Journal of Automated Reasoning (2023) 67:25
https://doi.org/10.1007/s10817-023-09674-2

Binary Codes that do not Preserve Primitivity

Štěpán Holub[1] · Martin Raška[1] · Štěpán Starosta[2]

Received: 31 January 2023 / Accepted: 12 June 2023 / Published online: 18 July 2023

Abstract
A set of words X is not primitivity-preserving if there is a primitive list of length at least two of elements from X whose concatenation is imprimitive. Here, a word or list is primitive if it is not equal to a concatenation of several copies of a shorter word or list, and imprimitive otherwise. We formalize a full characterization of such two-element sets $\{x, y\}$ in the proof assistant Isabelle/HOL. The formalization is based on an existing proof which we analyze and simplify using some innovative ideas. Part of the formalization, interesting on its own, is a description of the ways in which the square xx can appear inside a concatenation of words x and y if $|y| \leq |x|$. We also provide a formalized parametric solution of the related equation $x^j y^k = z^\ell$.

Keywords Binary code · Primitivity-preserving · Square interpretation

1 Introduction

Consider two words `abba` and `b`. It is possible to concatenate (several copies of) them as `b` · `abba` · `b`, and obtain a power of a third word, namely a square `bab` · `bab` of `bab`. In this paper, we describe a formalization in Isabelle/HOL of a full characterization of all ways how this can happen for two words. The question is part of the difficult problem of solving word equations. The simplest word equation is the commutation relation $xy = yx$, whose solutions are characterized by the existence of a word t and non-negative integers m and n such that $x = t^m$ and $y = t^n$. In fact, this characterizes solutions of any non-trivial equation in two variables. While the question of preserving primitivity is trivial for commuting words,

✉ Štěpán Holub
holub@karlin.mff.cuni.cz

Martin Raška
raska@karlin.mff.cuni.cz

Štěpán Starosta
stepan.starosta@fit.cvut.cz

1 Department of Algebra, Faculty of Mathematics and Physics, Charles University, Ke Karlovu 2027/3, 12116 Prague, Czech Republic

2 Department of Applied Mathematics, Faculty of Information Technology, Czech Technical University in Prague, Thákurova 9, 16000 Prague, Czech Republic

it becomes significantly more complicated in the case of non-commuting pairs, that is, in the case of *binary codes*, which explains the title of this article.

The corresponding theory has a long history. The question can be formulated as solving equations in three variables of the special form $W(x, y) = z^\ell$ where the left hand side is a product of x's and y's, and $\ell \geq 2$. The seminal result in this direction is the paper by Lyndon and Schützenberger [20] from 1962, which solves the equation $x^j y^k = z^\ell$, with $2 \leq j, k, \ell$, in a more general setting of free groups. It was followed, in 1967, by a partial answer to our question by Lentin and Schützenberger [18]. Solving an equation is equivalent to finding a monoid whose generators satisfy the relation given by the equation. A complete characterization of monoids generated by three words was provided by Budkina and Markov[4]. The characterization was later, in 1976, reproved in a different way by Lentin's student Spehner in his Ph.D. thesis [25], which even explicitly mentions the answer to the central question of this paper. See also a comparison of the two classifications by Harju and Nowotka [9]. In 1985, the result was again reproved by Barbin-Le Rest and Le Rest [1], this time specifically focusing on our question. Their paper contains a characterization of binary interpretations of a square as a crucial tool (see Sect. 4). The latter combinatorial result is interesting on its own, but is very little-known. In addition to the fact that, as far as we know, the proof is not available in English, it has to be reconstructed from Théorème 2.1 and Lemme 3.1 [1], it is long, technical and little structured, with many steps that require further clarification. It is symptomatic, for example, that Maňuch [21] cites the claim as essentially equivalent to his desired result but nevertheless provides a different, shorter but similarly technical proof. We hope that a formalization of the result will make it more convincing and approachable for the researchers and hence more widely known and used.

The proof we present here naturally contains some ideas of the original proof [1] but is significantly different. Our main objective was to follow the basic methodological requirement of a good formalization, namely to identify claims that are needed in the proof and formulate them as separate lemmas and as generally as possible so that they can be reused not only in the proof but also later. The resulting overall proof structure is new. In particular, we used the idea of "gluing" words which in itself is not new but it is not used in the original proof at all. The immediate consequence is that for our proof it is enough to consider the interpretation of the square xx, while the original proof must consider several other configurations. The analysis of the proof is therefore another important contribution of our formalization, besides the mere certainty that there are no gaps in the proof. We also provide a complete parametric solution of the equation $x^j y^k = z^\ell$ for arbitrary j, k and ℓ, a classification which is not very difficult, but maybe too complicated to be useful in a mere unverified paper form. We are not aware of a previous publication of this parametric solution, although it could be reconstructed from existing characterizations of monoids generated by three words mentioned above.

The formalization presented here is an organic part of a larger project of formalization of combinatorics of words (see an introductory description [14] or the project repository [11]). The presented version is archived [12]. The existence of the underlying library, which in turn extends the theories of `List` and `HOL-Library.Sublist` from the standard Isabelle distribution, critically contributes to a smooth formalization which is getting fairly close to the way a human paper proof would look like, outsourcing technicalities to the (reusable) background. We accompany claims in this text with names of their formalized counterparts.

Outline In Sect. 2 we introduce basic tools of combinatorics on words used in the paper. The section can be therefore understood as a brief tutorial of combinatorics on words. In Sect. 3 we introduce the main theorem of the paper, and outline its proof. The proof is then completed in Sects. 4–7; see the end of Sect. 3 for more details. Section 5 is dedicated to a special case of uniform binary codes, which is interesting on its own, and is later used for conjugate words. Section 8 provides some further details on the formalization of the result in Isabelle/HOL.

This is an extended version of a conference paper with the same name [16]. The introductory Sect. 2 is an extension of the appendix of the conference paper. We also added to the presentation many proofs omitted in the conference version. In particular, the substantial part of the proof of the key Theorem 25 is given, including proofs of several auxiliary lemmas. Numerous examples and figures were added. Section 3 was extended by Lemma 10 containing a new complementary result. Sect. 5 about uniform codes is a new one. In Sect. 7, we have added a passage about the non-overlapping set, which replaces and generalizes the brief remark about `sings-code` in the conference paper.

2 Notation and Basics of Combinatorics on Words

In this section we introduce basic notation, and explain most elementary facts, ideas and intuitions that we are going to use in the paper. For proofs and more details see [5, 19, 24].

2.1 Lists, Words and Monoids

Let Σ be a set, usually finite or countable, also called an *alphabet*.

Lists (i.e. finite sequences) $[x_1, x_2, \dots, x_n]$ of elements $x_i \in \Sigma$ are called *words* over Σ. The length of a word $u = [x_1, x_2, \dots, x_n]$ is denoted $|u|$ and equals n. The set of all words over Σ is usually denoted as Σ^*, using the Kleene star. A notorious ambiguity of this notation is related to the situation when we consider a set of words $X \subset \Sigma^*$, and are interested in lists over X. They should be denoted as elements of X^*. However, X^* usually means something else (in the theory of rational languages), namely the set of all words in Σ^* generated by the set X. To avoid the confusion, we will therefore follow the notation used in the formalization in Isabelle, and write `lists` X instead, to make clear that the entries of an element of `lists` X are themselves words. Note that Σ^* is equivalent to `lists` Σ, but we shall keep the former notation for the basic alphabet Σ. In order to further help to distinguish words over the basic alphabet from lists over a set of words, we shall use boldface variables for the latter. In particular, it is important to keep in mind the difference between a letter a and the word $[a]$ of length one, a distinction which is usually glossed over lightly in the literature on combinatorics on words. The set of words over Σ generated by X is then denoted as $\langle X \rangle$.

The (associative) binary operation of concatenation of two words u and v is denoted by $u \cdot v$. We prefer this algebraic notation to the Isabelle's original infix symbol `@`. Moreover, we shall sometimes omit the dot as usual. With respect to this operation, the set Σ^* is a monoid (with the empty word as the neutral element), and $\langle X \rangle$ is a submonoid of Σ^* for any $X \subseteq \Sigma^*$.

If $\mathbf{u} = [x_1, x_2, \dots, x_n] \in$ `lists` X is a list of words, then we write `concat` $\mathbf{u}$ for $x_1 \cdot x_2 \cdots x_n$. If $\mathbf{u}$ is nonempty, its first element x_1 is denoted `hd` $\mathbf{u}$ (head) and its last element x_n is denoted `last` $\mathbf{u}$. We write ε for the empty list (the empty word), and u^k for the concatenation of k copies of u (we use $u^{@}k$ in the formalization). We write $u \leq_p v$, $u <_p v$, $u \leq_s v$, $u <_s v$, and $u \leq_f v$ to say that u is a *prefix*, a *strict prefix*, a *suffix*, a *strict suffix* and a *factor* (that is, a contiguous sublist) of v respectively. The longest common prefix of u and v is denoted by $u \wedge_p v$. If u is a prefix of v or v is a prefix of u, we say that u and v are *prefix-comparable*.

2.1.1 Primitivity

A word is *primitive* if it is nonempty and not a power of a shorter word. Otherwise, we call it *imprimitive*. Each nonempty word w is a power of a unique primitive word $\rho(w)$, its *primitive root*. For example, $u = abab$ is imprimitive with $\rho(u) = ab$.

2.1.2 Periodic Root and Period

A word r is a *periodic root* of a word w if $w <_p r \cdot w$ (note that r must be nonempty). This is equivalent to w being a prefix of a sufficiently large power of r, and we shall sometimes write r^ω as an abbreviation of "sufficiently large power" (which remains just a notation, we do not deal with infinite words in this paper, nor in the formalization). Dual concept to (prefix-)periodic root is the *suffix-periodic root*, characterized by $w <_s w \cdot r$. This is just one instance of the natural duality given by the reversal (or mirror) symmetry, which is often exploited in human proofs on an intuitive basis. See Sect. 8 for more details on how reversal symmetry is used in our formalization.

If r is a periodic root of w, then we also say that w has a *period* $|r|$. Note that this is equivalent to w having a suffix-periodic root r' with $|r| = \left|r'\right|$.

A periodic root r of w need not be primitive, but it is always possible to consider the corresponding primitive root $\rho(r)$, which is also a periodic root of w. Note that any word has infinitely many periodic roots since we allow r to be longer than w. Nevertheless, a word can have more than one period even if we consider only periods shorter than $|w|$. For example, the word $aaabaa$ has periods 4 and 5, with corresponding periodic roots $aaab$ and $aaaba$. Such a possibility is controlled by the Periodicity Lemma, often called the Fine–Wilf Theorem [7]:

Lemma 1 (per-lemma-comm) *If $w \leq_p uw$ and $w \leq_p vw$, with $|u| + |v| - \gcd(|u|, |v|) \leq |w|$, then $uv = vu$.*

This implies, together with Lemma 4 below, that the word $aaabaa$ of length 7 is a word of the greatest length with periods 4 and 5 that is not a power of a single letter. Usually, the weaker test $|u| + |v| \leq |w|$ is sufficient to indicate that u and v commute.

2.1.3 Maximal Prefix

Given a word r, we define the *maximal r-prefix* of a word w as $w \wedge_p r^\omega$. For example, if $r = ab$, then the maximal r-prefix of $ababaabab$ is $ababa$. Sometimes, the easiest way to prove that two words are not equal is to show that they have different r-prefixes for a suitable r, see the proof of Lemma 24 below.

2.1.4 Conjugation

We say that two words u and v are *conjugate* and write $u \sim v$ if $u = rq$ and $v = qr$ for some words r and q. Note that conjugation is an equivalence whose classes are also called *cyclic words*. For example, aab and aba are conjugate. They are elements of the conjugacy class $\{aab, aba, baa\}$. A word u is a *cyclic factor* of w if it is a factor of some conjugate of w. If $|u| \leq |w|$, this is equivalent to u being a factor of ww.

Conjugation $u \sim v$ is characterized as follows:

Lemma 2 (conjugation) *If $uz = zv$ for nonempty u, then there exists words r and q and an integer k such that $u = rq$, $v = qr$ and $z = (rq)^k r$.*

A word w has a periodic root r if it is a prefix of r^ω. If w is a factor, not necessarily a prefix, of r^ω, then it has a periodic root which is a conjugate of r. In particular, if $|u| = |v|$, then $u \sim v$ is equivalent to u and v each being a factor of a power of the other word . Note also that if $u \sim v$ and $u = t^k$, then $v = s^k$ for some $s \sim t$.

2.1.5 Lyndon and Schützenberger

The following result [20], called the Lyndon–Schützenberger Theorem, has been mentioned in the introduction:

Theorem 3 (Lyndon–Schutzenberger) *If $x^j y^k = z^\ell$ with $j \geq 2$, $k \geq 2$ and $\ell \geq 2$, then the words x, y and z commute.*

In the context of our paper, the preferred reading is the following: If x and y do not commute (that is, they form a binary code, see below), and $x^j y^k$ is imprimitive, then $j = 1$ or $k = 1$.

2.2 Binary Codes

2.2.1 Code

A set of words X is a *code* if its elements do not satisfy any nontrivial relation, that is, they are a basis of a free monoid. A monoid M of words is free if and only if it satisfies the *stability condition* which is the implication

$$u, v, uz, zv \in M \Longrightarrow z \in M.$$

This is a useful characterization since it allows to recognize a free monoid without knowing its basis. Yet another formulation is that `concat` is a bijection between `lists` X and $\langle X \rangle$ for a code X. The name "code" is motivated by the latter fact: any concatenation of a list of elements of the code can be uniquely factorized ("decoded") back into the original list.

2.2.2 Commutation

It can be shown that a two-element set $\{x, y\}$ is a (binary) code if and only if x and y do not commute. Therefore, if we say that $B = \{x, y\}$ is a binary code, it is equivalent to saying that $xy \neq yx$, which is often preferred in the formalization. By the definition of code, the fact means that two words commute if and only if they satisfy a nontrivial relation. This makes commutation an important property, which can be characterized as follows:

Lemma 4 (comm) *$xy = yx$ if and only if $x = t^k$ and $y = t^m$ for some word t and some integers $k, m \geq 0$.*

Since every nonempty word has a (unique) primitive root, the word t can be chosen primitive (k or m can be chosen 0 if x or y is empty). This implies that two nonempty words x and y commute if and only if $\rho(x) = \rho(y)$. A useful consequence is that x and y commute if and only if x^j and y^k commute, where $0 < k, j$.

2.2.3 Synchronization

A crucial property of a primitive word t is that it cannot be a nontrivial factor of its own square. More specifically, for a general word u, the equality $u \cdot u = p \cdot u \cdot s$ implies that all three words p, s, u commute. Indeed, it follows that $u \cdot u \cdot u = p \cdot u \cdot s \cdot u = u \cdot p \cdot u \cdot s$, and thus $p \cdot u = u \cdot p$ and $s \cdot u = u \cdot s$. Therefore, p, s and u have a common primitive root t. Hence, the presence of a nontrivial factor u inside uu can be obtained exclusively by a shift by several t's. Especially, if u is primitive, i.e., $u = t$, we either have $p = u = t$ and s empty, or vice versa. We shall refer to this idea of shifting by the primitive root as *synchronization*. A slightly more general formulation, which can be seen as an instance of synchronization, says that if $w \cdot v$ is a prefix of v^ω, then w and v commute.

2.2.4 Decoding Delay

Let x and y be two words that do not commute. Equivalently, let $B = \{x, y\}$ be a binary code. Then the longest common prefix $(x \cdot y) \wedge_p (y \cdot x)$ of $x \cdot y$ and $y \cdot x$, denote it α, is a strict prefix of both $x \cdot y$ and $y \cdot x$. Let c_x and c_y be distinct letters following α in $x \cdot y$ and $y \cdot x$ respectively. A crucial property of α, not very difficult to prove, is that it is a prefix of any sufficiently long word in $\langle\{x, y\}\rangle$. Moreover, if $\mathbf{w} = [u_1, u_2, \ldots, u_n] \in \texttt{lists}\,\{x, y\}$ is such that $\texttt{concat}\,\mathbf{w}$ is longer than α, then $\alpha \cdot [c_x]$ is a prefix of $\texttt{concat}\,\mathbf{w}$ if $u_1 = x$ and $\alpha \cdot [c_y]$ is a prefix of $\texttt{concat}\,\mathbf{w}$ if $u_1 = y$. That is why the length of α is sometimes called *the decoding delay* of the binary code $\{x, y\}$. Note that the property indeed in particular implies that $\{x, y\}$ is a code, that is, it does not satisfy any nontrivial relation. For example, $\alpha = aba$ if $x = abaa$ and $y = ab$, with $c_x = a$ and $c_y = b$. Suppose that we want to decode a word starting with $abaaabaa \cdots \in \langle\{x, y\}\rangle$, that is, we want to find its unique decomposition into words x and y. We first see the prefix $\alpha = aba$ which says nothing about the decomposition since it is common to all messages. It is followed by $c_x = a$ which indicates that the first word in the decomposition is x.

The property is also behind our method `mismatch` (see Sect. 8). Finally, using this property, the proof of the following lemma is straightforward.

Lemma 5 (bin-code-lcp-concat') *Let $X = \{x, y\}$ be a binary code, and let $\mathbf{w}_0, \mathbf{w}_1 \in$ $\texttt{lists}\,X$ be such that $\texttt{concat}\,\mathbf{w}_0$ and $\texttt{concat}\,\mathbf{w}_1$ are not prefix-comparable. Then*

$$(\texttt{concat}\,\mathbf{w}_0) \wedge_p (\texttt{concat}\,\mathbf{w}_1) = \texttt{concat}(\mathbf{w}_0 \wedge_p \mathbf{w}_1) \cdot (xy \wedge_p yx).$$

3 Main Theorem

Let us introduce the central definition of the paper.

Definition 6 We say that a set X of words is *primitivity-preserving* if there is no list $\mathbf{w} \in \texttt{lists}\,X$ such that

- $|\mathbf{w}| \geq 2$;
- $\mathbf{w}$ is primitive; and
- $\texttt{concat}\,\mathbf{w}$ is imprimitive.

If X is not primitivity-preserving, then each primitive $\mathbf{w} \in \texttt{lists}\,X$ of length at least two such that $\texttt{concat}\,\mathbf{w}$ is imprimitive will be called a *witness* (of the fact that X is not primitivity-preserving).

Note that our definition does not take into account singletons $\mathbf{w} = [z]$. In particular, X can be primitivity-preserving even if some $z \in X$ is imprimitive. For example, if $X = \{aa, b\}$, then X is primitivity-preserving despite the fact that `concat` $\mathbf{w}$ of the primitive list $\mathbf{w} = [x]$ is an imprimitive word aa. On the other hand, in the binary case, Theorem 7 gives conditions under which x and y must be primitive if $\{x, y\}$ is not primitivity-preserving.

Mitrana [22] shows that X is primitivity-preserving if and only if it is the minimal set of generators of a "pure monoid", cf. [3, p. 276]. The latter definition requires that even individual code words are primitive, which is a significant difference from our definition (see, in particular, the concept of a *non-overlapping set* in Sect. 7). Mitrana formulates the primitivity of a set in terms of morphisms, that is, mappings $h : A^* \to \Sigma^*$ satisfying $h(u \cdot v) = h(u) \cdot h(v)$ for all $u, v \in A^*$. This is equivalent to our formulation in the following way. Consider a binary alphabet $A = \{a, b\}$, and a morphism $h : A^* \to \Sigma^*$ defined by $h : a \mapsto x, b \mapsto y$. Then h represents our set $X = \{x, y\}$, and we can write, for example, $h(abaa)$ instead of `concat`$[x, y, x, x]$. The concept of preserving primitivity puts our paper into a wider context of morphisms preserving a given property, most classically square-freeness; see for example a characterization of square-free morphisms over three letters by Crochemore [6].

The target claim of our formalization is the following characterization of lists witnessing that a binary code is not primitivity-preserving:

Theorem 7 (bin-imprim-code)

Let $B = \{x, y\}$ be a binary code that is not primitivity-preserving. Then there are integers $j \geq 1$ and $k \geq 1$, with $k = 1$ or $j = 1$, such that the following conditions are equivalent for any $\mathbf{w} \in$ `lists` B with $|\mathbf{w}| \geq 2$:

- $\mathbf{w}$ *is primitive, and* `concat` $\mathbf{w}$ *is imprimitive;*
- $\mathbf{w}$ *is conjugate with $[x]^j[y]^k$.*

Moreover, assuming $|y| \leq |x|$,

- *if $j \geq 2$, then $j = 2$ and $k = 1$, and both x and y are primitive;*
- *if $k \geq 2$, then $j = 1$ and x is primitive.*

First, note that the integers j and k depend on the set B only. Consequently, the theorem says that there is a unique witness of the form $[x]^j[y]^k$ for a given B, and all witnesses are conjugate with it.

Proof (overview) Let $\mathbf{w}$ be a witness. That is, $|\mathbf{w}| \geq 2$, $\mathbf{w}$ is primitive, and `concat` $\mathbf{w}$ is imprimitive. Since $[x]^j[y]^k$ and $[y]^k[x]^j$ are conjugate, we can suppose, without loss of generality, that $|y| \leq |x|$.

First, we want to show that $\mathbf{w}$ is conjugate with $[x]^j[y]^k$ for some $j, k \geq 1$ such that $k = 1$ or $j = 1$. Since $\mathbf{w}$ is primitive and of length at least two, it contains both x and y. If it contains one of these letters exactly once, then $\mathbf{w}$ is clearly conjugate with $[x]^j[y]^k$ for $j = 1$ or $k = 1$. Therefore, the difficult part is to show that no primitive $\mathbf{w}$ with `concat` $\mathbf{w}$ imprimitive can contain both letters at least twice. This is the main task of the rest of the paper, which is finally accomplished by Theorem 25 claiming that lists that contain at least two occurrences of x are conjugate with $[x, x, y]$. To complete the proof of the first part of the theorem, it remains to show that j and k are unique for a given $\{x, y\}$. This follows from Lemma 9.

Note that the imprimitivity of `concat` $\mathbf{w}$, with $\mathbf{w} = [x]^j[y]^k$, induces the equality $x^j y^k = z^\ell$ for some z and $\ell \geq 2$. The already mentioned seminal result of Lyndon and Schützenberger

(Theorem 3) shows that j and k cannot be simultaneously at least two, since otherwise x and y commute. For the same reason, considering its primitive root, the word y is primitive if $j \geq 2$. Similarly, x is primitive if $k \geq 2$. The primitivity of x when $j = 2$ is a part of Theorem 25. □

We start by giving a complete parametric solution of the equation $x^j y^k = z^\ell$ in the following theorem. This will eventually yield, after the proof of Theorem 7 is completed, a full description of not primitivity-preserving binary codes. Since the equation is mirror symmetric, we omit symmetric cases by assuming $|y| \leq |x|$.

Theorem 8 (LS-parametric-solution) *Let $j, k \geq 1$, $\ell \geq 2$ and $|y| \leq |x|$.*
The equality $x^j y^k = z^\ell$ holds in exactly the following cases:

A. *There exist a word r and integers $m, n, t \geq 0$ such that*

$$mj + nk = t\ell, \quad \text{and}$$
$$x = r^m, \quad y = r^n, \quad z = r^t;$$

B. *$j = k = 1$ and there exist non-commuting words r and q, and integers $m, n \geq 0$ such that*

$$m + n + 1 = \ell, \quad \text{and}$$
$$x = (rq)^m r, \quad y = q(rq)^n, \quad z = rq;$$

C. *$j = 1$ and $k \geq 2$ and there exist non-commuting words r and q such that*

$$x = (qr^k)^{\ell-1} q, \quad y = r, \quad z = qr^k;$$

D. *$j = 1$ and $k \geq 2$ and there exist non-commuting words r and q, an integer $m \geq 1$ such that*

$$x = (qr(r(qr)^m)^{k-1})^{\ell-2} qr(r(qr)^m)^{k-2} rq, \; y = r(qr)^m, \; z = qr(r(qr)^m)^{k-1};$$

E. *$j = \ell = 2$, $k = 1$ and there exist non-commuting words r and q and an integer $m \geq 2$ such that*

$$x = (rq)^m r, \quad y = qrrq, \quad z = (rq)^m rrq.$$

All the cases of the last theorem are illustrated in Fig. 1. See also Example 11.

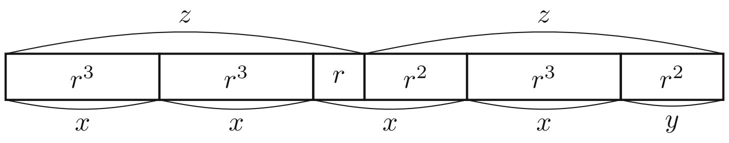
Proof If x and y commute, then all three words commute, hence they are powers of a common word. A length argument yields the solution A.

Assume now that $\{x, y\}$ is a code. It follows that z does not commute with x. We have shown in the overview of the proof of Theorem 7 that $j = 1$ or $k = 1$ by the Lyndon–Schützenberger Theorem 3. The solution is then split into several cases.

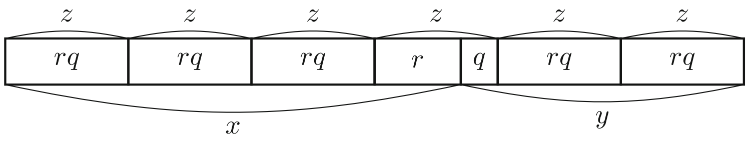
Case 1: $j = k = 1$.
Let m and r be such that $z^m r = x$ with r a strict prefix of z. By setting $z = rq$, we obtain the solution B with $n = \ell - m - 1$.

Case 2: $j \geq 2, k = 1$.
Since $|y| \leq |x|$ and $\ell \geq 2$, we have

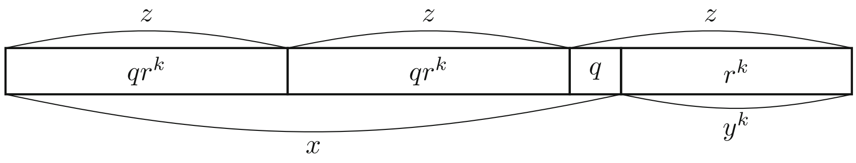
$$2|z| \leq \left|z^\ell\right| = \left|x^j\right| + |y| < 2\left|x^j\right|,$$



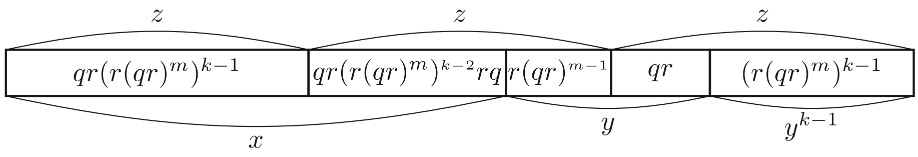
(a) Case A of Theorem 8 for $j = 4$, $k = 1$, $\ell = 2$, $m = 3$, $n = 2$, $t = 7$.



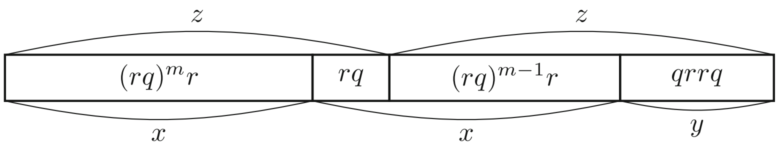
(b) Case B of Theorem 8 for $\ell = 3$, $m = 3$, $n = 2$.



(c) Case C of Theorem 8 for $\ell = 3$.



(d) Case D of Theorem 8 for $\ell = 3$.



(e) Case E of Theorem 8 for $\ell = 2$.

Fig. 1 Illustration of the distinct cases of Theorem 8

hence z is a strict prefix of x^j.

As x^j has periodic roots both z and x, and z does not commute with x, the Periodicity Lemma 1 implies $|x^j| < |z| + |x|$. That is, $z = x^{j-1}u$, $x^j = zv$ and $x = uv$ for some nonempty words u and v. Since x^j is a prefix of z^ℓ, we deduce that v is a prefix of $z^{\ell-1}$, and therefore also of x since x is a prefix of z. This implies that

$$x = uv = vu'$$

for some word u'. That is, the words u and u' are conjugate by v. The characterization of conjugate words yields $u = rq$, $u' = qr$ and $v = (rq)^n r$ for some words r, q and an integer n. Moreover, since x and y do not commute, the words r and q are nonempty.

We have

$$j\,|x| + |y| = \left|x^j y\right| = \left|z^\ell\right| = \left|(x^{j-1}u)^\ell\right| = \ell(j-1)\,|x| + \ell\,|u|\,,$$

and thus $|y| = (\ell j - \ell - j)\,|x| + \ell\,|u|$. From $|y| \le |x|$, $|u| > 0$, and $\ell \ge 2$, we deduce that $\ell j - \ell - j$ is not positive. Since $j, \ell \ge 2$, this implies $j = \ell = 2$. Then $z = x^{j-1}u = xu = uvu$. From $x^2y = z^2$, we have $uvuvy = uvuuvu = uvuvu'u$, hence $y = u'u$. Substituting $u = rq$, $u' = qr$, and $v = (rq)^n r$, we obtain the solution E with $m = n + 1$, where $m \ge 2$ follows from $|y| \le |x|$.
Case 3: $j = 1, k \ge 2, y^k \le_s z$.
We have $z = qy^k$ for some word q. Noticing that $x = z^{\ell-1}q$ and setting $y = r$ yields the solution C. The words r and q do not commute since x and y do not commute.
Case 4: $j = 1, k \ge 2, z <_s y^k$.
This case is analogous to Case 2. Using the Periodicity Lemma 1, we obtain $uy^{k-1} = z$, $y^k = vz$, and $y = vu$ with nonempty u and v. As v is a suffix of $z^{\ell-1}$ and is shorter than y, it is also a suffix of y, and we have $y = vu = u'v$ for some u' conjugate with u by v. We therefore have nonempty words r and q such that $u' = rq$, $u = qr$, and $v = (rq)^n r$. Using $y = u'v$, $z = uy^{k-1}$ and $z^{\ell-1} = xv$, we obtain the solution D with $m = n + 1$. Again, the words r and q do not commute since x and y, which are generated by r and q, do not commute.

The proof is completed by a direct verification of the converse. □

The case analysis in the previous proof also shows that at most one of the cases holds for given x, y and z.

Recall that Theorem 7 claims two things. First, if there is a word $\mathbf{w}$ witnessing that $\{x, y\}$ is not primitivity-preserving, then $\mathbf{w}$ is conjugate with $[x]^j[y]^k$ for some j and k. Second, there is at most one such pair (j, k) for a given $\{x, y\}$, (and exactly one if $\{x, y\}$ is not primitivity-preserving). The next lemma proves the second claim.

Lemma 9 (LS-unique) *Let $B = \{x, y\}$ be a binary code. Assume $j, k, j', k' \ge 1$. If both $x^j y^k$ and $x^{j'} y^{k'}$ are imprimitive, then $j = j'$ and $k = k'$.*

Proof Let z_1, z_2 be primitive words and $\ell, \ell' \ge 2$ be such that

$$x^j y^k = z_1^\ell \quad \text{and} \quad x^{j'} y^{k'} = z_2^{\ell'}. \tag{1}$$

Since B is a code, the words x and y do not commute. We proceed by contradiction.
Case 1: First, assume that $j = j'$ and $k \ne k'$.
Let, without loss of generality, $k < k'$. From (1) we obtain $z_1^\ell y^{k'-k} = z_2^{\ell'}$. The case $k' - k \ge 2$ is impossible due to the Lyndon–Schützenberger Theorem 3. Hence $k' - k = 1$. This is another place where the formalization led to a simple and nice general lemma (easily provable by the Periodicity Lemma 1) which will turn out to be useful also in the proof of Theorem 25. Namely, the lemma `imprim-ext-suf-comm` claims that if both uv, and uvv are imprimitive, then u and v commute (see also a comment in Sect. 8). We apply this lemma to $u = x^j y^{k-1}$ and $v = y$, obtaining a contradiction to the assumption that x and y do not commute.
Case 2. The case $k = k'$ and $j \ne j'$ is symmetric to Case 1.
Case 3. Let finally $j \ne j'$ and $k \ne k'$. The Lyndon–Schützenberger Theorem 3 implies that either j or k is one, and similarly either j' or k' is one. We can therefore assume that $k = j' = 1$ and $k', j \ge 2$. Moreover, we can assume that $|y| \le |x|$. Indeed, in the opposite case, we can consider the words $y^k x^j$ and $y^{k'} x^{j'}$ instead, which are also both imprimitive.

Theorem 8 now allows only the case E for the equality $x^j y = z_1^\ell$. We therefore have $j = \ell = 2$ and $x = (rq)^m r$, $y = qrrq$ for an integer $m \ge 2$ and some non-commuting words r and q. Assume that z_2 and rq have the same primitive root. Then $qr = rq$, since $|qr| = |rq|$ and $y = qrrq$ is a suffix of $z_2^{\ell'}$, a contradiction. Therefore z_2 and rq do not commute. Since $y = qrrq$ is a suffix of z_2^{ℓ}, this implies that z_2 and rq do not commute. Consider the word $x \cdot qr = (rq)^m rqr$, which is a prefix of xy, and therefore also of z_2^{ℓ}. This means that $x \cdot qr$ has two periodic roots, namely rq and z_2, and the Periodicity Lemma 1 implies that $|x \cdot qr| < |rq| + |z_2|$. Hence x is shorter than z_2. The equality $xy^{k'} = z_2^{\ell'}$, with $\ell' \geq 2$, now implies on one hand that $rqrq$ is a prefix of z_2, and on the other hand that z_2 is a suffix of $y^{k'}$. It follows that $rqrq$ is a factor of $(qrrq)^{k'}$. Hence $rqrq$ and $qrrq$ are conjugate and $qrrq$ is a square since $rqrq$ is a square, see Sect. 2.1.4. Thus they both have a period of length $|rq|$, which implies $qr = rq$, a contradiction. □

A natural question is whether the property of being primitivity-preserving it algorithmically decidable for given $\{x, y\}$. It follows from Theorem 7 that it is enough to check primitivity of elements from the set

$$\{xxy\} \cup \{xy^k \mid k \geq 1\}.$$

From the computational point of view, we therefore need an upper bound on k in terms of $|x|$ and $|y|$. Such a bound is given by the following lemma.

Lemma 10 (LS-exp-le) *Let $B = \{x, y\}$ be a binary code and let $x \cdot y^k = z^l$ with $k, \ell \geq 2$. Then*

$$k \leq \frac{|x| - 4}{|y|} + 2.$$

Proof By Theorem 8, it is enough to consider cases C and D. In case C, using successively $\ell \geq 2$, $|r| \geq 1$ and $|q| \geq 1$, we have

$$\frac{|x|-4}{|y|} + 2 = \frac{(\ell-1)(|q| + k\,|r|) + |q| - 4}{|r|} + 2 \geq \frac{2\,|q| + k\,|r| - 4}{|r|} + 2 \geq$$
$$\geq k + 2 + \frac{2\,|q| - 4}{|r|} \geq k + 2 - \frac{2}{|r|} \geq k.$$

Similarly, in case D, using $\ell \geq 2$ and $|qr| \geq 2$, we have

$$\frac{|x|-4}{|y|} + 2 = \frac{(\ell-2)\left|qr(r(qr)^m)^{k-1}\right| + 2\,|qr| + (k-2)\,|y| - 4}{|y|} + 2 \geq$$
$$\geq \frac{2\,|qr| + (k-2)\,|y| - 4}{|y|} + 2 \geq k + \frac{2\,|qr| - 4}{|y|} \geq k.$$

□

Note that the bound is sharp since we have equality for $\ell = 2$ and $|q| = |r| = 1$ in both cases as the following example points out.

Example 11 (examples-bound-optimality) For any $k \geq 2$ the triples

$$x = \mathtt{0}\mathtt{1}^k\mathtt{0} \qquad y = \mathtt{1} \qquad z = \mathtt{0}\mathtt{1}^k, \tag{C}$$

$$x = \mathtt{01}(\mathtt{101})^{k-2}\mathtt{10} \qquad y = \mathtt{101} \qquad z = \mathtt{01}(\mathtt{101})^{k-1} \tag{D}$$

corresponding to cases C and D (with $m = 1$) of Theorem 8, satisfy $|y| \leq |x|$, $x \cdot y^k = z \cdot z$, $x \cdot y \neq y \cdot x$ and $k = (|x|-4)/|y| + 2$.

We remark that the primitivity-preserving property is decidable for all finite sets due to the characterization of star-free regular languages as those with aperiodic syntactic monoid. See Mitrana [22, Corollary 6] for more details and further references.

The rest of the paper, and therefore also of the proof of Theorem 7, is organized as follows. In Sect. 4, we introduce a general theory of interpretations, which is behind the main idea of the proof. In Sect. 5, we apply it to the (relatively simple) case of a binary code with words of the same length. In Sect. 6, we characterize the unique disjoint extendable $\{x, y\}$-interpretation of the square of the longer word x. This is a result of independent interest, and also the cornerstone of the proof of Theorem 7 which is completed in Sect. 7 by showing that a list containing at least two x's witnessing that $\{x, y\}$ is not primitivity-preserving is conjugate with $[x, x, y]$.

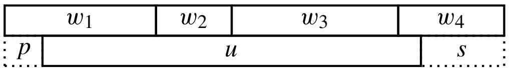
4 Interpretations and the Main Idea

Let X be a code, and let u be a factor of $\texttt{concat}\,\mathbf{w}$ for some $\mathbf{w} \in \texttt{lists}\,X$. The natural question is to decide how u can be produced as a factor of words from X, or, in other words, how it can be interpreted in terms of X. This motivates the following definition.

Definition 12 Let X be a set of words over Σ. We say that the triple $(p, s, \mathbf{w}) \in \Sigma^* \times \Sigma^* \times \texttt{lists}\,X$ is an X-*interpretation* of a word $u \in \Sigma^*$ if

- $\mathbf{w}$ is nonempty;
- $p \cdot u \cdot s = \texttt{concat}\,\mathbf{w}$;
- $p <_p \texttt{hd}\,\mathbf{w}$ and
- $s <_s \texttt{last}\,\mathbf{w}$.

The definition is illustrated by the following figure, where $\mathbf{w} = [w_1, w_2, w_3, w_4]$:



The second condition of the definition motivates the notation $p\,u\,s \sim_{\mathcal{I}} \mathbf{w}$ for the situation when $(p, s, \mathbf{w})$ is an X-interpretation of u.

Remark 13 For sake of historical reference, we remark that our definition of X-interpretation differs from the one used in [1]. The formulation in [1] of the situation depicted by the above figure would be that u is interpreted by the triple $(s', w_2 \cdot w_3, p')$ where $p \cdot s' = w_1$ and $p' \cdot s = w_4$. This is less convenient for two reasons. First, the decomposition of $w_2 \cdot w_3$ into $[w_2, w_3]$ is only implicit here (and even possibly ambiguous if X is not a code). Second, while it is required that the words p' and s' are a prefix and a suffix, respectively, of an element from X, the identity of that element is left open, and has to be specified separately.

If u is a nonempty element of $\langle X \rangle$ and $u = \texttt{concat}\,\mathbf{u}$ for $\mathbf{u} \in \texttt{lists}\,X$, then the X-interpretation $\varepsilon\,u\,\varepsilon \sim_{\mathcal{I}} \mathbf{u}$ is called *trivial*. Note that the trivial X-interpretation is unique if X is a code.

As nontrivial X-interpretations of elements from $\langle X \rangle$ are of particular interest, the following two concepts are useful.

Definition 14 An X-interpretation $p\,u\,s \sim_{\mathcal{I}} \mathbf{w}$ of $u = \texttt{concat}\,\mathbf{u}$ is called

- *disjoint* if $\texttt{concat}\,\mathbf{w}' \neq p \cdot \texttt{concat}\,\mathbf{u}'$ whenever $\mathbf{w}' \leq_p \mathbf{w}$ and $\mathbf{u}' \leq_p \mathbf{u}$;

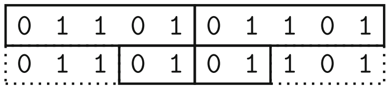


Fig. 2 Non-disjoint interpretation

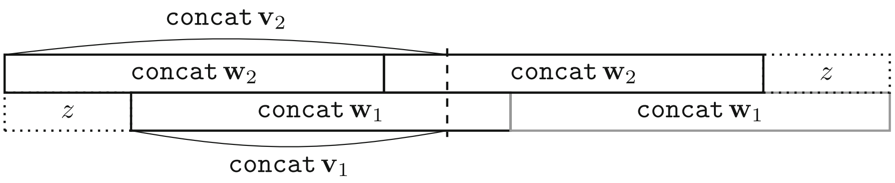


Fig. 3 The situation prohibited by Lemma 15

- *extendable* if $p \leq_s w_p$ and $s \leq_p w_s$ for some elements $w_p, w_s \in \langle X \rangle$.

Informally, an interpretation is disjoint if no "edge" between words in **u** fits an "edge" between words in **w** in the equality $p \cdot \texttt{concat}\,\mathbf{u} \cdot s = \texttt{concat}\,\mathbf{w}$. That is, the equality does not split in two shorter ones. Let, for example, $x = \texttt{01101}$ and $y = \texttt{01}$ and let $\mathbf{u} = [y, y]$. Then the interpretation

$$\texttt{011}\,\texttt{0101}\,\texttt{101} \sim_{\mathcal{I}} [x, x]$$

of $u = \texttt{0101} = \texttt{concat}[y, y]$ is not disjoint, see Fig. 2. Moreover, it is not extendable, since $\texttt{101}$ is not a prefix of any word from $\langle\{x, y\}\rangle$. We could also argue that it is not extendable because $\texttt{011}$ is not a suffix of any word from $\langle\{x, y\}\rangle$. In contrast, the interpretation in Fig. 5 is disjoint and extendable.

Note that a disjoint X-interpretation is not trivial, and that being disjoint is relative to a chosen factorization **u** of u (which is nevertheless unique if X is a code). It should be clear from the definition in which way an extendable interpretation of $\texttt{concat}\,\mathbf{u}$ can be "extended" into an X-interpretation of $\texttt{concat}\,\mathbf{w}$.

The definitions above are naturally motivated by **the main idea** of the characterization of sets X that do not preserve primitivity, which dates back to Lentin and Schützenberger [18]. If **w** is primitive while $\texttt{concat}\,\mathbf{w}$ is imprimitive, say $\texttt{concat}\,\mathbf{w} = z^\ell$, $\ell \geq 2$, then the shift by z provides a nontrivial and extendable X-interpretation of $\texttt{concat}\,\mathbf{w}$. (In fact, there are $\ell - 1$ such nontrivial interpretations). Moreover, the following lemma, formulated in a more general setting of two lists $\mathbf{w}_1$ and $\mathbf{w}_2$, implies that the X-interpretation is disjoint if X is a code.

Lemma 15 (shift-disjoint, shift-interpret) *Let X be a code. Let $\mathbf{w}_1, \mathbf{w}_2 \in \texttt{lists}\,X$ be such that $z \cdot \texttt{concat}\,\mathbf{w}_1 = \texttt{concat}\,\mathbf{w}_2 \cdot z$ where $z \notin \langle X \rangle$. Then $z \cdot \texttt{concat}\,\mathbf{v}_1 \neq \texttt{concat}\,\mathbf{v}_2$, whenever $\mathbf{v}_1 \leq_p \mathbf{w}_1^n$ and $\mathbf{v}_2 \leq_p \mathbf{w}_2^n$, $n \in \mathbb{N}$.*

*In particular $\texttt{concat}\,\mathbf{u}$ has a disjoint extendable X-interpretation for any nonempty prefix **u** of $\mathbf{w}_1$.*

Proof First, note that $z \cdot \texttt{concat}\,\mathbf{w}_1^n = \texttt{concat}\,\mathbf{w}_2^n \cdot z$ for any n. Let $\mathbf{w}_1^n = \mathbf{v}_1 \cdot \mathbf{v}_1'$ and $\mathbf{w}_2^n = \mathbf{v}_2 \cdot \mathbf{v}_2'$. If $z \cdot \texttt{concat}\,\mathbf{v}_1 = \texttt{concat}\,\mathbf{v}_2$, then also $\texttt{concat}\,\mathbf{v}_2' \cdot z = \texttt{concat}\,\mathbf{v}_1'$. This contradicts $z \notin \langle X \rangle$ by the stability condition. We have proved the first part of the lemma excluding the situation illustrated by Fig. 3. The corresponding lemma in formalization is $\texttt{shift-disjoint}$.

The second part is covered in the formalization by $\texttt{shift-interpret}$. An extendable X-interpretation of $\texttt{concat}\,\mathbf{u}$ is induced by the fact that $\texttt{concat}\,\mathbf{u}$ is a factor of

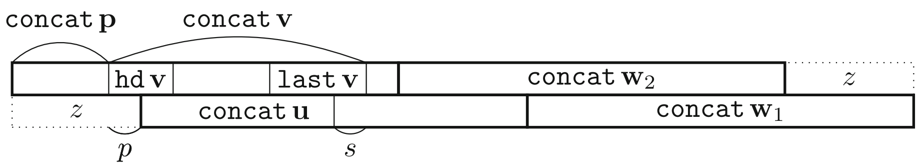


Fig. 4 Interpretation of concat **u** from Lemma 15

concat$(\mathbf{w}_2 \cdot \mathbf{w}_2)$. By Lemma 2, there are words q and r such that concat $\mathbf{w}_1 = rq$, concat $\mathbf{w}_2 = qr$ and $z = (rq)^m r$ for some r. Since $z \notin \langle X \rangle$, we deduce that also $r \notin X$ and the assumptions of the present lemma are satisfied for r. We can therefore assume that $z = r$. In particular, $|z| < |$concat $\mathbf{w}_2|$ and $z \cdot$ concat $\mathbf{u}$ is a prefix of concat $\mathbf{w}_2^2$. Let $\mathbf{p}$, $\mathbf{v}$ be such that

- $\mathbf{p} \cdot \mathbf{v} \leq \mathbf{w}_2^2$;
- $\mathbf{p}$ is the maximum prefix of $\mathbf{w}_2^2$ such that concat $\mathbf{p} \leq_p z$; and
- $\mathbf{p} \cdot \mathbf{v}$ is the minimum prefix of $\mathbf{w}_2^2$ such that $z \cdot$ concat $\mathbf{u} \leq_p$ concat$(\mathbf{p} \cdot \mathbf{v})$,

and let p, s be words such that $p \cdot$ concat $\mathbf{u} \cdot s =$ concat $\mathbf{v}$. The situation is illustrated by Fig. 4. Here p and s satisfy concat $\mathbf{p} \cdot p = z$ and $z \cdot$ concat $\mathbf{u} \cdot s =$ concat$(\mathbf{p} \cdot \mathbf{v})$. The maximality of $\mathbf{p}$, and the minimality of $\mathbf{p} \cdot \mathbf{v}$ also implies that $p <_p$ hd $\mathbf{v}$ and $s <_s$ last $\mathbf{v}$. Therefore we have p (concat $\mathbf{u}$) $s \sim_{\mathcal{I}}$ concat $\mathbf{v}$. The interpretation is disjoint by the first part of the proof. It is also extendable since s is a prefix of concat $\mathbf{s}$, where $\mathbf{u} \cdot \mathbf{s} = \mathbf{w}_1^2$, and p is a suffix of $\mathbf{w}_1$. □

Let $B = \{x, y\}$ be a binary code. In order to apply the above lemma to the imprimitive concat $\mathbf{w} = z^k$, $2 \leq k$, of a primitive $\mathbf{w} \in$ lists B, set $\mathbf{w}_1 = \mathbf{w}_2 = \mathbf{w}$. Let us verify the assumption $z \notin \langle B \rangle$. If $z =$ concat $\mathbf{z}$, with $\mathbf{z} \in$ lists B, then concat$(\mathbf{z} \cdot \mathbf{w}) =$ concat$(\mathbf{w} \cdot \mathbf{z})$, hence $\mathbf{z} \cdot \mathbf{w} = \mathbf{w} \cdot \mathbf{z}$ since B is a code. This implies that $\mathbf{w}$ and $\mathbf{z}$ have the same primitive root. Since concat $\mathbf{z}$ is strictly shorter than concat $\mathbf{w}$, we deduce that $\mathbf{w}$ is not primitive, a contradiction.

See also Fig. 7 which illustrates our main application of the lemma with $\mathbf{u} = [x, x]$, $\mathbf{w}_1 = \mathbf{w}_2$, and $\mathbf{v} = [x, y, x]$.

5 Uniform Binary Codes

In this section, we use the main idea in a relatively simple case of *uniform* binary codes, that is, binary codes $B = \{x, y\}$ with $|x| = |y|$. The key ingredient is the following technical lemma characterizing possible $\{x, y\}$-interpretations of the word $x \cdot y$ in the uniform case.

Lemma 16 (uniform-square-interp) *Let $B = \{x, y\}$ be a binary code with $|x| = |y|$. Let p $(x \cdot y)$ $s \sim_{\mathcal{I}}$ $\mathbf{v}$ be a nontrivial B-interpretation. Then $\mathbf{v} = [x, y, x]$ or $\mathbf{v} = [y, x, y]$ and $x \cdot y$ is imprimitive.*

Proof We have $p \cdot x \cdot y \cdot s =$ concat $\mathbf{v}$, where p and s are not empty (otherwise the interpretation is trivial). Therefore $0 < |p \cdot s| < 2\,|x|$, and a length argument yields that $|\mathbf{v}|$ is three. A straightforward way to prove the claim is to consider all eight possible candidates. If $\mathbf{v} = [x, y, x]$ or $\mathbf{v} = [y, x, y]$, then $x \cdot y$ is a nontrivial factor of its square $(x \cdot y) \cdot (x \cdot y)$, which yields the imprimitivity of $x \cdot y$ (see Sect. 2.2.3).

The remaining six cases can be easily excluded one by one. In each case we obtain $x = y$, a contradiction to B being a code.

- If $\mathbf{v} = [x, x, x]$ then $p \cdot x \cdot y \cdot s = x \cdot x \cdot x$ implies, by synchronization (see Sect. 2.2.3), that $y \cdot s$ and x commute, that is, $x \cdot y \cdot s = y \cdot s \cdot x$. This implies $x = y$ due to $|x| = |y|$. We can argue similarly for $\mathbf{v} = [y, y, y]$.
- If $\mathbf{v} = [x, x, y]$, then $p \cdot x \cdot y \cdot s = x \cdot x \cdot y$ implies that $x = p \cdot t$ for some word t. Then $p \cdot p \cdot t \cdot y \cdot s = p \cdot t \cdot p \cdot t \cdot y$ implies $p \cdot t = t \cdot p$ and $y \cdot s = t \cdot y$. Therefore $x \leq_p t \cdot x$ and $y \leq_p t \cdot y$. This implies that both x and y have a periodic root t, and since they are of the same length, we conclude $x = y$. A symmetric argument deals with $\mathbf{v} = [x, y, y]$.
- Let now $\mathbf{v} = [y, y, x]$. Then $p \cdot x \cdot y \cdot s = y \cdot y \cdot x$ and $y = p \cdot t$ for some word t. From $p \cdot x \cdot p \cdot t \cdot s = p \cdot t \cdot p \cdot t \cdot x$ and $|x| = |y| = |t \cdot p|$ we deduce $x = t \cdot p$. The equality $p \cdot t \cdot p \cdot y \cdot s = t \cdot p \cdot t \cdot y \cdot x$ now implies $y = p \cdot t = t \cdot p = x$. The last case $\mathbf{v} = [x, x, y]$ is again symmetric.

□

The previous proof is interesting from the point of view of the formalization. It is a case analysis in which each case is easy. It is nevertheless not always easy to check that the case analysis is complete. See Sect. 8 for a custom method performing such a verification in our formalization. Let us also remark that to show that $\mathbf{v}$ is indeed of length three, which needs no further justification in a human proof, requires some effort in the formalization. The difference reveals the often non-reflected intuition behind human dealing with small natural numbers.

Lemma 16 immediately implies the following characterization of primitive uniform binary morphisms (see [22, Theorem 10]).

Theorem 17 (bin-uniform-prim-morph) *Let $B = \{x, y\}$ be a binary code with $|x| = |y|$. The code B is primitivity-preserving if and only if $x \cdot y$ is primitive.*

Proof If B is primitivity-preserving, then $x \cdot y$ is primitive by definition.

Assume now that $x \cdot y$ is primitive and proceed by contradiction. Let $\mathbf{w}$ be primitive of length at least two such that `concat` $\mathbf{w}$ is imprimitive. Then $\mathbf{w}$ contains both letters x and y, hence it has either $[x, y]$ or $[y, x]$ as a factor. Conjugating $\mathbf{w}$, we can assume that $[x, y]$ is a factor of $\mathbf{w}$. The imprimitivity of `concat` $\mathbf{w}$ yields a nontrivial B-interpretation of $x \cdot y$, which implies that $x \cdot y$ is not primitive by Lemma 16, a contradiction. □

The previous theorem can be reformulated as saying that an imprimitivity witness for a uniform binary code must be of length two, which means that $x \cdot y$ (and hence also $y \cdot x$) must be imprimitive. Consequently, there is only one way to get a uniform binary code not preserving primitivity: take an odd power of a primitive word of even length, and split it in half. For example, the third power of the primitive word $t = abba$ yields the code $\{abbaab, baabba\}$, which is not primitivity-preserving.

Uniform binary codes also have the following property.

Lemma 18 (bin-uniform-imprim) *Let $B = \{x, y\}$ be a binary code with $|x| = |y|$. If $x \cdot y$ is not primitive, then both x and y are primitive.*

Proof Let $x \cdot y$ be imprimitive. Then there are words r and s and a positive integer i such that $r \cdot s$ is the primitive root of $x \cdot y$, $x = (r \cdot s)^i \cdot r$ and $y = (s \cdot r)^i \cdot s$. Note that $\{r, s\}$ is a uniform binary code. Since $r \cdot s$ is primitive, the code $\{r, s\}$ is primitivity-preserving by Lemma 17. Since both $[r, s]^i \cdot [r]$ and $[s, r]^i \cdot [s]$ are primitive, we conclude that x and y, their concatenations, are also primitive. □

This immediately yields the following improved version of Theorem 17:

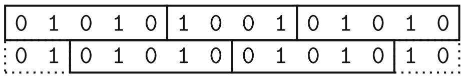


Fig. 5 Disjoint binary interpretation of a square

Theorem 19 (bin-uniform-prim-morph') *Let $B = \{x, y\}$ be a binary code with $|x| = |y|$. If $x \cdot y$ is primitive or if at least one of x and y is imprimitive, then B is primitivity-preserving.*

We will later need the following corollary.

Lemma 20 (bin-imprim-not-conjug) *Let $B = \{x, y\}$ be a uniform binary code which is not primitivity-preserving. Then x and y are not conjugate.*

Proof By Theorem 17, the word $x \cdot y$ is imprimitive. Let x and y be conjugate, and let $x = p \cdot q$ and $y = q \cdot p$. Since $x \cdot y = p \cdot q \cdot q \cdot p$ is imprimitive, also $p \cdot p \cdot q \cdot q$ is imprimitive. Then p and q commute by the Lyndon–Schützenberger Theorem 3, a contradiction to B being a code. □

6 Binary Interpretation of a Square

In Sect. 4, we have pointed out the main idea of the proof of our main goal, Theorem 7. Namely, the fact that an imprimitive `concat` $\mathbf{w}$ of a primitive $\mathbf{w} \in$ `lists`$\{x, y\}$ provides a disjoint extendable $\{x, y\}$-interpretation of `concat` $\mathbf{u}$ for any factor $\mathbf{u}$ of $\mathbf{w}$. More specifically, the plan is to apply this idea to $\mathbf{u} = [x, x]$. Consequently, the core technical component of the proof is a characterization of disjoint extendable $\{x, y\}$-interpretations of the square x^2, where $\{x, y\}$ is a binary code, and $|y| \leq |x|$. This is a very nice result which is relatively simple to state but difficult to prove, and which is valuable on its own. As we mentioned already, it can be obtained from Théorème 2.1 and Lemme 3.1 [1].

Theorem 21 (square-interp-ext.sq-ext-interp) *Let $B = \{x, y\}$ be a binary code such that $|y| \leq |x|$, both x and y are primitive, and x and y are not conjugate.*

Let $p\ (x \cdot x)\ s \sim_{\mathcal{I}} \mathbf{w}$ be a disjoint extendable B-interpretation. Then

$$\mathbf{w} = [x, y, x], \qquad s \cdot p = y, \qquad p \cdot x = x \cdot s.$$

In order to appreciate the connection of the theorem to the problem of preserving primitivity, note that the definition of interpretation implies

$$p \cdot x \cdot x \cdot s = x \cdot y \cdot x,$$

hence $x \cdot y \cdot x = (p \cdot x)^2$. This will turn out to be the only way how primitivity may not be preserved if x occurs at least twice in $\mathbf{w}$. Fig. 5 yields Here is an example with $x =$ `01010` and $y =$ `1001`.

Proof of Theorem 21 By the definition of a disjoint interpretation, we have $p \cdot x \cdot x \cdot s =$ `concat` $\mathbf{w}$, where $p \neq \varepsilon$ and $s \neq \varepsilon$. A length argument implies that $\mathbf{w}$ has length at least three. Since a primitive word is not a nontrivial factor of its square, we have $\mathbf{w} = [$`hd` $\mathbf{w}] \cdot [y]^k \cdot [$`last` $\mathbf{w}]$, with $k \geq 1$. Since the interpretation is disjoint, we can split the equality into $p \cdot x =$ `hd` $\mathbf{w} \cdot y^m \cdot u$ and $x \cdot s = v \cdot y^\ell \cdot$ `last` $\mathbf{w}$, where $y = u \cdot v$, both u and v are nonempty, and $k = \ell + m + 1$, as in Fig. 6.

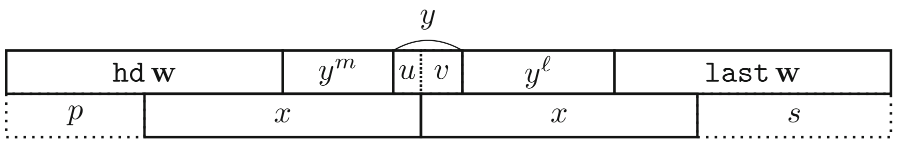


Fig. 6 Definition of u and v

We want to show $\texttt{hd}\ \mathbf{w} = \texttt{last}\ \mathbf{w} = x$ and $m = \ell = 0$. The situation is mirror symmetric so we can prove claims for hd and for last two at a time. We proceed by contradiction and exclude all unwanted situations. Some cases are concluded by showing that u and v commute, which contradicts the fact that y is primitive.

If $\texttt{hd}\ \mathbf{w} = \texttt{last}\ \mathbf{w} = y$, then x^2 and y^{k+2} share a factor of length at least $|x| + |y|$. Since x and y are primitive, this implies that they are conjugate, a contradiction. A similar argument applies when $\ell \geq 1$ and $\texttt{hd}\ \mathbf{w} = y$ (if $m \geq 1$ and $\texttt{last}\ \mathbf{w} = y$ respectively). Therefore, in order to prove $\texttt{hd}\ \mathbf{w} = \texttt{last}\ \mathbf{w} = x$, it remains to exclude the case $\texttt{hd}\ \mathbf{w} = y$, $\ell = 0$ and $\texttt{last}\ \mathbf{w} = x$ ($\texttt{last}\ \mathbf{w} = y$, $m = 0$ and $\texttt{hd}\ \mathbf{w} = x$ respectively). This is covered by one of the technical lemmas that we single out:

Lemma 22 (pref-suf-pers-short) *Let $x \leq_p v \cdot x$, $x \leq_s r \cdot u \cdot v \cdot u$ and $|x| > |v \cdot u|$ with $r \in \langle\{u, v\}\rangle$. Then $u \cdot v = v \cdot u$.*

Let us first explain how this lemma is used. With assumptions of the case we want to exclude, we obtain $x \cdot s = v \cdot x$ and $p \cdot x = y^{m+1} \cdot u = y^m \cdot u \cdot v \cdot u$. Since x is a factor of y^{m+2}, the inequality in $|u \cdot v| = |y| \leq |x|$ becomes strict because we assume that x and y are not conjugate. All hypotheses of Lemma 22 are therefore readily seen to be satisfied with $r = y^m$, the conclusion yielding a contradiction.

Proof of Lemma 22 The conclusion is trivial if u or v is empty. Assume they are nonempty. From $x \leq_s r \cdot u \cdot v \cdot u$ and $|x| > |v \cdot u|$, we obtain a nonempty word q such that $x = q \cdot v \cdot u$ and $q \leq_s r \cdot u$. From $x \leq_p v \cdot x$ we have that x is a prefix of v^ω. The equality $x = q \cdot v \cdot u$ now implies, by synchronization, that q commutes with v and $u \leq_p v \cdot u$. Then also $u \leq_p t \cdot u$, where t is the common primitive root of q and v. Moreover, t is a suffix of $r \cdot u$, where $r \in \langle\{u, t\}\rangle$. From the last fact, it is easy to see that t is a suffix of $t \cdot u^k$ for some positive k, which implies $t \leq_s t \cdot u$. From $t \leq_p u \cdot t$ and $t \leq_s u \cdot t$, it follows that u commutes with t, and therefore also with v.

Lemma 22 exemplifies virtues of formalization. First, the very fact that it is formulated as a general claim significantly improves the structure and readability of the proof of Theorem 21, in particular since it will be used several times. Second, note that we used Lemma 22 in the special case $r = y^m$. Therefore, its independent formulation led to a generalization. Third, the proof of Lemma 22 is relatively simple since it is formulated in terms of elementary principles (which themselves are independent lemmas in the formalization).

Even the rest of the proof of Theorem 21, which we only sketch here, has, in our approach, a similarly modular structure. We now have $\texttt{hd}\ \mathbf{w} = \texttt{last}\ \mathbf{w} = x$, hence $p \cdot x = x \cdot y^m \cdot u$ and $x \cdot s = v \cdot y^\ell \cdot x$. The natural way to describe this scenario is to observe that x has both the (prefix-)periodic root $v \cdot y^\ell$, and the suffix-periodic root $y^m \cdot u$.

Using again Lemma 22, we exclude situations when $\ell = 0$ and $m \geq 1$ ($m = 0$ and $\ell \geq 1$ resp.). It therefore remains to deal with the case when both m and ℓ are positive. We divide this into four lemmas according to the size of the overlap the prefix $v \cdot y^\ell$ and the suffix $y^m \cdot u$ have in x. More exactly, the cases are:

- $\left|v \cdot y^{\ell}\right| + |y^m \cdot u| \le |x|$ (no overlap)
- $|x| < \left|v \cdot y^{\ell}\right| + |y^m \cdot u| \le |x| + |u|$ (short overlap)
- $|x| + |u| < \left|v \cdot y^{\ell}\right| + |y^m \cdot u| < |x| + |u \cdot v|$ (medium overlap)
- $|x| + |u \cdot v| \le \left|v \cdot y^{\ell}\right| + |y^m \cdot u|$ (large overlap)

They are each solved by one auxiliary lemma. The first three cases yield that u and v commute, the first one being a straightforward application of the Periodicity Lemma 1 since both $\left|v \cdot y^{\ell}\right|$ and $|y^m \cdot u|$ are periods of x. The last one is an also straightforward application of the synchronization idea. Namely, if the prefix $v \cdot y^{\ell}$ and the suffix $y^m \cdot u$ of x overlap by at least $|y|$, then x is a factor of y^{ω}. Moreover, since $v \cdot u$ is both a prefix and a suffix of x, we obtain that x commutes with $v \cdot u$, a contradiction to x and y being primitive and not conjugate.

The technical part of the whole proof is concentrated in lemmas dealing with the second, and the third case (see lemmas `short-overlap` and `medium-overlap` in the theory `Binary-Square-Interpretation.thy`), which shows that for the length conditions referred to as "short overlap" and "medium overlap" above, the words u, v and x pairwise commute. The corresponding proofs are again further analyzed and decomposed into more general claims. The lemma `short-overlap` ultimately depends on the following technical claim, which is proved using elementary tools:

Lemma 23 (uvu-pref-uvv) *Let $p \cdot u \cdot v \cdot v \cdot s = u \cdot v \cdot u \cdot q$ where p is a prefix of u, both s and q are prefixes of some words from $\langle\{u, v\}\rangle$, and $|u| \le |s|$. Then u and v commute.*

This lemma is a natural ingredient of the whole proof, since it forbids a certain kind of overlap within the language generated by the binary code $\{u, v\}$. Observe that p commutes with $u \cdot v$, since $p \cdot u \cdot v$ is a prefix of $(u \cdot v)^2$. This in particular simplifies the equality in the lemma into $p \cdot v \cdot s = u \cdot q$. We therefore once more formulate a general claim whose proof is a relatively simple and intuitive application of the idea of comparison of maximal prefixes:

Lemma 24 (comm-puv-pvs-eq-uq) *Let $p \cdot u \cdot v = u \cdot v \cdot p$ and $p \cdot v \cdot s = u \cdot q$ where p is a prefix of u, both s and q are prefixes of some words from $\langle\{u, v\}\rangle$, and $|u| \le |s|$. Then u and v commute.*

Proof (sketch) We compare maximal t-prefixes of $p \cdot v \cdot s$ and $u \cdot q$, where t is the common primitive root of p and $u \cdot v$. Assume that $u \cdot v$ and $v \cdot u$ do not commute and let $\alpha = u \cdot v \wedge_p v \cdot u$. Since t is the primitive root of $u \cdot v$, we have that the maximal t-prefix of $v \cdot u$ is exactly α. Using the decoding delay principle, we deduce that the maximal t-prefix of $p \cdot v \cdot s$ is $p \cdot \alpha$. On the other hand, the word $u \cdot \alpha$, which is a prefix of $u \cdot q$, is also a prefix of t^{ω} since it is in particular a prefix of $(u \cdot v)^2$. The equality $p \cdot v \cdot s = u \cdot q$ therefore implies that $p \cdot \alpha = u \cdot \alpha$, and $u = p$. Hence u and v commute. □

We omit the proof of lemma `medium-overlap` which is of a similar nature and difficulty.

This completes the proof of $\mathbf{w} = [x, y, x]$. A byproduct of the proof is the description of words x, y, p and s. Namely, there are non-commuting words r and t, and integers m, k and ℓ such that

$$x = (rt)^{m+1} \cdot r, \qquad y = (tr)^{k+1} \cdot (rt)^{\ell+1}, \qquad p = (rt)^{k+1}, \qquad s = (tr)^{\ell+1}.$$

The claim $y = s \cdot p$ is then equivalent to $k = \ell$, and it is an easy consequence of the assumption that the interpretation is extendable. This completes the proof of Theorem 21.

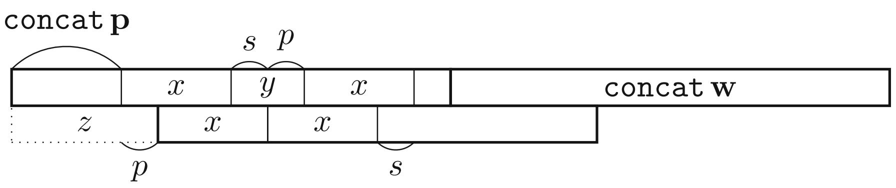


Fig. 7 Interpretation of xx induced by the shift by z

7 The Witness with Two *x*'s

In this section, we characterize lists witnessing that $\{x, y\}$ is not primitivity-preserving and containing at least two x's.

Theorem 25 (bin-imprim-longer-twice) *Let $B = \{x, y\}$ be a binary code such that $|y| \le |x|$. Let $\mathbf{w} \in$ `lists` $\{x, y\}$ be a primitive list which contains x at least twice such that* `concat` $\mathbf{w}$ *is imprimitive.*

Then $\mathbf{w} \sim [x, x, y]$ and both x and y are primitive.

We divide the proof in three steps.

7.1 The Core Case

We first prove the claim with two additional assumptions which will be subsequently removed. Namely, the following lemma shows how the knowledge about the B-interpretation of $x \cdot x$ from the previous section is used. The additional assumptions are displayed as items.

Lemma 26 (bin-imprim-primitive) *Let $B = \{x, y\}$ be a code with $|y| \le |x|$ where*

- *both x and y are primitive,*

and let $\mathbf{w} \in$ `lists` B be primitive such that `concat` $\mathbf{w}$ *is imprimitive, and*

- *$[x, x]$ is a cyclic factor of $\mathbf{w}$.*

Then $\mathbf{w} \sim [x, x, y]$.

Proof Choosing a suitable conjugate of $\mathbf{w}$, we can suppose, without loss of generality, that $[x, x]$ is a prefix of $\mathbf{w}$. Now, we want to show $\mathbf{w} = [x, x, y]$. By Lemma 20, we know that x and y are not conjugate.

Let z be the primitive root of `concat` $\mathbf{w}$, and let `concat` $\mathbf{w} = z^k$, $2 \le k$. Since $\mathbf{w}$ is primitive and $\{x, y\}$ is a code, the word z is not in $\langle\{x, y\}\rangle$. Lemma 15 yields a disjoint extendable B-interpretation of `concat` $\mathbf{w}$. In particular, the induced disjoint extendable B-interpretation of the prefix $x \cdot x$ is of the form $p\,(x \cdot x)\,s \sim_{\mathcal{I}} [x, y, x]$ by Theorem 21. Let $\mathbf{p}$ be the (nonempty and strict) prefix of $\mathbf{w}$ such that `concat` $\mathbf{p} \cdot p = z$. Then $\mathbf{p} \cdot [x, y, x]$ is a prefix of $\mathbf{w}^2$,

$$\texttt{concat}(\mathbf{p} \cdot [x, y]) = z \cdot xp, \text{ and } \texttt{concat}[x, x, y] = (xp)^2, \tag{2}$$

see Fig. 7.

Proceed by contradiction and assume $\mathbf{w} \neq [x, x, y]$. Since both $\mathbf{w}$ and $[x, x, y]$ are primitive, this implies $\mathbf{w} \cdot [x, x, y] \neq [x, x, y] \cdot \mathbf{w}$. Since $\{x, y\}$ is a code, we also have

$$\texttt{concat}(\mathbf{w} \cdot [x, x, y]) \neq \texttt{concat}([x, x, y] \cdot \mathbf{w}),$$

which yields that z and $x \cdot p$ do not commute using (2) a $\mathbf{w} = z^k$. Therefore $\{z, xp\}$ is a binary code, as well as $\{x, y\}$, and we can use its decoding delay property. Write

$$\alpha_{z,xp} = z \cdot xp \wedge_p xp \cdot z, \text{ and}$$
$$\alpha_{x,y} = x \cdot y \wedge_p y \cdot x.$$

Then

$$\begin{aligned}\alpha_{z,xp} &= z^k \cdot (xp)^2 \wedge_p (xp)^2 \cdot z^k = \\ &= \texttt{concat}(\mathbf{w} \cdot [x, x, y]) \wedge_p \texttt{concat}([x, x, y] \cdot \mathbf{w}) = \\ &= \texttt{concat}(\mathbf{w} \cdot [x, x, y] \wedge_p [x, x, y] \cdot \mathbf{w}) \cdot \alpha_{x,y},\end{aligned} \tag{3}$$

where the last equality follows from Lemma 5. Similarly, we have

$$\begin{aligned}z \cdot \alpha_{z,xp} &= z \cdot (z^k \wedge_p xp) = z^k \cdot z \cdot xp \wedge_p z \cdot xp \cdot z^k = \\ &= \texttt{concat}(\mathbf{w} \cdot \mathbf{p} \cdot [x, y]) \wedge_p \texttt{concat}(\mathbf{p} \cdot [x, y] \cdot \mathbf{w}) = \\ &= \texttt{concat}(\mathbf{w} \cdot \mathbf{p} \cdot [x, y] \wedge_p \mathbf{p} \cdot [x, y] \cdot \mathbf{w}) \cdot \alpha_{x,y}.\end{aligned} \tag{4}$$

Again, the last equality follows from Lemma 5, but we have to verify the hypothesis that $\texttt{concat}(\mathbf{w} \cdot \mathbf{p} \cdot [x, y])$ and $\texttt{concat}(\mathbf{p} \cdot [x, y] \cdot \mathbf{w})$ are not comparable. This is equivalent to $\mathbf{w} \cdot \mathbf{p} \cdot [x, y] \neq \mathbf{p} \cdot [x, y] \cdot \mathbf{w}$ since $\{x, y\}$ is a code. This is further equivalent to $\mathbf{p} \cdot [x, y] \neq \mathbf{w}$ since $\mathbf{w}$ is primitive, and $\mathbf{p} \cdot [x, y]$ is a strict prefix of $\mathbf{w} \cdot \mathbf{w}$. The possibility $\mathbf{p} = [x]$ leads to the claim we want to prove (and which we presently assume not to be true). If $\mathbf{p} \neq [x]$, then $\mathbf{p} \cdot [x, y] \neq \mathbf{w}$ follows from $\texttt{concat}\,\mathbf{p} \cdot p = z$ and $\texttt{concat}\,\mathbf{w} = z^k$ by a length argument: the word $\texttt{concat}\,\mathbf{w}$ is longer than $\texttt{concat}\,\mathbf{p} \cdot [x, y]$.

Denote

$$\mathbf{v}_1 = \mathbf{w} \cdot [x, x, y] \wedge_p [x, x, y] \cdot \mathbf{w}, \qquad \mathbf{v}_2 = \mathbf{w} \cdot \mathbf{p} \cdot [x, y] \wedge_p \mathbf{p} \cdot [x, y] \cdot \mathbf{w}.$$

From (3) and (4) we now have $z \cdot \texttt{concat}\,\mathbf{v}_1 = \texttt{concat}\,\mathbf{v}_2$. Since $\mathbf{w} \cdot [x, x, y] \neq [x, x, y] \cdot \mathbf{w}$ and $\mathbf{w} \cdot \mathbf{p} \cdot [x, y] \neq \mathbf{p} \cdot [x, y] \cdot \mathbf{w}$ we have $\mathbf{v}_1 \leq_p \mathbf{w} \cdot [x, x]$. Therefore both $\mathbf{v}_1$ and $\mathbf{v}_2$ are prefixes $\mathbf{w}^3$, a contradiction to Lemma 15. □

7.2 Dropping the Primitivity Assumption

We first deal with the situation when x and y are not primitive. A natural idea is to consider the primitive roots of x and y instead of x and y. This means that we replace the list $\mathbf{w}$ with $\mathcal{R}\,\mathbf{w}$, where $\mathcal{R}$ is the morphism mapping $[x]$ to $[\rho(x)]^{e_x}$ and $[y]$ to $[\rho(y)]^{e_y}$ where $x = \rho(x)^{e_x}$ and $y = \rho(y)^{e_y}$. For example, if $x = abab$ and $y = aa$, and $\mathbf{w} = [x, y, x] = [abab, aa, abab]$, then $\mathcal{R}\,\mathbf{w} = [ab, ab, a, a, ab, ab]$.

Let us check which hypotheses of Lemma 26 are satisfied in the new setting, that is, for the code $\{\rho(x), \rho(y)\}$ and the list $\mathcal{R}\,\mathbf{w}$. The following facts are straightforward:

- $\texttt{concat}\,\mathbf{w} = \texttt{concat}(\mathcal{R}\,\mathbf{w})$;
- if $[c, c]$ is a cyclic factor $\mathbf{w}$, where $c \in \{x, y\}$, then $[\rho(c), \rho(c)]$ is a cyclic factor of $\mathcal{R}\,\mathbf{w}$.

Let us consider the next required property:

- if $\mathbf{w}$ is primitive of length at least two, then $\mathcal{R}\,\mathbf{w}$ is also primitive. $(*)$

First, note that the above property does not necessarily hold if $\mathbf{w}$ is a singleton. Indeed, the situation we are dealing with here, namely that x or y is imprimitive, means that also $\mathcal{R}\,[x]$ or $\mathcal{R}\,[y]$ is not primitive. However, we assume that some $[c, c]$ is a factor of $\mathbf{w}$, which means that $\mathbf{w}$ can be primitive only if it contains both letters. In our formalization, the required property is proved for a more general class of codes (whose introduction was triggered by the formalization of the present result). Namely we define a locale for *non-overlapping* sets, sets in which two different elements have no overlap.

Definition 27 We say that a set C of words is *non-overlapping* if

- $\varepsilon \notin C$;
- if a nonempty word z is a suffix of u and a prefix of v, $u, v \in C$, then $u = v$;
- if u is a factor of v, then $u = v$.

Note that a non-overlapping set is a code, even a bifix code, that is, no two distinct code words are prefix (suffix) comparable. From this it follows that $\texttt{concat}\,\mathbf{u} \leq_p \texttt{concat}\,\mathbf{v}$ implies $\mathbf{u} \leq_p \mathbf{v}$ ($\texttt{concat}\,\mathbf{u} \leq_s \texttt{concat}\,\mathbf{v}$ implies $\mathbf{u} \leq_s \mathbf{v}$) for any $\mathbf{u}, \mathbf{v} \in \texttt{lists}\,C$. Moreover, a non-overlapping code has the following property, which can be seen as a weaker version of a the self-synchronization property of codes (self-synchronizing codes are also called *comma-free codes*, see [3, p. 285]). The property is weaker just because an element of a non-overlapping code can be imprimitive, and hence a nontrivial factor of its own square.

Lemma 28 (non-overlapping.fac-concat-fac) *Let C be a non-overlapping set. Assume that $\mathbf{u}, \mathbf{v} \in \texttt{lists}\,C$ and that $\mathbf{u}$ contains at least two distinct elements of C. If $p \cdot \texttt{concat}\,\mathbf{u} \cdot s = \texttt{concat}\,\mathbf{v}$, then there exists $\mathbf{p}$ and $\mathbf{s}$ such that $\mathbf{p} \cdot \mathbf{u} \cdot \mathbf{s} = \mathbf{v}$, $\texttt{concat}\,\mathbf{p} = p$ and $\texttt{concat}\,\mathbf{s} = s$.*

Proof Let $\mathbf{u}_1 \cdot \mathbf{u}_2 = \mathbf{u}$ be a factorization of $\mathbf{u}$ into nonempty lists such that $\texttt{last}\,\mathbf{u}_1 \neq \texttt{hd}\,\mathbf{u}_2$. Then $\texttt{last}(\texttt{concat}\,\mathbf{u}_1) \neq \texttt{hd}\,(\texttt{concat}\,\mathbf{u}_2)$ since C is non-overlapping. The non-overlapping property now implies that the edge between $\texttt{concat}\,\mathbf{u}_1$ and $\texttt{concat}\,\mathbf{u}_2$ must correspond to an edge inside $\texttt{concat}\,\mathbf{v}$. That is, there are lists $\mathbf{v}_1$ and $\mathbf{v}_2$ such that

$$p \cdot \texttt{concat}\,\mathbf{u}_1 = \texttt{concat}\,\mathbf{v}_1, \quad \texttt{concat}\,\mathbf{u}_2 \cdot s = \texttt{concat}\,\mathbf{v}_2, \quad \text{and} \quad \mathbf{v}_1 \cdot \mathbf{v}_2 = \mathbf{v}\,.$$

Since C is a bifix code, we deduce that $\mathbf{u}_1$ is suffix of $\mathbf{v}_1$, and $\mathbf{u}_2$ is a prefix of $\mathbf{v}_2$, which concludes the proof. □

A non-overlapping set forms a primitivity-preserving set of words, which is stated as the next theorem.

Theorem 29 (non-overlapping.prim-morph) *Let C be a non-overlapping set. Let $\mathbf{w} \in \texttt{lists}\,C$ be a primitive list of length at least 2. Then $\texttt{concat}\,\mathbf{w}$ is primitive.*

Proof Assume that $\texttt{concat}\,\mathbf{w}$ is not primitive. Hence there are k and z such that $z^k = \texttt{concat}\,\mathbf{w}$ with $k \geq 2$. It follows that $z \cdot \texttt{concat}\,\mathbf{w} \cdot z^{k-1} = \texttt{concat}(\mathbf{w} \cdot \mathbf{w})$. As $\mathbf{w}$ is primitive and of length at least 2, it contains two distinct elements of C. Hence, since C is non-overlapping, by Lemma 28, there exists $\mathbf{v}$ such that $\mathbf{v} \leq_p \mathbf{w} \cdot \mathbf{w}$ and $\texttt{concat}\,\mathbf{v} = z$. Therefore, $\mathbf{v}^k \in \texttt{lists}\,C$. As $\texttt{concat}(\mathbf{v}^k) = \texttt{concat}\,\mathbf{w}$, we conclude that $\mathbf{v}^k = \mathbf{w}$, which is a contradiction since $\mathbf{w}$ is primitive. □

The previous lemma implies that the morphism $\mathcal{R}$ has the required property (∗). This conclusion is straightforward but slightly technical. We use it as an opportunity to illustrate several properties of morphisms on lists. We assume that x and y are of type `'a list`. Hence $\mathbf{w}$ is of type `'a list list` and $\mathcal{R}$ is a morphism of type `'a list list` $\Rightarrow$ `'a list list`. The morphism $\mathcal{R}$ is defined by its *core* mapping $\mathcal{R}^{\mathcal{C}}$ of type `'a list` $\Rightarrow$ `'a list list` which maps $x \mapsto [\rho(x)]^{e_x}$ and $y \mapsto [\rho(y)]^{e_y}$. The relation between $\mathcal{R}$ and $\mathcal{R}^{\mathcal{C}}$ is given by

$$\mathcal{R}\,\mathbf{w} = \texttt{concat}\left(\texttt{map}\ \mathcal{R}^{\mathcal{C}}\ \mathbf{w}\right).$$

Since $\mathcal{R}^{\mathcal{C}}\ x \neq \mathcal{R}^{\mathcal{C}}\ y$, it is trivial that map $\mathcal{R}^{\mathcal{C}}\ \mathbf{w}$ (of type `'a list list list`) is primitive if and only if $\mathbf{w}$ (of type `'a list list`) is primitive, for any $\mathbf{w} \in \texttt{lists}\,\{x, y\}$. Using the above example $x = abab$ and $y = aa$, and $\mathbf{w} = [x, y, x]$ we have

$$\mathcal{R}^{\mathcal{C}}\ x = [ab, ab], \quad \mathcal{R}^{\mathcal{C}}\ y = [a, a], \quad \texttt{map}\ \mathcal{R}^{\mathcal{C}}\ \mathbf{w} = [[ab, ab], [a, a], [ab, ab]]\,,$$
$$\mathcal{R}\,\mathbf{w} = \texttt{concat}\,(\texttt{map}\ \mathcal{R}\,\mathbf{w}) = [ab, ab, a, a, ab, ab].$$

Since x and y do not commute, we have $\rho(x) \neq \rho(y)$. Hence $\{[\rho(x)]^{e_x}, [\rho(y)]^{e_y}\}$ is a non-overlapping set. Theorem 29 now implies that $\mathcal{R}\,\mathbf{w}$ is primitive if $\mathbf{w}$ is primitive of length at least two as required in $(*)$.

Consequently, the only missing hypothesis preventing the use of Lemma 26 is $|y| \leq |x|$ since it may happen that $|\rho(x)| < |\rho(y)|$. In order to solve this difficulty, we shall ignore for a while the length difference between x and y, and obtain the following intermediate lemma.

Lemma 30 (bin-imprim-both-squares, bin-imprim-both-squares-prim) *Let* $B = \{x, y\}$ *be a binary code, and let* $\mathbf{w} \in \texttt{lists}\ B$ *be a primitive list such that* $\texttt{concat}\,\mathbf{w}$ *is imprimitive. Then* $\mathbf{w}$ *cannot contain both* $[x, x]$ *and* $[y, y]$ *as cyclic factors.*

Proof Assume that $\mathbf{w}$ contains both $[x, x]$ and $[y, y]$ as cyclic factors.

Consider the list $\mathcal{R}\,\mathbf{w}$ and the code $\{\rho(x), \rho(y)\}$. Since $\mathcal{R}\,\mathbf{w}$ contains both $[\rho(x), \rho(x)]$ and $[\rho(y), \rho(y)]$, Lemma 26 implies that $\mathcal{R}\,\mathbf{w}$ is conjugate either with the list $[\rho(x), \rho(x), \rho(y)]$ or with $[\rho(y), \rho(y), \rho(x)]$, which is a contradiction with the assumed presence of both squares. $\square$

7.3 Concluding the Proof by Gluing

It remains to deal with the existence of squares. We use an idea that is our main innovation with respect to the proof from [1], and that contributes significantly to the reduction of the length of the proof, and also to its increased clarity. Let $\mathbf{w}$ be a list over a set of words X. The idea is to choose one of the words, say $u \in X$, and to concatenate (or "glue") blocks of u's to words following them. For example, if $\mathbf{w} = [u, v, u, u, z, u, z]$, then the resulting list is $[uv, uuz, uz]$. This procedure is in the general case well defined on lists whose last "letter" is not the chosen one and it leads to a new alphabet $\{u^i \cdot v \mid v \neq u\}$, which is a code if and only if X is. This idea is used in an elegant proof of the Graph Lemma [2, 14]. Consider the binary case, which is of interest here. If $\mathbf{w}$ does not contain a square of some letter, say $[x, x]$, then the new code is again binary, namely $\{x \cdot y, y\}$. Moreover, the resulting glued list $\mathbf{w}'$ has the same concatenation, and it is primitive if (and only if) $\mathbf{w}$ is. Note that gluing is in this case closely related to the Nielsen transformation $y \mapsto x^{-1}y$ known from the theory of automorphisms of free groups.

Induction on $|\mathbf{w}|$ now easily leads to the proof of Theorem 25.

Proof of Theorem 25 If $\mathbf{w}$ contains y at most once, then we are left with the equation $x^j \cdot y = z^\ell$, $\ell \geq 2$. The equality $j = 2$ follows from the Periodicity Lemma 1, see Case 2 in the proof of Theorem 8.

Assume for contradiction that y occurs at least twice in $\mathbf{w}$. Lemma 30 implies that at least one square, $[x, x]$ or $[y, y]$ is missing as a cyclic factor. Let $\{x', y'\} = \{x, y\}$ be such that $[x', x']$ is not a cyclic factor of $\mathbf{w}$. We can therefore perform the gluing operation, and obtain a new, strictly shorter list $\mathbf{w}' \in \texttt{lists}\,\{x' \cdot y', y'\}$. The longer element $x' \cdot y'$ occurs at least twice in $\mathbf{w}'$, since the number of its occurrences in $\mathbf{w}'$ is the same as the number of occurrences of x' in $\mathbf{w}$, the latter list containing both words at least twice by assumption. Moreover, $\mathbf{w}'$ is primitive, and concat $\mathbf{w}' =$ concat $\mathbf{w}$ is imprimitive. Therefore, by induction on $|\mathbf{w}|$, we have $\mathbf{w}' \sim [x' \cdot y', x' \cdot y', y']$. In order to show that this is not possible we can successfully reuse the lemma `imprim-ext-suf-comm` mentioned in the proof of Lemma 9, this time for $u = x'y'x'$ and $v = y'$. The words u and v do not commute because x' and y' do not commute. Since uv is imprimitive, the word $uvv \sim$ concat $\mathbf{w}'$ is primitive. □

This also completes the proof of our main goal, Theorem 7.

8 Additional Notes on the Formalization

The formalization is implemented in the proof assistant Isabelle/HOL [17, 23]. It is a part of a larger combinatorics on words formalization project. The project's most recent version is published at GitLab [11] while the presented version is archived [12]. The formalization of the presented results relies on the project's backbone session, called *CoW*, a version of which is also available in the Archive of Formal Proofs [10]. An overview of this session is available in [14]. The formalization itself is available in the Archive of Formal Proofs as a separate entry [13]. The backbone session covers all basics concepts used in this article, including the Periodicity Lemma 1, and many more elementary concepts of combinatorics on words. The concept of gluing used in Sect. 7.3 is covered by another session of the project, *Graph Lemma*, and its theory `Glued-Codes`.

The main results of this article are in a separate session in two dedicated theories: `Binary-Square-Interpretation` and `Binary-Code-Imprimitive`. The first contains lemmas and locales dealing with $\{x, y\}$-interpretation of the square xx (for $|y| \leq |x|$), culminating in Theorem 21. The latter contains Theorems 7 and 25. A third theory `Binary-Imprimitive-Decision` covers Lemma 10 and Example 11. However, the formalized proof of Lemma 10 is an alternative simple proof independent of the parametric solution Theorem 8.

8.1 Formalization Highlights

Let us give a few concrete highlights of the formalization. We start by showing selected statements and a definition. The main result, Theorem 7, is formalized as follows:

theorem bin-imprim-code: **assumes** x · y ≠ y · x **and** ws ∈ lists {x,y} **and**
  2 ≤ |ws| **and** primitive ws **and** ¬ primitive (concat ws)
obtains j k **where** 1 ≤ j **and** 1 ≤ k **and** j = 1 ∨ k = 1
  ⋀ws. ws ∈ lists {x,y} ⟹ 2 ≤ |ws| ⟹
    (primitive ws ∧ ¬ primitive (concat ws) ⟷ ws ∼ [x]$^{@}$j · [y]$^{@}$k) **and**
  |y| ≤ |x| ⟹ 2 ≤ j ⟹ j = 2 ∧ primitive x ∧ primitive y **and**
  |y| ≤ |x| ⟹ 2 ≤ k ⟹ j = 1 ∧ primitive x

Definition 14 is not formalized directly as a definition, but as two locales:

```
proof(cases)
  case 1
  then show ?thesis
    using LS-unique-same
      assms(1, 4–8) by blast
next
  case 2
  then show ?thesis
 using LS-unique-same[reversed]
      assms(1, 3, 5–8) by blast
```

(a) Using the `reversed` attribute to solve symmetric cases.

```
have primitive [x,x,y]
  using ‹x ≠ y›
 by primitivity-inspection

from ‹|ws| = 3› ‹ws ∈ lists {x,y}›
‹x ≠ y› ‹[x, x] ≤f ws · ws›
‹[y, y] ≤f ws · ws›
 show False
 by list-inspection simp-all

from ‹p · t · s = t · t · p›
have p · t = t · p
  by mismatch
```

(b) Methods `primitivity-inspection`, `list-inspection` and `mismatch`.

Fig. 8 Highlights from the formalization in Isabelle/HOL

```
locale square-interp =
 fixes x y p s ws
 assumes
  prim-x: primitive x and prim-y: primitive y and
  y-le-x: |y| ≤ |x| and
  ws-lists: ws ∈ lists {x,y} and
  nconjug: ¬ x ∼ y and
  disjoint:  ⋀ p1 p2. p1 ≤p [x,x] ⟹ p2 ≤p ws ⟹ p · concat p1 ≠ concat p2
and
  interp: p (x·x) s ∼𝓘 ws

locale square-interp-ext = square-interp +
 assumes p-extend: ∃ pe. pe ∈ ⟨{x,y}⟩ ∧ p ≤s pe and
  s-extend: ∃ se. se ∈ ⟨{x,y}⟩ ∧ s ≤p se
```

Theorem 21 is then stated within the last locale:

```
theorem sq-ext-interp: ws = [x, y, x] s · p = y p · x = x · s
```

The next highlight is the usage of the `reversed` attribute — very useful tool, which is part of the CoW session. The attribute produces a symmetrical fact where the symmetry is induced by the mapping **rev**, i.e., the mapping which reverses the order of elements in a list. For instance, the fact stating that if p is a prefix of v, then p a prefix of $v \cdot w$, is transformed by the reversed attribute into the fact saying that if s is a suffix of v, then s is a suffix of $w \cdot v$. The attribute is based on rewriting and relies on ad hoc defined (possibly conditional) rules which induce the symmetry. In the example, the main reversal rule is

```
(rev u ≤p rev v) = u ≤s v.
```

The attribute is used frequently in the present formalization. For instance, Fig. 8 shows the formalization of the proof of Cases 1 and 2 of Theorem 9. Namely, the proof of Case 2 is smoothly deduced from the lemma that deals with Case 1, avoiding writing down the same proof again up to symmetry.

The third highlight of the formalization is the use of simple but useful proof methods. The first method, called `primitivity-inspection`, is able to show primitivity or imprimitivity of a given word.

Another method named `list-inspection` is used to deal with claims that consist of straightforward verification of some property for a set of words given by their length and alphabet. For instance, this method painlessly provides the case analysis needed in the proof of lemma `bin-imprim-both-squares-prim`. The method automatically divides the goal into eight (relatively easy) subgoals corresponding to eight possible words. This removes the tedious verification that the case analysis is complete, which is typically rather implicit in human proofs.

The last method we want to mention is `mismatch`. It is designed to prove that two words commute using the decoding delay property of a binary code explained in Sect. 2.2.4. Namely, if a product of words from $\{x, y\}$ starting with x shares a prefix of length at least $|xy|$ with another product of words from $\{x, y\}$, this time starting with y, then x and y commute. Examples of usage of the attribute `reversed` and all three methods are given in Fig. 8.

We conclude with an example, illustrating one of the main virtues of a good formalization, namely identification of auxiliary claims, and their independent formulation in a general form. As pointed out in the introduction, we see this aspect as our main contribution to the topic of the present paper. Consider the lemma `imprim-ext-suf-comm` mentioned twice above (see p. 12 and p. 28):

Lemma 31 (imprim-ext-suf-comm) *Let both $u \cdot v$ and $u \cdot v \cdot v$ be imprimitive. Then u and v commute.*

Proof Since $u \cdot v$ and $u \cdot v \cdot v$ are imprimitive, also $v \cdot u$ and $v \cdot u \cdot v$ are imprimitive. Let z_1 be the primitive root of $v \cdot u$ and z_2 the periodic root of $v \cdot u \cdot v$. Note that z_1 is a periodic root of $v \cdot u \cdot v$. Hence $z_1 = z_2$ by the Periodicity Lemma 1. Indeed, both z_1 and z_2 are periodic roots of $v \cdot u \cdot v$ which is of length at least $|z_1| + |z_2|$. □

In the formalization, this elegant proof is divided into several claims. First, `imprim-ext-suf-comm` is proved as a consequence of `imprim-ext-pref-comm` which claims that u and v commute if $v \cdot u$ and $v \cdot u \cdot v$ are both imprimitive. The lemma `imprim-ext-pref-comm`, in turn, is proved using `per-le-prim-iff`:

Lemma 32 (per-le-prim-iff) *Let u be of length at least twice of the length of its periodic root r. Then u is imprimitive if and only if it commutes with p.*

The latter lemma is a specific application of the weak version of the Periodicity Lemma 1 as explained in the above proof of `imprim-ext-suf-comm`. We want to compare this approach with the proof by Mitrana [22, Theorem 5, p. 278], within which a claim equivalent to our `imprim-ext-suf-comm` is proved ad hoc for two particular words. Consequently, the fact, interesting in itself, cannot be reused, it makes the proof in which it is included harder to parse, and, moreover, it is proved using particular properties of the two words, which are not necessary for the claim (without making the proof easier at that).

8.2 Contribution to the Main Project

As mentioned above, the two dedicated theories containing the formalization of the main results of this article rely on backbone sessions of the project of formalization of combinatorics on words. In turn, the formalization presented in this paper led to an expansion of those backbone sessions. Let us briefly list the main components of that expansion.

- extension of the reversal attribute in `Reversal-Symmetry.thy` to work with lists of lists;
- substantial expansion of the elementary theory `Equations-Basic.thy` which provides auxiliary lemmas and definitions related to word equations;
- expansion of the elementary theory `Lyndon-Schutzenberger.thy` by the parametric solution of the equation $x^j y^k = z^\ell$, specifically Theorem 8 and Lemma 9;
- substantial expansion of existing support for the idea of gluing as mentioned in Sect. 7 into a separate theory called `Glued-Codes.thy` (which is part of the session `CoW-Graph-Lemma`);
- locale formalizing non-overlapping sets (Sect. 7.2).

9 Conclusion

The results presented in this paper have been known in some form since 1985 [1]. Nevertheless, their negligible use in literature proves that they have not been fully absorbed by the community of combinatorics on words. Our own experience indicates that this is so at least partly due to the complex nature and not sufficiently clear structure of the proof. Our effort is an example of the crucial role a formalization can play in approaching this kind of results. The formalization enforces a better proof structure, fills steps based on vague intuition with clearly formulated lemmas, facilitates natural generalizations and allows for transparent and systematic reuse of key ideas.

A broader context of the result we present in this paper are relations between small sets of words, or, seen from another perspective, solutions of simple word equations [15]. As it is often the case in combinatorics, and as the present paper shows, simply formulated problems of this kind can be very difficult. The work described in this paper is a natural starting point for exploring further problems from this area. A concrete example we have in mind, which served as a motivation for the present work, is the classification of binary equality words. This is a task still significantly more complex, in which a research unsupported by formalization may easily reach the borderline of what is humanly feasible [8].

Acknowledgements The authors acknowledge support by the Czech Science Foundation Grant GAČR 20-20621S.

Funding Open access publishing supported by the National Technical Library in Prague.

References

1. Barbin-Le Rest, E., Le Rest, M.: Sur la combinatoire des codes à deux mots. Theoret. Comput. Sci. **41**, 61–80 (1985)
2. Berstel, J., Perrin, D., Perrot, J.F., Restivo, A.: Sur le théorème du défaut. J. Algebra **60**(1), 169–180 (1979). https://doi.org/10.1016/0021-8693(79)90113-3

3. Berstel, J., Perrin, D., Reutenauer, C.: Codes and Automata. Cambridge University Press, Cambridge (2010)
4. Budkina, L.G., Markov, A.A.: F-semigroups with three generators. Mat. Zametki **14**, 267–277 (1973). Translated from Mat. Zametki **14**(2) (1973) 267–277
5. Choffrut, C., Karhumäki, J.: Handbook of formal languages, vol. 1, pp. 329–438. Springer, Berlin (1997). Chap. Combinatorics of Words. https://www.springer.com/gp/book/9783642638633
6. Crochemore, M.: Sharp characterizations of squarefree morphisms. Theoret. Comput. Sci. **18**(2), 221–226 (1982). https://doi.org/10.1016/0304-3975(82)90023-8
7. Fine, N.J., Wilf, H.S.: Uniqueness theorems for periodic functions. Proc. Am. Math. Soc. **16**(1), 109 (1965). https://doi.org/10.1090/S0002-9939-1965-0174934-9
8. Hadravová, J., Holub, Š: Large simple binary equality words. Int. J. Found. Comput. Sci. **23**(6), 1385–1403 (2012). https://doi.org/10.1142/S0129054112500207
9. Harju, T., Nowotka, D.: On the independence of equations in three variables. Theoret. Comput. Sci. **307**(1), 139–172 (2003). https://doi.org/10.1016/S0304-3975(03)00098-7
10. Holub, Š., Raška, M., Starosta, Š.: Combinatorics on words basics. Archive of Formal Proofs (2021). https://isa-afp.org/entries/Combinatorics_Words.html, Formal proof development
11. Holub, Š., Raška, M., Starosta, Š.: Combinatorics on words formalized. GitLab. https://gitlab.com/formalcow/combinatorics-on-words-formalized (2023)
12. Holub, Š., Raška, M., Starosta, Š.: Combinatorics on words formalized. https://doi.org/10.5281/zenodo.7897462
13. Holub, Š., Raška, M.: Binary codes that do not preserve primitivity. Archive of Formal Proofs (2023). https://isa-afp.org/entries/Binary_Code_Imprimitive.html, Formal proof development
14. Holub, Š., Starosta, Š.: Formalization of basic combinatorics on words. In: Cohen, L., Kaliszyk, C. (eds.) 12th International Conference on Interactive Theorem Proving (ITP 2021). Leibniz International Proceedings in Informatics (LIPIcs), vol. 193, pp. 22–12217. Schloss Dagstuhl—Leibniz-Zentrum für Informatik, Dagstuhl, Germany (2021). https://doi.org/10.4230/LIPIcs.ITP.2021.22 . https://drops.dagstuhl.de/opus/volltexte/2021/13917
15. Holub, Š.: Commutation and beyond (extended abstract). In: Combinatorics on Words. Lecture Notes in Comput. Sci., vol. 10432, pp. 1–5. Springer, Cham (2017). https://doi.org/10.1007/3-540-45005-X . https://doi-org.ezproxy.is.cuni.cz
16. Holub, Š, Raška, M., Starosta, Š: Binary codes that do not preserve primitivity. In: Blanchette, J., Kovács, L., Pattinson, D. (eds.) Automated Reasoning, pp. 369–387. Springer, Cham (2022)
17. Isabelle. https://isabelle.in.tum.de
18. Lentin, A., Schützenberger, M.-P.: A combinatorial problem in the theory of free monoids. In: Combinatorial Mathematics and Its Applications (Proc. Conf., Univ. North Carolina), pp. 128–144. Univ. North Carolina Press, Chapel Hill, N.C. (1969)
19. Lothaire, M.: Combinatorics on Words, p. 238. Cambridge Mathematical Library. Cambridge University Press, Cambridge (1997). https://doi.org/10.1017/CBO9780511566097
20. Lyndon, R.C., Schützenberger, M.-P.: The equation $a^m = b^n c^p$ in a free group. Mich. Math. J. **9**(4), 289–298 (1962). https://doi.org/10.1307/mmj/1028998766
21. Maňuch, J.: Defect effect of bi-infinite words in the two-element case. Discret. Math. Theoret. Comput. Sci. **4**(2), 273–290 (2001)
22. Mitrana, V.: Primitive morphisms. Inf. Process. Lett. **64**(6), 277–281 (1997). https://doi.org/10.1016/s0020-0190(97)00178-6
23. Nipkow, T., Paulson, L.C., Wenzel, M.: Isabelle/HOL—A Proof Assistant for Higher-Order Logic. Lecture Notes in Computer Science, vol. 2283. Springer, Heidelberg (2002)
24. Shallit, J.: A Second Course in Formal Languages and Automata Theory, 1st edn. Cambridge University Press, Cambridge (2008)
25. Spehner, J.C.: Quelques problèmes d'extension, de conjugaison et de présentation des sous-monoïdes d'un monoïde libre. PhD thesis, Université Paris VII, Paris (1976)