CONTRIBUTIONS TO GENERAL ALGEBRA 12
Proceedings of the Vienna Conference, June 3–6, 1999
Verlag Johannes Heyn, Klagenfurt 2000

AN ALGEBRAIC PERTURBATION THEORY FOR STATE AUTOMATA

TIM BOYKETT

ABSTRACT. Automata are investigated for which the effects of certain perturbations of the input can be reproduced by appropriately perturbing the output. This is shown to properly generalise a property of linear automata. The resulting automata have input and output alphabets that are transformation semigroups. An intuitive form is demonstrated to be very similar to the situation of linear automata. Perturbations which are permutations are investigated and shown to naturally lead to nearrings of group automata.

1. INTRODUCTION

As is often the case in nolinear automata theory, we are interested in looking at finite state automata that behave nicely. In our case, by nicely we mean that when we fiddle about with the input sequence, the output sequence changes in a consistent way, that we can look at only the variations in input and output without having to look at the inputs and outputs themselves.

The research is inspired in part by the way that this behaviour can be observed in linear automata. The primary motivation for this work came however from the desire to investigate systems where the input is somehow intrisic or included in the system, where we only have access to ways of perturbing or permuting the input stream, see [1]. Questions were arising that required a more solid theoretical base, so this work was begun.

The idea can be seen to derive from some kind of hope for a general algebraic theory of perturbations in systems. Given a type of mapping, i.e. perturbation in the input, we might like to know whether its effects die out or are amplified over time. By considering sequences of mappings where most of the mappings are identity, only one or a small cluster being nontrivial, we might want to know how long and in what way the output will deviate from its original unperturbed sequence.

In linear automata such behaviour can be seen to be the norm and to be a source of the ease of analysis of linear systems.

The structure of the paper is as follows. First we introduce the automata theoretic terminology and the definition of a mappable automaton. Linear automata are introduced as the standard example possessing the structure of interest.

After we have seen that our property of interest, mappability, exists and is non–trivial, we are interested in finding out what characterizes mappability in more general automata and for more general classes of mappings.

In the following section we look at an explicit form of mappability where the effect of the mappings upon the state are closely followed. This generalises the linear form, amd we see by examples that not all examples are of this type.

Over the paper we often use perturbations that are invertible, i.e. permutations. We close in on this as the most interesting case of examples and see that the study of mappable automata where we allow only permutations of the input. We find that this relates strongly to work on group automata originating from Pilz, Hofer and others [2, 3].

2. Definitions and Motivations

An *automaton* is a system $\Sigma = (S, A, \phi, \omega, \lambda_0)$ with S the *state set* A the *alphabet* $\phi : A \times S \to S$ the *state transition function*, $\omega : A \times S \to A$ the *output function* and $\lambda_0 \in S$ the *start state*.

Time is taken to be discrete. At each time step i, the automaton receives an input $a_i \in A$, moves to a new state $\lambda_{i+1} = \phi(a_i, \lambda_i)$ and emits an output $o_i = \omega(a_i, \lambda_i)$. We talk of the languages A^* and S^* of finite sequences of elements from A and S respectfully. We define a *length* operation $|\cdot|$ on S^* and A^* that returns the number of symbols in a sequence, i.e. $|a|$ is the length of the sequence $a \in A^*$. We extend the mappings ϕ and ω to these sets by $\phi(a_0 \dots a_n, \lambda) = \phi(a_1 \dots a_n, \phi(a_0, \lambda))$ and $\omega(a_0 \dots a_n, \lambda) = b_0 \dots b_n$ with $b_i = \omega(a_i, \phi(a_0 \dots a_{i-1}, \lambda))$. *Reachability* of a state $q \in S$ refers to the existence of a string $a \in A^*$ such that $\phi(a, \lambda_0) = q$. A *reachability class* is the set of states reachable by strings of a certain length. In general we assume that all states are reachable from the start state in order to avoid pathological systems.

Sometimes we allow the input and output alphabets to differ, in which case we write an automaton as $\Sigma = (S, A, B, \phi, \omega, \lambda_0)$, B the *output alphabet* and $\omega : A \times S \to B$ the *output function* with all other symbols as above.

We use the terminology $M(A)$ to refer to the set of mappings of a set A into itself. In general we can talk about an automaton as a mapping from A^* into itself defined by the local functions ϕ and ω. Thus automata define a class of mappings in $M(A^*)$. We refer to this mapping as $f_\Sigma : A^* \to A^*$ (or $A^* \to B^*$ for non–identical alphabets, from now on we will mention the seperate output alphabet only in the cases where it is particularly relevant or interesting) defined as $f_\Sigma(a) = \omega(a, \lambda_0)$. Note that these mappings are *prefix preserving* in that if $a, b \in A^*$ agree in some prefix, e.g. $a_i = b_i \ \forall i \leq k$, then $f_\Sigma(a)_i = f_\Sigma(b)_i$ for all $i \leq k$. Moreover, this is an equivalence, prefix–preserving maps on A^* are always representable by an automaton.

An automaton is *through mappable* or more concisely *mappable* if given a sequence of maps $f_i : A \to A$ there is a sequence of maps (possibly not uniquely defined) $\bar{f}_i : A \to A$ such that for all inputs $\sigma_0, \sigma_1, \ldots \in A$,

$$\lambda_0 = \lambda'_0 \tag{1}$$

$$\lambda_{i+1} = \phi(\sigma_i, \lambda_i) \qquad \lambda'_{i+1} = \phi(f_i\sigma_i, \lambda'_i) \tag{2}$$

$$\theta_i = \omega(\sigma_i, \lambda_i) \qquad \theta'_i = \omega(f_i\sigma_i, \lambda'_i) \tag{3}$$

then $\theta'_i = \bar{f}_i\theta_i$.

If there is a certain class $C \subseteq M(A)$ of functions from which the input mappings f_i can be selected so that the automaton Σ is mappable, we will say Σ is C–*mappable*. If C is implicit we will not necessarily state it.

As the mapping from f to $\bar{f}$ is prefix–preserving, we know that it is defined by an automaton with input alphabet C and output alphabet $M(A)$. This automaton will be called the *mapping automaton* of the mappable automaton.

In general we will also talk about sequences of functions $f \in M(A)^*$ acting on sequences of elements of A of the same length in that if $f \in M(A)^*$ and $a \in A^*$ with $|f| = |a|$, then $fa \in A^*$ is also the same length $|fa| = |a|$ and $(fa)_i = f_i(a_i)$. Thus we can rewrite the condition for C–mappability by requiring that for all $f \in C^* \exists \bar{f} \in M(A^*)$ s.t.

$$\omega(fa, \lambda_0) = \bar{f}(\omega(a, \lambda_0)) \tag{4}$$

for $f \in C^*$, $a \in A^*$ and $|f| = |a|$.

A trivial class of examples comes to light when we consider C a set of constant maps. The mapping automaton is the same as the original automaton, the output maps $\bar{f}$ are constant maps that produce the output forced by the transitions caused by the constant maps. In short, if we write the elements of C as $f^a : x \mapsto a \in A$ and $f_i = f^{b_i}$,

$$\lambda'_{i+1} = \phi(f_i(a), \lambda'_i) = \phi(b_i, \lambda'_i) \tag{5}$$

and $\bar{f}_i = f^{c_i}$, $c_i = \omega(b_i, \lambda'_i)$. Then the automaton is C–mappable.

A less trivial example can be found in linear automata over an (additively written) abelian group. That is, $(A, +)$ and $(S, +)$ are abelian groups and the mappings ϕ and ω are linear. If we consider the class of translation mappings, $C_A = \{f_a : x \mapsto x + a | a \in A\}$, we can see that

Lemma 1. *A linear automaton* $(S, A, \phi, \omega, \lambda_0)$ *is* C_A*–mappable.*

Proof: Let f be a sequence of mappings $f_k : x \mapsto x + \alpha_k$. We claim that Σ is mappable, with output maps of the mapping automaton $\bar{f}_k : x \mapsto x + \omega(\alpha_k, \phi(\alpha_0 \ldots \alpha_{k-1}, 0))$.

First we show that $\phi(fa, \lambda_0) = \phi(a, \lambda_0) + \phi(\alpha_0 \ldots \alpha_k, 0)$ with $k = |f| = |a|$. For $k = 1$,

$$\phi(fa, \lambda_0) = \phi(f_0a_0, \lambda_0) = \phi(a_0 + \alpha_0, \lambda_0 + 0) \tag{6}$$

$$= \phi(a_0, \lambda_0) + \phi(\alpha_0, 0) \tag{7}$$

Assuming true for $k \leq n$, let f and a be of length $n+1$ and

$$\begin{aligned}
\phi(fa, \lambda_0) &= \phi(f_n a_n, \phi(f_0 a_0 \dots f_{n-1} a_{n-1}, \lambda_0)) && (8)\\
&= \phi(a_n + \alpha_n, \phi(a_0 \dots a_{n-1}, \lambda_0) + \phi(\alpha_0 \dots \alpha_{n-1}, 0)) && (9)\\
&= \phi(a_n, \phi(a_0 \dots a_{n-1}, \lambda_0)) + \phi(\alpha_n, \phi(\alpha_0 \dots \alpha_{n-1}, 0)) && (10)\\
&= \phi(a, \lambda_0) + \phi(\alpha, 0) && (11)
\end{aligned}$$

Now notice that

$$\begin{aligned}
&\omega(f_n a_n, \phi((f_0 a_0) \dots (f_{n-1} a_{n-1}), \lambda_0)) && (12)\\
= \; &\omega(a_n + \alpha_n, \phi(a_0 \dots a_{n-1}, \lambda_0) + \phi(\alpha_0 \dots \alpha_{n-1}, 0)) && (13)\\
= \; &\omega(a_n, \phi(a_0 \dots a_{n-1}, \lambda_0)) + \omega(\alpha_n, \phi(\alpha_0 \dots \alpha_{n-1}, 0)) && (14)
\end{aligned}$$

thus the mapping $\bar{f}_n : x \mapsto x + \omega(\alpha_n, \phi(\alpha_0 \dots \alpha_{n-1}, 0))$ is the output map of the mapping automaton for the mappable automaton Σ. □

Note that the mapping automaton structure derives directly from the original automaton. It can be viewed in two equivalent ways.

In the first view, this automaton has the same state set, the start state is 0, the identity in $(S, +)$, and the input/output alphabet is the set C_A. The state transition function is $(f_a, s) \mapsto \phi(a, s)$, the output function $(f_a, s) \mapsto (x \mapsto x + \omega(a, s))$.

A second and more interesting view is to take the state set to be the set of mappings $x \mapsto x + s$ for $s \in S$, with corresponding state transition and output functions as above. The start state, rather than being the group element 0 is the (identity) mapping on S, $x \mapsto x + 0$. In general we interpret the state s as the mapping $x \mapsto x + s$ on S. The second formulation is of interest as we will see a generalisation of this construct later in Section 5.

Now that we see that mappability exists and is non–trivial, we are interested in finding out what characterizes mappability in more general automata and for more general classes C of mappings.

3. State–based mappings

It is of interest to develop a formulation in terms of the state–based mappings $\omega_\lambda : a \mapsto \omega(a, \lambda)$. We hope that in certain cases necessary and sufficient conditions for mappability may become readable from the properties of the state maps combined with some simple analysis of the reachability properties of the state transition graph.

First we define two relations. For each state $\lambda \in S$, define the relation $\sim_\lambda$ on A by $a \sim_\lambda b \Leftrightarrow \omega(a, \lambda) = \omega(b, \lambda)$. It is clearly an equivalence relation. For a given state λ, $\sim_\lambda$ is the equivalence relation induced by the mapping ω_λ on A.

We also define a relation $\rightsquigarrow_C$ on S by

$$\begin{aligned}
\lambda_1 \rightsquigarrow_C \lambda_2 \Leftrightarrow \exists a \in A^*,\ f \in C^* \ |a| = |f| \ s.t. \\
\lambda_1 = \phi(a, \lambda_0),\ \lambda_2 = \phi(fa, \lambda_0)
\end{aligned} \quad (15)$$

In general this relation is neither symmetric nor transitive.

The following proposition makes the determination of mappability easier.

Proposition 2. *If an automaton Σ is C–mappable then*

$$\lambda_1 \rightsquigarrow_C \lambda_2 \Rightarrow (a \sim_{\lambda_1} b \Rightarrow f(a) \sim_{\lambda_2} f(b) \, \forall a, b \in A, \ \forall f \in C)$$

Proof: Take $\lambda_1 \rightsquigarrow_C \lambda_2$ with $\lambda_1 = \phi(c, \lambda_0)$ and $\lambda_2 = \phi(gc, \lambda_0)$ for some $c \in A^*$ and $g \in C^*$ of the same length. Take some $a, b \in A$ with $a \sim_{\lambda_1} b$ then Σ being C–mappable implies that for all $f \in C \ \exists \bar{f} \in M(A)$ such that

$$\begin{aligned} \omega_{\lambda_2}(fa) &= \bar{f}\omega_{\lambda_1}(a) && (16)\\ &= \bar{f}\omega_{\lambda_1}(b) && (17)\\ &= \omega_{\lambda_2}(fb) && (18)\\ \Rightarrow \ f(b) &\sim_{\lambda_2} f(a) && (19) \end{aligned}$$

and we are done.

□

We know certain things about the structures here when C has more structure. For instance, if C is transitive, then $\rightsquigarrow_C$ is symmetric. If C is invertible, then $\rightsquigarrow_C$ is symmetric and $a \sim_{\lambda_1} b \Leftrightarrow f(a) \sim_{\lambda_2} f(b)$. When $\rightsquigarrow_C$ is symmetric, then it is interesting to take the transitive closure $\sim_C$.

Corollary 3. *If a system Σ is C–mappable and C is 2–transitive on A, then*

$$\lambda_1 \sim_C \lambda_2 \Rightarrow \begin{cases} \omega_{\lambda_1} \text{ and } \omega_{\lambda_2} \text{ are both constant maps, or} \\ \omega_{\lambda_1} \text{ and } \omega_{\lambda_2} \text{ are both permutations} \end{cases} \tag{20}$$

Proof: Note that transitivity of C, $\rightsquigarrow_C$ is symmetric.

Assume without loss of generality that $\lambda_1 \rightsquigarrow_C \lambda_2$.

If $a \sim_{\lambda_1} b$ for some $a \neq b \in A$, then $x \sim_{\lambda_2} y$ for all $x, y \in A$ and then by the symmetry of $\rightsquigarrow_C$ $x \sim_{\lambda_1} y$ for all $x, y \in A$. That is, ω_{λ_1} is a constant map.

If $a \sim_{\lambda_1} b \Leftrightarrow a = b$ then ω_{λ_1} is a permutation.

□

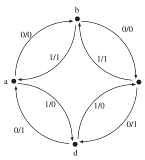
4. Four state example

A slightly more complex example can be constructed, demonstrating the class of examples introduced above.

The automaton in below has input/output alphabet $A = \{0, 1\}$ and will be acted on by the permutations $C = \{i, \sigma\}$ where i is the identity and σ is the involution on A. Note that C is 2–transitive.

The state set is $S = \{a, b, c, d\}$, the functions

(21)

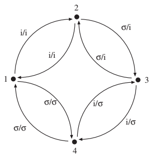


ϕ	0	1
a	b	d
b	c	a
c	d	b
d	a	c

ω	0	1
a	0	0
b	0	1
c	1	1
d	1	0

The mapping automaton for this automaton is as follows, with state set $\{1, 2, 3, 4\}$, state transition function δ and output function γ.

(22)



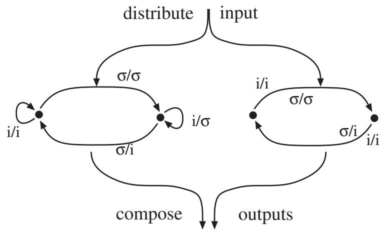
δ	i	σ
1	2	4
2	1	3
3	4	2
4	3	1

γ	i	σ
1	i	i
2	i	σ
3	σ	σ
4	σ	i

The states may be interpreted as follows:

1. = in state a or c, correctly.
2. = in state b or d, correctly.
3. = in state a or c, reversed.
4. = in state b or d, reversed.

This automaton has a two–part structure, closely related to the automaton for which it is the mapping automaton. The automaton can be split into two smaller automata and their outputs composed to produce the output of the whole machine.



The first automaton, which we might think of as that which follows the effect of the permutations, is a two state automaton with the following state transition and output functions:

(23)

δ	i	σ
1	1	2
2	2	1

γ	i	σ
1	i	σ
2	σ	i

The second automaton remembers in which reachability class we are, whether the even or odd path lengths, and has the following state transition and output functions:

(24)

δ	i	σ
1	2	2
2	1	1

γ	i	σ
1	i	σ
2	i	i

This breakdown into two parallel automata makes the structure of the automaton above more clear.

5. Explicit Mappability

In this section we introduce a form of more explicitly mappable automata. The intuition behind this construction is to maintain an explicit memory of how 'wrong' we are as a result of the mappings. It is related to the second interpretation of the mapping automaton for linear automata.

Proposition 4. *Let* $\Sigma = (S, A, \phi, \omega, \lambda_0)$ *be an automaton and* $C \subseteq M(A)$*. If there is a set* $B \subseteq M(S)$ *containing the identity and mappings*

$$\delta : C \times B \rightarrow B \tag{25}$$
$$\gamma : C \times B \rightarrow M(A) \tag{26}$$

with

$$\delta(f,\beta)(\phi(a,\lambda)) = \phi(fa, \beta\lambda) \tag{27}$$
$$\gamma(f,\beta)(\omega(a,\lambda)) = \omega(fa, \beta\lambda) \tag{28}$$

for all $f \in C$, $\beta \in B$, $a \in A$, $\lambda \in S$, *then* Σ *is* C*–mappable.*

Proof: We show this by using the definition of mappability.

Let $f = f_0 \ldots f_k \in C^*$ be a series of maps and $a = a_0 \ldots a_k \in A^*$ an input word of the same length. As usual define $\lambda_{i+1} = \phi(a_i, \lambda_i)$ for $i \geq 1$. Let $\lambda'_0 = \lambda_0$ and $\lambda'_{i+1} = \phi(f_i a_i, \lambda'_i)$. Let β_0 be the identity and $\beta_{i+1} = \delta(\beta_i, f_i)$. We claim that $\lambda'_i = \beta_i \lambda_i$. This is true for $i = 0$. Assume true for $i \leq j$. Then

$$\lambda'_{j+1} = \phi(f_j a_j, \lambda'_j) \tag{29}$$
$$= \phi(f_j a_j, \beta_j \lambda_j) \tag{30}$$
$$= \delta(\beta_j, f_j)(\phi(a_j, \lambda_j)) \tag{31}$$
$$= \beta_{i+1} \lambda_{j+1} \tag{32}$$

so we see that $\lambda'_i = \beta_i \lambda_i$ for all $i \geq 0$.

Let $\theta_i = \omega(a_i, \lambda_i)$ $\theta'_i = \omega(f_i a_i, \lambda'_i)$ and $\bar{f}_i = \gamma(\beta_i, f_i)$. Then

$$\bar{f}_i \theta_i = \gamma(\beta_i, f_i)(\omega(a_i, \lambda_i)) \tag{33}$$
$$= \omega(f_i a_i, \beta_i \lambda_i) \tag{34}$$
$$= \omega(f_i a_i, \lambda'_i) \tag{35}$$
$$= \theta'_i \tag{36}$$

so we see that Σ is mappable.

□

Mappable automata of this type will be called *explicit* (C*–*)*mappable automata.* The mapping automaton is then $(B, C, \delta, \gamma, i)$ with $i \in B$ the identity map.

An example can be constructed as follows. Take $A = \{0, 1\}$ and $C = \{i, \bar{0}\}$ where $\bar{0}$ is the constant 0 map. Let $S = \{a, b, c\}$ and define the transition and output maps by

(37)

ϕ	0	1
a	b	b
b	a	c
c	b	b

ω	0	1
a	0	0
b	0	0
c	0	1

Let μ be the mapping on S taking $a, c \mapsto a$, $b \mapsto b$ then the explicit mapping automaton is defined by

$$
\begin{array}{c|cc}
\delta & i & \bar{0} \\
\hline
i & i & \mu \\
\mu & i & \mu
\end{array}
\qquad
\begin{array}{c|cc}
\delta & i & \bar{0} \\
\hline
i & i & \bar{0} \\
\mu & \bar{0} & \bar{0}
\end{array}
\tag{38}
$$

This automaton is explicit but not linear.

All linear automata are explicit. The four state example from the previous section was not explicit. This can be seen by noting that the map $q \mapsto \delta(i, q)$ is not a morphism, by Theorem 6 below it must be. Thus we have the structure

$$\text{linear} \subset \text{explicit mappable} \subset \text{mappable} \tag{39}$$

with all inclusions proper.

6. Group Automata

This section looks at the case where C is a collection of permutations of A, in particular when C possesses a group structure. One motivation for this investigation is the observation that there are two extreme forms of mappings on a set, the constant maps at one extreme and the permutations at the other. We previously saw that all automata are C–mappable for a set C of constant maps. Unfortunately no such similar result exists in the case of C a set of permutations, but the preliminary results are still interesting.

Corollary 5. *If C is a 2–transitive group, and Σ a C–mappable automaton, then all $\bar{f}$ in the output of the mapping automaton can be selected to be invertible.*

Proof: We already know by Corollary 3 that all ω_λ are constant maps or permutations.

If $f \in C^*$ is of length k, and some ω_λ is a permutation for λ at distance k from the origin λ_0, then ω_λ is a permutation for all λ at distance k from the origin λ_0. Then the condition $\bar{f}\omega_{\lambda_1} = \omega_{\lambda_2} f$ can be rewritten as $\bar{f} = \omega_{\lambda_2} f \omega_{\lambda_1}^{-1}$, and $\bar{f}$ is thus a permutation.

Assume that we must use a non–invertible $\bar{f}$. By the previous paragraph, if some output map $\bar{f}$ must be uninvertible, then it must be forced by the existence of $a, b \in A^*$ of the same length k, $f \in C^*$ also of the same length,

$$\lambda_1 = \phi(a, \lambda_0) \qquad \lambda_2 = \phi(fa, \lambda_0) \tag{40}$$

$$\lambda_3 = \phi(b, \lambda_0) \qquad \lambda_4 = \phi(fb, \lambda_0) \tag{41}$$

and $\omega_{\lambda_1}, \omega_{\lambda_2}, \omega_{\lambda_3}, \omega_{\lambda_4}$ all constant maps. Then there is some $f_0 \in C$ such that $\bar{f}_0$ must be not one to one, i.e. there exist some $x \neq y \in A$ with $\bar{f}_0 x = \bar{f}_0 y$. Since we require $\bar{f}_0 \omega_{\lambda_1} = \omega_{\lambda_2} f_0$ and $\bar{f}_0 \omega_{\lambda_3} = \omega_{\lambda_4} f_0$ as Σ is mappable, there must be some $\lambda_1, \lambda_2, \lambda_3, \lambda_4$ such that $\omega_{\lambda_2} = \omega_{\lambda_4}$ $(= \bar{f}_0 x = \bar{f}_0 y)$ while $(x =)\omega_{\lambda_1} \neq \omega_{\lambda_3} (= y)$.

Since C is invertible we can contruct f^* from f above by inverting every entry. If we define $\alpha = fa$ and $\beta = fb$, then

$$\lambda_2 = \phi(\alpha, \lambda_0) \qquad \lambda_1 = \phi(f^*\alpha, \lambda_0) \tag{42}$$

$$\lambda_4 = \phi(\beta, \lambda_0) \qquad \lambda_3 = \phi(f^*\beta, \lambda_0) \tag{43}$$

Then since Σ is mappable, for any $f_1 \in C$ there is some mapping $\bar{f}_1$ such that

$$\bar{f}_1 \omega_{\lambda_2} = \omega_{\lambda_1} f_1 \tag{44}$$

$$\bar{f}_1 \omega_{\lambda_4} = \omega_{\lambda_3} f_1 \tag{45}$$

but $\omega_{\lambda_2} = \omega_{\lambda_4}$ while $\omega_{\lambda_3} \neq \omega_{\lambda_1}$ by assumption above. Thus $\bar{f}_1$ is not a mapping, a contradiction. Thus our assumption is wrong and no $\bar{f}$ is forced to be noninvertible.

□

Of course this does not force the $\bar{f}$ to be invertible mappings, only partial one–to–one mappings, but if A is a finite alphabet they can be extended (non–uniquely) to permutations of the alphabet.

Thus looking at mappable systems, a class of them, given invertible mappings as their "input", will produce only invertible mappings as their output. In this case we can view the automata as mapping a group alphabet of permutations to itself, we have a group automaton in the sense of Pilz [3].

If we look at the explicit mappable automata with C being a set of permutations closed under the inverse operation, i.e. in a permutation f is in C, then the inverse permutation $f^{-1} \in C$, we find the following.

Theorem 6. *If Σ with $\phi(A, S) = S$ is explicit C–mappable with C a set of permutations closed under the inverse operation, then $B \subseteq Symm(S)$. If C is closed under the operation of composition, i.e. forming a group, then so is B and $(B, C, \delta, \gamma, i)$ is then a group automaton. If C is abelian, then so is B.*

Proof: The result follows by induction. We are interested in the state set of the mapping automaton $(B, C, \delta, \gamma, i)$. The following two paragraphs show that if we know that all states $\beta \in B$ at distance k from the start state i are permutations, and the inverse of β is also at distance k from the start state, then all states in B at distance $k+1$ are permutations and their inverses are also at distance $k+1$. We start at distance 0, knowing that only the identity map, which is its own inverse, is at distance 0 from the start state. The rest will then follow by induction on k.

Let $\beta_1 \in B$ be invertible at distance k, let $\beta_2 = \delta(f, \beta_1)$ be at distance $k+1$ for some $f \in C$. Then for all $a \in A,\ \lambda \in S$,

$$\phi(a, \lambda) = \phi(f^{-1}(fa), \beta_1^{-1}(\beta_1 \lambda)) \tag{46}$$

$$= \delta(f^{-1}, \beta_1^{-1})(\phi(fa, \beta_1 \lambda)) \tag{47}$$

$$= (\delta(f^{-1}, \beta_1^{-1}) \circ \beta_2)(\phi(a, \lambda)) \tag{48}$$

Thus since $\phi(A, S) = S$, we know that $\delta(f^{-1}, \beta_1^{-1}) \circ \beta_2$ is the identity map on S. Similarly $\beta_2 \circ \delta(f^{-1}, \beta_1^{-1})$ is the identity of S, so β_2 must be invertible with $\beta_2^{-1} = \delta(f^{-1}, \beta_1^{-1})$.

By the inductive hypothesis β_1 and β_1^{-1} are at distance k from the start state i, thus states β_2 and β_2^{-1} are at distance $k+1$ from the start state.

Thus the induction holds.

Now assume C is closed. Then it is closed under inverses. Once again we work with an induction on distance from the start state i. The hypothesis is that if $\beta_1, \beta_2 \in B$ are at distance at most k from the origin, then $\beta_1\beta_2 \in B$. This is vacuously true for $k = 0$. Assume it to be true for some k. Take $\beta_2 = \delta(f_1, \beta_1)$ and $\beta_4 = \delta(f_3, \beta_3)$ at distance $k+1$ from the start state with β_1, β_3 at distance k from the start state. Then

$$\delta(f_1 f_3, \beta_1\beta_3)\phi(a, \lambda) = \phi(f_1 f_3 a, \beta_1\beta_3\lambda) \tag{49}$$

$$= \delta(f_1, \beta_1)\delta(f_3, \beta_3)\phi(a, \lambda) \tag{50}$$

$$= (\beta_2\ \beta_4)\phi(a, \lambda) \tag{51}$$

so $(\beta_2\ \beta_4) = \delta(f_1 f_3, \beta_1\beta_3) \in B$. Thus B is closed under multiplication by the inductibe hypothesis. By finiteness it is a subgroup.

Note also that the state transition map δ in the mapping automaton is a morphism.

A similar argument to the closure argument can then be used to show that if C is abelian, then so is B.

□

Note that this result also shows a certain structure in the state set, regarding the inverses of the states.

7. Conclusion

We have shown that a property of linear automata dealing with their behaviour under perturbations can be generalised to some nonlinear automata. A stronger structure for a class of mappable automata called explicit automata has ben demonstrated, and in the case of invertible perturbations more structure in the resulting mapping automaton has been found. An example demonstrates that the parallel automata composition used in e.g. [3] appears naturally with mapping automata.

References

[1] Tim Boykett. Loop. In *Closing the Loop 98*. Time's Up, 1999.

[2] Gerhard Hofer. Principal series and group automata. In *Contributions to General Algebra 5*, pages 185–188, 1987.

[3] Günter Pilz. Near-rings and non-linear dynamical systems. In Gerhard Betsch, editor, *Near-rings and Near-fields*, pages 211–232. North-Holland, 1987.

Time's Up, Industriezeile 33b, A-4020 Linz, Austria and Institüt für Mathematik, Uni Linz, A-4040 Linz, Austria

E-mail address: `tim@algebra.uni-linz.ac.at`