Definability and Compression

Foto Afrati[1] and Hans Leiß[2] and Michel de Rougemont[3]

[1] University of Athens, Greece
[2] Universität München, CIS
D-80583 München, Germany
[3] Université Paris-II & LRI Bâtiment 490,
F-91405 Orsay Cedex, France

October 1999

Abstract. We study the first-order definability on compressed structures of properties of strings and images. Simple first-order properties of strings are presented which are not first-order definable on strings compressed with the Lempel-Ziv compression schema. Conversely, there are properties that are first-order definable on Lempel-Ziv compressed strings, but not on strings.

We show that all properties of strings that are first-order definable on strings are definable on Lempel-Ziv compressed strings in an extension of first-order logic by a transitive closure operator. We define a subclass $\mathcal{C}$ of the first-order properties of strings such that if L is defined by a property in $\mathcal{C}$, it is also first-order definable on the Lempel-Ziv compressed strings.

We also consider a naive compression schema where all first-order properties of strings are first-order definable on compressed strings, but where this fails for 2-dimensional strings (images).

1 Introduction

A classical search problem on strings asks for the existence of a given substring and recent papers study the same question on compressed strings. This problem can be approached from two different points of view : the search for specific algorithms that solve the problem or the search for specific expresssions in a given language that define the problem. The first efficient algorithms to search for a substring in a compressed string were given in [Far91]. We follow the second point of view and study the definability of various problems on classes of compressed finite structures.

For the substring problem, a binary string w of length n is a finite structure $\mathcal{S}(w)$ of size n with a unary predicate U and a linear order $<$ (the predicate $U(i)$ is true iff the i-th bit of w is 1). There exists a first-oder formula, i.e. a logical expression in the language $\{U, <\}$ using logical connectives and first-order quantifiers, which defines the substring problem, i.e. which is true on $\mathcal{S}(w)$ iff w contains s as a substring. Suppose w is given in a compressed form (as a .gz file for example, i.e. generated by the classical Lempel-Ziv compression schema). The compressed string $LZ(w)$ is another finite structure $\mathcal{LZ}(w)$ which we will precisely define. Given such a compressed string, we ask if there is another formula in the language of the compressed structure which defines the existence of a substring s in w. More generally given a property $\mathcal{P}$ on a class of finite structures and a compression schema, we ask if we can define $\mathcal{P}$ on the class of compressed structures.

It is an essential feature to search in compressed structures for texts but also for images and multimedia files. To find the relevant information without decompressing the files is a fundamental requirement for computations based on multimedia information. In case of the universal Lempel-Ziv compression schema, we will show that the substring problem is not first-order definable on the compressed structures.

It is not surprising that other compression schemas such as coding common pairs of letters in one byte [Man94] or omitting letters predictable using an antidictionary of the text [CMRS98] are beeing investigated. They allow a good compression and the possibility to search for substrings. In the case of video files, the definition of new compression formats (such as MPEG 7) is principally motivated by indexing and search, but there is no proposal (at this point) for a query language. We need a better understanding of the intricate links between compression and definability on general structures.

In this paper we take the simplest possible example of strings and binary images compressed by a naive or by the classical Lempel-Ziv [ZL77,ZL78] compression schema and ask the general question: given any first-order property on strings, can we define this property on the compressed structure? We will answer this question negatively and show that natural properties (like the existence of a particular substring or subimage) can not be defined by a first-order formula on compressed strings or images. Our main results are:

1. for the naive compression schema, 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 schema, 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 $\mathcal{C}$ of first-order formulas such that if a language is definable by a formula in $\mathcal{C}$, its compression is first-order definable on the compressed structure.
3. For both the naive and the Lempel-Ziv compression, the class of languages that are first-order definable as compressed strings is not closed under concatenation.

In section 2, we describe how finite structures represent strings and images, strings compressed by a naive compression schema and by Lempel-Ziv. In section 3, we study the definability on strings and images compressed by the naive compression schema. In section 4, we study the definability on strings compressed by the classical Lempel-Ziv compression schema.

2 Compression and finite structures

A *structure* over a finite domain D is a sequence $\mathcal{D} = (D, R_1, \ldots, R_l, f_1, \ldots, f_k)$, where each R_i is a relation over D and each f_j is a function from D into D. The sequence of arities of the R_i's and f_j's is the *type* of the structure. A *query* is a function from a class of structures (of some fixed type) to relations of fixed arity over the same domain which is invariant under isomorphism. The language of the structure is $\{R_1, \ldots, R_l, f_1, \ldots, f_k\}$.

2.1 Strings as finite structures

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

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

where $D_n = \{0, 1, \ldots, n-1\}$, $<$ 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. (Note that the logical axiom $\forall x\varphi \to \exists x\varphi$ is only valid in $\mathcal{S}(w)$ because the empty string w is excluded.) 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 such as $\varphi(x+1) := \exists y(\varphi(y) \wedge x < y \wedge \neg\exists z(x < z \wedge z < y))$. 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 or *formal language* $L \subseteq \Sigma^+$ of strings is *first-order definable*, if there is a first-order sentence φ in the language $\{<\} \cup \{U_a \mid a \in \Sigma\}$ such that $L = \{w \in \Sigma^+ \mid \mathcal{S}(w) \models \varphi\}$. L is *monadic-second-order definable*, if the same holds for a sentence φ which may also have quantifiers $\exists X$ and $\forall X$ ranging over subsets and atomic formulas $X(y)$ for membership in subsets. A language L is **-star free* (resp. *regular*), if it can be built from finite languages by union, elementwise concatenation, and also complement (resp. closure under concatenation).

The basic facts on definable properties of strings are the following classical results:

Theorem 1 *[Büc62,Elg61,MP71] Suppose $L \subseteq \Sigma^+$. Then*

1. *(McNaughton/Papert) L is star-free iff L is first-order definable.*
2. *(Büchi,Elgot) L is regular iff L is monadic-second-order definable.*

In particular, $L^+ := \{w_1 \cdots w_n \mid n \geq 1, w_1, \dots, w_n \in L\}$ need not be first-order definable when L is. Even for a single string s, $s^+ := \{s\}^+$ is not first-order definable in general: if we could define s^+ for $s = aa$, we could define linear orders of even length, which is impossible (cf.[EF91], Example 1.3.5). It follows that L^+ need not be first-order when L contains a periodic word: simply, take $L = \{aa\} \cup M$ where M is a language whose members do not contain the letter a. If L^+ were first-order definable by φ, then $\varphi \wedge \forall j\, U_a(j)$ would define $\{aa\}^+$, contradicting the undefinability of evenness.

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. The following fact about when L^+ is first-order is worth noting.

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^+ is first-order definable.
3. There are aperiodic words s_1, s_2 such that $\{s_1, s_2\}^+$ is not first-order definable.

Proof: 1.) If $s = p^k$ is periodic, then clearly s has at least three occurrences in ss. Conversely, if $ss = usv$ for some non-empty u, v, then $s = uv$ because $u \leq_{\mathit{prefix}} s, v \leq_{\mathit{suffix}} s$ and $|uv| = |s|$; hence $ss = (uv)(uv) = u(uv)v$, so $uv = vu$. If $|u| = |v|$, then we have $s = p^2$ for $p = u = v$. If $|u| < |v|$, say, then $v = uw$ for some $w \neq \epsilon$, and we get $s = vu = uwu = uv$, so $v = uw = wu$. By induction we get $u = p^k, w = p^l$ for some $k, l > 0$ and hence $s = p^m$ for some $m > 0$.

2.) Let $U_s(x)$ be the formula such that $U_s(j)$ is true in $\mathcal{S}(w)$ iff w has an occurrence of s beginning at position j. A first-order definition for s^+ is the following induction principle,

$$\mathit{Ind}_s := \; U_s(0) \;\wedge\; \forall j < max - |s|' \; (U_s(j) \to U_s(j + |s|)), \tag{1}$$

where $|s|'$ means $|s| - 1$. If a string $\mathcal{S}(w)$ satisfies Ind_s, then $w \in s^+$: let k be the largest number such that $w = s^{k+1}u$ for some u. Then $|s^{k+1}| \geq max - |s|'$. If $|u|$ is not 0, then $j = |s^k| < max - |s|'$ is a position with $U_s(j) \wedge \neg U_s(j + |s|)$, contradicting Ind_s. If s is aperiodic, then the only occurrences of s in $s^k \in s^+$ are at positions $j = |s^i|$, $i < k$, so s^k satisfies Ind_s. (If $s = p^k$ is periodic, then s^{m+2} does not satisfy Ind_s, because $U_s(x)$ holds at $max - |p|(k+1)$, but not $|s|$ positions to the right, at $max - |p|$.)

3.) Suppose φ is a first-order definition of $\{s_1, s_2\}^+$ with $s_1 = aba$ and $s_2 = b$. Then

$$\psi = \varphi \wedge U_a(0) \wedge U_b(max) \wedge \neg \exists j (U_{aa}(j) \vee U_{bb}(j))$$

defines $(s_1 s_2)^+ = \{abab\}^+$. Since $abab$ is periodic, we would get a first-order definition of evenness again. □

2.2 Naive compression of strings

The naive compression, or *N-compression*, of a string codes the number of repetitive symbols as an integer. Formally, it decomposes a string $w \in \Sigma^+$ into a sequence of subwords or blocks $B_i \in \Sigma^+$ such that $w = B_0 \cdots B_{m-1}$. The first block B_0 consists of the longest prefix of w whose letters are all the same. 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 N-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 naive block decomposition of $w = aaaaaabbbbaa$ is $aaaaaa.bbbb.aa$ and the naive compression of w is $N(w) = (6,a)(4,b)(2,a)$. A more familiar notation is $w = a^6 b^4 a^2$.

We represent naively compressed strings $N(w)$ as first-order structures

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

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

In the first-order language talking about $\mathcal{N}(w)$, variables must indicate if they range over D_n or D_m. For example, the formula $\exists j \in D_n \forall i \in D_m (f(i) = j)$ defines the set of compressed strings of the form $a^j b^j c^j b^j \cdots$.

Notice that the naive coding can compress with an exponential gain. The string a^n is compressed as (n, a) and requires $\log n$ bits. The LZ structure we present below requires $\sqrt{n}$ bits, but some more elaborate variations of Lempel-Ziv also achieve an exponential gain.

2.3 Lempel-Ziv compression of strings

The classical Lempel-Ziv compression is a family of algorithms with many variations. We follow a simple version presented in [CT91]. The compressed string is a sequence of pairs $p_n = (k, a)$ which represent subsequent blocks in the original string. The first component k is either 0 or points to a previous pair p_{k-1} that encodes the longest strict prefix B_{k-1} of the substring B_n encoded by p_n. The second component a is the letter of the alphabet which makes $B_n = B_{k-1} a$.

More formally, the LZ-compression decomposes a word $w \in \Sigma^+$ into a sequence of subwords 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 $decode((n+1, a)) = decode(p_n)a$.

We insert dots in a string to show its decomposition into LZ-blocks. For example, the string

$$w_1 \;=\; a.aa.aaa.aaaa.aaaaa$$

is encoded as $LZ(w_1) = (0, a)(1, a)(2, a)(3, a)(4, a)$. The string

$$w_2 \;=\; a.b.bb.aa.bba.bbab$$

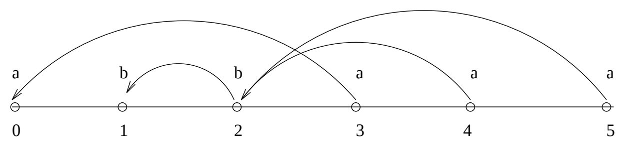
is encoded as $LZ(w_2) = (0, a)(0, b)(2, b)(1, a)(3, a)(5, b)$.

We represent LZ-compressed strings $LZ(w) = p_0 \cdots p_{m-1}$ as first-order structures

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

which are finite labelled ordered graphs. As in the naive 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 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 $w = a.b.bb.aa.bba.bba.$ with $LZ(w) = (0, a)(0, b)(2, b)(1, a)(3, a)(3, a)$, the graph $\mathcal{LZ}(w)$ is



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 the predicate U_0.

Notice that the graphs are very special : if the string w is over an alphabet with d letters, the in-degree of $\mathcal{LZ}(w)$ is at most $d + 1$, the out-degree at most 1, and $\mathcal{LZ}(w)$ has at most d roots.

2.4 2-dimensional images and their compressions

We consider images as 2-dimensional arrays on a finite domain Σ of colours. The 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. We only use the binary value domain $\Sigma = \{\circ, \bullet\}$, so that images are bitmaps.

Compressed images are obtained by compressing the strings that arise from the concatenation of all lines, i.e. $\chi(0,0)\cdots\chi(0,n-1)\ \cdots\ \chi(n-1,0)\cdots\chi(n-1,n-1)$. For Lempel-Ziv compression of a bitmap, we obtain a structure

$$\mathcal{LZ}(\mathcal{I}) = (D_m, <, U, E)$$

as before. For the naive compression of a bitmap, we obtain a structure

$$\mathcal{N}(\mathcal{I}) = (D_m, D_{n^2}, <_m, <_{n^2}, U, f).$$

In the case of a blank image of size $n \times n$, its naive compression is simply one block $(n^2, \circ)$, i.e. $\mathcal{N}(\mathcal{I})$ has $m = 1$, $\neg U(0)$ and $f(0) = n^2 - 1$.

2.5 Two example problems for definability under compression

Suppose we have a scheme C that maps strings w to compressed strings $C(w)$, and suppose we represent $C(w)$ by a first-order structure $\mathcal{C}(w)$, i.e. we fix how to talk about the compressed strings in first-order. We say $L \subseteq \Sigma^+$ *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 then also say φ *is a first-order definition of* $\mathcal{C}(L) := \{\mathcal{C}(w) \mid w \in L\}$, ignoring the infinite models of φ.) To express properties of strings we can use either the language of strings or the language of compressed strings.

We are interested in the question how the properties definable on strings are related to the properties definable on compressed strings. As an example, we look at

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 $\{(w,i) \mid \mathcal{S}(w) \models U_s(i)\}$.

We will show that the substring query is first-order definable on naively compressed strings, but not on *LZ*-compressed strings. In fact, all first-order properties on strings are first-order on N-compressed strings, and are definable on LZ-compressed strings in first-order extended by a transitive closure operator. In both cases, we also treat formulas with free variables, so we can answer with occurrence positions, which are represented by tuples of elements of the compressed string. For images, we consider a similar subimage query and show that already the subsquare query is not first-order on N-compressed images.

We are also interested in how the class of properties of strings that are definable on compressed strings behaves. As an example, we consider

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?

We show that the answer is negative both for N-compression and for LZ-compression; it is positive for N-compression when $\mathcal{N}$-structures are equipped with additon.

3 Definability on naively compressed strings and images

We compare the properties of strings that are first-order definable on strings with those that are first-order definable on naively compressed strings.

3.1 First-order on N-compressed strings need not be first-order on strings

Some properties of strings that are first-order on N-compressed strings are not first-order on strings.

Example 1 *The language $L = \{0^n 1^n \mid 0 < n\}$ is not regular, hence by Büchi's result it is not even monadic-second-order on strings. But it is first-order on N-compressed strings, because*

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

3.2 First-order on strings is first-order on N-compressed strings

First-order queries on strings can be expressed as first-order queries on naively compressed strings; in fact, this extends to first-order relations between positions in a string:

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

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

Here, h is the mapping from elements of $\mathcal{S}(w)$ to pairs of elements of $\mathcal{N}(w)$ such that $h(i) = (k, j)$ iff position i is the $j+1$-st relative position in block B_k, i.e. $i = \Sigma_{i'<k}(f(i')+1)+(j+1)$.

Proof: By induction on the formula $\varphi(x_1, \ldots, x_n)$, we define $\varphi^N(x_1, y_1, \ldots, x_n, y_n)$ and verify the property (2) for all $i_1, \ldots, i_n \in \mathcal{S}(w)$, writing $h(i_1) = (k_1, j_1), \ldots, h(i_n) = (k_n, j_n)$.

1. $(U_a(x_1))^N := U_a(x_1)$. If $\mathcal{S}(w) \models U_a(i_1)$, then position i_1 lies in block B_{k_1} and letter i_1 of w is a, so $N(w)(k_1) = (j, a)$ for some $j_1 \leq j \leq f(k_1)$ and hence $\mathcal{N}(w) \models U_a(k_1)$. The converse is similar.
2. $(x_1 = x_2)^N := (x_1 = x_2 \wedge y_1 = y_2)$. Claim (2) just means $i_1 = i_2 \iff h(i_1) = h(i_2)$, which is true since h is injective.
3. $(x_1 < x_2)^N := x_1 <_m x_2 \vee (x_1 = x_2 \wedge y_1 <_n y_2)$. If $\mathcal{S}(w) \models i_1 < i_2$, then either these positions are in different blocks, so $\mathcal{N}(w) \models k_1 <_m k_2$, or they are in the same block, so $\mathcal{N}(w) \models k_1 = k_2 \wedge j_1 <_n j_2$. The converse is also clear from the relation between i's, k's and j's given by h.
4. $(\neg\varphi)^N := \neg(\varphi^N)$ and $(\varphi_1 \wedge \varphi_2)^N := (\varphi_1^N \wedge \varphi_2^N)$. In these cases, (2) is clear by induction.
5. $(\exists x_{n+1} \varphi(x_1, \ldots, x_n))^N := \exists x_{n+1} \exists y_{n+1} [y_{n+1} \leq_n f(x_{n+1}) \wedge \varphi^N(x_1, y_1, \ldots, x_{n+1}, y_{n+1})]$. Suppose $\mathcal{S}(w) \models \varphi(i_1, \ldots, i_{n+1})$ for some $i_{n+1} \in \mathcal{S}(w)$. By induction, we have $\mathcal{N}(w) \models \varphi^N(k_1, j_1, \ldots, k_{n+1}, j_{n+1})$, and since j_{n+1} is a position in block k_{n+1}, we also have $j_{n+1} \leq f(k_{n+1})$. Conversely, if $\mathcal{N}(w) \models j_{n+1} \leq_n f(k_{n+1}) \wedge \varphi^N(k_1, j_1, \ldots, k_{n+1}, j_{n+1})$, then from the first conjunct we know that $(k_{n+1}, j_{n+1}) = h(i_{n+1})$ for some i_{n+1}, whence $\mathcal{S}(w) \models \varphi(i_1, \ldots, i_{n+1})$ follows by induction, so $\mathcal{S}(w) \models \exists x_{n+1} \varphi(i_1, \ldots, i_n)$.

This completes the proof. $\square$

Corollary 1 *The substring query is first-order definable on N-compressed strings.*

For example, the query $\exists x_1\, U_{aba}(x_1)$ can be stated on naively compressed strings by translating its complete form,

$$\exists x_1 \exists x_2 \exists x_3 [U_a(x_1) \wedge U_b(x_2) \wedge U_a(x_3) \wedge x_1 < x_2 < x_3 \wedge \neg \exists x_4 (x_1 < x_4 < x_2 \vee x_2 < x_4 < x_3)],$$

which gives a somewhat complicated formula because of the translation of the last conjunct.

Corollary 2 *Every *-free language L is first-order definable on N-compressed strings.*

Proof: By McNaughton's theorem, L is definable by a first-order sentence φ on strings, so its translation φ^N defines the naive compression of L. (A sentence φ with $L = \{w \mid \mathcal{N}(w) \models \varphi\}$ can effectively be determined from a *-free extended regular expression r for L.) $\square$

Theorem 2 implies that if L_1 and L_2 are first-order definable, then $\mathcal{N}(L_1 L_2)$ is first-order. We connot replace assuming that L_k is first-order by assuming that $\mathcal{N}(L_k)$ is first-order:

Proposition 2 *Language concatenation is not first-order on N-compressed strings.*

Proof: Both $L_1 = \{0^n 1^n \mid 0 < n\}$ and $L_2 = \{1^m 0^m \mid 0 < m\}$ are first-order on N-compressed strings, by Example 1. To define $L_1 L_2$ on N-compressed strings, we would need a first-order formula expressing

$$max = 2 \wedge \neg U(0) \wedge U(1) \wedge \neg U(2) \wedge f(1) = f(0) + f(2),$$

but we don't have addition on D_n. Suppose φ were a first-order sentence of quantifier rank k that defined $L_1 L_2$ on N-compressed strings. Pick $w_1 \in L_1$ and $w_2 \in L_2$ such that $|w_i| > 4^k$. One can show that $\mathcal{N}(w_1 w_2)$ and $\mathcal{N}(w_1 1 w_2)$ are k-elementarily equivalent, by defining a winning strategy for the Duplicator in the k-round Ehrenfeucht-Fraisse-game between these structures. Hence $\mathcal{N}(w_1 1 w_2) \models \varphi$, because $\mathcal{N}(w_1 w_2) \models \varphi$, contradicting $w_1 1 w_2 \notin L_1 L_2$. $\square$

However, if we allow addition on D_n of $\mathcal{N}$-structures, the class of languages that are first-order under N-compression is closed under concatenation:

Proposition 3 *Language concatenation is first-order definable on naively compressed strings, if we represent naively compressed strings $N(w)$ as first-order structures*

$$\mathcal{N}_+(w) = (D_m, D_n, <_m, U_{a_1}, \ldots, U_{a_l}, f, +, 0, 1)$$

where $+, 0, 1$ are the partial addition on D_n with zero and unit elements. That is, $\mathcal{N}_+(L_1 L_2)$ is the class of finite models of a first-order sentence, if $\mathcal{N}_+(L_1), \mathcal{N}_+(L_2)$ are.

Proof: Suppose that, for $k = 1, 2$, ψ_k is a first-order definition of $\mathcal{N}_+(L_k)$. Note that $\mathcal{N}_+(w) \in \mathcal{N}_+(L_1 L_2)$ iff there are words w_1, w_2 such that $w = w_1 w_2$ and $\mathcal{N}_+(w_k) \models \psi_k$. If the last letter of w_1 differs from the first of w_2, we obtain $\mathcal{N}_+(w_1)$ and $\mathcal{N}_+(w_2)$ by splitting $\mathcal{N}_+(w)$ between two points i and $i+1$. In this case, to say that the parts satisfy ψ_k we can use formulas $\psi_k[x, y]$, obtained from ψ_k by relativizing all quantified variables ranging over D_m to the interval between x and y.

If the last letter of w_1 is the same as the first of w_2, we obtain $\mathcal{N}_+(w_1)$ and $\mathcal{N}_+(w_2)$ by splitting a point $i \in \mathcal{N}_+(w)$ into two, i.e. by letting i be the last point of the first segment and also the first of the second segment, and adjusting the multiplicity $f(i)$ of the common symbol for the two segments (i.e. we split block B_i). To reinterpret f for the parts, we replace all atomic expressions $f(x) = y$ in $(\psi_1 \wedge f(max) = p)$ by the formula

$$\varphi_1(x, y, p) := (x < max \wedge f(x) = y) \vee (x = max \wedge y = p)$$

and in $(\psi_2 \wedge f(0) = q)$ by the formula

$$\varphi_2(x, y, q) := (x = 0 \wedge y = q) \vee (0 < x \wedge f(x) = y)$$

to obtain formulas $\psi_1^*(p)$ and $\psi_2^*(q)$, before relativizing to the segments. The formula defining $\mathcal{N}_+(L_1L_2)$ then is

$$\begin{aligned}\psi := \exists i \in D_m \; \{ \; &(\psi_1[0,i] \wedge \psi_2[i+1, max]) \;\vee \\ &\exists p, q \in D_n (f(i) = p + q + 1 \wedge \psi_1^*(p)[0,i] \wedge \psi_2^*(q)[i, max])\}.\end{aligned}$$

Note that the meaning of the pseudo-constants $0, max$ in ψ_k^* changes under relativisation. □

3.3 First-order on images need not be first-order on N-compressed images

For images, we now take a somewhat complex example which illustrates how the existence of simple patterns can become very difficult even with simple compression schemas.

The subsquare query The (monochrome) subsquare query asks for the existence of a square of four $\bullet$ symbols, i.e. the existence of two consecutive lines of the image where the pattern $\bullet\bullet$ occurs at the same horizontal position. We can define this query in first-order 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.

The goal of this section is to show:

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

Before we go into technicalities we shall give an outline of our proof of Theorem 3. Suppose the subsquare query were first-order definable on compressed images by a sentence φ_{sq}^N of quantifier rank k. We will construct two images, $\mathcal{I}_1$ without the square pattern, $\mathcal{I}_2$ with the square pattern in the first two lines (and identical from the third line on) such that their compressions $\mathcal{N}(\mathcal{I}_1)$ and $\mathcal{N}(\mathcal{I}_2)$ are k-elementarily equivalent. This contradicts the assumption that $\mathcal{N}(\mathcal{I}_2)$ satisfies φ_{sq}^N and $\mathcal{N}(\mathcal{I}_1)$ does not.

We will prove that $\mathcal{N}(\mathcal{I}_1)$ and $\mathcal{N}(\mathcal{I}_2)$ are k-elementarily equivalent by showing that the Duplicator will win an Ehrenfeucht-Fraisse game of k rounds on $\mathcal{N}(\mathcal{I}_1)$ and $\mathcal{N}(\mathcal{I}_2)$. To do so, we first construct from $\mathcal{N}(\mathcal{I}_1)$ and $\mathcal{N}(\mathcal{I}_2)$ two simpler structures $\mathcal{E}_1$ and $\mathcal{E}_2$, show that the Duplicator has a winning strategy on these and then transfer the winning strategy to $\mathcal{N}(\mathcal{I}_1)$ and $\mathcal{N}(\mathcal{I}_2)$. To construct $\mathcal{E}_1$ and $\mathcal{E}_2$, we consider $\mathcal{N}(\mathcal{I}_1)$ and $\mathcal{N}(\mathcal{I}_2)$ as sequences of regions (intervals), each corresponding to a substring of the first or second line of the images $\mathcal{I}_1$ and

$\mathcal{I}_2$; in addition, regions representing substrings in the first and second line of the images $\mathcal{I}_1$ and $\mathcal{I}_2$ are related. The structures $\mathcal{E}_1$ and $\mathcal{E}_2$ represent these relations as an ordering of tupels. Thus, the $\mathcal{E}$-structures allow to talk about the first two compressed lines of the images simultaneously, which makes it easier to explain the Duplicator's strategy.

Now for the formal proof of Theorem 3. We need to define the k-round Ehrenfeucht-Fraisse game only for the subclass of $\mathcal{N}$-structures where $D_n = D_4$, because the longest blocks in the example images are of length 4.[1] We call these structures $\mathcal{N} = (D_m, D_4, U, f, <_m, <_4)$ the *restricted* $\mathcal{N}$-structures:

Definition 1 *The k-round Ehrenfeucht-Fraisse game is played between two players, the* Spoiler *and the* Duplicator*, on two restricted $\mathcal{N}$-structures $\mathcal{N}$ and $\mathcal{N}'$. At round r, the spoiler picks a point p_r in one of the domains of one of the structures; the duplicator responds by picking a point q_r in the corresponding domain of the other structure. For each r, let t_r, t'_r be the points associated with $\mathcal{N}$, $\mathcal{N}'$ respectively; so $\{t_r, t'_r\} = \{p_r, q_r\}$. The duplicator wins the game if for all $i, j \leq k$ and all $a \in \Sigma$:*

$$t_i = t_j \text{ iff } t'_i = t'_j,\ f(t_i) = f(t'_i),\ t_i < t_j \text{ iff } t'_i < t'_j, \text{ and } U_a(t_i) \text{ iff } U_a(t'_i).$$

The well-known theory of Ehrenfeucht-Fraisse games (cf. [EF91]) gives the following result:

Theorem 4 *Let Q be a property of restricted $\mathcal{N}$-structures. The following are equivalent:*

(a) Q is not expressible in first-order logic for restricted $\mathcal{N}$-structures.
(b) For each k, there exist restricted $\mathcal{N}$-structures $\mathcal{N}$, $\mathcal{N}'$ which differ with respect to Q such that the spoiler wins the k-round Ehrenfeucht-Fraisse game on $\mathcal{N}$, $\mathcal{N}'$.

The example images We will now construct the two image structures $\mathcal{I}_1$ and $\mathcal{I}_2$. Each image is a square 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:

$$\begin{array}{ll}
\mathcal{I}_1 : & \bullet(\circ\bullet\circ\bullet)^{i_1}(\circ\circ\bullet\bullet)^{i_2}\ \circ\circ\circ\circ\bullet\bullet\circ\bullet\bullet\circ\bullet\bullet(\circ\circ\bullet\bullet)^{j_1}(\circ\circ\bullet\bullet)^{j_2}\circ\circ \\
& \bullet\bullet(\circ\circ\bullet\bullet)^{i_1}(\circ\circ\bullet\bullet)^{i_2}\circ\circ\bullet\bullet\circ\circ\circ\bullet\bullet\circ\bullet\bullet(\circ\circ\bullet\bullet)^{j_1}(\circ\circ\bullet\bullet)^{j_2}\circ \\
& \\
\mathcal{I}_2 : & \bullet(\circ\bullet\circ\bullet)^{i_1}(\circ\circ\bullet\bullet)^{i_2}\ \circ\circ\circ\circ\bullet\bullet\circ\underline{\bullet\bullet}\ \circ\bullet\bullet(\circ\circ\bullet\bullet)^{j_1}(\circ\circ\bullet\bullet)^{j_2}\circ\circ \\
& \bullet\bullet(\circ\circ\bullet\bullet)^{i_1}(\circ\circ\bullet\bullet)^{i_2}\circ\circ\bullet\bullet(\circ\circ\bullet\bullet)^{j_1}\circ\circ\circ\bullet\bullet\circ\bullet\ \bullet(\circ\circ\bullet\bullet)^{j_2}\circ
\end{array}$$

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 $0 < j_1$.

Using the abbreviations $a = (1, \circ), b = (2, \circ), d = (1, \bullet), c = (3, \bullet), e = (3, \circ), f = (4, \circ)$ and $i = i_1 + i_2, j = j_1 + j_2$, the naive compressions are

$$\mathcal{N}(\mathcal{I}_1) : \quad d(ad)^{2i_1}(bc)^{i_2} fcacac(bc)^j b.c(bc)^i bcecac(bc)^j a$$

$$\mathcal{N}(\mathcal{I}_2) : \quad d(ad)^{2i_1}(bc)^{i_2} fcacac(bc)^j b.c(bc)^{i+j_1} bcecac(bc)^{j_2} a$$

where, to improve readability, we inserted the . to mark the end of the first line.

[1] Therefore, the following also works when using $\mathcal{N}_+(w)$ rather than $\mathcal{N}(w)$ to represent $N(w)$, because addition on D_4 can be replaced by $<_4$.

In both structures, the first line consists of a segment of *dada . . . dad* followed by the pattern *bcbc . . . bcb* with a string of *fcacac* somewhere in the middle. The second line consist of the pattern *cbcbc . . . bca* interrupted (at different places in the two structures) by a string *bcecac*.

In order for Duplicator to win a game with k rounds, we will later set the integers i_1, i_2, j_1, j_2 large compared to k. Notice that the neighborhoods on an original image are 2-dimensional as we have an order $<_1$ on the first coordinate and an order $<_2$ on the second. On the naively compressed image, the neighborhoods are 1-dimensional. In order to cope simultaneously with the first two compressed lines, we first work on modifications of $\mathcal{N}(\mathcal{I}_1)$ and $\mathcal{N}(\mathcal{I}_2)$ where regions in the first two lines are coupled.

A simpler form of the compressed images Consider words over the alphabet $\Sigma = \{A, \ldots, G\}$, represented as labelled ordered structures

$$\mathcal{E} = (D_m, <, U_A, \ldots, U_G).$$

We can write the compressed images $\mathcal{N}(\mathcal{I}_1)$ and $\mathcal{N}(\mathcal{I}_2)$ as words $\mathcal{E}_1$ and $\mathcal{E}_2$ over $\{A, \ldots, G\}$, using these letters as the codes

$$A = \begin{array}{|c|}\hline \mathrm{d} \\ \hline \mathrm{c} \\ \hline \end{array} \quad B = \begin{array}{|c|}\hline \mathrm{ad} \\ \hline \mathrm{bc} \\ \hline \end{array} \quad C = \begin{array}{|c|}\hline \mathrm{bc} \\ \hline \mathrm{bc} \\ \hline \end{array} \quad D = \begin{array}{|c|}\hline \mathrm{fcacac} \\ \hline \mathrm{bcbcbc} \\ \hline \end{array} \quad E = \begin{array}{|c|}\hline \mathrm{bcbc} \\ \hline \mathrm{ecac} \\ \hline \end{array} \quad F = \begin{array}{|c|}\hline \mathrm{bc} \\ \hline \\ \hline \end{array} \quad G = \begin{array}{|c|}\hline \mathrm{b} \\ \hline \mathrm{a} \\ \hline \end{array};$$

each of these letters represents two regions of blocks in a compressed image, the first region having the upper component line of the letter as labels, the second the lower component line. Reading the concatenation of these letters componentwise, we can view the first compressed structure $\mathcal{N}(\mathcal{I}_1)$ as :

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

and the second compressed structure $\mathcal{N}(\mathcal{I}_2)$ as :

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

where i_1, i_2, j_2, j_2 are constrained so that all exponents are nonnegative. The only difference between $\mathcal{E}_1$ and $\mathcal{E}_2$ is that letter E occurs in another position. For example, $\mathcal{N}(\mathcal{I}_1)$ with $i_1 = 4, i_2 = 5, j_1 = 1, j_2 = 4$ can be written as $\mathcal{E}_1 = AB^8C^2ECDCF^4G$, or

d	ad	ad	ad	ad	ad	ad	ad	ad	bc	bc	bcbc	bc	fcacac	bc	bc	bc	bc	bc	b
c	bc	bc	bc	bc	bc	bc	bc	bc	bc	bc	ecac	bc	bcbcbc	bc					a

.

In fact, we can a little bit further simplify the way we view $\mathcal{E}_1$ and $\mathcal{E}_2$ by letting $p = i_1 - 3$, $q = i_2 - i_1 + 1$ and $m = j + j_2 - i_1$; then we can re-write the two structures to:

$$\mathcal{E}_1 : AB^{2(p+3)}C^qEC^pDC^mF^{p+3}G,$$
$$\mathcal{E}_2 : AB^{2(p+3)}C^{j+q}EC^{p-j}DC^mF^{p+3}G$$

where the constraints now are: q is even and $j << p << q << m$.

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

Proof: The Duplicator has a classical winning strategy on labelled orderings: he always plays points with same label as the last point played by the Spoiler, satisfying the same ordering relations; moreover, if the Spoiler in round r plays at a distance $k-r$ from a border point or a point played earlier, the Duplication plays at the same distance from the corresponding border point; otherwise, he plays at a distance $> 2^{k-r}$ from all border and already played points. This is possible for k rounds if the numbers $i_1, ..., j_2$ are large enough. □

We can now refine the Duplicator's winning strategy on $\mathcal{E}_1$ and $\mathcal{E}_2$ to obtain:

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

Proof: Note that the word $\mathcal{E}_1$ decomposes the structure $\mathcal{N}(\mathcal{I}_1)$ into $|\mathcal{E}_i|$ pairs of regions: for example, point r of $\mathcal{E}_1$, labelled by $C = (bc, bc)$, fixes two occurrences of bc in the sequence $N(\mathcal{I}_1) \in \{a, \ldots, f\}^+$, hence two regions of size 2 in the structure $\mathcal{N}(\mathcal{I}_1)$. Conversely, each point r of the structure $\mathcal{N}(\mathcal{I}_1)$ corresponds to a point of $\mathcal{E}_1$, namely the one which contains r in one of the regions it defines on $\mathcal{N}(\mathcal{I}_1)$. The same holds for $\mathcal{E}_2$ and $\mathcal{N}(\mathcal{I}_2)$.

Therefore, the Duplicator can use the following strategy: If the last point played by the Spoiler belongs to the second domain D_4, the Duplicator just plays the same point in the other structure.

If the last point played by the Spoiler belongs to D_m, the Duplicator first translates the points played so far on the D_m-domains of structures $\mathcal{N}(\mathcal{I}_1)$ and $\mathcal{N}(\mathcal{I}_2)$ into the corresponding points on the structures $\mathcal{E}_1$ and $\mathcal{E}_2$. He then plays on the $\mathcal{E}$-structures according to his winning strategy of the proof of Lemma 1, translates the point of D_m he has played into a pair of regions in the $\mathcal{N}(\mathcal{I})$-structures, and then plays a point in one of the regions depending on the Spoiler's last move: if, for example, the spoiler played a point r_1 of $\mathcal{N}(\mathcal{I}_1)$ labelled b, whose corresponding point e_1 in $\mathcal{E}_1$ is labelled C, and if r_1 is the first point of the second region defined by e_1, then the point e_2 of $\mathcal{E}_2$ provided by the winning strategy is also labelled C, so the Duplicator answers with r_2, the first point of the second region of $\mathcal{N}(\mathcal{I}_2)$ defined by e_2.

It remains to be checked that at the end of the game, a partial isomorphism between $\mathcal{N}(\mathcal{I}_1)$ and $\mathcal{N}(\mathcal{I}_2)$ is obtained. The corresponding points on the $\mathcal{E}$-structures give a partial isomorphism between $\mathcal{E}_1$ and $\mathcal{E}_2$, by the strategy used for $\mathcal{E}$-structures. The strategy used on the $\mathcal{N}$-structures refines this to a partial isomorphism between $\mathcal{N}(\mathcal{I}_1)$ and $\mathcal{N}(\mathcal{I}_2)$ as far as labelling and ordering are concerned, and for the points of the domains D_4, the correspondence is the identity. □

By Lemma 2 and Lemma 1, $\mathcal{N}(\mathcal{I}_1)$ and $\mathcal{N}(\mathcal{I}_2)$ are k-elementarily equivalent. Since one of $\mathcal{I}_1, \mathcal{I}_2$ contains the subsquare but the other does not, the subsquare query cannot be defined on the compressed images by a first-order sentence of quantifier depth k. This completes the proof of Theorem 3.

4 Definability on Lempel-Ziv compressed strings

We can consider strings w as labelled finite orders $\mathcal{S}(w)$ or, when compressed using the Lempel-Ziv compression, as labelled finite ordered graphs $\mathcal{LZ}(w)$. For each version, we have a first-order language by which we can express properties of strings as definable classes of structures $\mathcal{S}$, or $\mathcal{G}$, respectively. What is the relation between the properties definable in the two first-order languages?

4.1 First-order on LZ-compressed strings need not be first-order on strings

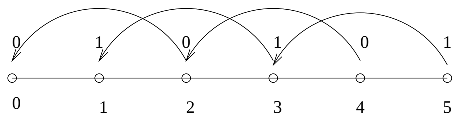
First-order sentences on the class of compressed (binary) strings, i.e. sentences in the language $\{U, E, <\}$, define properties of compressed strings $\mathcal{LZ}(w)$ and hence properties of strings w. Are they definable in the language of strings, i.e. with only $\{U, <\}$? The following lemma answers this question negatively.

Lemma 3 *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

$$\varphi := \neg U(0) \wedge U(1) \wedge \forall i \geq 2\,(U(i) \leftrightarrow U(i-2)) \;\wedge\; \forall i \forall j (E(i,j) \leftrightarrow (i \geq 2 \wedge j = i-2))$$

A graph of size 6 satisfying φ is



Note that

$$\mathcal{LZ}(w) \models \varphi \iff w = 0.1.00.11.000.111.\ \ldots\ .0^p.1^p \text{ for some } p.$$

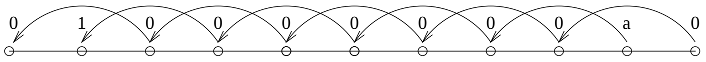
The set of these strings w is not a regular language. By the result of Büchi, it can not be defined by a first-order formula in $\{<, U\}$, not even by a monadic second-order formula. □

4.2 First-order on strings need not be first-order on LZ-compressed strings

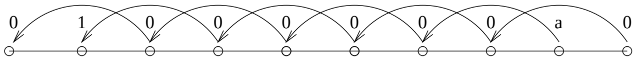
We now turn to the reverse question. First-order sentences in the language of strings, i.e. with the unary predicates $\{U_a \mid a \in \Sigma\}$ and the order predicate $<$, define properties of strings. We will show that, in general, they can not be expressed by sentences on LZ-compressed strings.

In particular, we show that the substring query is not first-order on LZ-compressed strings. Thus, finding a substring is 'difficult' to do on LZ-compressed structures. The main reason is that under LZ-compression of a string wsv, the compressed structure $\mathcal{LZ}(w)$ will directly influence the compression of s in the string wsv. The LZ algorithm always looks at previous prefixes and therefore the code of a substring occurrence depends on its left context.

Example 2 *Consider the alphabet $\Sigma = \{0, 1, a\}$ and the language $L = \{0,1\}^*\ a\ 0^*$. We call $LZ(L)$ the corresponding set of compressed strings. Let $LZ(w_1)$ and $LZ(w_2)$ be the two compressed strings represented by the graphs below.*



The compressed string $\mathcal{LZ}(w_1)$ of length 11



The compressed string $\mathcal{LZ}(w_2)$ of length 10

Notice that the parities of the graphs differ. By decompression, the corresponding strings are

$$w_1 \ : \ 0.1.00.10.000.100.0000.1000.00000.1000a.000000 \ \in L$$

and

$$w_2 \ : \ 0.1.00.10.000.100.0000.1000.0000a.10000 \ \notin L.$$

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

Proof: Consider the language L of example 2 which is first-order definable. Let us show that the corresponding set $\mathcal{LZ}(L)$ of compressed strings is not first-order definable. Suppose $\mathcal{LZ}(L)$ were first-order definable by a formula 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 $\mathcal{LZ}(w_1) \equiv_k \mathcal{LZ}(w_2)$. In a Ehrenfeucht-Fraisse game of rank k, the Duplicator has a winning strategy : at stage $i \leq k$, he keeps the intervals of size 2^{k-i} around the minimum, the maximum and the points played isomorphic.

As $w_1 \in L, w_2 \notin L$ and compression is injective, we have $LZ(w_1) \in LZ(L)$ and $LZ(w_2) \notin LZ(L)$. Because $\mathcal{LZ}(w_1) \models \psi_k$ and $\mathcal{LZ}(w_1) \equiv_k \mathcal{LZ}(w_2)$, the graph $\mathcal{LZ}(w_2)$ should also satisfy ψ_k, a contradiction. □

Corollary 3 *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. Hence no first-order formula of quantifier rank k in the language of the compressed structures can define the substring query. □

Corollary 4 *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_1L_2 is not. □

4.3 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 an extension of the first-order formulas of LZ-structures. A formula is in the language $FO(TC)$ if it can be built when we add the following syntactic construction to those for first-order formulas : given a binary formula $\psi(x,y)$ the expression $TC.\psi(x,y)$ is also a formula, with free variables x, y. The meaning of $TC.\psi(x,y)$ is the transitive closure of the graph defined by $\psi(x,y)$.

Let us see how TC 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*

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

In the first case, we check the symbols 0 at the end of block 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 i, but to check whether 0 is the first *symbol of block $i+1$ requires to follow the back-edges (using TC) from node $i+1$ all the way to find the block j containing the prefix of block $i+1$ with length 1.*

The example suggests a general way to recover a position in the original string from a pair of nodes in the LZ-compressed string. Namely, 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 $\{(j,k) \mid \mathcal{LZ}(w) \models E^*(k,j)\}$, where E^* is the transitive reflexive closure of E. Thus, if $E^*(k,j)$, we can view (k,j) as the end position of block j as a prefix of block k.

For example, if $w = a.aa.ab.aba.aa$, the positions 5,6,7 contained in block 3 are represented by $(3,0),(3,2),(3,3)$, because the nonempty prefixes of B_3 are $a = B_0$, $ab = B_2$, and $aba = B_3$. The last position of the last block $B_4 = aa = B_1$ is represented by $(4,4)$, not by $(4,1)$.

Theorem 5 *For every first-order formula $\varphi(x_1,\ldots,x_n)$ on strings, there is a FO(TC)-formula $\varphi^{LZ}(x_1,y_1,\ldots,x_n,y_n)$ on LZ-graphs, such that for each $\mathcal{S}(w)$ and all $i_1,\ldots,i_n \in \mathcal{S}(w)$,*

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

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 $x = y \vee TC.E(x,y)$. Since E is a deterministic relation, the translation in fact will go to $FO(DTC) \subset FO(TC)$.

Proof: By induction on the formula $\varphi(x_1,\ldots,x_n)$.

1. If $\varphi(x) = U_a(x)$, let $\varphi^{LZ}(x,y)$ be $U_a(y)$: 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 then the label at node j of $\mathcal{LZ}(w)$, so $\mathcal{LZ}(w) \models U_a(j)$. Similarly for the converse.
2. If $\varphi(x_1,x_2) = x_1 < x_2$, let $\varphi^{LZ}(x_1,y_1,x_2,y_2)$ be $(x_1 < x_2) \vee (x_1 = x_2 \wedge y_1 < y_2)$. If $i_1 < i_2$, then either i_1 belong to an earlier block than i_2 or they belong to the same block, but the relative position of i_1 is smaller than that of i_2.
   Conversely, assume $\mathcal{LZ}(w) \models \varphi^{LZ}(h(i_1),h(i_2))$ where $h(i_1) = (k_1,j_1)$ and $h(i_2) = (k_2,j_2)$. Since blocks do not overlap, $k_1 < k_2$ gives $i_1 < i_2$. If $k_1 = k_2$ and $j_1 < j_2$, then since the relative positions refer to the same block, the absolute positions satisfy $i_1 < i_2$.[2]
3. Clearly, $(\varphi_1 \vee \varphi_2)^{LZ}$ is $(\varphi_1^{LZ} \vee \varphi_2^{LZ})$ and $(\neg\varphi)^{LZ}$ is $\neg(\varphi^{LZ})$.
4. If $\varphi(x_1,\ldots,x_n) = \exists x_{n+1}.\psi$, let φ^{LZ} be $\exists x_{n+1}\exists y_{n+1}.(\psi^{LZ} \wedge E^*(x_{n+1},y_{n+1}))$. If $\mathcal{S}(w) \models \varphi(i_1,\ldots,i_n)$, there is $i_{n+1} \in \mathcal{S}(w)$ such that, by induction, $\mathcal{LZ}(w) \models \psi^{LZ}(h(i_1),\ldots,h(i_{n+1}))$. Let $h(i_{n+1}) = (k,j)$. Then i_{n+1} is a position in block k, and by definition of h, $E^*(k,j)$. So $\mathcal{LZ}(w) \models \varphi^{LZ}(h(i_1),\ldots,h(i_n))$.
   Conversely, assume $\mathcal{LZ}(w) \models \varphi^{LZ}(h(i_1),\ldots,h(i_n))$ and pick $k,j \in \mathcal{LZ}(w)$ such that

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

   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, $\mathcal{S}(w) \models \psi(i_n,\ldots,i_{n+1})$. Hence $\mathcal{S}(w) \models \exists x_{n+1}\psi(i_1,\ldots,i_n)$.

[2] We might take $E^*(x_i,y_2) \wedge E^+(y_2,y_1)$ instead of $y_1 < y_2$, which also might be more efficient. But by definition of h and since $E^+(j_2,j_1) \Rightarrow j_1 < j_2$, this is not necessary.

(By B_j we sometimes mean just the word, not its occurrence fixed by the LZ-algorithm.) $\square$

Since the substring query for an arbitrary string s is definable on strings by $\exists x\, U_s(x)$, the previous example generalizes to:

Corollary 5 *The substring query is $FO(TC)$ definable on LZ-compressed strings.*

An m-ary relation $R \subseteq D_n^m$ on $\mathcal{S}(w)$ is recovered in the compressed structure $\mathcal{LZ}(w)$ from the $2m$-ary relation $h(R)$, where h is the above mapping from $\mathcal{S}(w)$ to $\mathcal{LZ}(w) \times \mathcal{LZ}(w)$. Hence we can also use LZ-compression on strings with some additional relations R. For example, we can compress labelled graphs if we first impose a total linear order and then use LZ-compression on the resulting string with the edge relation.

If we extend the above translation to second-order formulas using

$$(\exists X^{(m)}.\psi)^{LZ} := \exists X^{(2m)}.\psi^{LZ} \quad \text{and} \quad X^{(m)}(x_1,\ldots,x_m)^{LZ} := X^{(2m)}(x_1,y_1,\ldots,x_m,y_m),$$

we still have

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

for existential second-order formulas $\varphi(x_1,\ldots,x_n)$ on strings. The converse fails since there exist $2m$-ary relations on $\mathcal{LZ}(w)$ that are not the f-image of m-ary relations on $\mathcal{S}(w)$.

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

The class of languages that are first-order definable on compressed strings contains the finite languages and is of course closed under the boolean operations of intersection, union and complement. By Corollary 4, it is not closed under concatenation. We will below present a class of first-order properties of strings which remain first-order on the compressed structures. This class is closed under a limited use of iterated concatenation.

Recall that for definability on strings, s^+ is first-order for aperiodic strings s, by Proposition 1. Likewise, a limited use of $^+$ can be first-order on LZ-compressed strings:

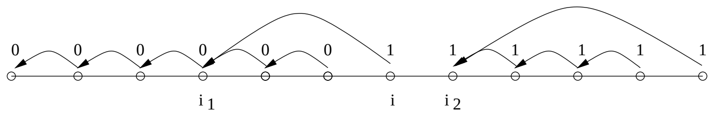
Example 4 *The regular language $L = 00^+11^+ = \{0^m\ 1^p \mid m,p \geq 2\}$ is defined by*

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

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 can be defined by

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

With considerable effort, we will now show:

Lemma 5 *If $s \in \Sigma^+$ is aperiodic, then s^+ is first-order definable on LZ-compressed strings.*

The main observation is that for $w \in s^+$ large enough, the ordered labelled 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 5, 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.

The LZ-graph of the infinite string s^ω as a domino sequence. Let $s^\omega = sss\cdots$ be the infinite word 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 $B_0, B_1, \ldots, B_{m-1}$. Then (3) gives an *infinite* LZ-graph,

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

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 words t. Note that s^ω has only a finite *suffix-set*, namely

$$\{vs^\omega \mid v <_{\mathit{suffix}} p_s\}, \qquad \text{where } p_s \text{ is the shortest } p \leq_{\mathit{prefix}} s \text{ such that } p^k = s \text{ for some } k.$$

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

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[3] $l, r < |p_s|$ 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, we have $B_j = v_l s^k u_r$ for some k.

Example 5 *Consider the aperiodic string $s = aba$. The LZ-block decomposition of s^ω gives a sequence of blocks beginning as follows:*

$$a\ .\ b\ .\ aa\ .\ ba\ .\ ab\ .\ aab\ .\ as\ .\ s\ .\ sa\ .\ baa\ .\ baab\ .\ asa\ .\ bas\ .\ sab\ .\ asab\ .\ as^2.\ \cdots$$

The classes of these blocks are

$$[0,2][2,1][1,2][2,0][0,1]\,[1,1]\,[1,0][0,0][0,2]\ \ [2,2]\ \ [2,1]\ \ [1,2]\ \ [2,0]\ \ [0,1]\ \ [1,1]\ \ [1,0]\cdots\ .$$

For example, 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 $1 = |a|$.

[3] But the *word* B_j may have different occurrences in p_s (or concatenations of a suffix with a prefix of p_s, as aa in $abaaca$), which in turn define different suffix-pairs of s^ω. For example, with $s = abaaba$, the subword a of s^ω has occurrences of class $[0,2]$ and occurrences of class $[1,0]$.

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

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

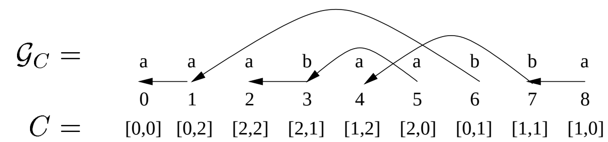
of dominoes $[l_j,r_j]$ over p_s is called a *LZ-domino sequence*, if it satisfies two conditions:

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 labelled ordered graph $\mathcal{G}_C = (D_{|C|},<,U_{a_1},\ldots,U_{a_l},E)$, where $U_a(i)$ iff $av_{r_i} \leq_{\mathit{suffix}} p_s$ 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.

For example, if $p_s = aba$ and $C = [0,0][0,2][2,2][2,1][1,2][2,0][0,1][1,1][1,0]$, we obtain



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

$$C \;=\; C_0\,C_1\,C_2\cdots$$

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.

Proposition 4 *Let C be an infinite LZ-domino sequence. If S is a subword of C containing each domino once, then $C = TS^\omega$, where TS is any prefix of C ending in S.*

Proof: Let d^2 be the number of dominoes, so $|S| = d^2$. Let C_m be the first domino in S and $C_{m'} := C_{m+d^2}$ the domino following S in C. It is sufficient to show $C_{m'} = C_m$, because then we likewise consider $S' = C_{m+1}\cdots C_{m'}$. Let $C_{m'} = [l',r']$. Since each domino of type l' occurs once in S, these occur ordered like $[l',r'],\ldots,[l',r]$ where $r \succ r'$. Suppose $C_m \neq [l',r']$. Then there are $d+1$ dominoes of type l' in $SC_{m'}$, and their left neighbours are $d+1$ dominoes with second component l', lying in S. But this is impossible, since there are only d different dominoes with second component l', and S contains no domino twice. So $C_{m'} = C_m$. $\square$

Lemma 6 *For each infinite LZ-domino sequence C there are finite words S,T such that $C = TS^\omega$, where S contains each domino once.*

Proof: Let d^2 be the number of dominoes. Since C is infinite, at least one domino is used infinitely often. Pick one of these and let $C_m = C_{m+k}$, $m < m+k$, be two of its occurrences; we may assume that $S := C_{m+1}\cdots C_{m+k}$ is the longest suffix of $C_0\cdots C_{m+k}$ containing no domino twice, whence $1 \leq k \leq d^2$. If $k = d^2$, the claim holds with $T := C_0 \cdots C_m$ by Proposition 4. So assume $k < d^2$. We show that there is $m' \geq m$ and $k' > k$ such that $S := C_{m'+1} \cdots C_{m'+k'}$ is the longest suffix of $C_0 \cdots C_{m'+k'}$ containing no domino twice.

If C_{m+k+1} does not occur in $C_{m+1} \cdots C_{m+k}$, put $m' = m, k' = k + 1$. If C_{m+k+1} occurs in $C_{m+1} \cdots C_{m+k}$ *and* $C_{m+k+1} = C_{m+1}$, 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.

So suppose C_{m+k+1} occurs in $C_{m+1} \cdots C_{m+k}$. In order to prove $C_{m+k+1} = C_{m+1}$, we show that a maximal sequence of pairwise different dominoes that obey the domino- and LZ-rules can only be continued by its first element. Consider the case $d = 3$, then the domino components are $i, i-1, i-2 \pmod 3$. Up to cyclic shifts, the maximal sequences of dominoes without repetitions are:

$$
\begin{array}{l}
[i,i][i,i-1][i-1,i][i,i-2][i-2,i] \\
[i,i][i,i-1][i-1,i][i,i-2][i-2,i-1][i-1,i-1][i-1,i-2][i-2,i-2][i-2,i] \\
[i,i][i,i-1][i-1,i][i,i-2][i-2,i-2][i-2,i] \\
[i,i][i,i-1][i-1,i-1][i-1,i-2][i-2,i][i,i-2][i-2,i-1][i-1,i] \\
[i,i][i,i-1][i-1,i-1][i-1,i-2][i-2,i-1][i-1,i][i,i-2][i-2,i-2][i-2,i] \\
[i,i][i,i-1][i-1,i-1][i-1,i-2][i-2,i-2][i-2,i][i,i-2][i-2,i-1][i-1,i] \\
[i,i][i,i-1][i-1,i-2][i-2,i][i,i-2][i-2,i-1][i-1,i] \\
[i,i][i,i-1][i-1,i-2][i-2,i-1][i-1,i][i,i-2][i-2,i-2][i-2,i] \\
[i,i][i,i-1][i-1,i-2][i-2,i-2][i-2,i][i,i-2][i-2,i-1][i-1,i-1][i-1,i].
\end{array}
$$

By inspection, the only domino that extends any of these according to the domino- and LZ-rules is $[i,i]$, the first one of each sequence. The same applies for d different from 3. It follows that one has to run into the first case, where $k < d^2$ is increased. □

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

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

where $\mathcal{G}\restriction_{\{j,\ldots,j+|C|-1\}}$ is the restriction of $\mathcal{G}$ to nodes $j, \ldots, j + |C| - 1$. This extends to the graphs of order type ω we are interested in:

Lemma 7 *For every $s \in \Sigma^+$, there is a first-order formula $\varphi_{TS^\omega}(x)$ such that for each ordered labelled graph $\mathcal{G} = (\omega, <, U_{a_1}, \ldots, U_{a_l}, E)$ of order type ω,*

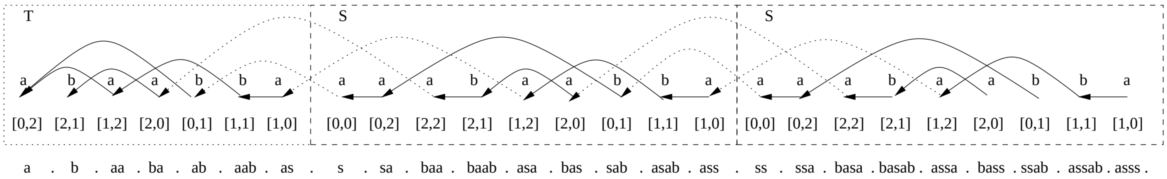
$$\mathcal{G} \models \varphi_{TS^\omega}(0) \iff \mathcal{G} \simeq \mathcal{LZ}(s^\omega).$$

Proof: For the infinite LZ-graph of s^ω, we clearly have $\mathcal{LZ}(s^\omega) = \mathcal{G}_{TS^\omega}$, using the block class sequence $C = TS^\omega$ of s^ω in its representation by finite T, S according to Lemma 6. To define $\mathcal{G}_{TS^\omega}$ in first order, we can take

$$
\begin{aligned}
\varphi_{TS^\omega}(x) \;:=\; & \varphi_{TSS}(x) \wedge \forall z \geq x + |T| \, (\varphi_{SS}(z) \rightarrow \varphi_{SS}(z + |S|)) \\
& \wedge \forall y \geq x + |TS| \, \forall z \, (E(y,z) \rightarrow y - |S| \leq z < y).
\end{aligned}
$$

It is clear that $\varphi_{TSS}(x)$ implies $\varphi_{SS}(x + |T|)$. Note that $\mathcal{G} \models \varphi_{SS}(j)$ says that at nodes j and $j + |S|$ there are copies of $\mathcal{G}_S$ which are appropriately connected by backedges (dotted in the following picture). The third conjunct excludes edges in $\mathcal{G}$ connecting nodes $> |T|$ of a distance $\geq |S|$. □

Example 4 *(Cont.) For $s = aba$ as above, when picking $|T| = 7$ the initial segment $\mathcal{G}_{TSS}$ described by $\varphi_{TSS}(0)$ looks as follows:*



The 9 blocks corresponding to the $k-th$ occurrence of S in C are, by induction on k:

$$s^k \ . \ s^k a \ . \ bas^{k-1}a \ . \ bas^{k-1}ab \ . \ as^k a \ . \ bas^k \ . \ s^k ab \ . \ as^k ab \ . \ as^{k+1}. \tag{4}$$

We now return to the finite. Since s^+ is a set of finite prefixes of s^ω, our next goal is a first-order description of a suitable set of initial segments of $\mathcal{LZ}(s^\omega) \simeq \mathcal{G}_{TS^\omega}$, the graphs $\mathcal{G}_{TS^n}$ for finite n. (Roughly, this is a first-order definition of the language TS^+ of domino-words.)

Corollary 6 *Let $s \in \Sigma^+$ and $C = TS^\omega$ be a representation of the LZ-block class sequence of s^ω according to Lemma 6. There is a first-order formula $\varphi_{TSS^+}(x, y)$ such that for each finite ordered labelled graph $\mathcal{G}$,*

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

Proof: To say that the segment from node x up to node y of an ordered labelled graph $\mathcal{G}$ is isomorphic to $\mathcal{G}_{TSS^n}$ for some $n > 1$, we impose an upper bound on $\forall z$ in the formula $\varphi_{TS^\omega}(x)$ of Lemma 6, obtaining

$$\begin{aligned} \varphi_{TSS^+}(x, y) := \varphi_{TSS}(x) \wedge \forall z(x + |T| \le z < y - |SS| \wedge \varphi_{SS}(z) \to \varphi_{SS}(z + |S|)) \\ \wedge \forall z_1 \ge x + |TS| \, \forall z_2 \, (E(z_1, z_2) \to z_1 - |S| \le z_2 < z_1). \end{aligned} \tag{5}$$

Then if $\mathcal{G} \models \varphi_{TSS^+}(0, max)$ is finite, clearly $\mathcal{G} \simeq \mathcal{G}_{TSS^n}$ for some $n > 0$. However, in order to see that $\mathcal{G}_{TSS^n} \models \varphi_{TSS^+}(0, max)$, we must know that when $\mathcal{G}_{TSS^n} \models \varphi_{SS}(j)$ with $|T| \le j < max - |SS|$, then $j \le max - |SSS|$. In other words, we must ensure that in $\mathcal{G}_{SSS}$ the formula $\varphi_{SS}(x)$ only holds for nodes 0 and $|S|$. This is done in Proposition 5 below. (Cf. the proof of Proposition 1, claim 2., for strings.) $\square$

We need to extend the notion of an aperiodic word to the case of *LZ*-graphs. To account for the backedges between copies of $\mathcal{G}_S$, the definition is restricted to graphs of the form $\mathcal{G}_{SS}$. We call a labelled ordered graph of the form $\mathcal{G}_{SS}$ *periodic*, if $\mathcal{G}_{SSS}$ contains at least 3 segments isomorphic to $\mathcal{G}_{SS}$.

Proposition 5 *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. The minimal such k divides $|S|$. 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 labelling on $\mathcal{G}_{S^\omega}$, the labels at n and $n+k$ are the same; so the last letter of $u_{r_n} \leq_{\mathit{prefix}} p_s$ 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. □

Finally, it is convenient that we can find blocks containing given subwords of s^ω:

Proposition 6 *Let w be a subword of s^ω with $|s| \leq |w| < \omega$. There is j with $B_j = w$.*

Proof: Let B_j be the LZ-blocks of s^ω and C_j the class of B_j. Since $s^\omega = p_s^\omega$, we may assume that s is aperiodic. Pick l, m, r such that $w = v_l s^m u_r$. We show that

$$B_{j+1}B_{j+2}\cdots = v_l s^\omega \qquad \text{for infinitely many } j. \tag{6}$$

Then if $j+1$ is large enough, $w = v_l s^m u_r$ is a prefix of B_{j+1}, and since the set of blocks is closed under prefixes, w is the content of a block. To show (6), note that there is at least one class $[l', r']$ such that $C_j = [l', r']$ for infinitely many j. So for each $k \in \mathbb{N}$, there are $k' > k$ and j' with $B_{j'} = v_{l'} s^{k'} u_{r'}$. Since the set of blocks is closed under prefixes, $v_{l'} s^k u_l = B_j$ for some j. Since s is aperiodic, $B_{j+1}\cdots = v_l s^\omega$. As j increases with k, (6) is shown. □

We can now combine the arguments to prove the announced

Lemma 5 *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. Roughly speaking, for each of the subsets of s^+ obtained this way we give a first-order definition of its LZ-compression.

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 6. We may assume that S begins with $[0,0]$ and that $B_{|T|} = w_0 \in s^+$, by Proposition 6. 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:

$$\forall n, k\, \forall i < |S|\, (B_{|T|+i} = v_l s^k u_r \;\Rightarrow\; B_{|T|+i+n|S|} = v_l s^{k+n} u_r). \tag{7}$$

In particular, block $B_{|TS^n|}$ contains $w_0 s^n$. Note that s^+ is the union of the following $2|S|+1$ sublanguages, where P ranges over non-empty suffixes of S:

a) $\{w \in s^+ \mid w \leq_{\mathit{prefix}} B_0 \cdots B_{|TSS|}\}$,
b) $\{w \in s^+ \mid w = B_0 \cdots B_{|TSS^nP|-1} \text{ for some } n > 0\}$,
c) $\{w \in s^+ \mid B_0 \cdots B_{|TSS^nP|-1} <_{\mathit{prefix}} w <_{\mathit{prefix}} B_0 \cdots B_{|TSS^nP|} \text{ for some } n > 0\}$).

It remains to be shown that each of these is first-order definable on LZ-compressed strings.

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

Proof of b): If the last domino of P is not of the form $[l, 0]$ for some $l < |s|$, the set is empty. Otherwise, $\mathcal{G}_{TSS^nP}$ is the LZ-graph of $B_0 \cdots B_{|TSS^nP|-1} \in s^+$, for each $n > 0$. These graphs are the finite models of

$$\varphi_{TSS^+}(0, max - |P|) \wedge \varphi_{SP}(max - |SP| + 1), \tag{8}$$

where $\varphi_{TSS^+}(x, y)$ is the formula of Corollary 6 and $\varphi_{SP}(max - |SP| + 1)$ forces an end segment isomorphic to $\mathcal{G}_{SP}$.

Proof of c): Note that if

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

then $\mathcal{LZ}(s^m)$ 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 labelled 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 (7). 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

$$\psi_{T,S,P}(z) := \bigvee_{j_T \in J_T} z = j_T \ \vee \ \exists y(\varphi_S(y) \wedge z = y + j_S) \ \vee \ z = max - |P| + 1 + j_P,$$

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

$$\begin{aligned} &\varphi'_{TSS^+}(0, max - |P|) \ \wedge \ \varphi_{SQ}(max - |SP| + 1) \ \wedge \\ &\exists z \, (\psi_{T,S,P}(z) \ \wedge \ \forall y(y = z \leftrightarrow E(max, y)) \ \wedge \ U_a(max)), \end{aligned} \tag{9}$$

where a is the last letter of s and $\varphi'_{TSS^+}(x, y)$ is formula (5) except that $\forall z_1$ does not range over max, to allow that z may be far away from max. $\square$

To define s^+ on strings, we needed s to be aperiodic because otherwise the induction formula Ind_s 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 that we can represent its block class sequence C as TS^ω, where $|S| = |p_s|^2$, and define $LZ(p_s^+)$ as above. To define $LZ(s^+)$, we have to be more restrictive; for example, in case c) we must not allow the back-edge from max to go to an *arbitrary* block of class $[j, 1]$.

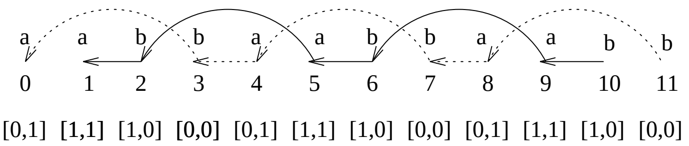
Example 5 *Consider the periodic word $s = abab$. The infinite string s^ω has only two suffixes, $v_0 = s^\omega$ and $v_1 = bs^\omega$. We obtain the following sequence of blocks and block classes for s^ω:*

B	a	b	ab	aba	b a	bab	abab	abab a	bab a	bab ab	ab abab	abab aba	b abab a	bab abab	⋯
C	[0,1]	[1,0]	[0,0]	[0,1]	[1,1]	[1,0]	[0,0]	[0,1]	[1,1]	[1,0]	[0,0]	[0,1]	[1,1]	[1,0]	⋯
n	0	1	2	3	4	5	6	7	8	9	10	11	12	13	⋯

We have $C = TS^\omega$ and $|S| = |p_s|^2 = |ab|^2 = 4$ with four different choices for S, for example

$$T.S = \ [0,1][1,0][0,0].[0,1][1,1][1,0][0,0] \ .$$

Indeed, $\mathcal{G}_{SS}$ is an aperiodic graph, since $\mathcal{G}_{SSS}$, i.e.



does not contain three copies of $\mathcal{G}_{SS}$: the formula $\varphi_{SS}(x)$ holds at nodes 0 and 4 only; because of the labels, it does not hold at node 2. In order to define $LZ(s^+)$, we would need, for example, the graphs $\mathcal{G}_{TS^n}$ extended by a final element max with label b and a back-pointer to the m-th node of class $[0,1]$ from the end, where m is odd (resp. even) if n is even (resp. odd). These distinctions cannot be made in first order.

More simply, let s be the periodic word aa. 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 $E(max, m)$. Note that $\mathcal{LZ}(w)$ and $\mathcal{LZ}(wa)$ only differ in that the latter has $E(max, m+1)$. Clearly, the duplicator has a winning strategy for the k-round Ehrenfeucht-Fraisse game between these structures, so they are k-elementarily equivalent. But then both w and wa belong to $(aa)^+$, which is impossible. Hence, s^+ is not first-order on LZ-compressed strings.

Definition 2 *Let $\mathcal{C}$ 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, $\mathcal{C}$ is a class of first-order definable languages.

Theorem 6 *If $L \in \mathcal{C}$, then L is first-order definable on LZ-compressed strings.*

Proof: by induction on the definition of L. The LZ-compression of a finite language is definable since $\mathcal{LZ}(w)$ is definable for each word $w \in \Sigma^+$. For union, intersection and complement we obtain straightforward formulas using the logical connectives.

Let u be a word and s be aperiodic. To show that $u \cdot s^+$ is first-order on LZ-compressed strings, we modify the proof of Lemma 5 and the previous reasoning. The representation of the LZ-graph $LZ(us^\omega)$ by an LZ-domino sequence TS^ω has to be slightly changed by using additional dominoes describing blocks that intersect with u. These are part of T and influence the repeating S, but do not affect the argumentation otherwise. We leave it to the reader to check the details. $\square$

Example 4, defining $LZ(00^+11^+)$, is not explicitly handled by Theorem 6. However, since the two iterations 0^+ and 1^+ operate on words with disjoint alphabets, the example can be reduced to the case of Theorem 6.

5 Conclusion

We have considered properties of strings that are definable on strings or on compressed strings, using a naive and the Lempel-Ziv compression schema. First-order queries on strings can be translated to first-order queries on naively compressed strings, and to queries definable in first-order extended with transitive closure on Lempel-Ziv compressed strings. Since we treat formulas, the translations cover the case of reporting occurrence positions of matches found.

We showed that the subsquare query is not first-order on naively compressed images, and the substring query is not first-order on Lempel-Ziv compressed strings. Moreover, the class of properties that are first-order on the compressed strings is not closed under concatenation and contains properties that are non-regular, hence not monadic second-order on strings.

Similar questions as we have treated can be asked for other compression schemes. For example, in the compression by omitting letters that can be predicted using an antidictionary [CMRS98] of the string, the compressed word and the antidictionary are separate structures, and it is not clear how to decompose properties of the string into properties of the compressed string and properties of the antidictionary.

In many applications, for example in computational biology, it is not sufficient to view strings as labelled orders as we have done here, but one needs to talk about distances between patterns in a string. For such cases, strings would better be represented by structures $\mathcal{S}_+(w) = (D_m, +, U_{a_1}, \ldots, U_{a_l})$, where the order $<$ is replaced by the (partial) addition on D_m. It would be an important extension of our investigation to relate first-order properties of strings with addition with those that can be defined on the corresponding compressed strings.

A further direction for future research is the study of definability and compression for arbitrary finite structures and more general logics.

References

[Büc62] 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.

[CMRS98] M. Crochemore, F. Mignoni, A. Restive, and S. Salemi. Text compression using antidictionaries. Technical report, Université de Marne-la-Vallée, Institut Gaspard-Monge, 1998. `www-igm.univ-mlv.fr/~mac/RAC/DCA.html`.

[CT91] T. Cover and J. Thomas. *Elements of Information Theory*. John Wiley, 1991.

[EF91] H.-D. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer-Verlag, 1991.

[Elg61] C.C. Elgot. Decision problems of finite automata design and related arithmetics. *Transactions of the American Mathematical Society*, 98:21–52, 1961.

[Far91] M. Farach. String matching in Lempel-Ziv compressed strings. In *ACM Symposium on the Theory of Computing*, 1991.

[Man94] U. Manber. A text compression scheme that allows fast searching directly in the compressed file. In *Combinatorial Pattern Matching*, volume 807 of *Lecture Notes in Computer Science*, pages 113–124. Springer Verlag, 1994.

[MP71] R. McNaughton and S. Papert. *Counter-Free Automata*. MIT-Press, Cambridge, Mass., 1971.

[ZL77] J. Ziv and A. Lempel. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*, pages 337–343, 1977.

[ZL78] J. Ziv and A. Lempel. Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*, pages 530–536, 1978.