MOLECULAR SWITCHING BY TURING AUTOMATA

Miklós Bartha[1] and Miklós Krész[2]

[1]Department of Computer Science, Memorial University of Newfoundland, Canada
Email: bartha@mun.ca

[2]Department of Applied Informatics, University of Szeged, Hungary
Email: kresz@jgypk.u-szeged.hu

Abstract

We study the switching aspects of molecular computing from a novel algebraic point of view. Our approach is based on the concept indexed monoidal algebra, which provides an equivalent formalism for compact closed categories being used recently in the literature in connection with quantum computing. The point is to separate syntax, as the algebra $\mathcal{G}$ of graphs, from semantics, which is related to the algebra $\mathcal{T}$ of Turing automata, and define meaning as a homomorphism. Eventually, the syntax is restricted to the Gallai-Edmonds algebra $\mathcal{G}$-$\mathcal{E}$ of graphs having a perfect internal matching, and the corresponding semantical structure, defined as the algebra $\mathcal{S}$ of soliton automata, is the quotient of an appropriate subalgebra of $\mathcal{T}$.

1. Introduction

The idea of molecular switching is about capturing the dynamic behavior of so called conjugated systems in the molecules of certain organic compounds. Such a system is a group of atoms covalently bonded with an alternating pattern of single and double bonds, so that the result of a "switch" is an appropriate rearrangement of these bonds inside the molecule. Some practical aspects of designing circuits at the molecular level along these lines have been explained in [10].

The first mathematical model to describe the phenomenon of molecular switching by means of automata appeared in [11] under the name soliton automaton. The underlying object of a soliton automaton is an undirected graph representing the topological structure of a molecule. In terms of graph theory, a conjugated system appears in this graph as a perfect matching [19]. States are therefore perfect matchings in soliton automata, and transitions are induced by making suitable alternating walks in the underlying graph. Building on this observation, the study of soliton automata was continued on the grounds of matching theory by a series of our own works. For the interested reader, the results of this study have been summarized as a book chapter in [3].

The aim of the present paper is to put soliton automata in a new perspective, showing that they are in fact simple Turing machines in a slightly generalized sense. As computational devices, these automata are naturally reversible and fit nicely in the algebraic/category theoretical framework developed by Abramsky and others [1, 21] to capture some of the basic properties of quantum-mechanical systems in terms of abstract equational axioms. For an evidence, the indexed monoidal algebra of soliton automata given in Section 4 is completely analogous to that of quantum Turing automata introduced in [2] by the first author just recently. We believe that this a very promising sign regarding the future of soliton automata, even though their present computational power appears to be rather limited.

2. Turing graph machines and automata

Turing graph machines are natural generalizations of classical Turing machines, in which the rigid topology of tape cells as a linear array is replaced by a flexible graph structure. In the small, a

Turing automaton is an abstract machine associated with one cell in this structure. At large, such an automaton also captures the behavior of the whole graph as a system.

The intuitive idea of Turing graph machines is dual to that of cellular automata. A graph machine comes with an underlying undirected graph G in which some of the vertices with degree 1 are distinguished and labeled, each with a different label. These vertices are called *external*, for they identify interfaces to the system modeled by G. All other vertices are *internal*, and they represent cells in the corresponding graph machine. The labels of the external vertices will always be chosen from the set $[n] = \{1, \dots n\}$, where n is the number of such vertices. Graph G itself is called *open* if $n \geq 1$, and *closed* if $n = 0$. We write $G : n$ to explicitly indicate the number of external vertices in G. As usual, the set of vertices and edges of G will be denoted by $V(G)$ and $E(G)$, respectively. An edge $e \in E(G)$ is called *internal* if both endpoints of e are such, *external* if exactly one endpoint is external, and *interface* if both endpoints are external.

Each internal vertex v with degree $d(v)$ is associated with a Turing automaton A_v, which tells how the local state at v changes when control passes through v entering from direction i and leaving in direction j, $i, j \leq d(v)$. The behavior of the whole system represented by G is an automaton of the same kind, describing how the global state (i.e., the product of the local ones) changes when control is initiated at interface (external vertex) k and is expected to come out of the maze of local transitions at interface l. Thus, G is just a flowchart [15] in which all interconnections are bidirectional. In comparison with classical Turing machines, a local state is a symbol of the tape alphabet, and a direction (or degree) corresponds to a combination of the current state and current direction of the Turing machine. At this time G is supposed to be finite, however, unlike the set of states of the automata, which might be countably infinite. Nevertheless, the general idea is that a local Turing automaton A_v in graph G corresponds to a single tape cell drawn from a hypothetical Turing machine, hence it has a finite number of states. It is the graph structure that needs to be stretched by appropriate means (e.g. by graph grammars) to construct Turing automata with an infinite number of states.

To be more precise about the syntax of Turing graph machines, each internal vertex v in the underlying graph G with $d(v) = n$ is labeled by a symbol $\sigma \in \Sigma_n$, where Σ is a fixed ranked alphabet. The symbol σ is a syntactical reference to the Turing automaton A_v. In addition, the degrees of v themselves must be distinguished, e.g. by assigning a linear order to them. The intention is to clearly indicate the flow of control through v. Two graphs $G : n$ and $G' : n$ are called *isomorphic* if they are such as ordinary graphs by an isomorphism that preserves the external vertices and the labeling information of the internal vertices as well. To make the necessary distinction, graphs over the alphabet Σ will be called Σ-*graphs*. The alphabet Σ^u in which there is a single symbol σ_n for each rank n will play a distinguished role in our presentation. This alphabet will be referred to as the *uniform* one.

The formal definition of Turing automata is adopted from [4]. For a set A, let $A_\star = A + \{\star\}$, where $\star$ is a fixed symbol, called the *anchor*.

Definition 1. A *Turing automaton* (TA, for short) $T : A$ is a triple (A, Q, δ), where A is a set of *interfaces*, Q is a nonempty set of *states*, and $\delta \subseteq (Q \times A_\star)^2$ is the *transition relation*.

Two automata $T = (A, Q, \delta)$ and $T' = (A, Q', \delta')$ are called *isomorphic*, notation $T \cong T'$, if there exists a bijection $\chi : Q \to Q'$ such that

$$(\chi^{-1} \times id_{A_\star}) \circ \delta = \delta' \circ (\chi \times id_{A_\star}).$$

In the sequel we shall not distinguish between isomorphic automata, unless otherwise stated. In practice, however, isomorphism classes of automata will be dealt with through representatives, bearing in mind the issue of compatibility with the operations to be defined in the next section.

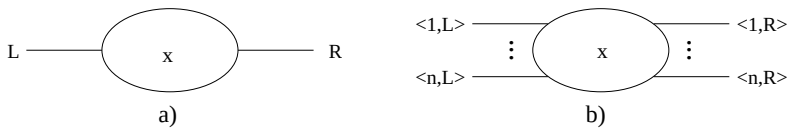

Figure 1: One tape cell in a Turing machine.

Due to the discrete nature of the semantics involved, we shall assume that the cardinality of the sets A and Q is finite and countable, respectively. The anchor is a fixed "invisible" interface, which will allow the automaton T to have transitions even if $A = \emptyset$. As a distinction, interfaces different from the anchor will be called *real*. The transition relation δ can either be considered as a function $Q \times A_\star \to \mathcal{P}(Q \times A_\star)$ or a function $Q \times (A_\star \times A_\star) \to \mathcal{P}(Q)$, giving rise to a Mealy or Medvedev type automaton, respectively. We shall favor the latter interpretation, and call $\Theta_A = A_\star \times A_\star$ the *input alphabet* for T. Then, as it is customary in automata theory, the extension of δ to input strings in Θ_A will be denoted by δ, too. By the standard definition, T is *deterministic* if δ is a partial function in the chosen Medvedev sense. In contrast, T is *strongly deterministic* if δ is a partial function in the Mealy way. Despite the Medvedev style formalism we still say that T has a transition from a to b in state q, resulting in state r, if $((q, a), (r, b)) \in \delta$ (that is, $r \in \delta(q, (a, b))$).

Example 1. In Fig. 1a, consider the primitive model of one tape cell in a Turing machine M holding a symbol x. If M has tape alphabet Γ and states $\{s_1, \ldots, s_n\}$, then we represent this cell by the TA $C = (A + A, Q, \delta)$, where $A = [n]$, $Q = \Gamma$, and δ is essentially the transition relation of M. See Fig. 1b. Recall that the transition relation of M specifies one move of M as a 5-tuple (s_i, x, s_j, y, D), where $x, y \in \Gamma$ and D is a direction, i.e., L or R. Spelling this out, if M is in state s_i and its tape head is scanning a cell that holds symbol x, then M changes its state to s_j, rewrites the symbol x to y, and its tape head moves into direction D. Our interpretation δ of one move also takes into account the direction where the tape head is coming from, but this is a trivial issue not affecting the computational power of Turing machines. (Indeed, one might consider states (s_i, L) and (s_i, R) in a variant of M.) The main point of this analogy is the duality that states of M are viewed as interfaces (inputs) of C, whereas tape symbols (inputs) of M are in fact states of C.

To adjust the general definition of Turing automata to our current model, the set of interfaces A will always be chosen as $[n]$ for some $n \geq 0$. With the anchor fixed as $\star \equiv 0$, $A_\star = \langle n \rangle = \{0, \ldots, n\}$. We say that the automaton T is of *sort* n, and write $T : n$.

Example 2. The *n-ary atomic switch* is the Turing automaton $\mathcal{A}_n : n$ ($n \geq 1$) having states $[n]$, so that

$$\delta = \{((\mathbf{i}, j), (\mathbf{j}, i)) \,|\, 1 \leq i \neq j \leq n\} \cup \{((\mathbf{i}, i), (\mathbf{j}, j)) \,|\, 1 \leq i \neq j \leq n\}$$
$$\cup \{((\mathbf{i}, 0), (\mathbf{i}, j)), ((\mathbf{i}, j), (\mathbf{i}, 0)) \,|\, i, j \in [n]\}.$$

In addition, if $n = 1$, then $((\mathbf{1}, 1), (\mathbf{1}, 1)) \in \delta$. For better readability, states, indicating a selected edge in an n-star graph, are written in boldface.

Heuristically, the n-ary atomic switch captures the behavior of an atom in a molecule having n neighboring atoms to which it is connected by a single or double chemical bond in any given state q. Among these bonds exactly one is double, and the corresponding edge in the underlying star graph $K_{1,n} : n$ is referred to as the *positive* edge with respect to q. (Notice that, for $n = 1$, only one vertex of $K_{1,1}$ is external, even though the other has degree 1, too.) The rest of the edges in $K_{1,n}$ are called *negative* with respect to q. The mechanism of switching is then clear by the definition above. The active ingredient (control) in this process is called the *soliton*, which is a form of energy traveling in

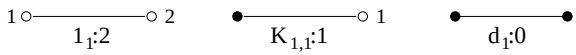

Figure 2: Three simple line graphs.

small packets through chains of alternating single and double bonds within the molecule, causing the affected bonds to be flipped from single to double (negative to positive) and vice versa. See [12] for the physico-chemical details, and [11, 5] for the corresponding mathematical model. Note that, by our definition above, whenever the soliton enters an atom with a unique chemical bond (which must be double since $n = 1$), it bounces back immediately, producing no state change.

In our algebraic framework we shall also need two trivial Turing automata $\bot : 0$ and $\mathcal{A}_0 : 0$. The automaton $\mathcal{A}_0$ has a single state $\mathbf{0}$ with an anchor-to-anchor transition in that state. Intuitively, $\mathcal{A}_0$ is the behavior of an isolated vertex (the graph $K_{1,0} : 0$) as a "nullary" atomic switch. The automaton $\bot$, as the meaning of the the empty graph $K_{0,0} : 0$, also has a single state, but it has no transitions whatsoever. The graph $1_1 : 1 + 1$, which consists of a single interface edge connecting the external vertices 1 and 2, is called the *identity*. Its meaning as a Turing automaton $\mathbf{1}_1 : 1 + 1$ has a single state $\mathbf{0}$, has a (unique) transition on input $(i, 2 - i)$ for both $i = 1, 2$, but no transitions involving the anchor.

At this point the reader might wonder how many non-isomorphic graphs we have with a single edge connecting two distinct vertices. The answer is given in Fig. 2: the identity graph $1_1 : 1 + 1$, the 1-star $1 = K_{1,1} : 1$, and the "dumbbell" $d_1 : 0$. We also introduce an abstract "sink" graph $\bot : 0$, which is just another isolated vertex labeled by the symbol $\bot$. The meaning of this graph as a Turing automaton will simply be $\bot$, the same as that of the empty graph $K_{0,0}$. The intention is to have the algebra of Σ-graphs freely generated as a traced monoidal category [17] from the labeled star graphs. Indeed, the trace of the identity $1_1 : 1 + 1(1 \to 1)$ as a graph of sort 0 $(0 \to 0)$ must not be the empty graph $1_0 = K_{0,0}$ by the free interpretation of trace, even though this is what one would normally expect under all reasonable interpretations. For a counterexample, in the well-known traced monoidal category of finite dimensional vector spaces over the real or complex field F, the trace $Tr_{\mathcal{V}} 1_{\mathcal{V}}$ of the identity on any object (space) $\mathcal{V}$ is the dimension of $\mathcal{V}$, and not the identity on F (represented by the number 1) as the morphism 1_F in that category. See [2] for details.

3. The algebra of graphs and Turing automata

There are a number of natural operations that are common to graphs and Turing automata. We are going to describe these operations as a many-sorted algebraic structure over the set N of non-negative integers as sorts. For the origins of this structure, the reader is referred to [4], where the general definition of indexed monoidal algebras over an arbitrary monoid M was worked out. We simply adopt that definition below by fixing the monoid M to be $(N, 0, +)$. One of the key ideas in this definition is indexing an element f of sort n by a permutation $\rho : n \to n$. Intuitively, this amounts to a relabeling of the interfaces according to ρ. For $n, m \in N$, let 1_n be the identity permutation $n \to n$, and $c_{n,m}$ be the symmetry (block transposition) $n + m \to m + n$. As it is well-known, permutations as morphisms define a strict symmetric monoidal category [20] Π over the objects (monoid) N. Composition and tensor in Π will be denoted by $\bullet$ and $\oslash$, respectively. Now the operations and constants of an N-indexed algebra $\mathcal{N} = \{\mathcal{N}_n | n \in N\}$ are the following:

– For each permutation $\rho : n \to n$, a unary operation $\rho : \mathcal{N}_n \to \mathcal{N}_n$.
– For each $n, m \in N$, a binary operation sum, $\oplus : \mathcal{N}_n \times \mathcal{N}_m \to \mathcal{N}_{n+m}$.
– For each $n \in N$, a constant $\mathbf{1}_n \in \mathcal{N}_{n+n}$.
– For each $n, m \in N$, a unary operation trace, $\updownarrow_n : \mathcal{N}_{n+n+m} \to \mathcal{N}_m$.

To emphasize the categorical nature of such algebras we call the elements $f \in \mathcal{N}_n$ morphisms and write $f : n$. We also write $f : n \to m$ as an alternative for $f : n + m$, being aware of the ambiguity arising from the indefinite choice of n and m. Note that left-cancellativity of our monoid N is needed to make the trace operation sound. Moreover, the accurate notation for trace would be $\updownarrow_{n,m}$, but the intended object (number) m will always be clear from the context. Also notice the boldface notation $\mathbf{1}_n : n \to n$ as opposed to $1_n : n \to n$ as a permutation. For better readability we shall write $f \cdot \rho$ for $f\rho$, that is, for indexing f by permutation ρ.

Composition ($\circ$) and tensor ($\otimes$) are introduced in $\mathcal{N}$ as derived operations in the following way.

– For $f : n \to m$ and $g : m \to k$, $f \circ g = \updownarrow_m ((f \oplus g) \cdot (c_{n,m+m} \oslash 1_k))$.
– For $f : n \to m$ and $g : k \to l$, $f \otimes g = (f \oplus g) \cdot (1_n \oslash c_{m,k} \oslash 1_l)$.

See Fig. 3. Again, the accurate notation for $\circ$ and $\otimes$ would use the objects n, m, k, l as subscripts, but these numbers will always be clear from the context. Observe that the above definition of composition and tensor is in line with the traced monoidal category axioms [6, 17].

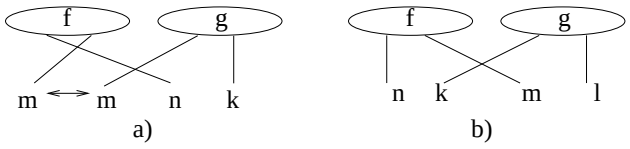

Figure 3: Composition (a) and tensor (b) in $\mathcal{N}$.

First let us identify the N-indexed algebra operations and constants in the structure of Σ-graphs.

— For a graph $G : n$ and permutation $\rho : n \to n$, $G \cdot \rho$ is the graph obtained from G by relabeling its external vertices according to ρ.
— For graphs $G : n$ and $H : m$, $G \oplus H : n + m$ is the disjoint union of G and H with the labels of H's external vertices incremented by n.
— For every $n \in N$, the graph $\mathbf{1}_n : n + n$ consists of n edges connecting external vertex i with $n + i$ for each $i \in [n]$.
— For a graph $G : n + n + m$, $\updownarrow_n G$ is constructed from G by gluing together the pairs of edges ending in external vertices i and $n + i$ for each $i \in [n]$, discarding the vertices i and $n + i$ themselves. As part of this procedure, whenever a number of external edges are glued together in a cycle with no intercepting vertices, a new copy of the graph $\perp$ is added to the graph. Finally, the label of each remaining external vertex is decremented by $n+n$. See Fig. 4 for the implementation of $\updownarrow_3$ on a graph $G : 3 + 3 + 1$.

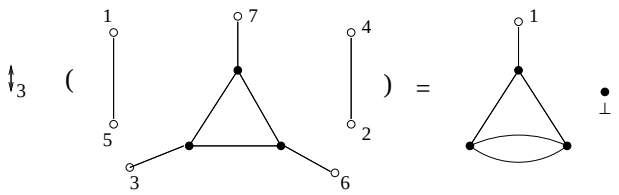

Figure 4: Taking the trace of a graph.

The algebra of Σ-graphs defined in this way is denoted by $\mathcal{G}(\Sigma)$. Now we turn to the formal definition of N-indexed algebras in terms of equational axioms.

Definition 2. An *N-indexed algebra* (NIA, for short) is an N-sorted algebra

$$\mathcal{N} = \{\mathcal{N}_n \,|\, n \in N\}$$

equipped with the operations and constants listed above, which satisfies the following equational axioms.

I1. *Functoriality of indexing*

$f \cdot (\rho_1 \bullet \rho_2) = (f \cdot \rho_1) \cdot \rho_2$ for $f : n$, $\rho_1 : n \to m$, and $\rho_2 : m \to k$;

$f \cdot 1_n = f$ for $f : n$.

I2. *Naturality of indexing*

$(f \oplus g) \cdot (\rho_1 \oslash \rho_2) = f \cdot \rho_1 \oplus g \cdot \rho_2$ for $f : n$, $g : m$, $\rho_1 : n \to k$, $\rho_2 : m \to l$;

$(\updownarrow_n f) \cdot \rho = \updownarrow_n (f \cdot (1_{n+n} \oslash \rho))$ for $f : n + n + m$, $\rho : m \to k$.

I3. *Associativity and symmetry of sum*

$(f \oplus g) \oplus h = f \oplus (g \oplus h)$ for $f : n$, $g : m$, $h : k$;

$f \oplus g = (g \oplus f) \cdot c_{n,m}$ for $f : n$, $g : m$.

I4. *Right identity*

$f \circ \mathbf{1}_n = f$ and $f \oplus \mathbf{1}_0 = f$ for $f : n \to m$.

I5. *Symmetry of identity*

$\mathbf{1}_n \cdot c_{n,n} = \mathbf{1}_n$.

I6. *Vanishing*

$\updownarrow_0 f = f$ for $f : n$;

$\updownarrow_{n+m} f = \updownarrow_m (\updownarrow_n f \cdot (1_n \oslash c_{m,n} \oslash 1_{m+k}))$ for $f : (n + m) + (n + m) + k$.

I7. *Superposing*

$\updownarrow_n (f \oplus g) = \updownarrow_n f \oplus g$ for $f : n + n + m$, $g : k$.

I8. *Trace swapping*

$\updownarrow_m (\updownarrow_n f) = \updownarrow_n (\updownarrow_m (f \cdot (c_{n+n,m+m} \oslash 1_k)))$ for $f : n + n + m + m + k$.

The reader can now easily verify that the structure $\mathcal{G}(\Sigma)$ of Σ-graphs is an NIA. The notation $\mathcal{G}$ is used for the graph algebra in which $\Sigma = \Sigma^u$ (i.e., the uniform alphabet), and, furthermore, the equations $K_{1,n} \cdot \rho = K_{1,n}$ hold for every $n \in N$ and permutation $\rho : n \to n$. ($K_{1,n}$ stands for the n-star with its internal vertex labeled by σ_n.) Under these assumptions there is no need to actually specify the label of a vertex (unless it is $\perp$, of course) or the ordering of the ports within that vertex. Thus, graphs in $\mathcal{G}$ are ordinary undirected graphs. Also, we shall impose the identity $\updownarrow_1 \mathbf{1}_1 = \mathbf{1}_0$ in $\mathcal{G}$, so that the graph $\perp$ virtually disappears in this algebra.

Let $\mathcal{N}$ and $\mathcal{N}'$ be N-indexed algebras. An *N-indexed homomorphism* $h : \mathcal{N} \to \mathcal{N}'$ is a family of mappings $\{h_n : \mathcal{N}_n \to \mathcal{N}'_n \mid n \in N\}$ that determine a homomorphism in the usual algebraic sense. The following result was proved in a more general setting in [4]. For the category theory background the reader is referred to [18, 20].

Theorem 1. *Every N-indexed algebra is equivalent to a self-dual compact closed category over the monoid N as objects. This connection defines an equivalence between the category of N-indexed algebras with N-indexed homomorphisms as morphisms, and the category of locally small self-dual compact closed categories over the fixed monoid N as objects, with strict monoidal functors preserving the given self-adjunctions as morphisms. For a ranked alphabet Σ, the algebra $\mathcal{G}(\Sigma)$ is freely generated by Σ.*

Next we identify the NIA operations in the structure of Turing automata. In this algebra $\mathcal{T}$, the morphisms of $\mathcal{T}_n$ are Turing automata $T : n$.

— A permutation $\rho : n \to n$ is interpreted as the relabeling of the interfaces according to ρ. (Elaboration of details regarding the transition relation is left to the reader.)

— The sum of $T : n$ and $T' : m$ having states Q and Q', respectively, is the automaton $T \oplus T' = ([n+m], Q \times Q', \delta \oplus \delta')$, where $\delta \oplus \delta' \subseteq ((Q \times Q') \times \langle n+m \rangle)^2$ is defined by

$$((q, q'), i), ((r, r'), j)) \in \delta \oplus \delta'$$

iff either $q' = r'$ and $((q, i), (r, j)) \in \delta$, or $q = r$ and $((q', i-n), (r', j-n)) \in \delta'$, with the understanding that the expression $i - n$ $(j - n)$, when not positive, is meaningful only if $i = 0$ (respectively, $j = 0$), in which case it stands for 0. In other words, taking the sum of T and T' amounts to the *selective performance* of δ or δ' on $Q \times Q'$. As part of this definition, transitions of $T \oplus T'$ involving the anchor 0 are the "union" of such transitions in T and T', that is, either one by T or one by T', nondeterministically on the product state space, always leaving the other component unchanged.

— The identity Turing automaton $\mathbf{1}_n : n + n$ has a single state, in which there is a transition from i to $n + i$ and back for every $i \in [n]$. There are no transitions to or from the anchor.

— The definition of $\updownarrow_n T$ for a TA $T : n + n + m$ is complicated but natural, and it holds the key to understanding the Turing-machine-like behavior of this automaton. Intuitively, the definition models the behavior of loops in flowchart algorithms [15], when implemented in an undirected environment. Control enters $\updownarrow_n T$ (viewed essentially as T) at an interface $j \in \langle m \rangle$ (that is, $n + n < j \leq n + n + m$ or $j = 0$ according to T), then, after alternating between corresponding interfaces of T belonging to $[n + n]$ any number of times, it leaves at another (or the same) interface $j' \in \langle m \rangle$. State changes are traced interactively during this process.

Formally, let $u = p_1 \dots p_l \in \Theta^*_{[n+n+m]}$ be a non-empty input string for T, where $p_i = (x_i, y_i)$, $i \in [l]$, $l \geq 1$. The string u is called *n-alternating* from x_1 to y_l ($u : x_1 \to y_l$, for short) if for every $i \in [l-1]$ there exist $n_i \in [n]$ and $j \in \{0, 1\}$ such that

$$y_i = jn + n_i \text{ and } x_{i+1} = (1 - j)n + n_i.$$

The transition function $\hat{\delta}$ of $\updownarrow_A T$ is defined for every state $q \in Q$ and input $p = (m_1, m_2)$ by

$$\hat{\delta}(q, p) = \cup(\delta(q, u) | u : m_1 \to m_2 \text{ is } n\text{-alternating}).$$

It is easy to see that all of the NIA operations are compatible with isomorphism of automata. Therefore the algebra $\mathcal{T}$ of isomorphism classes of TA is well-defined. The reader familiar with iteration theories [9] will notice that the above trace operation $\updownarrow$ is the undirected counterpart of iteration in Conway matrix theories. See also the example traced monoidal category $(\mathbf{Rel}, +)$ in Section 6 of [17], which originates from [9], too. The connection between these two operations has been made explicit in [4, 2] by characterizing Turing automata as matrices over the semiring of their state transformations. The following theorem is quoted again from [4].

Theorem 2. *The algebra $\mathcal{T}$ of Turing automata is indexed by the monoid N.*

Now we relate the algebra of graphs $\mathcal{G}(\Sigma)$ (the syntax) to that of Turing automata (the semantics). An *interpretation* of Σ in an arbitrary NIA $\mathcal{N}$ is a rank-preserving mapping Ω from Σ into $\mathcal{N}$. By Theorem 1, Ω can be extended in a unique way to a homomorphism $\Omega : \mathcal{G}(\Sigma) \to \mathcal{N}$.

Definition 3. A *Turing graph machine* over the ranked alphabet Σ is a couple $M = (G, \Omega)$, where G is a Σ-graph and Ω is an interpretation of Σ in $\mathcal{T}$. As an automaton, M stands for (the isomorphism class determined by) the TA $\Omega(G)$.

Definition 3 puts the intuitive concept of Turing graph machines described at the beginning of Section 2 in a precise algebraic form. Indeed, the transitions of the automaton $\Omega(G)$ in each state are recaptured by making appropriate walks in G. Recall from [19] that a *walk* $w = v_0, e_1, \ldots, v_n$ is an alternating sequence of edges and vertices, which starts and ends with a vertex, and in which each edge is incident with the vertex immediately preceding it and the one immediately following it. Regarding the walks corresponding to transitions of $\Omega(G)$, the first/last vertex of such a walk w is external (internal) iff the transition starts/ends at a real interface (respectively, the anchor). Only the two endpoints of w can be external. An empty walk $w = v_0$ corresponds to an anchor-to-anchor transition of the automaton at vertex v_0. Notice that such transitions do not exist in the atomic switches.

4. Soliton automata

Starting from the interpretation determined by the atomic switches, we are going to arrive at the model of soliton automata as Turing graph machines after a number of steps. As it will turn out from this discussion, the algebra of soliton automata does not follow the structure of Turing automata literally, rather, it is the quotient of an appropriate subalgebra of $\mathcal{T}$. Thus, the structure of soliton automata is an NIA, and as such it is a homomorphic image of the algebra $\mathcal{G}$ of ordinary undirected graphs.

Definition 4. A *pre-soliton automaton* is a Turing graph machine over the uniform alphabet Σ^u in which the interpretation of σ_n is fixed as the n-ary atomic switch.

Remember that the nullary atomic switch is the automaton $\mathcal{A}_0$. Since the atomic switches are circularly symmetric, the given interpretation Ω can also be extended in a unique way to a homomorphism from $\mathcal{G}$ to $\mathcal{T}$. Notice that, for example, the automaton determined by a single internal vertex having a loop around it still has two states, and not just one. Transitions between these two states can be triggered by the input $(0, 0)$ (anchor-to-anchor, that is).

Let $Pre(G)$ denote the pre-soliton automaton determined by graph G in this way. For technical reasons it will be more appropriate for us to work with walks in the underlying graphs of pre-soliton automata as sequences $w = e_1, v_1, \ldots, v_{n-1}, e_n$, that is, sequences that start and end with an *edge*, rather than a vertex. Indeed, the atomic switches do not have anchor-to-anchor transitions, so that the empty walks are ruled out. We can have a self-transition in each state, though, determined by a single internal edge as a walk. In the new sense, the *length* of w is the number of *vertices* occurring in w, and w is *closed* if $e_1 = e_n$. An *empty walk* is a closed one with length 0. A *path* is a walk with no repeating edges or vertices. A *cycle* is a path closed up by the unreferenced endpoint of the initial edge e_1 and the edge e_1 itself. We accept $w = e_1$ as a degenerate "empty cycle".

Walk w is *external* if e_1 or e_n is such, otherwise w is *internal*. An *alternating walk* in G with respect to some state q of $Pre(G)$ is a walk that determines a transition of $Pre(G)$ on some input in state q. A *soliton walk* is a two-way external alternating walk, which then corresponds to a transition of $Pre(G)$ from one real interface to another. An *alternating cycle* is an *even-length* cycle that is also an alternating walk. An *alternating unit* is either a soliton path or an alternating cycle.

Given a state q and an alternating walk w with respect to q, *making* w in q refers to the process of flipping the states of the atomic switches corresponding to the vertices (star graphs) along w in an interactive way. The state q' obtained at the end of this process is denoted by $S(q, w)$.

Observe that pre-soliton automata are naturally *reversible* as computational devices. Indeed, if w is an alternating walk with respect to state q, then w^R (the reverse of w) is alternating with respect to $S(q, w)$ and $S(S(q, w), w^R) = q$. By definition, pre-soliton automata form a sub-NIA of $\mathcal{T}$, which subalgebra is a homomorphic image of $\mathcal{G}$. Actually, the homomorphism Ω in this case turns out

to be injective, so that the algebra of pre-soliton automata is isomorphic to $\mathcal{G}$. In other words, $Pre(G) \cong Pre(G')$ iff $G \cong G'$. (Remember that the empty graph is identified with the sink graph in $\mathcal{G}$.) We are not concerned with this algebra directly, however. It is only the structure of transitions in $Pre(G)$, i.e., making alternating walks in G, which is characteristic of soliton automata.

Let q be a state of some pre-soliton automaton $Pre(G)$. An internal edge $e = (u, v) \in E(G)$ is called *consistently positive* (*negative*) with respect to q if e is positive (respectively, negative) as an edge of both star graphs centered at u and v with respect to the projection of q to these star graphs. (Remember that q is the tuple of the local states of all the star graphs G is composed of.) Edge e is *inconsistent* otherwise. Notice that we cannot call the star graphs in hand subgraphs, since G might have loops. An external edge is of course always consistent in this sense (either positive or negative), since it identifies a unique edge of a single star graph in G. Consistency is not an issue for interface edges. According to this definition, a looping edge can only be negative if consistent. From now on, by a positive (negative) edge with respect to q we shall always mean a consistent one. The sign of a consistent edge $e \in E(G)$ with respect to q will be denoted by $sign(q, e)$, and $cons(q)$ will stand for the number of consistent edges with respect to q.

Let M_q denote the set of positive edges with respect to state q. Clearly, M_q defines a matching in G. Recall from [19] that a *matching* is a subset $M \subseteq E(G)$ such that no vertex of G occurs more than once as an endpoint of some edge in M. It is understood, as part of this definition, that loops cannot be present in M. We say that q is *maximum consistent* if M_q is a matching that covers a maximum number of internal vertices. Such a matching is called *maximum internal* in G. If a matching M covers all of the internal vertices, then it is a *perfect internal matching*. See again [19, 3]. A *perfect matching* is just a perfect internal matching of a closed graph.

Now let $w = e_1, v_1, \ldots, e_n$ be a walk in G which determines a transition in $Pre(G)$ starting from the anchor in state q. By definition, the consistency of e_1 changes only if $n > 1$. Then the consistency of e_2 changes once if $n = 2$ and $e_1 \neq e_2$, and twice if $n > 2$ or $e_1 = e_2$. Following up on this idea, we obtain the three simple statements below.

Lemma 1. *For an arbitrary graph G, let $q \in Q$ be a state of $Pre(G)$ and $w = e_1, v_1, \ldots, e_n$ be an alternating walk in G with respect to q. Then, for the state $q' = S(q, w)$:*

(a) If w is a soliton walk, or w is internal and closed, then $cons(q) = cons(q')$.

(b) If w is external, but not two-way external, then $|cons(q) - cons(q')| = 1$.

(c) If w is internal and open, then $|cons(q) - cons(q')| = 2$ whenever the consistency of e_1 in q is the same as that of e_n, and $cons(q) = cons(q')$ if this is not the case.

Proof. Immediate by the definitions. □

By Lemma 1, a soliton walk, as well as an anchor-to-anchor transition determined by an even-length closed walk, will always take a perfect internal matching of G into a perfect internal matching. Moreover, perfect internal matchings of G are invariant only by such transitions. This is no longer true for just maximum consistent states, because a transition between such two states can also be induced by an open internal alternating walk w of type (c) above. See the Gallai-Edmonds Structure Theorem below.

For every graph G, let $Max(G)$ denote the sub-automaton of $Pre(G)$ determined by its maximum consistent states. Observe that the connection $q \mapsto M_q$ between states and maximum internal matchings is not one-to-one, therefore states cannot be identified with maximum internal matchings. It is easy to see, however, that the *internal deficiency* of G [19, 3], that is, the number of internal vertices not covered by a maximum internal matching, equals the number of inconsistent edges in any state

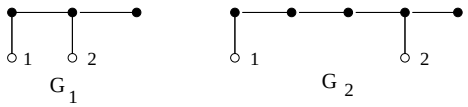

Figure 5: The two graphs of Example 3.

of $Max(G)$. See also the Gallai-Edmonds Structure Theorem below for an explanation. For any two graphs $G, G' : n$, we say that the automata $Max(G)$ and $Max(G')$ are *faithfully isomorphic*, in notation $Max(G) \cong_f Max(G')$, if they are such by an isomorphism χ that preserves the sign of the corresponding external edges. To spell this out, any external vertex i is connected to an internal one in G_1 by some edge e_1 iff it is connected to one in G_2 by edge e_2, and in that case $sign(q, e_1) = sign(\chi(q), e_2)$ for every state q of $Max(G_1)$. Notice that faithful isomorphism is a stronger requirement than ordinary isomorphism, and it is meaningful only if we specify the underlying graphs G_1 and G_2. (Remember that interface edges do not have a sign assigned to them.) Faithful isomorphism cannot be defined for Turing automata in general. Also notice that the faithful distinction is irrelevant in sort 0.

Example 3. Consider the simple graphs G_1 and G_2 of sort 2 in Fig. 5. Clearly, G_1 and G_2 both have a unique perfect internal matching, and $Max(G_1) \cong_f Max(G_2)$. On the other hand, $\updownarrow_1 G_1$ has 3 maximum internal matchings (out of 4 states), while $\updownarrow_1 G_2$ has 4 (out of 6 states). Thus, the automata $Max(\updownarrow_1 G_1)$ and $Max(\updownarrow_1 G_2)$ are not isomorphic.

Example 3 shows that it is impossible to define the NIA of "maximum" pre-soliton automata in a consistent way. Therefore we have to make further changes to our model to obtain the desired algebra of soliton automata. This time the change will be of a structural nature, captured in abstract terms by taking the quotient of automata. Before introducing the change, however, we need to learn more on the structure of maximum matchings in graphs.

4.1. The Gallai-Edmonds decomposition of graphs

One of the earliest fundamental results in matching theory is the so called Gallai-Edmonds Structure Theorem [16, 14]. The main idea of this theorem is to decompose an arbitrary (closed) graph G into three sets of vertices as follows.

- $D(G)$: vertices not covered by at least one maximum matching of G;
- $A(G)$: vertices in $V(G) - D(G)$ adjacent to at least one vertex in $D(G)$;
- $C(G) = V(G) - A(G) - D(G)$.

The five statements of the theorem are listed below. To explain statements (a) and (d), a closed graph G is called *factor-critical* if $G - v$ has a perfect matching for every $v \in V(G)$. In this case, a *near-perfect matching* of G is one that covers all vertices but one. Clearly, every factor-critical graph is connected and has an odd number of vertices. For statement (c), let G be a bipartite graph with bipartition (V_1, V_2). The *surplus* of G viewed from V_1 is the number

$$min\{(|\Gamma(X)| - |X|) \,|\, X \subseteq V_1, X \neq \emptyset\},$$

where $\Gamma(X)$ denotes the set of all vertices in V_2 adjacent to at least one vertex in X. Regarding (e), the *deficiency* of G, denoted $def(G)$, is the number of vertices left uncovered by an arbitrary maximum matching of G.

Theorem 3. (The Gallai-Edmonds Structure Theorem)

(a) The components of the subgraph induced by $D(G)$ are factor-critical.

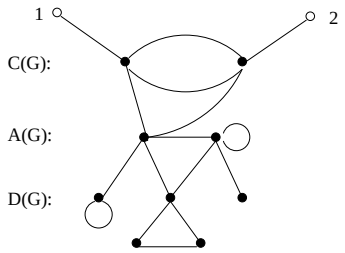

Figure 6: The Gallai-Edmonds decomposition of a graph.

(b) The subgraph induced by $C(G)$ has a perfect matching.

(c) The bipartite graph obtained from G by deleting the vertices of $C(G)$ and the edges spanned by $A(G)$ and by contracting each component of $D(G)$ to a single vertex has positive surplus (as viewed from $A(G)$).

(d) If M is any maximum matching of G, it contains a near-perfect matching of each component of (the graph induced by) $D(G)$, a perfect matching of $C(G)$, and matches all vertices of $A(G)$ with vertices in distinct components of $D(G)$.

(e) $def(G) = c(D(G)) - |A(G)|$, where $c(D(G))$ denotes the number of connected components in $D(G)$.

See Fig. 6 for a representative example. The counterpart of the Gallai-Edmonds Structure Theorem for maximum internal matchings was proved in [7]. As it turned out, the difference between the statement of the original theorem and that of its counterpart is of a rewording nature, and can be summarized as follows:

— the set $D(G)$, as well as $A(G)$, consists of internal vertices only;

— the subgraph induced by $C(G)$, which contains all of the external vertices, has a perfect internal matching;

— in general, the words "perfect matching" and "maximum matching" are replaced by "perfect internal matching" and "maximum internal matching", respectively;

— in statement (e), $def(G)$ is replaced by the internal deficiency of G.

In the light of Theorem 3 the reader may want to have another look at the graphs of Example 3. To capture the essence of the Gallai-Edmonds Theorem in algebraic terms, let $G : n$ be an arbitrary graph (morphism in $\mathcal{G}$). Construct $A(G) : n_C \to n_D$ to be the sum of vertices in $A(G)$ as star graphs (that is, with all the incident edges as external ones). Assign the labels to the external vertices in this sum in such a way that the first n_C and the last n_D of them point towards $C(G)$ and $D(G)$, respectively. Correspondingly, write $C(G)$ in the form $C(G) : n \to n_C$ and $D(G) : n_D(\to 0)$. See again Fig. 6. Then the Gallai-Edmonds decomposition of G is simply

$$C(G) \circ A(G) \circ D(G)$$

in the NIA (self-dual compact closed category) $\mathcal{G}$. Observe that $C(G)$, as a morphism $n \to n_C$, might pick up a few instances of the identity graph $1_1 : 0 \to 2$ which are not originally present in G. Indeed, edges connecting vertices in $A(G)$ must be closed up by such extra lines using "canonical trace", cf. [17, 4]. Formally we put these lines in $C(G)$, because they have a perfect internal matching by definition.

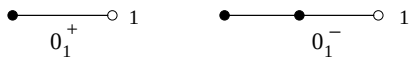

Figure 7: The erase graphs.

4.2. The algebra of soliton graphs and automata

Now we turn to the definition of soliton graphs and soliton automata. If G is a graph having a perfect internal matching, then we shall use the notation $Sol(G)$ for $Max(G)$.

Definition 5. A *soliton graph* is a graph having a perfect internal matching. The *soliton automaton* determined by a soliton graph G is the automaton $Sol(G)$. The soliton automata determined by graphs G and G' are *isomorphic* if $Sol(G) \cong_f Sol(G')$.

Observe that it is now safe to identify the states of a soliton automaton $Sol(G)$ with the perfect internal matchings of G. By Lemma 1, the only alternating walks in G that determine a transition in $Sol(G)$ are the soliton walks and the closed even-length internal ones. Such walks will therefore be called *transition walks* in G. Note that an odd-length closed internal alternating walk with respect toto a perfect internal matching simply does not exist. In Fig. 7, consider the two soliton graphs 0_1^+ and 0_1^- as morphisms $1 \to 0$ in $\mathcal{G}$. We call them the positive and negative *erase* morphisms (graphs) for obvious reasons. Both have a unique perfect internal matching, but their soliton automata are not faithfully isomorphic, just isomorphic. For an arbitrary graph G, define

$$\bar{C}(G) = C(G) \circ \oplus_{i=1}^{n_C} 0_1^-.$$

Clearly, $\bar{C}(G)$ is a soliton graph, and it reflects the intention to "sew up" the interfaces of $C(G)$ pointing towards $A(G)$ by negatively erasing them. See again Fig. 6. Notice that we are not losing degrees anywhere in the design, which is crucial from the algebraic point of view.

The operations of the *Gallai-Edmonds algebra* $\mathcal{G}$-$\mathcal{E}$ of soliton graphs are the same as those of the NIA $\mathcal{G}$, except that trace is defined in $\mathcal{G}$-$\mathcal{E}$ by the formula

$$(\uparrow_n G)_{\mathcal{G}\text{-}\mathcal{E}} = \bar{C}((\uparrow_n G)_{\mathcal{G}}) \quad \text{for } G : n + n + m.$$

The following important theorem was proved in [8].

Theorem 4. *The mapping $G \mapsto \bar{C}(G)$ is a homomorphism, called the Gallai-Edmonds map. Consequently, the algebra $\mathcal{G}$-$\mathcal{E}$ is an NIA.*

Extend the definition $Sol(G)$ for all graphs G by the formula:

$$Sol(G) = Sol(\bar{C}(G)).$$

This will allow us to speak of the soliton automaton determined by an arbitrary graph G.

Example 4. Consider the graphs G_1 and G_2 in Fig. 8. Clearly, the internal deficiency of both graphs is 1, but $C(G_1) : 0 \to 0$ is empty, while $C(G_2) : 0 \to 2$ is the identity graph 1_1. (Mind the edge connecting the two vertices in $A(G_2)$.) Open up the loops in both graph as shown in Fig. 8 by dashed lines, to obtain the graphs G_1' and G_2' of sort 2. Observe that these two graphs already have a perfect internal matching (in fact two), and $G_i = \uparrow_1 (G_i')$, $i = 1, 2$. Moreover, $Sol(G_1') \cong_f Sol(G_2')$. We want the soliton automata $Sol(G_1)$ and $Sol(G_2)$ to remain isomorphic. Since these automata are associated with the soliton graphs $\bar{C}(G_1)$ and $\bar{C}(G_2)$, respectively, the automaton $\perp = Sol(\perp)$ must be (faithfully) isomorphic to $Sol(1_1 \circ (0_1^- \oplus 0_1^-))$. The latter automaton, being isomorphic to $\mathcal{A}_0$, has an anchor-to-anchor self-transition, while the former does not.

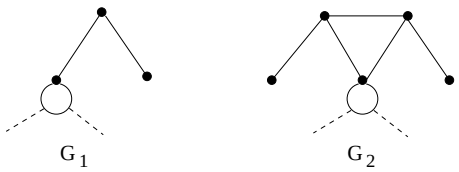

Figure 8: The graphs of Example 4.

For reasons explained in Example 4 above, the equation $\perp = \mathcal{A}_0$ will hold in the NIA of soliton automata, whether we like it or not. The immediate consequence of this equation is that all soliton automata (even the identities) will virtually have an anchor-to-anchor self-transition in each state. (Mind the right identity axiom I4, observing that $\perp = \mathbf{1}_0$ holds already in $\mathcal{G}$.) Thus, our effort to exclude such transitions in certain Turing automata was practically in vain for soliton automata. It was not, however, regarding the whole algebra $\mathcal{T}$ of general Turing automata, which is of course a lot richer than the very simple subalgebra of pre-soliton automata, a quotient of which is the NIA of soliton automata. The proof of the following lemma provides an excellent illustration of the phenomenon "soliton valving" [12, 11].

Lemma 2. *For an arbitrary graph G, let q be a maximum consistent state of $Pre(G)$. Let, furthermore, $w = e_1, v_1, \ldots, e_n$ be an alternating walk with respect to q that starts and ends in an edge belonging to $C(G)$. Then there exists an alternating walk w' in $\bar{C}(G)$ with respect to the restriction q' of q to $C(G)$, connecting e_1 with e_n in such a way that w' covers the exact same vertices and edges from $C(G)$ as w, in the same order and with the same multiplicity. Thus, $S(q', w')$ is the restriction of $S(q, w)$ to $C(G)$.*

Proof. Indeed, whenever the walk w leaves $C(G)$ on an edge e leading to a vertex u in $A(G)$, the sign of e is negative and w will continue on a positive edge e' pointing to a vertex v in $D(G)$. After that, w must remain in $D(G)$ until it comes back to v, only to make the edges e' and e in reverse, and then continue in $C(G)$ as if it has never left this subgraph. The reason is that, every time w hits a vertex $u' \in A(G)$ different from u, it will arrive on a negative edge and the "valve" (i.e., the positive edge) at u' will force w to continue towards $D(G)$. Applying this argument inductively, w must return to u. See Theorem 3(d). It also follows from Theorem 3(a,d) that w will run exclusively on consistent edges. The augmentation of $C(G)$ to $\bar{C}(G)$ is necessary, because w, when coming back to $C(G)$ from u, may start a backtrack on the edges made before. If we did not include and "sew up" the vertex u, this backtrack would not be feasible without the presence of $A(G)$ and $D(G)$. □

Lemma 2 shows that $Sol(G)$ is the quotient automaton of $Max(G)$ by the equivalence that puts two states in the same class iff their restriction to the subgraph $C(G)$ coincides. It is also immediate by Lemma 2 that an arbitrary non-empty alternating cycle with respect to any state q of $Max(G)$ will run exclusively in $C(G)$ or in $D(G)$.

Lemma 3. *Let $G : 1 + 1 + m$ be a soliton graph, and assume that $G' = \updownarrow_1 G$ does not have a perfect internal matching. Then for every edge e in $D(G')$, different from the newly created one when closing up the external edges e_1 and e_2 in G, and for every state (perfect internal matching) q of G in which e is positive, there exists an alternating path w from e_i to e in G with respect to q for either $i = 1$ or $i = 2$ (or both). The path w runs entirely in $D(G) \cup A(G)$.*

Proof. This is an immediate consequence of Theorem 3. The lack of such a path in state q would imply that the set of vertices $A(e, q)$ in $A(G')$ accessible by alternating paths originating from e with

respect to q has a zero surplus in the Gallai Edmonds decomposition of G', contradicting statement (c) of that theorem. Indeed, the internal deficiency of G' is 1, and the inconsistent edge in G' (with respect to q) created when joining e_1 with e_2 already identifies the factor-critical component D_0 in $D(G')$ which is not helped out by a positive edge from $A(G')$. Clearly, D_0 is not the component in which the edge e resides, because in that case a suitable alternating path would exist. (Compare with the idea of finding augmenting paths in the Edmonds matching algorithm [13, 19].) The second statement of the lemma follows directly from Lemma 2. □

Corollary 1. *Under the conditions of Lemma 3, if w is a non-empty alternating cycle in G with respect to q running entirely in $D(G')$, then the transition determined by w can also be made as a closed soliton walk in G starting from either e_1 or e_2.*

Proof. One can start e.g. from e_1, go down to a positive edge in w, make w, and backtrack to e_1. □

We shall also need the following result, quoted from [6]. Let G be a soliton graph and q be a perfect internal matching of G. An *alternating network* with respect to q is a set Γ of pairwise disjoint alternating units (that is, soliton paths or alternating cycles). Then the perfect internal matching $S(q, \Gamma)$ is obtained from q by making the units in Γ one-by-one in an arbitrary order.

Proposition 1. *([6, Theorem 3.1]) For any two perfect internal matching's q_1 and q_2 of a soliton graph G there exists an alternating network Γ such that $q_2 = S(q_1, \Gamma)$.*

Let w be a transition walk in a soliton graph G with respect to state q. Then, for an arbitrary subgraph G' of G, we say that w has a *visible trace* on G' if there exists at least one edge e in G' such that $sign(q, e) \neq sign(S(q, w), e)$.

Corollary 2. *Let G and H be soliton graphs of sort $1 + 1 + n$ with G as in Lemma 3, and assume that $Sol(G) \cong_f Sol(H)$ by a faithful isomorphism χ. Furthermore, let w be a transition walk in G on input p with respect to some state q of $Sol(G)$ that runs entirely in $C(G')$ such that $r = S(q, w) \neq q$. Then any transition walk $\chi(w)$ in H that induces the transition $\chi(q) \mapsto \chi(r)$ on p has a visible trace on $C(\updownarrow_1 H)$.*

Proof. Denote $H' = \updownarrow_1 H$, and let f_1, f_2 be the first two external edges in H. Since χ is faithful, H' does not have a perfect internal matching either. By Lemma 2 and Proposition 1, if $\chi(w)$ did not have a visible trace on $C(H')$, then making $\chi(w)$ would be equivalent to making a number of alternating cycles in $D(H')$. This is impossible, however, since Corollary 1 says that any of these cycles can be made as a closed soliton walk from either f_1 or f_2, while the transition from q to $r \neq q$ in $Sol(G)$ is definitely not available as a sequence of transitions on inputs (1,1) and (2,2) only.

Notice that the above argument remains valid if the transition $q \mapsto r$ is induced by an alternating network Γ, rather than just a transition walk w. In that case one must consider the cumulative effect of making the walks in H corresponding to the units in Γ. Clearly, each of these walks will be a transition walk in H, and if their cumulative trace on $C(H')$ is empty, then this leads to the same contradiction as above. □

Now we are ready to define the algebra $\mathcal{S}$ of soliton automata. The morphisms in $\mathcal{S}_n$ are isomorphism classes of soliton automata determined by soliton graphs $G : n$. The interpretation of relabeling, sum, and that of the constants $\mathbf{1}_n$ in $\mathcal{S}$ is straightforward: downgrade the soliton automata as operands to pre-soliton ones, perform the operation in $\mathcal{T}$, and upgrade the result as a soliton automaton again.

Lemma 1 ensures that this argument is correct. Compatibility of faithful isomorphism with these operations is trivial. The only challenging operation is trace. For a soliton graph $G : n + n + m$, let

$$\updownarrow_n Sol(G) = Sol(\updownarrow_n G). \tag{1}$$

In order to make this definition applicable to isomorphism classes of soliton automata, we need the following theorem.

Theorem 5. *Let G and H be soliton graphs of sort $n + n + m$ such that $Sol(G) \cong_f Sol(H)$. Then $Sol(\updownarrow_n G) \cong_f Sol(\updownarrow_n H)$.*

Proof. By an induction argument we can assume, without loss of generality, that $n = 1$. (Remember the vanishing axiom I6, which holds in $\mathcal{G}$-$\mathcal{E}$.) If either of the external vertices 1 and 2 is the endpoint of an interface edge in G, then it is one in H, too, and the statement is trivial. Otherwise, as in Corollary 2, let G' and H' denote the graphs $\updownarrow_1 G$ and $\updownarrow_1 H$, and let χ be a faithful isomorphism between $Sol(G)$ and $Sol(H)$. If G' still has a perfect internal matching, then so does H', for χ is faithful. In this case the states of $Sol(G')$ and $Sol(H')$ are identified by those perfect internal matchings of G and H in which the sign of e_1 and e_2 (respectively, f_1 and f_2) is equal. Remember that e_1 and e_2 (respectively, f_1 and f_2) are the external edges to be joined in G and H. Thus, the restriction of χ to these states defines a faithful one-to-one correspondence ρ between the states of $Sol(G')$ and $Sol(H')$. Moreover, since χ is an isomorphism of automata, ρ will be one, too. (Use the downgrade-upgrade argument again with regard to transitions, implementing them in the corresponding pre-soliton automata.)

Assume therefore that G' and H' no longer have a perfect internal matching. Take a perfect internal matching of $C(G')$ as a state q' of $Sol(G')$, and extend it to a perfect internal matching q of G according to Theorem 3(d) in an arbitrary way, so that $sign(q, e_1) \neq sign(q, e_2)$. Define the state $\rho(q')$ of $Sol(H')$ to be the restriction of $\chi(q)$ to $C(H')$. By Proposition 1 and Corollary 1, $\rho(q')$ does not depend on how the extension from q' to q was chosen. For the same reason, in the light of Corollary 2, the mapping ρ is injective and onto the states of $Sol(H')$. Clearly, ρ is faithful as well.

Now let w' be a transition walk in $\bar{C}(G')$ from state q' to $r' = S(q', w')$ on input p. Then, by Lemma 2 and Corollary 2, there exists a transition walk in $\bar{C}(H')$ from $\rho(q')$ to $\rho(r')$ on p, which can be constructed as the restriction of the walk in H that corresponds to w', when considered as a transition walk in G. □

Corollary 3. *The algebra $\mathcal{S}$ is a homomorphic image of the algebra $\mathcal{G}$-$\mathcal{E}$, therefore it is indexed by N.*

Proof. Immediate by equation (1) and Theorem 5. □

5. Conclusions

We have given a characterization of soliton automata as primitive Turing graph machines, in which the behavior of each cell is a simple atomic switch capturing the transposition of the unique double bond connecting a carbon atom in a hydrocarbon molecule to one of its neighbors. First we defined the model of pre-soliton automata as a subalgebra of the N-indexed algebra of all Turing automata. Then we identified soliton automata as appropriate quotients of certain maximal sub-automata of pre-soliton automata. The states of soliton automata turned out to be the perfect internal matchings of their underlying graph. We have discussed the Gallai-Edmonds decomposition of graphs in an algebraic setting, and showed that the Gallai-Edmonds algebra of graphs having a perfect internal matching is a homomorphic image of the free N-indexed algebra of general graphs. Finally we proved that the algebra of soliton automata, being a homomorphic image of the Gallai-Edmonds algebra, is also indexed by N.

References

[1] ABRAMSKY, S., COECKE, B., A categorical semantics of quantum protocols, in: Proc. 19th Annual Symposium on Logic in Computer Science, IEEE Computer Society Press, 2004, 415–425.

[2] BARTHA, M., Quantum Turing automata (`http://www.cs.mun.ca/~bartha/linked/quant.pdf`).

[3] BARTHA, M., KRÉSZ, M., Soliton circuits and network-based automata: review and perspectives, in: Mathematics, Computing, Language, and Life: Frontiers in Mathematical Linguistics and Language Theory, C. Martín-Vide (Ed), Vol. 2, Imperial College Press, 2010, 585–631.

[4] BARTHA, M., Turing automata and graph machines, Electronic Proceedings in Theoretical Computer Science 26, (2010) 19–31 (`http://www.cs.mun.ca/~bartha/linked/mono.pdf`).

[5] BARTHA, M., KRÉSZ, M., Structuring the elementary components of graphs having a perfect internal matching, Theoretical Computer Science 299 (2003), 179–210.

[6] BARTHA, M., GOMBÁS, É., On graphs with perfect internal matchings, Acta Cybernetica 12 (1995), 111-124.

[7] BARTHA, M., GOMBÁS, É., A structure theorem for maximum internal matchings in graphs, Information Processing Letters 40 (1991), 289-294.

[8] BARTHA, M., GOMBÁS, É., The Gallai-Edmonds algebra of graphs, Technical Report No. 9105, Department of Computer Science, Memorial University of Newfoundland, St. John's, NL, Canada, 1991 (`http://www.cs.mun.ca/~bartha/linked/g-e.pdf`).

[9] BLOOM, S. L., ÉSIK, Z., Iteration Theories: The Equational Logic of Iterative Processes, Springer-Verlag, Berlin 1993.

[10] CARTER, F. L. et al., Soliton switching and its implications for molecular electronics, in: Molecular Electronic Devices II, Marcel Dekker Inc. 1987, 149–182.

[11] DASSOW, J., JÜRGENSEN, H., Soliton automata, J. Comput. System Sci. 40 (1990), 154-181.

[12] DAVYDOV, A. S., Solitons in Molecular Systems, Reidel, Dordrecht 1985.

[13] EDMONDS, J., Paths, trees and flowers, Canad. J. Math. 17 (1965), 449–467.

[14] EDMONDS, J., Maximum matching and a polyhedron with 0,1-vertices, J. Res. Nat. Bur. Standards Sect. B (1968), 125–130.

[15] ELGOT, C. C., (1975) Monadic computations and iterative algebraic theories, in: Proc. Logic Colloquium 1973, Studies in Logic and the Foundations of Mathematics 80 (1975) 175–230.

[16] GALLAI, T., Maximale Systeme unabhängiger Kanten, Magyar Tud. Akad. Mat. Kutató Int. Közl. 9 (1964), 401–413.

[17] JOYAL, A., STREET, R., VERITY, D., Traced monoidal categories, Math. Proc. Camb. Phil. Soc. 119 (1996), 447–468.

[18] KELLY, G. M., LAPLAZA, M. L., Coherence for compact closed categories, J. Pure Appl. Algebra 19 (1980), 193–213.

[19] LOVÁSZ, L., PLUMMER, M. D., Matching Theory, North Holland, Amsterdam, 1986.

[20] MAC LANE, S., Categories for the Working Mathematician, Springer-Verlag, Berlin, 1997.

[21] SELINGER, P., Towards a quantum programming language, Mathematical Structures in Computer Science 14 (2004), 527–586.