Theoretical Computer Science 720 (2018) 55–63

Contents lists available at ScienceDirect

Theoretical Computer Science

www.elsevier.com/locate/tcs

On semi-perfect de Bruijn words

Damian Repke, Wojciech Rytter *,[1]

Department of Mathematics, Computer Science and Mechanics, Warsaw University, Warsaw, Poland

ARTICLE INFO

Article history:
Received 14 February 2017
Received in revised form 14 November 2017
Accepted 5 February 2018
Available online 10 February 2018
Communicated by M.J. Golin

Keywords:
de Bruijn word
Semi-perfect
Straight-line program
Algorithm

ABSTRACT

We show an application of Lempel's recursive construction of De Bruijn words to the generation of binary words having many *factor-rich prefixes*. A binary word is said to be *factor-rich* iff it has the largest number of distinct factors among binary words with the same length. A linear de Bruijn word of rank n is a shortest word containing (as a factor) exactly once each binary word of length n. It is factor-rich and its length equals $\Delta_n = 2^n + n - 1$. We construct for each n a binary linear de Bruijn word of rank n which is *semi-perfect* in the following sense: each of its prefixes of length $m > \Delta_{n-1}$ is factor-rich. The number Δ_{n-1} is the best possible (for $n > 2$ there is no linear binary de Bruijn word with factor-rich prefix of length $m = \Delta_{n-1}$). We show an efficient algorithm constructing compact description of binary semi-perfect de Bruijn words.

1. Introduction

A binary word is called *factor-rich* iff it has the largest number of distinct factors (also called subwords) among binary words with the same length. For example 00110 is factor-rich, but 01011 is not, since the first one contains one more factor (the word 00) and is of the same length. Factor-rich words are closely related to de Bruijn words. A construction of a factor-rich word of a given length was given in [4] using graph theoretic properties of de Bruijn graphs. We show here an alternative algorithm. Our main construction is syntactic and it is quite simple *"how it works"*, however correctness proof of *"why it works"* is rather tedious.

By an n-word we mean a binary word of length n. A circular de Bruijn binary word of rank n is a 2^n-word containing in the cyclic sense each n-word exactly once. The length should be 2^n since the word should in a cyclic sense contain each of 2^n distinct binary words of length n. For example $0011 \in \mathbf{DB}_2$. Denote by $\mathbf{DB}_n$ the set of binary circular de Bruijn words of rank n.

A linear de Bruijn word of rank n is a shortest binary word containing each n-word exactly once as a linear (standard) word. Denote by $\mathrm{Lin}\mathbf{DB}_n$ the set of binary linear de Bruijn words of rank n. For example $00110 \in \mathrm{Lin}\mathbf{DB}_2$.

Let $\Delta_n = 2^n + n - 1$. Denote by $|w|$ the length of w. The following properties of circular and linear de Bruijn words are well known.

* Corresponding author.
E-mail address: rytter@mimuw.edu.pl (W. Rytter).

[1] The author is supported by grant no. NCN2014/13/B/ST6/00770 of the National Science Centre.

https://doi.org/10.1016/j.tcs.2018.02.008

Observation 1.

(a) Each linear de Bruijn word of rank n is of the form $u \cdot v$, where $u \in \mathbf{DB}_n$ and v is a prefix of u of length $n-1$, consequently $|u \cdot v| = \Delta_n$.
(b) Assume w is a binary word of length Δ_n. Then $w \in \mathrm{Lin}\mathbf{DB}_n$ iff w is factor-rich.

The following characterization of factor rich words is from [4].

Lemma 1. *Assume w is a binary word of length m, for $\Delta_{n-1} < |w| \leq \Delta_n$. Then w is factor-rich iff w contains all $(n-1)$-words and does not contain twice any n-word.*

We say that a word $w \in \mathrm{Lin}\mathbf{DB}_n$ is *semi-perfect* iff each prefix of w of length at least $\Delta_{n-1}+1$ is factor-rich.
As an immediate consequence of Lemma 1 we obtain the following fact.

Corollary 1. *A de Bruijn word w of rank n is semi-perfect iff its prefix of length $\Delta_{n-1}+1$ contains, as a factor, each word of length $n-1$.*

Example 1. The linear de Bruijn word 0111000101 of rank 3 is semi-perfect, all its prefixes of length at least Δ_2+1 are factor-rich. Its prefix 01110 of length Δ_2 is not factor-rich since it does not contain the 2-word 00.

The lexicographically first linear de Bruijn word 0001011100 is not semi-perfect, its prefix 000101 of size Δ_2+1 is not factor-rich (does not contain 11).

In the definition of semi-perfect de Bruijn words we cannot require the prefix of length Δ_{n-1} to be factor-rich, for $n > 2$. We show the proof of this fact for completeness, a similar property was also observed in [1].

Lemma 2. *For $n > 2$ a linear binary de Bruijn word of rank n cannot have a factor-rich prefix of length Δ_{n-1}.*

Proof. The proof is by contradiction. Assume that for $n > 2$ there are two binary factor-rich words $u \cdot v$, u of lengths Δ_n, Δ_{n-1}.

Due to Observation 1 we have: $u \in \mathrm{Lin}\mathbf{DB}_{n-1}$, $u \cdot v \in \mathrm{Lin}\mathbf{DB}_n$.

Assume u starts with 0 (opposite case is symmetric). Then u does not contain 1^n and 1^{n-1} cannot be the suffix of u since the suffix of u of length $n-2$ (equal to its prefix) contains 0. Consequently $01^{n-1}0$ is a factor of u (as well as of $u \cdot v$). Then $u \cdot v$ contains $01^{n-1}0$ and 01^n. Now we have a contradiction, since $u \cdot v$ contains twice the word 01^{n-1}. This completes the proof. □

It is shown in [1] that for each $n > 1$ there is a binary word $w \in \mathbf{DB}_{n-1}$ which is a prefix of a word in $\mathbf{DB}_{n+1}$. However this does not help in constructing semi-perfect binary words. In case of ternary alphabet it is even possible to construct de Bruijn words in which *every* prefix is factor-rich. Surprisingly the case of binary alphabet is much harder than that of larger alphabets.

Our construction gives a compact description of semi-perfect words in terms of so called straight-line programs (also called *grammar-based compression*). A *Straight-line program* is a sequence of assignment statements which describe values of new variables in terms of expressions involving already defined variables. Such statements use a restricted set of operations, and initial variables correspond to constants (in our case words of constant length). Such type of compact representation is a generalization of Lempel–Ziv compression and is often used to get compact description of large well-structured objects. We consider here representation of linear de Bruijn words of rank n, which have length exponential in n, due to Observation 1, in terms of straight-line programs of size $O(n)$. Our compact descriptions are using the operations of concatenation, cyclic shifts, special operation Ψ and taking prefixes/suffixes.

Our results. We show that there is a family of *semi-perfect* linear binary de Bruijn words which can be computed by a simple recursive linear-time algorithm. Moreover such semi-perfect words have compact representation.

Previously factor-rich binary words were constructed using de Bruijn graphs, and were not very closely related to de Bruijn words, unless the length was Δ_n, for some n. Here we show much closer relation of these words to de Bruijn words by constructing linear de Bruijn words which are simple generators of factor-rich words of any length. The construction is syntactic, omitting de Bruijn graphs. Usually words with very compact description have small number of distinct factors. Here we have a compact description and simultaneously the largest possible number of factors. The description applies, as a main tool, a function related to Lempel's homomorphism, see [3].

2. Preliminaries

We assume now that all considered de Bruijn words are binary. We count positions in a word starting at 1. We use $\cdot$ for concatenation of words, but use juxtaposition of a letter and a word (as in aw, for a letter a and a word w).

Denote by $\mathrm{Bin}(n)$ the set of all n-words. For $w \in \mathrm{Lin}\mathbf{DB}_n$ denote by $\mathcal{F}_n(w)$ the set of all n-words that are factors of w. A word is a circular factor of w iff it is a factor of ww. Denote by $\mathcal{C}_n(w)$ the set of all n-words that are circular factors of w.

Example 2. $\mathcal{F}_2(abc) = \{ab, bc\}$, $\mathcal{C}_2(abc) = \{ab, bc, ca\}$.

If $|u| \geq k$ denote by $pref_k(u), suf_k(u)$ the prefix, respectively suffix of u of length k. For $y \in \mathbf{DB}_n$ denote:

$$\mathrm{linear}(y) = y \cdot pref_{n-1}(y).$$

Observe that if $y \in \mathbf{DB}_n$ then $\mathcal{C}_n(y) = \mathcal{F}_n(\mathrm{linear}(y))$.

We introduce three basic operations $\oplus$, $\otimes$, Ψ on binary words of length 2^n.

2.1. Operation $\oplus$

Denote by $SHIFT(x, \gamma)$ the cyclic shift of the word x having the suffix equals γ, assuming γ occurs exactly once as a circular factor of x.

Example. $SHIFT(001011101, 011) = 101001011$.

For two words $x, y \in \mathrm{Bin}(2^n)$, such that $suf_n(x)$ is a circular factor of y, define

$$x \oplus y = x \cdot SHIFT(y, suf_n(x)).$$

For example $10110\underline{110} \oplus 0\underline{110}1010 = 10110\underline{110}\,10100\underline{110}$, where $suf_3(x)$ is underlined.

Observation 2. Assume $n \geq 1$ and $x, y \in \mathrm{Bin}(2^n)$. Then

(a) if x, y have a common suffix of length n or a common prefix of length n then $\mathcal{C}_{n+1}(x \cdot y) = \mathcal{C}_{n+1}(x) \cup \mathcal{C}_{n+1}(y)$.
(b) if $\mathcal{C}_{n+1}(x) \cup \mathcal{C}_{n+1}(y) = \mathrm{Bin}(n+1)$ and $suf_n(x) \in \mathcal{C}_n(y)$ then $x \oplus y \in \mathbf{DB}_{n+1}$

Proof. We start with point (a). Only the case of a common suffix is considered, the case of a common prefix is symmetric.

Observe first that, for two words v, w, if $|w| = n$ and $|v| \geq n$ then

$$\mathcal{C}_{n+1}(v \cdot w) = \mathcal{F}_{n+1}(w \cdot v \cdot w).$$

Assume now that x, y have a common suffix z with $|z| = n$. x, y are of the form $x = x' \cdot z$, $y = y' \cdot z$, where $z' \in \mathrm{Bin}(n)$. We have:

$$\mathcal{C}_{n+1}(x \cdot y) = \mathcal{F}_{n+1}(x \cdot y \cdot x) = \mathcal{F}_{n+1}(x' \cdot z \cdot y' \cdot z \cdot x' \cdot z).$$

Now point (a) follows from equalities

$$\mathcal{F}_{n+1}(x' \cdot z \cdot y' \cdot z \cdot x' \cdot z) = \mathcal{F}_{n+1}(z \cdot y' \cdot z) \cup \mathcal{F}_{n+1}(z \cdot x' \cdot z)$$
$$= \mathcal{C}_{n+1}(y' \cdot z) \cup \mathcal{C}_{n+1}(x' \cdot z) = \mathcal{C}_{n+1}(x) \cup \mathcal{C}_{n+1}(y).$$

This completes the proof of point (a).

In point (b) we can assume that x, y have the same suffix of length n, since at the start of the operation $\oplus$ they are changed to satisfy it. The sets of circular factors are not changing. Now point (a) can be applied, it gives $\mathcal{C}_{n+1}(x \cdot y) = \mathcal{C}_{n+1}(x) \cup \mathcal{C}_{n+1}(y) = \mathrm{Bin}(n+1)$, since we assumed in this point that $\mathcal{C}_{n+1}(x) \cup \mathcal{C}_{n+1}(y) = \mathrm{Bin}(n+1)$. This completes the proof. □

2.2. Operation $\otimes$

Assume that the length of a maximal block of the same consecutive letters p, for $p \in \{0, 1\}$ in a word x equals k, and such maximal block is unique in x for each letter p. Define operations:

- $add_p(x)$ is the word resulting by adding single bit p in the group p^k
- $rem_p(x)$ is the word resulting by removing a single p in the group p^k

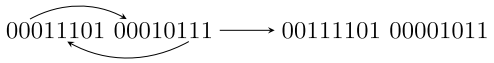

Fig. 1. Graphical illustration of $x \otimes y$.

Example 3. $add_1(0110) = 01110,\ rem_0(00010111) = 0010111.$

We define the following operation $\otimes$ on two circular de Bruijn words x, y of rank n:

If x, y do not start both with 0^n then assume that the operation $\otimes$ starts by shifting x and y cyclically to fulfill this condition. The operation $\otimes$ first moves single zero from the block of 0^n in x to the block of 0^n in y, and moves single 1 from the block of 1^n in y to the block of 1^n in x. Then the resulting words are concatenated, see Fig. 1.

More formally: $x \otimes y = rem_0(add_1(x)) \cdot rem_1(add_0(y))$, assuming that x, y are first shifted to start with maximal blocks of 0's.

2.3. Operation Ψ

The crucial operation is a special algebraic operation $\Psi(w)$ on words, the result of this operation is the binary word v of length $|w|$, where for every $1 \le i \le |w|$:

$$v[i] \ = \ (\, w[1] + w[2] + \ldots w[i]\,) \bmod 2.$$

The operation Ψ was used in the algorithm R, presented in Knuth's 4-th volume, see [2]. It also implicitly appears in Lempel's algorithm for de Bruijn words using homomorphism of de Bruijn graphs, see [3].

Denote by $\bar{x}$ the bitwise negation of the word x.

Example 4. If $w = 0010111011$ then $\Psi(w) = 0011010010$. It is possible that $\overline{\Psi(x)} \neq \Psi(\bar{x})$, e.g. for $x = 01$.

We introduce also an operation $\tau : \mathrm{Bin}(n+1) \to \mathrm{Bin}(n)$.

For an $(n+1)$-word w define $\tau(w)$ as an n-word such that for every $1 \le i \le n$:

$$\tau(w)[i] \ = \ (\, w[i] + w[i+1]\,) \bmod 2.$$

Example 5. $\tau(1101) = \tau(\overline{1101}) = 011.$

Lemma 3.

(a) $\tau(w) = \tau(\bar{w})$.
(b) $\tau(w) \ = \ \tau(w') \ \Leftrightarrow \ (\, w' = w \text{ or } w' = \bar{w}\,)$.
(c) *Assume $|x| > n$ and $\Psi(x)$ ends with 0. Then*

$$\mathcal{C}_n(x) = \{\tau(w) : w \in \mathcal{C}_{n+1}(\Psi(x))\}.$$

(d) *If $x \in \mathbf{DB}_n$, and $w \in \mathrm{Bin}(n+1)$ has a circular occurrence starting at some position in $\Psi(x)$, then $\tau(w)$ is a circular n-word starting at (cyclically) next position in x.*

Proof. We show separately each point.

Point (a).

This point follows from equality

$$(\, w[i] + w[i+1]\,) \bmod 2 \ = \ (\, \overline{w[i]} + \overline{w[i+1]}\,) \bmod 2.$$

Point (b).

The implication $\Leftarrow$ is a reformulation of point (a). We show now that

$$\tau(w) \ = \ \tau(w') \ \Rightarrow \ (\, w = w' \text{ or } w' = \bar{w}\,).$$

Assume $w = a_1, a_2, \ldots a_m,\ \ w' = b_1, b_2, \ldots b_m$. We have that for $i < m$:

$$(\, a_i + a_{i+1}\,) \bmod 2 \ = \ (\, b_i + b_{i+1}\,) \bmod 2.$$

Consequently:

$$a_i = b_i \Rightarrow a_{i+1} = b_{i+1} \text{ and } a_i = \overline{b_i} \Rightarrow a_{i+1} = \overline{b_{i+1}}.$$

Hence if $a_1 = b_1$ then $w' = w$ else $w' = \overline{w}$. This proves point (b).

Points (c) and (d).

These points are closely related. First we get rid of circularity. Observe that $\Psi(x \cdot x) = \Psi(x) \cdot \Psi(x)$, since $\Psi(x)$ ends with 0 (it is essential).

Hence we can reformulate point (c) in terms of linear factors:

$$\mathcal{F}_n(x \cdot x) = \{\tau(w) : w \in \mathcal{F}_{n+1}(\Psi(x \cdot x))\} \tag{1}$$

Now Equation 1, as well as point (d), follows from the following simple joint property of functions Ψ and τ.

Claim 1. *Assume $m > 1$ and $a_i \in \{0, 1\}$ for $i \le m$. Then*

$$\tau(\Psi(a_1 a_2 a_3 \ldots a_m)) = a_2 a_3 \ldots a_m. \tag{2}$$

(For example: $1100101011 \overset{\Psi}{\rightarrow} 1000110010 \overset{\tau}{\rightarrow} 100101011$.)

Proof. Assume $v = a_1 a_2 \ldots a_m$, $\Psi(v) = b_1 b_2 \ldots b_m$, $\tau(\Psi(v)) = c_1 c_2 \ldots c_{m-1}$.

It is enough to show that $c_i = a_{i+1}$ for $i < m$. We have:

$$c_i = (b_i + b_{i+1}) \bmod 2,\ b_{i+1} = (b_i + a_{i+1}) \bmod 2.$$

Consequently $c_i = (2 \cdot b_i + a_{i+1}) \bmod 2 = a_{i+1}$, as required. This completes the proof of the Equation 2 and of the claim. □

Denote now by $\hat{x}$ the word x with the first letter removed.

We have: $\mathcal{F}_n(x \cdot x) = \mathcal{F}_n(\hat{x} \cdot x)$ since $|x| > n$. Now observe that

$$\{\tau(w) : w \in \mathcal{F}_{n+1}(\Psi(x \cdot x))\} = \mathcal{F}_n(\tau(\Psi(x \cdot x)) = \mathcal{F}_n(\hat{x} \cdot x) = \mathcal{F}_n(x \cdot x).$$

This proves Equation 1 and point (c).

In point (d) we have $x \in \mathbf{DB}_n$, hence the last bit of $\Psi(x)$ is zero. The point (c) applies. Equation 2 implies that $(n+1)$-factors in $\Psi(x \cdot x)$ correspond to n-factors in $x \cdot x$ shifted by one position. The correspondence is given by τ. This completes the proof. □

3. A version of Lempel's recursive algorithm

We use the following basic properties of the operation Ψ.

Observation 3. Assume $x \in \mathbf{DB}_n$ and $y = \Psi(x)$. Then:

(1) All circular $(n+1)$-factors of y are pairwise distinct.
(2) if z is a circular $(n+1)$-factor of y then $\bar{z}$ is not a circular $(n+1)$-factor of y.
(3) $\mathcal{C}_{n+1}(y) \cap \mathcal{C}_{n+1}(\bar{y}) = \emptyset$.

Proof. We prove separately each point.

Point (1) We show it by contradiction. Assume a circular $(n+1)$-word appears twice in $\Psi(x)$. Then the number of $(n+1)$-factors in $\Psi(x)$ is smaller then 2^n (the length of $\Psi(x)$). However according to Lemma 3.c each circular n-factor in x equals $\tau(w)$ for some circular $(n+1)$-word in $\Psi(x)$. Then the number of circular n-factors in x is smaller than 2^n, which contradicts $x \in \mathbf{DB}_n$. This completes the proof of this point.

Point (2) Assume z and $\bar{z}$ are circular factors in $\Psi(x)$. Then $\tau(z)$ and $\tau(\bar{z})$ occur in different places in x, according to Lemma 3.d. However $\tau(z) = \tau(\bar{z})$. Consequently the same circular n-factor occurs twice in x, which again contradicts $x \in \mathbf{DB}_n$.

Point (3) This point is a reformulation of point (2). □

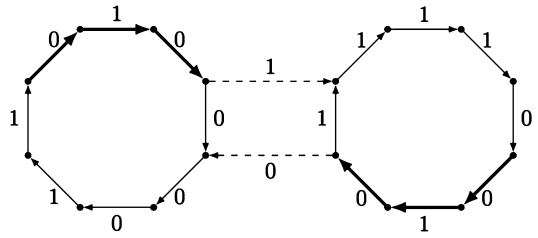

Fig. 2. Graphical interpretation of computing $NEXT(00010111)$. Two cycles correspond to $\Psi(00010111) = 00011010$, $\overline{\Psi(x)} = 11100101$. $SHIFT(11100101, 010) = 11110010$. The operation $\oplus$ applied to these two circular words corresponds to interchanging successors of two nodes (dotted arrows). The result is the circular word $00011010\,11110010 \in \mathbf{DB}_4$. The edges corresponding to the word 010 are shown in bold.

For $x \in \mathbf{DB}_n$ we introduce another basic operation:

Operation $NEXT(x)$
/* A version of Lempel's algorithm */
$x := SHIFT(x, 1^n)$;
return $\Psi(x) \oplus \overline{\Psi(x)}$.

In this operation if x does not end with 1^n then it is first shifted to have 1^n at the end, where n is the largest group of 1's in x.

Example 6. Let $x = 00010111$, we have: $SHIFT(x, 1^3) = x$, $\Psi(x) = 00011010$ (see Fig. 2). $NEXT(x) = 00011\underline{010} \oplus 1110\underline{010}1 = 00011010 \cdot 11110010$.

The recursive algorithm given in the following lemma can be viewed as a syntactic version of Lempel's graph-theoretic algorithm.

Lemma 4. *If* $x \in \mathbf{DB}_n$ *and* $n \geq 2$ *then* $NEXT(x) \in \mathbf{DB}_{n+1}$.

Proof. It is enough to show that words $\Psi(x)$, $\overline{\Psi(x)}$ satisfy the assumptions of Observation 2.b.

Claim 2. *If* $x \in \mathbf{DB}_n$ *then* $\mathcal{C}_{n+1}(\Psi(x)) \cup \mathcal{C}_{n+1}(\overline{\Psi(x)}) = \mathrm{Bin}(n+1)$.

The word x is de Bruijn word of rank n, so we know that $|\mathcal{C}_n(x)| = 2^n$. Let $y = \Psi(x)$. Due to Observation 3.1 $|\mathcal{C}_{n+1}(y)| = 2^n$.

Also, due to Observation 3.3, $\mathcal{C}_{n+1}(y) \cap \mathcal{C}_{n+1}(\bar{y}) = \emptyset$.

Therefore, the union of $\mathcal{C}_{n+1}(\Psi(x)) \cup \mathcal{C}_{n+1}(\overline{\Psi(x)})$ is of size 2^{n+1}. This completes the proof of the claim and of the lemma (due to Observation 2.b). □

4. The main algorithm

An important component of the algorithm is the construction of pairs of de Bruijn words of a given rank n which have the smallest number of common $(n+1)$-factors. Two words x, y in $\mathbf{DB}_n$ are said to be *complementary* (denoted $x \perp y$) iff the set of their common circular factors of length $n+1$ is:

$$Common(n) = \{0^n1,\ 1^n0,\ 01^n,\ 10^n\}.$$

Observe that $Common(n) \subseteq \mathcal{C}_{n+1}(x)$ for each word $x \in \mathbf{DB}_n$.

Example 7. The word 0011 is self-complementary, we have: $0011 \perp 0011$.
The following two distinct words are complementary: $00011101 \perp 00010111$.

We postpone the proof of the following *technical lemma* to the last section.

Lemma 5. [Complementarity Lemma]
If $x, y \in \mathbf{DB}(n)$, $x \perp y$, *and* $n \geq 2$ *then* $\textsc{Next}(x) \perp \overline{\textsc{Next}(y)}$.

We combine the operations $\oplus$, $\otimes$, Ψ (used in the definition of $NEXT$) as components of our main algorithm:

```
function CONSTRUCT(n)
    if n = 1 then return 01
    if n = 2 then return 00110
    x := y := 0011 ( invariant: x ⊥ y )
    for i = 3 to n − 1 do
        x := NEXT(x); y := NEXT(y)‾
        invariant: x ⊥ y
    return linear(x ⊗ y)
```

We have:

$$Construct(3) = 0111000101,\ \ Construct(4) = 0010111100001101001.$$

Two complementary words in $\mathbf{DB}_n$ contain all circular factors of length $n+1$ except:

$$Unused(n) = \{0^{n+1},\ 1^{n+1},\ 01^{n-1}0,\ 10^{n-1}1\}.$$

Similarly, none of words in $\mathbf{DB}_n$ contains any of these four unused $(n+1)$-factors.
We are now ready to show our main result.

Theorem 1.

(a) *The word Construct(n), of length 2^n+n-1, is a semi-perfect linear de Bruijn word of rank n. It can be constructed in $O(2^n)$ time.*
(b) *For $n>2$ the following word is semi-perfect:*

$$\text{linear}(\,NEXT^{n-3}(0011)\otimes(\overline{NEXT})^{n-3}(0011)\,).$$

Proof. The following claim reduces construction of semi-perfect de Bruijn words to the construction of a pair of complementary words.

First we show the following fact:

Claim 3. *Assume $u, w \in \mathbf{DB}(n)$, $u \perp w$ and both start with 0^n. Then $u \otimes w \in \mathbf{DB}_{n+1}$.*

Proof. Let $u' = rem_0(add_1(u))$ and $w' = rem_1(add_0(w))$.

Example 8. For $u = 00010111,\ w = 00011101$ we have:

$$u' = 00101111,\ w' = 00001101,\ u\otimes w = u'\cdot w' = 00101111\,00001101.$$

Note that both u' and w' start with 0^{n-1} and end with 1. Let u'', w'' be the words u', w' with the last letter 1 shifted to the beginning. The word $u''\cdot w''$ is a cyclic shift of $u'\cdot w'$. Hence

$$\mathcal{C}_{n+1}(u'\cdot w') = \mathcal{C}_{n+1}(u''\cdot w''),\ \mathcal{C}_n(u) = \mathcal{C}_n(u''),\ \mathcal{C}_n(w') = \mathcal{C}_n(w'').$$

The words u'' and w'' have the same prefix of length n, so by Observation 2 we have that $\mathcal{C}_{n+1}(u''\cdot w'') = \mathcal{C}_{n+1}(u'')\cup \mathcal{C}_{n+1}(w'')$. Consequently:

$$\begin{aligned}&\mathcal{C}_{n+1}(u\otimes w) = \mathcal{C}_{n+1}(u'\cdot w') = \mathcal{C}_{n+1}(u''\cdot w'')\\ &= \mathcal{C}_{n+1}(u'')\cup\mathcal{C}_{n+1}(w'') = \mathcal{C}_{n+1}(u')\cup\mathcal{C}_{n+1}(w').\end{aligned}$$

Consider now how adding and removing bits change the set of circular $(n+1)$-factors of words $u, w \in \mathbf{DB}_n$. Operation add_p ($p\in\{0,1\}$) just adds extra $(n+1)$-factor p^{n+1}. Moreover, operation rem_p removes factors $\bar{p}p^n$ and $p^n\bar{p}$ and adds new factor $\bar{p}p^{n-1}\bar{p}$. Hence:

$$\mathcal{C}_{n+1}(u') = (\mathcal{C}_{n+1}(u)\cup\{10^{n-1}1,\ 1^{n+1}\}) - \{10^n,\ 0^n1\}$$

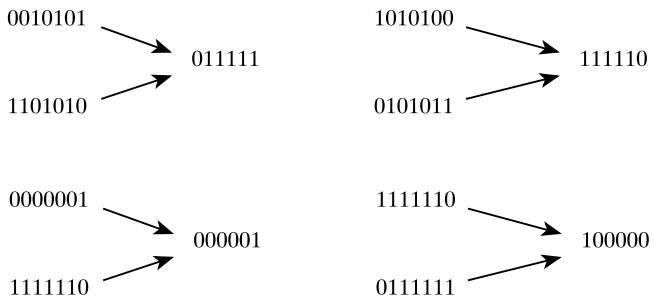

Fig. 3. The arrows correspond to the operation τ. The words $\alpha_6 0$, $\beta_6 1$, $\overline{\alpha_6 0}$, $\overline{\beta_6 1}$ are all words mapped onto {011111, 111110} (first row). $Common(6)$ is mapped onto $\{0^5 1, 10^5\}$ (second row). $\tau^{-1}(Common(5)) = Common(6) \cup \{\alpha_6 0, \beta_6 1, \overline{\alpha_6 0}, \overline{\beta_6 1}\}$.

$$\mathcal{C}_{n+1}(w') = (\mathcal{C}_{n+1}(w) \cup \{01^{n-1}0,\ 0^{n+1}\}) - \{01^n,\ 1^n 0\}$$

Notice that every removed factor belongs to $Common(n)$, so each of them still belongs to the union of $\mathcal{C}_{n+1}(u')$ and $\mathcal{C}_{n+1}(w')$. Furthermore, every element of $Unused(n)$ is added, so finally we have:

$$\mathcal{C}_{n+1}(u \otimes w) = \mathcal{C}_{n+1}(u) \cup \mathcal{C}_{n+1}(w) \cup Unused(n) = \mathrm{Bin}(n+1).$$

Hence $u \otimes w \in \mathbf{DB}_{n+1}$, which proves the claim. □

Claim 4. *Assume $u, w \in \mathbf{DB}(n)$, $u \perp w$ and both start with 0^n. Then* linear$(u \otimes w)$ *is semi-perfect.*

Proof. By Corollary 1 it is enough to show: the prefix of linear$(u \otimes w)$ of length $\Delta_n + 1$ contains, as a factor, each binary word of length n.

Notice that this prefix is of the form $u' \cdot 0^n$ and its set of n-factors is the same as the set of circular n-factors of word $u' \cdot 0$ or any of its shift, e.g. $0 \cdot u' = add_1(u)$. The word u is a de Bruijn word of rank n, so it contains all n-factors.

Obviously, adding a single one in the group 1^n does not remove any of n-factors, so $\mathcal{C}_n(add_1(u)) = \mathcal{F}_n(u' \cdot 0^n) = \mathrm{Bin}(n)$. This completes the proof of the claim. □

Our main result follows now directly from Lemma 5. This completes the whole proof. □

Remark 1. An $O(n)$-sized compact description of $Construct(n)$ can be computed in $O(n)$ time as a straight-line program using operations of concatenation, negation, shifts, prefix/suffix cutting and Ψ using formula from point (b) of the theorem.

5. Proof of complementarity lemma

We will show that the invariant related to complementarity is preserved in the algorithm *Construct*. There are two sequences of words α_n, β_n which play here the key role.

Denote by α_k the word of length k which does not contain 00 or 11 and ends with 0. Let β_k be the word α_k with the first letter negated.

Example 9. $\alpha_4 = 1010,\ \beta_4 = 0010,\ \alpha_5 = 01010,\ \beta_5 = 11010.$

5.1. Properties of the words $\alpha_n,\ \beta_n$

The words α_k, β_k have the following important properties.

Observation 4.

(a) Let $n \geq 2$, $x \in \mathbf{DB}_n$, x ends with 1^n and $u = \Psi(x),\ v = SHIFT(\overline{\Psi(x)}, suf_n(u))$. Then

$$suf_{n+1}(u) = \alpha_{n+1},\ \ suf_{n+1}(u) = \beta_{n+1}.$$

(b) $\tau^{-1}(1^n 0) = \{\alpha_{n+1}0, \overline{\alpha_{n+1}0}\},\ \ \tau^{-1}(01^n) = \{\beta_{n+1}1, \overline{\beta_{n+1}1}\}$ (see Fig. 3).
(c) $\tau^{-1}(Common(n)) = Common(n+1) \cup \{\alpha_{n+1}0, \beta_{n+1}1, \overline{\alpha_{n+1}0}, \overline{\beta_{n+1}1}\}$.

We need a fact which is a straightforward extension of Observation 2.

Observation 5. If $u,\ v \in \mathrm{Bin}(2^n)$ have a common suffix of length n then

$$\mathcal{C}_{n+2}(u \cdot v) = \mathcal{C}_{n+2}(u) \cup \mathcal{C}_{n+2}(v) \cup \{suf_{n+1}(u)b,\ suf_{n+1}(v)a\} - \{suf_{n+1}(u)a,\ suf_{n+1}(v)b\},$$

where $a = u_1,\ b = v_1$.

Lemma 6. *For any de Bruijn word z of rank n ending with 1^n we have:*

$$\mathcal{C}_{n+2}(\textsc{Next}(z)) = X \cup \overline{X} - \{\alpha_{n+1}0,\ \beta_{n+1}1\} \cup \{\alpha_{n+1}1,\ \beta_{n+1}0\},$$

where $X = \mathcal{C}_{n+2}(\Psi(z))$.

Proof. Let $u = \Psi(z),\ v = SHIFT(\bar{u}, suf_n(u))$. Then $NEXT(z) = u \cdot v$. According to Observation 4 we have: $suf_{n+1}(u) = \alpha_{n+1},\ \ suf_{n+1}(v) = \beta_{n+1}$. Also we have $u_1 = 0,\ v_1 = 1$. Now the thesis follows directly from Observation 5. □

5.2. Proof of Lemma 5

We recall the statement of the **complementarity lemma**:

if $x, y \in \mathbf{DB}(n),\ x \perp y, n \geq 2$ and the words x, y end with 1^n,
then $\textsc{Next}(x) \perp \overline{\textsc{Next}(y)}$.

Proof. Let $X = \mathcal{C}_{n+2}(\Psi(x)),\ Y = \mathcal{C}_{n+2}(\Psi(y))$. From Lemma 6 and Observation 5 we know that:

$$(X \cup \overline{X}) \cap (Y \cup \overline{Y}) = Common(n+1) \cup \{\alpha_{n+1}0,\ \beta_{n+1}1,\ \overline{\alpha_{n+1}0},\ \overline{\beta_{n+1}1}\}.$$

Due to Lemma 6 in the operation $NEXT$ the set

$$\{\alpha_{n+1}0,\ \beta_{n+1}1,\ \overline{\alpha_{n+1}0},\ \overline{\beta_{n+1}0}\}$$

is changed into

$$Z_n = \{\alpha_{n+1}1,\ \beta_{n+1}0,\ \overline{\alpha_{n+1}0},\ \overline{\beta_{n+1}1}\}.$$

The words of Z_n are the only common $(n+2)$-factors of $NEXT(x)$ and $NEXT(y)$ besides $Common(n+1)$.

In the operation of changing $NEXT(y)$ into $\overline{NEXT(y)}$ the set Z_n becomes $\overline{Z_n}$. Now the thesis follows from the fact that $Z_n \cap \overline{Z_n} = \emptyset$. We have:

$$Z_n \cap \overline{Z_n} = \{\alpha_{n+1}1,\ \beta_{n+1}0,\ \overline{\alpha_{n+1}0},\ \overline{\beta_{n+1}1}\} \cap \{\alpha_{n+1}0,\ \beta_{n+1}1,\ \overline{\alpha_{n+1}1},\ \overline{\beta_{n+1}0}\} = \emptyset.$$

Consequently the only common circular $(n+2)$-factors of $NEXT(x)$ and $\overline{NEXT(y)}$ are in $Common(n+1)$. Thus $NEXT(x)$ and $\overline{NEXT(y)}$ are complementary.

This completes the proof. □

Example 10. We have:

$$Z_3 = \{10101,\ 00100,\ \overline{10100},\ \overline{00101}\} = \{10101,\ 00100,\ 01011,\ 11010\}.$$
$$Z_3 \cap \overline{Z_3} = \{10101,\ 00100,\ 01011,\ 11010\} \cap \{01010,\ 11011,\ 00101\} = \emptyset.$$

We pose the following open question:

How many elements of the set $\mathrm{Lin}\mathbf{DB}_n$ are semi-perfect?

References

[1] Veronica Becher, Pablo Ariel Heiber, On extending de Bruijn sequences, Inform. Process. Lett. 111 (2011) 930–932.
[2] Donald E. Knuth, The Art of Computer Programming, Volume 4 (Fascicle 2): Generating All Tuples and Permutations, Stoughton, Massachusetts, 2005.
[3] Abraham Lempel, On a homomorphism of the de Bruijn graph and its applications to the design of feedback shift registers, IEEE Trans. Comput. 19 (1970) 1204–1209.
[4] Jeffrey Shallit, On the maximum number of distinct factors of a binary string, Graphs Combin. 9 (1993) 197–200.