Acta Cybernetica **17** (2005) 199–211.

Cycle Structure in Automata and the Holonomy Decomposition

Attila Egri-Nagy* and Chrystopher L. Nehaniv*

Abstract

The algebraic hierarchical decomposition of finite state automata can be applied wherever a finite system should be 'understood' using a hierarchical coordinate system. Here we use the holonomy decomposition for characterizing finite automata using derived hierarchical structure. This leads to a characterization according to the existence of different cycles within an automaton. The investigation shows that the problem of determining holonomy groups can be reduced to the examination of the cycle structure of certain derived automata. The results presented here lead to the improvements of the decomposition algorithms bringing closer the possibility of the application of the cascaded decomposition for real-world problems.

1 Introduction

The aim of this paper is to study the cycle structure in automata associated to the holonomy decomposition in Krohn-Rhodes Theory. With a recent computational tool [5] (developed by the authors) the Krohn-Rhodes theory [9] finally has computational means to foster further research in it and to show its real significance to scientists working outside theoretical computer science. The main aim of this paper is to summarize the theoretical insights gained from the systematic study of finite state automata by examining their derived hierarchical decomposition computed by the implemented holonomy decomposition [6, 4], and show how these insights may be used for improving the algorithms. It also can be considered as a first – although still theoretical – computational application of the Krohn-Rhodes theory remaining within the confines of algebraic automata theory. Further possible applications come up in all different fields where we deal with hierarchical models of systems: physics [13], software-development [10], artificial intelligence [11], evolutionary biology [12], etc.

As the holonomy decomposition mainly deals with certain sets of subsets of an automaton's state set that are permuted by input words, our investigation concentrates on the question of when *nontrivially* permuted sets of appropriate subsets really exist and of recognizing when automata are completely without them.

*School of Computer Science, University of Hertfordshire, College Lane, Hatfield, Herts AL10 9AB, United Kingdom, Email: {A.Nagy | C.L.Nehaniv}@herts.ac.uk

2 Mathematical Preliminaries and Notations

Here we establish the close connection between finite state automata and some algebraic structures called semigroups as it is more convenient to handle automata algebraically. The connection between these structures is outlined here with special emphasis on the cascaded product of automata, together with the notions of division and wreath product. For more details see [4, 1, 6].

2.1 Transformation Semigroups

Semigroups. A *semigroup* is a set S equipped with an associative binary operation $\mu : S \times S \to S$. Instead of $\mu(s_1, s_2)$ we write $s_1 \cdot s_2$ or more briefly $s_1 s_2$. If A and B are subsets of a semigroup, then AB means the set $\{ab : a \in A, b \in B\}$. An element 1 is the identity element of S if $s1 = 1s = s$, for all $s \in S$. The identity is unique if it exists. By S^1 we denote S if it has an identity otherwise $S \cup \{1\}$. By S^I we mean $S \cup \{I\}$ where I acts as an identity on S and itself, the identity of S (if it exists) ceases to be an identity as it fails on I. The *order* of a semigroup S is its cardinality $|S|$. We say that G generates the semigroup $\langle G \rangle = S$ if $G \subseteq S$ and all elements of S can be expressed as a finite product of elements in G. A semigroup S is *aperiodic* if for each element $s \in S$ there is a positive natural number n such that $s^n = s^{n+1}$; for a finite semigroup this means that it contains no nontrivial subgroups.

Homomorphisms. Let S and T be semigroups with operations $\circ, \diamond$ respectively, and having a mapping $\psi : S \to T$ such that $\psi(s_1 \circ s_2) = \psi(s_1) \diamond \psi(s_2)$, for all $s_1, s_2 \in S$. Then we say that ψ is a *homomorphism* from S to T, a mapping which preserves products. If a homomorphism is bijective then it is an *isomorphism*.

Groups. A semigroup is a *monoid* if it has an identity element. A monoid is a *group* if for every $s \in S$ there is an inverse $s^{-1} \in S$ such that $ss^{-1} = s^{-1}s = 1$. A subset T of a semigroup S is a *subsemigroup* if it is closed under the multiplication of S. Subgroups are defined analogously. A subgroup H of a group G is *normal* if $gH = Hg \ \forall g \in G$. A nontrivial group is *simple* if it has no nontrivial normal subgroups.

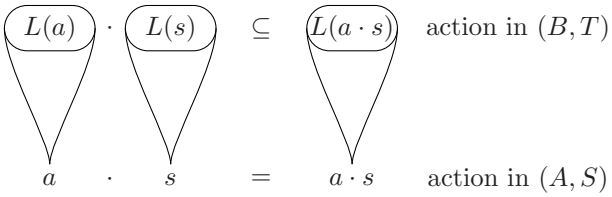
Transformations. For a nonvoid finite set A, a mapping $\varphi : A \to A$ is called a *transformation* of A. If the mapping is bijective, then it is a *permutation.* The *image* of φ is defined as $\{a\varphi : a \in A\}$ denoted by $\mathrm{im}(\varphi)$. If the image of a mapping is a singleton then the mapping is *constant*. The *rank* of a transformation is the cardinality of its image. The set T of all transformations of A form a semigroup under the operation of function composition of transformations and it is called the *full transformation semigroup* denoted by $\mathcal{T}_A = (A, T)$. If S is a subsemigroup of T then (A, S) is called a *transformation semigroup* on A (or briefly a *ts*), and we say that S *acts* on A. (A, S) is a *permutation group* if each elements $s \in S$ acts on A by permutation. We write $a \cdot s$ for the image of state a under the transformation s, and we have $(a \cdot s_1)s_2 = a \cdot (s_1 s_2)$ for all $a \in A, s_1, s_2 \in S$. It is a basic fact of semigroup theory that every finite semigroup can be represented as a ts using

the *right regular representation* (S^1, S) where S acts on S^1 by multiplication on the right [3]. If (A, S) is a transformation semigroup, we denote by $(A, \overline{S})$ the transformation semigroup with transformations $\overline{S} = \{t \mid t \in S \text{ or } t \text{ is constant}\}$.

Division. We say that a transformation semigroup (A, S) *divides* (B, T) denoted by $(A, S) \mid (B, T)$ if we can choose for all $a \in A$ at least one $\tilde{a} \in B$ as a *lift* and and for each $s \in S$ at least one $\tilde{s} \in T$ as a *lift*, such that the following hold:

1. Each member of B (resp. T) is a lift of at most one element of A (resp. S), i.e. the (non-empty) lift sets are non-intersecting,
2. If $\tilde{a}$ is any lift of a and $\tilde{s}$ is any lift of s, then $\tilde{a} \cdot \tilde{s}$ is some lift of $a \cdot s$, i.e. the products are respected.

Denote the set of lifts of a state a by $L(a)$ (and $L(s)$ for a transformation s respectively). Note that in general $L(a) \cdot L(s) \subseteq L(a \cdot s)$, instead of being equal.



Words and the free semigroup.[15] Let X the set of letters be called the *alphabet*. A *word* over the alphabet X is a finite sequence of elements of X: $(x_1, x_2, \ldots, x_n)$, $x_i \in X$. The empty word is denoted by λ. X^+ is the set of all non-empty finite words. X^+ is a semigroup under the operation of concatenation, it is called the *free semigroup*. $X^* = X^+ \cup \{\lambda\}$ is the *free monoid*.

A word $v \in X^*$ is a *factor* of a word $z \in X^*$ if there exist words $u, w \in X^*$ such that $z = uvw$. v is a *left factor* of z if there exists a word $w \in X^*$ such that $z = vw$. A word w is *primitive* if it is not a power of another word. For any nonempty word w, the smallest factor u such that $w = u^n$, $n \geq 1$ is the *primitive root* of w. We also use the notation $u = \sqrt{w}$.

2.2 Finite State Automata

By a finite state *automaton*, we mean a triple $\mathcal{A} = (A, X, \delta)$ where A is the (finite nonempty) *state set*, X is the *input alphabet* and $\delta : A \times X \to A$ is the *transition function*. We do not explicitly consider the output of the automaton as it can be recovered from the state and the input symbol. We tacitly use the state as the output.

We can naturally extend the transition function for words i.e. sequences of input symbols: for the empty word $\delta(a, \lambda) = a$, and for arbitrary words $u, v \in X^*$, $\delta(a, uv) = \delta(\delta(a, u), v)$. There is a natural equivalence relation, the *congruence induced by* $\mathcal{A}$ on words $u \equiv v$ if $\delta(a, u) = \delta(a, v)$ $\forall a \in A$, i.e. identifying words with

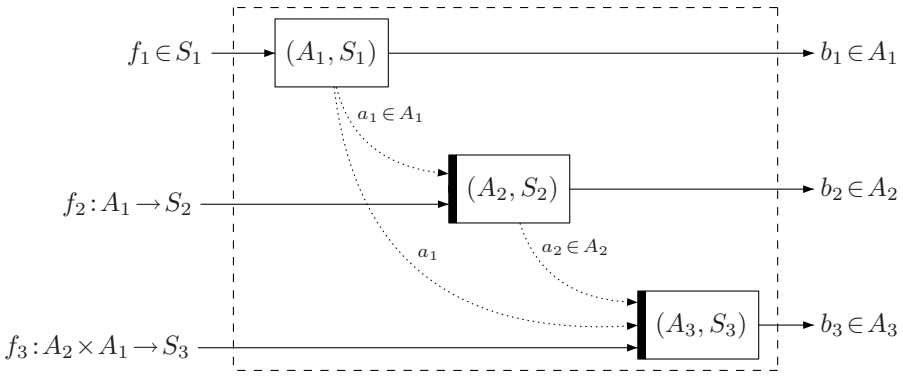


Figure 1: State transition in the wreath product $(A_3, S_3) \wr (A_2, S_2) \wr (A_1, S_1)$. The transformation (f_3, f_2, f_1) is applied to state (a_3, a_2, a_1) yielding $(b_3, b_2, b_1) = (a_3 \cdot f_3(a_2, a_1), a_2 \cdot f_2(a_1), a_1 \cdot f_1)$. The black bars denote the applications of functions f_2, f_3 according to hierarchical dependence. Note that the applications of these functions happen exactly at the same moment since their arguments are the previous states of other components, therefore there is no need to wait for the other components to calculate the new states. We use the state as the output of the automaton.

the same action on A. The *characteristic semigroup* $S(\mathcal{A})$, also called the *semigroup of the automaton*, is the set equivalence classes $X^+/\equiv$ of this congruence, with associative operation induced by concatenation. With the characteristic semigroup we can handle an automaton $\mathcal{A}$ as a transformation semigroup $(A, S(\mathcal{A}))$. Conversely if S is a semigroup then the corresponding automaton is $\mathcal{A}_S = (S^1, S)$, where the transition function is the right action of S on S^1.

An automaton $\mathcal{A}$ *emulates* another one $\mathcal{B}$ with states B if every computation which can be done in $\mathcal{B}$ can be done in $\mathcal{A}$ as well, i.e. $(B, S(\mathcal{B}))$ divides $(A, S(\mathcal{A}))$.

Using automata terminology constant mappings in transformation semigroups are often called *resets.* A *permutation-reset* automaton is an automaton such that each of its inputs acts either as a permutation or a constant map on states.

The *state transition graph* $D(\mathcal{A})$ of an automaton $\mathcal{A} = (A, X, \delta)$ is a digraph with A as the set of vertices and (a, x, b) is a labelled edge if $a \cdot x = b$, where $a, b \in A,\ x \in X$. It is a *loop-edge* if $a = b$. A *path* is a sequence of edges (a_i, x_i, b_i) $1 \leq i \leq n$ with $a_{i+1} = b_i$ for all $1 \leq i < n$, and the *label* of the path is $x_1 \ldots x_n$. A *loop* is a path with $b_n = a_1$.

2.3 Wreath Product Explained

Although the concept of the wreath product is not so complicated, it is not as easy to present the intuitive idea how the loop-free cascaded product works. After reading the formal definition a figure may shed light on how state transitions happen in the product (Fig. 1). It is also a great help first to consider a simpler product

with no dependence between the components.

Let $(A_n, S_n), \ldots, (A_1, S_1)$ be transformation semigroups called *components*. The indices $1, \ldots, n$ are called *coordinates*. The *direct product* $(A_n, S_n) \times \ldots \times (A_1, S_1)$ is the ts $(A_n \times \ldots \times A_1, S_n \times \ldots \times S_1)$ with the componentwise action

$$(a_n, \ldots, a_1) \cdot (s_n, \ldots, s_1) = (a_n \cdot s_n, \ldots, a_1 \cdot s_1).$$

Direct product is also called *parallel composition* as the components' state transitions do not depend on each other, and the order of the components does not really matter up to isomorphism.

Now we introduce an order-dependent connection between the components. Let $A = A_n \times \ldots \times A_1$ and $\mathcal{T}_A$ the full ts on A. Let S be the subsemigroup of $\mathcal{T}_A$ consisting of all transformations $s : A \to A$ satisfying the condition of *hierarchical dependence* of coordinates. Denote $p_k : A \to A_k$ the kth projection map, then for each $k = 1, \ldots, n$ there exists $f_k : A_{k-1} \times \cdots \times A_1 \to S_k$ such that

$$p_k\big((t_n, \ldots, t_{k+1}, t_k, \ldots, t_1) \cdot s\big) = t_k \cdot f_k(t_{k-1}, \ldots, t_1) = t'_k$$

$$\text{where } s \in S,\ \ t_k, t'_k \in A_k,\ \ k = 1, \ldots, n.$$

That is, the new kth coordinate t'_k resulting from the action of s depends only on the values of the old first k coordinates and on the transformation s. Moreover, it is given by acting with an element of S_k which depends only on s and $(t_{k-1}, \ldots, t_1)$. We can write this transformation as the ordered list of these functions: $s = (f_n, \ldots, f_1)$.

Then the transformation semigroup $(A, S) = (A_n, S_n) \wr \ldots \wr (A_1, S_1)$ is the *wreath product* of transformation semigroups $(A_n, S_n), \ldots, (A_1, S_1)$. Reading from left to right the last component is the top level of the hierarchy.

3 Holonomy Decomposition Theorem

The holonomy decomposition originates from Zeiger's method of proving the Krohn-Rhodes Theorem [16, 17, 7]. This algorithm work by the detailed study of how the semigroup S of an automaton (A, X, δ) acts on subsets of A. It looks for groups induced by S permuting some set of subsets of A. These groups are called the *holonomy groups*. These groups are the building blocks for the components of the decomposition. As we go deeper in the hierarchy of the cascade composition we have components that act on subsets with smaller cardinality.

The sketch of the algorithm to obtain a decomposition: First calculate the set of images of transformations in S. From now on, let $\mathcal{I}$ denote this set extended by A itself and its singletons. On $\mathcal{I}$ there is a preorder relation called *subduction* defined. A subset P is subduction related to a subset Q if P is contained in a resulting set of acting by some $s \in S$ on Q, i.e. $P \subseteq Q \cdot s$. The mutual relation of elements induces an associated equivalence relation $P \equiv Q \iff P \leq Q$ and $Q \leq P$. The set of equivalence classes are partially ordered by the subduction relation. The set of equivalence classes and their partial order are called the *subduction picture*. The

tiles B_P of a subset P ($P \in \mathcal{I}, |P| > 1$) are its proper subsets directly below it in the subduction preorder. The union of its tiles equals to P. The length of a longest strict path from a singleton to a subset P in the partial order of subduction equivalence classes defines the *height* of the subsets within the equivalence class of P. Equivalence classes with the same height are on the same hierarchical level. The sets of tiles for each element $Q \in \mathcal{I}$ form the *tiling picture*. The holonomy group H_Q of Q is the group (arising from elements of S^1) permuting the tile set B_Q of Q. The component $\overline{\mathcal{H}_i}$ of one hierarchical level i is the direct product of the holonomy groups belonging to the representative elements of equivalence classes with height i augmented with the constant mappings.

Theorem 1 (Holonomy Decomposition [6, 4])**.** *Let* (A, S) *be a finite transformation semigroup then* (A, S) *divides a wreath product of its holonomy permutation-reset transformation semigroups* $(\mathcal{B}_1, \overline{\mathcal{H}_1}) \wr \cdots \wr (\mathcal{B}_h, \overline{\mathcal{H}_h})$.

This strong formulation of part of the Krohn-Rhodes theorem is slightly different from the original since the components here are groups extended with constants and not simple groups and the divisors of the flip-flop. But these permutation-reset components can be easily decomposed into flip-flops and groups. Moreover the groups can be further decomposed into a series of simple groups using the Lagrange Coordinate Decomposition Theorem and Jordan-Hölder Theorem [8, 4]. Note that the top level of the hierarchy is the component with highest index, not 1.

4 Cycles in Automata

Definition 2. *A* graphical cycle *in an automaton* (A, X, δ) *is a cycle in its state transition digraph together with a word* $w \in X^+$*, i.e. a sequence of states* $a_1, \ldots, a_n$ $n \geq 2$*, where the states in the sequence are pairwise distinct except* $a_1 = a_n$*, and* $w = x_1 \ldots x_{n-1}$*,* $x_i \in X$ *such that* $a_i \cdot x_i = a_{i+1}$ *for all* $1 \leq i \leq n-1$*. The word* $w = x_1 \ldots x_{n-1}$ *is called the* label *of the cycle.*

Since $n \geq 2$ a loop edge is not a graphical cycle, and also, since $a_i \neq a_{i+1}$ within a graphical cycle, loop edges are not allowed.

Definition 3. *An* algebraic cycle *in an automaton* $\mathcal{A} = (A, X, \delta)$ *is a permutation group* $(\{a_1, \ldots, a_n\}, \langle w \rangle)$ *for which* $a_i = a_j \Rightarrow i = j$*,* $n > 1$*, and* w *is a word in* X^+ *such that* $a_i \cdot w = a_{i+1}$ *for all* $1 \leq i < n$*, and* $a_n \cdot w = a_1$*.*

The word w generates a cyclic group which acts faithfully on $\{a_1, \ldots, a_n\}$ by permutations. (Of course $\langle w \rangle$ might not act by permutations on A.) Obviously w^n is the identity element. Moreover, n being greater than 1 excludes trivial one-element groups. Note that loops are not generally algebraic cycles. The *generator* of the algebraic cycle is w, and its label is w^n.

5 Graphically Cycle-Free Automata

Definition 4. *An automaton is* graphically cycle-free *if it does not have any graphical cycle.*

The very simple structure of graphically cycle-free automata is reflected in their subduction pictures in the following way:

Lemma 5. *(A, S) is graphically cycle-free iff on every height level in each subduction relation equivalence class there is only one element.*

Proof: Let $P, Q \in \mathcal{I}$ and $P \equiv Q$ but $P \neq Q$. Since P, Q are finite $|P| = |Q|$. Clearly by finiteness there is at least one $x \in Q$ such that $x \notin P \cap Q$, otherwise P, Q would be the same. Due to the equivalence of P and Q we have $s, t \in S$ bijective mappings such that $P = Q \cdot s$ and $Q = P \cdot t$ and thus $(st)^n$ is the identity on Q for some $n > 0$, by the finiteness of P, Q. Since $x \cdot s = x' \neq x$ while $x \cdot (st)^n = x$, there must be a graphical cycle.

Conversely, a graphical cycle ensures the existence of an equivalence class with at least two elements at height zero. □

Another way to think about the proof of this lemma is to recognize that for the singleton subsets of the state set (at height zero) the equivalence classes are exactly the strongly connected components of the automaton's state transition graph.

This result can be exploited in the decomposition algorithm since if the equivalence classes are detected to all be singleton classes, then there is no need to look for holonomy groups at all and the holonomy identity-reset ts's can be built immediately.

6 Algebraically Cycle-Free Automata

It is a well-known result of algebraic automata theory that the star-free rational languages are recognized by exactly those automata whose characteristic monoid is aperiodic (having no nontrivial subgroup)[14]. It is also known that deciding aperiodicity for a finite automaton is PSPACE-complete[2]. We are interested in this problem for certain derived automata that arise naturally in the holonomy decomposition.

Intuitively one might expect that the state transition graph of an aperiodic automaton contains no cycles at all, but this is not true in general: there might be graphical cycles in it, while remaining aperiodic (see Fig 2). But with another type of cycles the notion of aperiodicity can be expressed.

Definition 6. *An automaton $\mathcal{A} = (A, X, \delta)$ is* algebraically cycle-free *if it does not have any algebraic cycle.*

The property of algebraic cycle-freeness is tied up with the primitivity of words, which act on some states as the identity.

Lemma 7. *An automaton $\mathcal{A} = (A, X, \delta)$ is algebraically cycle-free iff for all states $a \in A$ and for all words $w \in X^+$ such that $a \cdot w = a$, one of the following statements holds.*

1. *w is primitive.*
2. *w is not primitive but has primitive root $u \in X^+$, i.e. $w = u^n$, and $a \cdot u = a$.*

Proof: If w is primitive, then we are done. Otherwise $w = u^n$ where u is primitive. Let's suppose indirectly that $a \cdot u \neq a$. Let k be the least integer that $a \cdot u^k = a$ $(1 < k \leq n)$. Then $(\{a, a \cdot u, \ldots, a \cdot u^{k-1}\}, \langle u \rangle)$ is a cyclic permutation group (with at least two elements), therefore we have an algebraic cycle, contradicting our assumptions.

The converse is obvious due to the fact that a trivial permutation group does not constitute an algebraic cycle, and the conditions $1-2$ allow only trivial permutation groups. □

Remark 8. *Obviously Lemma 7 holds even if $a \cdot z \neq a$ for some left factor z of w.*

It is clear that in the absence of graphical cycles there cannot be any algebraic cycle. Thus,

Proposition 9. *If an automaton is graphically cycle-free then it is algebraically cycle-free.*

Now we show that aperiodic automata are exactly the algebraically (not the graphically) cycle-free ones.

Theorem 10. *The following are equivalent for an automaton $\mathcal{A} = (A, X, \delta)$ with corresponding transformation semigroup (A, S):*

1. *$\mathcal{A}$ is algebraically cycle-free.*
2. *S is aperiodic.*
3. *Holonomy groups are trivial for (A, S).*

Proof: (1) ⇒ (2): Suppose S is not aperiodic, then we have a cyclic group $\langle v \rangle$ in S of order $n \geq 2$, where $v \in X^+$ is a word representing the generator. Thus v^n is the identity of the cyclic group, $v \equiv v^{n+1}$ and $v \not\equiv v^2$. Therefore $\exists a$ such that $a \cdot v \neq a \cdot v^2$ and $a \cdot v = a \cdot v^{n+1}$. Let $a' = a \cdot v$, thus $a' \cdot v^n = a'$ and since $\mathcal{A}$ is algebraically cycle-free we can apply Lemma 7: let $u = \sqrt{v^n} = \sqrt{v}$, then we have $a' \cdot u = a'$, $a' \cdot v = a'$ and finally $a \cdot v^2 = a \cdot v$, which is a contradiction.

(2) ⇒ (1): For the converse we use again an indirect proof: Suppose there is an algebraic cycle, i.e. $(\{a_1, \ldots, a_n\}, \langle w \rangle)$ is a permutation group with $a_i \in A, w \in X^+$ and $n > 1$. Therefore $\mathbb{Z}_n$, the cyclic group with n elements, divides S. This cannot happen when S is aperiodic.

Figure 2: Automaton $\mathcal{A}$ has an algebraic cycle $(\{1,2\},\langle a\rangle)$. Automaton $\mathcal{B}$ has graphical cycles ab, ba, but they are labelled with primitive words.

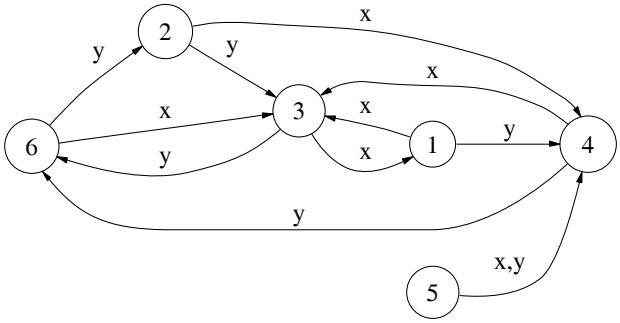


Figure 3: An automaton $\mathcal{A}$ with state set $A = \{1,2,3,4,5,6\}$ and alphabet $\{x,y\}$, where x and y are transformations with $x = (3\ 4\ 1\ 3\ 4\ 3)$, $y = (4\ 3\ 6\ 6\ 4\ 2)$.

(2) $\Leftrightarrow$ (3): The components of the holonomy decomposition are all divisors of the original semigroup, thus aperiodic semigroups have only trivial holonomy groups, and wreath products and divisors of aperiodic transformation semigroups are aperiodic. □

Corollary 11. *An automaton $\mathcal{A} = (A, X, \delta)$ is aperiodic if and only if*

$$\forall a \in A,\ w \in X^+,\ x \cdot w = a \Rightarrow a \cdot \sqrt{w} = a.$$

The distinction between algebraically cycle-free aperiodic and nonaperiodic automata is rather subtle. Two automata having the same state-transition graphs regarding their connectivity might belong to different classes depending on how the input symbols act on the state set (Fig. 2).

7 Non-Aperiodic Automata

A main concern of the holonomy decomposition is to find the nontrivial holonomy groups. Fortunately the tiling picture provides tools for locating the elements of $\mathcal{I}$ for which there exist nontrivial holonomy groups.

Lemma 12. *For an element Q of $\mathcal{I}$ in the tiling picture of (A,S) if there is a nontrivial holonomy group H_Q, then in its set of tiles B_Q there are at least two distinct tiles t_1, t_2 such that $t_1 \equiv t_2$.*

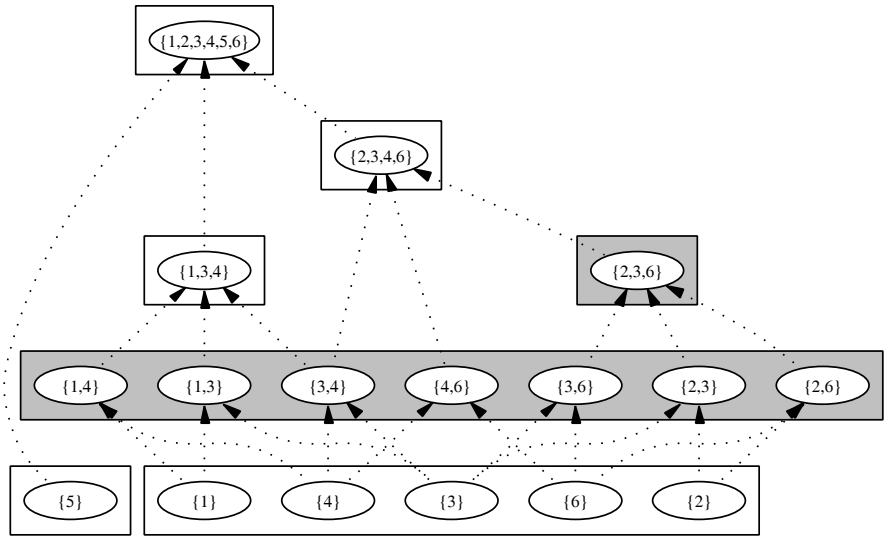


Figure 4: The tiling picture of automaton $\mathcal{A}$ in Fig. 3. The equivalence classes are denoted by boxes. Equivalence classes with elements having nontrivial holonomy groups are shaded. Dotted edges denote the 'tile of' relation.

Proof: H_Q being nontrivial means that there are some pair(s) of tiles for which there are transformations permuting them and thus they are mutually subduction related. □

The converse is not generally true as we can see in the example of an automaton (Fig 3) with tiling picture (Fig 4). For a trivial H_Q the set of tiles B_Q may contain distinct equivalent tiles, see Fig 5. In order to determine whether we have a nontrivial holonomy group for a $Q \in \mathcal{I}$ we define an extended automaton and examine its cycle structure. Denote the equivalence classes of subduction relation by E_1 to E_N.

Lemma 13. *If $P \in E_i$ and for some $s \in S$, $P \cdot s = Q$ such that $Q \notin E_i$ (leaving the equivalence class) then there is no transformation $t \in S$ such that $Q \cdot t \in E_i$ (no way back to the original equivalence class).*

Proof: Suppose there is such a t that $Q = P \cdot s$ and $P' = Q \cdot t$ with $P \equiv P'$. Due to the equivalence we have $P = P' \cdot s''$ for some $s'' \in S$, therefore $Q \cdot (ts'') = P' \cdot s'' = P$, thus $Q \equiv P$, which contradicts the original assumption that we leave the equivalence class of P. □

Let's define E_Q as the union of equivalence classes which contain at least one tile of $Q \in \mathcal{I}$. Formally: $E_Q = \bigcup_{E_i \cap B_Q \neq \emptyset} E_i$. Then the *tile automaton* of Q is

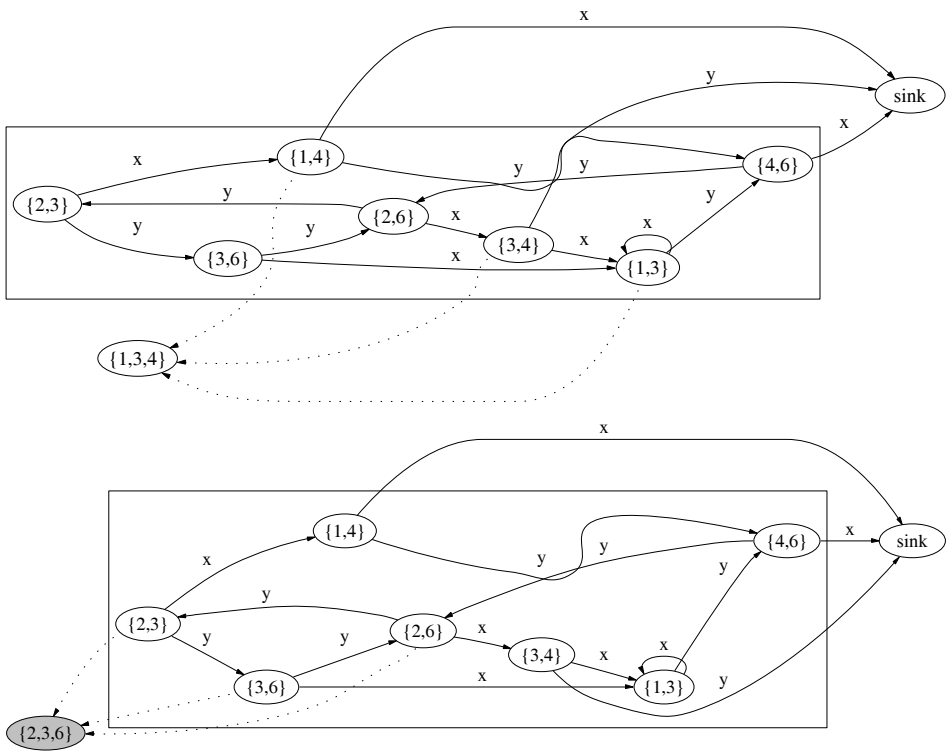


Figure 5: Two tile automata of automaton $\mathcal{A}$ in Fig. 3. $\mathcal{A}_{\{1,3,4\}}$ is trivial, while $\mathcal{A}_{\{2,3,6\}}$ is nontrivial with generator word y.

defined as $\mathcal{A}_Q = (E_Q \cup \{\varsigma\}, X, \gamma)$, where ς is a sink state, the input alphabet X is the same as the original automaton's, and γ is the natural extension of δ to act on subsets of A providing that if the image is not in some E_i then it is ς. This way ς represents going to another equivalence class not contained in E_Q, but according to Lemma 13 this can be represented as a sink since there is no way to come back.

The equivalence classes in E_Q form strongly connected components in $\mathcal{A}_Q$. When determining the nontriviality of H_Q we look for algebraic cycles in these components. We look not simply for independent algebraic cycles in each component as a word of a cycle might not permute the tile elements in another component, but for parallel algebraic cycles. This way we can recast the characterization of a holonomy group element in terms of algebraic cycles. More formally:

Proposition 14. *H_Q is nontrivial iff there exists a word $w \in A^+$ and B_Q can be partitioned into $\{T_1, \ldots, T_k\}$ subsets such that either*

1. *T_i consists of exactly one tile and $T_i \cdot w = T_i$, or*
2. *$T_i \cdot \langle w \rangle \subseteq B_Q \cap E_j$ for some $1 \leq j \leq N$, and $(T_i \cdot \langle w \rangle, \langle w \rangle)$ is an algebraic cycle in $\mathcal{A}_Q$*

holds for all T_i, $1 \leq i \leq k$, and (2) must hold for at least one T_i.

In short the proposition characterizes when the transformation induced by w nontrivially permutes B_Q. This transformation is clearly a nontrivial holonomy group element. From Lemma 13, $T_i \cdot w^n \in (B_Q \cap E_j)$ follows for any $n \geq 0$. Therefore the algebraic cycles contained in B_Q generated by w are all disjoint. If all intersections $(B_Q \cap E_j)$ are singletons, or none of them contains an algebraic cycle then H_Q is trivial. This fact can be exploited in efficient decomposition algorithms of the holonomy decomposition by excluding cases where the construction of the holonomy group should not be attempted.

8 Conclusions

Using an implementation of the holonomy decomposition we could get new insights about its working mechanism and found a relation between the cycle structure of an automaton and its holonomy components. We also showed that detecting cycles with primitive words helps in excluding elements of $\mathcal{I}$ when searching for holonomy groups. Currently we are investigating the possibility of efficient construction of holonomy groups by using the extended tile automata replacing the current algorithm which is based on a breadth-first search in the space of semigroup elements. These results will eventually lead to improvements of the decomposition algorithms providing efficient and scalable tools for attacking real-world problems such as analyzing metabolic networks [13], understanding biological complexity [12], AI applications [11] and so on.

References

[1] Michael A. Arbib, editor. *Algebraic Theory of Machines, Languages, and Semigroups*. Academic Press, 1968.

[2] Sang Cho and Dung T. Huynh. Finite-automaton aperiodicity is pscpace-complete. *Theoretical Computer Science*, 88:99–116, 1991.

[3] A.H. Clifford and G.B. Preston. *The Algebraic Theory of Semigroups (Mathematical Survey, No 7)*, volume 1 of *Mathematical Survey*. American Mathematical Society, 2nd edition, 1967.

[4] Pál Dömösi and Chrystopher L. Nehaniv. *Algebraic Theory of Finite Automata Networks: An Introduction*, chapter 3, The Krohn-Rhodes and Holonomy Decomposition Theorems. SIAM Series on Discrete Mathematics and Applications, 2005.

[5] Attila Egri-Nagy and Chrystopher L. Nehaniv. GrasperMachine, Computational Semigroup Theory for Formal Models of Understanding. (`http://graspermachine.sf.net`)., 2003.

[6] Samuel Eilenberg. *Automata, Languages and Machines*, volume B. Academic Press, 1976.

[7] Abraham Ginzburg. *Algebraic Theory of Automata*. Academic Press, New York, 1968.

[8] Marshall Hall. *The Theory of Groups*. The Macmillan Company, New York, 1959.

[9] Kenneth Krohn and John Rhodes. Algebraic theory of machines. I. Prime decomposition theorem for finite semigroups and machines. *Transactions of the American Mathematical Society*, 116:450–464, April 1965.

[10] Chrystopher L. Nehaniv. Algebraic engineering of understanding: Global hierarchical coordinates on computation for the manipulation of data, knowledge, and process. In *Proc. 18th Annual International Computer Software and Applications Conference (COMPSAC 94)*, pages 418–425. IEEE Computer Society Press, 1994.

[11] Chrystopher L. Nehaniv. Algebra and formal models of understanding. In Masami Ito, editor, *Semigroups, Formal Languages and Computer Systems*, volume 960, pages 145–154. Kyoto Research Institute for Mathematics Sciences, RIMS Kokyuroku, August 1996.

[12] Chrystopher L. Nehaniv and John L. Rhodes. The evolution and understanding of hierarchical complexity in biology from an algebraic perspective. *Artificial Life*, 6:45–67, 2000.

[13] John L. Rhodes. *Applications of Automata Theory and Algebra with the Mathematical Theory of Complexity to Finite-State Physics, Biology, Philosophy, Games, and Codes*. book submitted for publication.

[14] M. P. Schützenberger. On finite monoids having only trivial subgroups. *Information and Control*, 8:190–194, 1965.

[15] H. J. Shyr. *Free monoids and languages*. Hon Min Book Company, Taichung, Taiwan, 2001.

[16] H. Paul Zeiger. Cascade synthesis of finite state machines. *Information and Control*, 10:419–433, 1967. plus erratum.

[17] H. Paul Zeiger. Yet another proof of the cascade decomposition theorem for finite automata. *Math. Systems Theory*, 1:225–228, 1967. plus erratum.