人工知能学会研究会資料
SIG-FPAI-B501-09

文法圧縮のための逆引き辞書の省スペース化

Space Efficient Reverse Phrase Dictionary for Grammar Compression

高畠嘉将[1*]　坂本比呂志[1]
Yoshimasa Takabatake[1]　Hiroshi Sakamoto[1]

[1] 九州工業大学大学院情報工学府
[1] Graduate School of Computer Sceince and Systems Engineering,
Kyushu Institute of Technology, Japan

Abstract: Highlly repetitive texts have been increasing still now. Examples are individual genomes, version controlled documents and repositries of source codes. Although grammar compression is an efficient for highly repetitive texts, the working space is still large for a data structure called *reverse phrase dictionary*. In this paper, we propose two methods for ruducing the space of reverse phrase dictionary. One is a space efficient dynamic 1-dimentional range search for reverse phrase dictionary, the other is a space reduction method of reverse phrase dictionary for a succinct represetation of grammar compression.

1 はじめに

個人のゲノムやバージョン管理されたテキストやソースコードといった繰り返しの多いテキストが今なお増え続けている．文法圧縮はそういった繰り返しの多いテキストを効率良く圧縮する圧縮手法の一つである．文法圧縮は入力テキストからその入力テキストを一意に導出する文脈自由文法の一種である *straight line program(SLP)* を構築する圧縮手法である．SLP とは生成規則の形が $X_i \to X_jX_k (i > j, k)$ に制限されているものである．通常，文法圧縮は二段階に渡ってテキストを圧縮する．(i) できる限り生成規則の少ない SLP をテキストから構築する．(ii) 得られた SLP を様々なアプリケーションとして利用できるように符号化する．(i) に関しては最小の文法を見つけるのは NP-hard であることが知られており，LCA[9] や REPAIR[4] といった近似アルゴリズムが提案されている．(ii) に関しても q-gram 索引 [3]，圧縮したデータのみで検索可能な自己索引 [1, 2, 5, 10] などが提案されている．. (i) と (ii) を同時に行うオンライン型の文法圧縮法である Fully-online LCA(FOLCA)[6] や Online ESP-index(OESP-index)[11] という方法も提案されており，入力長に依存しない作業領域で構築することが可能である．また挿入や削除を許した入力長に依存しない領域で構築可能な動的な文法圧縮型の自己索引 [8] も提案されている．しかしながら，まだ文法圧縮における作業領域は大きい．一般的に文法圧縮の構築においては辞書と逆引き辞書という 2 つのデータ構造が必要であり，辞書に関しては SLP の情報理論的下限と漸近的に一致するデータ構造が FOLCA で提案されているので，逆引き辞書がその原因となっているといえる．(REPAIR においては逆引き辞書が存在しないが，文字のペアの頻度表のためのハッシュテーブルがこれに相当する．) ここで文法圧縮の辞書とは X_i が与えられて，それに対応する生成規則 $X_i \to X_jX_k$ の右辺 X_jX_k へのアクセスをサポートするデータ構造であり，逆引き辞書は逆に X_jX_k が与えられて，$X_i \to X_jX_k$ という生成規則がすでに現在構築中の SLP の中にあれば X_i を返すデータ構造である．逆引き辞書は一般的にハッシュテーブルで構築されるので，高速性を求める場合は loadfactor を $\alpha = 1$ とすると，FOLCA の場合は辞書のサイズが $2n + n\lg(n) + o(n)$ であるので，2 倍近くの領域が必要となる．そこで本研究では一次元領域探索を用いて省作業領域で構築可能な逆引き辞書を実装する方法と FOLCA で構築される辞書の実装方法である succinct post order SLP(SPOSLP) における逆引き辞書に登録する値の数の削減方法 (RRD) を提案する．

一次元領域探索 $In(Q, x_i, x_j)$ とは $M = \{0, 1, \cdots, m-1\}$，$Q \subseteq M$ と区間 $[x_i, x_j]$ が与えられて，Q 中の要素の中で x_i 以上 x_j 以下の要素を返すものである．領域探索は geometory の検索のために開発されたものであり，そのほかにも自己索引の検索 [1, 8] に応用されてる．領

*連絡先： 九州工業大学大学院情報工学府
福岡県飯塚市川津 680-4
E-mail: takabatake@donald.ai.kyutech.ac.jp

域探索は $In(Q, x_i, x_i)$ とすると連想配列としても用いられるので，ハッシュテーブルの変わりとして用いることが可能である，しかしながら，ここで逆引き辞書の変わりとなる新たな要素の挿入を許した動的一次元領域探索の単純な手法である平衡二分探索木では，生成規則の数を n とすると，$3n\lg(n)$ ビットもの領域がかかってしまう．そこで，本研究では，隣り合う数字が $+1$ もしくは -1 しか違わない超過配列のための動的一次元領域最大最小木 [7] を用いた領域探索法を応用して，SLP の生成規則の左辺 $X_i(1 \leq i \leq n)$ の列を右辺の X_jX_k の辞書式順序（$X_{j_1}X_{k_1} > X_{j_2}X_{k_2}$ とは $X_{j_1} > X_{j_2}$ であるか，$X_{j_1} == X_{j_2}$ のときに $X_{j_1} > X_{j_2}$ であること）で昇順に並び替えた列に対して，動的一次元領域最大木 (DRMT) を構築することにより新しい逆引き辞書 (DRMTRD) を提案する．FOLCA との比較を表 1 に示す．

SPOSLP は SLP を構文木表現したときに各ノードを後置順でラベリングし，一度出現したノードの子孫を枝刈りしたものを動的一次元領域最大最小木 [7] で符号化されたビット列と枝刈りされた構文木の葉ノードのラベル列によって表現された SLP である．この SPOSLP は各ノードから親への移動および子への移動がサポートされているので，親へ移動し，隣りの子へ移動すれば兄弟に移動可能である．後置順でみた i 番目の内部ノードは SLP の生成規則 X_i と対応しているので，逆引き辞書の機能を使うときに与えられる X_jX_k に対して，X_j と X_k に対応する内部ノードにアクセスし，その兄弟を調べれば内部ノードに関する生成規則を逆引き辞書に登録することは不要となる．残った逆引き辞書に登録する生成規則数が高々$\frac{n}{2}$ であることを証明し，n から高々$\frac{n}{2}$ に逆引き辞書に登録する領域を減らすことに成功した．この手法を用いた場合の逆引き辞書のサイズを表 1 に示す．

2 章でこの論文で使う基本概念について説明し，3 章で動的一次元領域探索を用いた逆引き辞書 DRMTRD について説明し，4 章で SPOSLP を用いた逆引き辞書に登録する生成規則の数の削減方法の詳細とその数の証明を行い，5 章で，まとめと今後の課題について述べる．

2 準備

2.1 基本概念

集合 C の要素数は $|C|$ とする．Σ を有限アルファベットの集合とし，$\sigma = |\Sigma|$ とする．文字列 R の長さを $|R|$ と表記する．入力文字列は S とし，$N = |S|$ とする．変数の帰納的可算集合 $\mathcal{X}$ は $\Sigma \cap \mathcal{X} = \emptyset$ とする．$\lg = \log_2$ である．計算機モデルは word RAM モデルを用いて，$\theta(\lg N)$ の演算は $O(1)$ で実行可能であることを仮定する．

2.2 文法圧縮

文法圧縮とは入力テキストを一意に導出する文脈自由文法の一つ *straight line program(SLP)* $\mathrm{G} = \{V, \Sigma, P, X_S\}$ を構築する圧縮手法である．ここで，$V \subseteq \mathcal{X}$，$P \subseteq (V \times (\Sigma \cup V)^2)$，$X_S \in V$ は SLP の開始記号であり，SLP の生成規則の形は $X_i \to X_jX_k (X_i \in V, i < j, k)$ に制限される．各 X_i は非終端記号と呼ぶ．$n = |P|$ とする．

2.3 辞書と逆引き辞書

文法圧縮を行うためには一般的に辞書 D と逆引き辞書 D^{-1} の二つのデータ構造が必要であるので，これらのデータ構造について説明する．文法圧縮の P を表現するデータ構造は辞書 D と呼ばれ，関数 $D(X_i)$ は $X_i \to X_jX_k \in P$ のとき, X_jX_k を返し, それ以外の時は，$X_\infty X_\infty (X_\infty \notin V)$ を返す．一般的に D は $2n\lg(n)$ ビットの二重配列で表現され，関数 $D(X_i)$ を $O(1)$ で返す．新しいデータの追加の行われない静的な D に関しては *Improved ESP-index(ESP-index-I)*[10] において，領域が $2n + n\lg n + o(n\lg n)$ ビットという SLP の情報理論的下限である $2n + \lg n! + o(n)$ 近いサイズで関数 $D(X_i)$ を $O(\lg\lg n)$ で返す方法が提案されている．新しいデータの追加が行われる場合に関しても *fully online LCA(FOLCA)*[6] では D を $2n + n\lg(n+\sigma) + o(n)$ ビットという SLP の情報理論的下限に漸近的に一致するサイズで $O(\frac{\lg(n)}{\lg(\lg(n))})$ 時間で末尾追加とアクセスする方法が提案されている．FOLCA の辞書の詳細については 2.5 節で説明する．

逆引き辞書 D^{-1} は X_jX_k が与えられて，$X_i \to X_jX_k \in P$ のとき，X_i を返すデータ構造である．この逆引き辞書の関数は $D^{-1}(X_jX_k) = X_i$ として表現する．逆引き辞書の一般的なデータ構造はチェイン法によるハッシュテーブルであり，loadfactor を $\alpha \in (0, 1]$ とすると，$n(1+\alpha)\lg(n+\sigma)$ ビットで表現でき，$D^{-1}(X_j, X_k)$ を $O(1/\alpha)$ 期待時間で返す．静的な場合には ESP-index-I では辞書 D と同じデータ構造を用いて，$O(\lg\lg n)$ 時間で返す方法が提案されている．FOLCA ではハッシュテーブルをリストに登録される値の単調増加性を用いて圧縮しており，その領域は $\alpha n\lg(n+\sigma) + n(1+\lg(\alpha n))$ であり，$D^{-1}(X_j, X_k)$ を $O(\frac{1}{\alpha})$ 期待時間で返す．

表 1: Comparison with FOLCA. RD is the size of reverse phrase dictionary, WS is the size of working space for constrution, and CT is the construction time, N is the length of text, α is a loadfactor of hash table, n is the number of variables in a grammar, σ is alphabet size, lg^* is the iteration of lg and lg stands for $\log_2$.

	RD(bits)	WS(bits)	CT
FOLCA[6]	$\alpha n \lg(n+\sigma) + n(1+\lg(\alpha n))$	$(1+\alpha)n\lg(n) + n(3+\lg(\alpha n))$	$\frac{N\lg(n)\lg^*(N)}{\lg(\lg(n))}$
FOLCA+DRMTRD	$n + n\lg(n) - o(n)$	$3n + 2n\lg(n) - o(n)$	$\frac{N(\lg(n))^2\lg^*(N)}{\lg(\lg(n))}$
FOLCA+RRD	$\frac{\alpha n}{2}\lg(n+\sigma) + \frac{n}{2}(1+\lg(\alpha n))$	$(\frac{1}{2}+\alpha)n\lg(n) + \frac{n}{2}(5+\lg(\alpha n))$	$\frac{N\lg(n)\lg^*(N)}{\lg(\lg(n))}$
FOLCA+DRMTRD+RRD	$\frac{n}{2} + \frac{n}{2}\lg(n) - o(n)$	$\frac{5}{2}n + \frac{3}{2}n\lg(n) + o(n)$	$\frac{N(\lg(n))^2\lg^*(N)}{\lg(\lg(n))}$

2.4 一次元領域探索

本論文では3章で逆引き辞書を動的一次元領域探索で表現するので，一次元領域探索について説明する．一次元領域探索 $Is(Q, x_i, x_j)$ と集合 $M = \{0, 1, \cdots, m-1\}$, $Q \subseteq M$ と区間 $[x_i, x_j](1 < x_i < x_j < m)$ が与えられて，区間 $[x_i, x_j]$ にある Q の集合に属する要素を返す．動的一次元領域探索とは $Is(Q, x_i, x_j)$ を行えるようにしながらかつ Q への新たな正の整数の削除および追加が可能である．

新たなデータの追加や削除といった更新が行われない静的な場合の一般的な手法だと Q の要素をソートして，二分探索で区間 $[x_i, x_j]$ の要素を探すと，$|Q|\lg(m)$ ビットの領域で $O(\lg(|Q|) + k)$ 時間 (k は解の個数) で $Is(Q, x_i, x_j)$ を行うことが可能である．構築時間は $O(|Q|\lg(|Q|))$ 時間である．

動的な場合には一般的な手法だと平衡二分探索木を用いて，$|Q|\lg(m)+2|Q|\lg(|Q|)$ ビットの領域と $O(\lg(|Q|)+k)$ 時間の領域で構築することが可能である．これの挿入 (insert)・削除 (delete) は最悪 $O(\lg(|Q|))$ 時間で，構築時間も $O(|Q|\lg(|Q|))$ 時間である．

したがって，辞書 D がある時に生成規則 $X_i \to X_jX_k$ を X_jX_k の辞書順 ($X_{j_1}X_{k_1} > X_{j_2}X_{k_2}$ とは $X_{j_1} > X_{j_2}$ であるか， $X_{j_1} == X_{j_2}$ のときに $X_{j_1} > X_{j_2}$ であること) で一次元領域探索を行えるようにしておけば，$D^{-1}(X_jX_k) = Is(D, X_jX_k, X_jX_k)$ とすることにより逆引き辞書を実装可能であるが，平衡二分探索木のような一般的な手法で行うと既存のハッシュテーブルを用いた手法よりも領域が多くかかってしまう．そこで提案手法では領域をハッシュテーブルより小さな領域で実装しつつ，高速に $D^{-1}(X_jX_k) = Is(D, X_jX_k, X_jX_k)$ を計算する方法を提案する．

2.5 Fully-online LCA(FOLCA)

4章で fully-online LCA(FOLCA)[6] の辞書のデータ構造を用いて，逆引き辞書の領域を削減する方法を提案するので，FOLCA について説明する．FOLCA ではオンラインで SLP の情報理論的下限と漸近的に一致するサイズの辞書 D を構築する．その辞書は succinct post order SLP(SPOSLP) と呼ばれる．post order SLP(POSLP) とは SLP を開始記号を根とする構文木と見たとき，各非終端記号を後置順にラベリングし，一度出現した内部ノードはその子孫を枝刈りした SLP 表現である．SPOSLP は POSLP を後置順に辿り，葉ノードなら 0 内部ノードなら 1 を並べたビット列 B と POSLP の葉ノードのラベルを後置順に並べた配列 L を用意する．B は動的区間最小最大木 [7] により符号化されている．符号化された B と L を用いると様々操作が可能となる [6, 11] が，本論文では $D(X_i)$ と $ParentInternal(i)$ の 2 つの操作について説明する．$D(X_i)$ は $O(\frac{\lg(n)}{\lg(\lg(n))})$ 時間で実行される．$ParentInternal(i)$ は後置順で見たときの非終端記号 X_i に対応する i 番目の内部ノードの親を返す関数であり，$O(\frac{\lg(n)}{\lg(\lg(n))})$ 時間で実行できる．FOLCA ではほぼ辞書とハッシュテーブルの領域のみで構築可能であり，以下の定理が成り立つ．

定理 1. (FOLCA[6]) 長さ N の入力テキストから $O(\frac{N\lg(n)\lg^*(N)}{\alpha\lg(\lg(n))})$ で POSLP の辞書と逆引き辞書を $(1+\alpha)n\lg(n+\sigma) + n(3+\lg(\alpha n))$ の作業領域で構築できる．

3 提案手法その1：逆引き辞書のための省領域な一次元領域探索 (DRMTRD)

提案する文法圧縮のための逆引き辞書は超過配列（ビット列）のための動的区間最小最大木 (DRmMT)[7] のデータ構造とほぼ同じデータ構造の動的区間最大木 (DRMT) により構築される一次元領域探索法により実現する．DRmMT では扱う列がビット列であったが，提案手法では生成規則を辞書式順序で昇順に並び替えた列であることが一番の違いである．まずデータ構造について述べ，それに対して $D^{-1}(X_jX_k)$ と $Insert(X_i)$ を実現する方法を説明する．

辞書 D が与えられていると仮定する．各生成規則の左辺の $X_i(i \in [1, n])$ の列 $X = X_1, X_2, \cdots, X_n$ を生成規則 $(X_i \to X_jX_k)$ の右側 X_jX_k が辞書順で昇順と

なるように並びかえた列 X_{sort} が与えられているとする．X_{sort} を長さ $3\lg(n) \le L < 6\lg(n)$ の k 個のブロック列 $Div = Div_1, Div_2, \cdots Div_k$ に分割する．各 Div_i を葉ノードとする平衡二分木を構築する．構築は隣りあうブロックの Div_{2i-1}, Div_{2i} を子とする内部ノード In_i からなる列 $In = In_1, In_2, \cdots, In_{k/2}$ を構築する．この操作で出来た内部ノードの列に対しても繰り返しノードの構築を行い，出来るノードの数が 1 になるまで行うことで Div を葉とする平衡二分木が構築される．各内部ノードは子孫の持つ生成規則の右辺 X_jX_k が辞書順で最大の生成規則の左辺 X_{max} と左の子および右の子へのポインタ (left,right) を格納しておく．葉のブロックには保持する列の長さ $|Div_i|(1 < i < k)$ を保持しておく．

定理 2. DRMTRD のサイズは $n + n\lg n - o(n)$ ビットである．

証明. 葉と内部ノードそれぞれが持っているデータ構造のサイズを確認し，全体のサイズを求める．葉ノードには生成規則の左辺 $X_i(i \in [1,n])$ からなる列とその列の長さ $|Div_i|(1 < i < k)$ を保持している．生成規則の左辺は一個あたり $\lg(n)$ ビットであり，生成規則は n 個なので，$n\lg(n)$ ビットである．各葉ノードの $|Div_i|$ は一つあたり $\lg(6\lg(n)) = \lg(\lg(n)) + \lg(6)$ ビットであり，葉ノードの数は $\frac{n}{3\lg(n)}$ なので，$|Div_i|$ のすべての合計サイズは $(\frac{n}{3\lg(n)}) * (\lg(\lg(n)) + \lg 6)) = \frac{n(\lg(\lg(n)) + \lg 6)}{3\lg(n)}$ ビットである．したがって葉ノードの合計サイズは $n\lg(n) + \frac{n(\lg(\lg(n)) + \lg 6)}{3\lg n}$ ビットである．内部ノードはサイズ $\lg(\frac{2n}{3\lg(n)}) = \lg(n) - \lg(\lg(n)) + 1 - \lg(3)$ ビットのポインタが 2 つあり，$\lg(n)$ ビットの X_{max} を持っているので、内部ノード一つあたり，$3\lg(n) - 2\lg(\lg(n)) + 2 - 2\lg(3)$ ビット必要である．内部ノードの数は $\frac{n}{3\lg(n)}$ なので，内部ノードの全てのサイズは $n + \frac{n(2 - 2\lg(\lg(n)) - 2\lg(3))}{3\lg(n)}$ である．したがって，このデータ構造の合計サイズは $n\lg(n) + \frac{n(\lg(\lg(n)) + \lg(6))}{3\lg(n)} + n + \frac{n(2 - 2\lg(\lg(n)) - 2\lg(3))}{3\lg(n)} = n + n\lg(n) + \frac{n(2 + \lg(6) - \lg(\lg(n)) - 2\lg(3))}{3\lg(n)}$ $= n + n\lg n - o(n)$ ビットである． □

定理 3. $D(X_i)$ の計算時間を $O(d)$ とすると，DRMTRD における $D^{-1}(X_jX_k)$ を $O(d\lg(n))$ 時間，新しい生成規則の登録 $insert(X_i)$ を $O(d\lg(n))$ 時間で実行可能である．

証明. $D^{-1}(X_jX_k)$ は根から葉へと辿り，X_jX_k 存在する可能性のある葉ノードをみつけ，その中で二分探索を行うことで実行する．各ノードで左の子の $D(X_{max}) = X_lX_r$ が辞書順で X_jX_k より大きい場合は左の子へ移動する．そうでなければ，右の子へ移動する．この操作を繰り返し，葉ノードにたどり着いたら，葉ノードの中で二分探索することで見つけることができる．$D^{-1}(X_jX_k)$ の時間計算量は木の根から葉への探索と葉ノードに格納されている非終端記号列の二分探索の合計である．木での探索は一回の $D(X_i)$ の時間は $O(d)$ 時間であり，木の高さ $\lg(n)$ であるので，$O(d\lg n)$ 時間である．葉ノードの二分探索は長さが $O(\lg(n))$ であるので，$O(\lg(\lg(n)))$ 時間である．したがって，$D^{-1}(X_jX_k)$ の時間計算量は $O(d\lg n) + O(\lg(\lg(n))) = O(d\lg(n)))$ 時間である．

$Insert(X_i)$ も葉まで $D^{-1}(X_jX_k)$ と同様にして辿り，葉ノードの挿入位置が見つかったら，葉ノードの非終端記号列の長さを一つ増やして，挿入位置以降の非終端記号列の要素をずらして X_i を挿入する．このとき非終端記号列の末尾に追加した場合はそこから根までのパスが右である限り X_{max} を X_i に更新する．この $insert(X_i)$ により葉ノードに格納されている非終端記号列の長さが $6\lg(n)$ となってしまった場合は葉ノードを $3\lg(n)$ ずつに分割して，新しい内部ノードを作り分割した葉ノードの親とする．このとき左右の木の高さの差が 2 以上になってしまった場合は回転操作を行い，木をバランスさせる．木の高さは $\lg(n)$ であり，葉の非終端記号列の長さも $O(\lg(n))$ であり，回転操作も $O(\lg(n))$ で実行できるので，$insert(X_i)$ も $O(d\lg(n))$ 時間で実行可能である． □

4 提案手法その２：SPOSLP の逆引き辞書の削減方法 (RRD)

FOLCA[6] のようにオンラインで文法圧縮を行い，SPOSLP で辞書の符号化を行っている場合の逆引き辞書の削減方法を提案する．SPOSLP は任意の非終端記号 X_j に対応する内部ノードからその親 $X_{Parent(X_j)}$ と兄弟 $X_{Sibling(X_j)}$ を $O(\frac{\lg(n)}{\lg(\lg(n))})$ 時間で発見できるという性質（詳細は以下）を利用して，逆引き辞書へ登録する非終端記号を SPOSLP と対応する内部ノードの両方の子どもが葉なっているもののみにする．そのような両方の子どもが葉となる内部ノードの数が高々 $\frac{n}{2}$ 個であることを補題 4 で示す．

まず任意の非終端記号 X_j に対応する内部ノードからその親 $X_{Parent(X_j)}$ と兄弟 $X_{Sibling(X_j)}$ を $O(\lg(n)/\lg(\lg(n)))$ 時間で発見する方法について説明する．逆引き辞書のように X_jX_k が与えられたとき，SPOSLP は $ParentInternal(j)$ により，任意の非終端記号 X_j と対応する内部ノードからその親 $X_{Parent(X_j)}$ にたどり，$D(X_{Parent(X_j)})$ とすると，X_j の兄弟 $X_{Sibling(X_j)}$ が分かる．$X_{Sibling(X_j)}$ が $D(X_{Parent(X_j)})$ の右側でかつ $X_k == X_{Sibling(X_j)}$ ならば，$D^{-1}(X_jX_k) = X_{Parent(X_j)}$ である．そうでなければ，X_k に対応する内部ノードに対しても同様に X_j が兄弟か調べる．$ParentInternal(i)$ および $D(X_i)$ の回数は定数回なので，$O(\frac{\lg(n)}{\lg(\lg(n))})$ 時間である．した

がって，内部ノードの X_j および X_k は調べることができるので，逆引き辞書に登録する非終端記号は対応する内部ノードの子どもが両方葉になっているものである．以下の補題によって，POSLP の内部ノードの子どもが両方葉である数が最大で何個になるかを証明する．

補題 4. 文法圧縮における SLP を SPOSLP で表現した場合，逆引き辞書に登録する非終端記号の数は高々 $\frac{n}{2}$ 個である．

証明. POSLP の内部ノードの数は $n-1$ 個であり，すべての内部ノードの兄弟が内部ノードである場合に逆引き辞書に登録する非終端記号の数が最大になる．なぜなら，一つでも内部ノードと葉ノードが兄弟になってしまえば，葉ノードが減り，両方の子とも葉である内部ノードが少なくなるからである．葉ノードの数は n 個なので，逆引き辞書に登録する葉ノードの数は高々 $\frac{n}{2}$ 個である． □

辞書を SPOSLP で表現する FOLCA とこの逆引き辞書の削減方法を用いると，以下の定理が成り立つ．

定理 5. 長さ N の入力テキストから POSLP の辞書と逆引き辞書を $O(\frac{N\lg(n)\lg^*(N)}{\alpha\lg(\lg(n))})$ 時間で $(1+\frac{\alpha}{2})n\lg(n+\sigma)+\frac{n}{2}(5+\lg(\alpha n)$ ビットの領域で構築できる．

証明. 定理. 1 の FOLCA の構築時間の $O(\frac{\lg(n)}{\alpha\lg(\lg(n))})$ は $D^{-1}(X_jX_k)$ の時間であり，それ以外の計算は FOLCA と同様に行われる．したがってこの手法を用いた場合の構築時間は $O(\frac{N\lg(n)\lg^*(N)}{\alpha\lg(\lg(n))})$ 時間である．ハッシュテーブルに登録される値の数は $\frac{n}{2}$ になるので，ハッシュテーブルのサイズは $\frac{\alpha n}{2}\lg(n+\sigma)+\frac{n}{2}(1+\lg(\alpha n))$ ビットの領域で構築できる．したがって，合計サイズは $2n+n\lg(n+\sigma)+o(n)+\frac{\alpha n}{2}\lg(n+\sigma)+\frac{n}{2}(1+\lg(\alpha n))=(1+\frac{\alpha}{2})n\lg(n+\sigma)+\frac{n}{2}(5+\lg(\alpha n))+o(n)$ ビットの領域で構築できる． □

また逆引き辞書を DRMTRD に替えると，以下の定理が成り立つ．

定理 6. 長さ N の入力テキストから POSLP の辞書と逆引き辞書を $O(\frac{N(\lg(n))^2\lg^*(N)}{\lg(\lg(n))})$ 時間で $\frac{5}{2}n+\frac{3}{2}n\lg(n)+o(n)$ ビットの領域で構築できる．

証明. $D^{-1}(X_jX_k)$ の時間は $O(d\lg(n))=O(\frac{(\lg(n))^2}{\lg(\lg(n))})$ 時間である．したがって構築時間は，$O(\frac{N(\lg(n))^2(\lg^*(N))}{\lg(\lg(n))})$ 時間である．作業領域は，$2n+n\lg(n)+o(n)+\frac{n}{2}+\frac{n}{2}\lg(n)-o(n)=\frac{5}{2}n+\frac{3}{2}n\lg(n)+o(n)$ ビットである． □

5 まとめと今後の課題

動的一次元領域探索を用いて，$O(d\lg n)$ 時間で挿入とアクセス可能な $n+n\lg n-o(n)$ ビットの文法圧縮のための動的な逆引き辞書を構築する方法を考案した．提案手法では超過配列のための動的区間最大最小木 [7] のアイデアを用いて，それとほぼ同じ動的区間最大木を生成規則の単調増加列に対して構築することにより実現した．また辞書が SPOSLP で表現される時の特殊な場合において，逆引き辞書を減らす方法を提案した．これらを用いて，オンライン文法圧縮の FOLCA を行うと，$O(\frac{N(\lg(n))^2\lg^*(N)}{\lg(\lg(n))})$ 時間で $\frac{5}{2}n+\frac{3}{2}n\lg(n)+o(n)$ ビットの領域という今までの文法圧縮の中で最も省メモリに圧縮を行うことが可能である．今後の課題としては実際に実装し，実行速度やメモリ使用量を測る必要がある．また静的な場合には SLP の情報理論的下限 $2n+\lg(n!)+o(n)$ に近いサイズの $2n+n\lg(n)+o(n\lg(n))$ で辞書と逆引き辞書の両方の機能有したデータ構造が提案 [10] されており，動的な場合に関してもこのサイズに近づける必要がある．さらに今回の提案した逆引き辞書はアクセス時間が低下してしまったので，これの高速化も今後課題である．元々の動的区間探索木は長さ $\frac{(\lg(n))^2}{\lg(\lg(n))}$ のブロックに分割して行い，その後構築される平衡木の分岐数も $\sqrt{\lg(n)}$ にすることで超過配列上で $O(\frac{\lg(n)}{\lg(\lg(n))})$ 時間を達成している．これに近づけることが目標である．

参考文献

[1] F. Claude and G. Navarro. Self-indexed grammar-based compression. *Fundam. Inform.*, 111(3):313–337, 2011.

[2] T. Gagie, P. Gawrychowski, J. Kärkkäinen, Y. Nekrich, and S.J. Puglisi. A faster grammar-based self-index. In *LATA*, pages 240–251, 2012.

[3] K. Goto, H. Bannai, S. Inenaga, and M. Takeda. Fast q-gram mining on SLP compressed strings. *JDA*, 18:89–99, 2013.

[4] N.J. Larsson and A. Moffat. Offline dictionary-based compression. *Proceedings of the IEEE*, 88(11):1722–1732, 2000.

[5] S. Maruyama, M. Nakahara, N. Kishiue, and H. Sakamoto. ESP-Index: A compressed index based on edit-sensitive parsing. *JDA*, 18:100–112, 2013.

[6] S. Maruyama, Y. Tabei, H. Sakamoto, and K. Sadakane. Fully-online grammar compression. In *SPIRE*, pages 218–229, 2013.

[7] G. Navarro and K. Sadakane. Fully-functional static and dynamic succinct trees. *ACM Transactions on Algorithms*, 2012. Accepted. A preliminary version appeared in SODA 2010.

[8] T. Nishimoto, T. I, S. Inenaga, H.Bannai, and M. Takeda. Dynamic index, lz factorization, and lce queries in compressed space. arXiv:1504.06954.

[9] H. Sakamoto, S. Maruyama, T. Kida, and S. Shimozono. A space-saving approximation algorithm for grammar-based compression. *IEICE trans. inf. syst.*, E92-D:158–165, 2009.

[10] Y. Takabatake, Y. Tabei, and H. Sakamoto. Improved ESP-index: a practical self-index for highly repetitive texts. In *SEA*, pages 338–350, 2014.

[11] Y. Takabatake, Y. Tabei, and H. Sakamoto. Online self-indexed grammar compression. In *SPIRE*, 2015.