Automata on Lempel-Ziv Compressed Strings

Hans Leiß[1] and Michel de Rougemont[2]

[1] Universität München, CIS, D-80538 München, Germany,
leiss@cis.uni-muenchen.de
[2] Université Paris-II & LRI Bâtiment 490, F-91405 Orsay Cedex, France,
mdr@lri.fr

Abstract. Using the Lempel-Ziv-78 compression algorithm to compress a string yields a dictionary of substrings, i.e. an edge-labelled tree with an order-compatible enumeration, here called an *LZ*-trie. Queries about strings translate to queries about *LZ*-tries and hence can in principle be answered without decompression. We compare notions of automata accepting *LZ*-tries and consider the relation between acceptable and MSO-definable classes of *LZ*-tries. It turns out that regular properties of strings can be checked efficiently on compressed strings by *LZ*-trie automata.

1 Introduction

We are interested in the *compressed model checking problem*: which properties of strings can be checked given the compressed strings? The challenge is to beat the decompress-and-then-check method. We restrict ourselves to the classical Ziv-Lempel[8] string compression algorithm *LZ*-78. It compresses a string $w \in \Sigma^+$ to a generally much shorter sequence $LZ(w) \in (\mathbb{N} \times \Sigma)^+$, where the numbers point to previous elements of the sequence.

As usual, we view a string w as a colored finite linear order $\mathcal{S}_w = (D, <, U_a)_{a \in \Sigma}$, where $D = \{0, \ldots, |w| - 1\}$ is the set of positions, ordered by $<$ as usual, and $U_a(i)$ means that a occurs at position i of w. Properties of strings are expressed in first-order (FO) or second-order (SO) logic of colored linear orders. In a similar way, we give two representations of *LZ*-78-compressed strings α by relational structures, one by node-labelled *LZ-graphs* $\mathcal{G}_\alpha$ and another one by *LZ-tries* $\mathcal{T}_\alpha$, which are a kind of edge-labelled trees.

A natural approach to the compressed model checking problem is to translate properties of strings to properties of compressed strings.

In fact, monadic second-order (MSO) formulas φ in the language of strings can be translated to dyadic second-order (DSO) formulas φ^{LZ} in the language of *LZ*-graphs $\mathcal{G}_\alpha$. One can therefore answer queries φ about $\mathcal{S}_w$ by evaluating φ^{LZ} on the smaller structure $\mathcal{G}_{LZ(w)}$. However, since the translation doubles the arity of relation variables, this is not guaranteed to provide an efficient solution.

By Büchi's well-known theorem (cf. [4], Theorem 5.2.3), MSO for strings is equally expressive as regular expressions, or finite automata, are. This raises the question whether MSO for *LZ*-graphs leads to a notion of *LZ*-automaton that provides an efficient method for checking a reasonably rich class of properties of compressed strings.

We study this question in the slightly more suitable format of *LZ*-tries. These come with an enumeration of their nodes and can be viewed as simple acyclic directed graphs. We introduce notions of *LZ-trie automata* by modifying corresponding notions of tree-automata. We show that deterministic bottom-up *LZ*-trie-automata are less powerful than non-deterministic ones, and that the latter capture $\exists$-MSO for *LZ*-tries, where $\exists$-MSO is the set of MSO-formulas of the form $\exists \boldsymbol{X} \psi$ where $\boldsymbol{X}$ are set variables and ψ is an FO-formula. Finally, we show that MSO-properties of strings can be checked efficiently on *LZ*-compressed strings by deterministic top-down *LZ*-automata. Thus, regular expression search in strings can be done on the *LZ*-compressed strings, without decompression.

2 Lempel-Ziv-78 Compression

We fix a finite alphabet Σ and, to avoid structures with empty universe, only consider non-empty strings $w \in \Sigma^+$. The classical Lempel-Ziv-78 compression algorithm has many variations (cf. [2], [3]). It decomposes a string $w \in \Sigma^+$ into a sequence of substrings or *blocks* $B_i \in \Sigma^+$, so that $w = B_0 \cdots B_{m-1}$. The first block B_0 consists of the first letter of w. Suppose for some $n > 0$, we have constructed blocks $B_0, \ldots, B_{n-1}$ such that $w = B_0 \cdots B_{n-1} v$ for some $v \in \Sigma^+$. Then B_n is the shortest non-empty prefix of v that is not among $\{B_0, \ldots, B_{n-1}\}$, if this exists[1], otherwise B_n is v. The *LZ-compression* $LZ(w)$ of w is the sequence $p_0 \cdots p_{m-1}$ of pairs

[1] and then B_n extends one of $B_0, \ldots, B_{n-1}$ by exactly one letter.

$p_n = (k, a)$ such that $B_n = B_{k-1}a$ (where $B_{-1} := \epsilon$) and $w = B_0 \cdots B_{m-1}$. The decompression is given by $B_k = decode(p_k)$ where $decode((0, a)) = a$ and $decode((n + 1, a)) = decode(p_n)a$.

Example 1. The blocks of $w = abbbaabbabbb$ are $a.b.bb.aa.bba.bbb$, and its compression is $LZ(w) = (0, a)(0, b)(2, b)(1, a)(3, a)(3, b)$.

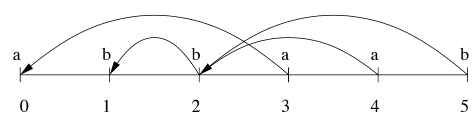
2.1 LZ-Graphs

Definition 1. *A compressed string $\alpha = p_0 \cdots p_{m-1}$ is represented as a* finite labelled ordered graph

$$\mathcal{G}_\alpha := (D_m, <, U_a, E)_{a\in\Sigma},$$

where $m = |\alpha|$ is the number of blocks, $D_m = \{0, \ldots, m - 1\}$, $<$ the natural order on D_m, $U_a(i)$ is true iff the last letter of block B_i is a. The binary relation E describes the reference to previous pairs: if $p_i = (k, a)$ for some $a \in \Sigma$ and $k > 0$, (i.e. the k-th block B_{k-1} is the longest strict prefix of B_i), then there is an edge $E(i, k-1)$ from node i to $k - 1$.

Example 1 (Cont.). For $w = a.b.bb.aa.bba.bbb. = B_0B_1B_2B_3B_4B_5$, the graph $\mathcal{G}_{LZ(w)}$ is



Observe that $\mathcal{S}_w$ can be interpreted in $\mathcal{G}_{LZ(w)}$ as a binary relation: a position i in w is mapped to the pair $h(i) = (k, j)$ in $LZ(w)$ iff i lies in block B_k and B_j is the nonempty prefix of B_k ending in i. Note that in $\mathcal{G}_{LZ(w)}$, node j can be reached from k by a path of E-edges.

Example 2. If $w = a.aa.ab.aba.aa$, the positions 5,6,7 occurring in block $B_3 = aba$ are represented by $(3, 0), (3, 2), (3, 3)$, because the nonempty prefixes of B_3 are $a = B_0$, $ab = B_2$, and $aba = B_3$.

Theorem 1 ([1]). *For every MSO-formula $\varphi(x_1, \ldots, x_n, \boldsymbol{X}^{(1)})$ about strings there is a DSO formula $\varphi^{LZ}(x_1, y_1, \ldots, x_n, y_n, \boldsymbol{X}^{(2)})$ about LZ-graphs, such that for each $\mathcal{S}_w$ and all $i_1, \ldots, i_n \in \mathcal{S}_w$ and $\boldsymbol{S} \subseteq \mathcal{S}_w$,*

$$\mathcal{S}_w \models \varphi[\boldsymbol{i}, \boldsymbol{S}] \iff \mathcal{G}_{LZ(w)} \models \varphi^{LZ}[h(\boldsymbol{i}), h(\boldsymbol{S})].$$

Proof. (Sketch) In DSO, we can define E^*, the reflexive transitive closure of E, and then define φ^{LZ} inductively using

$$\begin{aligned}
(U_a(x_i))^{LZ} &:= U_a(y_i),\\
(x_i \leq x_j)^{LZ} &:= x_i < x_j \vee (x_i = x_j \wedge y_i \leq y_j),\\
(\exists x_{n+1}\varphi)^{LZ} &:= \exists x_{n+1}\exists y_{n+1}\,(E^*(x_{n+1}, y_{n+1}) \wedge \varphi^{LZ}),\\
(\exists X^1\varphi)^{LZ} &:= \exists X^2(\forall x\forall y(X^2(x,y) \to E^*(x,y)) \wedge \varphi^{LZ}).
\end{aligned} \tag{1}$$

For the atomic cases, note that in φ^{LZ} a variable x_i stands for a block of w and y_i for a relative position in this block.

A property L of non-empty strings is *definable on strings* (resp. *on compressed strings* or *LZ-graphs*), if for some formula φ of the appropriate language, $L = \{w \mid \mathcal{S}_w \models \varphi\}$ (resp. $L = \{w \mid \mathcal{G}_{LZ(w)} \models \varphi\}$).

Remark 1. There are properties of strings that are FO-definable on strings, but not on LZ-graphs, like $\exists x(U_a(x) \wedge U_b(x+1))$. There are properties of strings that are FO-definable on LZ-graphs, but not even MSO-definable on strings (cf. [1]).

2.2 LZ-Tries

While compressing $w = B_0 \cdots B_{n-1}v$, the blocks $B_0, \ldots, B_{n-1}$ found are maintained as a *dictionary* of subwords of w and stored as a tree by sharing common prefixes. The linear order of the blocks in $LZ(w)$ amounts to an enumeration of the nodes of the tree.

Definition 2. *A (finite)* Σ-tree $(T, \leq, \xleftarrow{a}, 0)_{a\in\Sigma}$ *is a (finite) tree* $(T, \leq, 0)$ *with root 0, where* $\{\xleftarrow{a}\ \subseteq T \times T \mid a \in \Sigma\}$ *are pairwise disjoint minimal relations such that* $\leq$ *is the reflexive transitive closure of their union.*

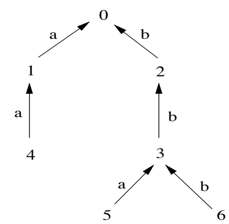
A Σ*-tree is a* Σ-trie *if to each node* $n \in T$ *and each* $a \in \Sigma$ *there is at most one* $n' \in T$ *such that* $n \xleftarrow{a} n'$*. A (finite)* enumerated Σ-trie

$$\mathcal{T} = (T, \leq, \xleftarrow{a}, 0, Succ)_{a\in\Sigma},$$

or an LZ-trie *for short, is a* Σ*-trie* $(T, \leq, \xleftarrow{a}, 0)_{a\in\Sigma}$ *with a successor relation*[2] *Succ on* T *that is compatible with the partial order* $\leq$*. We assume that* $T = \{0, 1, 2, \ldots, m\}$ *and* $Succ(i,j)$ *iff* $i + 1 = j$ *in* $\mathbb{N}$*.*

[2] i.e. a minimal binary relation $Succ$ whose transitive reflexive closure $Succ^*$ is a total ordering of T

Example 1 (Cont.). Enumerating the pairs of $LZ(w)$ by 1,2, etc. in a third component, we obtain a sequence $(0, a, 1)(0, b, 2)(2, b, 3)$ $(1, a, 4)(3, a, 5)(3, b, 6)$ of triples. These represent a tree in which block B_k labels the path from the root 0 to node $k + 1$:



A tuple $p_n = (k, a)$ of $LZ(w)$ is drawn as an edge $k \xleftarrow{a} n + 1$.

We write $\mathcal{T}_\alpha$ for the enumerated trie representing the compressed word α. We always assume that our strings w have a distinguished end symbol; then the final block of $LZ(w)$ is different from the previous ones and the tree of blocks indeed is a trie.

Remark 2. What differs in choosing $\mathcal{G}_\alpha$ or $\mathcal{T}_\alpha$ is the logical language used to talk about LZ-compessed strings α. Basically, we have

$$\mathcal{G}_\alpha \models E(i, j) \wedge U_a(i) \iff \mathcal{T}_\alpha \models (j + 1) \xleftarrow{a} (i + 1).$$

Modulo the additional root node in the trie, $\geq$ in the trie amounts to E^* in the graph, and $\leq$ in the graph to $Succ^*$ in the trie. Since E^* resp. $Succ^*$ is $\exists$- and $\forall$-MSO-definable in the language of LZ-graphs resp. LZ-tries, $\exists$-MSO-properties of LZ-graphs translate to $\exists$-MSO-properties of LZ-tries and vice versa.

Using (1), quantifiers Qx and QX about strings translate to *bounded* quantifiers $Q(x, y) \in E^*$ and $QX \subseteq E^*$. So when translating to the language of LZ-tries we only quantify over tuples (and relations of such) whose components lie on a *path* of $\mathcal{T}_{LZ(w)}$.

3 MSO-Equivalence for LZ-Graphs

For relational structures $\mathcal{A}, \mathcal{B}$ of the same signature, $\mathcal{A} \equiv_r^{MSO} \mathcal{B}$ says that $\mathcal{A}$ and $\mathcal{B}$ satisfy the same MSO-sentences of quantifier rank $\leq r$.

Two facts about MSO for strings imply the existence of finite automata that can check MSO-properties of strings (cf. [4]):

a) for each r, there are only finitely many $\equiv_r^{MSO}$-equivalence classes for word structures $\mathcal{S}_w$, and
b) for compound strings wa, the $\equiv_r^{MSO}$-class of $\mathcal{S}_{wa}$ depends only on the $\equiv_r^{MSO}$-classes of $\mathcal{S}_w$ and $\mathcal{S}_a$.

(The analogous situatation holds for trees over Σ.) *LZ*-compressed words $\alpha = p_0 \cdots p_{m-1}$ are words over the infinite alphabet $\mathbb{N} \times \Sigma$. Can we check MSO-properties of compressed strings by a kind of finite automaton for *LZ*-graphs? Since we deal with a finite relational language, we still have a):

Proposition 1. *For each r, the equivalence relation $\equiv_r^{MSO}$ between LZ-graphs has finite index.*

But what about b)? By extending winning strategies for duplicator in the Ehrenfeucht-Fraisse-game $G_r(\mathcal{G}_\alpha, \mathcal{G}_\beta)$, we can show:

Lemma 1. *(i) For LZ-compressed words $\alpha(0,a), \beta(0,a')$ over Σ,*

$$\mathcal{G}_\alpha \equiv_r^{MSO} \mathcal{G}_\beta \ \wedge a = a' \implies \mathcal{G}_{\alpha(0,a)} \equiv_r^{MSO} \mathcal{G}_{\beta(0,a')}.$$

(ii) For LZ-compressed words $\alpha(k+1,a)$ and $\beta(k'+1,a')$ over Σ,

$$(\mathcal{G}_\alpha, k) \equiv_r^{MSO} (\mathcal{G}_\beta, k') \ \wedge \ a = a' \implies \mathcal{G}_{\alpha(k+1,a)} \equiv_r^{MSO} \mathcal{G}_{\beta(k'+1,a')}.$$

However, in (ii) one has to assume that the elements k, k' pointed to from the new maximal elements share the same properties. Instead, one would need the stronger claim

$$\mathcal{G}_\alpha \equiv \mathcal{G}_\beta \wedge \mathcal{G}_\alpha \restriction k \equiv \mathcal{G}_\beta \restriction k' \wedge a = a' \implies \mathcal{G}_{\alpha(k+1,a)} \equiv \mathcal{G}_{\beta(k'+1,a')},$$

where $\mathcal{G}_\alpha \restriction k$ is the restriction of $\mathcal{G}_\alpha$ with k as its maximal element. The equivalence class of $\mathcal{G}_\alpha \restriction k$ would be the automaton state assigned to k. (Notice that $\mathcal{G}_\alpha \restriction k$ is a *LZ*-graph, but $(k+1, a)$ is not a *LZ*-compressed word.) The problem is that duplicator's winning strategies for $G_r(\mathcal{G}_\alpha, \mathcal{G}_\beta)$ and $G_r(\mathcal{G}_\alpha \restriction k, \mathcal{G}_\beta \restriction k')$ may pick *different* elements to answer spoilers playing of some element of, say, $\mathcal{G}_\alpha \restriction k$.

Thus, unlike in the case of strings or trees, for compound compressed words $\alpha(k, a)$ we have component *LZ*-graphs $\mathcal{G}_\alpha$ and $\mathcal{G}_\alpha \restriction k$ that are not disjoint, and winning strategies in games for these do not combine to winning strategies for composed *LZ*-graphs.

From this we conclude that we cannot use a Büchi-Myhill-Nerode construction to obtain from the $\equiv_r^{MSO}$-classes a finite sequential automaton for *LZ*-graphs, and likewise for *LZ*-tries.

4 LZ-Trie-Automata

If we view LZ-tries as trees with an additional edge *Succ* between nodes, we obtain directed acyclic graphs of a special kind: the successor child may be equal to some decendant with respect to the $\xleftarrow{a}$-child-relations. Since these are still very close to trees, it is natural to use a variation of tree-automata as an approximative notion of LZ-automaton for checking properties of LZ-tries.

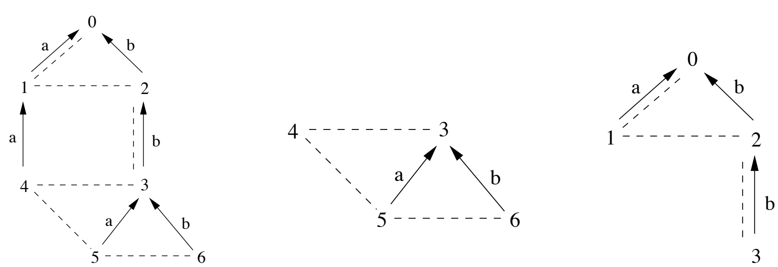
Definition 3. *Let $n \in G$ be a node in a graph (G, E). For $m \in \mathbb{N}$, the* sphere of radius m around n, *$s_m(n)$, is the set of nodes $k \in G$ such that there is a E-path of length $\leq m$ from n to k or vice versa. The* hemisphere of radius m around n, *$hs_m(n)$, is the set of nodes k such that there is an E-path of length $\leq m$ from n to k.*

Definition 4. *Let $n \in T$ be a node in the LZ-trie $\mathcal{T}$. The* bottom-up LZ-hemisphere of radius m around n, *$bu\text{-}hs^{\mathcal{T}}_m(n)$, is the restriction of $\mathcal{T}$ to the m-hemisphere around n in the graph (T, E), where*

$$E := \bigcup \{ \xleftarrow{a} \mid a \in \Sigma\} \cup \{Succ\}.$$

The top-down LZ-hemisphere of radius m around n, *$td\text{-}hs^{\mathcal{T}}_m(n)$, is the restriction of $\mathcal{T}$ to the m-hemisphere around n in the graph $(T, \breve{E})$, where $\breve{E}$ is the converse of E. An* LZ-hemisphere *is an LZ-hemisphere of some radius around some node in some LZ-trie $\mathcal{T}$.*

Example 1 (Cont.). An LZ-trie and the bottom-up resp. top-down 2-hemispheres of node 3 (with dashed edges for *Succ* resp. *Pred*):



Definition 5. *A finite* bottom-up (resp. top-down) m-LZ-automaton $\mathcal{A} = (Q, \Sigma, \delta, q_{in}, F)$ *consists of a finite set Q of* states, *sets $I, F \subseteq Q$ of* initial *and* final *states, a finite* alphabet Σ, *a finite*

transition relation δ *consisting of pairs* (P, q)*, written* $P \to q$*, where* $q \in Q$ *and* P *is a bottom-up (resp. top-down) LZ-hemisphere of radius* m *whose nodes except the root are labelled by elements of* Q*.*

A **run** *of* $\mathcal{A}$ *on an LZ-trie* $\mathcal{T}$ *is a function* $r : T \to Q$ *where* $r(\max) \in I$ *(resp.* $r(0) \in I$*) and for each* $n \in T$ *there is some* $(P, q) \in \delta$ *such that bu-*$hs^{\mathcal{T}}_m(n)$ *(resp. td-*$hs^{\mathcal{T}}_m(n)$*), expanded by the labelling of nodes given by* r*, is isomorphic to* P *with label* q *at its root.* $\mathcal{A}$ **accepts** $\mathcal{T}$ *if there is a run* r *of* $\mathcal{A}$ *on* $\mathcal{T}$ *such that* $r(0) \in F$ *(resp.* $r(\max) \in F$*). Let* $L(\mathcal{A}) := \{\mathcal{T} \mid \mathcal{A} \text{ accepts } \mathcal{T}\}$ *be* the class of *LZ*-tries accepted by $\mathcal{A}$.

$\mathcal{A}$ *is* deterministic *if* $|I| = 1$ *and* $q = q'$ *when* $(P, q), (P, q') \in \delta$*.*

While an m-*LZ*-automaton $\mathcal{A}$ sequentially follows the enumeration of a trie $\mathcal{T}_\alpha$, it can access some states reached at suffixes (resp. prefixes) of α. Strictly speaking, it does not have a 'finite memory'.

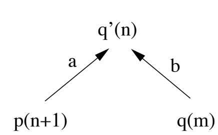
4.1 Bottom-up *LZ*-Trie-Automata

A 1-*LZ*-automaton working bottom-up the *LZ*-trie towards the root has transitions that determine the state at a node from the states at the node's Σ-children and successor. But it also has to distinguish which of the Σ-children is the successor of the node, if any.

Example 2. Consider the class $\mathcal{K}$ of *LZ*-tries over $\Sigma = \{a, b\}$ which have a node whose successor and a-child agree, i.e. which satisfy the sentence $\varphi := \exists x \exists y\ [y = x + 1 \wedge (x \xleftarrow{a} y)]$. We give a bottom-up 1-*LZ*-automaton $\mathcal{A}$ accepting $\mathcal{K}$. We write a transition in the form

$$(q_a, q_b, q_{succ}, i) \to p,$$

where q_a, q_b, q_{succ} are the states of the a-, b- and *Succ*-child or $\perp$, when there is no such child, and $i \in \{1, 2, 3, \perp\}$ says which of the children is equal to the successor node, if any. Thus, $(p, q, p, 1) \to q'$ corresponds to the transition $P \to q'$ which can be shown as



$\mathcal{A}$ has a final state q_1, which is assigned to all ancestors of the root of a subtrie satisfying φ, and an initial state q_0, which is assigned to all other nodes of the input trie. Letting q, p, q' range over $\{q_0, q_1\}$, the transition table is

$$\begin{array}{llll} a) & (\bot,\bot,\bot,\bot) \to q_0 & e) & (q,p,q',3) \to q' \\ b) & (q,\bot,q,1) \to q_1 & f) & (q,\bot,q',3) \to q' \\ c) & (q,p,q,1) \to q_1 & g) & (\bot,q,q',3) \to q' \\ d) & (q,p,q',2) \to q' & h) & (\bot,\bot,q',3) \to q'. \end{array}$$

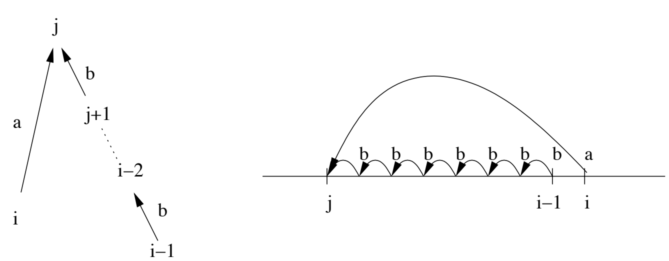
Rule $a)$ means that if there is no successor-node, $\mathcal{A}$ is in state q_0. Rules $b)$ and $c)$ say that if the successor-node is the a-child, then $\mathcal{A}$ goes to q_1 as we just saw the pattern φ. Rules $d) - f)$ say that if the successor node differs from the a-child, $\mathcal{A}$ remains in the state of the successor node. Similar for $g)$ and $h)$, when there is no a-child.

Theorem 2. *There is a property of LZ-tries that is recognized by a non-deterministic bottom-up* 1*-LZ-automaton but not by any deterministic bottom-up* m*-LZ-automaton.*

Proof. (Sketch) Let $\Sigma = \{a, b, c\}$ and consider the following property φ of enumerated Σ-tries:

> There are two subsequent nodes $i-1$ and i such that some node j is both the a-predecessor of i and a b-ancestor of $i-1$.

An *LZ*-trie satisfies φ iff it contains nodes linked as follows (in the trie and in the *LZ*-graph, respectively), where $j < i-1$:



(2)

When finding nodes connected like $i-1$ and i, a bottom-up 1-*LZ*-automaton can (non-deterministically) guess that a b-ancestor of $i-1$ will be the a-parent of i and check this while proceeding. But no deterministic bottom-up m-LZ-automaton can check the property φ because, intuitively speaking, the m-hemisphere of a node j is not big enough to see if the predecessor of its a-child is one of its b-descendants. The proof details are not quite obvious but have to be omitted for lack of space.

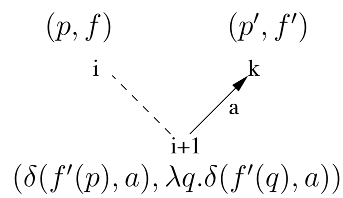
4.2 Top-down LZ-Trie-Automata

G.Navarro [5] has shown how to do regular expression search on LZ-78-compressed texts by simulating an automaton reading the original text, beating the decompression-and-search approach by a factor of 2. Actually, this simulation is a deterministic top-down 1-LZ-automaton on the LZ-compressed text:

Theorem 3. *The LZ-compression $\{\mathcal{T}_{LZ(w)} \mid w \in R\}$ of any regular set $R \subseteq \Sigma^+$ is accepted by a deterministic top-down 1-LZ-automaton.*

Proof. Let $\mathcal{A} = (Q, \Sigma, q_0, \delta, F)$ be a deterministic finite automaton accepting R. Define a deterministic top-down 1-LZ-automaton $\mathcal{A}' = (Q', \Sigma, \delta', q'_{in}, F')$ by $Q' := Q \times (Q \to Q)$, $q'_0 := (q_0, \lambda q.q)$, $F' := \{(q, f) \mid q \in F\}$ and δ' according to the following transitions:

$0 \cdot \ (q_0, \lambda q.q)$



for each $a \in \Sigma$ (including the case $i = k$).

Suppose r' is a run of $\mathcal{A}'$ on $\mathcal{T}_{LZ(w)}$ for a compressed word $LZ(w) = p_0 \cdots p_{m-1}$ where p_k represents block B_k of $w = B_0 \cdots B_{m-1}$. By induction we see that for each $k < m$,

$$r'(k) = (\delta(q_0, B_0 \cdots B_{k-1}), \lambda p.\delta(p, B_{k-1})). \tag{3}$$

For $k > 0$, suppose $r'(i) = (p, f) = (\delta(q_0, B_0 \cdots B_{i-1}), f)$ and $p_i = (k, a)$. Then $B_i = B_{k-1}a$, and with $r'(k) = (p', f') = (p', \lambda p.\delta(p, B_{k-1}))$

$$\begin{aligned} r'(i+1) &= (\delta(f'(p), a), \lambda q.\delta(f'(q), a)) \\ &= (\delta(p, B_{k-1}a), \lambda q.\delta(q, B_{k-1}a)) \\ &= (\delta(\delta(q_0, B_0 \cdots B_{i-1}), B_i), \lambda q.\delta(q, B_i)) \\ &= (\delta(q_0, B_0 \cdots B_i), \lambda q.\delta(q, B_i)). \end{aligned}$$

Hence $\mathcal{A}'$ accepts $\{\mathcal{T}_{LZ(w)} \mid w \in R\}$, because

$$w \in R \iff \delta(q_0, B_0 \cdots B_{m-1}) \in F \iff r'(m) \in F'.$$

4.3 Graph Acceptors and ∃-MSO for *LZ*-Tries

If we consider the states q of an automaton $\mathcal{A}$ as monadic predicates on nodes of tries, the condition "there is an accepting $\mathcal{A}$-run" can be expressed by an ∃-MSO-sentence $\exists \boldsymbol{q} \psi$ in the language of *LZ*-tries:

Proposition 2. *For every m-LZ-automaton $\mathcal{A}$ there is an ∃-MSO-sentence $\varphi_{\mathcal{A}}$ defining the class of LZ-tries accepted by $\mathcal{A}$.*

For sentences, this improves on the translation of Theorem 1:

Corollary 1. *Every property of strings which is definable in MSO on strings is definable in ∃-MSO on LZ-tries.*

Proof. By Büchi's theorem, Theorem 3 and Proposition 2.

The converse of Corollary 1 is wrong: $\{a.b. \cdots .b^n a.b^n b. \mid n \in \mathbb{N}\}$ is not a regular language, but has an FO-definable class of *LZ*-tries.

To show the converse of Proposition 2, we use the graph acceptors for directed acyclic graphs presented by W. Thomas [7].

On graphs $G = (V, E)$, a *tile* δ over a set Q is an m-neighbourhood of a node with a labelling in Q, i.e. a finite node-labelled graph. A *graph acceptor* is a triple $\mathcal{A} = (Q, \Delta, Occ)$ where Δ is a finite set of tiles over the finite set Q and Occ is a boolean combination of conditions $\chi_{p,\delta}$:= *there are* $\geq p$ *occurences of tile* δ. A *run* of $\mathcal{A}$ on G is a function $r : V \to Q$ such that each m-neighbourhood of G becomes a tile $\delta \in \Delta$. Then $\mathcal{A}$ *accepts* a graph G if there exists a run of $\mathcal{A}$ on G whose tiles satisfy Occ.

Note that the existence of an accepting run is non-constructive. The main result of [7] says that a class of graphs of bounded degree is definable in ∃-MSO iff it is accepted by a graph acceptor.

Theorem 4. *A class $\mathcal{K}$ of LZ-tries is ∃-MSO-definable iff for some m, $\mathcal{K}$ is accepted by an m-LZ-automaton.*

Proof. Note that each *LZ*-trie $\mathcal{T}$ satisfies the following conditions:

(i) each node of $\mathcal{T}$ has a degree $\leq |\Sigma| + 1$,
(ii) $\mathcal{T}$ is an acyclic (directed) graph with a designated out-edge (the successor) for each node,
(iii) $\mathcal{T}$ has a node that is reachable from any node by a path.

Using (i), by Theorem 3 of [7], $\mathcal{K}$ is $\exists$-MSO-definable iff $\mathcal{K}$ is recognizable by a graph acceptor. Moreover, from (i)-(iii) and Proposition 6 of [7], it follows that $\mathcal{K}$ is recognizable by a graph acceptor iff it is recognizable by a graph acceptor without occurrence constraints.

$\Leftarrow$: Suppose $\mathcal{K}$ is accepted by some m-LZ-automaton $\mathcal{A}$. We may assume that final states of $\mathcal{A}$ do not occur in the m-hemispheres P of transitions (P, q) of $\mathcal{A}$. Consider the transitions of $\mathcal{A}$ as a tiling system. Then an accepting run of $\mathcal{A}$ is a tiling that uses at least one of the tiles (P, q) where $q \in F$. Thus, $\mathcal{K}$ has a graph acceptor.

$\Rightarrow$: By the above remarks, $\mathcal{K}$ is recognizable by a graph acceptor without occurrence constraints. On the *LZ*-tries, the tiles have a root node, so we can view each tile as a transition rule saying that the automaton enters a state at the root depending on the m-sphere of the root and the states at the descendants in the sphere. Let each state be final; then the automaton accepts iff there is a tiling.

A graph $G = (V, E)$ is *k-colorable* if there is a partitioning of the set V of nodes into at most k classes (colors) such that any two adjacent nodes belong to different classes.

Clearly, k-colorability can be expressed by an $\exists$-MSO sentence, and hence has a graph acceptor. While 3-colorability on arbitrary graphs is an NP-complete problem, it is trivial on *LZ*-tries, because the maximal node m is connected to at most two nodes:

Proposition 3. *Every LZ-trie is 3-colorable.*

Potthoff e.a. [6] have shown that on the class of directed acyclic graphs whose edges are uniquely labelled with a bounded number of labels, 2-colorability is *not* recognizable by a deterministic graph acceptor, i.e. one with at most one accepting run on each graph. For the subclass of LZ-tries, however, it is (actually by a 4-state deterministic bottom-up-LZ-automaton):

Proposition 4. *On the class of LZ-tries, 2-colorability is recognizable by a deterministic graph acceptor.*

4.4 Strictness of the Automata Hierarchy

The bottom-up resp. top-down hierarchies of m-LZ-automata are strict, and no level exhausts the MSO-properties of tries:

Theorem 5. *For every m, there is an MSO-sentence defining a class of LZ-tries that is not accepted by any m-LZ-automaton.*

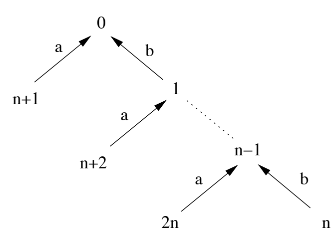
Proof. For $m = 1$, let $L = \{b^1 b^2 \cdots b^n a b^1 a b^2 a \cdots b^{n-1} a \mid n \in \mathbb{N}\} \subseteq \{a, b\}^+$. Each $w \in L$ has a LZ-block decomposition as indicated by

$$w_n = b^1.b^2.\cdots.b^n.a.b^1a.b^2a.\cdots.b^{n-1}a$$

and a compression

$$LZ(w_n) = (0, b)(1, b)\cdots(n-1, b)(0, a)(1, a)\cdots(n-1, a).$$

The corresponding tries $\mathcal{T}_{LZ(w_n)}$ look like



(4)

where the successor relation is given by the node numbers.

The class of enumerated tries in $LZ(L)$ can be defined by the MSO-sentence φ saying that the set B of nodes that are the root 0 or a b-child, has the following properties (of which (i) is neither $\exists$-MSO nor $\forall$-MSO):

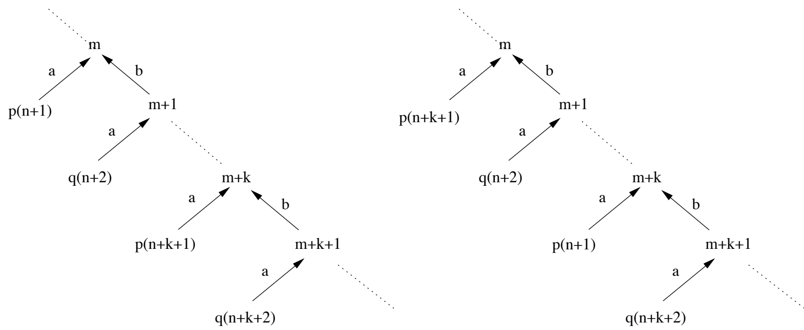
(i) $B \supseteq \{0\}$ is the smallest set being closed under b-children,
(ii) a node is not in B iff it is an a-child of a node in B,
(iii) for all nodes x, y, we have $y = x + 1$ iff one of the following holds:
  (a) y is the b-child of x,
  (b) y is the a-child of the b-child y' of some x' whose a-child is x,
  (c) y is the a-child of 0 and x the member of B that has no b-child.

Claim. $LZ(L)$ is not accepted by a 1-bottom-up-LZ-automaton.

Suppose that $\mathcal{A}$ is a 1-bottom-up-LZ-automaton that accepts the class defined by φ. Let $m > |Q|$ and $w = w_{2m}$. An accepting run of $\mathcal{A}$ on $\mathcal{T}_{LZ(w)}$ assigns the same state, say q, to at least two different a-childs, say nodes $n+k+2$ and $n+2$. The state of their predecessors is determined by a rule

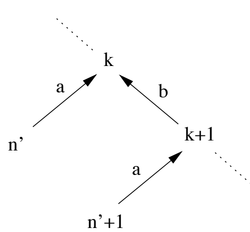
$$(\perp, \perp, q, 3) \rightarrow p.$$

Let $\mathcal{T}'$ be like $\mathcal{T}_{LZ(w)}$, except that nodes $n+1$ and $n+k+1$ are switched in the ordering. Then $\mathcal{A}$ will also accept the modified structure, as indicated by the states assigned to nodes as follows:

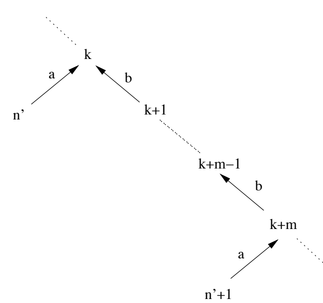


But the enumeration of a-children in $\mathcal{T}'$ does not conform to φ, so $\mathcal{T}' \in T(\mathcal{A}) \setminus LZ(L)$.

For the case $m > 1$, we modify the example as follows: along the B-part, between two nodes that have both an a-child and a b-child, we add $m-1$ nodes that have no a-child, i.e. the subgraphs



are replaced by



The LZ-tries arising this way are the compressions of the words

$$w_{n,k} = b^1.b^2.b^3.\cdots.b^{nm+k}.a.b^m a.b^{2m} a.\cdots.b^{nm} a.$$

Then $\{\mathcal{T}_{LZ(w_{n,k})} \mid n, k \in \mathbb{N}, k \leq m\}$ is accepted by a $(m+1)$-bottom-up-LZ-automaton, hence MSO-definable. But it is not accepted by any m-bottom-up-LZ-automaton: intuitively, the m-hemisphere of a node k does not tell whether the path from k following a and successor and the path following $b^m a$ end at the same node.

A similar argument shows that the class defined by φ is also not accepted by any m-top-down-LZ-automaton with the same m.

5 Conclusion

We have shown some relations between properties of strings, properties of their Lempel-Ziv-78-compressions, and automata accepting compressed strings, but the picture is not complete yet. For example:

Conjecture 1. On the class of all LZ-tries, not every MSO-formula is equivalent to an $\exists$-MSO-formula.

It seems likely that the class of acceptable LZ-tries is not closed under complement (cf. Theorem 2). We also believe that deterministic bottom-up-LZ-automata are much weaker than deterministic top-down-LZ-automata (cf. Theorem 3):

Conjecture 2. There is no deterministic bottom-up m-LZ-automaton that can check on $LZ(w)$ whether $w \in \{a, b\}^+$ has a subword ab.

References

1. Foto Afrati, Hans Leiß, and Michel de Rougemont. Definability and compression. In *15th Annual IEEE Symposium on Logic In Computer Science, LICS'2000. Santa Barbara, CA, June 22-26, 2000*, pages 151–172. Computer Society Press, 2000.
2. Timothy C. Bell, John G. Cleary, and Ian H. Witten. *Text Compression.* Prentice Hall, Englewood Cliffs, NJ, 1990.
3. T. Cover and J. Thomas. *Elements of Information Theory.* John Wiley, 1991.
4. H.-D. Ebbinghaus and J. Flum. *Finite Model Theory.* Springer-Verlag, 1991.
5. Gonzalo Navarro. Regular expression searching on compressed text. *Journal of Discrete Algorithms*, 2003 (to appear).
6. Andreas Potthoff, Sebastian Seibert, and Wolfgang Thomas. Nondeterminism versus determinism of finite automata over directed acyclic graphs. *Bull. Belg. Math. Soc.*, 1:285–298, 1994.
7. Wolfgang Thomas. Automata theory on trees and partial orders. In eds. M. Bidoit, M. Dauchet, editor, *TAPSOFT'97*, LNCS 1214, pages 20–38. Springer Verlag, 1997.
8. J. Ziv and A. Lempel. Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, pages 530–536, 1978.