未決定文字列における一意単語の検索の困難さ

Dominik Köppl and Jannik Olbrich

Abstract

未決定文字列は単純な文字列を一般化したものである．未決定文字列の中に，各位置に 1 つの文字が格納されているだけでなく，複数の文字の選択肢が認められている．さらにテキスト位置ですべての可能性を列挙するモデル化に適用できる．実用化のため，パターン照合や未決定文字列の類似性を測定するなど，単純な文字列に対して既知の技術を未決定文字列に応用し，様々なクエリの種類に応じることに関心が向けられているが，遺伝子データの解析において頻繁に利用されている一意単語の検索に関しては未検証である．単純な文字列において，最短の一意単語の検索は既に線形時間で出力可能であるが，未決定文字列上での計算は NP 困難であることを示す．

NP 困難・未決定文字列・一意単語・解集合プログラミング

1 はじめに

完全な情報を必要とする古典的なアルゴリズムの適応を妨げる要因の一つは不確実性である．本論は，すべての可能性を明示的に示すことで不確実性をモデル化する．このようなモデル（未決定文字列）は，遺伝子配列などの生物学的データを扱う際によく使われている．未決定文字列は，任意の位置に複数の代替文字を持つことができる文字列をモデル化する．未決定文字列の各要素は入力アルファベットの部分集合となり，これを**記号**と呼ぶ．記号は，DNA や RNA 配列における未決定の核酸を表現するために IUPAC 表記法 [21] を一般化したものである．図 1 は，未決定文字列の例を示している．

$$\widetilde{T} = \mathtt{A}\left\{\begin{matrix}\mathtt{A}\\ \mathtt{C}\end{matrix}\right\}\mathtt{C}\left\{\begin{matrix}\mathtt{G}\\ \mathtt{T}\end{matrix}\right\}, \quad \mathcal{L}(\widetilde{T}) = \{\mathtt{AACG}, \mathtt{AACT}, \mathtt{ACCG}, \mathtt{ACCT}\}.$$

Figure 1: 未決定文字列 $\widetilde{T}$ の例．$\widetilde{T}$ の言語 $\mathcal{L}(\widetilde{T})$ は右に示されている．この言語は，未決定文字列によって表される文字列の集合である．1 つの文字 c のみを格納する記号は，$\{c\}$ または単に c として書かれる．

2　関連研究

本論のテーマに関連する研究は，未決定文字列と極小一意単語に特化した研究である．以下では関連研究を紹介する．

2.1　未決定文字列

未決定文字列に関する研究の大部分は，パターン照合，構造的特性や規則性，必ずしも自己索引ではない索引からの未決定文字列の再構築，および 2 つの未決定文字列の比較に関心が向けられている．

パターン照合　　未決定文字列に対しては，Boyer–Moore の適応版 [17]，ShiftAnd と Boyer–Moore–Sunday の組み合わせ [31]，および KMP を元した方法 [28] が提案されている．

構造的特性　　未決定文字列の構造的特性については，カバーおよび/またはシードの計算 [7, 9]，Lyndon 分解の拡張 [13] が知られている．[22] は未決定文字列の新しい表現モデルを提案した．

再構築　　データ構造からの未決定文字列の再構築も活発な研究分野である．再構築は，ボーダー配列，接尾辞配列，および LCP 配列から構築が可能である [27]．接頭辞配列に基づくグラフから [5]，または頂点がテキスト位置で辺が一致する文字を持つテキスト位置であるグラフからも再構築が行われている [16]．[12] は，配列が単純な文字列または未決定文字列の接頭辞配列であるかどうかの特徴を示した．最後に，[10] は，未決定文字列の接頭辞配列と無向グラフおよびボーダー配列との関係を研究した．

2.2　極小一意単語

極小一意単語　　極小一意単語（MUS）の計算問題は Pei らにより導入された [29]．この問題は，シーケンスアラインメント [4]，ゲノム比較 [15]，および系統樹の構築 [11] などに応用されてきた．より一般的な極短一意部分文字列（SUS）は，局所的な極小性のみを求めるもので，テキスト位置または区間をカバーすることで表現できる．SUS の計算に対する研究注目は，時間量および領域量に関する改善につながった [19, 32, 20, 23, 25]．両方の量を均衡する解決策も提供された [14, 8]．また，スライディングウィンドウ内での計算 [26] や連超圧縮された文字列上での計算 [23] など，異なる設定が提案されている．理論的な保証を得るために，[24] は，与えられたクエリ位置をカバーする SUS の最大数を研究した．近年，SUS 計算に関するサーベイ記事が公開されている [3]．最後に，与えられた範囲内で SUS を計算するための変形が存在する [1, 2] または k ミスマッチを持つ SUS を計算する [18, 30, 6]．

3　背景知識

まず，文字列の基本概念を紹介し，その後，未決定文字列への一般化について表現する．

文字列　　アルファベットを Σ とする．Σ^* の要素は（単純な）**文字列**と呼ばれる．文字列 T が与えられたとき，T の i 番目の文字は $T[i]$ と表される（整数

$i \in [1..|T|]$ の場合），ここで $|T|$ は T の長さを表す．整数 i と j が $1 \leq i \leq j \leq |T|$ を満たすとき，位置 i から始まり位置 j で終わる T の部分文字列は $T[i..j]$ と表される．すなわち，$T[i..j] = T[i]T[i+1]\cdots T[j]$ である．T の部分文字列 P は，$P \neq T$ の場合に**真の部分文字列**と呼ばれる．

未決定文字列　単純な文字列の以下の拡張を研究する．そのために，アルファベット Σ の**文字**と，次の 3 種類の一般化のいずれかに属する文字列の**記号**を区別する．**未決定文字列**は，記号が Σ の空でない部分集合から引かれる文字列 $\widetilde{S}[1..n]$ である．すなわち，$\emptyset \neq \widetilde{S}[i] \subset \Sigma$ である．$r = r(\widetilde{S}) = \max_i |\widetilde{S}[i]|$ は最大の記号のサイズを示す．

4　一意単語の困難性

本章では，3-SAT の n 変数を持つ決定問題に対して，MUS の長さが最大 n であるかどうかに帰着する．3-SAT は NP 困難であると知られ，以下では 3-SAT から MUS 問題へ帰着することを示すことで，MUS 問題も同様に NP 困難となる．3-SAT の入力は，連言標準形（CNF: conjunctive normal form）の式 F である．すなわち，F は一連の節 C_i を連言（AND）で結合し，各節 C_i は 3 つのリテラルの選言である．n 個の変数 $x_1, \ldots, x_n$ が順序付けされていると仮定する．方針は，各節 C_i に対して C_i を満たさないすべての変数割り当てを指定することである．このような割り当ての集合は，未決定文字列 $\widetilde{S}$ にすべての変数を線形に書き込むことで表現できる．C_i で使用されていない変数は任意の値を取ることができる．構築により，これらの不足な割り当ての和集合をすべての可能な割り当てから引くことで，すべての充足する割り当てが得られる．すべての割り当ての集合を $\widetilde{S}$ の不定部分文字列として表現するならば，不可解な割り当ては $\widetilde{S}$ において，少なくとも 2 回現れる．その一方で，解を持つ割り当ては 1 回しか現れない．F が充足可能であるとき，かつそのとき限り，最短 MUS は F の充足する割り当てと一致することを以下で示す．

まず，単純な文字列に基づく SUS の形式的な定義から始める．単純な文字列 T の部分文字列 U は**一意単語** とは，T 中の出現が一つしかない．つまり，T 中に U と一致する部分文字列が他には存在しない場合である．特に，すべての U の部分文字列は T の一意単語ではない場合，U を**極小一意部分文字列** (MUS) と呼ぶ．

未決定文字列 $\widetilde{S}$ において，文字列 $P \in \Sigma^*$ が，$\widetilde{S}$ のテキスト位置 i から出現するとは，$\widetilde{S}$ の言語に属する文字列 X が存在し，$X[i..i+|P|-1] = P$ であることを意味する．P が $\widetilde{S}$ で一意であるとは，その出現の位置が一意に決定されるということである．さらに，P が**極小一意部分文字列** *(MUS)* であるとは，P の任意の真部分文字列 X が $\widetilde{S}$ において少なくとも 2 回出現するということである．

問題 1 (一意単語問題)．未決定文字列 $\widetilde{S}$ と整数 k が与えられたとき，k-一意単語問題は，長さが最大 k の一意単語が $\widetilde{S}$ に存在するかどうかを決定することである．

n 個の変数と m 個の節を持つ 3-CNF $F = C_1 \wedge \cdots \wedge C_m$ が与えられたとする．アルファベット $\{0, 1, \$\}$ 上の未決定文字列 $\widetilde{S}$ を構築し，F が充足可能である場合に限り，長さが最大 n の MAW が $\widetilde{S}$ に存在することを示す．変数は $x_1, \ldots, x_n$ と仮定する．

$$\widetilde{T_j} = \left\{ \begin{matrix} 0 \\ 1 \end{matrix} \right\}^{a-1} \{ v_a \} \left\{ \begin{matrix} 0 \\ 1 \end{matrix} \right\}^{b-a-1} \{ v_b \} \left\{ \begin{matrix} 0 \\ 1 \end{matrix} \right\}^{c-b-1} \{ v_c \} \left\{ \begin{matrix} 0 \\ 1 \end{matrix} \right\}^{n-c}.$$

Figure 2: 定理 1 を証明するために定義した $\widetilde{T_j}$.

$$\widetilde{S} = \widetilde{T_1} \{ \$ \} \ldots \{ \$ \} \widetilde{T_m} \{ \$ \} \left\{ \begin{matrix} 0 \\ 1 \end{matrix} \right\}^{n} \{ \$ \} \left\{ \begin{matrix} 0 \\ 1 \end{matrix} \right\}^{n-1} \{ \$ \} \left\{ \begin{matrix} 0 \\ 1 \end{matrix} \right\}^{n-1}.$$

Figure 3: 定理 1 を証明するために定義した $\widetilde{S}$.

節 $C_j = (\ell_a \vee \ell_b \vee \ell_c)$ のリテラルを ℓ_a, ℓ_b, ℓ_c とし，ℓ_i が変数 x_i のリテラルであるとする（$i \in a, b, c$）． 各 i に対して，ℓ_i が正の場合（$\ell_i = x_i$）は $v_i = 0$, それ以外の場合（$\ell_i = \neg x_i$）は $v_i = 1$ と設定し，v_i に基づいて図 2 の未決定文字列 $\widetilde{T_j}$ を構築する．

$\widetilde{S}$ を図 3 のように定義する．各 $\widetilde{T_j}$ は n の記号から成り立つので，$\widetilde{S}$ は $\mathcal{O}(nm)$ の記号を持つ，かつ，$r(\widetilde{S}) = 3$ なので，$\widetilde{S}$ を $\mathcal{O}(nm)$ 領域で表現できる．

ここで，$\widetilde{S}$ について以下のような観察を行う．

- 長さ n のアルファベット $\{0,1\}$ の文字列，すなわちビット列は少なくとも 1 回は現れる．具体的には，$\widetilde{S}[1+(n+1)m]$ 中に $\left\{ \begin{matrix} 0 \\ 1 \end{matrix} \right\}^{n}$ という記号がある．
- 長さ n 未満のビット列は複数回現れる．
- 長さ n 以下の部分文字列には，一つ以上の '$' は含まれない．
- '$' を含む長さ n 以下の文字列は複数回現れる．
- 長さ n のビット列 $B[1..n]$ は，$B[i] = 1$ のとき，かつそのとき限り，x_i が真であることを符号化されている．このようなビット列 B は，F を偽にする割り当てを符号化している．具体的には，B は C_i を偽にする割り当てを持つ場合に，$\widetilde{T_i}$ に現れる．

したがって，長さ n 未満の部分文字列には一意なものが存在しない．そして，長さ n の部分文字列が一意であることは，F が充足可能である場合に限られる．

定理 1. 一意単語 は $\sigma \geq 3$ および $r \geq 3$ の場合に NP 困難である．

例 1. 以下の 3-CNF に考える．

$$F = (x_1 \vee \neg x_2 \vee x_3) \wedge (x_2 \vee \neg x_3 \vee x_4),$$

ただし $n = 4$ は変数 x_1, x_2, x_3, x_4 の個数を示す．CNF F から上記のように記述された操作手順で作った $\widetilde{S}$ が 図 4 に表示されている．

文字列 1000($x_1 = 1, x_2 = x_3 = x_4 = 0$) は，$\widetilde{S}$ にただ 1 回しか現れないので，MUS である．$\{0,1\}$ のアルファベットから成り立つ長さ 3 の文字列は $\widetilde{S}$ に 2 回現れる．他の長さ 4 の文字列，例えば 0100 や 00$1 は 2 回現れる．よって，1000 は $\widetilde{S}$ の中で最も短い MUS の一つである．

$$\{0\}\{1\}\{0\}\left\{\begin{matrix}0\\1\end{matrix}\right\}\{\$\}\left\{\begin{matrix}0\\1\end{matrix}\right\}\{0\}\{1\}\{0\}\{\$\}\left\{\begin{matrix}0\\1\end{matrix}\right\}^4\{\$\}\left\{\begin{matrix}0\\1\end{matrix}\right\}^3\{\$\}\left\{\begin{matrix}0\\1\end{matrix}\right\}^3.$$

Figure 4: 例 1 で利用される $\widetilde{S}$.

4.1 SAT 定式化

以下では，未決定文字列において指定された長さ x の一意単語を計算するための SAT 定式化を提示する．SAT 定式化は，$x \in [1..n]$ に対する最適化目的を持つ MAX-SAT 定式化に変換できる．定式化のために，未決定文字列 $\widetilde{T}[1..n]$ の記号を文字のリストとして表現し，$\widetilde{T}[i]$ をリストとする．$\widetilde{T}[i][k]$ はリスト $\widetilde{T}[i]$ の k 番目の文字を表し，$|\widetilde{T}[i]|$ は $\widetilde{T}[i]$ に格納されている文字の数を表す．ブール変数の真偽値を整数 1 と 0 として解釈し，和のような表現を使用できるようにする．解答を長さ x の文字列 $X[1..x] \in \Sigma^x$ としてモデル化する．長さ x は（決定問題として）与えられるか，最短長を得るための最適化対象となる．x を次のように n 個のブール変数 x_i でモデル化できる．

$$\sum_{i=1}^{n} x_i = 1. \qquad \text{(LENX)} \qquad \min i \in [1..n] : x_i = 1. \quad \text{(MINX)}$$

以下では，定義されていない変数は偽（設定されていない）と見なす．

X の長さ $x \in [1..n]$ の一意な部分文字列を見つけるために，まず X の開始位置 $p \in [1..n]$ を選択する．このとき，n 個の真偽変数 p_i を用いて，$\widetilde{T}$ 中に X の唯一の開始位置を示す．すなわち，

$$\sum_{i=1}^{n} p_i = 1. \qquad \text{(POSP)}$$

X の各文字は，すべての $\ell \in [1..x]$ に対して $X[\ell] \in \widetilde{T}[p+\ell-1]$ となるように選択される．そのため，$X'[\ell, k]$ という二次元テーブルを作成し，$X'[\ell, k] = 1$ が $X[\ell] = \widetilde{T}[p+\ell-1][k]$ であるときに限る．これは，次の制約でモデル化する．

$$\forall \ell \in [1..x] : \sum_{k=1}^{|\widetilde{T}[p+\ell-1]|} X'[\ell, k] = 1. \qquad \text{(SELX)}$$

$\{\mathcal{O}(x), \mathcal{O}(r)\}$

灰色の波括弧は，生成される節の数に対する漸近的な上限と，各節の最大サイズに対する漸近的な上限の 2 つを示している．ここでは，各 X の位置に対してサイズ $O(r)$ の節を生成する．

X が一意しているかどうかを確認するためには，p 以外，すべてのテキスト位置 $t \in [1..n-x+1] \setminus \{p\}$ において X の出現が始まらないことを確認する必要がある．言い換えれば，各 t に対して X の位置 $\ell \in [1..x]$ が存在し，$X[\ell]$ が $\widetilde{T}[t+\ell-1]$ に含まれないことを確認する．これを式として表現するために，こ

れらの不一致をモデル化するブール変数 $M[k,t,\ell]$ の三次元グリッドを作成する．$X[\ell] \neq \widetilde{T}[t+\ell-1][k]$ の場合，$M[k,t,\ell]$ を真に設定する．ここで，$\widetilde{T}[t+\ell-1][k]$ はリスト $\widetilde{T}[t+\ell-1]$ の k 番目の文字を示す（存在する場合）．

$$\begin{aligned} \forall \ell \in [1..x], t \in [1..n-x+1], k \in [1..|\widetilde{T}[t+\ell-1]|] : \\ X[\ell] \neq \widetilde{T}[t+\ell-1][k] \implies M[k,t,\ell]. \end{aligned} \tag{M}$$

$\{\mathcal{O}(xnr), \mathcal{O}(1)\}$

$r = r(\widetilde{T}) \leq \sigma$ はリスト $\widetilde{T}[i]$ が格納できる文字の最大数である．次に，$M[k,t,\ell]$ を $M'[t,\ell]$ に縮小し，$X[\ell]$ がリスト $\widetilde{T}[t+\ell-1]$ のすべての文字と一致しない場合にのみ $M'[t,\ell]$ を真に設定する．すなわち，

$$\begin{aligned} \forall \ell \in [1..x], t \in [1..n-x+1] : \\ \sum_{k=1}^{|\widetilde{T}[t+\ell-1]|} M[k,t,\ell] = |\widetilde{T}[t+\ell-1]| \implies M'[t,\ell]. \end{aligned} \tag{M'}$$

$\{\mathcal{O}(xn), \mathcal{O}(r)\}$

したがって，$M'[t,\ell]$ は $X[\ell]$ が $\widetilde{T}[t+\ell-1]$ に含まれない場合にのみ真である．最後に，X の位置 ℓ が存在し，$X[\ell]$ を $\widetilde{T}[t+\ell-1]$ のいずれの文字とも一致させることができないことを要求する．すなわち，

$$\forall t \in [1..n-x+1] \setminus \{p\} : \sum_{\ell=1}^{x} M'[t,\ell] \neq 0. \tag{CONS}$$

$\{\mathcal{O}(n), \mathcal{O}(x)\}$

もし Eq. (CONS) が成り立つならば，X は一意単語でなければならない．合計で，$2n + rx$ の選択可能なブール変数がある．節の数は $\mathcal{O}(xnr)$ である．同じ上限が，すべての節のサイズの合計である CNF のサイズにも適用される．

リスト 1 は，上記の式のいずれかでコメントされた各行を含む解集合プログラミング（ASP: answer set programming）でのエンコーディングを示している．ランダムに生成された文字列に対する評価は 表 1 に示されている．評価は，Ubuntu 22.04 を搭載した Intel Xeon Gold 6330 CPU 上で，ASP の符号化を解釈するための clingo バージョン 5.7.1 を使用して実行された．

5 おわりに

本論では，未決定文字列における一意単語の困難性について検証した．極小一意単語の検索に必要な計算量を判断するため，既存研究において取り扱われることの少ない最短の一意単語に注目した．結果として，極小一意単語における最短の一意単語の検索は未決定文字列上で NP 困難であることを示した．

コード 1: MUS を計算する ASP の符号化． 最後の行（コマンド `show`）は出力用である．

```
1 1 { len(X) : X = 1..n } 1. %(LENX)
2 #minimize { X : len(X) }. %(MINX)
3 1 { pos(P) : P = 1..n } 1. %(POSP)
4 1 { x(L,C) : t(K,P+L-1,C) } 1 :- len(X), pos(P), L = 1..X. %(
    ↪ SELX)
5 m(K,T,L) :- t(K,T+L-1,C), x(L,D), C != D, len(X), T=1..n-X+1. %(
    ↪ M)
6 m(T,L) :- r { m(K,T,L) : K=1..r }, len(X), L = 1..X, T=1..n-X+1.
    ↪  %(M')
7 :- { m(T,L) } 0, len(X), pos(P), T=1..n-X+1, T != P. %(CONS)
8 #show x/2.
```

n	時間	x
10	0.03	2
100	171.60	3
150	580.68	4
200	1376.11	4
250	2708.00	5
260	3079.78	5
270	3428.21	5

Table 1: ランダムに生成された未決定文字列 $\widetilde{S}$ に対する最短 MUS の計算．$\sigma = 4$ および $r(\widetilde{S}) = 2$．時間は秒単位で，n は計算の入力として取られた $\widetilde{S}$ の接頭辞 $\widetilde{T}$ の長さである．

謝辞

本研究は JSPS 科研費 JP23H04378 の助成と山梨県若手研究者奨励事業費補助金 2291 の支援を受けたものである．

References

[1] Paniz Abedin, Arnab Ganguly, Solon P. Pissis, and Sharma V. Thankachan. Range shortest unique substring queries. In *Proc. SPIRE*, volume 11811 of *LNCS*, pages 258–266, 2019.

[2] Paniz Abedin, Arnab Ganguly, Solon P. Pissis, and Sharma V. Thankachan. Efficient data structures for range shortest unique substring queries. *Algorithms*, 13(11):276, 2020.

[3] Paniz Abedin, M. Oguzhan Külekci, and Sharma V. Thankachan. A survey on shortest unique substring queries. *Algorithms*, 13(9):224, 2020.

[4] Boran Adas, Ersin Bayraktar, Simone Faro, Ibraheem Elsayed Moustafa, and M. Oguzhan Külekci. Nucleotide sequence alignment and compression via shortest unique substring. In *Proc. IWBBIO*, volume 9044 of *LNCS*, pages 363–374, 2015.

[5] Ali Alatabbi, M. Sohel Rahman, and William F. Smyth. Inferring an indeterminate string from a prefix graph. *J. Discrete Algorithms*, 32:6–13, 2015.

[6] Daniel R. Allen, Sharma V. Thankachan, and Bojian Xu. An ultra-fast and parallelizable algorithm for finding k-mismatch shortest unique substrings. *IEEE ACM Trans. Comput. Biol. Bioinform.*, 18(1):138–148, 2021.

[7] Pavlos Antoniou, Maxime Crochemore, Costas S. Iliopoulos, Inuka Jayasekera, and Gad M. Landau. Conservative string covering of indeterminate strings. In *Proc. PSC*, pages 108–115, 2008.

[8] Hideo Bannai, Travis Gagie, Gary Hoppenworth, Simon J. Puglisi, and Luís M. S. Russo. More time-space tradeoffs for finding a shortest unique substring. *Algorithms*, 13(9):234, 2020.

[9] Md. Faizul Bari, Mohammad Sohel Rahman, and Rifat Shahriyar. Finding all covers of an indeterminate string in $o(n)$ time on average. In *Proc. PSC*, pages 263–271, 2009.

[10] Francine Blanchet-Sadri, Michelle Bodnar, and Benjamin De Winkle. New bounds and extended relations between prefix arrays, border arrays, undirected graphs, and indeterminate strings. *Theory Comput. Syst.*, 60(3):473–497, 2017.

[11] Supaporn Chairungsee. A new approach for phylogenetic tree construction based on minimal absent words. In *Proc. DEXA*, pages 15–19, 2014.

[12] Manolis Christodoulakis, Patrick J. Ryan, William F. Smyth, and Shu Wang. Indeterminate strings, prefix arrays & undirected graphs. *Theor. Comput. Sci.*, 600:34–48, 2015.

[13] Jacqueline W. Daykin and Bruce W. Watson. Indeterminate string factorizations and degenerate text transformations. *Math. Comput. Sci.*, 11(2):209–218, 2017.

[14] Arnab Ganguly, Wing-Kai Hon, Rahul Shah, and Sharma V. Thankachan. Space-time trade-offs for finding shortest unique substrings and maximal unique matches. *Theor. Comput. Sci.*, 700:75–88, 2017.

[15] Bernhard Haubold, Nora Pierstorff, Friedrich Möller, and Thomas Wiehe. Genome comparison without alignment using shortest unique substrings. *BMC Bioinform.*, 6:123, 2005.

[16] Joel Helling, Patrick J. Ryan, W. F. Smyth, and Michael Soltys. Constructing an indeterminate string from its associated graph. *Theor. Comput. Sci.*, 710:88–96, 2018.

[17] Jan Holub, William F. Smyth, and Shu Wang. Fast pattern-matching on indeterminate strings. *J. Discrete Algorithms*, 6(1):37–50, 2008.

[18] Wing-Kai Hon, Sharma V. Thankachan, and Bojian Xu. In-place algorithms for exact and approximate shortest unique substring problems. *Theor. Comput. Sci.*, 690:12–25, 2017.

[19] Xiaocheng Hu, Jian Pei, and Yufei Tao. Shortest unique queries on strings. In *Proc. SPIRE*, volume 8799 of *LNCS*, pages 161–172, 2014.

[20] Atalay Mert Ileri, M. Oguzhan Külekci, and Bojian Xu. A simple yet time-optimal and linear-space algorithm for shortest unique substring queries. *Theor. Comput. Sci.*, 562:621–633, 2015.

[21] IUPAC-IUB Commission on Biochemical Nomenclature (CBN). Abbreviations and symbols for nucleic acids, polynucleotides and their constituents. *Journal of Molecular Biology*, 55(3):299–310, 1971.

[22] Felipe A. Louza, Neerja Mhaskar, and W. F. Smyth. A new approach to regular & indeterminate strings. *Theor. Comput. Sci.*, 854:105–115, 2021.

[23] Takuya Mieno, Shunsuke Inenaga, Hideo Bannai, and Masayuki Takeda. Shortest unique substring queries on run-length encoded strings. In *Proc. MFCS*, volume 58 of *LIPIcs*, pages 69:1–69:11, 2016.

[24] Takuya Mieno, Shunsuke Inenaga, Hideo Bannai, and Masayuki Takeda. Tight bounds on the maximum number of shortest unique substrings. In *Proc. CPM*, volume 78 of *LIPIcs*, pages 24:1–24:11, 2017.

[25] Takuya Mieno, Dominik Köppl, Yuto Nakashima, Shunsuke Inenaga, Hideo Bannai, and Masayuki Takeda. Space-efficient algorithms for computing minimal/shortest unique substrings. *Theor. Comput. Sci.*, 845:230–242, 2020.

[26] Takuya Mieno, Yuki Kuhara, Tooru Akagi, Yuta Fujishige, Yuto Nakashima, Shunsuke Inenaga, Hideo Bannai, and Masayuki Takeda. Minimal unique substrings and minimal absent words in a sliding window. In *Proc. SOFSEM*, volume 12011 of *LNCS*, pages 148–160, 2020.

[27] Sumaiya Nazeen, M. Sohel Rahman, and Rezwana Reaz. Indeterminate string inference algorithms. *J. Discrete Algorithms*, 10:23–34, 2012.

[28] Mhaskar Neerja and William F. Smyth. Simple KMP pattern-matching on indeterminate strings. In *Proc. PSC*, pages 125–133, 2020.

[29] Jian Pei, Wush Chi-Hsuan Wu, and Mi-Yen Yeh. On shortest unique substring queries. In *Proc. ICDE*, pages 937–948, 2013.

[30] Daniel W. Schultz and Bojian Xu. Parallel methods for finding k-mismatch shortest unique substrings using GPU. *IEEE ACM Trans. Comput. Biol. Bioinform.*, 18(1):386–395, 2021.

[31] William F. Smyth and Shu Wang. An adaptive hybrid pattern-matching algorithm on indeterminate strings. *Int. J. Found. Comput. Sci.*, 20(6):985–1004, 2009.

[32] Kazuya Tsuruta, Shunsuke Inenaga, Hideo Bannai, and Masayuki Takeda. Shortest unique substrings queries in optimal time. In *Proc. SOFSEM*, volume 8327 of *LNCS*, pages 503–513, 2014.