho_R. \tag{7}$$

The soliton velocities are determined by k_{1I} and k_{2I}, and are therefore constant.

It turns out that the state change undergone by each colliding soliton takes on the very simple form of a linear fractional transformation (LFT) (also called *bilinear* or *Möbius* transformation). The coefficients are simple functions of the other soliton in the collision. Explicitly, the LFTs are

$$\rho_2 = \frac{[(1-g)/\rho_1^* + \rho_1]\rho_L + g\rho_1/\rho_1^*}{g\rho_L + (1-g)\rho_1 + 1/\rho_1^*}, \tag{8}$$

where

$$g(k_1, k_2) = \frac{k_1 + k_1^*}{k_2 + k_1^*}. \tag{9}$$

and

$$\rho_R = \frac{[(1-h^*)/\rho_L^* + \rho_L]\rho_1 + h^*\rho_L/\rho_L^*}{h^*\rho_1 + (1-h^*)\rho_L + 1/\rho_L^*}, \tag{10}$$

where

$$h(k_1, k_2) = \frac{k_2 + k_2^*}{k_1 + k_2^*}. \tag{11}$$

We assume here, without loss of generality, that $k_{1R}, k_{2R} > 0$.

Several properties of these transformations are derived in [48], including characterization of inverse operators, fixed points, and implicit forms. In particular, when viewed as an operator every particle has an *inverse*, and the two traveling together constitute an *inverse pair*. Collision with an inverse pair leaves the state of every particle intact.

5.3 Particle design for computation

In any particle collision we can view one of the particles as an "operator" and the other as "data." In this way we can hope to find particles and states that effect some useful computation. This is work in progress, and we give some examples that illustrate simple logical operations.

5.3.1 An i operator

A simple nontrivial operator is pure rotation by $\pi/2$, or multiplication by i. This changes linearly polarized solitons to circularly polarized solitons, and vice versa. A numerical search yielded the useful transformations

$$T_{\rho_L}(\rho) = T_{0,1-i}(\rho, 1+i) = (1 - h^*(1+i, 1-i))\rho = \frac{1}{\sqrt{2}} e^{-\frac{\pi}{4}i}\rho, \tag{12}$$

$$T_{\rho_L}(\rho) = T_{\infty,5-i}(\rho, 1+i) = \frac{\rho}{1 - h^*(1+i, 5-i)} = \sqrt{2} e^{\frac{3\pi}{4}i}\rho, \tag{13}$$

which, when composed, result in the transformation

$$U(\rho, 1+i) = i\rho. \tag{14}$$

(Here we think of the data as right-moving and the operator as left-moving.) We refer to U as an i operator. Its effect is achieved by first colliding a soliton $(\rho, 1+i)$ with $(0, 1-i)$, and then colliding the result with $(\infty, 5-i)$, which yields $(i\rho, 1+i)$.

5.3.2 A -1 operator (NOT processor)

Composing two i operators results in the -1 operator, which with appropriate encoding of information can be used as a logical NOT processor. Figure 9 shows a NOT processor with reusable data and operator solitons. The two right-moving particles represent data and are an inverse pair, and thus leave the operator unchanged; the left-moving group comprise the four components of the -1 operator. This figure was obtained by direct numerical simulation of the Manakov system, with initial state that contains the appropriate data and processor solitons.

This NOT processor switches the phase of the (right-moving ± 1) data particles, using the energy partition of the (left-moving 0 and ∞) operator particles. A kind of dual NOT gate exists, which switches the energy of data particles using only the phase of the operator particles. In particular, if we use the same k's as in the phase-switching NOT gate, code data as 0 and ∞, and use a sequence of four ± 1 operator particles, the effect is to switch 0 to ∞ and ∞ to 0—that is, to switch all the energy from one component of the data particles to the other (see fig. 10).

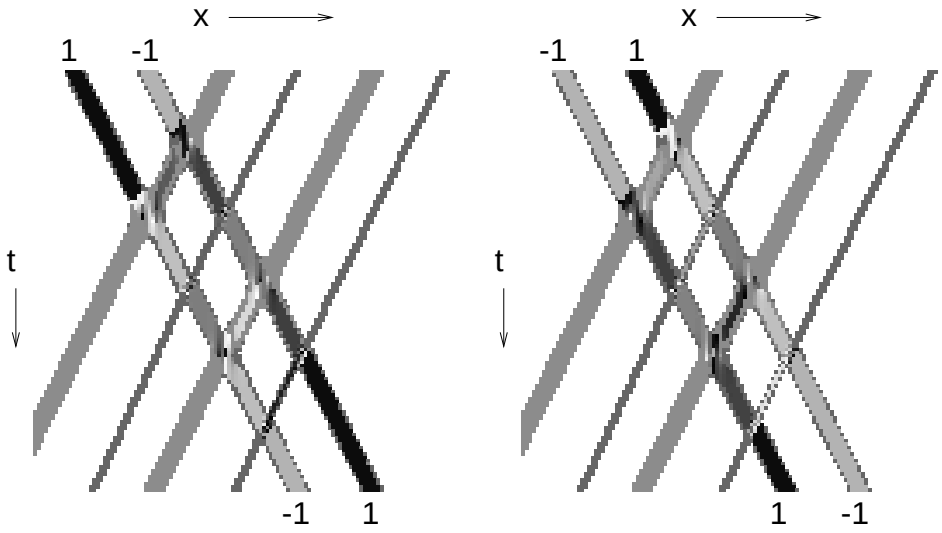

Figure 9: Numerical simulation of a NOT processor implemented in the Manakov system. These graphs display the color-coded phase of ρ for solitons that encode data and processors for two cases. In the initial conditions (top of graphs), the two leftmost (data) solitons are an inverse pair that can represent a 0 in the left graph, and a 1 in the right graph. In each graph, these solitons collide with the four rightmost (processor) solitons, resulting in a soliton pair representing a 1 and a 0, respectively. The processor solitons are unchanged. These graphs were obtained by numerical simulation of eq. 2 with $\mu = 1$.

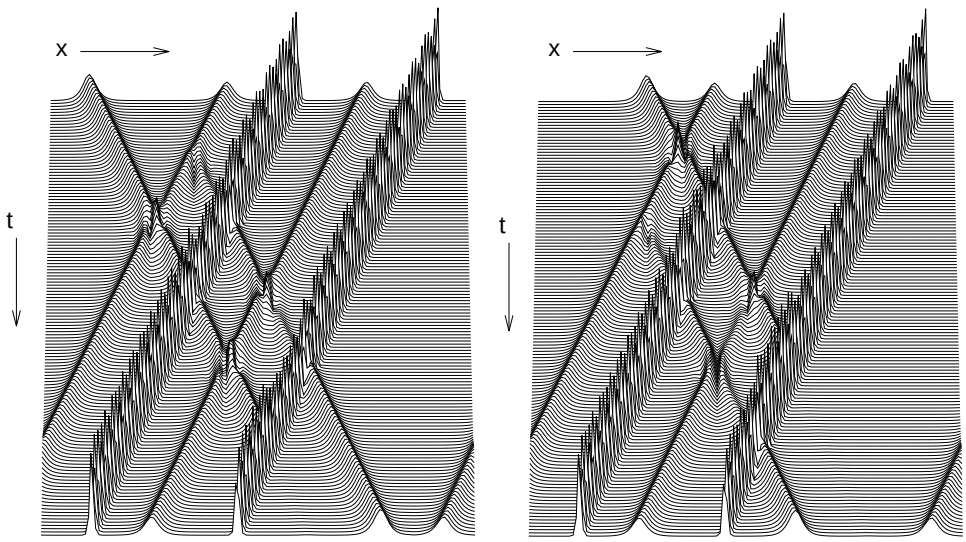

Figure 10: Numerical simulation of an energy-switching NOT processor implemented in the Manakov system. These graphs display the magnitude of one component, for the same two cases as in the previous figure. In this gate the right-moving (data) particles are the inverse pair with states $\infty, 0$ (left), or $0, \infty$ (right) and the first component is shown. As before, the left-moving (operator) particles emerge unchanged, but here have initial and final states ± 1.

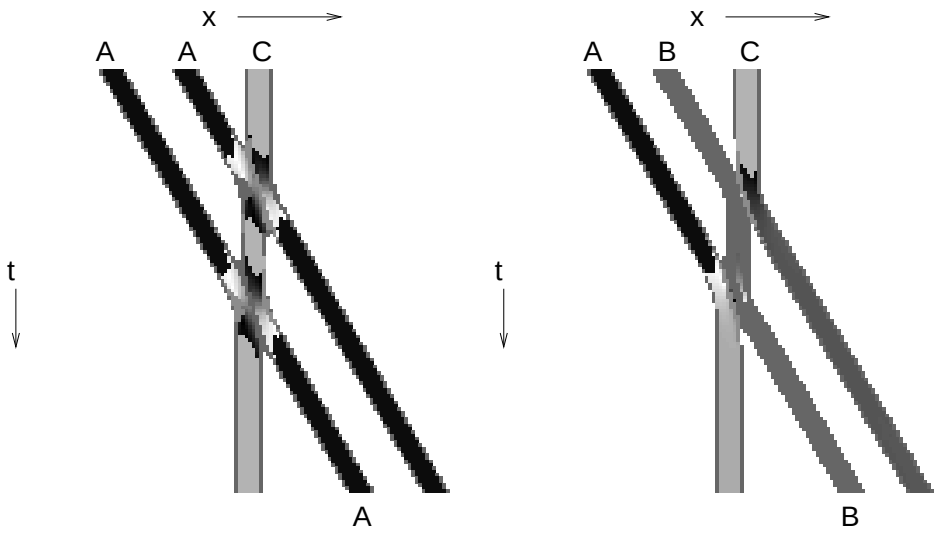

Figure 11: Numerical simulation of a "move" operation implemented in the Manakov system. These graphs display the color-coded phase of ρ. In each graph, the information contained in the middle particle in the initial conditions (top of graphs) is moved to the middle particle in the final conditions (bottom of graphs). The information transfer is effected by the "carrier" particle C. These graphs were obtained by numerical simulation of eq. 2 with $\mu = 1$.

5.3.3 A "move" operator

Figure 11 depicts a simple example of information transfer from one particle to another, reminiscent of an assembly-language MOVE instruction. In the initial conditions of each graph, a "carrier" particle C collides with the middle particle; this collision transfers information from the middle particle to C. The carrier particle then transfers its information to another particle via a collision. The appropriate particles A, B, and C for this operation were found through a numerical search, as with the particles for our NOT gate.

Note that "garbage" particles arise as a result of this "move" operation. In general, because the Manakov system is reversible, such "garbage" often appears in computations, and needs to be managed explicitly or used as part of computation, as with conservative logic [28]. Of course reversibility does not necessarily limit the computational power of the Manakov system, since reversible systems can be universal [50].

6 Conclusion

This brings us to current work, and suggests many open questions, some theoretical, some experimental, and some a mixture of the two. We conclude by mentioning some of these.

In the theoretical area:

- What is a complete mathematical characterization of the state LFTs obtainable by composing either a finite number—or an infinite number—of the Manakov collisions?
- How can we "program" Manakov solitons? (A major obstacle is the one-dimensional nature of the system, and the difficulty of "crossing wires.")
- Is the complex-valued polarization state used here for the Manakov system also useful in other multi-component systems, especially those that are near-integrable and support spatial solitons?
- What is the theoretical computational power of the Manakov and related systems from the point of view of implementing logic of some generality? In particular, which systems in

1+1 or 2+1 dimensions, integrable or nonintegrable, are Turing-equivalent and therefore universal?

- Can 2+1 and higher-dimensional soliton systems be used for efficient computation in uniform media? For example, can a 2+1 integrable system simulate the billiard-ball model of computation, and can such a system be useful without fixed barriers off which balls bounce?

On the experimental side of things:

- Can the Manakov system be implemented in a simple and accurate way?
- Can saturable materials like photorefractive crystals be made that are highly transactive with acceptable radiation?
- What new physical systems might be found that support solitons which can be easily used to compute?

We've followed one particular line of work, from the discovery of solitons in abstract cellular automata, to the abstract model of the particle machine, to possible harnessing of optical vector solitons for useful computation. This reflects one aspect of the growing field that is sometimes called "nonstandard computation," and includes alternatives to the lithographed silicon-chip based paradigm as a physical basis for computation.

As we've seen there are many fascinating questions of interest—to both computer scientists and physicists—about soliton information processing. The very notion that nonlinear waves/particles can encode and process information remains largely unexplored.

7 Acknowledgments

We owe a debt of gratitude to a number of colleagues and co-workers for important help at critical times, most notably Stephen Wolfram fifteen years ago, and Mordechai Segev very recently.

References

[1] S. Wolfram, editor. *Theory and Application of Cellular Automata.* World Scientific, Singapore, 1986.

[2] S. Wolfram. Universality and complexity in cellular automata. *Physica*, 10D:1–35, 1984.

[3] J. K. Park, K. Steiglitz, and W. P. Thurston. Soliton-like behavior in automata. *Physica D*, 19D:423–432, 1986.

[4] T. W. Parks and C. S. Burrus. *Digital Filter Design.* John Wiley, New York, 1987.

[5] K. Steiglitz, I. Kamal, and A. Watson. Embedding computation in one-dimensional automata by phase coding solitons. *IEEE Transactions on Computers*, 37(2):138–145, 1988.

[6] C. H. Goldberg. Parity filter automata. *Complex Systems*, 2:91–141, 1988.

[7] A. S. Fokas, E. Papadopoulou, and Y. Saridakis. Particles in soliton cellular automata. *Complex Systems*, 3:615–633, 1989.

[8] A. S. Fokas, E. Papadopoulou, and Y. Saridakis. Coherent structures in cellular automata. *Physics Letters*, 147A(7):369–379, 1990.

[9] M. J. Ablowitz, J. M. Keiser, and L. A. Takhtajan. Class of stable multistate time-reversible cellular automata with rich particle content. *Phys. Rev. A*, 44A(10):6909–6912, Nov. 15, 1991.

[10] M. Bruschi, P. M. Santini, and O. Ragnisco. Integrable cellular automata. *Phys. Lett. A*, 169:151–160, 1992.

[11] A. I. Adamatzky. Constructing a discrete generalized Voronoi diagram in reaction-diffusion media. *Neural Network World*, pages 635–643, June 1994.

[12] A. I. Adamatzky. On the particle-like waves in the discrete model of excitable medium. *Neural Network World*, pages 3–10, 1996.

[13] A. I. Adamatzky. Computation of shortest path in cellular automata. *Mathl. Comput. Modelling*, 23(4):105–113, 1996.

[14] P. Siwak. On automata of some discrete recursive filters that support filtrons. In S. Domek, R. Kaszynski, and L. Tarasiejski, editors, *Proc. Fifth Int. Symp. on Methods and Models in Automation and Robotics*, volume 3 (Discrete Processes), pages 1069–1074, Międzyzdroje, Poland, Aug. 25–29 1998. Wydawnictwo Politechniki Szczeciinskiej.

[15] P. Siwak. Filtrons and their associated ring computations. *Int. J. General Systems*, 27(1–3):181–229, 1998.

[16] P. Siwak. Iterons, fractals and computations of automata. In D. M. Dubois, editor, *Second Int. Conf. on Computing Anticipatory Systems, CASYS '98, conference proceedings 465*, pages 45–63, Woodbury, New York, August 1999. Amer. Inst. Phys.

[17] R. Squier and K. Steiglitz. Programmable parallel arithmetic in cellular automata using a particle model. *Complex Systems*, 8:311–323, 1994.

[18] H. T. Kung. Why systolic architectures? *IEEE Computer*, 15(1):37–46, January 1982.

[19] U. Frisch, D. d'Humie'res, B. Hasslacher, P. Lallemand Y. Pomeau, and J. P. Rivet. Lattice gas hydrodynamics in two and three dimensions. *Complex Systems*, 1:649–707, 1987.

[20] M. H. Jakubowski, K. Steiglitz, and R. K. Squier. Implementation of parallel arithmetic in a cellular automaton. In *1995 Int. Conf. on Application Specific Array Processors, Strasbourg, France (P. Cappello et al., ed.)*, Los Alamitos, CA, July 24–26, 1995. IEEE Computer Society Press.

[21] M. H. Jakubowski. *Computing with Solitons in Bulk Media (Ph.D. Thesis).* Princeton University, Princeton, NJ, 1998.

[22] F. T. Leighton. *Introduction to Parallel Algorithms and Architectures.* Morgan Kaufman Publishers, San Mateo, CA, 1992.

[23] H.-H. Liu and K.-S. Fu. VLSI arrays for minimum-distance classifications. In K. S. Fu, editor, *VLSI for Pattern Recognition and Image Processing.* Springer-Verlag, Berlin, 1984.

[24] W. Hordijk, J. P. Crutchfield, and M. Mitchell. Embedded-particle computation in evolved cellular automata. In T. Toffoli, M. Biafore, and J. Leão, editors, *Proc. Fourth Workshop on Physics and Computation (PhysComp96)*, pages 153–158, Boston, Mass., Nov. 22–24, 1996. New England Complex Systems Institute.

[25] N. Boccara, J. Nasser, and M. Roger. Particlelike structures and their interactions in spatiotemporal patterns generated by one-dimensional deterministic cellular-automaton rules. *Phys. Rev. A*, 44(2):866–875, 15 July 1991.

[26] D. Takahashi. On a fully discrete soliton system. In M. Boiti, L. Martina, and P. Pempinelli, editors, *Proc. 7th Workshop on Nonlinear Evolution Equations and Dynamical Systems (NEEDS '91)*, pages 245–249, Singapore, 1991. World Scientific.

[27] P. M. Santini. Integrability for algebraic equations, functional equations and cellular automata. In V. Makhankov, I. Puzynin, and O. Pashev, editors, *Proc. 8th Workshop on Nonlinear Evolution Equations and Dynamical Systems (NEEDS '92)*, pages 214–221, Singapore, 1992. World Scientific.

[28] E. Fredkin and T. Toffoli. Conservative logic. *International Journal of Theoretical Physics*, 21(3/4):219–253, 1982.

[29] N. Margolus. Physics-like models of computation. *Physica*, 10D:81–95, 1984.

[30] A. I. Adamatzky. Universal dynamical computation in multidimensional excitable lattices. *Int. J. Theor. Phys.*, 37(12):3069–3108, 1988.

[31] A. J. Atrubin. An iterative one-dimensional real-time multiplier. *IEEE Trans. Electron. Computers*, EC-14:394–399, 1965.

[32] E. R. Berlekamp, J. H. Conway, and R. K. Guy. *Winning Ways for Your Mathematical Plays.* Academic Press, New York, NY, 1982.

[33] T. Serizawa. Three-state Neumann neighbor cellular automata capable of constructing self-reproducing machines. *Systems and Computers in Japan*, 18(4):33–40, 1987.

[34] A. C. Scott, F. Y. F. Chu, and D. W. McLaughlin. The soliton: A new concept in applied science. *Proceedings of the IEEE*, 61(10):1443–1483, 1973.

[35] P. G. Drazin and R. S. Johnson. *Solitons: An Introduction.* Cambridge University Press, Cambridge, UK, 1989.

[36] M. J. Ablowitz and P. A. Clarkson. *Solitons, Nonlinear Evolution Equations, and Inverse Scattering.* Cambridge University Press, Cambridge, UK, 1991.

[37] C. Rebbi and G. Soliani. *Solitons and Particles.* World Scientific, Singapore, 1984.

[38] V. G. Makhankov. *Soliton Phenomenology.* Kluwer Academic Publishers, Norwell, MA, 1990.

[39] M. H. Jakubowski, K. Steiglitz, and R. K. Squier. When can solitons compute? *Complex Systems*, 10(1):1–21, 1996.

[40] M. Segev, G. C. Valley, B. Crosignani, P. DiPorto, and A. Yariv. Steady-state spatial screening solitons in photorefractive materials with external applied field. *Physical Review Letters*, 73:3211–3214, 1994.

[41] M. Shih, M. Segev, G. C. Valley, G. Salamo, B. Crosignani, and P. DiPorto. Observation of two-dimensional steady-state photorefractive screening solitons. *Electronics Letters*, 31(10):826–827, 1995.

[42] M. Shih and M. Segev. Incoherent collisions between two-dimensional bright steady-state photorefractive spatial screening solitons. *Optics Letters*, 21(19):1538–1540, 1996.

[43] V. Tikhonenko, J. Christou, and B. Luther-Davies. Three-dimensional bright spatial soliton collision and fusion in a saturable nonlinear medium. *Physical Review Letters*, 76:2698–2702, 1996.

[44] M. H. Jakubowski, K. Steiglitz, and R. K. Squier. Information transfer between solitary waves in the saturable Schrödinger equation. *Physical Review E*, 56:7267–7273, 1997.

[45] R. Radhakrishnan, M. Lakshmanan, and J. Hietarinta. Inelastic collision and switching of coupled bright solitons in optical fibers. *Physical Review E*, 56:2213–2216, 1997.

[46] S. V. Manakov. On the theory of two-dimensional stationary self-focusing of electromagnetic waves. *Soviet Physics: JETP*, 38(2):248–253, Feb. 1974.

[47] J. U. Kang, G. I. Stegeman, J. S. Aitchison, and N. N. Akhmediev. Observation of Manakov spatial solitons in AlGaAs planar waveguides. *Phys. Rev. Lett.*, 76:3699–3702, 1996.

[48] M. H. Jakubowski, K. Steiglitz, and R. Squier. State transformations of colliding optical solitons and possible application to computation in bulk media. *Physical Review E*, 58:6752–6758, 1998.

[49] A. Yariv and P. Yeh. *Optical Waves in Crystals.* Wiley, New York, 1984.

[50] C. H. Bennett. Logical reversibility of computation. *IBM Journal of Research and Development*, 17:525–532, 1973.