On the Cascaded Decomposition of Automata, its Complexity and its Application to Logic

(Draft)*

Oded Maler Amir Pnueli

November 15, 1994

Abstract

The primary decomposition theorem due to Krohn and Rhodes ([KR65]), which has been considered as one of the fundamental results in the theory of automata and semigroups, states that every automaton is homomorphic to a cascaded decomposition (wreath-product) of simpler automata of two kinds: *reset* automata and *permutation* automata. If the automaton is non-counting (and correspondingly its transformation semigroup is group-free) then it can be decomposed using only reset components.

There exist various proofs and partial proofs for the primary decomposition theorem e.g., [HS66, Ze67a, Ze67b, Gi68, MT69, La71, We76, Ei76]. None of them give explicit bounds on the size of the decomposition.[1] In this paper we give tight exponential bounds on the size of the decomposition as a function of the size of the original automaton. For the upper-bound we give an exponential algorithm by modifying the implicit construction appearing in [Ei74]. Our algorithm is constructive enough to allow implementation [??]. We apply the algorithm to give an exponential upper-bound on transforming star-free regular (resp. ω-regular) sets expressed by counter-free automata (resp. ω-automata) into past (resp. future) temporal logic formulae. These upper bounds improve upon previously-known non-elementary translations (MNP71, LPZ85, Zu86).

*Some of the results in this paper has been presented in: O. Maler, A. Pnueli, Tight Bounds on the Complexity of Cascaded Decomposition of Automata, *Proc. 31st Annual Symposium on Foundations of Computer Science*, St. Louis, Missouri, 672–682, IEEE Press 1990.

[1]To quote the last paragraph in Ginzburg's book ([Gi68]): *"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."*

Our decomposition construction is proved to be optimal by showing the existence of a family of automata such that the size of their minimal permutation-free decomposition is exponential in the size of the automaton (by size we refer to the total number of defined transitions).

1 Introduction

1.1 Overview and Historical Remarks

This papers has two main goals, the first one is to introduce the new results concerning the complexity of the cascaded decomposition and its implication for the translation between automata and temporal logic. The second goal is to reintroduce the Krohne-Rhodes primary decomposition theorem [KR65] to contemporary computer science audience, by providing a self-contained automata-theoretic and constructive proof.

There are many proofs of the Krohn-Rhodes primary decomposition theorem [KR65], e.g., [MT69, La71, We76], to mention a few. Among them, the proof of Zeiger ([Ze67a, 67b]) is more automata oriented (rather then semigroup oriented) and thus more useful for our purposes. Zeiger's proof has been corrected and presented more clearly by Ginzburg in [Gi68], based on some constructs in [Yo63]. Ginzburg's proof of the theorem contains some non-deterministic stages concerning the choice of semi-partitions. In addition, it does not discuss complexity issues explicitly. Another partial proof in this spirit appears in [HS66].

The proof in [Gi68] inspired Eilenberg to give a slight generalization of the primary decomposition called the *holonomy* decomposition ([Ei76], pp. 43-50). Eilenberg's theorem is cleaner and determinizes the choice of semi-partitions. It has however some deficiencies. It is a theorem on coverings of transformation semigroups, and as such it pays no attentions to the labels of the generators of the semigroup (i.e., the input alphabet, if we use automata-theoretic terminology). Consequently the outcome of the decomposition is not given explicitly as a valid automaton over the original alphabet. Another sociological problem associated with Eilenberg's construction is the elegant, concise and motivation-less algebraic style in which it is written, which makes it virtually inaccessible to many contemporary theoretical computer scientists.

The paper is organized as follows: in section 1.2 we discuss the general concept of automaton decomposition and give an intuitive introduction to the cascaded decomposition and the Krohn-Rhodes theorem. In section 3 we give the minimal background on automata and semigroups needed for the paper, define the cascaded decomposition and state the Krohn-Rhodestheorem in automata-theoretic terms. In section 2 we discuss the theoretical basis underlying the Zeiger-Ginzburg-Eilenberg family of proofs, namely the relation between cascaded decompositions and some trees of semi-partitions of Q. In section 3 we sketch a more algorithmic and automata-oriented version of Eilenberg's holonomy decomposition theorem, and analyze its complexity. In section 4 we establish the worst-case optimality of this construction, by giving a family of automata for which the minimal decompositions coincide with those produced by the algorithm. Section 5 we use our construction to improve some upper bounds on translating automata into temporal logic. Finally we discuss some application to non-deterministic and stochastic automata, and to everything else.

We hope that our relatively-ugly reconstruction, in addition to the new complexity results and its applications, will bring some important results form algebraic machine

theory back to mainstream computer science.

1.2 Decomposition in General

The problem of decomposing complex systems into simpler components is one of the fundamental problems in both science and engineering. The relationship between the behavior of individual components and their global "emergent" behavior when interconnected together is (either explicitly or implicitly) the subject matter of most disciplines ranging from physics to the social sciences. The particular case of interacting finite-state automata is the topic of computer science related sub-communities such as distributed computing, hardware realizations, semantics of parallelism, neural nets, cellular automata, distributed AI and behavior-based robotics – to mention a few. We will be concerned with the following problem:

The Problem: *Given an automaton, decompose it into several simple components such that their global behavior realizes the behavior of the original automaton.*

In the sequel we will be more specific about the following details:

- The type the elementary simple components.
- The mode of interconnection.
- The sense in which one behavior is realized by the other.

In general when we take several automata with their corresponding input-transition-output mechanisms and interconnect them, we get a compound object whose state-space is (encoded as) the cartesian product of the states of its components. One of the major characteristics of the decomposition is the degree of mutual influence of the components (dimensions) on each other, that is, to what extent does the behavior of one automaton depend on the state of another.[2] Technically this feature of the decomposition is captured by the interconnection scheme that directs output channels of certain automata to the input ports of others.

On one extreme of the spectrum of interconnection schemes lies the *direct product* in which each component responds to the external input, independently of the other automata. Such products are used, for example, in proving the closure of regular sets under Boolean operations. In the engineering terminology this is called *parallel decomposition.* On the other extreme lies the *unbounded feed-back decomposition* in which every automaton is sensitive to the behavior of others. Whenever a system can be decomposed into parallel "orthogonal" components it is good news for designers and analyzers, because each component can be treated independently.

More realistic models are based on intermediate degrees of inter-dependence. In cellular automata, neural nets and systolic computers, the components are located in a

[2]The degree of exposure of the various components to external (to the whole system) stimuli is another important feature usually ignored in the literature.

metric space and every automaton is influenced directly only by components that reside in some neighborhood around it. Such interconnection schemes look very reasonable because they conform with our geometrical and physical intuitions concerning the propagation of influence in space. Moreover, in hardware realizations of such models the communication channels can be routed more easily.

Another form of limited inter-dependence is achieved by partitioning the components into levels and letting components at level i influence components at level j, $j > i$ but not vice versa. This is the cascaded decomposition, the subject matter of the the rest of this paper.[3] All this notions are demonstrated in figure 1.

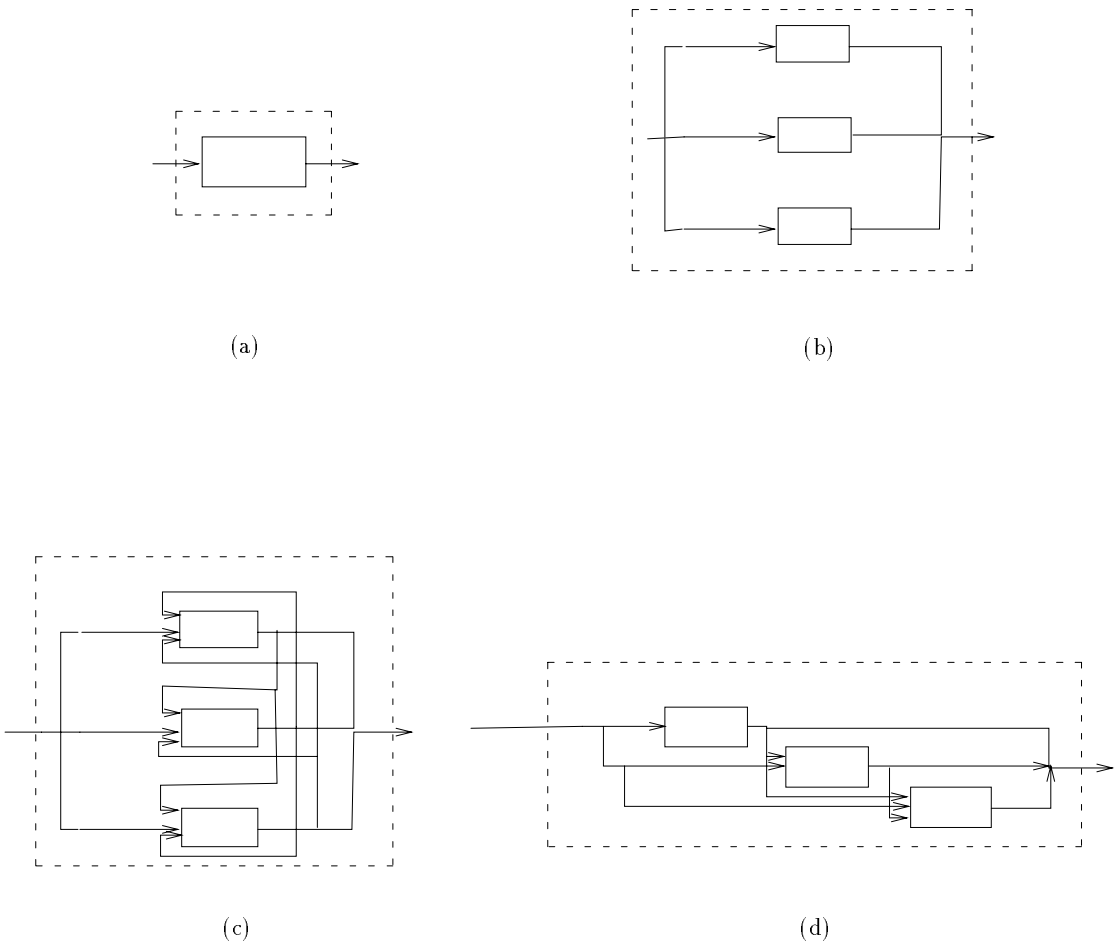

Figure 1: *An automaton (a) and various decomposition schemes: parallel or direct product (b), feed-back decomposition (c) and cascade product (d).*

[3]In fact, this type of decomposition is the lowest level α_i-hierarchy of decomposition types (see the monograph [Ge86]) where an automaton at level k cannot influence an automaton of level smaller than $k - i$.

1.3 Automata

We assume familiarity of the reader with finite automata and regular sets at the level of [HU79]. Our automaton model is a labeled state-transition graph:

Definition 1 (Automata) *An 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.*

Several variations are possible at this stage: the transition function can be total or partial – in the latter case it can be "totalized" by adding a sink state. It can be a relation rather than a function, but we defer the discussion on non-determinism to section **??**. The transition function can be extended naturally to sequences of input symbols, by letting $\delta(q, w\sigma) = \delta(\delta(q, w), \sigma)$, and to sets of states by letting $\delta(Q', \sigma) = \{\delta(q, \sigma) : q \in Q'\}$.

In its most "bare" version, the dynamics of the automaton, that is the transition function, is represented explicitly by a table in which every combination of an input symbol and an internal state has an entry. This form of representation can be visualized by a labeled directed graph whose nodes correspond to states and its edges to transitions (see figure 2). Although such a representation is finite, it might be impractical in many situations where the state-space is large, and more succint representations are used whenever possible. Programming languages or dataflow equations, to mention few examples, are among the formalisms that enable succint representations of certain transition systems.

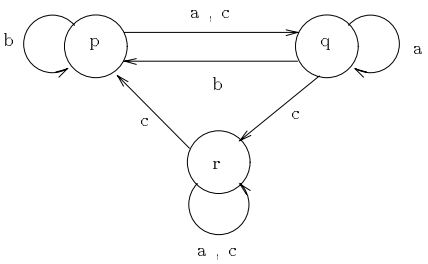

Figure 2: *An automaton*

An automaton can be made an *acceptor* by choosing an initial state $q_0 \in Q$ and a set of accepting states $F \subseteq Q$, and as such accept/recognize some *regular* language $U \subseteq \Sigma^*$ consisting of the set of all labels of trajectories from q_0 to state in F (see textbooks). 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 $\sigma \in \Sigma \cup \{\lambda\}$, which is closed under finitely many applications of concatenation and Boolean operations.*

It turns out that star-free sets have additional characterizations which will be discussed in the sequel.

Definition 3 (Automaton Homomorphisms) *A surjective function $\varphi : Q \to Q'$ is an automaton homomorphism.*[4] *from $\mathcal{A} = (\Sigma, Q, \delta)$ to $\mathcal{A}' = (\Sigma, Q', \delta')$ if it satisfies 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 two automata are mutually homomorphic we say they are isomorphic.

Intuitively $\mathcal{A}' \leq_\varphi \mathcal{A}$ means that anything that can be expressed or described using $\mathcal{A}'$ can be expressed using $\mathcal{A}$, and that $\mathcal{A}$ gives finer characterizations of phenomena than does $\mathcal{A}'$. Homomorphism is a partial-order relation and the canonical acceptor for a regular set U is the minimal element in the infinite lattice of all the automata accepting U, while the infinite tree acceptor is the maximal element.[5] Homomorphism is a special case of relational homomorphism defined below:

Definition 4 (Relational Homomorphisms) *A function $\varphi : Q \to 2^{Q'}$ such that $Q' = \bigcup_{q \in Q} \varphi(q)$ is a relational homomorphism from $\mathcal{A} = (\Sigma, Q, \delta)$ to $\mathcal{A}' = (\Sigma, Q', \delta')$ if it satisfies*

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

1.4 Semigroups

The theory of automata is strongly related to the algebraic theory of semigroups that deals with sets having an associative (but not necessarily invertible) binary operation defined on them. Two typical examples of semigroup are sequences of symbols under the concatenation operation, and transformations under the functional composition operation. Since a full exposition of semigroup theory will decrease the fraction of original work in this thesis below the limits of good taste, only a summary of the relevant notions will be given. The interested reader may consult [Gi68], [Ei76], [Pi86], [Ar69] or [La??].

Definition 5 (Semigroups, Monoids and Groups) *A Semigroup is a tuple $(S, \cdot)$ where S is a set and $\cdot$ is a binary associative operation ("multiplication") from $S \times S$ to S. For $s, t \in S$ we write st for their product. A Monoid $(M, \cdot, 1)$ is a semigroup containing an identity element 1 such that $m1 = 1m = m$ for every $m \in M$. A group $(G, \cdot, 1)$ is a monoid such that for every $g \in G$ there exists an element $g^{-1} \in G$ (an inverse) such that $gg^{-1} = 1$.*

Definition 6 (Subsemigroups, Generators) *A subsemigroup T of S is a subset $T \subseteq S$ that is closed under product, that is, $T^2 \subseteq T$. A subgroup of S is a subsemigroup which is a group. Let A be a subset of S. The smallest subsemigroup containing A is denoted by A^+ and it consists of all the elements of S that are a result of finitely many products of elements of A. Any subset $A \subseteq S$ such that $A^+ = S$ is called a generating set of S.*

[4] More precisely we define a *state*-homomorphism – we could extend the definition to include mappings between different input alphabets, output alphabets etc.

[5] Excluding, of course, automata with unreachable states.

Examples for semigroups are the natural numbers under addition or under multiplication, Boolean algebras under $\wedge$ or $\vee$, matrices under multiplications, and binary relations under composition. A finite semigroup can be described by its multiplication table. Every semigroup has a generating set (which might be S itself).

Definition 7 (Semigroup Homomorphisms) *A surjective function $\varphi : S \to S'$ is a semigroup homomorphism from $(S, \cdot)$ to $(S', *)$ if it satisfies $\varphi(s_1 \cdot s_2) = \varphi(s_1) * \varphi(s_2)$. In such a case we say that S' is homomorphic to S and denote it by $S' \leq_\varphi S$.* Semigroup homomorphic is transitive. Two mutually homomorphic semigroups are said to be isomorphic.

As in automata, homomorphism of semigroups corresponds the the intuitive notions of refinement/abstraction relations among structures.

Let $TR(Q)$ be the set of all total functions (*transformations*) of the form $s : Q \to Q$ for a finite set Q. One can see that $TR(Q)$ is a monoid under the operation of functional composition defined as $s \cdot t(q) = t(s(q))$ for every $q \in Q$. If the underlying set Q has n elements then $TR(Q)$ has n^n elements. The identity function on Q, I_Q, is the identity element of $TR(Q)$. A transformation as 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 notational conflict between algebraic, functional, and automata-theoretic conventions. Algebraically, the "action" of s on q, is denoted by qs and the associativity of composition is expressed as $qs \cdot s' = q(s \cdot s')$. 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 8 (Transformation Semigroups) *A transformation semigroup is $X = (Q, S)$ where Q the underlying set and S is a subsemigroup of $TR(Q)$, i.e., a set of transformations on Q closed under composition. Clearly if Q is finite, so is S.*

The importance of transformation semigroups comes from Cayley's theorem:

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

On the other hand every automaton gives rise to a natural transformation semigroup generated by the transformations induced by the letters of the input alphabets. It can be shown that *if two automata are homomorphic so are their corresponding transformation semigroups.*

Definition 9 (Rank) *Let Q be a set of n elements, and let $s \in Q \to Q$ be a transformation. The rank of s is defined as $|Qs|$, where $Qs = \{qs : q \in Q\}$.*

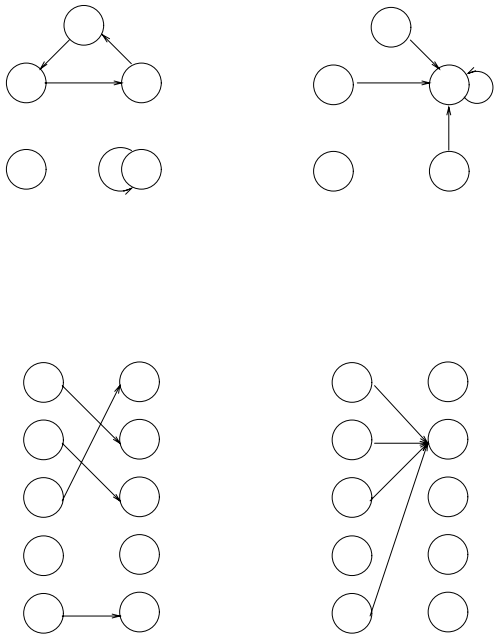

Figure 3: *A permutations (left) and a reset (right) illustrated as transition graphs (up) and as transformations (down).*

Permutations and resets (see figure 3) represent two extremes in the spectrum of transformations on Q. Among the n^n such transformations, the $n!$ permutations are those in which the domain and the range coincide and the rank is n. In resets, on the other hand, the rank is minimal, i.e., 1.

In an automaton, after the occurrence a reset the current state is determined precisely regardless of the previous one. On the other hand after applying a permutation we are no wiser then before if the previous state is unknown. From another point of view, a permutation is a reverse-deterministic, that is, by being at one state and knowing the last input event one can determine the previous state, contrary to resets in which 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 "multiplication table" of figure 4. Resets can be obtained by composing two non-reset transformations, e.g., $(122) \cdot (223) = (222)$, because composition can decrease the rank. On the other hand, because composition cannot increase the rank, permutation on Q cannot be composed from non-permutations on Q. However a permutation on a subset $\hat{Q} \subseteq Q$ can be composed from non-permutations as can be seen from fact 2.

Fact 2 *A transformation s permutes a subset $\hat{Q} \subseteq Q$ iff $s = s_1 s_1, \ldots, s_m$ for some $m \geq 1$ and there exists a sequence of subsets $\{Q_i\}_{i=0..m}$ such that $Q_0 = Q_m = \hat{Q}$ and the restriction of every s_i to Q_i is an injection to Q_{i+1}.*

$*$	p	r
p	p	r
r	r	r

Figure 4: Composition of permutations and resets

There are various ways to classify finite semigroups and their corresponding regular sets (see [Pin??]). An important sub-class of the 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 there is no word w that permutes a non-trivial subset of Q.*

It is not difficult to see that a semigroup is aperiodic iff it is group-free, and that an automaton is counter-free iff its semigroup of transformations is group-free. The following theorem relates aperiodic semigroups to star-free sets and consequently, to propositional temporal logic.

Theorem 3 (Schützenberger) *A regular set U is star-free if and only if its syntactic semigroup is aperiodic (and its minimal automaton is counter-free).*

1.5 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 $\mathcal{B}_1 = (\Sigma, Q_1, \delta_1)$ and $\mathcal{B}_2 = (Q_1 \times \Sigma, Q_2, \delta_2)$ be two automata. Their cascade product $\mathcal{B}_1 \circ \mathcal{B}_2 = (\Sigma, Q, \bar{\delta})$ is defined by letting $Q = Q_1 \times Q_2$ 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, $\mathcal{B}_1 \circ \mathcal{B}_2, \ldots \circ \mathcal{B}_k$ is defined as $(\ldots((\mathcal{B}_1 \circ \mathcal{B}_2) \circ \mathcal{B}_3 \ldots) \circ \mathcal{B}_k$.

Remark: Note that the "communication links" between $\mathcal{B}_1$ and $\mathcal{B}_2$ given implicitly via the definition of the input alphabet of $\mathcal{B}_2$. A neater definition using transducers will be given in chapter **??**.

Definition 12 (Permutation-Reset Automata) *An 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.*

We will sometime consider partial transition functions, that is, $\delta(q, \sigma)$ need not be defined for every $q \in Q$. In this case σ is said to induce a *partial permutation* if $\delta(q, \sigma) \neq \delta(q', \sigma)$ for every q, q' on which δ is defined, or a *partial reset* whenever $\delta(q, \sigma) = q'$ for

every such q such that $\delta(q,\sigma)$ is defined. In both cases it is straightforward to extend σ to become either a complete permutation (by letting $\delta(q,\sigma)=q$ for ever $q\in Q$ such that $\delta(q,\sigma)$ is undefined), or a complete reset (by letting $\delta(q,\sigma)=q'$ for every q).

The Krohn-Rhodes theorem states that the infinite class of permutation-reset automata is *homomorphically complete* for the cascade product, i.e., every automaton (up to inverse homomorphism) can be decomposed into a cascade of elements from this class. This celebrated theorem can be formulated as:

Theorem 4 (Krohn-Rhodes (Automata)) *For every automaton $\mathcal{A}$ there exists a cascaded decomposition $\mathcal{C}=\mathcal{B}_1\circ\mathcal{B}_2\circ\cdots\circ\mathcal{B}_k$ such that $\mathcal{A}\leq\mathcal{C}$, each $\mathcal{B}_i$ is a permutation-reset automaton, and any permutation group in some $\mathcal{B}_i$ is homomorphic to a subgroup of the transformation semigroup of $\mathcal{A}$* (this implies that if $\mathcal{A}$ is non-counting then all the permutations in $\{\mathcal{B}_i\}$ are trivial, i.e., identities).

It is this theorem that we are going to prove in detail within the following sections, as well as complexity bounds. Originally this theorem was expressed in terms of semigroups where $X_1\circ X_2$ is interpreted as the *wreath product* of X_1 and X_2. Since we mention this version only in the passing, we will spare the reader from the wreath product definition.

Theorem 5 (Krohn-Rhodes (Semigroups)) *Every transformation semigroup $X=(Q,S)$ admits a decomposition $X\leq X_1\circ\ldots\circ X_k$ where each X_i is either the monoid U_2* (the transformation monoid of the two-state reset-identity automaton) *or X_i is a simple group such that $X_i\leq X$.* Consequently if X is group-free (aperiodic) then all the X_i are isomorphic to U_2.

Concerning the binary reset automaton and its monoid U_2, it is worth mentioning that every n-state reset can be decomposed into a direct product of $\log n$ binary resets, so that all the results and complexity bounds to be mentioned in the sequel which are based on arbitrary resets apply to binary resets as well.

2 The Theoretical Basis of the Cascaded Decomposition

In this section we show the intimate relationship between cascaded decompositions of an automaton and certain semi-partitions of its set of states. This correspondence plays an important role both in the algorithm and in the lower-bound. In order to discuss it, let us first give more explicit definitions of the cascaded decomposition and of related structures.

In the sequel we will make a distinction between three "degrees" of decomposition. The first "plain" decomposition given below satisfies the homomorphism condition – no restrictions are imposed on the building blocks.

2.1 Decompositions and Configurations

Definition 13 (Cascaded Decomposition) *Let $\mathcal{A} = (\Sigma, Q, \delta)$ be an automaton. A cascaded decomposition for $\mathcal{A}$ is a pair $(\mathcal{C}, \varphi)$ where $\mathcal{C} = (\Sigma, P, \bar{\delta}) = \mathcal{B}_1 \circ \mathcal{B}_2 \circ \cdots \circ \mathcal{B}_k$ is a (possibly incomplete) automaton such that:*[6]

1. *For all i, $1 \leq i \leq k$, $\mathcal{B}_i = (Q_1 \times \cdots \times Q_{i-1} \times \Sigma, Q_i, \delta_i)$ where δ_i is possibly partial.*
2. *$P = Q_1 \times \ldots \times Q_k$ and the global transition function is evaluated coordinate-wise according to*

$$\bar{\delta}(\langle q_1, \ldots, q_k \rangle, \sigma) = \langle \delta_1(q_1, \sigma), \ldots, \delta_k(q_k, \langle q_1, \ldots, q_{k-1}, \sigma \rangle) \rangle \tag{1}$$

3. *$\varphi : Q_1 \times \ldots \times Q_k \to Q$ is a homomorphism from $\mathcal{C}$ to $\mathcal{A}$, that is, a surjective partial function such that*

$$\varphi(\bar{\delta}(\langle q_1, \ldots, q_k \rangle, \sigma)) = \delta(\varphi(\langle q_1, \ldots, q_k \rangle), \sigma) \tag{2}$$

   This fact is denoted by $\mathcal{A} \leq_\varphi \mathcal{C}$.

It follows from the definition that $\bar{\delta}(\langle q_1, \ldots, q_k \rangle, \sigma)$ is defined iff $\delta_i(q_i, \langle q_1, \ldots, q_{i-1}, \sigma \rangle)$ is defined for every i. The size of $\mathcal{A}$ is the number of defined transitions. In a complete automaton it is $|Q| \cdot |\Sigma|$. The size of $\mathcal{C}$ is the sum of the sizes of all the $\mathcal{B}_i$'s.

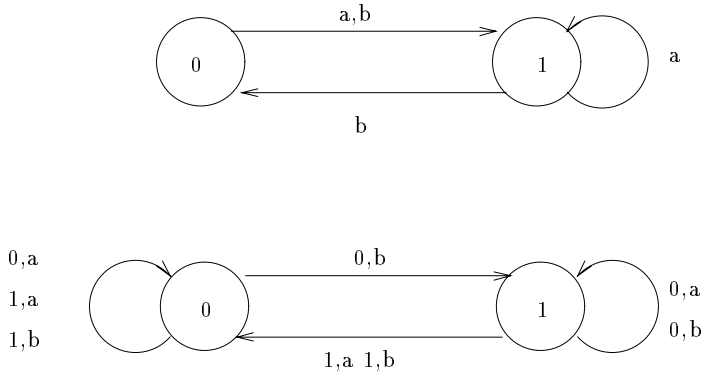

Figure 5: *A cascade.*

Example: In figure 5 one can see a cascade product whose global structure is depicted in figure 6. By defining the homomorphism $\varphi : \{0,1\} \times \{0,1\} \to \{q_1, q_2, q_3\}$ as $\varphi(\langle 0,1 \rangle) = q_1$, $\varphi(\langle 0,0 \rangle) = \varphi(\langle 1,1 \rangle) = q_2$ and $\varphi(\langle 1,0 \rangle) = q_3$ we obtain the automaton in figure 7.

Definition 14 (Configurations) *Let $\mathcal{C} = \mathcal{B}_1 \circ \mathcal{B}_2 \circ \cdots \circ \mathcal{B}_k$ be a a cascade product. An i-configuration, $1 \leq i \leq k$, is an element $\langle q_1, \ldots, q_i \rangle \in P_i = Q_1 \times \ldots \times Q_i$. Similarily we let $\mathcal{C}_i = \mathcal{B}_1 \circ \mathcal{B}_2 \circ \cdots \circ \mathcal{B}_i = (\Sigma, P_i, \bar{\delta}_i)$.*

[6] For convenience we assume that $\mathcal{B}_1$ is the trivial automaton over Σ with $Q_1 = \{p^*\}$. Sometimes $\mathcal{B}_1$ is omitted from the figures.

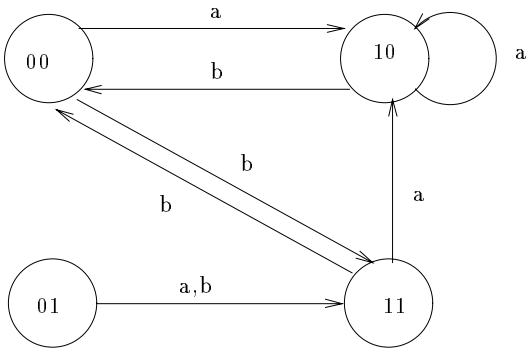

Figure 6: *The global automaton realized by the cascade of figure 5.*

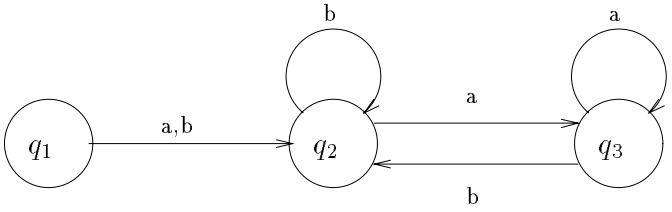

Figure 7: *A homomorphic image of the cascade described in figures 5 and 6.*

By definition $\mathcal{C} = (\Sigma, P, \bar{\delta}) = (\Sigma, P_k, \bar{\delta}_k) = \mathcal{C}_k$. When $p \in P_i$ and $p' \in Q_{i+1} \times, \ldots, \times Q_j$ we use $\langle p, p' \rangle$ to denote the corresponding j-configuration. In such a case we say that p is a *prefix* of $\langle p, p' \rangle$ and that $\langle p, p' \rangle$ *extends* p. (This hierarchical prefixing should not be confused with sequential concatenation). The set of all configuration has an obvious tree structure, with p being an ancestor of $\langle p, p' \rangle$. Note that by definition for any $i < j$ $P_j = P_i \times Q_{i+1} \times \ldots \times Q_j$ and $\mathcal{C}_j = \mathcal{C}_i \circ \mathcal{B}_{i+1} \circ \ldots \circ \mathcal{B}_j$.

Given that $(\mathcal{C}, \varphi)$ is a decomposition for $\mathcal{A} = (\Sigma, Q, \delta)$ and that a k-configuration $p \in P_k$ can be viewed as a (not necessarily unique) encoding of some state in Q, an i-configuration $r \in P_i$ for some $i < k$ corresponds to the set of $\mathcal{A}$-states which are encoded by some extensions of r. This intuitive notion of correspondence between configurations and subsets of Q is formalized in the following definition:

Definition 15 (Mapping Configurations to Subsets) *Let $(\mathcal{C}, \varphi)$ be a decomposition for $\mathcal{A}$, $\mathcal{C} = \mathcal{B}_1 \circ \mathcal{B}_2 \circ \cdots \circ \mathcal{B}_k$. We define a family of functions $\varphi_i : P_i \to 2^Q$, $1 \leq i \leq k$ using backward induction on i:*

$$\begin{aligned} \varphi_k(p) &= \begin{cases} \varphi(p) & \textit{when } \varphi(p) \textit{ is defined} \\ \emptyset & \textit{otherwise} \end{cases} \\ \varphi_{i-1}(p) &= \bigcup_{q_i \in Q_i} \varphi_i(\langle p, q_i \rangle) \end{aligned}$$

In other words, $\varphi_i(p)$ is the set of all states of $\mathcal{A}$ that correspond to k-configurations that are extensions of p. Clearly if $p \in P_i$ and $\langle p, p' \rangle \in P_j$ for some $j > i$ then $\varphi_j(\langle p, p' \rangle) \subseteq \varphi_i(p)$. A configuration $p \in P_i$ such that $\varphi_i(p) \neq \emptyset$ is called non-empty and the set of non-empty configurations is denoted by $\mathcal{P}$.

Example: The configuration tree for the decomposition described in figures 5, 6 and 7 appears in figure 8.

Claim 6 *Every φ_i is a relational homomorphism from $\mathcal{C}_i$ to $\mathcal{A}$, that is, it satisfies for every $p \in P_i$*

$$\delta(\varphi_i(p), \sigma) \subseteq \varphi_i(\bar{\delta}_i(p, \sigma))$$

Proof: For $i = k$ it follows from the fact that a homomorphism is a relational homomorphism. Suppose it is true for $i+1$, that is for every $p \in P_i$, $r \in Q_{i+1}$ we have

$$\delta(\varphi_{i+1}(\langle p, r \rangle), \sigma) \subseteq \varphi_{i+1}(\bar{\delta}_{i+1}(\langle p, r \rangle, \sigma))$$

To prove it for i we have

$$\begin{aligned}
\delta(\varphi_i(p), \sigma) &= \delta(\bigcup_r \varphi_{i+1}(\langle p, r \rangle), \sigma) \\
&\subseteq \bigcup_r \varphi_{i+1}(\bar{\delta}_{i+1}(\langle p, r \rangle, \sigma)) \\
&= \bigcup_r \varphi_{i+1}(\langle \bar{\delta}_i(p, \sigma), \delta_{i+1}(r, \langle p, \sigma \rangle) \rangle \\
&\subseteq \bigcup_r \varphi_{i+1}(\langle \bar{\delta}_i(p, \sigma), r \rangle) \\
&= \varphi_i(\bar{\delta}_i(p, \sigma))
\end{aligned}$$

■

2.2 TPSP Trees and Decomposition

The correspondence between configurations in the decomposition and subsets motivates the introduction of the following definitions:

Definition 16 (Semi-Partitions and Partitions) *Let Q be a finite set. A semi-partition on Q is a pair (M, ϕ) where M is a finite set and $\phi : M \to 2^Q$ is a total function such that $\bigcup_{m \in M} \phi(m) = Q$. A semi-partition (M, ϕ) is non-redundant if for every $m, m' \in M$, $\phi(m) \not\subseteq \phi(m')$.*

Definition 17 (Transition-Preserving Semi-Partitions) *Let $\mathcal{A} = (\Sigma, Q, \delta)$ be an automaton. A transition-preserving semi-partition (TPSP) for $\mathcal{A}$ is a system $(\Sigma, M, \Delta, \phi)$ where (Σ, M, Δ) is an automaton, (M, ϕ) is a semi-partition of Q and $\Delta : M \times \Sigma \to M$ satisfies[7] for all $m \in M$, $\sigma \in \Sigma$:*

$$\delta(\phi(m), \sigma) \subseteq \phi(\Delta(m, \sigma)) \tag{3}$$

[7] In other words, ϕ is a relational homomorphism from (Σ, M, Δ) to $\mathcal{A}$.

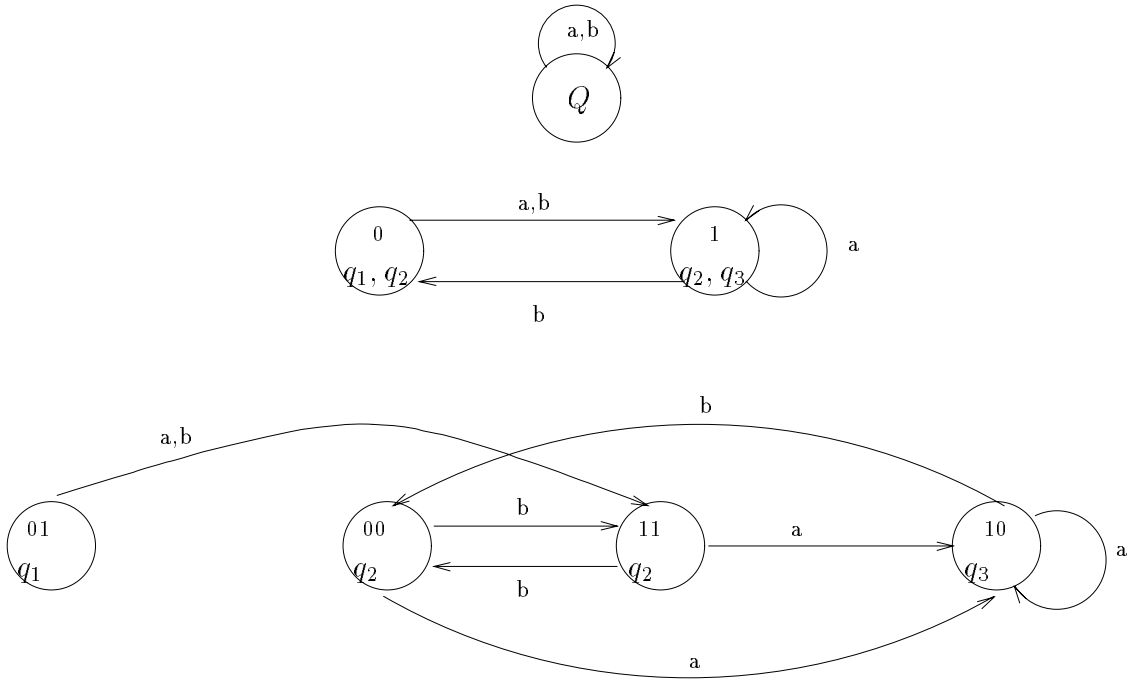

Figure 8: *A configuration tree. The configuration encodings appear at the upper part of each node while the subsets they are mapped to by φ at the bottom.*

Example: In figure 9 one can see a TPSP for the automaton in figure 7.

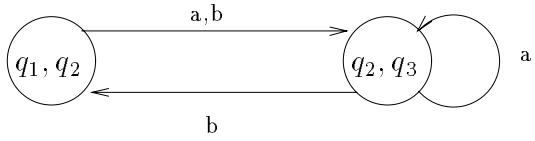

Figure 9: *A TPSP for the automaton in figure 7.*

In order to link cascaded decomposition and TPSPs we need a hierarchical structure on TPSPs that parallels the concept of configuration tree.

Definition 18 (TPSP Tree) *Let $\mathcal{A} = (\Sigma, Q, \delta)$ be an automaton. A TPSP tree is a system $\mathcal{T} = (\Sigma, M, \Delta, \pi, \phi)$ such that:*

1. *M is a finite set of nodes, containing a distinguished node $m^* \in M$, called the root of the tree.*

2. *$\pi : (M - \{m^*\}) \to M$ is a total parenthood function such that for every $m \neq m^*$, there exists some $i > 0$ such that $\pi^i(m) = m^*$. We will use $\pi^{-1}(m)$ to denote the "children" of m, and $\pi^{-i}(m)$ to denote descendants.* This function can be used

to partition M into levels according to the ancestral distance from m^* by letting $M_i = \{m \in M : m^* = \pi^i(m)\}$. The *height* of the tree, denoted by h, is the length of the longest ancestral chain plus 1.

3. *The tree is uniform, that is, every node has some descendant whose distance from the root is h.*

4. *$\phi : M \to 2^Q - \emptyset$ is a surjective labeling function, such that $\phi(m^*) = Q$. For every $q \in Q$, there exists some $m \in M_h$ such that $\phi(m) = \{q\}$, and for every $m \in M_h$, $|\phi(m)| = 1$.*

5. *For every $m \in M - M_h$, $(\pi^{-1}(m), \phi^{(m)})$ is a non-redundant semi-partition of $\phi(m)$ where $\phi^{(m)}$ is the restriction of ϕ to $\pi^{-1}(m)$.* We also denote the restriction of ϕ to M_i by ϕ_i.

6. *$\Delta : M \times \Sigma \to M$ is a transition function which can be decomposed into a union of level-preserving functions of the form $\Delta_i : M_i \times \Sigma \to M_i$.*

7. *Δ is ancestor-consistent: $\Delta(\pi(m), \sigma) = \pi(\Delta(m, \sigma))$.*

8. *For every i, $(\Sigma, M_i, \Delta_i, \phi_i)$ is a TPSP for $\mathcal{A}$.*

It follows from these conditions that for every $i > 1$, $\phi_i(m) \subseteq \phi_{i-1}(\pi(m))$ and that the restriction of π to M_i defines a homomorphism from M_i to M_{i-1}. Moreover ϕ_h is a homomorphism from (Σ, M_h, Δ_h) to $\mathcal{A}$. A TPSP tree for the automaton in figure 7 appears in figure 10. The similarity between this TPSP tree and the configuration tree is not a coincidence and we are going to prove that 1) Given a decomposition, its set of non-empty configurations has a TPSP-tree structure, and 2) From a TPSP-tree $\mathcal{T}$ for $\mathcal{A}$ one can construct a decomposition for $\mathcal{A}$ such that the associated tree of non-empty configurations is isomorphic to $\mathcal{T}$.

Claim 7 (Decomposition $\Rightarrow$ Tree) *Every decomposition $(\mathcal{C}, \varphi)$ for $\mathcal{A} = (\Sigma, Q, \delta)$ with $\mathcal{C} = \mathcal{B}_1 \circ \mathcal{B}_2 \circ \cdots \circ \mathcal{B}_k$ implies a TPSP tree $\mathcal{T} = (\Sigma, M, \Delta, \pi, \phi)$ of height k.*

Proof: By letting $M = \mathcal{P}$, $\Delta_i = \bar{\delta}_i$ and $\phi_i = \varphi_i$ for every i, $1 \leq i \leq k$, and $\pi(\langle p, q \rangle) = p$ the desired TPSP is constructed. ∎

In order to prove the other direction and construct a decomposition from a TPSP tree we define a two functions that map nodes in the TPSP tree to states of the components and to configuration in the decomposition.

Definition 19 (Mapping Nodes to States and Configurations I) *Let $\mathcal{T} = (\Sigma, M, \Delta, \pi, \phi)$ be a TPSP tree for $\mathcal{A}$. For every $i > 0$ we let Q_i be a set such that $|Q_i| = \max\{|\pi^{-1}(m)|\}_{m \in M_{i-1}}$, and define a function $\theta_i : M_i \to Q_i$ such that for every $m \in M_{i-1}$, the restriction of θ_i to $\pi^{-1}(m)$ is an injection. From this function we can define an injection $\psi : M \to \mathcal{P}$,*

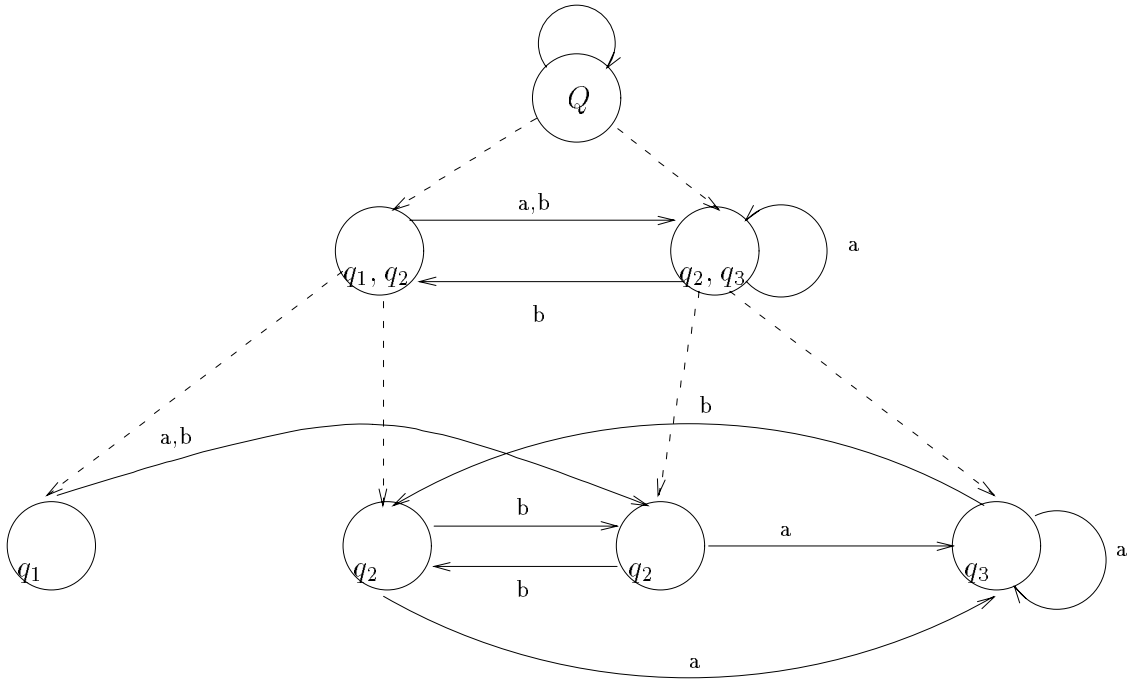

Figure 10: *A TPSP tree for the automaton in figure 7.*

that maps every node to a distinct configuration. As usual, ψ can be decomposed into $\psi_i : M_i \to P_i$. *The definition of* ψ_i *is by induction on* i*:*

$$\psi_1(m^*) = \theta_1(m^*) = q^* \tag{4}$$

$$\psi_{i+1}(m) = \langle \psi_i(\pi(m)), \theta_{i+1}(m) \rangle \tag{5}$$

Let $\theta'_i : Q_i \to 2^{M_i}$ be the inverse of θ_i. One can see that the inverse of ψ_i, $\psi'_i : P_i \to M_i$ admits the following definition:

$$\psi'_{i+1}(\langle p, q \rangle) = \begin{cases} \theta'_{i+1}(q) \cap \pi^{-1}(\psi'_i(p)) & \text{if } \theta'_{i+1}(q) \cap \pi^{-1}(\psi'_i(p)) \neq \emptyset \\ \perp & \text{otherwise} \end{cases} \tag{6}$$

Claim 8 (Tree $\Rightarrow$ Decomposition) *From a TPSP tree* $\mathcal{T} = (\Sigma, M, \Delta, \pi, \phi)$ *for* $\mathcal{A}$ *of height* k*, one can construct a cascaded decomposition* $(\mathcal{C}, \varphi)$*,* $\mathcal{C} = \mathcal{B}_1 \circ \mathcal{B}_2 \circ \cdots \circ \mathcal{B}_k$ *such that the sizes of* $\mathcal{T}$ *and* $\mathcal{C}$ *are equal.*

Proof: We define Q_i, θ and ψ as in definition 19. The transition function of $\mathcal{B}_i$ is defined as

$$\delta_i(q, \langle p, \sigma \rangle) = \begin{cases} \theta_i(\Delta_i(\psi'\langle p, q \rangle, \sigma)) & \text{if } \psi'\langle p, q \rangle \text{ is defined} \\ \perp & \text{otherwise} \end{cases} \tag{7}$$

In other words, the i-configuration $\langle p, q \rangle$ is translated into its corresponding node (if any), on which σ is applied according to the transition function of the TPSP tree, and

the resulting node is translated back into a state in Q_i. It suffices to show an isomorphism between the full decomposition $\mathcal{C}_k = (\Sigma, P_k, \bar{\delta}_k)$ and the lowest level of the TPSP tree (Σ, M_k, Δ_k), which is inverse-homomorphic to $\mathcal{A}$ by definition. The proof is by induction on i, the base case is trivial. Suppose it is true for i, that is, for every $m \in M_i$ and $\sigma \in \Sigma$ we have

$$\psi_i(\Delta_i(m, \sigma)) = \bar{\delta}_i(\psi_i(m), \sigma) \tag{8}$$

and we want to prove that for every $m' \in \pi^{-1}(m)$ we have

$$\psi_{i+1}(\Delta_{i+1}(m', \sigma)) = \bar{\delta}_{i+1}(\psi_{i+1}(m'), \sigma) \tag{9}$$

The left-hand side of (9) is transformed according to Definition 5 and the properties of TPSP trees, while to the the right-hand side we apply the definition of a cascade to obtain:

$$\langle \psi_i(\Delta_i(m, \sigma)), \theta_{i+1}(\Delta_{i+1}(m', \sigma)) \rangle = \langle \bar{\delta}_i(\psi_i(m), \sigma), \delta_{i+1}(\theta_{i+1}(m'), \langle \psi_i(m), \sigma \rangle) \rangle \tag{10}$$

The identity of the first coordinates follows from the premises while the equality of the second coordinate is just a rephrasing of (7) with $p = \psi_i(m)$ and $q = \theta_{i+1}(m')$. ∎

Example: From the the TPSP tree of figure 10 one can obtain the decomposition of figure 5.

2.3 Injection-Reset TPSP Trees and Permutation-Reset Decompositions

So far we have shown the correspondence between arbitrary cascaded decompositions and TPSP trees. Our next step is to see what additional constraints are imposed on the configuration (and hence TPSP) trees when the building blocks are restricted to be permutation-reset automata.

Definition 20 (Injection-Reset TPSP Tree) *An injection-reset TPSP tree is a TPSP tree $\mathcal{T} = (\Sigma, M, \Delta, \pi, \phi)$ satisfying the following constraints on Δ: For every $m \in M_i$, $i < h$, and for every $\sigma \in \Sigma$ either*

- $\Delta_{i+1}(m_1, \sigma) \neq \Delta_{i+1}(m_2, \sigma)$ *for every* $m_1, m_2 \in \pi^{-1}(m)$,

or

- $\Delta_{i+1}(m_1, \sigma) = \Delta_{i+1}(m_2, \sigma)$ *for every* $m_1, m_2 \in \pi^{-1}(m)$

In other words if $m' = \Delta_i(m, \sigma)$ then σ induces either an injection from $\pi^{-1}(m)$ to $\pi^{-1}(m')$ or a reset from $\pi^{-1}(m)$ to some single $r' \in \pi^{-1}(m')$

Claim 9 (P-R-Decomposition ⇒ I-R-Tree) *Every decomposition $(\mathcal{C}, \varphi)$ for $\mathcal{A} = (\Sigma, Q, \delta)$ with $\mathcal{C} = \mathcal{B}_1 \circ \mathcal{B}_2 \circ \cdots \circ \mathcal{B}_k$ such that $\{\mathcal{B}_i\}_{1 \leq i \leq k}$ are permutation-reset automata, implies an injection-reset TPSP tree of height k.*

Proof: We show how the structure of the building blocks affects the structure of the configuration tree. Consider some $p \in P_i$ such that $\bar{\delta}_i(p,\sigma) = p'$. Suppose $\langle p,\sigma\rangle$ induces a reset in $\mathcal{B}_{i+1}$, that is $\delta_{i+1}(q_1,\langle p,\sigma\rangle) = \delta_{i+1}(q_2,\langle p,\sigma\rangle)$ for every $q_1,q_2 \in Q_{i+1}$ on which a $\langle p,\sigma\rangle$-labeled transition is defined. Consequently $\bar{\delta}_{i+1}(\langle p,q_1\rangle,\sigma) = \langle p',\delta_{i+1}(q_1,\langle p,\sigma\rangle)\rangle = \bar{\delta}_{i+1}(\langle p,q_2\rangle,\sigma)$ and σ is indeed a reset on P_{i+1}. If $\langle p,\sigma\rangle$ is a permutation in $\mathcal{B}_{i+1}$ then $\delta_{i+1}(q_1,\langle p,\sigma\rangle) \neq \delta_{i+1}(q_2,\langle p,\sigma\rangle)$ for every $q_1,q_2 \in Q_{i+1}$ and consequently $\bar{\delta}_{i+1}(\langle p,q_1\rangle,\sigma) \neq \bar{\delta}_{i+1}(\langle p,q_2\rangle,\sigma)$ and σ induces an injection from $\{p\} \times Q_{i+1}$ to $\{p'\} \times Q_{i+1}$. By virtue of claim 7 these properties of the configuration tree are reflected in the TPSP tree. ▟

Claim 10 (I-R-Tree ⇒ P-R-Decomposition) *From an injection-reset TPSP tree* $\mathcal{T} = (\Sigma, M, \Delta, \pi, \phi)$ *of height* $k-1$*, one can construct a cascaded decomposition* $\mathcal{C} = \mathcal{B}_1 \circ \mathcal{B}_2 \circ \cdots \circ \mathcal{B}_k$ *made of permutation-reset automata such that* $|\mathcal{T}| = |\mathcal{C}|$.

Proof: The proof is by the same construction in the proof of claim 8. What remains to show is that for every i, $p \in P_{i-1}$, $\sigma \in \Sigma$, $\langle p,\sigma\rangle$ is indeed a reset or a permutation in $\mathcal{B}_i$. From (7) it follows that for every $q_1,q_2 \in Q_i$ $\delta_i(q_1,\langle p,\sigma\rangle) = \delta_i(q_2,\langle p,\sigma\rangle)$ iff $\Delta_i(\psi'(\langle p,q_1\rangle),\sigma)) = \Delta_i(\psi'(\langle p,q_2\rangle),\sigma))$ and since $\mathcal{T}$ is an injection-reset TPSP tree, the result follows. ▟

The automaton in figure 11 admits an injection-reset TPSP as in figure 12, which yields the permutation-reset decomposition of figure 13. The construction so far (definition 19 and the proof of claim 8) involves what we call *"injection folding"*, that is, injections between "cousins" in the TPSP tree are transformed into permutations in the cascade, due to the encoding of different nodes by the same states in the cascade.

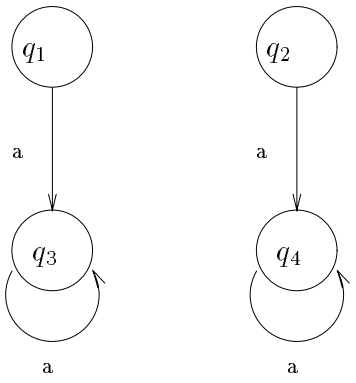

Figure 11: *An automaton.*

Next we show that every Q there exists a generic tree of subsets that can be made an injection-reset TPSP tree for any automaton whose set of states is Q, and lead to generic permutation-reset decomposition.

Claim 11 (The Generic TPSP Tree) *For every automaton* $\mathcal{A} = (\Sigma, Q, \delta)$ *with* $|Q| = n$*, it is possible to construct an injection-reset TPSP tree* $\mathcal{T} = (\Sigma, M, \Delta, \pi, \phi)$ *of height* n *and size* $O(2^n)$.

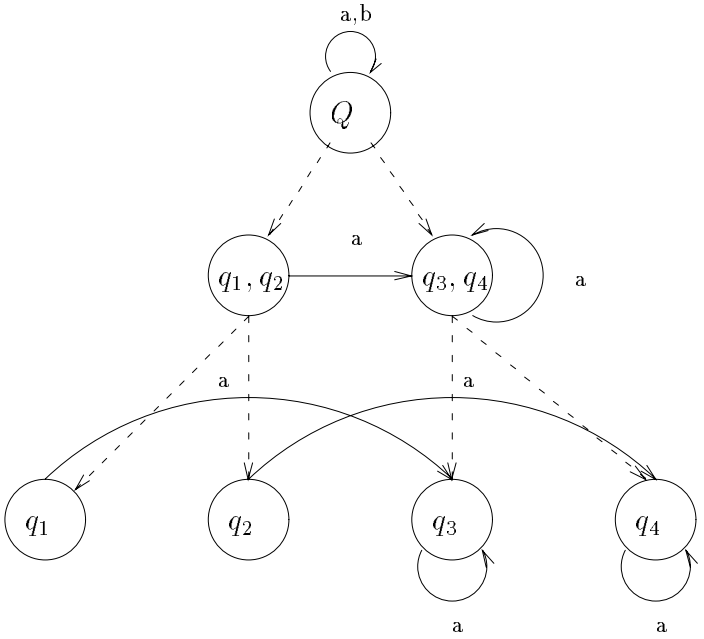

Figure 12: *An injection-reset TPSP tree for the automaton in figure 11.*

Proof: The construction goes as follows: The root is m^* and $\phi(m^*) = Q$. For every $m \in M$ with $\phi(m) = \{q_1, \ldots, q_l\}$ we define $\pi^{-1}(m) = \{m_1, \ldots, m_l\}$ and let $\phi_{i+1}(m_j) = \phi_i(m) - \{q_j\}$. The transition function is defined recursively: First we let $\Delta_1(m^*, \sigma) = m^*$ for every $\sigma \in \Sigma$. Then, for every m such that $\Delta_i(m, \sigma) = m'$ we calculate the transition function of its children according to the following two cases:

1. If $|\delta(\phi(m), \sigma)| = |\phi(m)|$ then σ induces a bijection between $\phi(m)$ to $\phi(m')$ and consequently between the sets associated with their children. So for every $r \in \pi^{-1}(m)$ we let $\Delta_{i+1}(r, \sigma) = r'$ where $r' \in \pi^{-1}(m')$ is *the* node satisfying $\delta(\phi_{i+1}(r), \sigma) =$

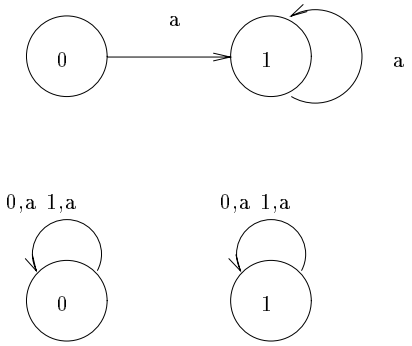

Figure 13: *A permutation-reset decomposition for the automaton in figure 11 constructed by injection folding from the injection-reset TPSP tree of figure 12.*

$\phi_{i+1}(r')$.

2. Otherwise, if $|\delta(\phi(m),\sigma)| < |\phi(m)|$ then there exists some $r' \in \pi^{-1}(m')$ such that $\delta(\phi(m),\sigma) \subseteq \phi(r')$ and for every $r \in \pi^{-1}(m)$ $\delta(\phi(r),\sigma) \subseteq \phi(r')$. Hence we can let σ be a reset on $\pi^{-1}(m)$ by defining $\Delta_{i+1}(m,\sigma) = m$'. ∎

An automaton and its corresponding generic TPSP tree are depicted in figures 14 and 15.

Corollary 12 *Every n-state automaton admits a decomposition into a cascade of n permutation-reset automata having an exponential total size.*

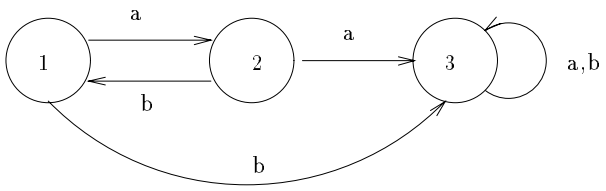

Figure 14: *An automaton.*

There are two problems related to this construction. First, it is a brute-force construction always leading to an exponential tree (and decomposition) regardless of the specific transition structure of $\mathcal{A}$. The first part of the algorithm to be presented in section 3 gets rid of the "obviously redundant" nodes and levels. In spite of these improvements, we will also show that in the worst case the construction has to be exponential anyway.

The second problem with injection-reset TPSPs in general is that some "superficial" permutations may be created by the folding process, without corresponding permutations in $\mathcal{A}$. The example in figures 16, 17 and 18 shows how this phenomenon might occur regardless of the choice of ψ.

2.4 Bijection-Reset TPSP Trees and Holonomy Decompositions

Now we are ready to impose further restrictions on the TPSP trees and on the decomposition and ensure that the decomposition reflects the sub-groups of $\mathcal{A}$. Before doing so let us give more explicit definitions of these sub-groups.

Definition 21 (Permutation Subgroups) *Let $X = (Q,S)$ be the transformation semigroup associated with an automaton $\mathcal{A} = (\Sigma,Q,\delta)$. The permutation subgroup $H_{\hat{Q}} = (\hat{Q},G_{\hat{Q}})$ associated with some $\hat{Q} \subseteq Q$ consists of all the permutations of the form $g : \hat{Q} \to \hat{Q}$, where g is the restriction of some $s \in S$ to $\hat{Q}$.*

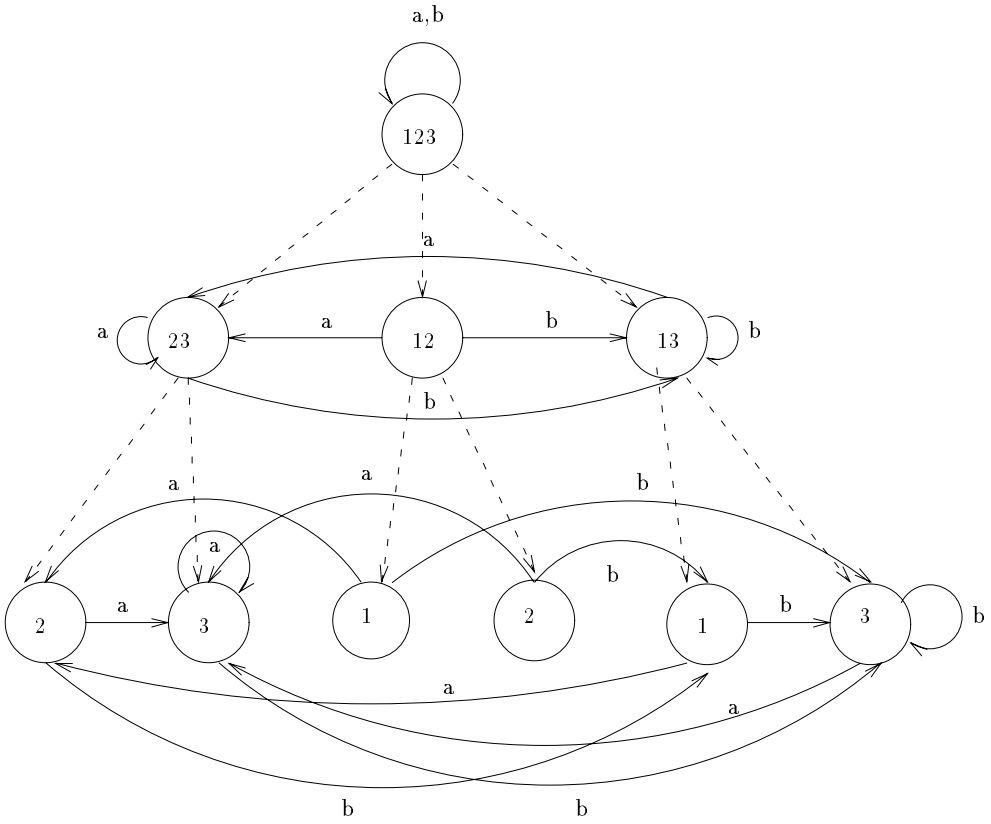

Figure 15: *The generic injection-reset TPSP tree for the automaton in figure 14.*

Definition 22 (Holonomy Decomposition) *A holonomy decomposition for $\mathcal{A}$ is a permutation-reset decomposition such that for every i, $\mathcal{B}_i$ satsifies the additional condition: Its state-space Q_i and its input alphabet Σ_i can be partitioned into $Q_{i1}, \ldots, Q_{il}$ and $\Sigma_{i1}, \ldots, \Sigma_{il}$ such that Σ_{ij} is defined exactly over Q_{ij}, and the permutation group on Q_{ij} (generated by the permutations in Σ_{ij})* denoted by *$H_{ij} = (Q_{ij}, G_{ij})$ is homomorphic to some permutation subgroup of $\mathcal{A}$.*

In particular, if $\mathcal{A}$ is a non-counting automaton then its holonomy decomposition consists only of reset-identity automata. It is an easy excercise to show that a holonomy decomposition (introduced by Eilenberg) implies the original Krohn-Rhodes decomposition where *every component* in the cascade is *either* a permutation automaton *or* a reset-identity automaton.

The following definitions are needed in order to put some constraints on the TPSP tree.

Definition 23 (Subset Equivalence) *Let $\mathcal{A} = (\Sigma, Q, \delta)$ be a (complete) automaton. Two subsets $\hat{Q}_1, \hat{Q}_2 \subseteq Q$ are equivalent if there exist $w, w' \in \Sigma^*$ such that:*

1. *$\delta(\hat{Q}_1, w) = \hat{Q}_2$ and $\delta(\hat{Q}_2, w') = \hat{Q}_1$*

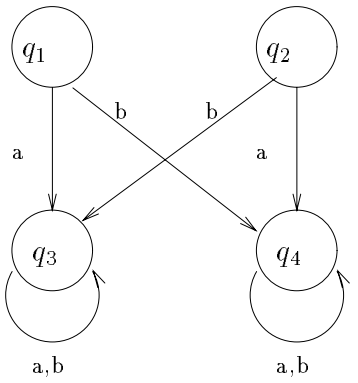

Figure 16: *A counter-free automaton.*

2. ww' *and* $w'w$ *induce identities on* $\hat{Q}_1$ *and* $\hat{Q}_2$ *respectively.*

This fact is denoted by $\hat{Q}_1 \overset{w,w'}{\sim} \hat{Q}_2$ *or simply* $\hat{Q}_1 \sim \hat{Q}_2$.

Remark: If for some $\hat{Q}_1, \hat{Q}_2 \subseteq Q$ and $w, w' \in \Sigma^*$ only condition 1 is satisfied, there exist some u, u' such that $\hat{Q}_1 \overset{u,u'}{\sim} \hat{Q}_2$. This is due to the fact that ww' is a permutation on $\hat{Q}_1$ and $w'w$ is a permutation on $\hat{Q}_2$, so for some l, $(ww')^l$ and $(w'w)^l$ are identities, and by letting $u = w$ and $u' = w'(ww')^{l-1}$ we have $\hat{Q}_1 \overset{u,u'}{\sim} \hat{Q}_2$.

Definition 24 (Node Equivalence) *Two nodes* $m, m' \in M_i$ *in a TPSP tree* $\mathcal{T} = (\Sigma, M, \Delta, \pi, \phi)$ *are equivalent if there exist* $w, w' \in \Sigma^*$ *such that:*

1. $\Delta(m, w) = m'$ *and* $\Delta(m', w') = m$.

2. $|\pi^{-1}(m)| = |\pi^{-1}(m')|$

3. $\phi(m) \overset{w,w'}{\sim} \phi(m')$ *(in the sense of subset equivalence).*

This fact is also denoted by $m \overset{w,w'}{\sim} m'$ *or simply* $m \sim m'$.

Definition 25 (Bijection-Reset TPSP Tree) *A bijection-reset TPSP tree is an injection-reset TPSP tree* $\mathcal{T} = (\Sigma, M, \Delta, \pi, \phi)$ *such that for every* $m, m' \in M_i$, $\sigma \in \Sigma$, *such that* $\Delta(m, \sigma) = m'$, *if* σ *induces an injection*[8] *from* $\pi^{-1}(m)$ *to* $\pi^{-1}(m')$ *then* $m \overset{\sigma,w}{\sim} m'$ *for some* $w \in \Sigma^*$.

With every node in a TPSP tree we associate a permutation group as follows:

[8]In the case when $|\pi^{-1}(m)| = 1$, we view σ as a partial reset, not as an injection, so that $m \sim m'$ need not hold.

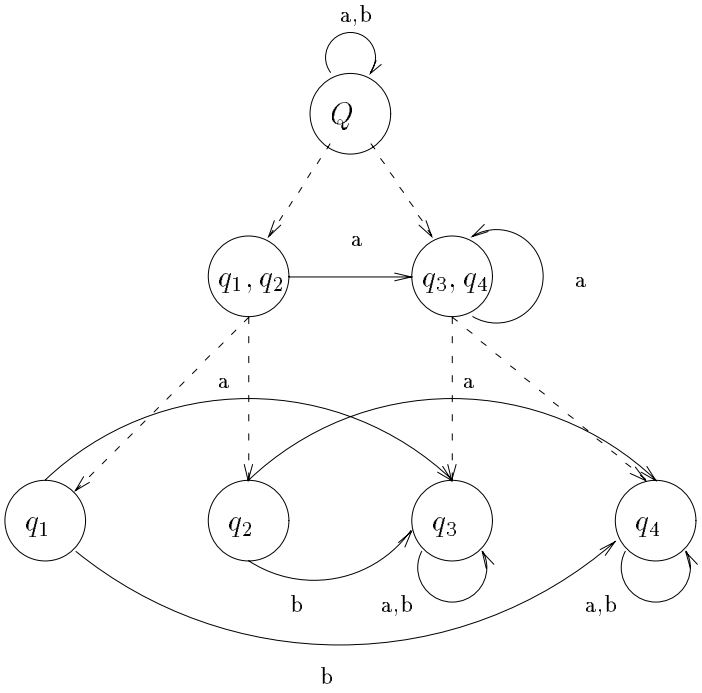

Figure 17: *An injection-reset TPSP tree for the automaton in figure 16.*

Definition 26 (Holonomy Groups) *Let $\mathcal{T} = (\Sigma, M, \Delta, \pi, \phi)$ be a TPSP tree, and let $m \in M$ be a node. The holonomy group of m is the permutation group $H_m = (\pi^{-1}(m), G_m)$ consisting of all the permutations of the form $g : \pi^{-1}(m) \to \pi^{-1}(m)$ induced by some words in Σ^*.* Note that H_m can be the trivial group.

Claim 13 (Holonomy Group $\leq$ Permutation Subgroup) *Let $\mathcal{T} = (\Sigma, M, \Delta, \pi, \phi)$ be a bijection-reset TPSP tree for $\mathcal{A} = (\Sigma, Q, \delta)$. For every $m \in M$, H_m is homomorphic to $H_{\phi(m)}$.*

Proof: We define a mapping $\mu : G_{\phi(m)} \to G_m$ as follows: for every $s : \phi(m) \to \phi(m) \in G_{\phi(m)}$ we associate $\mu(s) : \pi^{-1}(m) \to \pi^{-1}(m)$ such that for every $m' \in \pi^{-1}(m)$

$$\Delta(m', \mu(s)) = \phi^{-1}(\delta(\phi(m'), s)) \cap \pi^{-1}(m) \tag{11}$$

In other words, s is now applied[9] on the subset $\phi(m') \subseteq \phi(m)$. This mapping is well-defined due to the following reasons: s is a bijection on $\phi(m)$ and consequently on $\phi(m')$ and thus there must be at least one $m'' \in \pi^{-1}(m)$ such that $m'' = \phi^{-1}(\delta(\phi(m'), s))$ and due to non-redundancy it is unique. To show that μ is a homomorphism we need to show that for every $m' \in \pi^{-1}(m)$, $\Delta(\Delta(m', \mu(s)), \mu(s')) = \Delta(m', \mu(s \cdot s'))$:

[9] As a compromise between the algebraic notation $m \cdot s$ and the functional notation $s(m)$, we use a transition function notation $\Delta(m, s)$ which is a tolerable abuse of language.

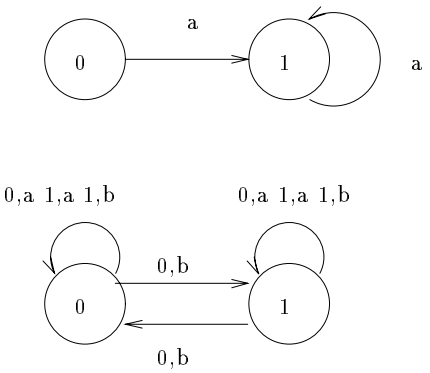

Figure 18: *A permutation-reset decomposition including a non-trivial counter for the automaton in figure 16 constructed by injection folding from the injection-reset TPSP tree of figure 17.*

$$\begin{aligned}
\Delta(\Delta(m', \mu(s)), \mu(s')) &= \\
\phi^{-1}(\delta(\phi(\phi^{-1}(\delta(\phi(m'), s)) \cap \pi^{-1}(m)), s')) \cap \pi^{-1}(m) &= \\
\phi^{-1}(\delta(\delta(\phi(m'), s)), s') \cap \pi^{-1}(m) &= \phi^{-1}(\delta(\phi(m'), s \cdot s') \cap \pi^{-1}(m) \\
&= \Delta(m', \mu(s \cdot s'))
\end{aligned}$$

∎

Claim 14 (Holonomy Decomposition ⇒ B-R-Tree) *Every holonomy decomposition* $\mathcal{C} = \mathcal{B}_1 \circ \mathcal{B}_2 \circ \cdots \circ \mathcal{B}_k \leq_\varphi \mathcal{A} = (\Sigma, Q, \delta)$ *implies a bijection-reset TPSP tree of height* k.

Proof: As in the proof of claims 7 and 9, we look at the configuration tree generated by the decomposition and see that it is indeed a bijection-reset TPSP tree. Let $p, p' \in P_i$ be such that $\Delta(p, \sigma) = p'$ and σ is an injection from $\{p\} \times Q_{i+1}$ to $\{p'\} \times Q_{i+1}$. This implies that $\langle p, \sigma \rangle$ is a permutation on Q_{i+1} and since $\mathcal{C}$ is a holonomy decomposition we have a w such that σ and w induce mutually-inverse bijections between $\varphi_{i-1}(p)$ and $\varphi_{i-1}(p')$, and consequently $p \overset{\sigma,w}{\sim} p'$ and the configuration tree is indeed a bijection-reset TPSP tree. ∎

In order to prove that such a tree implies a holonomy decomposition we need to introduce a modified encoding procedure. Unlike the previous construction, $\pi^{-1}(m)$ and $\pi^{-1}(m')$ are encoded using the same set of states $Q_{ij} \subseteq Q_i$ *only if* $m \sim m'$. Moreover, the mapping of $\pi^{-1}(m)$ into Q_{ij} is not arbitrary, but depends on the mappings of $\pi^{-1}(m')$ for every $m \sim m'$.

Definition 27 (Mapping Nodes to States and Configurations II) *For every* i, *let* $M_{i-1,j} \subseteq M_{i-1}$ *be an* $\sim$*-class, and let* $\hat{m} \in M_{i-1,j}$ *be an arbitrary node. We define the set*

$Q_{i,j}$ to be $\pi^{-1}(\hat{m})$ and call the nodes in $\pi^{-1}(\hat{m})$ representatives. For every $m \in M_{i-1,j}$, let $u_m, v_m \in \Sigma^*$ be words such that $\hat{m} \overset{u_m,v_m}{\sim} m$. The function $\theta_{i,j} : \bigcup_{m \in M_{i-1,j}} \pi^{-1}(m) \to Q_{i,j}$ is defined for every $m \in M_{i-1,j}$, $r \in \pi^{-1}(m)$ as:

$$\theta_{i,j}(r) = (\Delta(r, v_m)) \tag{12}$$

Note that $\hat{m} \overset{\lambda,\lambda}{\sim} \hat{m}$, so $u_{\hat{m}} = v_{\hat{m}} = \lambda$. The set of all cascade states at level i is $Q_i = \bigcup_j Q_{i,j}$, and $\theta_i : M_i \to Q_i$ is defined as $\theta_i = \bigcup_j \theta_{i,j}$.

This mapping can be extended naturally into $\psi : M \to \mathcal{P}$ as in equation 5:

$$\psi_i(m) = \langle \psi_{i-1}(\pi(m)), \theta_i(m) \rangle \tag{13}$$

The inverse mapping $\psi' : \mathcal{P} \to M$ satisfies

$$\psi'(\langle p, q \rangle) = \Delta(q, u_m) \tag{14}$$

where $m = \psi'(p)$. In other words, $q \in Q_{i,j}$ (which is just a representative from $\pi^{-1}(\hat{m})$) is decoded back to the original node according to the word that decodes its parent. The mapping from nodes to states and configurations is illustrated in figure 19.

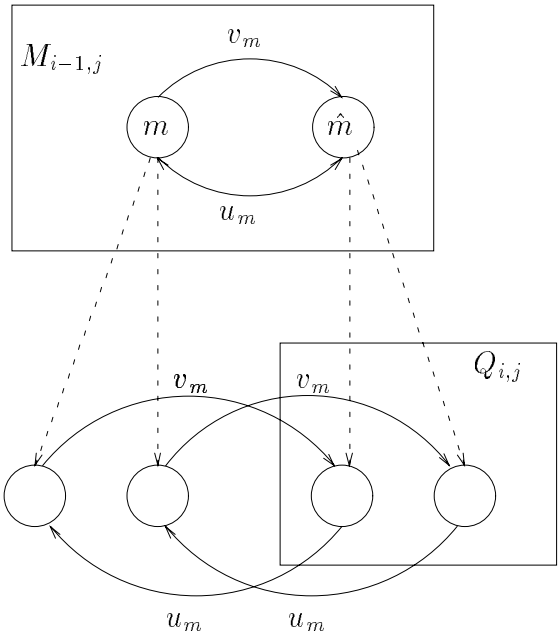

Figure 19: *Mapping nodes in M_i to states in Q_i.*

Claim 15 (B-R-Tree ⇒ Holonomy Decomposition) *From a bijection-reset TPSP tree $\mathcal{T} = (\Sigma, M, \Delta, \pi, \phi)$ of height k, one can construct a holonomy decomposition $\mathcal{C} = \mathcal{B}_1 \circ \mathcal{B}_2 \circ \cdots \circ \mathcal{B}_k$ such that $|\mathcal{T}| = |\mathcal{C}|$.*

Proof: The construction is the same as in claims 8 and 10, but this time with Q_i and ψ_i as in definition 27. The fact that it is indeed a permutation-reset decomposition follows from previous claims. In order to show that it is a holonomy decomposition we have to show that for every Q_{ij}, H_{ij} is homomorphic to some holonomy group of $\mathcal{T}$ and hence to a subgroup of $\mathcal{A}$.

Let us reproduce the definition of the transition function:

$$\delta_i(q, \langle p, \sigma \rangle) = \begin{cases} \psi_i(\Delta(\psi_i'(\langle p, q \rangle), \sigma)) & \text{if } \psi_i'(\langle p, q \rangle) \text{ is defined} \\ \perp & \text{otherwise} \end{cases} \tag{15}$$

Let $\bar{\delta}_i(p, \sigma) = p'$, $\psi'(p) = m$, $\psi'(p') = m'$ and let $\hat{m}$ be the parent of the relevant representatives. The action of $\langle p, \sigma \rangle$ on every $r \in \pi^{-1}(\hat{m})$ is $\Delta(r, u_m \sigma v_{m'}) = r' \in \pi^{-1}(\hat{m})$ (see figure 20). Thus every permutation induced by any $\langle p, \sigma \rangle$ on $Q_{i,j}$ corresponds a permutation on $\pi^{-1}(\hat{m})$. Moreover, for every two permutations $\langle p_1, \sigma_1 \rangle$ and $\langle p_2, \sigma_2 \rangle$, where $\bar{\delta}_i(p_1, \sigma_1) = p_1'$, $\psi'(p_1) = m_1$, $\psi'(p_1') = m_1'$, $\bar{\delta}_i(p_2, \sigma_2) = p_2'$, $\psi'(p_2) = m_2$ and $\psi'(p_2') = m_2'$ the action of their product $\langle p_1, \sigma_1 \rangle \cdot \langle p_2, \sigma_2 \rangle$ corresponds to the permutation induced by $u_{m_1} \sigma_1 v_{m_1'} u_{m_2} \sigma_2 v_{m_2'}$. Hence every H_{ij} is homomorphic to the holonomy group $H_{\hat{m}}$ and by claim 13 to the permutation subgroup $H_{\phi(\hat{m})}$. ∎

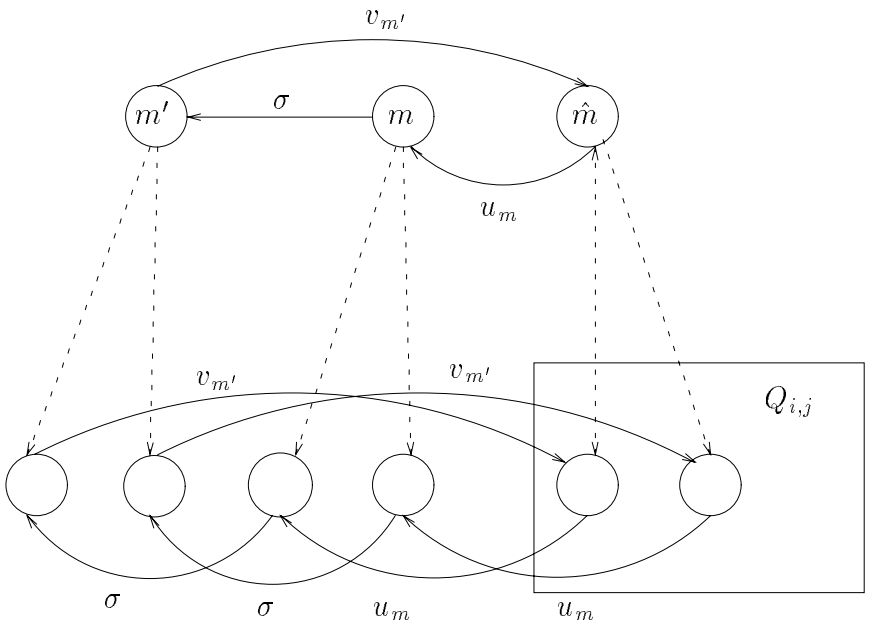

Figure 20: *The action of $\langle p, \sigma \rangle$ on $Q_{i,j}$ is the same as the permutation induced by $u_m \sigma v_{m'}$ on $\pi^{-1}(\hat{m})$.*

Corollary 16 *1) If there exists a bijection-reset TPSP tree for $\mathcal{A}$ of size c then $\mathcal{A}$ admits a Krohn-Rhodes decomposition of size c. 2) The size of any Krohn-Rhodes decomposition for $\mathcal{A}$ is at least the size of the minimal bijection-reset TPSP tree for $\mathcal{A}$.*

In section 3 we will show how to construct an exponential bijection-reset TPSP tree for any automaton, and thus reprove the primary decomposition theorem and give an upper-bound. In section 4 we present a family of automata for which the minimal bijection-reset TPSP tree is exponential, and thus give a tight lower-bound.

3 An Exponential Decomposition Algorithm

In this section we give an exponential algorithm that constructs from a given deterministic automaton, a bijection-reset TPSP tree, and according to the theory presented in section 2, it virtually produces an exponential Krohn-Rhodes decomposition. This algorithm is an automata-theoretic reformulation of the holonomy decomposition theorem of Eilenberg, a reformulation that pays attention not only to the semigroups involved, but also to the input sequences that generate them.

The algorithm will be described as a series of transformations (procedures) leading from an automaton to a bijection-reset TPSP tree via several intermediate structures. An outline of the main transformation is given below.

1. Construct a *tree subset automaton* from the original automaton (from A to TSA). The tree subset automaton is an automaton whose states correspond to some subsets of Q arranged in a tree where parenthood conforms with inclusion.
2. Compute the height of the nodes and "balance" the TSA so that it can be partitioned into levels (from TSA to BTSA).
3. Redirect the transitions so that all the transitions outgoing from a node in some level lead to nodes in the same level (from BTSA to TPSP).

3.1 Tree Subset Automata

Definition 28 (Tree Subset Automata) *: Let $\mathcal{A} = (\Sigma, Q, \delta)$ be an automaton. The tree subset automaton (TSA) associated with $\mathcal{A}$ is $\mathcal{T} = (\Sigma, M, \Delta, \pi, \phi)$ where:*

1. *M is a finite set of nodes containing a distinguished node $m^* \in M$, called the root of the tree.*
2. *$\pi : M - \{m^*\} \to M$ is the parenthood function, such that for every $m \neq m^*$, there exists some $i > 0$ such that $\pi^i(m) = m^*$.* The set of ancestors of m is denoted by $\pi^*(m)$.
3. *$\phi : M \to 2^Q$ is a labeling function whose range consists of all the singletons $\{q\} \in Q$ and all the sets $\delta(Q, w)$ for every $w \in \Sigma^*$. In particular $\phi(m^*) = Q$ and for every m, $\phi(m) \subseteq \phi(\pi(m))$.*

4. $\Delta : M \times \Sigma \to M$ *is the transition function such that for every* $m \in M$, $\sigma \in \Sigma$, $\phi(\Delta(m,\sigma)) = \delta(\phi(m),\sigma)$. *Since* $\mathcal{A}$ *is deterministic, this implies* $|\Delta(m,\sigma)| \leq |m|$. *The transition function is ancestor-consistent: for every* $m \in M - \{m^*\}$ *and* $\sigma \in \Sigma$ *there is some* $j \geq 0$ *such that* $\pi^j(\Delta(m,\sigma)) = \Delta(\pi(m),\sigma)$.

5. *For every* $m \in M$ *such that* $|\phi(m)| > 1$, $(\pi^{-1}(m), \phi)$ *is a non-redundant semi-partition.*

From A to TSA: The TSA is constructed inductively. Initially $M = \{\hat{m}\}$ and $\phi(\hat{m}) = Q$. For every node m and $\sigma \in S$, if there already exists a node $m' \in M$ such that $\phi(m') = \delta(\phi(m),\sigma)$ and $\Delta(\pi(m),\sigma) = \pi^j(m')$ for some $j \geq 0$, then we let $\Delta(m,\sigma) = m'$. Otherwise we add a node m' to M, let $\phi(m') = \delta(\phi(m),\sigma)$, let $\Delta(m,\sigma) = m'$ and update the parenthood function as follows: we define $\pi(m') = r$ where r is a node satisfying $\pi^j(r) = \Delta(\pi(m),\sigma)$ for the largest j. If there is a node r' such that $\pi(r') = r$ we change it to $\pi(r') = m'$. When the process is over we search the tree top-down and for every non-singleton m and every $q \in \phi(m) - \bigcup_{r \in \pi^{-1}(m)} \phi(r)$ we add a node m' with $\phi(m') = \{q\}$ and $\pi(m') = m$. The transitions outgoing from those singletons are calculated according to the same procedure.

An example automaton is depicted in figure 21 and its corresponding TSA in figure 22. The size of a TSA is at most $\sum_{i=1}^{n} i! = O(n!) = O(2^{n \log n})$ were n is the size of the original automaton.

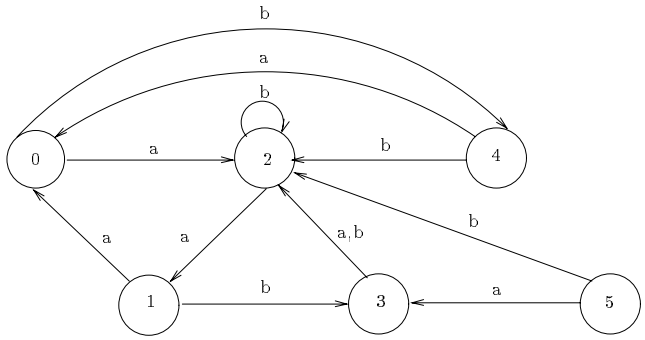

Figure 21: *The original automaton.*

3.2 Partial Order, Height and Balanced TSAs

The next step toward transforming the TSA into a TPSP tree involves the rearrangement of the nodes into levels, and then duplicating some nodes in order to make every level a semi-partition. To this end, we define a partial order on the nodes, which is, roughly speaking, the composition of Δ-reachability and ancestorhood, and then define a height function based on this order.

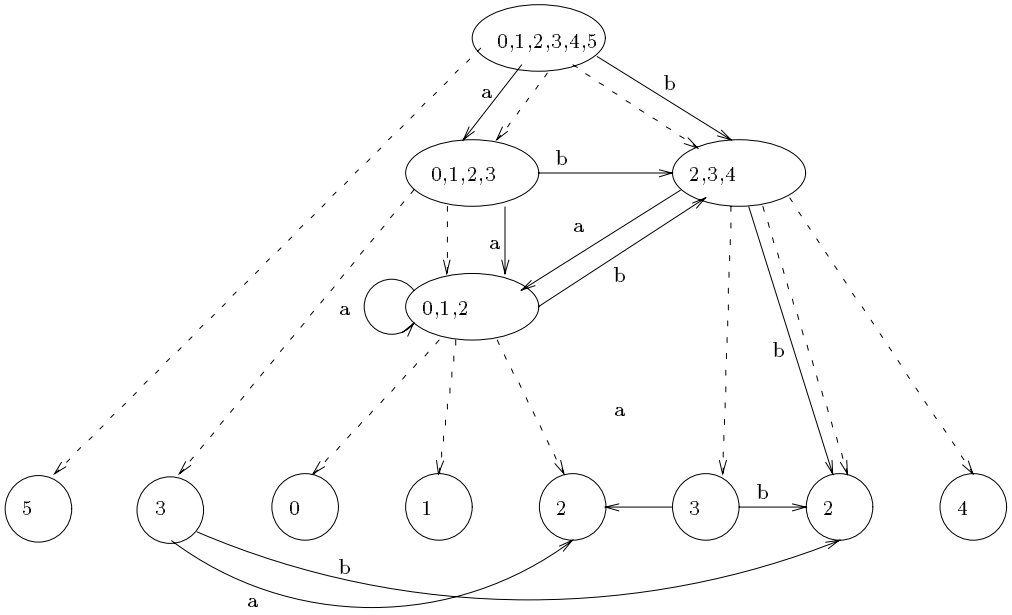

Figure 22: *The TSA for the automaton in figure 21. The names of the nodes are omitted, and the numbers indicate the subsets they are mapped to by ϕ. All the transition outgoing from singletons are omitted except those leaving $\phi^{-1}(\{3\})$ that are included in order to illustrate inherited destination.*

Definition 29 (Order) *Let $\mathcal{T} = (\Sigma, M, \Delta, \pi, \phi)$ be a TSA. We define on M the following preorder relation: $m' \preceq m$ if $\Delta(m, w) \in \pi^*(m')$ for some $w \in \Sigma^*$. We write $m \sim m'$ when both $m \preceq m'$ and $m' \preceq m$ hold, and $m \prec m'$ when it holds only in one direction.*

Clearly $m \preceq m'$ implies $|\phi(m)| \leq |\phi(m')|$, and $m \sim m'$ implies $|\phi(m)| = |\phi(m')|$. The relation $\sim$ is an equivalence relation and its equivalence classes are the maximally strongly-connected components (MSCCs) of the TSA. It can be easily verified that this definition of $\sim$ coincides with definition 24 in chapter 2.

The first definition of a height function just specifies the minimal requirements from such a function (compatibility with $\preceq$ and the absence of unnecessary "gaps").

Definition 30 (Height Function: Requirements) *A height function for a TSA $\mathcal{T} = (\Sigma, M, \Delta, \pi, \phi)$ is a function $h : M \to I\!N$, satisfying:*

1. *$h(m) = 1$ if $\phi(m)$ is a singleton and m has no children.*
2. *$m \sim m'$ implies $h(m) = h(m')$.*
3. *$m \prec m'$ implies $h(m) < h(m')$.*
4. *For every i, $1 \leq i \leq h(m^*)$, $h^{-1}(i) \neq \emptyset$.*

Definition 31 (Smallest Height Function) *The smallest height function for a TSA $\mathcal{T}$ is defined for every $m \in M$ as $h(m) = l$ where l is the length of the longest chain of the form $m_1 \prec \ldots \prec m_l \prec m'$ where m' is any node satisfying $m \sim m'$. The height of a TSA is $h(m^*)$.*

The TSA of figure 22 arranged according to h appears in figure 23

Computing the Height: The algorithm works by sueccesively creating subsets of M, $L_1, \ldots L_h$ where L_i corresponds to all nodes of height i. Initially $L_1 = \{m : |\phi(m)| = 1\}$ After having computed L_i, we let $L_{i+1} = \{\pi(m) : m \in L_i\}$. Then for every $m \in L_{i+1}$ if there is some m' such that $m' \notin \bigcup_{j=1}^{i} L_j$, $m \not\sim m'$ and either $\pi(m') = m$ or $\Delta(m, \sigma) = m'$ then we remove m from L_{i+1}. This process is iterated until no updating takes place and L_{i+1} stabilizes. The complexity of this computation is bounded by the height of the tree which is bounded by its size.

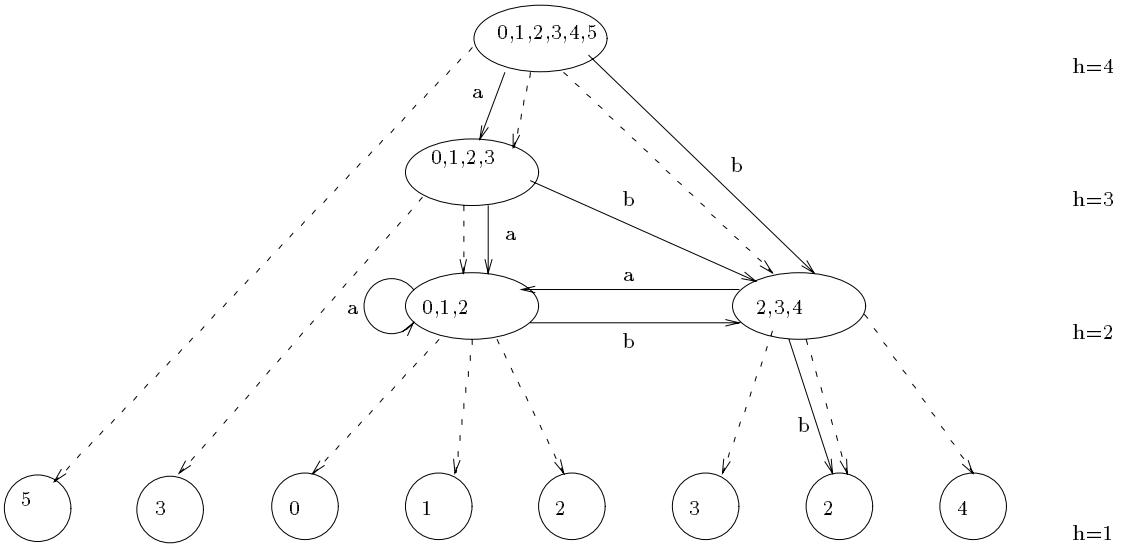

Figure 23: *The TSA of figure 22 rearranged according to the smallest height function.*

The tree may contain some height gaps along the parenthood chains, i.e., $h(\pi(m)) - h(m) > 1$. We want to "balance" the tree by adding copies of m between itself and its ancestor, such that that every node of height j will have all its children with height equal to $j - 1$.

Definition 32 (Balanced TSA) *A TSA $\mathcal{T} = (\Sigma, M, \Delta, \pi, \phi)$ is balanced (with respect to a height function h) if for every $m \in M - \{\hat{m}\}$, $h(\pi(m)) = h(m) + 1$.*

Balancing a TSA: The construction is rather simple. The TSA is traversed bottom-up. Whenever a jump is detected, that is, a node m with $\pi(m) = m'$ and $h(m') - h(m) > 1$, we insert a new node r to M, and let $\phi(r) = \phi(m)$, $\pi(m) = r$ and $\pi(r) = m'$. For every

$\sigma \in \Sigma$ we let $\Delta(r,\sigma) = \Delta(m,\sigma)$. We can partition M into levels $M_1, M_2, \ldots, M_{h(m^*)+1}$ by letting $M_i = \{m \in M : h(m) = h(m^*) - i + 1\}$. The balanced TSA for our ongoing example appears in figure 24. Since no state needs to be duplicated more than $h(m^*)$ times, the size of the balanced TSA (BTSA) is polynomial in the size of the TSA.

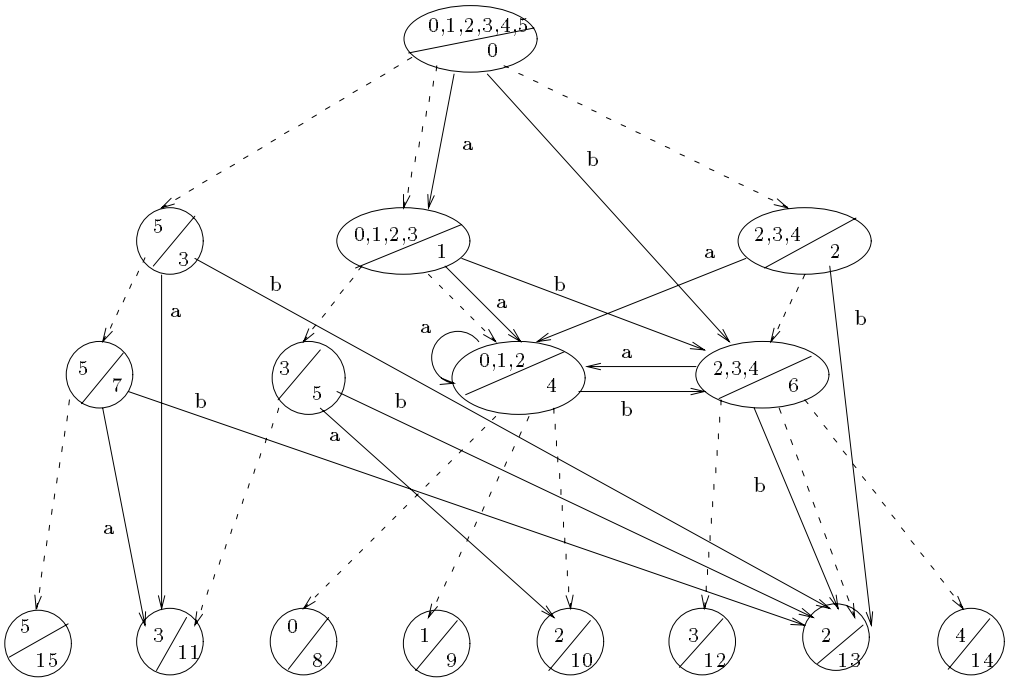

Figure 24: *A BTSA for the TSA in figure 23. The names of the nodes appear in the lower parts.*

3.3 Constructing Bijection-Reset TPSP Trees

A BTSA already satisfies conditions 1-5 in the definition of a TPSP tree (definition 18). The last step needed is to modify the transition-function such that it will be level-preserving, and satisfy definition 25. The latter means for every level M_i is a TPSP of $\mathcal{A}$, and for every $m, m' \in M_i$, such that $\Delta(m,\sigma) = m'$, either σ induces a reset from $\pi^{-1}(m)$ to some $r' \in \pi^{-1}(m')$ or $m \sim m'$ and σ induces an injection from $\pi^{-1}(m)$ to $\pi^{-1}(m')$. The following construction transforms a BTSA $\mathcal{T}' = (\Sigma, M, \Delta', \pi, \phi)$ into a TPSP tree $\mathcal{T} = (\Sigma, M, \Delta, \pi, \phi)$ by replacing Δ' by an appropriate transition function Δ. For every $m \in M_i$ and $j < i$ we use $\pi_j(m) = \pi^*(m) \cap M_j$ to denote the ancestor of m in level j.

From BTSA to TPSP Tree: We scan the tree top-down, level by level. The transition function is defined for every i, $1 \leq i \leq h(m^*) + 1$, $m \in M_i$ and $\sigma \in \Sigma$, as:

$$\Delta_i(m,\sigma) = \pi_i(\Delta'(m,\sigma)) \tag{16}$$

In other words the transitions are "lifted" until they become horizontal. The result of applying this transformation to the BTSA of figure 24 appears in figure 25, and the corresponding decomposition in figure 26.

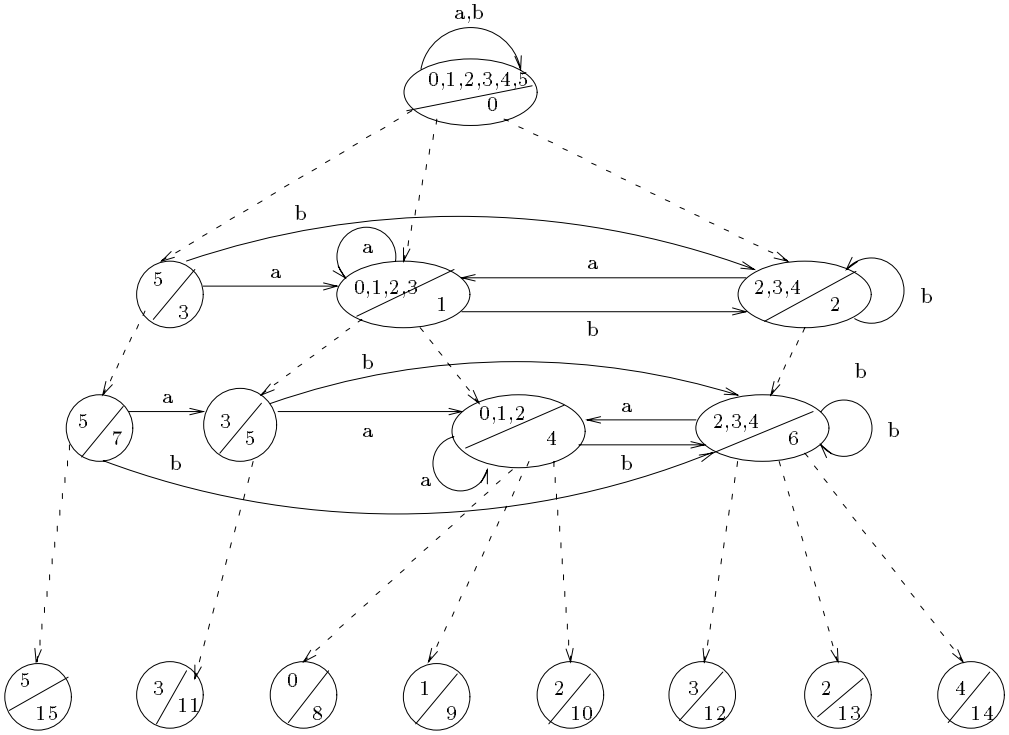

Figure 25: *The bijection-reset TPSP tree constructed from the BTSA of figure 24.*

Claim 17 *Let $\mathcal{T} = (\Sigma, M, \Delta, \pi, \phi)$ be the result of the above transformation. For every i, $(\Sigma, M_i, \delta_i, \phi_i)$ is a TPSP for $\mathcal{A}$.*

Proof: It remains to show that Δ is transition preserving, that is,

$$\delta(\phi(m), \sigma) \subseteq \phi(\Delta_i(m, \sigma)) \tag{17}$$

By substituting the definition of Δ we get

$$\delta(\phi(m), \sigma) \subseteq \phi(\pi_i(\Delta'(m, \sigma))) \tag{18}$$

Since every TSA satisfies

$$\phi(\Delta'(m, \sigma)) = \delta(\phi(m), \sigma) \tag{19}$$

the transition-preservation condition now becomes

$$\phi(\Delta'(m, \sigma)) \subseteq \phi(\pi_i(\Delta'(m, \sigma))) \tag{20}$$

which is true because for every m, $\phi(m) \subseteq \phi(\pi_i(m))$. ∎

Claim 18 *For every $m, m' \in M_{i-1}$, such that $\Delta(m, \sigma) = m'$, either σ induces a reset from $\pi^{-1}(m)$ to some $r' \in \pi^{-1}(m')$ or $m \sim m'$ and σ induces an injection from $\pi^{-1}(m)$ to $\pi^{-1}(m')$. Hence, $\mathcal{T}$ is a bijection-reset TPSP tree for $\mathcal{A}$.*

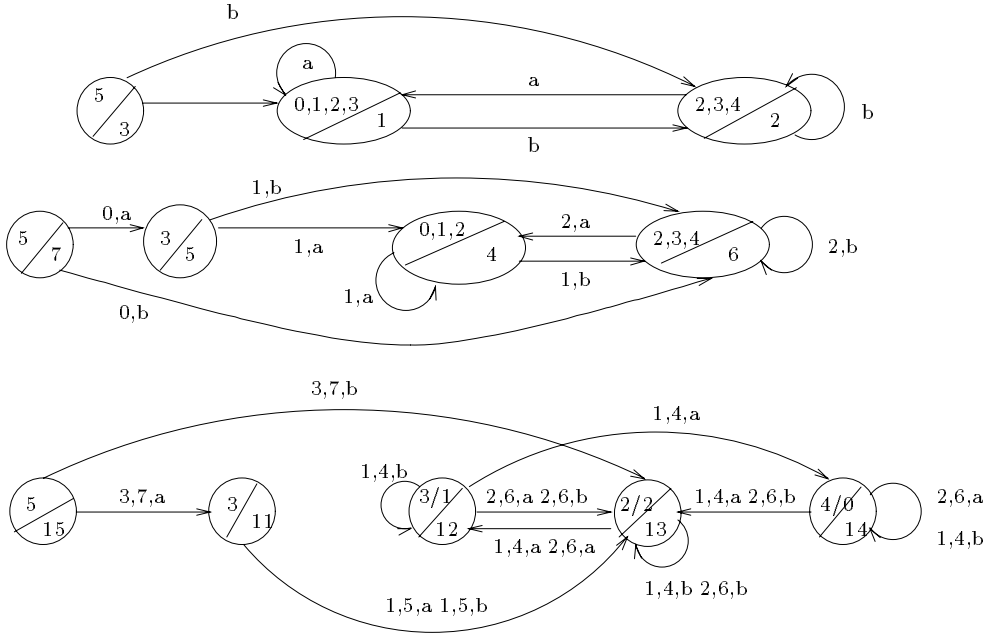

Figure 26: *The cascade product for the automaton of figure 21, constucted from the TPSP tree in figure fig:tpsp. Note that $\langle 2, 6, a\rangle$ is a modulo-2 counter and that $\langle 1, 4, a\rangle$ is a modulo-3 counter.*

Proof: Let m belong to M_{i-1}. For every $r \in \pi^{-1}(m)$, we have $\Delta(r, \sigma) = \pi_i(\Delta'(r, \sigma))$. We consider two cases according to the relation between m and $\Delta'(m, \sigma)$ in the BTSA: 1) Suppose $\Delta'(m, \sigma) \prec m$. This implies that in the BTSA there must be a node corresponding to $\Delta'(m, \sigma)$ at M_j for some $j \geq i$. Consequently, for every $r \in \pi^{-1}(m)$, a node corresponding to $\Delta(r, \sigma)$ cannot exist before M_{i+1}. Thus $\pi_i(\Delta'(r, \sigma))$ is the same for every $r \in \pi^{-1}(m)$ and σ is a reset. 2) Suppose $\Delta'(m, \sigma) = m' \sim m$ and consequently $\delta(\phi(m), \sigma) = \phi(m')$. In this case σ induces an injection (in the TSA) from $\pi^{-1}(m)$ to $\pi^{-1}(m')$ such that for every $r \in \pi^{-1}(m)$ there exists $r' \in \pi^{-1}(m')$ satisfying $\Delta'(r, \sigma) = r'$ and $h(r) = h(r')$. Every rearrangement of $\pi^{-1}(m)$ according to h will have its counterpart in $\pi^{-1}(m')$ so that σ will remain an injection. ∎

Corollary 19 *There exist an exponential algorithm that constructs for any automaton an exponential bijection-reset TPSP tree.*

Corollary 20 (Upper Bound) *There exist an exponential algorithm that decomposes any automaton into an exponential holonomy cascaded decomposition.*

4 An Exponential Lower-Bound for the Cascaded Decomposition

In this section we prove our main original result concerning the complexity of cascaded decomposition of automata. Throughout this chapter we consider holonomy decomposi-

tions satisfying definition 22. We also use the definitions of configurations (definition 14) and of their mappings into subsets via φ_i (definition 15).

We assume in this section that the original automaton $\mathcal{A}$ is counter-free and strongly-connected. Some of the claims in the sequel are valid in more general setting, but the family of automata used for proving the lower-bound satisfies these properties. A holonomy decomposition for such automata consists of reset-identity components.

The proof is essentially a proof of a lower-bound on the size of the minimal configuration tree for a certain family of automata. Alternatively we could develop it in terms of the minimal bijection-reset TPSP tree for this family but we preferred the configuration tree terminology because its relation to the decomposition is more direct. This choice makes this section almost self-contained.

4.1 Auxiliary Properties

The following claim sets lower bounds on $\varphi_i(p)$ based on incoming transitions.

Claim 21 (Necessary Configurations) *(1) If $p \in P_i$ is a configuration such that $\bar{\delta}(p,w) = p'$ for some $w \in \Sigma^*$ then $\delta(\varphi_i(p), w) \subseteq \varphi_i(p')$. (2) Let $p \in P_i$ be a configuration such that $\bar{\delta}(p,\sigma) = p'$. If the letter $\langle p, \sigma \rangle$ induces a reset in $\mathcal{B}_{i+1}$ whose destination is some $r' \in Q_{i+1}$, then $\delta(\varphi_i(p), \sigma) \subseteq \varphi_{i+1}\langle p', r' \rangle$.*

Proof: Both claims follow immediately from the homomorphism from $\mathcal{C}$ to $\mathcal{A}$ and the definition of φ_i. ∎

The next claim relates permutations in $\mathcal{B}_i$ with bijections in $\mathcal{C}_i$

Claim 22 (Permutations and Bijections) *(1) If $\langle p, \sigma \rangle$ induces a permutation[10] on Q_{i+1} in $\mathcal{B}_{i+1}$ then σ is an injection on $\{p\} \times Q_{i+1}$ in $\mathcal{C}_{i+1}$. (2) If in addition $\bar{\delta}(p,\sigma) = p$ in $\mathcal{C}_i$ then σ is a bijection on $\{p\} \times Q_{i+1}$ in $\mathcal{C}_{i+1}$. (3) If in addition $\langle p, \sigma \rangle$ induces a trivial permutation (identity) on Q_{i+1} in $\mathcal{B}_{i+1}$ then σ is an identity on $\{p\} \times Q_{i+1}$ in $\mathcal{C}_{i+1}$.*

Proof: (1) Since there are no $r, r' \in Q_{i+1}$ that are mapped by $\langle p, \sigma \rangle$ to the same state, there are no $\langle p, r \rangle, \langle p, r' \rangle \in P_{i+1}$ that are mapped to the same configuration by σ. Claims (2) and (3) are direct corollaries of (1). ∎

4.2 Redundancy Elimination

In order to simplify the proof we introduce three reduction principles that enable us to eliminate obvious redundancies from a decomposition without affecting its properties. All the three principles are of the form: *Given a decomposition $(\mathcal{C}, \varphi)$ such that $\mathcal{A} \leq_\varphi \mathcal{C}$, there exists a "less redundant" decomposition $(\mathcal{C}', \varphi')$ such that $\mathcal{A} \leq_{\varphi'} \mathcal{C}' \leq \mathcal{C}$ and $|\mathcal{C}'| \leq |\mathcal{C}|$.*

[10] We mean a partial permutation on a subset of Q_{i+1} on which $\langle p, \sigma \rangle$ is defined.

Definition 33 (Core) *Let $\mathcal{C}_i$ be a partial decomposition. A subset $P'_i \subseteq P_i$ is called a core of $\mathcal{C}_i$ if $\bigcup_{p \in P'_i} \varphi_i(p) = Q$ and P'_i is closed under transitions, i.e., $\bar{\delta}_i(P'_i, \Sigma) \subseteq P'_i$. A minimal core is a core such that none of its subsets is a core.*

The following reduction principle states that one can restrict a decomposition to a core and discard the rest, since the configurations that extend the core "cover" all the states in Q.

Claim 23 (Reduction Principle I) *Let $\mathcal{C} = \mathcal{C}_i \circ \mathcal{B}_{i+1} \circ \ldots \circ \mathcal{B}_k$ be a decomposition such that $\mathcal{A} \leq_\varphi \mathcal{C}$ for some φ. There exists a decomposition $\mathcal{C}' = \mathcal{C}'_i \circ \mathcal{B}'_{i+1} \circ \ldots \circ \mathcal{B}'_k$ and φ' such that $\mathcal{A} \leq_{\varphi'} \mathcal{C}' \leq \mathcal{C}$ and $\mathcal{C}'_i$ is a restriction of $\mathcal{C}_i$ to a core P'_i. Moreover for all j, $i \leq j \leq k$ we have $\mathcal{B}'_j \leq \mathcal{B}_j$ and $\mathcal{B}'_j$ is also a reset-identity automaton.*

Proof: Let P'_i be a core of $\mathcal{C}_i$. For all $j \geq i$, $p \in P_i$ and $\langle p, r \rangle \in P_j$ we let $\varphi'_j(\langle p, r \rangle) = \varphi_j(\langle p, r \rangle)$ if $p \in P'_i$ and $\varphi'_j(\langle p, r \rangle) = \emptyset$ otherwise. Every $\mathcal{B}'_j$ will consist of a restriction of $\mathcal{B}_j$ to the alphabet $P'_i \times Q_{i+1} \times \ldots \times Q_{j-1} \times \Sigma$. What we have to show is that $\varphi' = \varphi'_k$ is a homomorphism (transition-preserving and surjective) and that $\mathcal{B}'_{i+1}, \ldots, \mathcal{B}'_k$ are reset-identities. Since $P' = P'_k$ is also a core, the range of φ' is Q. Now $\delta(\varphi(\langle p, q_{i+1}, \ldots, q_k \rangle), \sigma) = \varphi(\bar{\delta}(\langle p, q_{i+1}, \ldots, q_k \rangle, \sigma))$ implies $\delta(\varphi(\langle p, q_{i+1}, \ldots, q_k \rangle)), \sigma) = \varphi'(\bar{\delta}(\langle p, q_{i+1}, \ldots, q_k \rangle, \sigma))$ because either both sides are empty for $p \notin P'_i$, or φ' coincides with φ (recall that $\bar{\delta}(p, \sigma) \in P'_i$ because P'_i is closed under transitions). Finally, every $\mathcal{B}'_j$ is a reset-identity automaton because every $\mathcal{B}_j$ is, and the removal of transitions does not affect this property. ∎

According to claim 21, $\delta(\varphi_i(p), w) \subseteq \varphi_i(p')$ for every w such that $\bar{\delta}(p, w) = p'$. The following reduction principle complements this claim by stating that $\varphi_i(p)$ can be reduced to the minimal subset implied by claim 21.

Claim 24 (Reduction Principle II) *Let $\mathcal{C} = \mathcal{B}_1 \circ \ldots \circ \mathcal{B}_k$ be a decomposition such that $\mathcal{A} \leq_\varphi \mathcal{C}$. There exists a decomposition $\mathcal{C}' = \mathcal{B}'_1 \circ \ldots \circ \mathcal{B}'_k$ and φ' such that $\mathcal{A} \leq_{\varphi'} \mathcal{C}' \leq \mathcal{C}$ and for every i, $1 \leq i \leq k$, and every $p \in P_i$,*

$$\varphi'_i(p) = \bigcup_{\{p' \in P'_i, \sigma \in \Sigma : \bar{\delta}_i(p', \sigma) = p\}} \delta(\varphi'_i(p'), \sigma) \tag{21}$$

where P'_i is the configuration space of $\mathcal{C}'_i$. Moreover for every j, $i \leq j \leq k$ we have $\mathcal{B}'_j \leq \mathcal{B}_j$ and $\mathcal{B}'_j$ is also a reset-identity automaton.

Proof: Suppose condition (21) is not satisfied by φ_i. This means that there exists some $\langle q_1, \ldots, q_k \rangle \in P_k$ which is not reachable in $\mathcal{C}$ such that $\varphi(\langle q_1, \ldots, q_k \rangle) = q$. So we let $\varphi'_k(\langle q_1, \ldots, q_k \rangle) = \emptyset$ and for every $j, i \leq j < k$ recalculate φ'_j upward. As a result $\varphi'_j(\langle q_1, \ldots, q_j \rangle) = \varphi_j(\langle q_1, \ldots, q_j \rangle) - \{q\}$. Then we eliminate all the transitions of the form $(\langle q_j, (q_1, \ldots, q_{j-1}, \sigma \rangle))$ from δ_j whenever $\varphi'_j(\langle q_1, \ldots, q_j \rangle) = \emptyset$. As in claim 23, φ' is still a homomorphism and the components remain reset-identities. ∎

Note that by applying this procedure repeatedly to all the elements of P_i, P_k is reduced to strongly-connected components.

Claim 25 (Reduction Principle III) *Let* $\mathcal{C} = \mathcal{B}_1 \circ \ldots \mathcal{B}_i \circ \mathcal{B}_{i+1} \ldots \mathcal{B}_k \geq_\varphi \mathcal{A}$ *be a decomposition such that* $\mathcal{C}_i$ *is isomorphic to* $\mathcal{C}_{i+1}$*. Then there exists another decomposition* $\mathcal{C}' = \mathcal{B}_1 \circ \ldots \mathcal{B}_i \circ \mathcal{B}_{i+2} \ldots \mathcal{B}_k \geq_{\varphi'} \mathcal{A}$*.* (In other words, $\mathcal{B}_{i+1}$ is eliminated).

Proof: Obviously if $\mathcal{C}_i \circ \mathcal{B}_{i+1}$ is isomorphic to C_i then $\mathcal{B}_{i+1}$ is a trivial one-state automaton whose corresponding coordinate adds no information. ∎

4.3 Bounded-Counters

Now we introduce the family of automata for which we prove the lower-bound.

Definition 34 (Bounded-Counters) *For* $n > 0$ *and* $i, j \leq n$ *we let*

$$\begin{array}{rcl} i \overset{n}{+} j & = & \min(i+j, n) \\ i \overset{n}{-} j & = & \max(i-j, 1) \end{array} \tag{22}$$

An n*-bounded-counter*[11] *is an automaton* $\mathcal{A}^{(n)} = (\Sigma, Q, \delta)$ *where* $\Sigma = \{a, b\}$*,* $Q = \{q_1, \ldots, q_n\}$ *and*

$$\begin{array}{rcl} \delta(q_j, a) & = & q_{j \overset{n}{-} 1} \\ \delta(q_j, b) & = & q_{j \overset{n}{+} 1} \end{array} \tag{23}$$

The automaton $\mathcal{A}^{(4)}$ is depicted in figure 27 as an example.

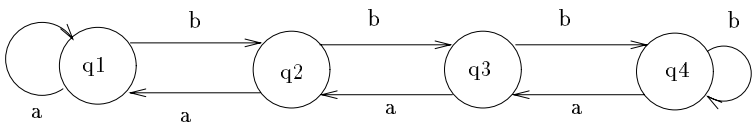

Figure 27: *The automaton* $\mathcal{A}^{(4)}$*.*

Definition 35 (Consecutive Subsets) *Let* $Q = \{q_1, \ldots, q_n\}$ *be the set of states of a bounded-counter* $\mathcal{A}^{(n)}$*. A consecutive subset of* Q *is a set* $\{q_j, q_{j+1}, \ldots, q_k\}$*,* $j \leq k$*, denoted by* $Q_{[j..k]}$*. We define two functions on the set of consecutive subsets of* Q*, for* $j < k$*:*

$$\begin{array}{rcl} L(Q_{[j..k]}) & = & Q_{[j..k-1]} \\ R(Q_{[j..k]}) & = & Q_{[j+1..k]} \end{array} \tag{24}$$

Obviously the consecutive subsets are closed under applications of L *and* R*.*

Claim 26 *The family* $\mathcal{A}^{(n)}$ *is counter-free for every* n*.*

[11] In other places this automaton has been called the *bounded buffer* or the *elevator* automaton.

Proof: This will be done by showing that in any member of $\mathcal{A}^{(n)}$, if $\delta(q_i, w) = q_j$ and $\delta(q_j, w) = q_k$ then either $i \leq j \leq k$ or $i \geq j \geq k$ hold. This is true for $w \in \{a, b\}$ and by induction on the length of w, it is true for every w. Now if some w permutes $\{q_{i_1}, q_{i_2}, \ldots, q_{i_m}\}$ then we must have either $i_1 \leq i_2 \leq \ldots \leq i_m \leq i_1$ or $i_1 \geq i_2 \geq \ldots \geq i_m \geq i_1$ which can be satisfied only if all the indices are identical and the permutation is trivial. ∎

The next claim implies the existence of necessary configurations in decompositions of bounded-counters.

Claim 27 *Let $p \in P_i$ be a configuration in a decomposition for a bounded-counter. If $\varphi_i(p) \neq \emptyset$ and $\bar{\delta}_i(p, a) = \bar{\delta}_i(p, b) = p$ then $\varphi_i(p) = Q$.*

Proof: Suppose $q_j \in \varphi_i(p)$ for some j, $1 \leq j \leq n$. Then $Q_{[1..j-1]} \subseteq \delta(\varphi_i(p), a)$ and $Q_{[j+1..n]} \subseteq \delta(\varphi_i(p), b)$. Since both $\delta(\varphi_i(p), a) \subseteq \varphi_i(p)$ and $\delta(\varphi_i(p), b) \subseteq \varphi_i(p)$ must hold, we have $\varphi_i(p) = Q_{[1..j-1]} \cup \{q_j\} \cup Q_{[j+1..n]} = Q$. ∎

4.4 Proof of the Lower-Bound

Claim 28 (The Lower Bound) *For every $n \in \mathbb{N}$, the smallest reset-identity decomposition $(\mathcal{C}, \varphi)$ such that $\mathcal{A}^{(n)} \leq_\varphi \mathcal{C}$ is $\mathcal{C} = \mathcal{B}_0 \circ \mathcal{B}_1 \circ \mathcal{B}_2 \circ \cdots \circ \mathcal{B}_{n-1}$, satisfying for every i, $0 \leq i \leq n-1$:*

1. *$Q_i = \{0, 1\}$ and consequently $P_i = \{0, 1\}^i$.* (We let $\{0, 1\}^0 = p^*$).
2. *$\varphi_0(p^*) = Q$, and for every $p \in P_i$, $\varphi_{i+1}(\langle p, 0\rangle) = L(\varphi_i(p))$ and $\varphi_{i+1}(\langle p, 1\rangle) = R(\varphi_i(p))$.* This implies that the range of φ_i is the set of consecutive subsets of size $n - i$.
3. *$\langle 0^{i-1}, a\rangle$ and $\langle 1^{i-1}, b\rangle$ are resets in $\mathcal{B}_i$ to 0 and 1 respectively.* In terms of configurations this means that $\bar{\delta}_i(0^i, a) = 0^i$ and $\bar{\delta}_i(1^i, b) = 1^i$. *The rest of the letters induce identities.*
4. *For every $p, p' \in P_i$ there exist $w \in \Sigma^*$ and l, $-i \leq l \leq i$ such that $\bar{\delta}_i(p, w) = p'$, and for every $q_j \in \varphi_i(p)$, $\delta(q_j, w) = q_{j+l} \in \varphi_i(p')$.* In other words, there exists an order-preserving bijection between $\varphi_i(p)$ and $\varphi_i(p')$.

Proof: The proof is by induction on i:

- *Base case* ($i = 1$): Let $\mathcal{C}$ be a decomposition and $\mathcal{B}_1$ the first non-trivial automaton. There are four possibilities concerning the transformations induced by a and b:
  1. Both a and b are identities.
  2. One of $\{a, b\}$ is a reset and the other is an identity.

3. Both a and b are resets to the same state.
4. Both a and b are resets to different states.

The first three possibilities imply the existence of at least one configuration $p \in P_1$ such that $\varphi_1(p) \neq \emptyset$ and $\bar{\delta}_1(p,a) = \bar{\delta}_1(p,b) = p$. According to claim 27, $\varphi_1(p) = Q$. Since $\{p\}$ is a core, by using claim 23 we can reduce $\mathcal{C}$ into a smaller decomposition where $\mathcal{B}_1$ is restricted to $\{p\}$. But now $\mathcal{B}_1$ is isomorphic to $\mathcal{B}_0$ and according to claim 25, it can be removed from the decomposition. This can be repeated until $\mathcal{B}_1$ satisfies possibility 4. We denote the destinations of a and b by 0 and 1 respectively. According to claim 24 we may assume $\varphi(0) = Q_{[1..n-1]}$ and $= \varphi(1) = Q_{[2..n]}$. Since $\{0,1\}$ is a core, we can discard the rest of the states and remain with $\mathcal{B}_1$ as the two-state reset automaton. Since $\delta(\varphi_1(0),b) = \varphi_1(1)$ and $\delta(\varphi_1(1),a) = \varphi_1(0)$, $\mathcal{B}_1$ is strongly-connected by bijections and $\mathcal{B}_1$ satisfies the inductive hypothesis.

- *Inductive Case*: Suppose it is true for $\mathcal{C}_i$. We consider two cases concerning $\mathcal{C}_{i+1}$:

  1. If $\langle 0^i, a\rangle$ is an identity in $\mathcal{B}_{i+1}$ then we have for every $q \in Q_{i+1}$, $\bar{\delta}_{i+1}(\langle 0^i, q\rangle, a) = \langle 0^i, q\rangle$. In particular their is some $q \in Q_{i+1}$ such that $q_{n-i} \in \phi_{i+1}(\langle 0^i, q\rangle)$ and consequently $\varphi_{i+1}(\langle 0^i, q\rangle = \varphi_i(0^i) = Q_{[1..n-i]}$. By the inductive hypothesis, for every $p' \in P_i$, there exist bijections from $\varphi_i(0^i)$ to $\varphi_i(p')$ and vice versa, hence there exists some $q' \in Q_{i+1}$ such that $\varphi_{i+1}\langle p', q'\rangle = \varphi_i(p')$. Thus P_{i+1} contains a core isomorphic to P_i and $\mathcal{B}_{i+1}$ can be discarded.
  2. If $\langle 0^i, a\rangle$ is a reset in $\mathcal{B}_{i+1}$ then, according to claims 21 and 24, there exists a configuration $\langle 0^i, q\rangle \in P_{i+1}$ such that $\varphi_{i+1}(\langle 0^i, q\rangle) = \delta(\varphi_i(0^i), a) = L(\varphi_i(0^i))$. Any word that induces a bijection between two subsets Q' and Q'' induces a bijection between $L(Q')$ and $L(Q'')$, so for every $p \in P_i$ there exists a configuration $\langle p, 0\rangle \in P_{i+1}$ such that $\varphi_{i+1}(\langle p, 0\rangle) = L(\varphi_i(p))$. Thus all the configurations in $P_i \times \{0\}$ are strongly-connected by bijections. In a similar way, the reset $\langle 1^i, b\rangle$ implies the existence of the strongly connected configurations $P_i \times \{1\}$. Finally the transitions $\bar{\delta}_{i+1}(\langle 0^i, 1\rangle, a) = 0^{i+1}$ and $\bar{\delta}_{i+1}(\langle 1^i, 0\rangle, b) = 1^{i+1}$ make all the configuration strongly-connected by bijections.

  Thus, if $\mathcal{C}_i$ satisfies the inductive hypothesis so does $\mathcal{C}_{i+1}$

This concludes the proof. ∎

Corollary 29 (Lower Bound) *There exists a family $\{\mathcal{A}^{(n)}\}_{n\in\mathbb{N}}$ of automata such that for every n, the size of $\mathcal{A}^{(n)}$ is $2n$, and the smallest decomposition for $\mathcal{A}$ is $\mathcal{C} = \mathcal{B}_0 \circ \mathcal{B}_2 \circ \cdots \circ \mathcal{B}_{n-1}$, having $O(2^n)$ transitions.*

The decomposition and the configurations for $\mathcal{A}^{(4)}$ are given in figures 28 and 29.

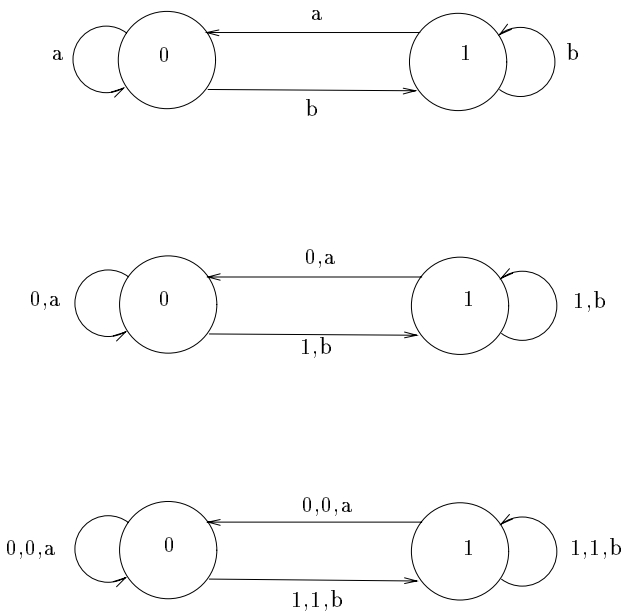

Figure 28: *The decomposition for $\mathcal{A}^{(4)}$. The letters that induce identities are omitted*

4.5 Discussion

What is the relevance of this result? After all, the components of the cascade are very simple and their transition functions can be expressed by very short boolean formulae. The first answer to this is that the "rules of the game" of computational complexity are obeyed, that is, the same complexity measure (number of transitions) is used for the input and for the exponentially blown-up output. Bounded-counters admit a very simple representation (logarithmic at most) if bounded addition and subtraction are taken as a basis. In some sense, bounded-counters are "almost" counters, almost beyond the reach of identity-reset decompositions and propositional temporal logic formulae. This "explains" why they are hard to describe using these formalisms.

More importantly, our result implies that some states of $\mathcal{A}$ *must be encoded by exponentially many different configurations.* Considering the procedure of transforming automata into past temporal logic formulae (chapter 5), a formula for describing the words that lead to some $q \in Q$ is a disjunction of the respective formulae that express the sequences leading to the various configurations in $\varphi^{-1}(q)$. This is not, of course, a general exponential lower bound on the size of the temporal formula that expresses the language accepted by an automaton, but it indicates an inherent complexity in bounded-counters, due to the variety of different "classes" of input histories that could lead to some state. Our conjecture is that an exponential lower-bound on the size of the minimal temporal formula for bounded-counters can be established as well.

A final remark concerning the relation to other computational complexity issues: In the Krohn-Rhodes decomposition, as we have seen in chapter 2, counters which are induced by *arbitrary words* in the original automaton, are induced by the *generating letters*

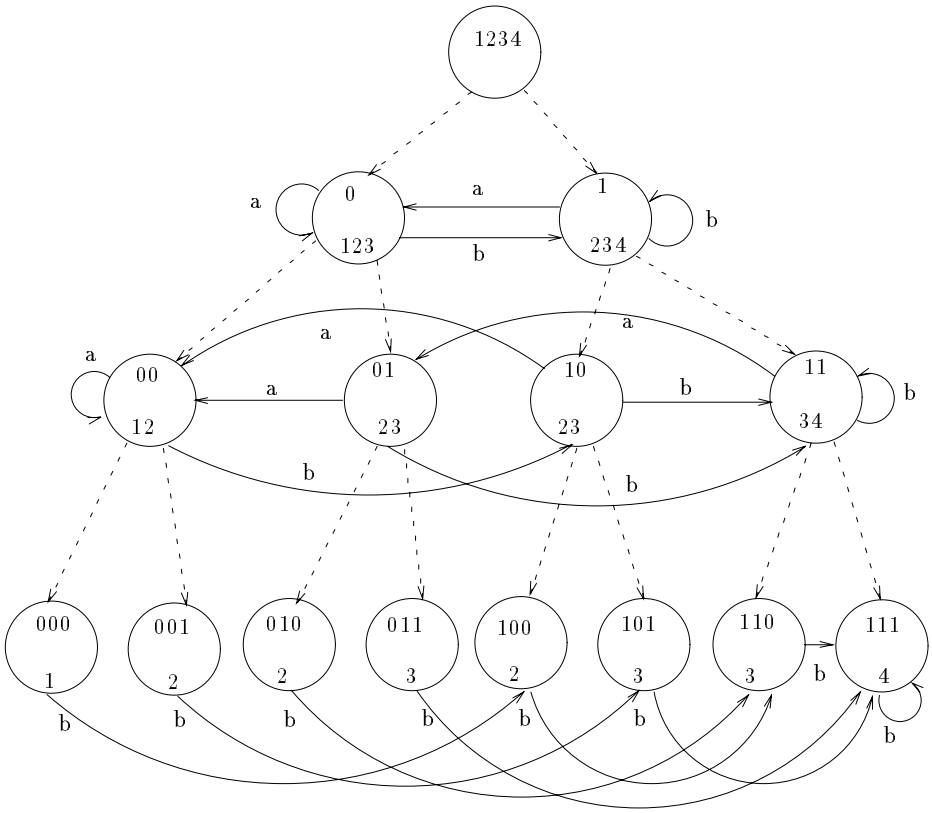

Figure 29: *The configuration tree for $\mathcal{A}^{(4)}$. The configuration "name" appears on top while its φ_i on the bottom of every node. The a-transitions in the lowest level are omitted for clarity.*

in the cascade components. Thus, in order to check whether a cascaded decomposition contains counters, it is sufficient to scan the transition table of the cascade – a polynomial procedure. On the other hand, Stern has shown ([Ste85]) that to check whether an automaton is counter-free is NP-hard. If we had a polynomial procedure for decomposition, we would have a polynomial algorithm for this problem and prove P=NP. Moreover, according to [CH89], automaton counter-freedom is even PSPACE-hard, which further decreases the a-priori likeliness of a polynomial decomposition.

5 Translating Automata to Temopral Logic

Logic is an alternative formalism that can be used for defining sequences and sets of sequences. The underlying idea is that a logical formula can stand for the set of all sequences that satisfy it. Propositional Temporal Logic is a modal logic that is naturally suitable for this task. It has been introduced to the computer science literature in ([Pn77]) and ever since has played a major role in many specification and verification systems (see the best-seller [MP91] for exciting details).

As an application of the algorithm given in section 3 we show how to translate a star-free regular set, represented by an accepting non-counting automaton, into a formula in past temporal logic whose size is at most exponential in the size of the original automaton. This result is extended naturally for translating an ω-regular set, represented by an accepting non-counting ω-automaton, into a formula in future temporal logic. The syntax and the semantics of the temporal logic used in this section are defined below.

5.1 Past Temporal Logic

We consider past temporal logic as a formalism for specifying properties of finite sequences.

Definition 36 (Past Temporal Formulae) *Let Σ be an alphabet. A past temporal formula is defined inductively as:*

1. *σ is a past temporal formula for every $\sigma \in \Sigma$.*
2. *If Φ and Ψ are formulae, so are $\Phi \vee \Psi$, $\neg\Phi$, $\ominus\Phi$ (previously Φ) and $\Phi\,\mathcal{S}\,\Psi$ (Φ since Ψ).*

The rest of the Boolean and temporal operators can be derived from $\vee$, $\neg$, $\ominus$ and $\mathcal{S}$. The set of all such formulae is denoted $TLP(\Sigma)$.

Definition 37 (Models for Past Formulae) *A model for a past temporal formula Π is a word*

$$w = s_1, \ldots, s_m$$

where $s_i \in \Sigma$, for $i = 1, \ldots, m$, and $m \geq 0$. We refer to m as the length *of w and write $|w| = m$.*

Given a word w, we define the notion of a temporal formula holding at position j, $0 \leq j \leq |w|$.

Definition 38 (Satisfiability of Past Formulae) *Let Φ and Ψ be past formulae. Satisfiability at position j, $1 \leq j \leq |w|$, in a word w is defined by:*

$$\begin{array}{lcl}
(w,j) \models \sigma & \iff & j > 0 \text{ and } s_j = \sigma \\
(w,j) \models \neg\Phi & \iff & (w,j) \not\models \Phi \\
(w,j) \models \Phi \vee \Psi & \iff & (w,j) \models \Phi \text{ or} \\
 & & (w,j) \models \Psi \\
(w,j) \models \ominus\Phi & \iff & (w,j-1) \models \Phi \\
(w,j) \models \Phi\,\mathcal{S}\,\Psi & \iff & \text{for some } k, 1 \leq k \leq j \\
 & & (w,k) \models \Psi \text{ and} \\
 & & \text{for every } i, k < i \leq j \\
 & & (w,i) \models \Phi
\end{array}$$

Let w be a word of length m and Φ a past formula. If $(w, m) \models \Phi$, we say that w *end-satisfies* Φ and write

$$w \dashv \Phi.$$

In other words, past formulae are satisfied "backwards" by finite words if they hold in the last position of these words. A formula Φ defines a set of sequences $L_\Phi \subseteq \Sigma^*$ where

$$L_\Phi = \{w \in \Sigma^* : w \dashv \Phi\} \tag{25}$$

The sets that are of the form L_Φ for some $\Phi \in TLP(\Sigma)$ are exactly the star-free regular subsets of Σ^+ (see [LPZ85, Zu86, CPP89]).

Note the interesting similarity and difference between this formalism and star-free regular expressions. The membership of $w \in \Sigma^*$ in both $L_\Phi \mathcal{S} L_\Psi$ and $L_\Psi \cdot L_\Phi$ is determined by the existence of a factorization $w = uv$ such that $u \in L_\Psi$, but in the former case *every prefix of* v must be in L_Φ, while in the latter it is sufficient that v *alone* is in L_Φ.

5.2 Future Temporal Logic

Future temporal logic is considered as a formalism for describing infinite sequences.

Definition 39 (Future Temporal Formulae) *A future temporal formula is defined inductively as:*

1. *For every past temporal formula* Π, $\Box\Diamond\Pi$ *(infinitely often* Π*) is a future temporal formula.*
2. *If* Φ *and* Ψ *are future formulae, so are* $\Phi \vee \Psi$ *and* $\neg\Phi$.

The rest of the Boolean and future temporal operators can be derived from $\vee$, $\neg$ and $\Box\Diamond$. The set of all such formulae is denoted by $TLF(\Sigma)$.

Definition 40 (Models for Future Formulae) *A model for a future temporal formula* Φ *is an infinite word*

$$\alpha = s_0, s_1 \ldots$$

Given $\alpha \in \Sigma^\omega$, we define the notion of a future temporal formula holding at position j, where j is finite. We let $\alpha_{[j..k]}$ denote $s_j, s_{j+1} \ldots s_k$.

Definition 41 (Satisfiability of future Formulae) *Let* Π *be a past formula and* Φ, Ψ *be future formulae. Satisfiability at position* j, $0 \leq j < \omega$, *in* α *is defined by:*

$$
\begin{array}{lcl}
(\alpha, j) \models \Box\Diamond\Pi & \iff & \textit{for infinitely many } k, j \leq k \\
 & & (\alpha_{[j..k]}, k) \dashv \Pi \\
(\alpha, j) \models \neg\Phi & \iff & (\alpha, j) \not\models \Phi \\
(\alpha, j) \models \Phi \vee \Psi & \iff & (\alpha, j) \models \Phi \textit{ or} \\
 & & (\alpha, j) \models \Psi
\end{array}
$$

If $(\alpha, 0) \models \Phi$ we abbreviate it as $\alpha \models \Phi$. In other words, future formulae are satisfied "forward" by infinite words if they hold in the first position of these words (which means that some Boolean combination of past formulae are satisfied backwards by infinitely many finite prefixes). A formula Φ defines a set of sequences $L_\Phi \subseteq \Sigma^\omega$ where

$$L_\Phi = \{\alpha \in \Sigma^\omega : \alpha \models \Phi\} \tag{26}$$

The sets that are of the form L_Φ for some $\Phi \in TLF(\Sigma)$ are exactly the star-free ω-regular subsets of Σ^ω, i.e., finite unions of sets of the form UV^ω where U and V are star-free subsets of Σ^*, (see [LPZ85, Zu86, CPP89]).

We will also need the infinitary equivalent of an accepting automaton (see [Th??] for a survey of the topic):

Definition 42 (Muller ω-Automaton) *Let a Muller ω-automaton is $\mathcal{A} = (\Sigma, Q, \delta, q_0, \mathcal{F})$ where $\mathcal{F} \subseteq 2^Q - \emptyset$. The set $L_\mathcal{A} \subseteq \Sigma^\omega$ accepted by $\mathcal{A}$ is:*

$$L_\mathcal{A} = \{\alpha \in \Sigma^\omega : Inf(\alpha) = F \text{ for some } F \in \mathcal{F}\} \tag{27}$$

where

$$Inf(\alpha) = \{q \in Q : \delta(q_0, \alpha[1..k]) = q \text{ for infinitely many } k\}$$

5.3 Translating Automata into Past Temporal Logic

Given a counter-free automaton $\mathcal{A} = (\Sigma, Q, \delta, q_0, F)$ accepting some set $L_\mathcal{A}$, we construct a past temporal logic formula Φ such that $L_\mathcal{A} = L_\Phi$. Initially we show how to construct the appropriate formula for a reset automaton, and then extend it to a cascade.

Let $\mathcal{A}$ be a reset automaton and let $\Sigma' \subseteq \Sigma$ be the subset of letters that induce resets in $\mathcal{A}$. For every state $q \in Q$, we define the following past formulae:

$$\text{in}_q = \bigvee_{\{\sigma \in \Sigma' : \sigma \text{ enters } q\}} \sigma \tag{28}$$

$$\text{out}_q = \bigvee_{\{\sigma \in \Sigma' : \sigma \text{ leaves } q\}} \sigma \tag{29}$$

Claim 30 *The set L_q of finite sequences that lead to $q \in Q - \{q_0\}$ is exactly the set of sequences satisfying the formula Φ_q defined as:*

$$\Phi_q = (\neg \mathrm{out}_q)\, \mathcal{S}\, (\mathrm{in}_q) \tag{30}$$

Proof: If $w = s_1, s_2, \ldots, s_m$ satisfies Φ_q then for some $j \leq m$, s_j satisfies in_q and for every $j < i \leq m$, s_i satisfies $\neg \mathrm{out}_q$ and thus $\delta(q_0, w) = q$. Conversly, if w does not satisfy Φ_q then either no prefix of w enters q or an exit from q occurs after the last entry. For the initial state q_0 we have to cover the case where the sequence contains no letter of in_q but also no letter of out_q. By definition, $\neg\Sigma$ is satisfied only by the empty word, so the formula Φ_{q_0} defined as:

$$\Phi_{q_0} = (\neg \mathrm{out}_{q_0})\, \mathcal{S}\, (\mathrm{in}_{q_0} \vee \neg\Sigma) \tag{31}$$

is satisfied exactly by the set L_{q_0} of (possibly empty) finite sequences that lead to q_0. ∎

Corollary 31 *Every regular subset of Σ^*, consisting of the sequences leading to some state in a reset automaton, can be expressed by a $TLP(\Sigma)$-formula of size $O(|\Sigma|)$.*

In order to extend this construction to a cascade we need the following:

Claim 32 *Let $\mathcal{B}_1 = (\Sigma, Q_1, \delta_1)$ be an automaton such that for every $q \in Q_1$ there exists a corresponding $TLP(\Sigma)$-formula Φ_q, and let $\mathcal{B}_2 = (Q_1 \times \Sigma, Q_2, \delta_2)$ be a reset automaton. Then for every $q' \in Q_2$ there exists a corresponding $TLP(\Sigma)$-formula $\Phi_{q'}$ satisfied exactly by the sequences leading to q' when $\mathcal{B}_2$ is connected to $\mathcal{B}_1$ in the cascade product $\mathcal{C} = \mathcal{B}_1 \circ \mathcal{B}_2$.*

Proof: Since $\mathcal{B}_2$ is a reset automaton, every $q' \in Q_2$ is reachable exactly by the sequences satisfying the $TLP(Q_1 \times \Sigma)$-formula

$$\Psi_{q'} = (\neg \mathrm{out}_{q'})\, \mathcal{S}\, (\mathrm{in}_{q'}) \tag{32}$$

where $\mathrm{in}_{q'}$ and $\mathrm{out}_{q'}$ are finite disjunctions of formulae of the form $\langle q, \sigma \rangle$. Let ξ be a sequence in $(Q_1 \times \Sigma)^*$ and let $w \in \Sigma^*$ be its projection. By the definition of the cascade product we have:

$$\begin{array}{rcl} (\xi, j) \models \langle q, \sigma \rangle & \iff & (w, j) \models \sigma \text{ and } (w, j-1) \models \Phi_q \\ & \iff & (w, j) \models \sigma \wedge \ominus \Phi_q \end{array}$$

Thus we can replace every occurence of $\langle q, \sigma \rangle$ in $\Psi_{q'}$ by $(\sigma \wedge \ominus \Phi_q)$ and obtain the $TLP(\Sigma)$-formula $\Phi_{q'}$. ∎

By a straightforward induction it can be shown that every state at level i can be expressed by a formula consisting of an i-nesting of the *since*-operator and that a configuration $p = \langle q_1, \ldots, q_k \rangle$ can be expressed as a conjunction:

$$\Phi_p = \Phi_{q_1} \wedge \ldots \wedge \Phi_{q_k} \tag{33}$$

Finally, any state q of the original automaton can be expressed as a disjunction of the formulae for its corresponding configurations in the cascade:

$$\hat{\Phi}_q = \bigvee_{p \in \varphi^{-1}(q)} \Phi_p \tag{34}$$

The set accepted by the automaton is expressed by the following disjunction:

$$\Phi_{\mathcal{A}} = \bigvee_{q \in F} \hat{\Phi}_q \tag{35}$$

From all this it follows that:

Corollary 33 *Every star-free regular set accepted by an automaton of size $O(n)$ can be expressed by a past temporal logic formula of size $O(2^{n \log n})$.*

A somewhat similar first-order characterization appears in [Me69]. Our result improves upon previously known non-elementary translations ([Zu86]).

5.4 Translating ω-Automata into Future Temporal Logic

Given a counter-free automaton $\mathcal{A} = (\Sigma, Q, \delta, q_0)$, the set of infinite sequences that visit q infinitely often can be expressed by the future formula $\Gamma_q = \Box\Diamond\Phi_q$, where Φ_q is the past formula corresponding to the finite sequences reaching q. Let F be a subset of Q. The formula describing all the infinite sequences α such that $Inf(\alpha) = F$ is:

$$\Gamma_F = \bigwedge_{q \in F} \Box\Diamond\Phi_q \wedge \bigwedge_{q' \notin F} \neg\Box\Diamond\Phi_{q'} \tag{36}$$

Finally if $\mathcal{A} = (\Sigma, Q, \delta, q_0, \mathcal{F})$ is a Muller ω-automaton, where $\mathcal{F} \subseteq 2^Q$, then $L_{\mathcal{A}} \subseteq \Sigma^\omega$ can be expressed by the formula

$$\Gamma_{\mathcal{A}} = \bigvee_{F \in \mathcal{F}} \Gamma_F \tag{37}$$

Corollary 34 *Every star-free ω-regular subset of Σ^ω accepted by an ω-automaton of size $O(n)$ can be expressed by a $TLF(\Sigma)$ formula of size $O(2^{n \log n})$.*

References

[Arb69] M.A. Arbib, *Theories of Abstract Automata*, Prentice-Hall, Englewood Cliffs, NJ, 1969.

[CH89] S. Cho and D.T. Huynh, *Finite Automata Aperiodicity is PSPACE-Complete*, Manuscript, University of Texas at Dallas, 1989.

[CPP89] J. Cohen, D. Perrin and J.E. Pin, *On the Expressive Power of Temporal Logic*, TR 89-84, LITP, Université Paris 7, 1989.

[Eil74] S. Eilenberg, *Automata, Languages and Machines, Vol. A*, Academic Press, New-York, 1974.

[Eil76] S. Eilenberg, *Automata, Languages and Machines, Vol. B*, Academic Press, New-York, 1976.

[Gec86] F. Gecseg, *Products of Automata*, Springer, Berlin, 1986.

[Gin68] A. Ginzburg, *Algebraic Theory of Automata*, Academic Press, New-York, 1968.

[HS66] J. Hartmanis and R.E. Stearns, *Algebraic Structure Theory of Sequential Machines*, Prentice-Hall, Englewood Cliffs, NJ, 1966.

[HU79] J.E. Hopcroft and J.D. Ullman, *Introduction to Automata Theory, Languages and Computation*, Addison-Wesley, Reading, MA, 1979.

[Koh78] Z. Kohavi, *Switching and Finite Automata Theory*, McGraw-Hill, New-York, 1978.

[KR65] K. Krohn and J.L. Rhodes, Algebraic Theory of Machines, I Principles of Finite Semigroups and Machines, *Transactions of the American Mathematical Society* 116, 450-464, 1965.

[Lal71] G. Lallement, On The Prime Decomposition Theorem of Finite Monoids, *Mathematical Systems Theory*, 5, 8-12, 1971.

[Lal79] G. Lallement, *Semigroups and Combinatorial Applications*, Wiley, New-York, 1979.

[LPZ85] O. Lichtenstein, A. Pnueli, and L. Zuck, The Glory of the Past, *Proc. Conf. Logics of Programs*, LNCS 193, 196-218, Springer, Berlin, 1985.

[Ma90] O. Maler, *Finite Automata: Infinite Behavior, Learnability and Decomposition*, Ph.D. thesis, Weizmann Institute, Rehovot, 1990.

[MaP90] O. Maler and A. Pnueli, Tight Bounds on the Complexity of Cascaded Decomposition of Automata, *Proc. 31st FOCS*, 672-682, 1990.

[MP90] Z. Manna and A. Pnueli, A Hierarchy of Temporal Properties, TR-CS-88-08, Weizmann Institute, 1988.

[MP91] BOOK

[MNP71] R. McNaughton and S. Papert, *Counter Free Automata*, MIT Press, Cambridge, MA, 1971.

[Me69] A.R. Meyer, A Note on Star-Free Events, *Journal of the ACM* 16, 220-225, 1969.

[MT69] A.R. Meyer and C. Thompson, Remarks on the Algebraic Decomposition of Automata, *Mathematical Systems Theory*, 3, 110-118, 1969.

[Mil89] R. Milner, *Communication and Concurrency*, Prentice-Hall, Englewood Cliffs, NJ, 1989.

[Pin86] J.-E. Pin, *Varieties of Formal Languages*, Plenum, New-York, 1986.

[Pnu77] A. Pnueli, The Temporal Logic of Programs, *Proc. 18th FOCS*, 46-57, 1977.

[Sch65] M.P. Schützenberger, On Finite Monoids Having only Trivial Subgroups, *Information and Control* 8, 190-194, 1965.

[Ste85] J. Stern, Complexity of some Problems from the Theory of Automata, *Information and Control* 66, 163-176, 1985.

[Tho90] W. Thomas, Automata on Infinite Objects, in J. Van Leeuwen (Ed.), *Handbook of Theoretical Computer Science*, Vol. B, 133-191, Elsevier, Amsterdam, 1990.

[TB73] B.A. Trakhtenbrot and Y.M. Barzdin, *Finite Automata: Behavior and Synthesis*, North-Holland, Amsterdam, 1973.

[VW86] M.Y. Vardi and P. Wolper, Automata-Theoretic Techniques in Modal Logics of Programs, *J. of Computer and Systems Science* 32, 183-221, 1986.

[Wel76] C. Wells, Some Applications of the Wreath Product Construction, *American Mathematical Monthly* 83, 317-338, 1976.

[Yoe63] M. Yoeli, *Decomposition of Finite Automata*, TR-10, US Office of Naval Research, Information Systems Branch, Hebrew University, Jerusalem, 1963.

[Zei67a] H.P. Zeiger, Yet Another Proof of the Cascaded Decomposition of Finite Automata, *Mathematical Systems Theory* 1, 225-228, 1967.

[Zei67b] H.P. Zeiger, Cascaded Synthesis of Finite-State Machines, *Information and Control* 10, 419-433, 1967.

[Zuc86] L. Zuck, *Past Temporal Logic*, Ph.D. thesis, Weizmann Institute, Rehovot, 1986.