Definability and Compression

Foto Afrati
afrati@softlab.ece.ntua.gr
National Technical University
Athens
Greece

Hans Leiß
leiss@cis.uni-muenchen.de
Universität München, CIS
D-80538 München
Germany

Michel de Rougemont
mdr@lri.fr
Université Paris-II
& LRI Bâtiment 490
F-91405 Orsay Cedex
France

Abstract

A compression algorithm takes a finite structure of a class **K** as input and produces a finite structure of a different class **K'** as output. Given a property P on the class **K** defined in a logic $\mathcal{L}$, we study the definability of property P on the class **K'**. We consider two compression schemes on unary ordered structures (strings), compression by run-length encoding and the classical Lempel-Ziv-78 scheme.

First-order properties of strings are first-order on run-length compressed strings, but this fails for images, i.e. 2-dimensional strings. We present simple first-order properties of strings which are not first-order definable on strings compressed with the Lempel-Ziv compression scheme.

We show that all properties of strings that are first-order definable on strings are definable on Lempel-Ziv compressed strings in $FO(TC)$, the extension of first-order logic with the transitive closure operator. We define a subclass $\mathcal{F}$ of the first-order properties of strings such that if P is a property in $\mathcal{F}$, it is also first-order definable on the Lempel-Ziv compressed strings.

Monadic second-order properties of strings, i.e. regular languages, are dyadic second-order definable on Lempel-Ziv compressed strings.

1 Introduction

When data are implicitly given, be it in compressed form or by a symbolic representation, it is important to know which problems can be efficiently solved without changing the representation of the data. Examples of the first type arise when large amount of data are communicated and processed in compressed form: we need tools to efficiently determine properties of data without having to decompress them first. Efficient algorithms to search for substrings in compressed strings have been proposed recently for various compression schemes [9, 12, 14] and we try to understand which other problems can be solved. An example of the second type is symbolic model checking, where a program can be viewed as a symbolic representation of a large state transition system, its Kripke structure. In order to verify properties of the program, we can only access the Kripke structure in a compressed form, for example as an OBDD

(Ordered Binary Decision Diagram, [2]). The evaluation of formulas may be efficient with this representation in some specific cases which need to be characterized. The complexity of graph properties when the graphs are represented by succinct circuits is studied in [11] and in [10] when the graphs are represented by OBDDs. In both cases, simple graph properties become intractable.

We generalize these approaches and study the definability of problems for various compression schemes from a descriptive complexity point of view. We view a compression scheme $\mathcal{C}$ as an encoding of a (finite) structure $\mathcal{U} \in \mathbf{K}$ by a more succinct representation, $\mathcal{C}(\mathcal{U})$, a structure of generally different signature. A natural question then arises: given a compressed structure $\mathcal{C}(\mathcal{U})$ and a sentence ψ in the language of $\mathcal{U}$, does there exist a sentence φ such that $\mathcal{U} \models \psi$ iff $\mathcal{C}(\mathcal{U}) \models \varphi$? We call this problem the *compressed model checking problem*. We can rephrase it as: given a property P on $\mathbf{K}$, what is the descriptive complexity of checking P, if $\mathcal{C}(\mathcal{U})$ rather than $\mathcal{U} \in \mathbf{K}$ is given. I.e., is there a sentence φ_P in a logic $\mathcal{L}$ such that $\mathcal{U}$ has the property P iff $\mathcal{C}(\mathcal{U}) \models \varphi_P$? Investigating the compressed model checking problem will offer methods for deciding the existence of efficient algorithms on compressed data; for example, if a property can be expressed by a first-order sentence, we know how to check it in log-space.

We study the compressed model checking problem for the case when $\mathcal{U}$ is a labeled finite order. In this case, $\mathcal{C}$ is compression of strings, and the compressed model checking problem generalizes the *compressed pattern matching problem*, i.e. the problem of checking on a compressed text whether its original version contains a given string, and probably determine its occurrence positions. This problem is important for text files but also for images and multimedia files. Searching through archives of compressed data without decompressing is a fundamental requirement for computations based on multimedia information. The first efficient algorithms to search for a substring in Lempel-Ziv compressed strings were given in [9]. Other compression schemes such as coding common pairs of letters in one byte [12] or omitting letters predictable using an antidictionary of the text [5] have been proposed, with different trade-offs between a good compression rate and the possibility to search for substrings. More generally, for each compression scheme, one can ask: which problems can be efficiently solved on compressed data?

In this paper we consider two compression schemes $\mathcal{C}$ on strings, the classical Lempel-Ziv-78 [15] compression scheme (optimal from the information theory point of view), which we call $\mathcal{LZ}$, and a naive compression by run-length encoding, which we call $\mathcal{R}$. A binary string w of length n is a finite structure $\mathcal{U} = \mathcal{S}(w)$ of size n with a unary predicate U and a linear order $<$, such that $U(i)$ is true iff the i-th bit of w is 1. For both the run-length compression and the Lempel-Ziv compression schemes $\mathcal{C}$, we will precisely define compressed structures $\mathcal{C}(\mathcal{U})$, called $\mathcal{R}(w)$ and $\mathcal{LZ}(w)$. The choice of these structures will be motivated by basic properties of the run-length compressed version $R(w)$ or the Lempel-Ziv compressed version $LZ(w)$ of w, and induce various logical languages in which properties of the compressed strings can be formulated.

The substring problem can be expressed by an existential first-order sentence ψ on $\mathcal{S}(w)$. Suppose w is given in Lempel-Ziv compressed form $LZ(w)$, for example as a .gz file. How can we express the substring problem on $\mathcal{LZ}(w)$? We will show that it is not first-order definable on $\mathcal{LZ}(w)$, but it is first-order definable on $\mathcal{R}(w)$. Generalizing the substring problem, we will focus on the descriptive complexity class of first-order definability, for both the run-length and the Lempel-Ziv compression. Our main results are:

1. For the run-length compression scheme, first-order properties on strings are also first-order definable on compressed strings. This is not the case for 2-dimensional strings (images).

2. For the Lempel-Ziv compression scheme, the first-order definability is not kept on compressed

strings. However, the first-order properties of strings can be defined by a $FO(TC)$ formula on compressed strings, i.e. a first-order formula with the Transitive Closure operator. Moreover, we define a class $\mathcal{F}$ of properties such that if a language is in $\mathcal{F}$, then its *LZ*-compression is a first-order definable language.

3. Properties expressed in $FO(TC)$ on strings can also be defined in $FO(TC)$ on compressed strings by Lempel-Ziv, whereas properties expressed in monadic SO (second-order logic) on strings are definable in binary SO on compressed strings by Lempel-Ziv.

4. For both the run-length and the Lempel-Ziv compression, the class of languages that are first-order definable on compressed strings is not closed under concatenation.

Some of the results extend to the case of strings with additional relations, or labeled ordered relational structures $\mathcal{U}$ compressed by run-length encoding or by Lempel-Ziv.

In section 2, we define the finite structures representing the strings compressed by run-length encoding and by Lempel-Ziv, and we give the formal definition of our problem in 2.4. In section 3, we study the definability on strings and images compressed by the run-length compression scheme. In section 4, we study the definability on strings compressed by the classical Lempel-Ziv compression scheme.

2 Strings and compressed strings as finite structures

A non-empty *string* w of length $n > 0$ over an alphabet $\Sigma = \{a_1, \dots, a_l\}$ is represented as a finite relational structure

$$\mathcal{S}(w) = (D_n, <, U_{a_1}, \dots, U_{a_l})$$

where $D_n = \{0, 1, \dots, n-1\}$, $<$ is the linear order on D_n inherited from $I\!N$, and $U_a \subseteq D_n$ is the unary predicate such that $\mathcal{S}(w) \models U_a(i)$ iff the $i+1$-st letter in w is a. For a binary string over $\Sigma = \{0, 1\}$, we simplify the notation $(D_n, <, U_1, U_0)$ to $(D_n, <, U)$, where $U = U_1$ and $D_n - U = U_0$. To write formulas in the first-order language $\{<, U_{a_1}, \dots, U_{a_l}\}$, we use some abbreviations like $\varphi(x+1) := \exists y(\varphi(y) \wedge x < y \wedge \neg\exists z(x < z \wedge z < y))$. Similarly for $\varphi(x+k), \varphi(x-k)$ where k is fixed. By $\varphi(max)$ we mean the formula saying $\varphi(x)$ holds at the last element; for fixed k, bounded quantifiers like $\forall x < max - k$ can be rephrased without max, k, and $-$. Moreover, we use $U_{abc}(x) := U_a(x) \wedge U_b(x+1) \wedge U_c(x+2)$ and likewise $U_w(x)$ for other strings w over Σ.

A set $L \subseteq \Sigma^+$ of strings is *definable in a logic* $\mathcal{L}$ for orders labeled by letters of Σ, if there is a sentence φ of $\mathcal{L}$ such that $L = \{w \in \Sigma^+ \mid \mathcal{S}(w) \models \varphi\}$. L is **-star free* (resp. *regular*), if it can be built from finite languages by union, elementwise concatenation, and complement (resp. closure under concatenation). The basic facts on definable properties of strings are:

Theorem 1 *[3, 8, 13] Suppose $L \subseteq \Sigma^+$. Then L is star-free (resp. regular) iff L is first-order (resp. monadic second-order) definable.*

In particular, L^+, the closure of L under concatenation, need not be first-order definable when L is. The following fact about when L^+ is first-order is worth noting. A string is called *periodic*, if it is of the form p^k with $p \neq \epsilon$ and $k > 1$, otherwise we call it *aperiodic*. For example, *abab* is periodic and *baab* is aperiodic.

Proposition 1 *1. A string s is aperiodic iff s has only two occurrences in ss.*

2. If a string s is aperiodic, then $s^+ := \{s\}^+$ is first-order definable by

$$Ind_s := U_s(0) \wedge \forall j < max - (|s| - 1)\ (U_s(j) \rightarrow U_s(j + |s|)). \tag{1}$$

In general, s^+ is not first-order definable: if we could define s^+ for $s = aa$, we could define linear orders of even length, which is impossible (cf.[7], Example 1.3.5).

2.1 Run-length compression of strings

Many string compression schemes decompose the input string $w \in \Sigma^+$ into a sequence of substrings or blocks $B_i \in \Sigma^+$ such that $w = B_0 \cdots B_{m-1}$ and return a sequence of representations of the blocks. We shall insert dots in a string to show its decomposition into blocks.

In run-length compression, or *R-compression*, if $B_0, \ldots, B_{n-1}$ are defined and $w = B_0 \cdots B_{n-1}v$ for some $v \in \Sigma^+$, then B_n is the longest prefix of v whose letters are all the same. The R-compression of w is the sequence $p_0 \cdots p_{m-1}$ of pairs $p_n = (j, a)$ such that $B_n = a^j$.

For example, the run-length block decomposition of $w = aaaaaabbbbaa$ is $aaaaaa.bbbb.aa$ and the run-length compression of w is $R(w) = (6, a)(4, b)(2, a)$. A more familiar notation is $w = a^6b^4a^2$.

We represent run-length compressed strings $R(w)$ as first-order structures

$$\mathcal{R}(w) = (D_m, D_n, <_m, <_n, U_{a_1}, \ldots, U_{a_l}, f)$$

where $(D_m, <_m, U_{a_1}, \ldots, U_{a_l})$ represents the string of second components of $R(w)$, i.e. m is the length of $R(w)$, $<_m$ the natural order on D_m, and $U_a(i)$ iff $R(w)(i) = (j + 1, a)$ for some j; further, n is the maximal block length, i.e. the maximal j such there are i and a with $R(i) = (j, a)$; $f : D_m \to D_n$ is the block length -1, i.e. $f(i) = j$ iff $R(w)(i) = (j + 1, a)$ for some a; and $<_n$ is the natural order on D_n.

In the first-order language talking about $\mathcal{R}(w)$, variables must indicate if they range over D_n or D_m, as in the formula $\exists j \in D_n \forall i \in D_m(f(i) = j)$.

2.2 Lempel-Ziv compression of strings

The classical Lempel-Ziv compression is a family of algorithms with many variations (cf. [1]). We follow a simple version of *LZ*-78 presented in [4]. The *LZ*-compression 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, otherwise B_n is v. The *LZ*-compression $LZ(w)$ of w is the sequence $p_0 \cdots p_{m-1}$ of pairs $p_n = (k, a)$ 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 $decode(p_0 \ldots p_{m-1}) = decode(p_0) \cdots decode(p_{m-1})$ where $decode((0, a)) = a$ and $decode((n + 1, a)) = decode(p_n)a$.

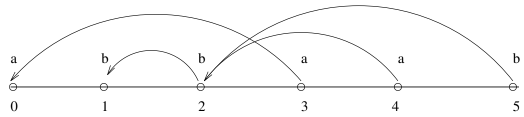
For example, the *LZ* block decomposition of $w = abbbaabbabbb$ is $a.b.bb.aa.bba.bbb$, and its *LZ*-compression is $LZ(w) = (0, a)(0, b)(2, b)(1, a)(3, a)(3, b)$.

We represent *LZ*-compressed strings $LZ(w) = p_0 \cdots p_{m-1}$ as finite labeled ordered graphs,

$$\mathcal{LZ}(w) = (D_m, <, U_{a_1}, \ldots, U_{a_l}, E).$$

As for run-length compression, $(D_m, <, U_{a_1}, \ldots, U_{a_l})$ represents the string of second components of $LZ(w)$, so m is the length of $LZ(w)$, $<$ the natural order on D_m, and $U_a(i)$ 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 node $k-1$.

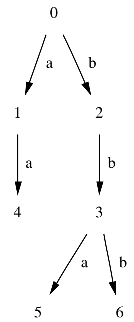
For the string $w = a.b.bb.aa.bba.bbb.$ with $LZ(w) = (0,a)(0,b)(2,b)(1,a)(3,a)(3,b)$, the graph $\mathcal{LZ}(w)$ is



Note that if we number the pairs by 1,2, etc. and add the numbers as a third component, the resulting triples

$$(0,a,1)(0,b,2)(2,b,3)(1,a,4)(3,a,5)(3,b,6)$$

represent a tree in which block B_k labels the path from the root 0 to node $k+1$:



This tree is a *trie*, i.e. all branches leaving a node are labeled with different letters (except when the last block equals a previous one, as for $w' = a.b.bb.aa.bba.bba.$).

In case of a compressed binary string $LZ(w)$, we have a structure $\mathcal{LZ}(w) = (D_m, <, U, E)$ where U is the unary predicate U_1 and its complement is U_0.

2.3 Definability under compression

We identify a property P of strings over Σ with the formal language $L = \{w \in \Sigma^+ \mid w \text{ has } P\} \subseteq \Sigma^+$. We say that $L \subseteq \Sigma^+$ is *first-order definable on strings*, if there is a first-order sentence φ in the language $\{<, U_{a_1}, \ldots, U_{a_l}\}$ of $\mathcal{S}$-structures such that $L = \{w \in \Sigma^+ \mid \mathcal{S}(w) \models \varphi\}$.

Suppose we have a scheme $\mathcal{C}$ that maps strings w of a class **K** to compressed strings $\mathcal{C}(w)$ viewed as finite structures of a class **K'**, and that $\mathcal{L}$ is a logic for structures in **K'**.

Definition 1 *A property P of strings is $\mathcal{C}$-*definable in $\mathcal{L}$ *if there exists a sentence φ of $\mathcal{L}$ such that for all strings w: $w \in P$ iff $\mathcal{C}(w) \models \varphi$. We say that $L \subseteq \Sigma^+$ is* first-order definable on $\mathcal{C}$-compressed strings *if there is a first-order sentence φ in the language of $\mathcal{C}$-structures such that $L = \{w \in \Sigma^+ \mid \mathcal{C}(w) \models \varphi\}$.*

We are interested in how the properties definable on strings are related to the properties definable on compressed strings, and in how the class of properties of strings that are definable on compressed strings behaves. We study the following example problems:

The substring query The substring query for a fixed pattern string s asks whether a given string w contains s as a substring. It is first-order definable on strings by $\exists x\, U_s(x)$. A more general classical question is the problem of finding the occurrences of the pattern in the given string, i.e. of computing $\{(w,i) \mid \mathcal{S}(w) \models U_s(i)\}$.

Closure under concatenation The concatenation of two first-order definable languages is a first-order definable language, by Theorem 1. Is the same true under compression, i.e. if L_1 and L_2 are first-order definable on compressed strings, is their concatenation L_1L_2 also first-order on compressed strings?

3 Definability on run-length compressed strings

We show that first-order queries on strings can be expressed as first-order queries on run-length compressed strings. To translate non-boolean queries, we use the the mapping h from elements of $\mathcal{S}(w)$ to pairs of elements of $\mathcal{R}(w)$ such that $h(i) = (k, j)$ iff position i is the $j+1$-st relative position in block B_k, i.e. iff $i = \Sigma_{i'<k}(f(i')+1) + (j+1)$.

Theorem 2 *For every first-order formula $\varphi(x_1,\ldots,x_n)$ on strings, there is a first-order formula $\varphi^R(x_1,y_1,\ldots,x_n,y_n)$ on run-length compressed strings, such that for each w and all $i_1,\ldots,i_n \in \mathcal{S}(w)$,*

$$\mathcal{S}(w) \models \varphi(i_1,\ldots,i_n) \iff \mathcal{R}(w) \models \varphi^R(h(i_1),\ldots,h(i_n)). \tag{2}$$

Proof: By induction on the formula $\varphi(x_1,\ldots,x_n)$, we define $\varphi^R(x_1,y_1,\ldots,x_n,y_n)$ by

$$\begin{array}{rcl}
(U_a(x_1))^R & := & U_a(x_1), \\
(x_1 = x_2)^R & := & (x_1 = x_2 \wedge y_1 = y_2), \\
(x_1 < x_2)^R & := & x_1 <_m x_2 \vee (x_1 = x_2 \wedge y_1 <_{|w|} y_2), \\
(\neg\varphi)^R & := & \neg(\varphi^R), \\
(\varphi_1 \wedge \varphi_2)^R & := & (\varphi_1^R \wedge \varphi_2^R), \\
(\exists x_{n+1}\varphi\,(x_1,\ldots,x_n))^R & := & \exists x_{n+1}\exists y_{n+1}\,[\,y_{n+1} \leq_{|w|} f(x_{n+1}) \wedge \varphi^R(x_1,y_1,\ldots,x_{n+1},y_{n+1})\,].
\end{array}$$

The verification of (2) is left to the reader. (See the definition of $\mathcal{R}(w)$ for the relations $<_m$ and $<_{|w|}$.) $\Box$

Corollary 1 *Every *-free language L is first-order definable on R-compressed strings.*

Corollary 2 *The substring query is first-order definable on R-compressed strings.*

Some properties of strings that are first-order on R-compressed strings are not first-order on strings. For example, $L = \{0^n1^n \mid 0 < n\}$ is not regular, hence by Büchi's result it is not even monadic-second-order on strings. But L is first-order on R-compressed strings, because $w \in L$ iff

$$\mathcal{R}(w) \models \quad \neg U(0) \wedge U(max) \wedge f(0) = f(max) \wedge \exists x \in D_m \forall y (U(y) \leftrightarrow x < y).$$

The previous example shows directly:

Corollary 3 *Language concatenation is not first-order definable on R-compressed strings (but is definable on structures with the addition function).*

An interesting generalization of the substring query is the subimage query. We consider images as 2-dimensional strings, i.e. an image is a function $\chi : D_n \times D_n \to \Sigma$ which gives for each pixel (i_1, i_2) its colour $\chi(i_1, i_2)$. We represent images as structures

$$\mathcal{I} = (D_n, <, U_{a_1}, \ldots, U_{a_l}),$$

where each U_a is the *binary* predicate on D_n given by $U_a(i_1, i_2)$ iff $\chi(i_1, i_2) = a$, and $<$ the ordering on D_n. Compressed images are obtained by compressing the strings that arise from the concatenation of all lines, i.e. $\chi(0,0)\cdots\chi(0,n-1)\ \cdots\ \chi(n-1,0)\cdots\chi(n-1,n-1)$.

The *subsquare query* over the binary alphabet $\Sigma = \{\circ, \bullet\}$ asks for the existence of a square of four $\bullet$ symbols. We can define it in first-order on images by

$$\varphi_{sq} := \exists i_1 \exists i_2 [U_\bullet(i_1, i_2) \wedge U_\bullet(i_1+1, i_2)\ \wedge\ U_\bullet(i_1, i_2+1) \wedge U_\bullet(i_1+1, i_2+1)\,],$$

where (i_1, i_2) refers to the position of the bottom left corner of the square. We can then prove:

Theorem 3 *The subsquare query is not first-order on run-length compressed images.*

Proof: We construct two images, $\mathcal{I}_1$ without the square pattern, $\mathcal{I}_2$ with the square pattern, and use a standard Ehrenfeucht-Fraïssé game to show that their compressions $\mathcal{R}(\mathcal{I}_1)$ and $\mathcal{R}(\mathcal{I}_2)$ are k-elementarily equivalent. Hence the subsquare query cannot be defined by a first-order sentence of quantifier rank k on run-length compressed images.

The example images Each image $\mathcal{I}_1$ and $\mathcal{I}_2$ is an $n \times n$-matrix filled in with $\circ$'s and $\bullet$'s. All lines of the matrix are filled with $\circ$'s except for the first two, which look as follows, for suitable i_1, i_2, j_1, j_2 to be fixed later:

$$\begin{array}{ll}\mathcal{I}_1 : & \bullet(\circ\bullet\circ\bullet)^{i_1}(\circ\circ\bullet\bullet)^{i_2}\ \circ\circ\circ\circ\bullet\bullet\circ\bullet\bullet\circ\bullet\bullet(\circ\circ\bullet\bullet)^{j_1}(\circ\circ\bullet\bullet)^{j_2}\ \circ\circ \\ & \bullet\bullet\ (\circ\circ\bullet\bullet)^{i_1}(\circ\circ\bullet\bullet)^{i_2}\ \circ\circ\bullet\bullet\circ\circ\circ\bullet\bullet\circ\bullet\bullet(\circ\circ\bullet\bullet)^{j_1}(\circ\circ\bullet\bullet)^{j_2}\circ\end{array}$$

$$\begin{array}{ll}\mathcal{I}_2 : & \bullet(\circ\bullet\circ\bullet)^{i_1}(\circ\circ\bullet\bullet)^{i_2}\ \circ\circ\circ\circ\bullet\ \bullet\circ\underline{\bullet\bullet}\ \circ\bullet\ \bullet\ (\circ\circ\bullet\bullet)^{j_1}(\circ\circ\bullet\bullet)^{j_2}\ \circ\circ \\ & \bullet\bullet\ (\circ\circ\bullet\bullet)^{i_1}(\circ\circ\bullet\bullet)^{i_2}\ \circ\circ\bullet\bullet(\circ\circ\bullet\bullet)^{j_1}\ \circ\circ\circ\ \bullet\bullet\circ\bullet\ \ \bullet\ (\circ\circ\bullet\bullet)^{j_2}\circ\end{array}$$

The first line of the two images is the same. In $\mathcal{I}_1$, the $\bullet$'s are misaligned in the first two lines, hence a subsquare of four $\bullet$'s does not exist. In $\mathcal{I}_2$, the subsquare occurs at the underlined position, if $0 < j_1$.

Writing $\circ^m$ and $\bullet^m$ instead of $(m, \circ)$ and $(m, \bullet)$, and using $i = i_1 + i_2, j = j_1 + j_2$, the run-length compressions of the images are:

$$\begin{array}{l}\mathcal{R}(\mathcal{I}_1) : \bullet^1(\circ^1\bullet^1)^{2i_1}(\circ^2\bullet^2)^{i_2}\ \circ^4\ \bullet^2\ \circ^1\ \bullet^2\ \circ^1\ \bullet^2(\circ^2\bullet^2)^j\ \circ^2\ .\ \bullet^2\ (\circ^2\bullet^2)^i\ \circ^2\ \bullet^2\ \circ^3\ \bullet^2\ \circ^1\ \bullet^2(\circ^2\bullet^2)^j\circ^{1+n'} \\ \mathcal{R}(\mathcal{I}_2) : \bullet^1(\circ^1\bullet^1)^{2i_1}(\circ^2\bullet^2)^{i_2}\ \circ^4\ \bullet^2\ \circ^1\ \bullet^2\ \circ^1\ \bullet^2(\circ^2\bullet^2)^j\ \circ^2\ .\ \bullet^2\ (\circ^2\bullet^2)^{i+j_1}\ \circ^2\ \bullet^2\ \circ^3\ \bullet^2\ \circ^1\ \bullet^2(\circ^2\bullet^2)^{j_2}\circ^{1+n'},\end{array}$$

where $n' = n(n-2)$; to improve readability, we inserted the . to mark the end of the first line.

To define a winning strategy in an Ehrenfeucht-Fraïssé-game on the R-compressed images, it is convenient to use 2-dimensional letters

$$A = \boxed{\begin{array}{c}\bullet^1 \\ \bullet^2\end{array}}\quad B = \boxed{\begin{array}{c}\circ^1\bullet^1 \\ \circ^2\bullet^2\end{array}}\qquad C = \boxed{\begin{array}{c}\circ^2\bullet^2 \\ \circ^2\bullet^2\end{array}}\quad D = \boxed{\begin{array}{c}\circ^4\ \bullet^2\ \circ^1\ \bullet^2\ \circ^1\bullet^2 \\ \circ^2\ \bullet^2\ \circ^2\ \circ^2\ \bullet^2\end{array}}$$

$$E = \boxed{\begin{array}{c}\circ^2\ \bullet^2\ \circ^2\bullet^2 \\ \circ^3\ \bullet^2\ \circ^1\bullet^2\end{array}}\quad F = \boxed{\begin{array}{c}\circ^2\bullet^2 \\ \ \end{array}}\quad G = \boxed{\begin{array}{c}\circ^2 \\ \circ^{1+n'}\end{array}}$$

A	B^{2i_1}	$C^{i_2-i_1+1}$	E	C^{i_1-3}	D	C^{j-i_1}	F^{i_1}	G
$\bullet^1$	$(\circ^1\bullet^1)^{2i_1}$	$(\circ^2\bullet^2)^{i_2-i_1+1}$	$(\circ^2\bullet^2)^2$	$(\circ^2\bullet^2)^{i_1-3}$	$\circ^4(\bullet^2\circ^1)^2\bullet^2$	$(\circ^2\bullet^2)^{j-i_1}$	$(\circ^2\bullet^2)^{i_1}$	$\circ^2$
$\bullet^2$	$(\circ^2\bullet^2)^{2i_1}$	$(\circ^2\bullet^2)^{i_2-i_1+1}$	$\circ^3\ \bullet^2\ \circ^1\bullet^2$	$(\circ^2\bullet^2)^{i_1-3}$	$(\circ^2\bullet^2)^3$	$(\circ^2\bullet^2)^{j-i_1}$		$\circ^{1+n'}$

Figure 1: $\mathcal{R}(\mathcal{I}_1)$ as sequence $\mathcal{E}_1$ with two-dimensional letters

by which we relate two regions of $\mathcal{R}(\mathcal{I}_j)$ that belong to the first and second lines of $\mathcal{I}_j$. The R-compressed image can then be written as a string over $\{A, \ldots, G\}$, i.e. it can be transformed to a 1-dimensional structure $\mathcal{E} = (D_m, <, U_A, \ldots, U_G)$.

The compressed images $\mathcal{R}(\mathcal{I}_1)$ and $\mathcal{R}(\mathcal{I}_2)$ can be viewed (cf. figure 1) as strings

$$\mathcal{E}_1 = AB^{2i_1}C^{i_2-i_1+1}EC^{i_1-3}DC^{j_1+j_2-i_1}F^{i_1}G$$

and, with E shifted j_1 positions to the right,

$$\mathcal{E}_2 = AB^{2i_1}C^{j_1+i_2-i_1+1}EC^{i_1-j_1-3}DC^{j_1+j_2-i_1}F^{i_1}G.$$

If the parameters i_1, j_1, i_2, j_2 are large enough relative to k, one can exhibit a winning strategy for Duplicator: the string in each structure is generally homogeneous except in six points where the local neighborhoods are distinct: the neighborhood of the letters A, E, D and G and the borders BC and CF. We call these points *boundary* points. We can now specify the values of $i_1, j_1, i_2, j_2 = 2^k$ and prove:

Lemma 1 $\mathcal{E}_1 \equiv_k \mathcal{E}_2$.

Proof: We give a winning strategy for Duplicator in a classical Ehrenfeucht-Fraïssé game by induction on the number r of moves from 1 to k. For $i = 1$ or $i = 2$, let $S_{i,r} = \{a_1, ..., a_{r+6}\}$ be the points played up to stage r together with the boundary points in $\mathcal{E}_i$ and let $\mathcal{E}_{i,r}$ be the substructure of $\mathcal{E}_i$ restricted to neighborhoods of $S_{i,r}$ at distance 2^{k-r+1}. We show that if $\mathcal{E}_{1,r}$ and $\mathcal{E}_{2,r}$ are partially isomorphic, so are $\mathcal{E}_{1,r+1}$ and $\mathcal{E}_{2,r+1}$.

Assume that Spoiler plays in $\mathcal{E}_{1,r}$ a new point a_{r+7}. If a_{r+7} is within a distance 2^{r-k+1} of $a' \in S_{1,r}$, then Duplicator plays the point b_{r+7}, the isomorphic copy of a_{r+7} in the neighborhood of b', the isomorphic copy of a'. It exists by induction hypothesis. If a_{r+7} is far from any point of $S_{1,r}$, i.e. at distance more than 2^{k-r+1}, Duplicator plays b_{r+7} on a similar neighborhood far from any points in $S_{2,r}$: it is always possible because $i_1, j_1, i_2, j_2 = 2^k$. We conclude that $\mathcal{E}_{1,r+1}$ and $\mathcal{E}_{2,r+1}$ are partially isomorphic. $\Box$

Lemma 2 $\mathcal{R}(\mathcal{I}_1) \equiv_k \mathcal{R}(\mathcal{I}_2)$.

Proof: We use Duplicator's winning strategy on $\mathcal{E}_1$ and $\mathcal{E}_2$ to obtain a winning strategy for him on $\mathcal{R}(\mathcal{I}_1)$ and $\mathcal{R}(\mathcal{I}_2)$. For $i = 1$ or $i = 2$, each element of $\mathcal{E}_i$, i.e. occurrence of a letter in the string $\mathcal{E}_i$, corresponds to two intervals on $\mathcal{R}(\mathcal{I}_i)$, which in turn represent an interval on the first and an interval on the second line of $\mathcal{I}_i$. (The latter possibly extends to the end of the remaining lines of $\mathcal{I}_i$.) Let h_i be the mapping from $\mathcal{R}(\mathcal{I}_i)$ to $\mathcal{E}_i$ that maps a point $x \in \mathcal{R}(\mathcal{I}_i)$ to the corresponding point in $\mathcal{E}_1$, i.e. the letter X such that x belongs to one of the two intervals corresponding to X on $\mathcal{R}(\mathcal{I}_i)$.

At any stage r from 1 to k, if Spoiler plays a point a in $\mathcal{R}(\mathcal{I}_1)$, consider its image point $a' = h_1(a) \in \mathcal{E}_1$ as Spoiler's move in the Ehrenfeucht-Fraïssé game on $\mathcal{E}_1$ and $\mathcal{E}_2$. Let $b' \in \mathcal{E}_2$ be Duplicator's response according to his winning strategy for the game on $\mathcal{E}_1$ and $\mathcal{E}_2$. Then a' and b' carry the same letter X, so the pairs of intervals in $\mathcal{R}(\mathcal{I}_1)$ and $\mathcal{R}(\mathcal{I}_2)$ corresponding to a' and b' are isomorphic pairs of labeled orders. Duplicator picks the image of a under this isomorphism as his response b with $h(b) = b'$. This clearly defines a winning strategy for Duplicator on $\mathcal{R}(I_1)$ and $\mathcal{R}(I_2)$. Hence the compressed structures $\mathcal{R}(\mathcal{I}_1)$ and $\mathcal{R}(\mathcal{I}_2)$ are k-elementarily equivalent. □

This concludes the proof of theorem 3. □

Notice that the elementary equivalences would generalize on structures with additions, i.e.

$$\mathcal{R}(w) = (D_m, D_n, <_m, +_n, U_{a_1}, \ldots, U_{a_l}, f)$$

instead of order on D_n, with slightly more complicated strategies.

4 Definability on Lempel-Ziv-compressed strings

We will show that first-order properties on strings generally are not first-order definable on *LZ*-compressed strings but are $FO(TC)$ definable. We then define a class of first-order properties on strings which *are* first-order on *LZ*-compressed strings. Let us start with such an example.

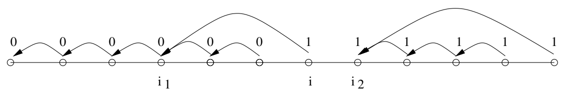
Example 1 *The regular language* $L = 00^+11^+ = \{0^m\ 1^p \mid m, p \geq 2\}$ *is defined by*

$$\psi = \ \exists i < max\, [1 < i \wedge \forall j (i \leq j \leftrightarrow U(j))].$$

The string $s = 0^{25}1^{13} \in L$*,*

$$0.00.000.0000.00000.000000.00001.1.11.111.1111.11,$$

has the following LZ-graph:



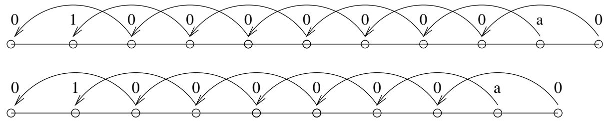
The compression $\mathcal{LZ}(L)$ *of L is made of two consecutive linear orders (except for the last points), one for the sequence of "0's" and one for the sequence of "1's". It can be defined by* $\psi^{LZ} =$

$$\begin{aligned}\exists i, i_1, i_2 \quad & [\ (i_1 < i < i_2) \\ & \wedge (\forall j (U(j) \leftrightarrow i \leq j) \vee \forall j (U(j) \leftrightarrow i < j)) \\ & \wedge \forall j \forall k (\ E(k,j) \leftrightarrow \quad (k = i \wedge j = i_1) \vee (k = max \wedge j = i_2) \\ & \qquad\qquad\qquad\qquad \vee (k \neq 0 \wedge k \neq i \wedge k \neq i+1 \\ & \qquad\qquad\qquad\qquad\quad \wedge k \neq max \wedge j = k-1))\ ].\end{aligned}$$

4.1 First-order properties on strings and on *LZ*-compressed strings

We now show that first-order on strings in general can not be expressed by first-order sentences on *LZ*-compressed strings. Not even the substring query is first-order on *LZ*-compressed strings. Thus, finding a substring is 'difficult' to do on *LZ*-compressed structures, essentially because the *LZ*-code of a substring occurrence(!) depends on its left context.

Example 2 *Consider the alphabet $\Sigma = \{0, 1, a\}$ and the language $L = \{0, 1\}^*\ a\ 0^*$. Let $LZ(w_1)$ and $LZ(w_2)$ be the two compressed strings represented by the following graphs $\mathcal{LZ}(w_1)$ and $\mathcal{LZ}(w_2)$:*



The decompressed strings are

$$\begin{aligned} w_1 \in L &\ :\ 0.1.00.10.000.100.0000.1000.00000.1000a.000000, \\ w_2 \notin L &\ :\ 0.1.00.10.000.100.0000.1000.0000a.10000. \end{aligned}$$

Lemma 3 *There is a first-order property on strings which is not first-order definable on LZ-compressed strings.*

Proof: Consider the first-order definable language L of example 2. Suppose it were first-order definable on *LZ*-compressed strings by a sentence ψ_k of quantifier rank k. If $\mathcal{LZ}(w_1)$ and $\mathcal{LZ}(w_2)$ are two orders of length m and $m-1$ where $m > 2^{k+1}$, then $\mathcal{LZ}(w_1) \equiv_k \mathcal{LZ}(w_2)$, by an Ehrenfeucht-Fraïssé argument. As $w_1 \in L, w_2 \notin L$ and compression is injective, we have $LZ(w_1) \in LZ(L)$, $LZ(w_2) \notin LZ(L)$. Hence $\mathcal{LZ}(w_1) \models \psi_k$ and $\mathcal{LZ}(w_2) \not\models \psi_k$, contradicting $\mathcal{LZ}(w_1) \equiv_k \mathcal{LZ}(w_2)$. □

Corollary 4 *The substring query is not first-order definable on LZ-compressed strings.*

Proof: Let $s = a0$ and consider example 2. The substring s occurs in w_1 but not in w_2, yet the compressed strings are k-elementarily equivalent. So no first-order formula of quantifier rank k in the language of the compressed structures can define the substring query. □

Corollary 5 *Language concatenation is not first-order definable on LZ-compressed strings.*

Proof: We note that the compression of $L_1 = \{0, 1\}^+$ and of $L_2 = a\{0\}^+$ are first-order definable. By a slight modification of the above example, the compression of $L_1 L_2$ is not. □

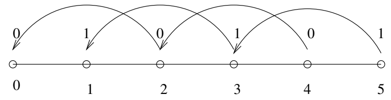
We can also answer the reverse question:

Lemma 4 *There exists a first-order property on the class of LZ-compressed strings which is not first-order definable on the class of strings.*

Proof: Consider the class of compressed strings defined by

$$\begin{aligned} \varphi \ :=\ & \neg U(0) \wedge U(1) \wedge \forall i \geq 2\,(U(i) \leftrightarrow U(i-2)) \\ & \wedge \forall i \forall j (E(i,j) \leftrightarrow (i \geq 2 \wedge j = i-2)) \end{aligned}$$

A graph of size 6 satisfying φ is



Note that $\mathcal{LZ}(w) \models \varphi$ iff

$$w = 0.1.00.11.000.111.\ \ldots\ .0^p.1^p \text{ for some } p.$$

The set of these strings w is not a regular language, hence it is not first-order on strings. □

4.2 First-order on strings is first-order in TC on *LZ*-compressed strings

We now show that we can define all first-order properties of strings in a classical extension of the first-order formulas of *LZ*-structures. Notice that it is P-complete to obtain the *LZ*-decomposition [6] and that the E relation is not $FO(TC)$ definable on the original string.

A formula is in the language $FO(TC^k)$ if it can be built when the constructions for first-order formulas are extended by: if φ is a formula and $\overline{x}, \overline{y}$ are two sequences of length k of pairwise distinct variables, then $TC(\overline{x}, \overline{y})\varphi$ is a formula, where $\overline{x}, \overline{y}$ are free in $TC(\overline{x}, \overline{y})\varphi$ but bound in its subformula φ. For $\varphi(\overline{x}, \overline{y}, \overline{z})$, the meaning is

$$\begin{aligned} &\mathcal{A} \models (TC(\overline{x}, \overline{y})\varphi)[\overline{a}, \overline{b}, \overline{c}] \\ &\quad \iff \quad (\overline{a}, \overline{b}) \text{ is in the transitive closure of } \{(\overline{u}, \overline{v}) \mid \mathcal{A} \models \varphi[\overline{u}, \overline{v}, \overline{c}]\}. \end{aligned}$$

If $\varphi(x, y)$ has at most two free variables, we write $\varphi^+(x, y)$ instead of $TC(x, y)\varphi$. Let us see how TC^1 can be used to define a substring query on *LZ*-compressed strings.

Example 3 *Consider occurrences of the substring $s = 10$ in binary strings. Since each prefix of a block is (the content of) a previous block, it is sufficient to define occurrences of 10 with block borders as in $10.$ or 1.0, and these can be defined by the formula*

$$\exists i, j\, [\, (\neg U(i) \wedge E(i, j) \wedge U(j)) \ \vee \ (U(i) \wedge E^+(i + 1, j) \wedge \forall k \neg E(j, k) \wedge \neg U(j))\,].$$

In the first case, we check the symbols 0 at the end of block B_i and 1 at the node obtained by following one back-edge. In the second case we can check 1 at the end of block B_i, but to check whether 0 is the first *symbol of block B_{i+1} requires to follow the back-edges (using TC) from node $i + 1$ all the way to find the block B_j containing the prefix of block B_{i+1} with length 1.*

The example suggests to code a position in the original string by a pair of nodes in the *LZ*-compressed string: the positions in block B_k uniquely correspond to the blocks containing the non-empty prefixes of B_k. Hence the set of positions $i \in \mathcal{S}(w)$ covered by B_k is represented in $\mathcal{LZ}(w)$ by

$$\{(k, j) \mid \mathcal{LZ}(w) \models E^*(k, j)\},$$

where E^* is the transitive reflexive closure of E. For example, if $w = a.aa.ab.aba.aa$, the positions 5,6,7 contained in $B_3 = aba$ 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 4 *For every first-order formula $\varphi(x_1, \ldots, x_n)$ on strings, there is a $FO(TC^1)$-formula $\varphi^{LZ}(x_1, y_1, \ldots, x_n, y_n)$ on LZ-graphs, such that for each $\mathcal{S}(w)$ and all $i_1, \ldots, i_n \in \mathcal{S}(w)$,*

$$\mathcal{S}(w) \models \varphi(i_1, \ldots, i_n) \quad \iff \quad \mathcal{LZ}(w) \models \varphi^{LZ}(h(i_1), \ldots, h(i_n)).$$

Here, h is the mapping from elements of $\mathcal{S}(w)$ to pairs of elements of $\mathcal{LZ}(w)$ such that $h(i) = (k, j)$ iff position i lies in block B_k and B_j is the prefix of B_k ending in position i.

We will only need TC to define E^*, using $x = y \vee TC(x, y)E(x, y)$. Since E is a deterministic relation, the translation in fact will go to $FO(DTC^1) \subseteq FO(TC^1)$.

Proof: Omitting the cases of equality and boolean connectives, we define φ^{LZ} by

$$\begin{array}{lcl}(U_a(x_1))^{LZ} & := & U_a(y_1),\\ (x_1 < x_2)^{LZ} & := & x_1 < x_2 \vee (x_1 = x_2 \wedge y_1 < y_2),\\ (\exists x_{n+1}\varphi\,(x_1,\ldots,x_n))^R & := & \exists x_{n+1}\exists y_{n+1}\,[\,E^*(x_{n+1},y_{n+1}) \wedge \varphi^{LZ}(x_1,y_1,\ldots,x_{n+1},y_{n+1})\,].\end{array}$$

We partly verify the first and last clause. If $\mathcal{S}(w) \models U_a(i)$, and $h(i) = (k,j)$, then B_j is the shortest prefix of B_k that covers position i; therefore, B_j ends in the letter a, which is the label at node j of $\mathcal{LZ}(w)$, so $\mathcal{LZ}(w) \models U_a(j)$. Similarly for the converse. For the last clause, assume that

$$\mathcal{LZ}(w) \models E^*(k,j) \wedge \varphi^{LZ}(h(i_1),\ldots,h(i_n),(k,j))$$

for $i_1,\ldots,i_n \in \mathcal{S}(w)$ and some $k,j \in \mathcal{LZ}(w)$. Nodes k and j represent blocks B_k and B_j of the LZ block decomposition of w. Since j can be reached from k by an E-chain, block B_j is a prefix of block B_k. So, if $i_{n+1} \in \mathcal{S}(w)$ is the endposition of this prefix of block B_k, we have $(k,j) = h(i_{n+1})$ and thus, by induction, $\mathcal{S}(w) \models \varphi(i_n,\ldots,i_{n+1})$. Hence $\mathcal{S}(w) \models \exists x_{n+1}\varphi(i_1,\ldots,i_n)$. □

Corollary 6 *The substring query is $FO(TC)$ definable on LZ-compressed strings.*

Because the translation only uses DTC^1, all first-order properties on strings are LOGSPACE computable on *LZ*-compressed strings.

Remark: We can reconstruct positions in $\mathcal{S}(w)$ from answers $(k,j) \in E^*$ to *LZ*-translated queries in $\mathcal{LZ}(w)$. To see this, note that

$$h(i) = (k,j) \iff i = |B_0| + \ldots + |B_{k-1}| + |B_j| - 1.$$

But we can define $|B_k|$ from k on $\mathcal{LZ}(w)$ by *recursion* to the prefixes B_j of B_k using E, and apply external arithmetics to sum the $|B_m|$.

4.3 Translating FO(TC)- and second-order properties

The above translation can be extended to $FO(TC^k)$, using –here for $k = 1$–

$$(TC(x_1,x_2)\varphi)^{LZ} \quad := \quad TC((x_1,y_1),(x_2,y_2))(E^*(x_1,y_1) \wedge E^*(x_2,y_2) \wedge \varphi^{LZ}).$$

Corollary 7 *A $FO(TC^k)$-property on strings is $FO(TC^{2k})$ on LZ-compressed strings.*

It can also be extended to second-order formulas, by doubling the arity of quantified relations, as in

$$(\exists X^1\varphi)^{LZ} \;:=\; \exists X^2(\forall x\forall y(X^2(x,y) \rightarrow E^*(x,y)) \;\wedge\; \varphi^{LZ}).$$

Corollary 8 *A monadic SO-property on strings is dyadic SO on LZ-compressed strings.*

This works because h of the theorem is a bijection between $|w|$ and E^*, so an m-ary relation R on $\mathcal{S}(w)$ corresponds to the $2m$-ary relation $h(R)$ on $\mathcal{LZ}(w)$.

Similarly, we can also use *LZ*-compression on strings with additional relations. For example, we can compress labeled graphs if we first impose a total linear order and then use *LZ*-compression on the resulting string with the edge relation. However, properties involving an additional relation R can be read off from the compressed structure only if it is equipped with the relation $h(R)$.

4.4 When first-order on strings remains first-order on *LZ*-compressed strings

We will now present a class of first-order properties of strings which remain first-order on *LZ*-compressed strings. This class is closed under a limited use of iterated concatenation.

Definition 2 *Let $\mathcal{F}$ be the class of languages $L \subseteq \Sigma^+$ built by union, intersection and complement from finite languages and the languages us^+, where $u \in \Sigma^*$ and $s \in \Sigma^+$ is aperiodic.*

By Theorem 1 and Proposition 1, $\mathcal{F}$ is a class of first-order definable languages.

Theorem 5 *If $L \in \mathcal{F}$, then L is first-order definable on LZ-compressed strings.*

The proof is by induction on the definition of L and a slight generalization of

If $s \in \Sigma^+$ is aperiodic, then s^+ is first-order definable on LZ-compressed strings.

The rest of the paper outlines the proof of this claim (Lemma 7 below), which transfers Proposition 1 to *LZ*-compressed strings. The main observation is that for $w \in s^+$ large enough, the ordered labeled graphs $\mathcal{LZ}(w)$ have a repeated pattern, a sequence of isomorphic subgraphs, of which any two adjacent ones are linked by back-edges in the same way. The existence of such a pattern, though not sufficient for Lemma 7, is not straightforward. It is best shown by ignoring how *LZ*-compression behaves at the end of w; therefore, we turn to the compression of the infinite string s^ω.

Let $s^\omega = sss\cdots$ be the infinite string over Σ obtained by the concatenation of infinitely many copies of s. The *LZ*-block decomposition of s^ω is the infinite sequence

$$B_0, B_1, B_2, B_3 \cdots, \tag{3}$$

where $B_0 B_1 \cdots = s^\omega$ and $B_m \in \Sigma^+$ is the shortest nonempty prefix of $B_m B_{m+1} \cdots$ that is not among $B_0, B_1, \dots, B_{m-1}$. Then (3) gives an *infinite LZ*-graph,

$$\mathcal{LZ}(s^\omega) = (D_\omega, <, U_{a_1}, \dots, U_{a_l}, E)$$

of order type ω, whose label at node $i \in D_\omega$ is the last letter of block B_i.

It is useful to think of $\mathcal{LZ}(s^\omega)$ as a sequence of dominoes, as follows. For a string t, let $u <_{\mathit{prefix}} t$ mean that $t = uv$ for some $v \in \Sigma^+$, and $v <_{\mathit{suffix}} t$ that $t = uv$ for some $u \in \Sigma^+$. We also use $v <_{\mathit{suffix}} t$ for infinite strings t. Note that s^ω has only a finite *suffix-set*, namely

$$\{vs^\omega \mid v <_{\mathit{suffix}} p_s\},$$

where p_s is the shortest $p \leq_{\mathit{prefix}} s$ such that $p^k = s$ for some k.

For example, if $s = abaaba$, the suffix-set of s^ω is $\{vs^\omega \mid v <_{\mathit{suffix}} aba\}$, because $baabas^\omega = bas^\omega, aabas^\omega = as^\omega, abas^\omega = s^\omega$.

Let $p_s = u_0 v_0 = \ldots = u_{|p_s|} v_{|p_s|}$ be all splittings of p_s into prefix u_i and suffix v_i, where $|v_i| = i$. For each j there are unique $l, r < |p_s|$ such that $B_j B_{j+1} \cdots = v_l s^\omega$ and $B_{j+1} \cdots = v_r s^\omega$; we call the ordered pair $C_j = [l, r]$ *the class* of block B_j. Except for small j, $B_j = v_l s^k u_r$ for some k.

Example 4 *Consider the aperiodic string $s = aba$. The LZ-block decomposition of s^ω gives a sequence of blocks B_n with classes C_n beginning as follows:*

$$\begin{array}{ccccccccccccccccc} a & b & aa & ba & ab & aab & as & s & sa & baa & baab & asa & bas & sab & asab & as^2 & \cdots \\ [0,2] & [2,1] & [1,2] & [2,0] & [0,1] & [1,1] & [1,0] & [0,0] & [0,2] & [2,2] & [2,1] & [1,2] & [2,0] & [0,1] & [1,1] & [1,0] & \cdots \end{array}$$

For example, the class C_0 of the first block $B_0 = a$ is $[0,2]$, because $B_0B_1 \cdots = s^\omega = v_0 s^\omega$ and $B_1B_2 \cdots = bas^\omega = v_2 s^\omega$. Block $B_{10} = baab$ has class $C_{10} = [2,1]$, because $B_{10}B_{11} \cdots = bas^\omega$ gives the first component $2 = |ba|$ and $B_{11} \cdots = as^\omega$ gives the second component $1 = |a|$.

A *domino* (over p_s) is just an ordered pair $[l,r]$ of numbers $l,r < |p_s|$; its first component is its *type*. On $\{0,\ldots,|p_s|-1\}$, let $\succ := \{(k+1,k) \mid k < |p_s|-1\} \cup \{(0,|p_s|-1)\}$. A sequence

$$C = [l_0,r_0][l_1,r_1][l_2,r_2]\cdots$$

of dominoes $[l_j,r_j]$ over p_s is called a *LZ-domino sequence*, if it satisfies two conditions:

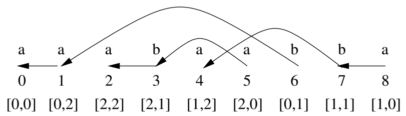
1. **Domino rule**: For all $j \in I\!N$, $l_{j+1} = r_j$, i.e. $[l_j,r_j][l_{j+1},r_{j+1}]$ match like dominoes,
2. *LZ*-**rule**: For all $j \in I\!N$, if j' is the smallest index $> j$ with $l_j = l_{j'}$, then $r_j \succ r_{j'}$.

The domino rule says that the *LZ*-block B_{j+1} is a prefix of the suffix of s^ω left by removing its prefix $B_0 \cdots B_j$. The *LZ*-rule expresses that when a prefix block $B_{j'}$ is taken from $v_l s^\omega$, it extends the last block B_j taken from $v_l s^\omega$ by a single letter.

Every *LZ*-domino sequence $C = C_0C_1\cdots$ over p_s with $C_i = [l_i,r_i]$ defines a labeled ordered graph $\mathcal{G}_C = (D_{|C|}, <, U_{a_1},\ldots,U_{a_l}, E)$, where $U_a(i)$ iff $av_{r_i} \leq_{\textit{suffix}} p_s$ and $E(i,j)$ iff $j < i$, $l_i = l_j$, $r_j \succ r_i$ and there is no C_k with $j < k < i$ of type l_i. This also works if C has length ω.

Example 4 *(Cont. 1) If $p_s = aba$ and $C = [0,0][0,2][2,2][2,1][1,2][2,0][0,1][1,1][1,0]$, then*

$\mathcal{G}_C =$
$C =$



In particular, the LZ-graph of s^ω is completely determined by p_s and the sequence

$$C \;=\; C_0\,C_1\,C_2\cdots \tag{4}$$

of block classes C_j of the B_j: one easily checks that $\mathcal{G}_C = \mathcal{LZ}(s^\omega)$. This representation of $\mathcal{LZ}(s^\omega)$ as an *LZ*-domino sequence C allows us to show the existence of a repeating subgraph by showing that C is ultimately periodic.

Lemma 5 *Let C be an infinite LZ-domino sequence. There are finite strings S,T such that $C = TS^\omega$, where S contains each domino exactly once.*

Proof: We first show that there is a prefix TS of C where S contains each domino once. Let $d := |p_s|$, and let $C_m = C_{m+k}$ for some $k > 0$ be two occurrences of the same domino in C. We may assume that $C_{m+1}\cdots C_{m+k}$ is the longest suffix of $C_0\cdots C_{m+k}$ containing no domino twice, so $1 \leq k \leq d^2$. For $k < d^2$, we show that there is $k' > k$ and $m' \geq m$ such that $C_{m'+1}\cdots C_{m'+k'}$ is the longest suffix of $C_0\cdots C_{m'+k'}$ containing no domino twice.

Case a) C_{m+k+1} does not occur in $C_{m+1}\cdots C_{m+k}$. We can choose $m' := m$, $k' := k+1$. Case b) C_{m+k+1} occurs in $C_{m+1}\cdots C_{m+k}$, and has type i, say. If $C_{m+k+1} = C_{m+r}$ with $1 < r \leq k$, then by the *LZ*-rule all d dominoes of type i occur among $C_{m+r}\cdots C_{m+k}$, and since $C_m = C_{m+k}$, C_{m+1} has the same type as C_{m+k+1}, contradicting the assumption that no domino of type i occurs twice in $C_{m+1} \cdots C_{m+k}$. Hence $C_{m+k+1} = C_{m+1}$. But then $C_{m+2} \cdots C_{m+k+1}$ is the longest suffix of $C_0 \cdots C_{m+k+1}$ containing no domino twice, again of length $k < d^2$; by the *LZ*-rule this cannot occur k times in a sequence. It follows that one has to run into Case a), where $k < d^2$ is increased.

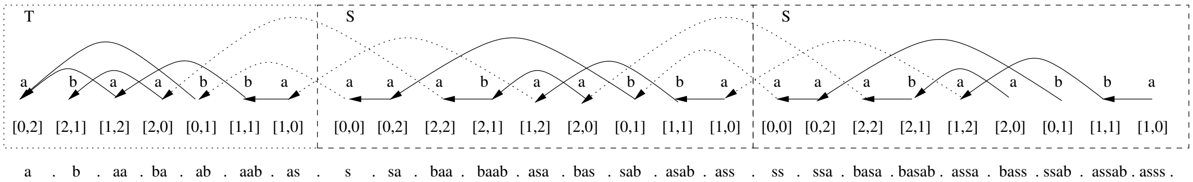
When we reach $k = d^2$, $S := C_{m+1} \cdots C_{m+k}$ (with updated m) contains each domino once. As argued for Case b), we have $C_{m+k+1} = C_{m+1}$, and similarly $C_{m+k+r} = C_{m+r}$ for all $r \geq 0$. With $T = C_0 \cdots C_{m-1}$, this gives $C = TS^\omega$. □

For a finite domino sequence C there is an obvious first-order formula $\varphi_C(x)$ such that for every ordered labeled graph $\mathcal{G}$,

$$\mathcal{G} \models \varphi_C(j) \iff \mathcal{G}\restriction_{\{j,\ldots,j+|C|-1\}} \simeq \mathcal{G}_C$$

where $\mathcal{G}\restriction_{\{j,\ldots,j+|C|-1\}}$ is the restriction of $\mathcal{G}$ to nodes $j, \ldots, j + |C| - 1$.

Example 4 *(Cont. 2) For $s = aba$ and $|T| = 7$, the initial segment $\mathcal{G}_{TSS}$ described by $\varphi_{TSS}(0)$ is*



By (4) and Lemma 5, $\mathcal{LZ}(s^\omega) = \mathcal{G}_C$ for the infinite $C = TS^\omega$. Expressing the unbounded repetition in $\mathcal{G}_C$ by $\forall z \geq |T|(\varphi_{SS}(z) \to \varphi_{SS}(z + |SS|))$, we find a first-order characterization $\varphi_C(0)$ of $\mathcal{LZ}(s^\omega)$:

Lemma 6 *For every $s \in \Sigma^+$, there is a first-order formula $\varphi_{TS^\omega}(x)$ such that for each ordered labeled graph $\mathcal{G} = (\omega, <, U_{a_1}, \ldots, U_{a_l}, E)$ of order type ω,*

$$\mathcal{G} \models \varphi_{TS^\omega}(0) \iff \mathcal{G} \simeq \mathcal{LZ}(s^\omega).$$

Next, we want to define the set of graphs $\mathcal{G}_{TSS^n}$ for the finite prefixes TSS^n of $C = TS^\omega$, so we have to express a *bounded* repetition. To do so, we need to extend the notion of an aperiodic string to the case of *LZ*-graphs. We call a labeled ordered graph of the form $\mathcal{G}_{SS}$ *periodic*, if $\mathcal{G}_{SSS}$ contains at least 3 segments isomorphic to $\mathcal{G}_{SS}$. (To account for the back-edges between copies of $\mathcal{G}_S$, we consider graphs of the form $\mathcal{G}_{SS}$.)

Proposition 2 *Let $s \in \Sigma^+$ be a power of some aperiodic string p_s. Let the LZ-block class sequence of $s^\omega = p_s^\omega$ be TS^ω where $|S| = |p_s|^2$. Then $\mathcal{G}_{SS}$ is aperiodic.*

Proof: Suppose $\mathcal{G}_{SS}$ is periodic, i.e. $\mathcal{G}_{SSS} \models \varphi_{SS}(k)$ for some $0 < k < |S|$. Then $\mathcal{G}_{S^\omega} \models \varphi_{S^\omega}(k)$, and $n \mapsto n + k$ defines an embedding of $\mathcal{G}_{S^\omega}$ to an isomorphic end segment. On $\mathcal{G}_{S^\omega}$, E only relates nodes whose class are of the same domino type $i < |p_s|$, so $E = \bigcup\{E_i \mid i < |p_s|\}$ is a union of graphs E_i with disjoint fields. Note that $n \mapsto n + k$ preserves E, so that dominoes C_n of the same type i are mapped to dominoes C_{n+k} of the same type $\pi(i)$, for some bijection $\pi : |p_s| \to |p_s|$. Hence, by the domino rule, if in $C = S^\omega$ we have $C_n = [l_n, r_n]$, then $C_{n+k} = [\pi(l_n), \pi(r_n)]$. Moreover, π preserves the relation $\succ$, so $\pi(i-1) = \pi(i) - 1 \pmod{|p_s|}$. Since π cannot be the identity, we have $\pi(i) = i + k' \pmod{|p_s|}$ for some $0 < k' < |p_s|$.

Since $n \mapsto n + k$ respects the labeling on $\mathcal{G}_{S^\omega}$, the labels at n and $n + k$ are the same; so the last letter of $u_{r_n} \leq_{\textit{prefix}} p_s$ equals the last letter of $u_{r_{n+k}} = u_{\pi(r_n)} = u_{r_n + k'}$. But this implies that p_s is periodic, contradicting the assumptions. □

Proposition 3 *Let $s \in \Sigma^+$ and $C = TS^\omega$ be a representation of the LZ-block class sequence of s^ω according to Lemma 5. There is a first-order formula $\varphi_{TSS^+}(x, y)$ such that for each finite ordered labeled graph $\mathcal{G}$,*

$$\mathcal{G} \models \varphi_{TSS^+}(0, max) \iff \mathcal{G} \simeq \mathcal{G}_{TSS^n} \text{ for some } n > 0.$$

Proof: We impose an upper bound on $\forall z$ in the formula $\varphi_{TS^\omega}(x)$ of Lemma 5, obtaining $\varphi_{TSS^+}(x, y) :=$

$$\begin{aligned} &\varphi_{TSS}(x) \ \wedge\ \forall z(x + |T| \leq z < y - |SS| \wedge \varphi_{SS}(z) \to \varphi_{SS}(z + |S|)) \\ &\qquad \wedge\ \forall z_1 \geq x + |TS|\, \forall z_2\, (E(z_1, z_2) \to z_1 - |S| \leq z_2 < z_1). \end{aligned}$$

For $\Leftarrow$, we must know that when $\mathcal{G}_{TSS^n} \models \varphi_{SS}(j)$ with $|T| \leq j < max - |SS|$, then $j \leq max - |SSS|$. In other words, we need that in $\mathcal{G}_{SSS}$, $\varphi_{SS}(z)$ only holds for nodes 0 and $|S|$, which follows from Proposition 2. (Note the analogy to (1) in Proposition 1.) □

We can now combine the arguments to prove the announced

Lemma 7 *If $s \in \Sigma^+$ is aperiodic, then s^+ is first-order definable on LZ-compressed strings.*

Proof: The language s^+ is a set of finite prefixes of s^ω. We distinguish the elements $w \in s^+$ according to the class of the *LZ*-block of s^ω in which w ends. Consider the sequence $B = B_0 B_1 \cdots$ of *LZ*-blocks of s^ω and its associated sequence $C = C_0 C_1 \cdots$ of block classes. Let $C = TS^\omega$ as in Lemma 5. We may assume that S begins with $[0, 0]$ and that $B_{|T|} = w_0 \in s^+$. Since s is aperiodic, $|S| = |s|^2$. Because subsequent blocks of the same class appear at a distance of $|S|$ blocks, and the second extends the first by $|s|$ subsequent letters of s^ω, we have:

$$\forall n, k\, \forall i < |S|\ (B_{|T|+i} = v_l s^k u_r \quad \Rightarrow \quad B_{|T|+i+n|S|} = v_l s^{k+n} u_r). \tag{5}$$

In particular, block $B_{|TS^n|}$ contains $w_0 s^n \in s^+$. Note that s^+ is the union of the following $|S| + 1$ sublanguages, where P ranges over non-empty prefixes of S:

a) $\{w \in s^+ \mid w \leq_{\textit{prefix}} B_0 \cdots B_{|TSS|}\}$,

b) $\{w \in s^+ \mid B_0 \cdots B_{|TSS^nP|-1} <_{\textit{prefix}}\ w \leq_{\textit{prefix}} B_0 \cdots B_{|TSS^nP|},\ n > 0\}$.

It remains to be shown that these are first-order definable on *LZ*-compressed strings.

Proof of a): This is clear since we are dealing with a finite language.

Proof of b): Note that if $B_0 \cdots B_{|TSS^nP|-1} <_{\textit{prefix}}\ s^m\ \leq_{\textit{prefix}} B_0 \cdots B_{|TSS^nP|}$, then $\mathcal{LZ}(s^m)$ generally is not an initial segment of $\mathcal{LZ}(s^\omega) = \mathcal{G}_{TS^\omega}$; it extends $\mathcal{G}_{TSS^nP}$, the *LZ*-graph of $B_0 \cdots B_{|TSS^nP|-1}$, by a maximal node that is labeled with the last letter of s and has $E(max, z)$ for a suitable node z of $\mathcal{G}_{TSS^nP}$. We have to express the possible values of z.

Let $P = Q[i, j]$ and let $[j, r]$ be the class of block $B_{|TSS^nP|}$. The blocks $B_{|TSS^nP|-1}$ and $B_{|TSS^nP|}$ contain $v_i s^k u_j$ and $v_j s^{k'} u_r$, for some $k > 0$, where S and P determine whether $k' = k - 1, k$ or $k + 1$, independently of k by (5). The final block of $LZ(s^m)$ has class $D = [j, 0]$ and contains $v_j s^l$ for some $l \leq k'$. It is related in $\mathcal{LZ}(s^m)$ by E to the block containing the longest strict prefix of $v_j s^l$, which is a block among the first $|T|$ blocks, or a later block containing $v_j s^{l-1} u_1$, which has class $[j, 1]$.

Let J_T be the numbers $< |T|$ of blocks containing the longest strict prefix of some $v_j s^l$, and $j_S < |S|$ resp. $j_P < |P|$ the position of domino $[j, 1]$ in S resp. P. Let

$$\psi_{T,S,P}(z) \quad := \quad z = max - |P| + j_P\ \vee \bigvee_{j_T \in J_T} z = j_T\ \vee\ \exists y(\varphi_S(y) \wedge z = y + j_S).$$

The sublanguage b) of s^+ is then defined in first-order on *LZ*-compressed strings by

$$\varphi'_{TSS^+}(0, max - |P|) \wedge \varphi_{SQ}(max - |SP| + 1) \\ \wedge \quad \exists z\,(\psi_{T,S,P}(z) \wedge \forall y(y = z \leftrightarrow E(max, y)) \wedge U_a(max),$$

where a is the last letter of s and $\varphi'_{TSS^+}(x, y)$ is a variation of formula $\varphi_{TSS^+}(x, y)$ of Proposition 3 which does not restrict the back-edge from node max. □

Example 4 *(Cont. 3) Take $s = aba$ and let T and S be the class sequences as in the figure on page 15. Notice that $B_{|T|} = s$ and $B_{|TSS^n|} = s^{n+2} \in s^+$. Consider the prefix $P = [0, 0][0, 2][2, 2][2, 1][1, 2]$ of S with length 5. For the language in b) of the proof, suppose*

$$B_0 \cdots B_{|TSS^nP|-1} <_{prefix} s^m \leq_{prefix} B_0 \cdots B_{|TSS^nP|}.$$

We determine the formula $\psi_{T,S,P}(z)$ that defines the positions z of $\mathcal{LZ}(s^m)$ with $E(max, z)$. Since P ends in $[i, j] = [1, 2]$, the final block B_{max} of $LZ(s^m)$ has class $[2, 0]$, and contains a non-empty prefix of $B_{|TSS^nP|} = v_2 s^{n+1} u_0 = bas^{n+2}$ ending in $v_2 = ba$ or s. Among $B_0, \ldots, B_6$ only $B_1 = b$ is the longest strict prefix of some $v_2 s^l$ with $l \leq n + 2$, so $J_T = \{1\}$. The predecessor domino $[2, 1]$ of $[2, 0]$ with equal type $j = 2$ occurs in P and S at position $j_P = j_S = 3$. Hence,

$$\psi_{T,S,P}(z) = [z = max - 2 \vee z = 1 \vee \exists y(\varphi_S(y) \wedge z = y + 3)].$$

In $\mathcal{LZ}(s^m)$, we have $E(max, max-2)$ if $B_{max} = bas^{n+2}$, $E(max, 1)$ if $B_{max} = ba$, and $E(max, y+3)$ where $\varphi_S(y)$ if $B_{max} = bas^l$ for $0 < l < n + 2$.

To define s^+ on strings, we needed s to be aperiodic because otherwise the induction formula Ind_s of (1) would not define s^+. To define s^+ on *LZ*-compressed strings, we need s to be aperiodic for a different reason. When s is periodic, say $s = p_s^k$, the *LZ*-graphs of s^ω and p_s^ω are the same, so we can define $LZ(p_s^+)$ as above. But to define $LZ(s^+) \subset LZ(p_s^+)$, in case b) of the proof we must not allow the back-edge from max to go to an *arbitrary* block of class $[j, 1]$. The necessary restriction cannot be expressed in first-order: Let $s = aa$ and suppose φ has quantifier rank k and defines s^+ on *LZ*-compressed strings. Choose w such that $LZ(w)$ has length $2m + 1 > 2^k$ and $E(max, m)$. Then $\mathcal{LZ}(w) \equiv_k \mathcal{LZ}(wa)$, because they only differ in that the latter has $E(max, m + 1)$. Hence both w and wa belong to $(aa)^+$, which is impossible. So s^+ is not first-order on *LZ*-compressed strings.

5 Conclusion

We have considered properties of strings that are definable on compressed strings, using the run-length and the Lempel-Ziv compression scheme. First-order queries on strings can be translated to first-order queries on run-length compressed strings, and to queries definable in first-order extended with a transitive closure on Lempel-Ziv compressed strings.

This approach generalizes the compressed pattern matching problem as we characterized a class $\mathcal{F}$ of first-order problems on strings which are also first-order definable on *LZ*-compressed strings. It is interesting to note that new compression schemes (as the antidictionary of [5]) are often designed to make search problems easier on compressed structures. It is hoped that the definability point of view will be helpful to design compression schemes that preserve some key properties.

If we extend our approach to binary structures, we would generalize the model-checking approach to new compression schemes as OBDD's may not be the only useful compressed or symbolic representation to verify a property. In the classical case, we verify a formula ψ by iterating the construction of binary decision diagrams following the syntax of the formula.

In our approach, we may very well define a new compression scheme C for which there is a *different* formula φ_C which defines the same property. We may end up with a smaller compressed (or symbolic) structure but with a formula in a richer language.

References

[1] T. C. Bell, J. G. Cleary, and I. H. Witten. *Text Compression*. Prentice Hall, Englewood Cliffs, NJ, 1990.

[2] R.E. Bryant Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, 35 (8) : 677-691, 1986.

[3] J. R. Büchi. On a decision method in restricted second order arithmetic. In E. Nagel, editor, *International Congress on Logic, Information and the Philosophy of Science*, volume 1. Stanford University Press, 1962.

[4] T. Cover and J. Thomas. *Elements of Information Theory*. John Wiley, 1991.

[5] M. Crochemore, F. Mignoni, A. Restive, and S. Salemi. Text compression using antidictionaries. In J. Wiedermann and M. Nielsen, editors, *26th International Colloquium on Automata, Languages and Programming, ICALP'99, Prague, July 11-15, 1999*, LNCS 1644, Berlin, 1999. Springer Verlag.

[6] S. de Agostino. P-complete problems in data compression. *Theoretical Computer Science*, 127:181–186, 1994.

[7] H.-D. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer-Verlag, 1991.

[8] C. Elgot. Decision problems of finite automata design and related arithmetics. *Transactions of the American Mathematical Society*, 98:21–52, 1961.

[9] M. Farach and J. Thorup. String matching in Lempel-Ziv compressed string. In *ACM Symposium on the Theory of Computing*, pages 703–712, 1995.

[10] F. Feigenbaum and S. Kannan and M. Vardi and M. Viswanathan. Complexity of Problems of Graphs represented as OBDDs. Symposium on Theoretical Aspects of Computer Science, 1998.

[11] H. Galperin and A. Wigderson. Succinct representations of graphs. *Information and Control*, 56:183–198, 1983.

[12] U. Manber. A text compression scheme that allows fast searching directly in the compressed file. In *Combinatorial Pattern Matching*, LNCS 807, pages 113–124, Berlin, 1994. Springer Verlag.

[13] R. McNaughton. *Elementary computability, formal languages, and automat*. Prentice-Hall, Englewood Cliffs, N.J., 1982.

[14] Y. Shibata, M. Takeda, A. Shinohara, and S. Arikawa. Pattern matching in text compression by using antidictionaries. In *10th Annual Symposium on Combinatorial Pattern Matching, CPM'99, Warwick, July 22-24, 1999*, LNCS 1645, Berlin, 1999. Springer Verlag.

[15] J. Ziv and A. Lempel. Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, pages 530–536, 1978.