Open access • Book Chapter • DOI:10.1007/978-3-642-13754-9_12

On the Krohn-Rhodes cascaded decomposition theorem — Source link

Oded Maler

Institutions: Centre national de la recherche scientifique

Published on: 01 Jan 2010

Topics: Deterministic automaton, Two-way deterministic finite automaton, Timed automaton, Büchi automaton and ω-automaton

On the Krohn-Rhodes Cascaded Decomposition Theorem

Oded Maler

CNRS-VERIMAG, 2 Av. de Vignate
38610 Gières, France

Dedicated to the memory of Amir Pnueli, deeply missed.

Abstract. The Krohn-Rhodes theorem states that any deterministic automaton is a homomorphic image of a cascade of very simple automata which realize either resets or permutations. Moreover, if the automaton is counter-free, only reset automata are needed. In this paper we give a very constructive proof of a variant of this theorem due to Eilenberg.

1 Introduction

More than 20 years ago my PhD advisor Amir Pnueli convinced me to postpone some dead-ends I was pursuing around the middle of my thesis and look at the Krohn-Rhodes decomposition theorem. His correct intuition was that this theorem can help in establishing a lower-complexity translation from automata to temporal logic. The best known complexity at that time was non-elementary [6], based on a series of transformation adapted from the monograph by McNaughton and Papert [9] which dealt with different characterizations (logical, algebraic, language-theoretic, automatic) of the same class of objects, the *star-free regular sets* [13].

The result of Kenneth Krohn and John Rhodes, announced almost 50 years ago [12, 5], states that any *deterministic* automaton can be decomposed into a cascade of simple automata, whose structure reflects the algebraic structure of the transformation semigroup associated with the automaton. For some time in the 60s and 70s, their theorem, which got them 2 simultaneous PhD titles from Harvard and MIT, respectively, was considered to be a cornerstone of automata theory. When I started to look at at the topic in the late 80s the results have been practically forgotten in the Computer Science mainstream, excluding some specialized islands.

Although I started to acquaint myself with the algebraic (and French) vocabulary of transformation semigroups, it was not easy for me to understand the purely-algebraic versions of the theorem expressed in terms of *wreath product* of semigroups or groups. Fortunately, the book of Ginzburg [2] gave a clear automata-theoretic presentation from which one could understand that a cascade of automata is a particular type of composition where the automata are ordered and each automaton reads the common input alphabet *and* the states of its preceding automata or, equivalently, the *output* of its predecessor, as illustrated in Fig. 1. The theorem states that any automaton, up to homomorphism, can be realized by a cascade of elementary automata of two types, *permutation* *automata* where each letter induces a permutation of the state space, and *reset* automata where each letter is either a reset (sends all states to a fixed single state) or an identity (a self-loop from every state). However, as noted in the last paragraph of [2] *"Finally, notice that the above theory does not indicate how many particular basic building blocks are needed to construct a cascade product covering of a given semiautomaton."*

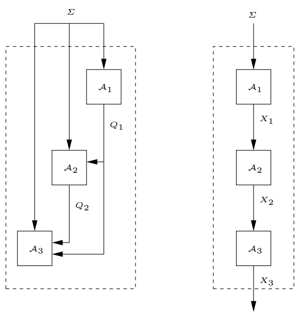


Fig. 1. A cascade product of 3 automata using the state-as-input convention (left) and the input-output convention (right).

I had the privilege to discuss the topic with the late Marcel-Paul Schützenberger who encouraged me to look at the *holonomy* decomposition theorem, a variant of the Krohn-Rhodes theorem, written on eight dense pages of volume B of Eilenberg's book [1] to which he contributed. It took me a *long* time to decipher this motivation-less algebraic prose, translate the construction to my own automata-theoretic language, and verify that it is indeed exponential. From there an exponential translation from counter-free automata to temporal logic, very similar to the construction of Meyer [10] for star-free regular expressions, followed immediately. We also managed to give a lower bound on the size of the decomposition, obtained via a bounded two-way counter, also known as the *elevator* automaton. Apart from a short abstract [7] and a draft on the web [8] we have not published this work, which is what I intend to do presently. Unfortunately due to timing constraints the presentation is not complete, including only the reconstruction of the holonomy decomposition without the lower bound and the translation to temporal logic. The interested reader is referred to [7, 8] for those.

The rest of the paper is organized as follows. In Section 2 we give the basic definitions concerning the algebraic theory of automata and semigroups. In section 3 we define the cascade product and state the theorem. Section 4 is devoted to the study of a particular structure, the holonomy tree, tightly related to a cascaded decomposition. It is a combination of a tree whose nodes are labeled by subsets of the states of the automaton, and on which a transition function, satisfying certain constraints is defined. After establishing the close relationship between such a tree and a cascaded decomposition we describe in Section 5 an algorithm for computing the tree and thus completing a constructive version of the proof. The subtle part in these two sections is how to avoid introducing *spurious permutations* and to assure that if the automaton is counter-free the decomposition will consist exclusively of reset automata.

2 Preliminaries

A total function f from X to Y is an *injection* if $f(x) \neq f(x')$ whenever $x \neq x'$. It is a *surjection* if $\forall y \exists x \; f(x) = y$ and a *bijection* if it is both an injection and a surjection. The latter case implies $|X| = |Y|$ as well as the existence of an inverse function $f' : Y \to X$.

2.1 Automata

We assume familiarity with finite automata and regular sets at the level of [4]. We use Σ^* to denote the set of finite sequences over an alphabet Σ and use ϵ for the empty sequence.

Definition 1 (Automaton). *A deterministic automaton is triple $\mathcal{A} = (\Sigma, Q, \delta)$ where Σ is a finite set of symbols called the input alphabet, Q is a finite set of states and $\delta : Q \times \Sigma \to Q$ is the transition function. A partial automaton is such where δ may be undefined for some combinations of states and symbols.*

The transition function can be lifted naturally to sets of states, by letting $\delta(P, \sigma) = \{\delta(q, \sigma) : q \in P\}$, and to input sequences, by letting $\delta(q, w\sigma) = \delta(\delta(q, w), \sigma)$.

An automaton can be made an *acceptor* by choosing an initial state $q_0 \in Q$ and a set of accepting states $F \subseteq Q$. As such it accepts/recognizes a set of sequences, also known as a *language*, defined as $L(\mathcal{A}) = \{w : \delta(q_0, w) \in F\}$. Kleene's Theorem states that the class of languages recognizable by finite automata coincides with the *regular* languages. A subclass of the regular sets is the class of *star-free* sets defined as:

Definition 2 (Star-Free Regular Sets). *The class of star-free regular sets over Σ is the smallest class containing Σ^* and the sets of the form $\{\sigma\}$ where $\sigma \in \Sigma \cup \{\epsilon\}$, which is closed under finitely many applications of concatenation and Boolean operations.*

Star-free sets have additional characterizations to be discussed in the sequel.

Definition 3 (Automaton Homomorphism). *A surjection $\varphi : Q \to Q'$ is an automaton homomorphism from $\mathcal{A} = (\Sigma, Q, \delta)$ to $\mathcal{A}' = (\Sigma, Q', \delta')$ if for every $q \in Q$, $\sigma \in \Sigma$*

$$\varphi(\delta(q, \sigma)) = \delta'(\varphi(q), \sigma)$$

In such a case we say that $\mathcal{A}'$ is homomorphic to $\mathcal{A}$ and denote it by $\mathcal{A}' \leq_\varphi \mathcal{A}$. When φ is a bijection, $\mathcal{A}$ and $\mathcal{A}'$ are said to be isomorphic.

Intuitively $\mathcal{A}' \leq_\varphi \mathcal{A}$ means that $\mathcal{A}'$ is an abstraction of $\mathcal{A}$ and anything that can be expressed using $\mathcal{A}'$ can be expressed, possibly in more detail using $\mathcal{A}$. Homomorphism is transitive and induces a partial-order relation among automata.

2.2 Semigroups

The theory of automata is strongly related to the algebraic theory of *semigroups* dealing with sets closed under an associative (but not necessarily invertible) binary operation. Two typical examples of semigroups are *sequences* of symbols under the *concatenation* operation and *transformations* (functions from a set to itself) under *function composition*. In fact, the theory of formal languages and automata is, to some extent, a theory about the relation between these two semigroups.

Definition 4 (Semigroups, Monoids and Groups). *A Semigroup is a pair $(S, \cdot)$ where S is a set and $\cdot$ is a binary associative operation ("multiplication") from $S \times S$ to S. A Monoid $(S, \cdot, 1)$ is a semigroup admitting an identity element 1 such that $s \cdot 1 = 1 \cdot s = s$ for every $s \in S$. A group is a monoid such that for every $s \in S$ there exists an element $s^{-1} \in S$ (an inverse) such that $s \cdot s^{-1} = 1$.*

Definition 5 (Subsemigroups, Generators). *A subsemigroup T of S is a subset $T \subseteq S$ which is closed under multiplication, that is, $T^2 \subseteq T$. A subgroup of S is a subsemigroup which is a group. The smallest subsemigroup of S containing a subset $A \subseteq S$ is denoted by A^+ and it consists of all elements of S obtained by finitely many products of elements of A. Any subset $A \subseteq S$ such that $A^+ = S$ is called a generating set of S.*

A finite semigroup can be described by its multiplication table. The trivial semigroup consisting of the singleton set $\{e\}$ is of course a monoid and a group. In the sequel we will not make a distinction between a semigroup and a monoid. As with automata, one can define semigroup homomorphism which is transitive and corresponds to the intuitive notions of refinement/abstraction among structures.

Definition 6 (Semigroup Homomorphisms). *A surjective function $\varphi : S \to S'$ is a semigroup homomorphism from $(S, \cdot)$ to $(S', *)$ if for every $s_1, s_2 \in S$,*

$$\varphi(s_1 \cdot s_2) = \varphi(s_1) * \varphi(s_2)$$

In such a case we say that S' is homomorphic to S and denote it by $S' \leq_\varphi S$. Two mutually homomorphic semigroups are said to be isomorphic.

Let $\text{TR}(Q)$ be the set of all total functions (*transformations*) of the form $s : Q \to Q$ over a finite set Q, $|Q| = n$. One can see that $\text{TR}(Q)$ is a monoid of n^n elements under the operation of *function composition* defined as $s \cdot t(q) = t(s(q))$ for every $q \in Q$. The identity function on Q, 1_Q, is the identity element of $\text{TR}(Q)$. A transformation can be represented as an n-tuple $(q_{i_1}, \ldots, q_{i_n})$ where $q_{i_j} = s(q_j)$.
Remark: There is some conflict between algebraic, functional, and automata-theoretic notational conventions. Algebraically, the *action* of s on q is denoted by qs and the associativity of composition is expressed as $(qs)t = q(s \cdot t)$. On the other hand, the automata-theoretic notation $\delta(q, s)$ is preferable when we have to refer to *several* transition functions. We will try not to confuse the reader.

Definition 7 (Transformation Semigroups). *A transformation semigroup is a pair $X = (Q, S)$ where Q is the underlying set and S is a subsemigroup of $\text{TR}(Q)$, that is, a set of transformations on Q closed under composition. Clearly if Q is finite, so is S.*

The importance of transformation semigroups as more concrete representations of abstract semigroups comes from the following theorem:

Theorem 1 (Cayley). *Every semigroup is isomorphic to a transformation semigroup.*

On the other hand, every automaton gives rise to a transformation semigroup $X_{\mathcal{A}}$ whose generators are the transformations $\{s_\sigma\}_{\sigma\in\Sigma}$ induced by input letters. The following definition gives an intermediate representation of this semigroup.

Definition 8 (Expanded Automaton). *Let $\mathcal{A} = (\Sigma, Q, \delta)$ be an automaton and let $X_{\mathcal{A}} = (Q, S)$ be its transformation semigroup. The expansion of $\mathcal{A}$ is the automaton $\hat{\mathcal{A}} = (S, Q, \delta)$ with $\delta(q, s) = q \cdot s$.*

It can be shown that the existence of a homomorphism between two automata implies the existence of a homomorphism between their corresponding transformation semigroups. On the other hand, a homomorphism from $X = (Q, S)$ to $X' = (Q', S')$ can be obtained without an automaton state-homomorphism, just by taking $Q' \subseteq Q$ and letting S' be the set of transformation on Q' obtained from transformations in S by projection (which consitutes the semigroup homomorphism from S to S'). Mechanically this semigroup can be computed by constructing $\hat{\mathcal{A}} = (S, Q, \delta)$ and then restricting it to Q' and to an alphabet $S' \subseteq S$ consisting of all transformations satisfying $\delta(Q', s) \subseteq Q'$.

Definition 9 (Rank). *The rank of a transformation $s \in \mathrm{TR}(Q)$ is defined as the cardinality of its range $Qs = \{qs : q \in Q\}$.*

Permutations and resets (see Fig. 2) represent two extreme types of transformations in terms of rank. The $n!$ permutations are those in which the domain and the range coincide and the rank is n while the n resets are the constant transformations of rank 1. It is worth looking at the effect of resets and permutations from the following angle,

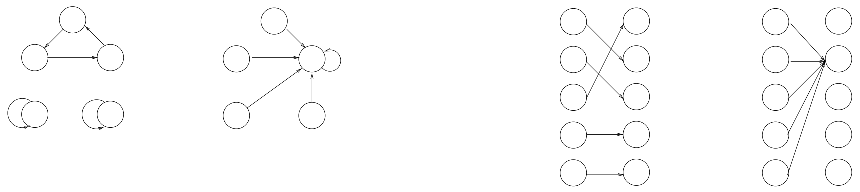


Fig. 2. A permutation and a reset illustrated as transition graphs (left) and as transformations (right).

emphasizing what is known about the state of the automaton upon the occurrence of a generic transition $q' = \delta(q, \sigma)$. If σ is a reset we do not need to know q in order to determine q', however knowing q' we *cannot* determine q. On the other hand if σ is a permutation we know nothing about q' if we do not know what q was, but if we know q', q is uniquely determined. In other words, a permutation is reverse-deterministic, while in resets the degree of reverse non-determinism is maximal.

Permutations and resets are closed under composition or more precisely, if we denote a reset by R and a permutation by P we get the following multiplication table:

$*$	P	R
P	P	R
R	R	R

Resets can be obtained by composing non-reset transformations, for example, $(122) \cdot (223) = (222)$, because composition can decrease the rank. On the other hand, because composition cannot increase the rank, a permutation on Q cannot be composed from non-permutations on Q. However a permutation on a subset $R \subseteq Q$ can be composed from non-permutations as can be seen from Fact 1.

Fact 1 *A transformation s permutes a subset $R \subseteq Q$ iff $s = s_1 \cdots s_m$ for some $m > 0$ and there exists a sequence of subsets $\{R_j\}_{j=0..m}$ such that $R_0 = R_m = R$ and the restriction of every s_j to R_j is an injection to R_{j+1}.*

There are various ways to classify finite semigroups and their corresponding automata and regular sets [11]. An important sub-class of semigroups is defined as follows:

Definition 10 (Group-Free Semigroups). *A semigroup S is aperiodic if there exists a number k such that $s^k = s^{k+1}$ for every element $s \in S$. A semigroup is group-free if it has no non-trivial subgroups. An automaton is counter-free if no word induces a permutation other than identity on any subset of Q.*

A semigroup is aperiodic iff it is group-free and an automaton is counter-free iff its transformation semigroup is group-free. The following theorem relates these objects to star-free sets and, consequently, to propositional temporal logic.

Theorem 2 (Schützenberger). *A regular set U is star-free if and only if its syntactic semigroup is aperiodic.*

The syntactic semigroup of a language is the transformation semigroup of the minimal *deterministic* automaton which recognizes it. This automaton is unique and, following the theorem, it is counter-free if the language is star-free.

3 The Krohn-Rhodes Primary Decomposition Theorem

The definition of the cascade product of two or more automata is given below:

Definition 11 (Cascade Product). *Let $\mathcal{B}_1 = (\Sigma, Q_1, \delta_1)$ be an automaton, and let $\mathcal{B}_2 = (Q_1 \times \Sigma, Q_2, \delta_2)$ be a (possibly partial) automaton such that for every $q_1 \in Q_1$ and $q_2 \in Q_2$, either $\delta_2(q_2, \langle q_1, \sigma\rangle)$ is defined for every $\sigma \in \Sigma$ or it is undefined for every σ. The cascade product $\mathcal{B}_1 \circ \mathcal{B}_2$ is the automaton $\mathcal{C} = (\Sigma, P, \bar{\delta})$ where*

$$P = \{(q_1, q_2) : \delta_2(q_2, \langle q_1, \sigma\rangle) \text{ is defined}\}$$

and

$$\bar{\delta}(\langle q_1, q_2\rangle, \sigma) = (\delta_1(q_1, \sigma), \delta_2(q_2, \langle q_1, \sigma\rangle)).$$

The cascade product of more than two automata is defined as

$$\mathcal{B}_1 \circ \mathcal{B}_2, \ldots \circ \mathcal{B}_k = (\ldots((\mathcal{B}_1 \circ \mathcal{B}_2) \circ \mathcal{B}_3 \ldots) \circ \mathcal{B}_k.$$

Fig. 3 shows a cascade product of two automata. Note that the communication links between $\mathcal{B}_1$ and $\mathcal{B}_2$ are given implicitly via the definition of the input alphabet of $\mathcal{B}_2$ as the product of Σ and the state-space of $\mathcal{B}_1$. A alternative definition using transducers (Mealy machines) is possible as illustrated in Fig. 1.

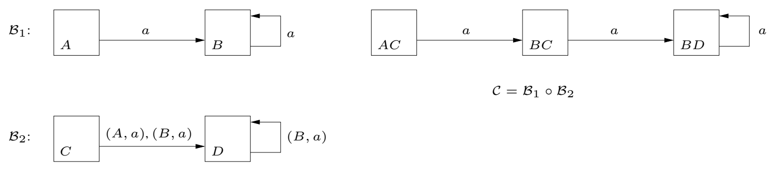


Fig. 3. A cascade $\mathcal{C} = \mathcal{B}_1 \circ \mathcal{B}_2$.

Definition 12 (Permutation-Reset Automata). *A (potentially partial) automaton $\mathcal{A} = (\Sigma, Q, \delta)$ is a permutation-reset automaton if for every letter $\sigma \in \Sigma$, σ is either a permutation or reset with respect to the set of states on which it is defined.*[1] *If the only permutations are identities, we call it a reset automaton.*

The Krohn-Rhodes theorem states that one can realize any automaton (up to homomorphism) as a cascade of permutation-reset automata and that non-trivial permutations are required only if the transformation semigroup of the automaton admits non-trivial subgroups. Based on the Jordan-Hölder Theorem, the groups can be decomposed further into a cascade of *simple* groups but we will not be concerned much with the group part of the theorem beyond guaranteeing that it vanishes for counter-free automata. The original formulation of the theorem was stated in terms of semigroups and its automata-theoretic version can be phrased as follows.

Theorem 3 (Krohn-Rhodes: Automata). *For every automaton $\mathcal{A}$ there exists a cascade $\mathcal{C} = \mathcal{B}_1 \circ \mathcal{B}_2 \circ \cdots \circ \mathcal{B}_k$ such that:*

1. *Each $\mathcal{B}_i$ is a permutation-reset automaton;*
2. *There is a homomorphism φ from $\mathcal{C}$ to $\mathcal{A}$;*
3. *Any permutation group in some $\mathcal{B}_i$ is homomorphic to a subgroup of the transformation semigroup of $\mathcal{A}$.*

The pair $(\mathcal{C}, \varphi)$ is called a cascaded decomposition of $\mathcal{A}$.

[1] Partial resets and partial permutations can be completed to full ones by appropriately defining the missing transitions.

The third condition implies that if $\mathcal{A}$ is counter-free then each $\mathcal{B}_i$ is a reset automaton. It is this theorem that we are going to prove in constructive detail in the sequel. We sometimes assume an additional trivial one-state automaton $\mathcal{B}_0$ composed in front of the cascade. We will often use a notation of the form $\langle p, q_i \rangle$ for $\langle q_1, q_2, \ldots, q_{i-1}, q_i \rangle$.

4 Structures Associated with a Cascade

Let $\mathcal{C} = (\Sigma, P, \bar{\delta}) = \mathcal{B}_1 \circ \cdots \circ \mathcal{B}_k$ be a cascade, and let φ be a homomorphism from $\mathcal{C}$ to an automaton $\mathcal{A} = (\Sigma, Q, \delta)$. Let $\mathcal{C}_i = (\Sigma, P_i, \bar{\delta}_i) = \mathcal{B}_1 \circ \cdots \circ \mathcal{B}_i$ be the product of the first i components, $i \leq k$. Elements of P_i are called i-configurations and they admit a natural hierarchical structure, the *configuration tree*, where each i-configuration $p_i = \langle p_{i-1}, q_i \rangle$ is an extension of a *parent* configuration p_{i-1}. We associate a family of mappings $\varphi_i : P_i \to 2^Q$ indicating for each configuration which states of Q are encoded by its extensions, that is,

$$\varphi_k(p) = \{\varphi(p)\}$$

and

$$\varphi_{i-1}(p) = \bigcup_{\langle p,q \rangle \in P_i} \varphi_i(\langle p, q \rangle).$$

A decomposition is *redundant* if there are two i-configurations $\langle p, q \rangle$ and $\langle p, q' \rangle$ such that $\varphi_i(\langle p, q' \rangle) \subseteq \varphi_i(\langle p, q \rangle)$. In this case we can remove configuration $\langle p, q' \rangle$ by letting $\delta_i(q', \langle p, \sigma \rangle)$ be undefined and redirecting all transitions entering q' to q. The restriction of φ_k to the remaining configurations, those which are not extensions of $\langle p, q' \rangle$, still covers the whole Q and is a homomorphism. Repeating this procedure until all redundancies are removed we can conclude that the existence of a decomposition is equivalent to the existence of a non-redundant decomposition.

The hierarchical relation between cascade configurations and subsets of Q motivates the following definition.

Definition 13 (Subset Transition Tree). *A subset transition tree (STT) for an automaton $\mathcal{A} = (\Sigma, Q, \delta)$ is a tuple $\mathcal{T} = (\Sigma, M, \Delta, \pi, \phi)$ where:*

$M = M_0 \uplus \ldots \uplus M_k$ is a set of nodes partitioned into levels, with $M_0 = \{m_*\}$;
- *$\Delta = \Delta_0 \uplus \cdots \uplus \Delta_k$ is a transition function with $\Delta_i : M_i \times \Sigma \to M_i$. Each level can be viewed as an automaton $N_i = (\Sigma, M_i, \Delta_i)$;*
- *$\pi : M - \{m_*\} \to M$ is a parenthood function, mapping every element of M_i to an element of M_{i-1}. We use Π_m to denote the set of sons of a node m, whose elements are called brothers;*
- *The transition function is ancestor-preserving: $\pi(\Delta(m, \sigma) = \Delta(\pi(m), \sigma)$;*
- *The action of every letter on any set Π_m of brothers is either a reset or an injection.*
- *$\phi : M \to 2^Q$ is a function mapping nodes to sets of states whose restriction to M_i is denoted by ϕ_i and which satisfies:*
  - *ϕ_k maps the leaves of the tree to singletons and constitutes a homomorphism from $N_k = (\Sigma, M_k, \Delta_k)$ to $\mathcal{A}$;*

- *For every $i < k$*
$$\phi_i(m) = \bigcup_{m' \in \Pi_m} \phi_{i+1}(m').$$
- *No redundancy: $\phi(m) \not\subseteq \phi(m')$ for any pair of brothers.*

Next we prove a weak version of the fundamental fact underlying the decomposition. It is weak because it speaks of a decomposition satisfying only conditions 1-2 of Theorem 3.

Proposition 1. *There exists a cascade decomposition $\mathcal{C} = \mathcal{B}_1 \circ \cdots \circ \mathcal{B}_k \leq_\varphi \mathcal{A}$ with each $\mathcal{B}_i$ being a permutation-reset automaton, iff there exists an STT $\mathcal{T}$ for $\mathcal{A}$ which is isomorphic to the configuration tree.*

Proof. The construction of the STT from the configuration tree of the cascade is straightforward, obtained by letting $M_i = P_i$ and $\Delta_i = \bar{\delta}_i$. The mapping of nodes to states of $\mathcal{A}$ is defined by the encoding, that is, $\phi_i = \varphi_i$, and parenthood is defined naturally as $\pi(\langle p, q\rangle) = p$. The fact that letters induce injections and resets on brothers is obvious and ancestor-preservation follows from:

$$\pi(\bar{\delta}_i(\langle p, q\rangle, \sigma)) = \pi(\langle \bar{\delta}_{i-1}(p, \sigma), \delta_i(q, \langle p, \sigma\rangle)\rangle) = \bar{\delta}_{i-1}(p, \sigma) = \bar{\delta}_{i-1}(\pi(\langle p, \sigma\rangle), \sigma).$$

For the other direction we need to show how to build a cascade from $\mathcal{T}$. Let

$$d_i = \max\{|\Pi_m| : m \in M_{i-1}\}$$

be the size of the largest set of brothers at level i and let $Q_i = \{q_1, \ldots, q_{d_i}\}$. We define for every i a mapping $\theta_i : M_i \to Q_i$ whose restriction to any set of brothers is an injection. This encoding induces a bijection $\psi : M \to P$, which decomposes into $\psi_0 \uplus \cdots \uplus \psi_k$ with $\psi_i : M_i \to P_i$ defined inductively as $\psi_0(m_*) = \theta_0(m_*)$ and

$$\psi_i(m) = \langle \psi_{i-1}(\pi(m)), \theta_i(m)\rangle.$$

The transition function at each level is defined for every $\langle p, q\rangle \in P_i$ as

$$\delta_i(q, \langle p, \sigma\rangle) = \Delta_i(\psi^{-1}(\langle p, q\rangle), \sigma)$$

All that remains to be shown is that $\varphi : P \to Q$, defined as

$$\varphi(p) = \phi_k(\psi_k^{-1}(p))$$

is a homomorphism and this follows from the fact that ψ_k is an isomorphism between $\mathcal{C}$ and N_k and ϕ is an homomorphism from N_k to $\mathcal{A}$. ∎

The idea of the second direction is rather simple. We want to build a cascade whose configurations encode the subsets corresponding to the nodes of the tree. Each level i can be partitioned into several sets of nodes, each of which consisting of all brothers sharing the same ancestor. Let m and m' be nodes at level $i-1$ and let Π_m and $\Pi_{m'}$ be their respective sets of sons. Since m and m' are already encoded by distinct configurations p and p', their sons can be encoded by extensions of p and p' that use *the same* set of states Q_i whose size is the size of the largest set of brothers at level i. Thus we encode elements of Π_m by configurations in $\{p\} \times Q_i$ and elements of $\Pi_{m'}$ by configurations in $\{p'\} \times Q_i$. The transitions that correspond to the former will be labeled by $\langle p, \sigma \rangle$ and those of the latter by $\langle p', \sigma \rangle$. Doing so, every *injection* induced by some σ on some Π_m becomes a *permutation* induced by $\langle p, \sigma \rangle$ on Q_i. The hard part of the proof of the full theorem is to show that this *injection folding* can be done without creating spurious permutations (not implied by permutation subgroups of the automaton) and, in particular, if $\mathcal{A}$ is non-counting there will be no permutations.

Fig. 4 shows how a particular choice of encoding may lead to the introduction of spurious permutations. Automaton $\mathcal{A}$ is a union of two reset automata, hence clearly counter-free. An STT for $\mathcal{A}$ has an upper level with two nodes m and m' mapped naturally to sets $\{q_1, q_3\}$ and $\{q_2, q_4\}$. The first element in the cascade is the reset automaton $\mathcal{B}_1$ whose states A and B encode, respectively, these two subsets. The choice of the second coordinate makes a difference. Let AC encode q_1 while AD encodes q_3. Then we have two ways to encode q_2 and q_4. Encoding them with BC and BD, respectively, the second element in the cascade is the identity automaton $\mathcal{B}_2$. However, if we choose to encode q_2 by BD and q_4 by BC we obtain $\mathcal{B}'_2$ in which the letter (A, a) induces a non-trivial permutation. As it turns out there is an additional condition on the structure of the STT as well as a general encoding scheme that avoids this phenomenon.

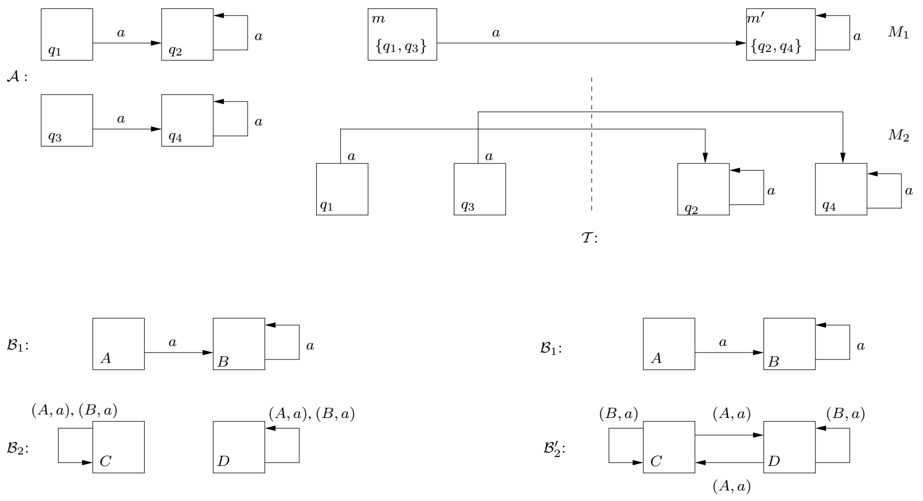


Fig. 4. A counter-free automaton $\mathcal{A}$, an STT with sons of m and m' separated by the dashed line, and two choices of encoding, the second leading to a permutation.

Definition 14 (Equivalence). *Let $\mathcal{A} = (\Sigma, Q, \delta)$ be a (complete) automaton and let $\mathcal{T} = (\Sigma, M, \Delta, \pi, \phi)$ be an STT for $\mathcal{A}$.*

1. *Two subsets $R_1, R_2 \subseteq Q$ are equivalent if there exist $w, w' \in \Sigma^*$ such that*
   (a) $\delta(R_1, w) = R_2$ and $\delta(R_2, w') = R_1$;

   (b) ww' *and* $w'w$ *induce identities on* R_1 *and* R_2*, respectively;*
   This fact is denoted by $R_1 \overset{w,w'}{\sim} R_2$ *or simply* $R_1 \sim R_2$.
2. *Two nodes* $m, m' \in M_i$ *are equivalent if there exist* $w, w' \in \Sigma^*$ *such that:*
   (a) $\Delta(m, w) = m'$ *and* $\Delta(m', w') = m$*;*
   (b) $\phi(m) \overset{w,w'}{\sim} \phi(m')$*, in the sense of subset equivalence;*
   This fact is denoted as well by $m \overset{w,w'}{\sim} m'$ *or simply* $m \sim m'$.

Note that if w and w' satisfy condition 1-(a) but not 1-(b), then there exist some u, u' satisfying the latter. Since ww' is a permutation on R_1 and $w'w$ is a permutation on R_2, there is some l such that $(ww')^l$ and $(w'w)^l$ are identities. By letting $u = w$ and $u' = w'(ww')^{l-1}$ we have $R_1 \overset{u,u'}{\sim} R_2$. Equivalence between nodes implies an additional constraint on the definition of ϕ over their sons.

Proposition 2. *Let* $m \overset{w,w'}{\sim} m'$ *be two equivalent nodes in an STT. Then for every* $r \in \Pi_m$ *there exists* $r' \in \Pi_{m'}$ *such that* $\delta(\phi(r), w) = \phi(r')$*. Consequently* $|\Pi_m| = |\Pi_{m'}|$.

Proof: Suppose, on the contrary, that $\delta(\phi(r), w) \subset \phi(r')$ and hence $\delta(\phi(r'), w') \not\subseteq \phi(r)$ which violates the STT definition. ▟

Definition 15 (Holonomy Tree). *A holonomy tree is an STT* $\mathcal{T} = (\Sigma, M, \Delta, \pi, \phi)$ *such that for every* $m, m' \in M_i$ *and* $\sigma \in \Sigma$*, such that* $\Delta(m, \sigma) = m'$*,* σ *induces an injection*[2] *from* Π_m *to* $\Pi_{m'}$ *only if* $m \sim m'$.

To motivate this definition let us observe that injection folding is necessary in order to transform an injection from Π_m to $\Pi_{m'}$ to a permutation on Q_i. On the other hand, a reset from Π_m to some $r \in \Pi_{m'}$ remains a reset in $\mathcal{B}_i$ even if Π_m and $\Pi_{m'}$ are encoded using different states in $\mathcal{B}_i$. The essence of the additional condition in the holonomy tree is to restrict injection folding to occur only among sons of equivalent nodes where permutations really exist. If m and m' are not equivalent, σ will induce a *reset* from Π_m to $\Pi_{m'}$.

The proof of the equivalence between the existence of a holonomy tree and a cascaded decomposition satisfying condition 3 of Theorem 3 involves the following steps:

1. Associate with every node m in the holonomy tree a (possibly-trivial) permutation group H_m called a holonomy group;
2. Provide an encoding scheme which guarantees that any permutation subgroup in the cascade is isomorphic to a holonomy group;
3. Show that any holonomy group is homomorphic to a subgroup of $X_{\mathcal{A}}$.

Definition 16 (Holonomy Group). *Let* $\mathcal{T} = (\Sigma, M, \Delta, \pi, \phi)$ *be a holonomy tree with* $N_i = (\Sigma, M_i, \Delta_i)$ *being the automaton of level* i *and let* $\hat{N}_i = (S_i, M_i, \Delta_i)$ *be its expansion. The holonomy group* H_m *associated with a node* $m \in M_{i-1}$ *is the restriction of* $\hat{N}_i$ *to* Π_m *and to transformations that induce permutations on it.*

It can be shown that when $m \sim m'$, the groups H_m and $H_{m'}$ are isomorphic. The procedure for state encoding and injection folding for an equivalence class of $\sim$ is given below. The definition is inductive, assuming the encoding of the preceding levels has already been completed.

[2] When $|\Pi_m| = 1$ we view σ as a partial reset, not an injection, and $m \sim m'$ is not required.

Definition 17 (Faithful Injection Folding). *Let $\{m_1, \ldots m_l\}$ be an equivalence class of $\sim$ at level $i-1$ whose elements are encoded by configurations $\{p_1, \ldots, p_l\}$, respectively and that for every j, $m_1 \overset{w_j, w'_j}{\sim} m_j$. Let $\mathcal{M} = \bigcup_j \Pi_{m_j}$ be the set of all sons of these nodes. We will use the set $Q_i = \Pi_{m_1}$ to encode the i^{th} coordinate of all elements of $\mathcal{M}$. The function $\theta : \mathcal{M} \to \Pi_{m_1}$ is defined for every m_j and every $r \in \Pi_{m_j}$ as $\theta(r) = \Delta(m, w'_j)$. Then for every $q \in \Pi_{m_1}$ and for every m_j and $m_{j'}$ such that $\Delta_i(m_j, \sigma) = m_{j'}$ we let*

$$\delta_i(q, \langle p_j, \sigma \rangle) = \Delta_i(q, w_j \sigma w'_{j'}).$$

Proposition 3 (Injection Folding and Holonomy). *For every non-leaf node $m \in M_{i-1}$ the permutation group (induced by letters of the form $\langle p, \sigma \rangle$) in cascade element $\mathcal{B}_i$ constructed according to Definition 17 is isomorphic to H_m.*

Proof: We need to show that each permutation in $\mathcal{B}_i$ is identical to a permutation in H_m and vice versa. One direction follows immediately from the injection folding procedure: the action of $\langle p, \sigma \rangle$ on $Q_i = \Pi_{m_1}$ is defined to be identical to the action of some $w_j \sigma w'_{j'}$ on Π_{m_1}. For the other direction consider a word $u = \sigma_1 \cdot \sigma_2 \cdots \sigma_l$ inducing a cycle from $m = m_1$ to itself passing through nodes $m_2, m_3, \ldots, m_l$. Since each $w'_j w_j$ induces an identity on Π_{m_j}, the word

$$u' = \sigma_1 \cdot w'_2 \cdot w_2 \cdot \sigma_2 \cdot w'_3 \cdot w_3 \cdots \sigma_l$$

induces the same permutation on Π_m as does u. All the remains to be shown is that the word

$$\langle p_1, \sigma_1 \rangle \cdot \langle p_2, \sigma_2 \rangle \cdots$$

induces the same permutation on Π_m as does u' and this follows from defining the action of $\langle p_j, \sigma_j \rangle$ in $\mathcal{A}_i$ to be identical to that of $w_j \sigma_j w'_{j+1}$ in N_i. ▪

Proposition 4 (Holonomy and Subgroups). *The holonomy group H_m is homomorphic to the subgroup of $X_{\mathcal{A}}$ associated with $\phi(m)$.*

Proof: We need to show how to map a permutation $s : \phi(m) \to \phi(m)$ to a permutation $s' : \Pi_m \to \Pi_m$. This is done by letting, for every $r \in \Pi_m$

$$r \cdot s' = \phi^{-1}(\delta(\phi(r), s)),$$

that is, s is applied to the subset $\phi(r)$ associated with r and the resulting set is decoded back into an element of Π_m. The fact that ϕ^{-1} exists and is unique follows from Proposition 2. ▪

To complete the construction of the cascade from the holonomy tree we just need to partition every level in the tree into equivalence classes of $\sim$, build a cascade component with states corresponding to each class and apply to each equivalence class the procedure of Definition 17.

Corollary 1. *There exists a cascaded decomposition $\mathcal{C} = \mathcal{B}_1 \circ \cdots \circ \mathcal{B}_k \leq_\varphi \mathcal{A}$, with each $\mathcal{B}_i$ being a permutation-reset automaton and each permutation group homomorphic to a subgroup of $X_{\mathcal{A}}$, iff there exists a holonomy tree $\mathcal{T}$ for $\mathcal{A}$ isomorphic to the configuration tree.*

5 A Decomposition Algorithm

In this section we show how to build for an automaton $\mathcal{A}$ a holonomy tree whose size is at most exponential in the size of $\mathcal{A}$. The procedure involves the following steps (see Fig. 6):

1. Construct from $\mathcal{A}$ a *tree subset automaton* (TSA). This construction which is similar to the famous subset construction, computes all the subsets reachable from Q, that is, $\{\delta(Q, w) : w \in \Sigma^*\}$, plus the singleton sets which are not reachable from Q. In addition the TSA admits a parenthood function and in order to make its transition function ancestor-preserving, some reachable subsets will be represented by more than one node in the tree;
2. Compute a *height* function over the nodes and rearrange the tree into levels according to the height;
3. Complete the levels by duplicating nodes and redirect transitions to make each level a complete automaton.

It will then remain to show that a holonomy tree is obtained, from which the cascaded decomposition follows. The subtle point is the rearrangement of the tree into levels so as to restrict injections to occur only among sons of equivalent nodes. For a parenthood function π we let $\pi^0(m) = m$ and $\pi^j(m) = \pi(\pi^{j-1}(m))$. We use Π_m, Π^*_m and $\pi^*(m)$, respectively, do denote sons, descendants and ancestors of m.

Definition 18 (Tree Subset Automaton). *A tree subset automaton (TSA) for an automaton $\mathcal{A} = (\Sigma, Q, \delta)$ is a tuple $\mathcal{T} = (\Sigma, M, \Delta, \pi, \phi)$ where:*

- *M is a set of nodes with a distinguished root element m_*;*
- *$\pi : M - \{m_*\} \rightarrow M$ is a parenthood function, such that for every $m \neq m_*$, there exists some $j > 0$ such that $\pi^j(m) = m_*$;*
- *$\Delta : M \times \Sigma \rightarrow M$ is an ancestor-preserving transition function, that is, for every m, σ there is some j such that $\Delta(\pi(m), \sigma) = \pi^j(\Delta(m, \sigma))$;*
- *$\phi : M \rightarrow 2^Q$ is a function mapping nodes to sets of states, satisfying*
  - *The range of ϕ is $\{\delta(Q, w) : w \in \Sigma^*\} \cup \{\{q\} : q \in Q\}$;*
  - *$\phi(m_*) = Q$;*
  - *$\phi(\Delta(m, \sigma)) = \delta(\phi(m), \sigma)$*
  - *$\phi(m) \subseteq \phi(\pi(m))$;*
  - *No redundancy: $\phi(m) \not\subseteq \phi(m')$ for any pair of brothers.*

Algorithm A-TSA for constructing a TSA is depicted in Table 1. It is a typical on-the-fly graph exploration algorithm which uses an auxiliary list L of newly-discovered nodes (those for which the transition function has not yet been computed). The algorithm works in two phases: first it computes nodes that correspond to sets of the form $\delta(Q, w)$, determines their respective parents and computes Δ for them in an ancestor-preserving manner. The determination of the parent for a newly-created node r' is illustrated in Fig. 5. Note that $m' \in \Pi^*_{m'}$ and $F = \delta(\phi(r), \sigma) \subseteq \phi(m')$ so that a node $z \in \Pi^*_{m'}$ satisfying $F = \phi(r') \subseteq \phi(z)$ always exists. Note also that z may be non-unique if m' has two incomparable descendants whose sets contain F and in this case an arbitrary choice of a parent can be made. In the second phase of the algorithm we add to the tree all the remaining *singletons* by adding to each node m sons that correspond to singleton nodes not covered by the union of its existing sons. Then we compute $\mathit{\Delta}$ for the newly-added nodes according to the same principle.

```
**Algorithm** A-TSA

$M := L := \{m_*\}$; $\phi(m_*) := Q$

repeat  pick $r \in L$, with $\pi(r) = m$
  for every $\sigma \in \Sigma$
    $F := \delta(\phi(r), \sigma)$
    $m' = \Delta(m, \sigma)$
    if $\neg\exists r' \in M$ s.t. $\phi(r') = F$ and $\pi(r') \in \Pi^*_{m'}$
      create a node $r'$ with $\phi(r') = F$ and insert it to $L$ and $M$
      let $\pi(r')$ be a minimal $z \in \Pi^*_{m'}$ s.t. $F \subseteq \phi(z)$
      for every node $z'$ s.t. $\pi(z') = z$ and $F \subseteq \phi(z')$
        $\pi(z') := r'$
    endif
    $\Delta(r, \sigma) := r'$
  remove $r$ from $L$
until $L = \emptyset$

for every $m \in M$
  for every $q \in \phi(m) - \bigcup_{r \in \Pi_m} \phi(r)$
  insert a new node $r$ to $M$ and $L$
  $\phi(r) := \{q\}$; $\pi(r) := m$
repeat
  take $r \in L$, $m = \pi(r)$
    for every $\sigma \in \Sigma$
      $F := \delta(\phi(r), \sigma)$
      $m' = \Delta(m, \sigma)$
      Let $r'$ be a node with $\phi(r') = F$ and $\pi(r') \in \Pi^*_{m'}$
      $\Delta(r, \sigma) := r'$
  remove $r$ from $L$
until $L = \emptyset$
```

Table 1. The TSA Construction Algorithm

Proposition 5. *Algorithm* A-TSA *terminates and produces a TSA for $\mathcal{A}$.*

Proof: All sets of the form $\delta(Q, w)$ as well as the other singletons are eventually covered by nodes and ancestor preservation is guaranteed by construction. ▟

The next step involves the rearrangement of the nodes into levels according to a *height function* that we define below. Note that the definition of node equivalence (Def-

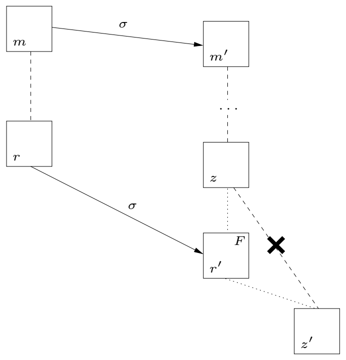


Fig. 5. Determining the parent of r' to be a minimal element of $\Pi^*_{m'}$ which contains $\phi(r')$; Redirecting the parenthood of z' from z to r'.

inition 14) holds also for TSA, and that apart from transitions among members of an equivalence class of $\sim$, the transition graph of the TSA is acyclic.

Definition 19 (Height). *A height function for a TSA $\mathcal{T} = (\Sigma, M, \Delta, \pi, \phi)$ is a function $h : M \to \mathbb{N}$ defined inductively as*

$$h(m) = 0 \ \textit{ if } \ |\phi(m)| = 1$$

$$h(m) = \max \left\{ \begin{array}{l} \max\{h(r) + 1 : r \in \Pi_m\}, \\ \max\{h(r) + 1 : \exists\sigma\Delta(m, \sigma) = r \not\sim m\}, \\ \max\{h(m') : m \sim m'\} \end{array} \right\}$$

In other words, $h(m)$ is the length of the longest path from m to a singleton node, not counting transitions among equivalent nodes. The height function can be computed polynomially using a shortest-path algorithm variant. After computing the height we partition M into $M_0 \uplus \ldots \uplus M_k$ with $k = h(m_*)$ and $M_i = \{m \in M : h(m) = k - i\}$.

The next step is to transform $\mathcal{T} = (\Sigma, M, \Delta, \pi, \phi)$ into a *balanced* TSA $\mathcal{T}' = (\Sigma, M', \Delta', \pi', \phi')$ in which all the ancestral chains from a singleton to m_* are of the same length. The completion of each level with missing nodes is performed bottom up by letting $M'_k = M_k$ and then computing M'_i based on M'_{i+1} as follows. For every $r \in M'_{i+1}$ such that $\pi(r) \notin M_i$ we create a new node $m \in M'_i$ and let $\phi'(m) = \phi(r)$, $\pi'(m) = \pi(r)$, $\pi'(r) = m$ and $\Delta'(m, \sigma) = \Delta(r, \sigma)$ for every σ. The mapping of existing nodes remains the same, that is, $\phi'(m) = \phi(m)$ when $m \in M$. As a result of this procedure each node has an ancestor (possibly identical to itself) in every level. The final step which transforms $\mathcal{T}'$ into a holonomy tree consists of lifting transitions that go from a node m to a lower-level node m' so that they preserve the level. In other words, for every i, $m \in M_i$ and σ we let $\Delta_i(m, \sigma) = m''$ where m'' is the *ancestor* of $m' = \Delta(m, \sigma)$ at M_i. The whole procedure is demonstrated in Fig. 6-(a,b,c,d).

To prove that we obtain a holonomy tree we need to show that for every two nodes m and m', a letter induces an injection from Π_m to $\Pi_{m'}$ only if $m \sim m'$. Suppose $\Delta(m, \sigma) = m'$ and $m \not\sim m'$, both belonging to level i. This implies that in the TSA

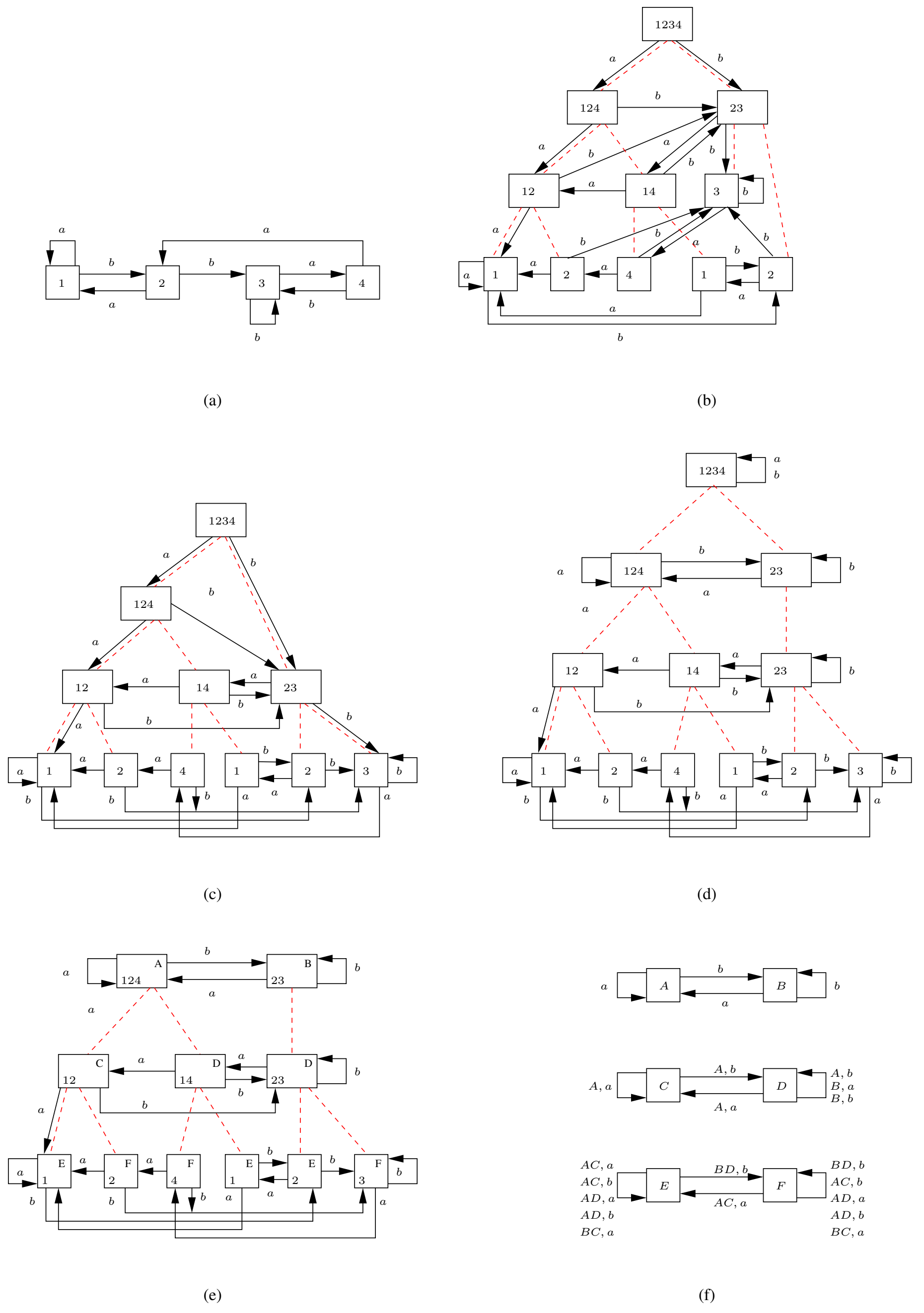


Fig. 6. The decomposition process: (a) An automaton; (b) its TSA (parenthood indicated by dashed lines); (c) the TSA rearranged according to height; (d) the holonomy tree obtained after completion and redirection; (e) state encoding; (f) the decomposition.

there was some node m'' with $h(m'') < h(m)$ such that $\Delta(m, \sigma) = m''$ (Fig. 7-(a)). After the rearrangement and completion procedure, m'' is a node in level $i + 1$ and σ induces a *reset* from Π_m to it (Fig. 7-(b)). The injection at level $i + 2$ has been *separated* into two resets induced by two *distinct* input letters. Fig. 6-(e,f) shows how the holonomy tree is transformed to a cascade via state encoding. The global automaton associated with the cascade and its homomorphism to the original automaton are shown in Fig. 8.

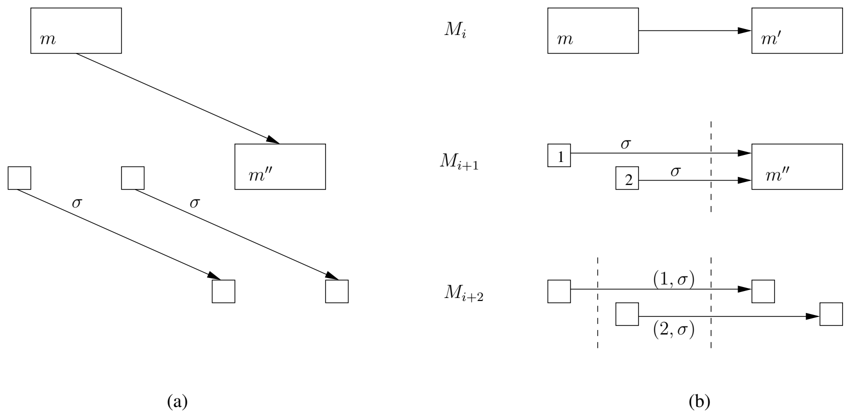


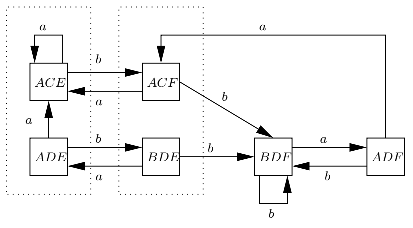
Fig. 7. The crux of the matter: (a) the situation before height rearrangement and completion with an injection between non-equivalent nodes; (b) after the procedure the sons of m make a reset to m'' and the transition functions of their sons are defined over distinct alphabets.

Corollary 2 (Main Result). *Every automaton $\mathcal{A}$ can be decomposed into a cascade of permutation-reset automata, satisfying the conditions of Theorem 3, whose size is at most exponential in the size of $\mathcal{A}$.*

The reason for the exponential blow-up is that to satisfy ancestor-preservation (which is crucial for the hierarchical coordinate system underlying the cascade) some states may need to split to exponentially-many copies, each representing a different class of input histories that leads to the same state. The reader is invited to construct the holonomy tree for the automaton of Fig. 9.

6 Concluding Remarks

Let us sketch the historical roots of this construction. Among the numerous proofs of the Krohn-Rhodes primary decomposition theorem those of Zeiger [15, 16] were more automata oriented. Zeiger's proof has been corrected and presented more clearly in Ginzburg's book [2] based on some constructs attributed to Yoeli [14]. Ginzburg's proof



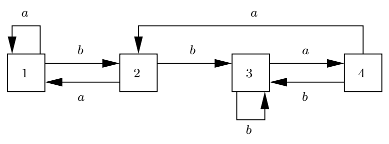


Fig. 8. The global automaton associated with the cascade. The homomorphism to the original automaton is defined as $\varphi(ACE) = \varphi(ADE) = 1$, $\varphi(ACF) = \varphi(BDE) = 2$, $\varphi(BDF) = 3$ and $\varphi(ADF) = 4$.

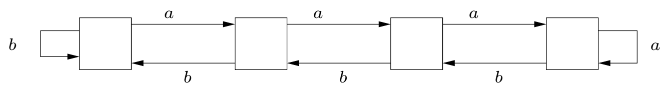


Fig. 9. The elevator automaton, the hardest counter-free automaton to decompose.

of the theorem contains some non-deterministic stages concerning the choice of semi-partitions of Q. In addition, it does not discuss complexity issues explicitly. Another incomplete proof in the same spirit appears in [3].

The proof in [2] inspired Eilenberg to give a slight generalization of the primary decomposition, the *holonomy* decomposition ([1], pp. 43-50). The holonomy decomposition is cleaner and determinizes the choice of semi-partitions. Its major drawback is that it is a theorem on coverings of transformation semigroups and as such it pays no attention to the *labels* of the *generators* of the semigroup, that is, the input alphabet. Consequently, the outcome of the decomposition is not given explicitly as a valid automaton over the *original* alphabet. Another cultural problem associated with this construction is the elegant, concise and motivation-less algebraic style in which it is written, which makes it hardly accessible to many. It remains to be seen if the present exposition improves the situation..

As a final note, since this work has not undergone a complete review process, it probably contains inaccuracies for which I apologize and urge the reader to notify me of. I would like to thank O. Gauwin for proofreading and E. Asarin for helping me to catch up with my former self.

References

1. S. Eilenberg. *Automata, Languages, and Machines*. Academic Press, 1976.
2. A. Ginzburg. *Algebraic Theory of Automata*. Academic Press, 1968.
3. J. Hartmanis and R.E. Stearns. *Algebraic Structure Theory of Sequential Machines*. Prentice-Hall, 1966.
4. J.E. Hopcroft and J.D. Ullman. *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley, 1979.

5. K. Krohn and J. Rhodes. Algebraic theory of machines. I. Prime decomposition theorem for finite semigroups and machines. *Transactions of the American Mathematical Society*, 116:450–464, 1965.
6. O. Lichtenstein, A. Pnueli, and L.D. Zuck. The glory of the past. In *Logic of Programs*, volume 193 of *LNCS*, pages 196–218, 1985.
7. O. Maler and A. Pnueli. Tight bounds on the complexity of cascaded decomposition of automata. In *FOCS*, pages 672–682, 1990.
8. O. Maler and A. Pnueli. On the cascaded decomposition of automata, its complexity and its application to logic. Unpublished manuscript `http://www-verimag.imag.fr/~maler/Papers/decomp.pdf`, 1994.
9. R. McNaughton and S. Papert. *Counter-Free Automata*. MIT Press Cambridge, Mass., 1971.
10. A.R. Meyer. A note on star-free events. *J. ACM*, 16(2):220–225, 1969.
11. J.-E. Pin. *Varieties of Formal Languages*. Plenum, 1996.
12. J.L. Rhodes and K. Krohn. Algebraic theory of machines. In *Proc. Symp. on Math. Theory of Automata*. Polytechnic Press, Brooklyn, 1962.
13. M.-P. Schützenberger. On finite monoids having only trivial subgroups. *Information and Control*, 8(2):190–194, 1965.
14. M. Yoeli. Decomposition of finite automata. Technical Report TR-10, US Office of Naval Research, Information Systems Branch, Hebrew University, Jerusalem, 1963.
15. H.P. Zeiger. Cascade synthesis of finite-state machines. *Information and Control*, 10(4):419–433, 1967.
16. H.P. Zeiger. Yet another proof of the cascade decomposition theorem for finite automata. *Mathematical Systems Theory*, 1(3):225–228, 1967.