Theoretical Computer Science 475 (2013) 34–46

Contents lists available at SciVerse ScienceDirect

Theoretical Computer Science

journal homepage: www.elsevier.com/locate/tcs

Deciding representability of sets of words of equal length☆

F. Blanchet-Sadri [a,∗], Sean Simmons [b]

[a] Department of Computer Science, University of North Carolina, P.O. Box 26170, Greensboro, NC 27402–6170, USA

[b] Department of Mathematics, Massachusetts Institute of Technology, Building 2, Room 236, 77 Massachusetts Avenue, Cambridge, MA 02139–4307, USA

ARTICLE INFO

Article history:
Received 14 May 2012
Received in revised form 24 November 2012
Accepted 30 December 2012
Communicated by M. Crochemore

Keywords:
Computational problems
Algorithms
Complexity classes $\mathcal{P}$ and $\mathcal{NP}$
Combinatorics on words
Partial words
Subwords
Representable sets

ABSTRACT

Partial words are sequences over a finite alphabet that may have holes that match, or are compatible with, all letters in the alphabet; partial words without holes are simply words. Given a partial word w, we denote by $\text{sub}_w(n)$ the set of subwords of w of length n, i.e., words over the alphabet that are compatible with factors of w of length n. We call a set S of words h-representable if $S = \text{sub}_w(n)$ for some integer n and partial word w with h holes. Using a graph theoretical approach, we show that the problem of whether a given set is h-representable can be decided in polynomial time. We also investigate other computational problems related to this concept of representability.

1. Introduction

In the past several years, algorithms and combinatorics on *words*, or sequences of letters over a finite alphabet, have been developing and many important applications in several areas including emergent areas, such as Bioinformatics and DNA computing, have been found (see, for instance, [10,15]). The concept of *subword* in particular has been extensively investigated [13–15]. The Rauzy graphs, useful tools for studying subwords and closely related to the de Bruijn graphs, have been applied to the study of infinite words with small sets of subwords, i.e., low subword complexity, including Sturmian words and DOL words [1,9,11]. A de Bruijn graph is a Rauzy graph for the set of all words of a fixed length over an alphabet of a fixed size, while a Rauzy graph is a subgraph of a de Bruijn graph over some alphabet [14].

In 1999, being motivated by molecular biology of nucleic acids, Berstel and Boasson [2] introduced the terminology of *partial words* for sequences that may have undefined positions, called holes, that match any letter in the alphabet. Algorithms and combinatorics on partial words have been the subject of much investigation (see, for instance, [3]). In this context, de Bruijn graphs have been modified for the construction of compressed sequences containing all words of a given length over a given alphabet [4,7] and Rauzy graphs have been applied to the efficient generation of the so-called minimal Sturmian partial words [6].

In this paper, we introduce a few computational problems on partial words related to subwords and apply Rauzy graphs to their solution. We denote by $\text{sub}_w(n)$ the set of subwords of a partial word w of length n, i.e., words over the alphabet

☆ This material is based upon work supported by the National Science Foundation under Grant No. DMS–1060775. Part of this paper was presented at DCFS 2012 [8]. We thank the referees of preliminary versions of this paper for their very valuable comments and suggestions. We also thank B. J. Wyatt for technical assistance.

∗ Corresponding author. Tel.: +1 13362561125; fax: +1 13363345949.
E-mail addresses: blanchet@uncg.edu (F. Blanchet-Sadri), seanken@mit.edu (S. Simmons).

0304-3975/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
doi:10.1016/j.tcs.2012.12.042

that are compatible with factors of w of length n. In particular, we define Rep, or the problem of deciding whether a set S of words of length n can be represented by a partial word w, i.e., whether $S = \text{sub}_w(n)$. If h is a non-negative integer, we also define h-Rep, or the problem of deciding whether S can be represented by a partial word with exactly h holes. Recently, Blanchet-Sadri et al. [5] have started the study of representing languages by infinite partial words. Here, we deal mostly with finite representing partial words rather than infinite ones.

As an example, consider the set S equal to

$$\{aaaa,\quad aaab,\quad aaba,\quad abaa,\quad abab,\quad abba, baaa,\quad baab,\quad baba,\quad bbaa,\quad bbab,\quad bbba\}.$$

We can check that the set S is 3-representable by the partial word $\diamond\diamond\diamond ab$, 2-representable by $\diamond bba\diamond aabab$ but not 2-representable by any partial word with two consecutive $\diamond$'s. We can also check that S is neither 0-representable nor 1-representable. To see the former, a representing word would have to start with *abba* and contain *bbba*, and thus would have an occurrence of *abbb*, a contradiction. To see the latter, in order to avoid the previous contradiction, the partial word with one hole would need to start with $\diamond bba$. Because of the fact that it would need to contain *bbaa* and *bbab*, it would have the form $\diamond bba \cdots bba \cdots$. This would result in an occurrence of *abbb* or a second occurrence of *abba*, which both lead to contradictions.

The contents of our paper are as follows: In Section 2, we give some definitions that are needed in the sequel, among them is the definition of the Rauzy graph. In Section 3, we prove the membership of Rep and h-Rep in $\mathcal{NP}$. In Section 4, this result on h-Rep is strengthened. For any fixed non-negative integer h, we describe an algorithm that runs in polynomial time which, given a set S of words of length n, decides if there is a partial word w with h holes such that $S = \text{sub}_w(n)$ (our algorithm actually constructs w), showing the membership of h-Rep in $\mathcal{P}$. In Section 5, we prove that some natural subproblem of Rep is in $\mathcal{P}$. In Section 6, we prove other results related to Rep and h-Rep. First we discuss h_1-Rep versus h_2-Rep, where h_1, h_2 are distinct non-negative integers. Next we approximate the problem of finding a partial word w such that $\text{sub}_w(n) = S$ with instead finding the largest subset T of S such that $\text{sub}_w(n) = T$ for some partial word w, i.e., finding a partial word w that is as close as possible to representing S. It turns out that if S is almost equal to A^n, where A is the alphabet over which S is defined, then there exists a subset T of S that contains almost all elements in S and that satisfies $T = \text{sub}_w(n)$ for some partial word w. We also discuss representability by infinite words. Finally in Section 7, we conclude with some remarks.

2. Definitions

We need some background material on partial words (for more information, we refer the reader to [3]). An *alphabet* A is a non-empty finite set of letters. A *(full) word* $w = a_0 \cdots a_{n-1}$ over A is a finite concatenation of letters $a_i \in A$. The *length* of w, denoted by $|w|$, is the number of letters in w. The *empty word* ε is the unique word of length zero. A *partial word* w over A is a sequence of symbols over the extended alphabet $A \cup \{\diamond\}$, where $\diamond \notin A$ plays the role of a hole symbol. The symbol at position i is denoted by $w[i]$. The *set of defined positions of* w, denoted by $D(w)$, consists of the i's with $w[i] \in A$ and the *set of holes of* w, denoted by $H(w)$, consists of the i's with $w[i] = \diamond$. If $H(w) = \emptyset$, then w is a (full) word.

For two partial words w and w' of equal length, we denote by $w \subset w'$ the *containment* of w in w', i.e., $w[i] = w'[i]$ for all $i \in D(w)$; we denote by $w \uparrow w'$ the *compatibility* of w with w', i.e., $w[i] = w'[i]$ for all $i \in D(w) \cap D(w')$. A *completion* $\hat{w}$ is a full word compatible with a given partial word w. For example, $ab\diamond\diamond b \subset ab\diamond ab$, $ab\diamond\diamond b \uparrow a\diamond a\diamond\diamond$, and *ababb* is one of the four completions of $ab\diamond\diamond b$ over the binary alphabet $\{a, b\}$.

If w is a partial word over A, then a *factor* of w is a block of consecutive symbols of w and a *subword* of w is a full word over A compatible with a factor of w. For instance, $ab\diamond\diamond b$ is a factor of $aaab\diamond\diamond ba\diamond$, while *abaab*, *ababb*, *abbab*, *abbbb* are the subwords compatible with that factor. The factor $w[i]w[i+1]\cdots w[j-1]$ will be abbreviated by $w[i..j)$, the discrete interval $[i..j)$ being the set $\{i, i+1, \ldots j-1\}$. Then $\text{sub}(w)$ is the set of all subwords of w; similarly, $\text{sub}_w(n)$ is the set of all subwords of w of length n. Letting h be a non-negative integer, we call a set S of words *h-representable* if $S = \text{sub}_w(n)$ for some integer n and partial word w with h holes; we call S *representable* if it is h-representable for some h.

Let S be a finite set of words of length n over A. For any non-negative integer m, let $\text{sub}_S(m) = \{x \mid |x| = m \text{ and } x \text{ is a subword of some } s \in S\}$. The *Rauzy graph of order* $n-1$ *associated with* S is the digraph $G_S = (V, E)$, where $V = \text{sub}_S(n-1)$ and $E = S$. Each $s \in S$ corresponds to an edge as follows: writing $s = s[0..n-1)a = ua = bv = bs[1..n)$ for some letters $a, b \in A$, there is an edge $(u, v) = (s[0..n-1), s[1..n))$ from u to v labelled by the word s. In other words, each $s \in S$ corresponds to an edge having s's prefix of length $n-1$ as starting vertex and s's suffix of same length as ending vertex. If $u = u_0, u_1, \ldots, u_l = v$ is a path from u to v in G_S, then we associate with it the word $w = u_0 u_1[n-2] u_2[n-2] \cdots u_l[n-2]$. Using this correspondence between paths and words in G_S, we refer also to w as a path in G_S.

3. Membership of Rep and *h*-Rep in $\mathcal{NP}$

In this section, we show that Rep and h-Rep are both in $\mathcal{NP}$. To do this we need the following lemmas.

Lemma 1. *Let S be a set of words of length n. If S is representable, then there exists a partial word w with $|w| \le n(2|S|-1) + \frac{|S|(|S|-1)}{2}$ such that $S = \text{sub}_w(n)$.*

Proof. Assume that w is the shortest partial word such that $S = \text{sub}_w(n)$. Set $S = \{s_0, \ldots, s_{|S|-1}\}$. Let i_j be the smallest integer such that $s_j \uparrow w[i_j..i_j + n)$. Without loss of generality, we can assume that $0 = i_0 \leq i_1 \leq i_2 \leq \cdots \leq i_{|S|-1}$. Clearly, the partial word $w[0..i_{|S|-1} + n)$ contains every word in S as a subword, so since w is minimal it must be the case that $w = w[0..i_{|S|-1} + n)$, which implies

$$|w| = i_{|S|-1} + n = n + \sum_{j=1}^{|S|-1} (i_j - i_{j-1}).$$

Now, assume towards a contradiction that $i_j - i_{j-1} > j + 2n$ for some j, where $1 \leq j \leq |S| - 1$. By definition of i_j, this implies that if $i_{j-1} \leq l < i_j$ then $w[l..l + n)$ is compatible with one of $s_0, \ldots, s_{j-1}$. However, since $i_j - i_{j-1} > j + 2n$ there must be at least $j + 1$ integers in the discrete interval $[i_{j-1} + n..i_j - n)$. By the pigeonhole principle, this implies that we can find j', l_1, and l_2 such that $0 \leq j' \leq j - 1$, $i_{j-1} + n \leq l_1 < l_2 < i_j - n$, $w[l_1..l_1 + n) \uparrow s_{j'}$, and $w[l_2..l_2 + n) \uparrow s_{j'}$. Since $s_{j'}$ is a full word, we have both containments $w[l_1..l_1 + n) \subset s_{j'}$ and $w[l_2..l_2 + n) \subset s_{j'}$.

Thus consider the partial word $w' = w[0..l_1)s_{j'}w[l_2 + n..|w|)$. We want to prove that $\text{sub}_{w'}(n) = S$. First, consider $s_l \in S$. If $l \leq j - 1$ we get $i_l + n \leq l_1$, thus $w[i_l..i_l + n)$ is a factor of $w[0..l_1)$, which by definition of i_l means s_l is a subword of $w[0..l_1)$, and thus is a subword of w'. A similar argument works when $l \geq j$, so $S \subseteq \text{sub}_{w'}(n)$. Next, consider $s \in \text{sub}_{w'}(n)$. Then s is a subword of either $w[0..l_1)s_{j'}$ or $s_{j'}w[l_2 + n..|w|)$. Without loss of generality, assume it is a subword of $w[0..l_1)s_{j'}$. Since $w[l_1..l_1 + n) \subset s_{j'}$, we have $w[0..l_1 + n) \subset w[0..l_1)s_{j'}$. This implies that s is a subword of w, and thus must be in S. Therefore, $S = \text{sub}_{w'}(n)$.

Note, however, that w' is strictly shorter than w, which contradicts the minimality of w. Therefore, $i_j - i_{j-1} \leq j + 2n$ for all $j \in [1..|S|)$. So we get

$$|w| = n + \sum_{j=1}^{|S|-1} (i_j - i_{j-1}) \leq n + \sum_{j=1}^{|S|-1} (j + 2n) = n(2|S| - 1) + \frac{|S|(|S| - 1)}{2}. \quad \square$$

Lemma 2. *Let S be a set of words of length n. If S is h-representable, then there exists a partial word w with h holes such that $|w| \leq n + (|S| + n + 1)(|S| + h - 1)$ and such that $S = \text{sub}_w(n)$.*

Proof. The proof is similar to the one of Lemma 1. Assume that w is the shortest partial word with h holes such that $S = \text{sub}_w(n)$. Here, we can construct a sequence $0 = i_0 \leq i_1 \leq i_2 \leq \cdots \leq i_{m-1}$ such that

- $m \leq |S| + h$;
- if $w[i] = \diamond$ then $i = i_j$ for some j;
- if $s \in S$, there exists some i_j such that i_j is the smallest integer with $w[i_j..i_j + n) \uparrow s$.

Note that $w = w[0..i_{m-1} + n)$ has h holes.

Now, assume towards a contradiction that $i_j - i_{j-1} > |S| + n + 1$ for some $j \in [1..m)$. Since $i_j - i_{j-1} > |S| + n + 1$ there must be at least $|S| + 1$ integers l such that $i_{j-1} < l < i_j - n$. Since every subword of w of length n is in S, by the pigeonhole principle this implies we can find l_1 and l_2 such that $i_{j-1} < l_1 < l_2 < i_j - n$ and such that $w[l_1..l_1 + n) \uparrow w[l_2..l_2 + n)$. However by construction, both $w[l_1..l_1 + n)$ and $w[l_2..l_2 + n)$ must be full words, so in fact must be equal. Thus consider the word $w' = w[0..l_1 + n)w[l_2 + n..|w|)$. Then by a similar argument to the one in Lemma 1 we get that $\text{sub}_{w'}(n) = S$. Moreover w' has exactly h holes but is strictly shorter than w. This is a contradiction. Therefore $i_j - i_{j-1} \leq |S| + n + 1$, so we get that

$$|w| = n + \sum_{j=1}^{m-1} (i_j - i_{j-1}) \leq n + \sum_{j=1}^{m-1} (|S| + n + 1) \leq n + (|S| + n + 1)(|S| + h - 1). \quad \square$$

Note that the bound in Lemma 2 is not optimal, but it serves our purpose.

Proposition 1. Rep *and* h-Rep *are in* $\mathcal{NP}$.

Proof. This is an immediate consequence of Lemmas 1 and 2. $\square$

The question arises as to whether the problems Rep and h-Rep are in $\mathcal{P}$.

4. Membership of h-Rep in $\mathcal{P}$

We also need some background material on graph theory. For instance, recall that a digraph G is *strongly connected* if, for every pair of vertices u and v, there exists a path from u to v. For other concepts not defined here, we refer the reader to [12].

It is known that 0-Rep is in $\mathcal{P}$. Indeed, finding a word w such that $\text{sub}_w(n) = S$ is the same as finding a path in G_S that includes every edge at least once. For example, if $S = \{aaa, aab, aba, baa, bab\}$ then $w = aaababaa$ is a path in G_S that includes every edge at least once, showing that S is 0-representable; note that S is also 1-representable by $\diamond aabab$, 2-representable by $aa\diamond a\diamond$, etc. However, showing the membership of h-Rep in $\mathcal{P}$ is not that simple.

In this section, we show that h-Rep is in $\mathcal{P}$ for any fixed non-negative integer h. We describe a polynomial time algorithm, Algorithm 3, that given a set S of words of length n, decides if there is a partial word w with h holes such that $S = \text{sub}_w(n)$. If so, this algorithm constructs one such w.

The following definition partitions the set of vertices V of a digraph G into disjoint sets $V_0, \ldots, V_r$ with respect to the relation $\rightharpoonup$ defined by: if $u, v \in V$, then we write $u \rightharpoonup v$ if there exists a path in G from u to v. This partition has some useful properties, proved in Lemma 3, that will be exploited later on to construct representing partial words.

Definition 1. Let $G = (V, E)$ be a digraph. The *decomposition of V with respect to* $\rightharpoonup$ is the partition $V_0, \ldots, V_r$ of V, where r is some non-negative integer, defined by

$$V_0 = \{v \in V \mid \text{if } u \in V \text{ and } u \rightharpoonup v, \text{ then } v \rightharpoonup u\}$$

and for $i > 0$,

$$V_i = \left\{ v \in V - \bigcup_{j=0}^{i-1} V_j \mid \text{if } u \notin \bigcup_{j=0}^{i-1} V_j \text{ and } u \rightharpoonup v, \text{ then } v \rightharpoonup u \right\}.$$

In some sense, we can consider V_0 to consist of all minimal elements in V with respect to $\rightharpoonup$, V_1 to consist of all minimal elements in $V - V_0$, and so on. This comes naturally from thinking of $\rightharpoonup$ as a preorder.

Example 1. Consider the set S consisting of the following 30 words of length six, numbered from 1 to 30:

1	*aaaaaa*	6	*aabbaa*	11	*abbbaa*	16	*baabbb*	21	*bbabab*	26	*bbbabb*
2	*aaaaab*	7	*aabbba*	12	*abbbab*	17	*bababb*	22	*bbabbb*	27	*bbbbaa*
3	*aaaabb*	8	*aabbbb*	13	*abbbba*	18	*babbba*	23	*bbbaaa*	28	*bbbbab*
4	*aaabba*	9	*ababbb*	14	*abbbbb*	19	*babbbb*	24	*bbbaab*	29	*bbbbba*
5	*aaabbb*	10	*abbaab*	15	*baabba*	20	*bbaabb*	25	*bbbaba*	30	*bbbbbb*.

Now consider the digraph $G_S = (V, E)$ where $E = S$ and $V = \text{sub}_S(5)$ is the set consisting of the following 20 words of length five, numbered from 1 to 20:

1	*aaaaa*	5	*aabbb*	9	*abbbb*	13	*bbaaa*	17	*bbbaa*
2	*aaaab*	6	*ababb*	10	*baabb*	14	*bbaab*	18	*bbbab*
3	*aaabb*	7	*abbaa*	11	*babab*	15	*bbaba*	19	*bbbba*
4	*aabba*	8	*abbba*	12	*babbb*	16	*bbabb*	20	*bbbbb*.

Then the decomposition of V with respect to $\rightharpoonup$ consists of the sets:

$V_0 = \{aaaaa\}$
$V_1 = \{aaaab\}$
$V_2 = \{aaabb\}$
$V_4 = \{bbaaa\}$
$V_3 = V - (V_0 \cup V_1 \cup V_2 \cup V_4)$.

Fig. 1 illustrates this example.

The following lemma gives useful properties of the decomposition of Definition 1.

Lemma 3. *Let $G = (V, E)$ be a digraph and let $V_0, \ldots, V_r$ be the decomposition of V with respect to $\rightharpoonup$. If $i < j$, $u \in V_j$ and $v \in V_i$, then $u \not\rightharpoonup v$. Moreover, if $v \in V_{i+1}$ then there exists $u \in V_i$ such that $u \rightharpoonup v$. Finally for $i < r$, there exist vertices $u \in V_i$ and $v \in V_{i+1}$ such that $(u, v) \in E$.*

Proof. First, consider $i < j$. Assume $u \in V_j$ and $v \in V_i$ are such that $u \rightharpoonup v$. Since $u \notin \bigcup_{l=0}^{i-1} V_l$, it follows by the definition of V_i that $v \rightharpoonup u$. Thus if, for any vertex w, $w \notin \bigcup_{l=0}^{i-1} V_l$ and $w \rightharpoonup u$, the assumption that $u \rightharpoonup v$ implies $w \rightharpoonup v$. Since $v \in V_i$ this implies $v \rightharpoonup w$, so since $u \rightharpoonup v$ it follows that $u \rightharpoonup w$. We get $u \in V_i$, which is impossible.

Next, consider $v \in V_{i+1}$. Assume there is no $u \in V_i$ such that $u \rightharpoonup v$. Since $v \in V_{i+1}$, if, for any vertex w, $w \notin \bigcup_{j=0}^{i} V_j$ and $w \rightharpoonup v$, then $v \rightharpoonup w$. Furthermore, if $w \notin \bigcup_{j=0}^{i-1} V_j$ and $w \rightharpoonup v$, then $v \rightharpoonup w$. This, however, implies by definition that $v \in V_i$, a contradiction.

Finally, consider $i \in [0..r)$ and let $v \in V_{i+1}$. By the above, there exists $u \in V_i$ such that $u \rightharpoonup v$. Let $u = u_0, u_1, \ldots, u_l = v$ be a path from u to v. Note that since there is no path from any vertex in $V_{r'}$ to any vertex in V_{i+1} for $r' > i + 1$, it follows, since $u_l \in V_{i+1}$, that if $u_j \in V_{r'}$, then $r' \leq i + 1$. By a similar argument, $r' \geq i$. Then let l' be the smallest integer such that $u_{l'} \in V_{i+1}$. The above tells us that $u_{l'-1} \in V_i$, so $(u_{l'-1}, u_{l'})$ is the desired edge. □

The following definition introduces our set S_h, given a set S of words of length n. This set is crucial in the description of our algorithm. We then show, in a lemma, that if w is a partial word with h holes whose set of subwords of length n is a non-empty subset of S, then w can be built from a h-holed sequence in S_h.

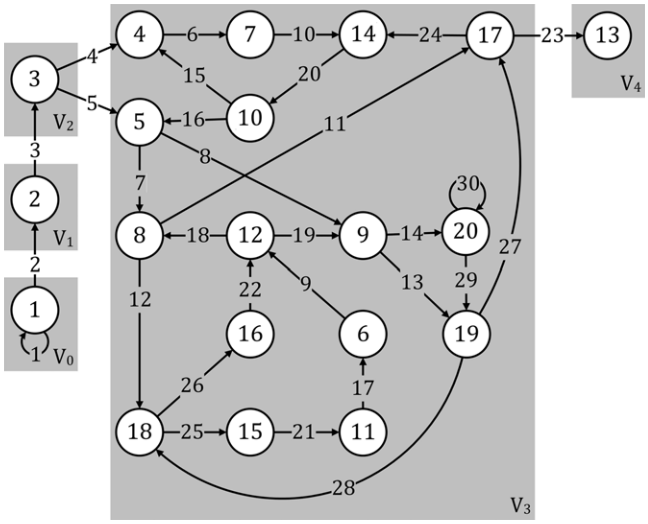

Fig. 1. The decomposition V_0, V_1, V_2, V_3, V_4 of the vertex set V in the graph $G_S = (V, E)$ associated with the set S of Example 1; we let $G_0, \ldots, G_4$ be the subgraphs of G_S spanned by $V_0, \ldots, V_4$, respectively.

Definition 2. Given a set S of words of length n, we define the set S_h such that $(s_0, \ldots, s_{l-1}) \in S_h$ if $l > 0$ and the following conditions hold:

1. Each s_i is a partial word with $|s_i| \geq n - 1$;
2. The partial word $s_0 \cdots s_{l-1}$ has exactly h holes;
3. Each s_i, except possibly s_0 and s_{l-1}, has at least one hole;
4. If x is a full word and a factor of some s_i, then $|x| < 2n$;
5. If $s_i[j] = \diamond$, then for $i > 0$ we have that $j \geq n - 1$, and for $i < l - 1$ we have that $j < |s_i| - n + 1$;
6. For each i and for every $m \leq n$, $\mathrm{sub}_{s_i}(m) \subseteq \mathrm{sub}_S(m)$.

Lemma 4. *Let S be a set of words of length n and w be a partial word with h holes. If $\mathrm{sub}_w(n) \subseteq S$ and $\mathrm{sub}_w(n) \neq \emptyset$, then there exist a positive integer l and a tuple $(s_0, \ldots, s_{l-1})$ in S_h such that $w = s_0 w_0 s_1 w_1 \cdots w_{l-2} s_{l-1}$, where each w_i is a full word.*

Proof. We proceed by induction on $|w|$. This holds trivially if $|w| = n$ by letting $s_0 = w$ and $l = 1$. Therefore assume that the claim holds for all w' with $|w'| < |w|$. If w does not contain any full word of length greater than or equal to $2n$ as a factor, letting $l = 1$ and $s_0 = w$, gives us what we want. Therefore, assume that w contains a factor y that is a full word of length at least $2n$. Furthermore, assume that $|y|$ is maximal. There exists an i such that $w[i..i + |y|) = y$. Furthermore, the maximality of y implies that either $i + |y| = |w|$, $i = 0$, or $w[i - 1] = w[i + |y|] = \diamond$.

First, consider the case $w[i - 1] = w[i + |y|] = \diamond$. Then $w = x\diamond y\diamond z = x\diamond w[i..i + n - 1)y'w[i + |y| - n + 1..i + |y|)\diamond z$ for some y'. Assume $x_0 = x\diamond w[i..i + n - 1)$ has h_0 holes and $z_0 = w[i + |y| - n + 1..i + |y|)\diamond z$ has h_1 holes. Then by the inductive hypothesis, there exist $(t_0, \ldots, t_{l_0-1}) \in S_{h_0}$ and full words $w_0, \ldots, w_{l_0-2}$ such that $x_0 = t_0 w_0 \cdots w_{l_0-2} t_{l_0-1}$. Similarly, there exist $(t'_0, \ldots, t'_{l_1-1}) \in S_{h_1}$ and full words $w'_0, \ldots, w'_{l_1-2}$ such that $z_0 = t'_0 w'_0 \cdots w'_{l_1-2} t'_{l_1-1}$. We can let $(s_0, \ldots, s_{l-1}) = (t_0, \ldots, t_{l_0-1}, t'_0, \ldots, t'_{l_1-1}) \in S_h$ when both t_{l_0-1} and t'_0 have holes; otherwise, in the case of t_{l_0-1} having a hole and t'_0 having no hole for instance, we can let $(s_0, \ldots, s_{l-1}) = (t_0, \ldots, t_{l_0-1}, t'_1, \ldots, t'_{l_1-1})$.

To see the latter, we check that Conditions 1–6 of Definition 2 hold. For Condition 1, each t_i, $0 \leq i \leq l_0 - 1$, and each t'_i, $1 \leq i \leq l_1 - 1$, are partial words with $|t_i|, |t'_i| \geq n - 1$. For Condition 2, the partial word $s_0 \cdots s_{l-1} = t_0 \cdots t_{l_0-1} t'_1 \cdots t'_{l_1-1}$ has exactly $h_0 + h_1 = h$ holes since t'_0 has no hole. For Condition 3, $t_1, \ldots, t_{l_0-1}, t'_1, \ldots, t'_{l_1-2}$ have at least one hole each. For Condition 4, if x' is a full word and a factor of some t_i, $0 \leq i \leq l_0 - 1$, then $|x'| < 2n$ (similarly if x' is a full word and a factor of some t'_i, $1 \leq i \leq l_1 - 1$). For Condition 5, if $t_i[j] = \diamond$, then for $i > 0$ we have that $j \geq n - 1$, and for $i < l_0 - 1$ we have that $j < |t_i| - n + 1$ and similarly if $t'_i[j] = \diamond$, then for $i > 0$ we have that $j \geq n - 1$, and for $i < l_1 - 1$ we have that $j < |t'_i| - n + 1$; since $\diamond w[i..i + n - 1)$ and t_{l_0-1} are both suffixes of x_0, and $w[i..i + n - 1)$ is a full word and t_{l_0-1} has a hole, $\diamond w[i..i + n - 1)$ is a suffix of t_{l_0-1}, so if $t_{l_0-1}[j] = \diamond$, then $j < |t_{l_0-1}| - n + 1$. Finally for Condition 6, for every $m \leq n$, $\mathrm{sub}_{t_i}(m) \subseteq \mathrm{sub}_S(m)$ for each i, $0 \leq i \leq l_0 - 1$, and $\mathrm{sub}_{t'_i}(m) \subseteq \mathrm{sub}_S(m)$ for each i, $1 \leq i \leq l_1 - 1$.

Next, consider the case $i = 0$ (the case $i + |y| = |w|$ is almost identical). Let $s_0 = w[0..n - 1)$ and $w' = w[n + 1..|w|)$. Applying the inductive hypothesis to w', there exist $(t_0, \ldots, t_{l_0-1}) \in S_h$ and full words $w_0, \ldots, w_{l_0-2}$ such that $w' = t_0 w_0 \cdots w_{l_0-2} t_{l_0-1}$. Then if t_0 contains a hole, we let $(s_0, \ldots, s_{l-1}) = (s_0, t_0, \ldots, t_{l_0-1})$. Otherwise, we let $(s_0, \ldots, s_{l-1}) = (s_0, t_1, \ldots, t_{l_0-1})$. Conditions 1–6 of Definition 2 can be checked similarly as above. □

Example 2. Returning to Example 1, let $n = 6$ and $h = 5$. Consider

$$w = aaaaaaaaaaaaaaabb\diamond abbbbbababbbba\diamond bbbb\diamond\diamond bbbaa\diamond.$$

Here, $\mathrm{sub}_w(6) = S - \{baabba\}$. Using the proof of Lemma 4, we can factorize w as follows:

$$\underset{s_0}{aaaaa} \quad aaaaa \quad \underset{s_1}{aaaaabb\diamond abbbb} \quad baba \quad \underset{s_2}{bbbba\diamond bbbb\diamond\diamond bbbaa\diamond}.$$

By Definition 2, $(s_0, s_1, s_2) \in S_5$.

Our next step is to prove that Algorithm 1, given below, generates S_h in polynomial time. The idea behind the algorithm is simple. Basically if $(s_0, \dots, s_{l-1}) \in S_h$, then $l \le h+2$. Furthermore, there exists a constant c such that $|s_i| < cn$, and each s_i can be created by concatenating subwords of elements of S. Using this, it is easy to produce S_h by enumerating all such $(s_0, \dots, s_{l-1})$'s. Algorithm 1 works as follows:

- Creates T_0, the set of all $t_0 t_1 \cdots t_{2h+1}$, where each $t_j \in \mathrm{sub}(S)$ ($\mathrm{sub}(S)$ denotes the set of subwords of elements of S);
- For $h' = 1, \dots, h$, creates $T_{h'}$ by inserting h' holes into the elements of T_0 (i.e., by replacing h' positions by $\diamond$'s);
- Creates $T = T_0 \cup T_1 \cup \cdots \cup T_h$;
- Creates $S' = T \cup T^2 \cup \cdots \cup T^{h+2}$;
- Removes from S' any sequence $(s_0, \dots, s_{l-1})$ that does not satisfy one of the conditions 1–6 of Definition 2;
- Returns $S_h = S'$.

The size of the set $\mathrm{sub}(S)^{2h+2}$ is bounded by a polynomial in the size of the input.

Algorithm 1 Generating S_h, where S is a set of words of length n

1: Let $T'_0 = \emptyset$
2: **for** $(t_0, \dots, t_{2h+1}) \in \mathrm{sub}(S)^{2h+2}$ **do**
3: $\quad T'_0 = T'_0 \cup \{t_0 \cdots t_{2h+1}\}$
4: Let $T_0 = T'_0$
5: **for** $h' = 1$ to h **do**
6: $\quad$ Let $T_{h'} = \emptyset$
7: $\quad$ **for** $t \in T_{h'-1}$ **do**
8: $\quad\quad$ **for** $j = 0$ to $|t| - 1$ **do**
9: $\quad\quad\quad$ **if** $t[j] \ne \diamond$ **then**
10: $\quad\quad\quad\quad$ Letting $t' = t$, replace $t'[j]$ by $\diamond$ and add t' to $T_{h'}$
11: Let $T = \bigcup_{h'=0}^{h} T_{h'}$
12: Let $S' = \bigcup_{l=1}^{h+2} T^l$
13: **for** $s = (s_0, \dots, s_{l-1}) \in S'$ **do**
14: $\quad$ **for** $i = 0$ to $l - 1$ **do**
15: $\quad\quad$ **if** s_i is a full word and $i \notin \{0, l-1\}$, or s_i contains a $\diamond$ in its prefix of length $n-1$ and $i \ne 0$, or s_i contains a $\diamond$ in its suffix of length $n-1$ and $i \ne l-1$, or $|s_i| < n-1$, or s_i contains a full word t of length at least $2n$ as a factor **then**
16: $\quad\quad\quad$ remove s from S'
17: $\quad\quad$ **for** $m = 1$ to n **do**
18: $\quad\quad\quad$ **if** $\mathrm{sub}_{s_i}(m) \not\subseteq \mathrm{sub}_S(m)$ **then**
19: $\quad\quad\quad\quad$ remove s from S'
20: $\quad$ **if** $s_0 \cdots s_{l-1}$ does not contain exactly h holes **then**
21: $\quad\quad$ remove s from S'
22: **return** $S_h = S'$

Lemma 5. *For any fixed non-negative integer h, Algorithm 1 generates S_h given a set S of words of length n. Furthermore, there exists a polynomial $f_h(x, y)$ such that $|S_h| \le f_h(|S|, n)$. Algorithm 1 is exponential in h, which – since h is fixed – means that it runs in polynomial time.*

Proof. Let T'_0, T_h, T, etc. be as in the algorithm. First we want to show that if $(s_0, \dots, s_{l-1}) \in S_h$, then $s_i \in T$. To see this, let $\hat{s}_i$ be any completion of s_i. Then the facts that s_i contains at most h holes and no full word of length greater than or equal to $2n$ as a factor imply that $|\hat{s}_i| = |s_i| \le 2n - 1 + h(2n) = 2(h+1)n - 1$. This means that $|\hat{s}_i| = qn + q'$ for some integers q and q', where $0 \le q' < n$ and $q < 2h + 2$. Thus we can write $\hat{s}_i = t_0 t_1 \cdots t_{2h+1}$ where t_j is of length n for $j < q$, t_q is of length q', and $t_j = \varepsilon$ for all other j. Note for each j, since $|t_j| \le n$, we have by definition of S_h that $t_j \in \mathrm{sub}_{\hat{s}_i}(|t_j|) \subseteq \mathrm{sub}_{s_i}(|t_j|) \subseteq \mathrm{sub}_S(|t_j|)$. Therefore $(t_0, \dots, t_{2h+1}) \in \mathrm{sub}(S)^{2h+2}$, where $\mathrm{sub}(S)$ is the set of all subwords of S, so $\hat{s}_i = t_0 t_1 \cdots t_{2h+1} \in T'_0 = T_0$ by Lines 3–4.

Then by a simple induction argument, if s' is formed from $\hat{s}_i$ by inserting $h' \le h$ holes then $s' \in T_{h'} \subseteq T$ (see Lines 5–11). In particular, $s_i \in T$. Since this is true for all i, it follows that $(s_0, \dots, s_{l-1}) \in T^l$. Note that $l \le h+2$ since $s_0 \cdots s_{l-1}$ contains

h holes, and for $i \in [1..l-1)$, we know that s_i must contain at least one of the holes. Thus, $(s_0, \ldots, s_{l-1}) \in T^l \subseteq S'$ (see Line 12).

We have now reached the for loop on Line 13 of the algorithm. Assume that $s = (s_0, \ldots, s_{l-1}) \in S'$. Then, by looking at the interior of this for loop (Lines 14–21), s is not removed from S' if and only if the conditions 1–6 of Definition 2 hold. Furthermore, by construction $l > 0$. Therefore s is removed from S' if and only if $s \notin S_h$. Since $S_h \subseteq S'$ at the beginning of the loop, it follows that at the end of the loop $S_h = S'$. The algorithm then returns $S_h = S'$ on Line 22. We know that $|\text{sub}(S)| \leq |S|n^2 + 1$ (since each element of S contains at most n non-empty subwords beginning at each of its n positions). Thus $|\text{sub}(S)^{2h+2}| \leq (|S|n^2 + 1)^{2h+2}$, a polynomial in the input, and so there exists a polynomial $f_h(x, y)$ such that the size of S' is upper bounded by $f_h(|S|, n)$. From this point, seeing that the algorithm runs in polynomial time is just a standard running time analysis. $\square$

Algorithm 2 Checking words for $(s_0, \ldots, s_{l-1}) \in S_h$

Let $G = G_S = (V, E)$
if $l = 1$ **then**
  if $\text{sub}_{s_0}(n) = S$ **then**
    return s_0
  else
    return null
Decompose V into $V_0, \ldots, V_r$ with respect to $\rightharpoonup$
for $j = 0$ to $l - 1$ **do**
  if $j > 0$ **then**
    Let $s_{0,j} = s_j[0..n-1)$
    Let $i_{0,j}$ be the index with $s_{0,j} \in V_{i_{0,j}}$
  if $j < l - 1$ **then**
    Let $s_{1,j} = s_j[|s_j| - n + 1..|s_j|)$
    Let $i_{1,j}$ be the index with $s_{1,j} \in V_{i_{1,j}}$
for $j = 0$ to $l - 2$ **do**
  if $i_{1,j} > i_{0,j+1}$ **then**
    return null
  if $j \neq 0$ and $i_{0,j} > i_{1,j}$ **then**
    return null
for $i = 0$ to r **do**
  if $i_{1,j} \leq i \leq i_{0,j+1}$ for some j and G_i is not strongly connected **then**
    return null
for $i = 0$ to $r - 1$ **do**
  Choose $u_i \in V_i$ and $v_{i+1} \in V_{i+1}$ such that $(u_i, v_{i+1}) \in E$
for $j = 0$ to $l - 2$ **do**
  Choose a path $p_{i_{1,j}}$ from $s_{1,j}$ to $u_{i_{1,j}}$ that includes every edge in $E_{i_{1,j}}$
  for $i = i_{1,j} + 1$ to $i_{0,j+1} - 1$ **do**
    Choose a path p_i from v_i to u_i that includes every edge in E_i
  if $i_{1,j} \neq i_{0,j+1}$ (resp., $i_{1,j} = i_{0,j+1}$) **then** choose a path $p_{i_{0,j+1}}$ from $v_{i_{0,j+1}}$ (resp., $u_{i_{0,j+1}}$) to $s_{0,j+1}$ that includes every edge in $E_{i_{0,j+1}}$
  Let P_j be $p_{i_{1,j}}$, followed by the edge from $u_{i_{1,j}}$ to $v_{i_{1,j}+1}$, then $p_{i_{1,j}+1}$, then the edge from $u_{i_{1,j}+1}$ to $v_{i_{1,j}+2}$, and continuing until $s_{0,j+1}$
  Let w_j be the word associated with G's path P_j
Let $w = s_0[0..|s_0| - n + 1)w_0 s_1[n - 1..|s_1| - n + 1)w_1 s_2[n - 1..|s_2| - n + 1) \cdots w_{l-2} s_{l-1}[n - 1..|s_{l-1}|)$
if $\text{sub}_w(n) = S$ **then**
  return w
else
  return null

Our next step is to prove that Algorithm 2 constructs, in polynomial time, a partial word w with h holes such that $\text{sub}_w(n) = S$ from a given h-holed sequence $(s_0, \ldots, s_{l-1})$ in S_h if such a partial word exists. Algorithm 2 uses the decomposition of the vertex set V of $G = G_S = (V, E)$ with respect to $\rightharpoonup$, i.e., $V_0, \ldots, V_r$. The partial word w has the form

$$s_0[0..|s_0| - n + 1)w_0 s_1[n - 1..|s_1| - n + 1)w_1 s_2[n - 1..|s_2| - n + 1) \cdots w_{l-2} s_{l-1}[n - 1..|s_{l-1}|)$$

where each w_j is a path from $s_j[|s_j| - n + 1..|s_j|)$ to $s_{j+1}[0..n-1)$ satisfying some conditions related to the spanned subgraphs $G_0 = (V_0, E_0), \ldots, G_r = (V_r, E_r)$. The output w is constructed as follows: it starts with $s_0[0..|s_0| - n + 1)$, followed by $w_0 = s_{1,0} w_0' s_{0,1}$ where $s_{1,0} = s_0[|s_0| - n + 1..|s_0|) \in V_{i_{1,0}}$ and $s_{0,1} = s_1[0..n-1) \in V_{i_{0,1}}$, followed by $s_1[n - 1..|s_1| - n + 1)$, followed by $w_1 = s_{1,1}w'_1 s_{0,2}$ where $s_{1,1} = s_1[|s_1| - n + 1..|s_1|) \in V_{i_{1,1}}$ and $s_{0,2} = s_2[0..n-1) \in V_{i_{0,2}}, \ldots,$ followed by $w_{l-2} = s_{1,l-2}w'_{l-2}s_{0,l-1}$ where $s_{1,l-2} = s_{l-2}[|s_{l-2}| - n+1..|s_{l-2}|) \in V_{i_{1,l-2}}$ and $s_{0,l-1} = s_{l-1}[0..n-1) \in V_{i_{0,l-1}}$, and ends with $s_{l-1}[n-1..|s_{l-1}|)$. Note that in the description of Algorithm 2: at Line 24, this can be done by Lemma 3; at Line 26, this can be done since $G_{i_{1,j}}$ is strongly connected; at Line 28, this can be done since G_i is strongly connected; and at Line 29, this can be done since $G_{i_{0,j+1}}$ is strongly connected.

Lemma 6. *Let S be a set of words of length n and $(s_0, \ldots, s_{l-1}) \in S_h$. If there exists a partial word w' with h holes such that $\text{sub}_{w'}(n) = S$ and $w' = s_0x_0s_1x_1 \cdots x_{l-2}s_{l-1}$ for some full words x_j, then Algorithm 2 returns a partial word w with h holes such that $\text{sub}_w(n) = S$ and $w = s_0y_0s_1y_1 \cdots y_{l-2}s_{l-1}$ for some full words y_j. Otherwise, it returns null. Furthermore, the algorithm runs in polynomial time.*

Proof. The algorithm can return $w \neq$ null only on Line 4 when $l = 1$, or on Line 34 when $\text{sub}_w(n) = S$. The case $l = 1$ is trivial, thus assume it does so on Line 34. From Line 32, $w = s_0[0..|s_0| - n + 1)w_0s_1[n-1..|s_1| - n + 1)w_1s_2[n-1..|s_2| - n + 1) \cdots w_{l-2}s_{l-1}[n-1..|s_{l-1}|)$, where $w_j = s_j[|s_j| - n + 1..|s_j|)w'_j s_{j+1}[0..n-1)$ for some full word w'_j. Consequently, $w = s_0w'_0s_1w'_1 \cdots w'_{l-2}s_{l-1}$ for some full words w'_j.

On the other hand, assume the algorithm returns null. Suppose towards a contradiction that there exists w' with h holes such that $\text{sub}_{w'}(n) = S$ and $w' = s_0x_0s_1x_1 \cdots x_{l-2}s_{l-1}$ for some full words x_j. We will check each return statement one by one to see which returned null. Let $w'_i = s_i[|s_i| - n + 1..|s_i|)x_is_{i+1}[0..n-1)$, then note that each w'_i is a full word with

$$w' = s_0[0..|s_0| - n + 1)w'_0s_1[n-1..|s_1| - n + 1)w'_1s_2[n-1..|s_2| - n + 1) \cdots$$
$$w'_{l-2}s_{l-1}[n-1..|s_{l-1}|).$$

First, consider the return statement on Line 6. In this case $l = 1$. Clearly $s_0 = w'$, so Line 4 returns s_0. This is a contradiction, thus $l \neq 1$. Therefore, assume null was returned on Line 19 (the case of Line 17 is similar). For $j \in [1..l-1)$, we have that $i_{1,j} < i_{0,j}$. However since $s_{0,j}$ occurs in s_j before $s_{1,j}$, and $\text{sub}_{s_j}(n) \subseteq S$, there is a path in $G = G_S$ from $s_{0,j} \in V_{i_{0,j}}$ to $s_{1,j} \in V_{i_{1,j}}$, which contradicts Lemma 3.

Next, consider the return statement on Line 22. There exist i and j such that $i_{1,j} \le i \le i_{0,j+1}$ and G_i is not strongly connected. We consider the case $i_{1,j} < i < i_{0,j+1}$ (the cases $i = i_{0,j+1}$ and $i = i_{1,j}$ are similar). Let $u \in V_i$. Note that u is not a subword of $s_{i'}$ for any i'. Otherwise, there is a path from $s_{0,i'}$ to u and from u to $s_{1,i'}$, in which case $i_{0,i'} \le i \le i_{1,i'}$. Assuming $j > i'$ (the other cases being similar), we get the contradiction $i_{1,i'} \le i_{1,j} < i \le i_{1,i'}$. It follows that u is in some $w'_{j'}$. Therefore, consider $v \in V_i$. Then there is a completion of w', say $\hat{w}'$, such that v is a subword of $\hat{w}'$. It is easy to see that $w'_{j'}$ is a subword of $\hat{w}'$ as well. Since u is in $w'_{j'}$, we have that u is a subword of $\hat{w}'$. Since u and v are both subwords of $\hat{w}'$, there is a path in G from u to v or a path from v to u (using the correspondence between words and paths in G). Without loss of generality, $u \rightharpoonup v$. By definition of V_i, however, this implies $v \rightharpoonup u$, so by definition of $\rightharpoonup$, G_i must be strongly connected, a contradiction.

Next, consider the return statement on Line 36. This implies, if w is as in the algorithm, that $\text{sub}_w(n) \neq S$. Note that if $x \in \text{sub}_w(n)$, then either $x \in \text{sub}_{s_i}(n) \subseteq S$ for some i or $x \in \text{sub}_{w_i}(n) \subseteq S$ for some i. Thus, $\text{sub}_w(n) \subseteq S$ and there exists $e \in S$ such that $e \notin \text{sub}_w(n)$. Since $E = S$, e is an edge in the edge set E of G.

Suppose towards a contradiction that e is in E_i for some i. Consider the case $i_{1,j} \le i \le i_{0,j+1}$ for some j. Then by construction e occurs in p_i, and thus in P_j. This implies that e is a subword of w_j, and thus a subword of w, a contradiction. Next, consider the case $i_{0,j} < i < i_{1,j}$ for some j. Since s_j is a subword of w, it follows that e is not a subword of s_j. Thus, assume that e is a factor of $s_0[0..|s_0| - n + 1)w'_0 \cdots w'_{j-1}$ (the case of e being a factor of $w'_j \cdots w'_{l-2}s_{l-1}[n-1..|s_{l-1}|)$ is similar). Since $s_j[0..n-1)$ is a suffix of $s_0[0..|s_0| - n + 1)w'_0 \cdots w'_{j-1}$, it is easy to see that $u \rightharpoonup s_j[0..n-1)$ where $u \in V_i$, $s_j[0..n-1) \in V_{i_{0,j}}$ and $i_{0,j} < i$, contradicting Lemma 3. Therefore, either $i < i_{1,0}$ or $i > i_{0,l-1}$. However by similar arguments, these cases also lead to contradictions.

Thus, there exist $i \neq i'$ such that e is an edge from $u \in V_i$ to $v \in V_{i'}$. By Lemma 3, $i < i'$. Note that e is not a subword of s_j for any j, since otherwise it would be a subword of w. Thus, e is a subword of some w'_j. Lemma 3 implies that $i_{1,j} \le i < i' \le i_{0,j+1}$.

Assume that $i' > i + 1$. Set $w'_j = yez$ for some y, z. Every subword of w'_j of length $n-1$ is a subword of either $ye[0..n-1) = yu$ or $e[1..n)z = vz$. Since $V_{i+1} \neq \emptyset$, consider any $x \in V_{i+1}$. Then x cannot be a subword of yu since otherwise $x \rightharpoonup u$, contradicting Lemma 3. Similarly, it cannot be a subword of vz. By construction, however, x is a subword of w'_j, a contradiction.

Now, assume that $i' = i + 1$. By construction of P_j, there must exist some $u' \in V_i$ and $v' \in V_{i+1}$ such that $f = (u', v')$ is an edge in P_j. Thus f is a subword of w. Since e is not a subword of w, we have $f \neq e$. However, both e and f must occur as subwords of w'. This implies that there exists a completion $\hat{w}'$ of w' with f as a subword. Note, however, that since w'_j is full and w'_j is a factor of w', it must be a factor of $\hat{w}'$, so e is also a subword of $\hat{w}'$. Without loss of generality, we can assume that e occurs before f in $\hat{w}'$. This implies that v occurs before u' in $\hat{w}'$, so $v \rightharpoonup u'$ (since $\hat{w}'$ corresponds to a path in G). The latter along with $v \in V_{i+1}$ and $u' \in V_i$ contradict Lemma 3.

Finally using standard run time analysis techniques, it is easy to see that the algorithm can be made to run in polynomial time. $\square$

Example 3. Returning to Examples 1 and 2, given as input $(s_0, s_1, s_2) \in S_5$, Algorithm 2 computes the following values:

j	$s_{1,j}$	$i_{1,j}$	$s_{0,j}$	$i_{0,j}$
0	$aaaaa$	0		
1	$abbbb$	3	$aaaaa$	0
2			$bbbba$	3.

Then Algorithm 2 may output the following word w to represent the set S:

$$\underbrace{aaaaa}_{s_0} \quad w'_0 \quad \underbrace{aaaaabb\diamond abbbb}_{s_1} \quad w'_1 \quad \underbrace{bbbba\diamond bbbb\diamond\diamond bbbaa\diamond}_{s_2}$$

where $w'_0 = \varepsilon$ and $w'_1 = bbababbbbabbbabbbbaabbaabbbaa$. Note that

$$w_0 = s_0[|s_0| - n + 1..|s_0|)w'_0 s_1[0..n - 1) = aaaaaw'_0aaaaa$$

is a path from $aaaaa$ to $aaaaa$ visiting every edge in G_0 and

$$w_1 = s_1[|s_1| - n + 1..|s_1|)w'_1 s_2[0..n - 1) = abbbbw'_1bbbba$$

is a path from $abbbb$ to $bbbba$ visiting every edge in G_3.

Our next step is to prove that Algorithm 3 determines whether or not a given set of words of equal length is h-representable.

Algorithm 3 Deciding the h-representability of a set S of words of equal length

```
1: if S = ∅ then
2:    return ε
3: Generate S_h using Algorithm 1
4: for s ∈ S_h do
5:    Let w be the partial word produced by Algorithm 2
6:    if w ≠ null then
7:       return w
8: return null
```

Theorem 1. *If a given input set S of words of length n is not h-representable, then Algorithm 3 returns null. Otherwise, it returns a partial word w with h holes such that $\text{sub}_w(n) = S$. Furthermore, it runs in polynomial time.*

Proof. First, assume that there exists a partial word w' with h holes such that $\text{sub}_{w'}(n) = S$. If S is empty, then the algorithm returns ε as it should. Therefore, assume S is non-empty. We can write $w' = s_0 w_0 \cdots w_{l-2} s_{l-1}$ for some $s = (s_0, \ldots, s_{l-1}) \in S_h$, where each w_i is full by Lemma 4. Algorithm 3 then goes on to generate S_h at Line 3, and begins the for loop at Line 4. Either the for loop reaches s or exits beforehand. The only way it exits before reaching s is if Algorithm 3 returns $w \neq$ null (Lines 6–7), where w is output by Algorithm 2. This implies, however, that w has h holes and $\text{sub}_w(n) = S$. Therefore, assume Algorithm 3 does not exit before reaching s. Letting w be produced by Algorithm 2, by Lemma 6, the fact that $w' = s_0 w_0 \cdots w_{l-2} s_{l-1}$ implies that $w \neq$ null. Thus Algorithm 3 returns w. However, Lemma 6 also says that w has h holes and $\text{sub}_w(n) = S$.

Now, assume that there exists no such w'. Then S is not empty, and Lemma 6 implies that Algorithm 2 must return null for every $s \in S_h$. Thus Algorithm 3 returns null, proving the algorithm works.

Finally, Algorithm 3 runs in polynomial time. This follows easily from the fact that Lemma 5 implies that given any fixed non-negative integer h, there exists a polynomial $f_h(x, y)$ such that $|S_h| \leq f_h(|S|, n)$ (thus the for loop only iterates a polynomial number of times in the input size $n|S|$), the fact that generating S_h using Algorithm 1 takes polynomial time, and the fact that Algorithm 2 runs in polynomial time. □

Corollary 1. *h-Rep is in $\mathcal{P}$ for any fixed non-negative integer h.*

5. Membership of a subproblem of Rep in $\mathcal{P}$

In this section, we give a subproblem of Rep that is in $\mathcal{P}$, i.e., we prove membership in $\mathcal{P}$ of the problem of deciding whether a set S of words of length n can be represented by a partial word w such that every subword of w of length $n - 1$ occurs exactly once in w. To prove this membership, we give characterizing properties, that can be checked in polynomial time, of the corresponding graphs G_S. We first need some terminology.

Definition 3. Let S be a set of words of length n over some alphabet A, $|A| = k > 1$, and let $G = G_S = (V, E)$. A *partial word path* is a sequence $A_0, \ldots, A_m$ of non-empty subsets of $V = \text{sub}_S(n - 1)$ such that the following conditions 1–3 hold:

1. There exists a partial word u_0 satisfying $|u_0| = n - 1$ and $\mathrm{sub}_{u_0}(n-1) = A_0$;
2. For each $i > 0$, either
$$A_i = \{va \mid a \in A \text{ and } bv \in A_{i-1} \text{ for some } b \in A\} \tag{1}$$
or there exists an $a \in A$ such that
$$A_i = \{va \mid bv \in A_{i-1} \text{ for some } b \in A\} \tag{2}$$
(note that Eq. (1) is the equivalent of adding a hole);
3. If $bv \in A_{i-1}$ and $va \in A_i$ for some $a, b \in A$ and full word v, then $bva \in E$.

Let h' be the number of i's such that Eq. (1) holds. We say that the partial word path $A_0, \ldots, A_m$ has h holes if $h = \log_k |A_0| + h'$ (note that $\log_k |A_0|$ is the number of holes in u_0, defined in Statement 1, because each hole in u_0 can be filled by one of k letters).

We say that a partial word path contains an edge $e = (x, y)$ if there exists an i such that $x \in A_i$ and $y \in A_{i+1}$.

Finally, defining u_i recursively by $u_i = u_{i-1}\diamond$ if A_i satisfies Eq. (1) and $u_i = u_{i-1}a$ if A_i satisfies Eq. (2) for some $a \in A$, we say that u_m is a *partial word associated with the partial word path* $A_0, \ldots, A_m$.

The following example illustrates Definition 3.

Example 4. We refer to the partial word w with 5 holes of length 65 of Example 3. For $0 \le i \le 60$, let $A_i = \mathrm{sub}_{w[i..i+5)}(5)$. Here, $A_0 = A_1 = A_2 = A_3 = A_4 = A_5 = \{aaaaa\}$, $A_6 = \{aaaab\}$, $A_7 = \{aaabb\}$, $A_8 = \{aabba, aabbb\}, \ldots$. We can check that A_8 satisfies Eq. (1) and A_7 satisfies Eq. (2). The number of i's such that Eq. (1) holds is 5, so $A_0, \ldots, A_{60}$ is a partial word path with 5 holes which contains in particular the edge $(aaabb, aabba)$, labelled by $aaabba$.

In the zero-hole case, the following remark tells us that $S = \mathrm{sub}_w(n)$ for a full word w if and only if there is a path in G_S including every edge at least once. This is decidable in polynomial time, as we knew already. Note, however, that the remark also gives a polynomial time algorithm that works in the one-hole case.

Remark 1. Let S be a set of words of length n. Then there exists a partial word w with h holes such that $S = \mathrm{sub}_w(n)$ if and only if there exists a partial word path with h holes that includes every edge of G_S at least once.

To see this, assuming that such a w exists, let $A_i = \mathrm{sub}_{w[i..i+n-1)}(n-1)$. Then $A_0, \ldots, A_{|w|-n+1}$ is the partial word path we want. We will refer to it as *the partial word path associated with w*. On the other hand, assuming that such a path $A_0, \ldots, A_m$ exists, the partial word $w = u_m$ associated with the partial word path $A_0, \ldots, A_m$, as constructed in Definition 3, has h holes and satisfies $\mathrm{sub}_w(n) = S$.

We now have the terminology needed to prove the following lemma.

Lemma 7. *Let S be a set of words of length n and let $G = G_S = (V, E)$, where $V_0, \ldots, V_r$ is the decomposition of V with respect to $\rightharpoonup$. Then there exists a partial word w such that $S = \mathrm{sub}_w(n)$ and such that every subword of w of length $n-1$ is compatible with exactly one factor of w if and only if $V_0, \ldots, V_r$ is a partial word path including every edge.*

Proof. To show the backward implication, if w is the partial word associated with our partial word path, every subword of w of length $n-1$ occurs exactly once in w and $\mathrm{sub}_w(n) = S$. To show the forward direction, assume there is a partial word w such that each subword of w of length $n-1$ occurs exactly once, and $\mathrm{sub}_w(n) = S$. Let $A_0, \ldots, A_r$ be the partial word path associated with w, i.e., $A_i = \mathrm{sub}_{w[i..i+n-1)}(n-1)$. We want to prove that $A_i = V_i$.

Suppose towards a contradiction that this is not the case, and let j be the smallest index such that $A_j \neq V_j$. Then let $w' = w[j..|w|)$ and let $S' = \mathrm{sub}_{w'}(n)$. Let $G' = G_{S'} = (V', E')$. Then each word in $\mathrm{sub}_{w'}(n-1)$ occurs in w' exactly once. Since each word in $\mathrm{sub}_w(n-1)$ occurs in w exactly once, it follows that
$$\mathrm{sub}_{w'}(n-1) = \mathrm{sub}_w(n-1) - \bigcup_{i=0}^{j-1} A_i = \mathrm{sub}_w(n-1) - \bigcup_{i=0}^{j-1} V_i.$$
Let $V'_0, \ldots, V'_s$ be the decomposition of $V' = V - \bigcup_{i=0}^{j-1} V_i$ with respect to $\rightharpoonup$. By definition of decomposition, however, it is easy to see that $V'_i = V_{i+j}$. Furthermore, $A_j, \ldots, A_r$ is a partial word path in G'.

If $v \in A_j$ then v has no incoming edges in G', since if it has an incoming edge e then $A_j, \ldots, A_r$ must contain e. This implies v must occur in A_i for some $i > j$, contradicting the fact that each length $n-1$ subword of w' occurs exactly once in w'. Since no $v \in A_j$ has incoming edges, $A_j \subseteq V'_0 = V_j$. On the other hand, assume $v \in V'_0$, $v \in A_i$ for some $i > j$. This implies there is a path from some $u \in A_j$ to v. By definition of V'_0, this implies there is a path from v to u, contradicting the fact that u has no incoming edges. Therefore it must be that $V_j = V'_0 = A_j$. This is a contradiction, so our claim follows. □

Lemma 7 gives the following problem a membership in $\mathcal{P}$.

Proposition 2. *The problem of deciding whether a set S of words of length n can be represented by a partial word w, such that every subword of w of length $n-1$ occurs exactly once in w (in other words, every element in $\mathrm{sub}_S(n-1)$ is compatible with exactly one factor of w), is in $\mathcal{P}$.*

Proof. The proof reduces to checking that the graph G_S has the properties listed in Lemma 7. This check can clearly be done in polynomial time. $\square$

Proposition 2's proof amounts to checking, in polynomial time, properties that characterize the graph G_S corresponding to any S such that every element in $\text{sub}_S(n-1)$ is compatible with exactly one factor of a representing word. Lemma 7, which the proof of Proposition 2 depends on, uses that uniqueness property in a very strong way. So the cases not covered by Proposition 2 lead to entirely new challenges.

6. Other results on representability

In this section, we give other results on representing sets of words of equal length by (partial) words. In Section 6.1, we prove that for every non-negative integer h, there exists a set of words of equal length such that (1) it is h-representable and (2) the partial word representing it is unique. As a consequence, for any non-negative integers h_1 and h_2, we get that h_1-REP is not a subset of h_2-REP, so there cannot be a hierarchy of representability. In Section 6.2, for any set S that might not be representable, we give a lower bound on the size of a subset T of S that is representable. Finally in Section 6.3, we formulate a necessary and sufficient condition for the existence of a right-sided infinite word representing a given set of words of equal length.

6.1. h_1-REP versus h_2-REP

How does h_1-REP relate to h_2-REP when $h_1 \neq h_2$? Can we have h_1-REP $\subseteq h_2$-REP? As the next proposition shows, the answer is no.

Proposition 3. *Let A be a fixed alphabet with $|A| > 2$, and let h be a non-negative integer. Then if $n > h+2$, there exists a set S such that $S = \text{sub}_w(n)$ for some partial word w with exactly h holes, but such that there is no other partial word w' satisfying $\text{sub}_{w'}(n) = \text{sub}_w(n)$.*

Proof. Let $A = \{a_0, a_1, \ldots, a_{k-1}\}$ with $|A| = k > 2$, and let $w_n = \diamond^h a_0^{n-h-1} a_1$. Furthermore, let $S' = \text{sub}_{w_n}(n)$. We claim S' is the set S we want. We know w_n has exactly h holes. Write $G_n = G_{S'} = (V, E)$. We can decompose $V = V_0 \cup \cdots \cup V_r$ as usual. Then it is easy to see that $r = 1$, where

$$V_0 = \{ua_0^{n-h-1} \mid |u| = h\}$$

and

$$V_1 = \{ua_0^{n-h-1}a_1 \mid |u| = h-1\}.$$

Assume that w' is a partial word satisfying $\text{sub}_{w'}(n) = S'$, and that $A_0, A_1, \ldots, A_m$ is the associated partial word path in G_n. Write $A_0 = \text{sub}_v(n-1)$ for some partial word v with $|v| = n-1$. Note we need that $\text{sub}_v(n-1) = A_0 \subseteq V$. We know that $|A_0| = k^h - k^{h-1} > k^{h-1}$, so v must have at least h holes. On the other hand, $|V| = k^h \geq |A_0| = |\text{sub}_v(n-1)|$, so v must contain at most h holes. Thus v contains exactly h holes. It is easy to see that this implies $v = w_n[0..n-1)$.

Furthermore, since every word in S' must end in the letter a_1, every element in A_1 must end with a_1. This implies A_1 satisfies Eq. (2), and so $w'[0..n) = w_n[0..n-1)a_1 = w_n$. Finally, note that $m = 1$. To see this, assume towards a contradiction that $A_2 \neq \emptyset$. Then if $v' \in A_2$, we must have that $v'[|v'|-2] = a_1$. However there is no vertex u' in V with $u'[|u'|-2] = a_1$, since we always have $u'[|u'|-2] = a_0$. Thus, $w' = w_n$. $\square$

The construction in the proof of Proposition 3 implies that the 9-element set $\{uaab \mid u \in \{a, b, c\}^* \text{ and } |u| = 2\}$ is uniquely represented by the partial word $\diamond\diamond aab$.

In particular, Proposition 3 implies the following corollary.

Corollary 2. *If $h_1 \neq h_2$, there exists a word w_1 with h_1 holes such that if w_2 has h_2 holes then $\text{sub}_{w_1}(n) \neq \text{sub}_{w_2}(n)$.*

6.2. Approximating REP

The above was concerned with finding a partial word w such that $\text{sub}_w(n) = S$, for a given set S of words of length n. We might instead try to find the largest subset T of S such that $\text{sub}_w(n) = T$ for some w, i.e., to find a partial word w that is as close as possible to representing S.

Fixing an alphabet A of size k, if $u, v \in A^n$, $d(u, v)$ denotes the distance from u to v in G_{A^n} if we treat it as an undirected graph. We need some technical lemmas.

Lemma 8. *Let $G = (V, E)$ be a digraph where every vertex has k incoming edges and k outgoing edges. If $T \subseteq V$, then*

$$|\{v \in V \mid d(v, T) \leq m\}| \leq |T|(2k)^m$$

where $d(v, T)$ is the maximum of the $d(v, t)$'s with $t \in T$.

Proof. We proceed by induction on m. If $m = 0$, the claim clearly holds. If the claim holds for m, let $U = \{v \in V \mid d(v, T) \le m\}$. Then

$$|\{v \in V \mid d(v, T) \le m + 1\}| = |\{v \in V \mid d(v, U) \le 1\}| \le 2k|U| \le |T|(2k)^{m+1}$$

where the inequality follows from the fact that every vertex in G has at most $2k$ neighbours. □

Lemma 9. *Let S be a set of words of length n over an alphabet A of size k, and set $r = k^n - |S|$. Let $T = A^n - S$. If w_1 and w_2 are vertices in G_S such that $m_i = d(w_i, T) = \max_{t \in sub_T(n-1)} d(t, w_i) > \log_k(nr)$, then w_1 and w_2 are in the same weakly connected component of G_S. In fact, there is a path from w_1 to w_2 in G_S.*

Proof. Note that G_S can be viewed as a subgraph of $G_n = G_{A^n}$. Also note that $|T| = r$ by definition. If $m_1, m_2 \ge m > \log_k(nr)$, then there are $k^m > nr$ words of length m over the alphabet A. Furthermore, every word in T has at most $n - m < n$ subwords of length m. This implies $|\text{sub}_T(m)| \le nr < |A^m|$. Thus there exists a word $w \in A^m$ such that w is not a subword of any $t \in T$. In particular, w is not a subword of any element in $\text{sub}_T(n-1)$. Thus to see that w_1 and w_2 are in the same weakly connected component of G_S, consider the sequence

$$\begin{aligned}
&w_1, w_1[1..n)w[0..1), \ldots, w_1[m..n)w[0..m) = w_1[m..n)ww_2[0..0),\\
&w_1[m+1..n)ww_2[0..1), \ldots, w_1[n..n)ww_2[0..n-m) = w[0..m)w_2[0..n-m),\\
&w[0..m-1)w_2[0..n-m+1), \ldots, w[0..0)w_2[0..n) = w_2.
\end{aligned}$$

Note that no element in the above sequence is an element in $\text{sub}_T(n-1)$. This follows since the distance between w_1 and $w_1[j..n)w[0..j)$ for $j < m \le m_1$ is at most j, so $w_1[j..n)w[0..j) \notin \text{sub}_T(n-1)$. A similar argument works for $w[0..m-j)w_2[0..n-m+j)$. All other elements in the sequence have w as a subword, so cannot be elements in $\text{sub}_T(n-1)$. It fact, it is easy to see the sequence is actually a path in G_S from w_1 to w_2. □

Lemma 10. *Let S be a set of words of length n over an alphabet A of size k, and set $r = k^n - |S|$. Then there is a strongly connected component in G_S containing at least $k^{n-1} - r^3n^2$ vertices.*

Proof. Let $T = A^n - S$ and let G' be the strongly connected component of G_S that includes all v such that $\max_{t \in \text{sub}_T(n-1)} d(t, v) > \log_k(nr)$. Then note by Lemma 8 that

$$\{v \in A^{n-1} \mid \max_{t \in \text{sub}_T(n-1)} d(t, v) \le \log_k(nr)\}$$

contains at most $r(2k)^{\log_k(rn)} = r^2n(rn)^{\log_k 2} \le r^3n^2$ elements, thus the result follows. □

Proposition 4. *Let S be a set of words of length n over an alphabet of size k, and set $r = k^n - |S|$. Then there exists $T \subseteq S$ such that $T = sub_w(n)$ for some w, and such that $|T| \ge k^{n-1} - r^3n^2$.*

Proof. Let $G' = (V', E')$ be as in the proof of Lemma 10. Then $k^{n-1} - r^3n^2 \le |V'| \le |E'|$, so let $T = E'$. Since G' is strongly connected it follows there is a path that includes every edge in E'. This path corresponds to a word w such that $T = E' = \text{sub}_w(n)$. □

Proposition 4 implies that if S is almost equal to A^n, then S has a subset T that contains almost all elements in S and such that $T = \text{sub}_w(n)$ for some full word w. As an example, the set $S = \{aaa, aab, aba, baa, bab, bbb\}$ is not representable; however, the subset $T = \{aaa, aab, aba, baa, bab\}$ of S is representable by the word $aaababaa$.

The set A^n being representable for any positive integer n, if $S \subseteq A^n$ then there exists a minimal representable set T such that $S \subseteq T \subseteq A^n$. Some of the ideas presented in this section could be used to give a bound on the size of a representable superset T of a set S that might not be representable.

6.3. Representability by infinite words

We also state the following proposition concerning representing a set of words of equal length by an infinite word.

Proposition 5. *Let S be a set of words of length n. Then there exists a right-sided infinite word w such that $sub_w(n) = S$ if and only if there exist finite words w_1 and w_2, $w_1 \ne w_2$ and $sub_{w_1}(n) = sub_{w_2}(n) = S$, such that either $w_1[|w_1| - n..|w_1|) \ne w_2[|w_2| - n..|w_2|)$ or $w_1[|w_1| - n..|w_1|) = w_2[|w_2| - n..|w_2|) = a^n$ for some $a \in A$.*

Proof. First assume there exists a right-sided infinite word w such that $\text{sub}_w(n) = S$. Then there exists $i > 0$ such that if $w_1 = w[0..i)$, we get that $\text{sub}_{w_1}(n) = S$. Furthermore, if $w_2 = w[0..i+1)$ then $\text{sub}_{w_2}(n) = S$. Assume $w_1[|w_1| - n..|w_1|) = w_2[|w_2| - n..|w_2|)$. This implies $w[i-n..i) = w[i-n+1..i+1)$. This implies w_1 and w_2 both have period one, so $w_1[|w_1| - n..|w_1|) = a^n$ and $w_2[|w_2| - n..|w_2|) = b^n$ for some $a, b \in A$. By assumption $w_1[|w_1| - n..|w_1|) = w_2[|w_2| - n..|w_2|)$ so we get that $a = b$.

For the other direction, assume that there exist finite words w_1, w_2 such that $w_1 \ne w_2$ and $\text{sub}_{w_1}(n) = \text{sub}_{w_2}(n) = S$. If $w_1[|w_1| - n..|w_1|) = w_2[|w_2| - n..|w_2|) = a^n$ for some $a \in A$, then let $w = w_1a^\omega$. By construction

$$S = \text{sub}_{w_1}(n) = \text{sub}_{w_1}(n) \cup \{a^n\} = \text{sub}_{w_1}(n) \cup \text{sub}_{a^\omega}(n) = \text{sub}_w(n).$$

Next assume that $w_1[|w_1| - n..|w_1|) \neq w_2[|w_2| - n..|w_2|)$. Then there exists j_1 such that $w_2[j_1..j_1 + n) = w_1[|w_1| - n..|w_1|)$ since $w_1[|w_1| - n..|w_1|) \in S = \mathrm{sub}_{w_2}(n)$. Similarly, there exists j_2 such that $w_1[j_2..j_2 + n) = w_2[|w_2| - n..|w_2|)$. Then let

$$w = w_1(w_2[j_1 + n..|w_2|)w_1[j_2 + n..|w_1|))^{\omega}.$$

It is then easy to verify that $\mathrm{sub}_w(n) = S$ as above, proving the claim. □

To illustrate the last part of the proof, let $S = \{aaa, aab, aba, baa, bab\}$. Consider $w_1 = aaababaa$ and $w_2 = baaabab$, two representing words for S. Here $j_1 = 0$ and $j_2 = 3$, and we can check that

$$w_1(w_2[j_1 + 3..|w_2|)w_1[j_2 + 3..|w_1|))^{\omega} = aaababaa(ababaa)^{\omega}$$

is a right-sided infinite word representing S.

7. Conclusion

We provided a polynomial time algorithm to solve h-Rep, that is, given a set S of words of length n, our algorithm decides, in polynomial time with respect to the input size $n|S|$, whether there exists a partial word with h holes that represents S. Our algorithm also computes such a representing partial word. To find a more tractable algorithm is an open problem.

Whether or not Rep is in $\mathscr{P}$ is also an open problem. We have some hope that the following proposition might be useful in understanding Rep. Letting S be a set of words of length n, set

$$\mathrm{Comp}(S) = \{u \mid u \text{ is a partial word and every completion of } u \text{ is in } S\}.$$

The set $\mathrm{Comp}(S)$ is important because if $\mathrm{sub}_w(n) = S$, then every factor of length n of w is an element of $\mathrm{Comp}(S)$.

Proposition 6. *Assume $|A| > 1$. If S is a set of words of length n, then $|Comp(S)| \leq |S|^2$. Furthermore, Comp(S) can be computed in $O(n|S|^4)$ time.*

Proof. Assume that $u \in \mathrm{Comp}(S)$. Choose $a, b \in A$, $a \neq b$. Let $\hat{u_a}$ (resp., $\hat{u_b}$) be the word we get by replacing all the ⋄'s in u with a (resp., b). Then $\hat{u_a}, \hat{u_b} \in S$, by definition of $\mathrm{Comp}(S)$. Furthermore, u is the partial word with the least number of holes such that $u \subset \hat{u_a}$ and $u \subset \hat{u_b}$, in other words, u is the greatest lower bound of $\hat{u_a}$ and $\hat{u_b}$. Therefore,

$$\mathrm{Comp}(S) \subseteq \{u \mid u \text{ is the greatest lower bound of } (u_1, u_2) \in S^2\}.$$

However, the latter set has cardinality at most $|S^2| = |S|^2$, so $|\mathrm{Comp}(S)| \leq |S|^2$. Therefore, all we need to do in order to compute $\mathrm{Comp}(S)$ is to iterate through $(u_1, u_2) \in S^2$ (which takes $|S|^2$ iterations). In each iteration we calculate u, the greatest lower bound of u_1 and u_2. We then iterate through all completions of u until either we have checked them all (in which case, we add u to $\mathrm{Comp}(S)$), or until we find one that is not in S (in which case, u is not in $\mathrm{Comp}(S)$). This produces $\mathrm{Comp}(S)$. Furthermore, each iteration takes $O(n|S|^2)$ time, so the algorithm takes $O(n|S|^4)$ time. □

Proposition 6's proof is a step towards characterizing the sets S of words of length n that are representable since, as mentioned above, every factor u of length n of any representing partial word belongs to $\mathrm{Comp}(S)$, i.e., every completion of u is in S.

References

[1] J. Berstel, Recent results on extensions of Sturmian words, International Journal of Algebra and Computation 12 (2002) 371–385.
[2] J. Berstel, L. Boasson, Partial words and a theorem of Fine and Wilf, Theoretical Computer Science 218 (1999) 135–141.
[3] F. Blanchet-Sadri, Algorithmic Combinatorics on Partial Words, Chapman & Hall/CRC Press, Boca Raton, FL, 2008.
[4] F. Blanchet-Sadri, D. Allums, J. Lensmire, B.J. Wyatt, Constructing minimal partial words of maximum subword complexity, in: JM 2012, 14th Mons Days of Theoretical Computer Science, Université catholique de Louvain, Belgium, 2012.
[5] F. Blanchet-Sadri, B. Chen, L. Manuelli, S. Munteanu, J. Schwartz, S. Stich, Representing languages by infinite partial words. Preprint, 2011.
[6] F. Blanchet-Sadri, J. Lensmire, On minimal Sturmian partial words, Discrete Applied Mathematics 159 (2011) 733–745.
[7] F. Blanchet-Sadri, J. Schwartz, S. Stich, B. J. Wyatt, Binary de Bruijn partial words with one hole, in: J. Kratochvil, et al. (Eds.), TAMC 2010, 7th Annual Conference on Theory and Applications of Models of Computation, Prague, Czech Republic, in: Lecture Notes in Computer Science, vol. 6108, Springer-Verlag, Heidelberg, 2010, pp. 128–138.
[8] F. Blanchet-Sadri, S. Simmons, Deciding representability of sets of words of equal length, in: M. Kutrib, N. Moreira, R. Reis (Eds.), DCFS 2012, 14th International Workshop on Descriptional Complexity of Formal Systems, Braga, Portugal, in: Lecture Notes in Computer Science, vol. 7386, Springer-Verlag, Berlin, Heidelberg, 2012, pp. 103–116.
[9] J. Cassaigne, Special factors of sequences with linear subword complexity, in: Developments in Language Theory II, Magdeburg, Germany, World Scientific, NJ, 1996, pp. 25–34.
[10] M. Crochemore, C. Hancart, T. Lecroq, Algorithms on Strings, Cambridge University Press, 2007.
[11] A. E. Frid, On factor graphs of DOL words, Discrete Applied Mathematics 114 (2001) 121–130.
[12] J. L. Gross, J. Yellen, Handbook of Graph Theory, CRC Press, 2004.
[13] M. Lothaire, Combinatorics on Words, Cambridge University Press, Cambridge, 1997.
[14] M. Lothaire, Algebraic Combinatorics on Words, Cambridge University Press, Cambridge, 2002.
[15] M. Lothaire, Applied Combinatorics on Words, Cambridge University Press, Cambridge, 2005.