問い合わせ処理に適したXMLデータ圧縮形式に関する研究

東原　正智†　　田島　敬史‡

北陸先端科学技術大学院大学情報科学研究科　〒923-1292 石川県能美郡辰口町旭台１－１

E-mail:　†m-higa@jaist.ac.jp　‡tajima@jaist.ac.jp

あらまし　本研究では，効率の良い問い合わせ処理が可能なXMLデータの圧縮手法を提案する．我々の手法の特徴は，タグの木構造の情報とテキストデータの部分を分離して圧縮する非同型圧縮を用い，タグの木構造の情報の圧縮に関しては文法に基づく圧縮技術を適用する点と，圧縮データ中に，部分木のパーズをスキップするための索引を埋め込む点である．後者については，圧縮比を多少犠牲にすることになるが，この二つの手法により，圧縮された巨大なデータに対して問い合わせ処理を行って小さな部分のみを取り出す場合に，データ全体を解凍せず，必要な部分だけを解凍パーズして問い合わせの解を得ることができ，問い合わせ処理の効率が大きく向上する．

キーワード　XML，文法に基づくデータ圧縮，問い合わせ可能データ圧縮，問い合わせ処理，indexing

An XML Data Compression Method for Efficient Query Processing

Masanori HIGASHIHARA†　　Keishi TAJIMA‡

School of Information Science

Japan Advanced Institute of Science and Technology

1-1 Asahidai，Tatsunokuchi，Ishikawa，923-1292，JAPAN

E-mail:　†m-higa@jaist.ac.jp　‡tajima@jaist.ac.jp

Abstract　In this paper，we propose a new method of compressing XML data，which produces a query-friendly compressed data．Our method separately compress the information on tree structure of tags and text data in XML data，and applies grammar-based compression method to the information on tree structure．It also embeds indexes in the data so that we can skip the parsing of subtrees which are unrelated to the current query．Although the latter slightly reduces the compression ratio，these two techniques allow us to process queries on compressed data with decompressing/parsing not the whole data but only necessary portions of the data．This improves the efficiency of the processing of queries that extract only small portions from a huge compressed data.

Keyword　XML，data compression，grammar-based compression，query processing，indexing

1. はじめに

近年, XML がデータフォーマットとして産業界等で利用されている. それに伴い, XML に関する様々な技術が研究・開発されている. XML データの圧縮技術もそのうちの一つである. XML データの圧縮技術については, XML の構造を利用した圧縮手法を用いることで, 通常のテキスト圧縮のツールを用いるよりも圧縮率を改善することが可能となることを示した研究[1]が端緒となり, それ以降, 様々な試みがなされている. さらに, 最近では圧縮した XML データに対して直接問い合わせが実行可能であるような圧縮形式の研究が行われている. これは, 巨大な XML データに対して検索を行って, 小さなサイズのデータを抜き出したいような場合, データ全体を解凍して問い合わせを行うよりも, 圧縮した XML データに対して直接問い合わせを処理して, 解となる部分のみを解凍して返すことができれば, より効率がよいと考えられるからである.

そこで, 本研究では, 圧縮した状態の XML データに対して直接問い合わせを実行して, 必要な部分のみを解凍しながら問い合わせの解を抜き出せるような, XML データ用の圧縮形式を開発する.

2. 関連研究

まず，本節で，主な関連研究について簡単に解説する．最初に，これまでに提案されている XML データ用の圧縮形式について概観し，次に，本研究で我々が圧縮データ中に埋め込む索引に用いる方式である stream index について説明する．三番目に，本研究で提案する圧縮形式で，木構造部分の情報の圧縮に適用する文法に基づく圧縮技術について説明する．

2.1. XML compressor

XM のデータ構造を考慮した XML データ専用の圧縮形式は, 大きく分けて, 圧縮後のデータ全体の構造が, オリジナルの XML の木構造に直接対応するような形になっているものと, そうでないものに分類できる. 前者を同型圧縮, 後者を非同型圧縮と呼ぶ場合もある. また, 別の分類軸として, 圧縮されたデータに対して直接問い合わせ可能なものと, そうでないものに分類することもできる. これらの 4 通りの組合わせ毎に, これまでに提案されている圧縮形式についてまとめると以下のようになる.

I. 同型圧縮
  I-1 問い合わせ不可能
  ・xmlppm [4]
  ・xpst [8]
  I-2 問い合わせ可能
  ・XGrind [2]
  ・XPress [3]
II. 非同型圧縮
  II-1 問い合わせ不可能
  ・XMill [1]
  II-2 問い合わせ可能
  ・XCQ [5][6]
  ・XQueC [25][26]

xmlppm は, ppm 符号と MHM モデルを採用した圧縮手法だが, 圧縮率の向上を目的としており, 圧縮データに対して直接問い合わせを行うことは目的としていない.

xpst はテキスト予測モデル pst を DOM 木に拡張し, ppm 符号を採用した圧縮形式だが, これも同様に, 圧縮データに対して直接問い合わせを行うことは目的としていない.

XGrind は, 圧縮データに対して直接問い合わせが可能な圧縮形式としては最初に提案された物で, 基本的に, もとの XML データの木構造を残したまま, 各要素中のテキストデータの部分を個別に圧縮する. データの木構造が圧縮形式中にほぼそのまま残っているため, 圧縮したデータを先頭から解凍しながら問い合わせを実行していく処理が比較的簡単に実現できるが, 問い合わせを実行するためには, データ全体を(必ずしも解凍はしなくても)ディスクから一度読み出すことになるため, 圧縮データが非常に大きい場合は, I/O コストが大きくなる.

XPress は, やはり同型圧縮を用いる, 直接問い合わせが可能な圧縮形式で, 根から各要素へのパスの情報を算術符号を用いて圧縮する. 算術符号では, ある文字列 S_1 がある文字列 S_2 の接頭辞になっている場合, そのことが, S_1 と S_2 を圧縮した物のみを見て判定できるという特徴があり, これを利用して, 効率の良い問い合わせ処理を実現しているが, やはり, 問い合わせを実行するには, データ全体を一度ディスクから読み出す必要がある.

XMill は, XML データ専用の圧縮方式の研究の端緒となった研究で, XML データを, その木構造に関する情報と, テキストデータの部分とに分けて圧縮する. その際, テキストデータを, そのテキストデータが現れる要素へのパスに応じて別々の「コンテナ」に分類し, 各コンテナ毎に一般のテキスト圧縮ツール等を用いて圧縮する. 各コンテナに対してどんな圧縮ツールを用いるのが最適かはユーザが指定することができ, また, 指定されない場合は, ある程度の自動判定も行う. 似たようなパスの先には似たようなテキストデータが現れる可能性が高いため, パスに応じてテキストデータを分類して圧縮することによって, 高い圧縮比を得ている. XMill のオリジナル版である version 0. 7 は AT&T より配布されていたが, その後, AT&T の手を離れオープンソースとなり, 現在, version 0. 8 が配布されている. 現在の version 0. 8 では, テキストデータコンテナの圧縮に, version 0. 7 でも用いられていた gzip と bzip2 に加えて, ppmdi の圧縮アルゴリズムが用いられている.

XCQ は, DTD を利用して圧縮を行う, 非同型圧縮でかつ直接問い合わせが可能な圧縮形式であるが, DTD を持つ文書に対してのみ適用可能であるという欠点がある.

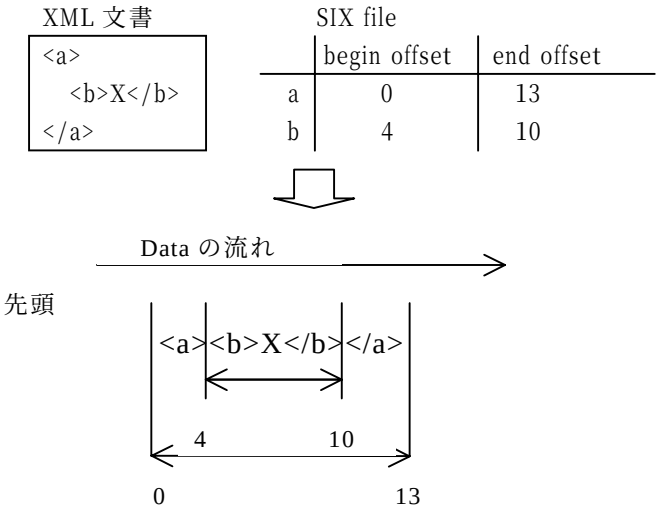
XQueC は, 非同型圧縮で, 圧縮アルゴリズムに ALM ア

ルゴリズムという圧縮後も順序構造を保つようなアルゴリズムを採用している. 木構造の情報の圧縮に, 木構造中に出現するパスの種類の要約にあたる DataGuide [28] を利用する点が特徴であり興味深いが, 文法圧縮を用いる我々のアプローチとは異なる. また, 問い合わせ時にコストベースの最適化を行うなど問い合わせに重点を置いている.

これらの他に, 商用では, XML-press (ICT), XML-CSV (富士通), XMLZIP (XML Solution) などがあるが, 詳しいアルゴリズムは不明である. 以上が現在, われわれが確認している XML のデータ構造を考慮した圧縮形式である.

2. 2 Stream index (SIX)

最近, ネットワークを流れる XML のデータストリームの研究が活発に行われている. Stream index (SIX) [24]は, このような XML データのストリーム処理のために提案されている索引構造の一種で, XML データ中の各ノードの開始タグと終了タグの位置の, データの先頭からのオフセットのペアを順に並べたものである. 例えば, 以下の Fig. 1 に示した XML データに対しての SIX の内容は, 以下の Fig. 1 に示したようなものになる. 図中に示した XML 文書との対応関係がわかりやすいようにタグ名も示したが, 実際の SIX には タグ名の部分は含まれない. 問い合わせ/c/d/e が発行されたときに, XML データのタグの<a>を見て先頭のタグの/c と照合し, 一致しないため, Stream index の offset 分をエレメントごと skip する. この例の場合, 0 から 13 までの subtree をエレメント X ごと skip する.



問い合わせ:/c/d/e の場合, 先頭のタグが違うから見なくてもよい.

Fig. 1 Stream index

論文[24]で提案されている XML ストリームデータ処理システムでは, サーバーから XML データのストリームと SIX データのストリームとがクライアントに送られ, クライアント側では SAX パーサと SIX マネージャーが, それぞれを受信する. クライアントは, SIX データとして送られてくる, 現在見ているノードとその祖先の終了タグ位置をスタックに保存していき, それと同時に, 今実行している問い合わせに関して現在のノード以下の部分木を見る必要があるかどうか判断し, 見る必要がなければ, スタックのトップに保存されているデータを使って, その部分木の終了位置までデータの処理をスキップする. また, 現在見ているノードの n 段上の祖先の終了位置までスキップしてよいことが明らかになった場合は, スタックの n 個下に詰まれているデータを使って, スキップを行う. これによって XML のストリームデータに対する問い合わせ処理の効率化を図っている.

2. 3 文法による圧縮アルゴリズム

文法に基づくデータ圧縮は, 圧縮したいデータ系列が与えられたとき, そのデータ系列のみを生成するような決定性のある, できるだけ簡単な文法規則を生成し, この文法規則を何らかの無ひずみ圧縮アルゴリズムで圧縮することによりデータを圧縮するという方法である. 文法に基づくデータ圧縮の提案は 1970 年代に遡るが, 最近になって2つのグループにより性能の良いアルゴリズムが発表され活発に研究されている[9] [23]. 1つ目のグループである Nevill-Manning と Witten は, SEQUITUR というアルゴリズムを提案している[12][13]. 彼らの目的は, DNA 配列, 楽譜, 生物などからパターンを抽出することであった. もう一方のグループは, Kieffer, Yang, Nelson, Cosman のグループである[14][15]. 彼らは, BISECTION, MPM, LONGEST MATCH などのアルゴリズムと SEQUITUR を改良した SEQUENTIAL というアルゴリズムを提案している. これらの研究が端緒となって, 様々なアルゴリズムの改良が研究されている. 例えば, SEQUITUR は, 文法を生成する場合に前から digram を見ていくが, これに対し, 1 パス目で頻出する digram を発見し, その digram を基準にして文法を生成する最頻 digram 法[21][22]や, 2 パスのアルゴリズムである MPM 符号化を改良して, 1 パスで逐次的に文法を生成するようにしたもの[20]などが提案されている. また, AVL 木を利用した圧縮法なども提案されている[18]. また, これらの文法を用いた圧縮法を最小文法問題の近似アルゴリズムという観点から捉え, アルゴリズムの性能評価と改良を提案している研究等もある[16][17][19].

ここで上に挙げたような, 既存の, 文法に基づくデータ圧縮方法は, いずれも

(a) データ系列から, そのデータ系列自身のみを生成するような「許容可能な文脈自由文法」を生成する.
(b) 生成された文脈自由文法は, 適応型算術符号によって圧縮する.

という方式を取っている. ここで, 許容可能な文法とは, 以下のように定義される.

[定義]許容可能な文法とは, 以下の条件を満たす文法 G

のことである.

(1) Gは, 決定性文法である.
(2) Gの任意の生成規則u→wにおいて, wは空列をもたない.
(3) L(G) ≠φ
(4) Gは, 生成規則の中に一度も現れない記号を持たない. つまり, 生成規則に元の文字列とその部分文字列を置換した記号以外の無関係な記号は存在しない.

与えられたデータに対して, 許容可能な文脈自由文法を得るには色々な方法が考えられるが, 例えば, Yang と Kieffer の Greedy Sequential Grammar Transformation では, データを先頭から一文字ずつ読みながら文法規則を生成していき, その過程で三つの縮約規則を使って文法規則を簡単化していくことで, 許容可能な文脈自由文法を生成する.

3. 提案手法

この節では, 本研究で我々が提案する XML データ圧縮手法の基本的な考え方について説明する.

本研究では, 圧縮された XML データに対する問い合わせの実行処理の効率化を目的とする. 特に, 非常に大きなデータから非常に小さなデータを抜き出すような問い合わせ処理を行う場合, データ全体を一度走査しないといけないような圧縮形式では, データ全体をディスクから読み出すI/O コストが大きな負荷となる. そこで, 本研究では, そのような問い合わせ処理において, データ全体を走査しなくても, 必要な部分のみをディスクから読み出して問い合わせの解を生成できるような圧縮方式を開発する.

本研究で提案する方式の特徴としては,

(1) 非同型圧縮を用いた, 直接問い合わせ可能な圧縮方式
(2) 文法に基づく圧縮アルゴリズムの木構造情報の部分への適用
(3) 索引情報を用いてデータ全体ではなく必要な部分のみをディスクから読み込む問い合わせ処理

の 3 点が挙げられる.

また, 実際のシステムの実装は, XMill のソースコードに改良を加える形で実現する. テキストデータの各コンテナの圧縮には, XMill 同様, コンテナ毎に最適な圧縮方式を選択する方式を取り, また, どのようなパス毎にコンテナに分類するかをユーザが手動で指定できる仕組みも, そのまま利用する.

以下, 上述の, 本研究で提案する方式の特徴となる三つの点について, 順に説明する.

3. 1 非同型圧縮の直接問い合わせ可能圧縮形式

直接問い合わせ可能な圧縮形式として当初提案された XGrind や XPress の論文では, 非同型圧縮は問い合わせに不利だと述べられていた. これは, パスに応じてコンテナに分類されたテキストデータに対してのポインタの管理が煩雑で難しいとみられていたためである. しかし, 非同型圧縮の代表的な処理系である XMill のように, あらかじめテキストデータがパスに応じてコンテナに分類されている場合, これに対して問い合わせを行う際に, その問い合わせに必要となるコンテナだけを解凍して, 必要ないことが明らかであるようなコンテナは解凍しないようにすることにより, データ全体を走査する同型圧縮のデータに対する問い合わせ処理より, I/O コストを軽減できると考えられる. そこで, 本研究では, 非同型圧縮を採用する.

3. 2 文法を用いた圧縮アルゴリズムの適用

本研究では非同型圧縮を用い, XMill 同様, XML データを, 木構造に関する情報とテキストデータに分離して個別に圧縮を行う. このうち, 木構造に関する情報の圧縮の部分に関しては, 最近活発に研究されている文法による圧縮アルゴリズム[9][23]を採用する. その理由として, XML データの木構造は, ある文法に従っていることが多いので, 文法に基づく圧縮に向いていることが予測されるからである. 文法に基づく圧縮アルゴリズムとしては, これまでに様々な物が提案されている. 本論文では, XML データの中でも比較的規則的な構造を持った物, 例えば関係データベースのスナップショットを XML 化して保存したような物を主要な対象として考えている. その場合, タグの並びのパターンで比較的長い物が, 多数繰り返し現れることが多い. その場合, 最頻 N-gram 法[21]を用いれば, 比較的容易にタグの並びの規則性を抽出できるのではないかと考えている. 最頻 N-gram 法は, 1パス目で頻出する繰り返しパターンを見つけ, それを基準にして文法を生成し, 適応型算術符号によって圧縮するアルゴリズムである. ただし, 現時点では, より実装が簡単な最頻 digram 法を用いて実装を行っており, 最頻 N-gram 法への拡張は今後の課題である.

3. 3 索引による問い合わせの高速化

前述のように, 本研究の目的は, 大きな XML データから小さな部分のみを取り出すような問い合わせ処理の高速化である. そのために, 非同型圧縮を用い, その問い合わせに関係するデータコンテナのみをディスクから読み出し解凍する点については既に述べたが, 木構造に関する情報の部分については, その全体を読み出しパーズすることになる. しかし, 小さな部分のみを取り出すような問い合わせの場合, このパーズ処理の大部分も無駄な処理となることになる. そこで, 問い合わせに関係ないことがわかっている部分のパーズは省くことができるように, 前述の stream index のような索引を用意し, 木構造のうち問い合わせに関係ない部分のパーズ処理をできるようにする. このような索引情報の与え方としては, 圧縮データ自身に埋め込む方法と, 必須ではないオプションとして, 別ファイルで与える方法が考えられるが, 本研究では, その双方について検討する予定である.

4. 圧縮/解凍処理

次に本節では，符号化と復号化のアルゴリズムについて説明をする．

符号化では，まず Fig. 2 にあるように，XMLが読み込まれる．次に，XML ファイルが，SAX パーサによってパースされ，SIX Manager によって offset がファイルに出力される．パースされたXMLファイルのテキスト部分は，Path Processor によって，パスに応じて分類される．

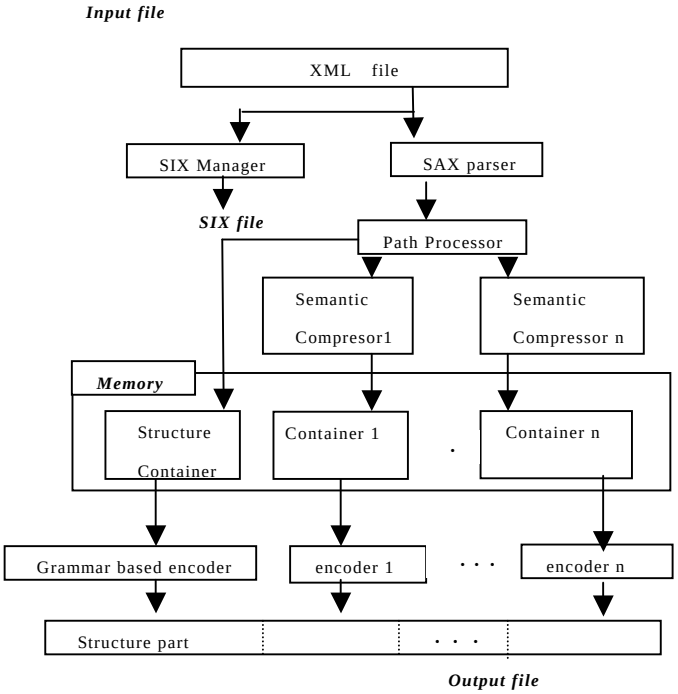


Fig. 2 Encoding

タグの部分は，structure container に格納される．semantic compressor は，各 container のデータを符号化する．structure container に格納されたタグの部分は，最頻 digram 法によって圧縮される．つまり，タグの部分を1パスで scan し，頻出するタグを基準に文法を生成する．生成された文法を2パス目に適応型算術符号化によって符号化する．各 container に格納されたデータは，データ型に応じて各圧縮アルゴリズムによって圧縮される．最後に圧縮した XML ファイルが出力される．

復号化では，圧縮された XML ファイルが読み込みこまれる．Fig. 2 にあるように，圧縮されている XML ファイルは，データ型によって，分類されている．まず，structure container が解凍される．structure container は，最頻 digram 統合法によって圧縮されているので，この部分を復号化する．次に，テキストデータは，各コンテナの中では，もとのデータに出現した順に格納されているので，structure container を先頭から解凍しながら，テキストデータが現れるはずの場所に来る度に，その場所のパスに対応するコンテナの先頭から一つデータを取ってくれば，木構造とテキストデータの対応が取れる．このようにして，structure container を解凍して得られる木構造にテキストデータをはめ込みながら，解凍された XML ファイルを出力する．

5. 問い合わせ処理

この節では，本研究での圧縮した XML ファイルに対しての問い合わせ処理の概略について説明する．問い合わせ言語としては，XPath を用いることとする．圧縮したファイルに対して，問い合わせが発行された場合，まず，文法に基づく圧縮方式で圧縮した木構造の情報を格納している structure container 部分(Fig. 3 では，C0 の部分)を先頭から解凍していく．この際に，タグのパターンと問い合わせとを照合しながら，必要に応じて SIX の情報も参照して，現在の問い合わせに不必要な部分のパーズはスキップしていく．ここで，T1, T2, T3･･･は，開始タグを符号化した物を表し，/は終了タグを符号化した物を表す．終了タグと開始タグとの対応関係によってタグ名を判定することができるので，タグ名を明示する必要はない．C1, C2, C3･･･は，その対応する箇所に入っていたテキストデータが格納されているコンテナを示している．そして，問い合わせにマッチするエレメント(T7C3)があればそのエレメントのテキストデータを格納しているコンテナのみ(C3)を解凍する．一部のコンテナのみしか解凍しないことによって，問い合わせの効率を向上できることが予想される．

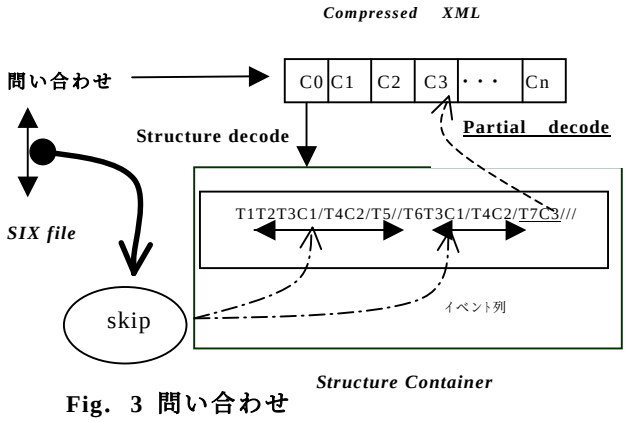


Fig. 3 問い合わせ

問題点としては，XMill では，各テキストデータは，もとの XML データに出てきた順番通りに各コンテナに保存されていき，解凍する時は，木構造と各コンテナを先頭から解凍していきながら，各エレメントとテキストデータの対応を取っていくのだが，本研究で提案する方式のように，木構造の復元を一部スキップする場合，それ以降のエレメントとテキストデータの対応がわからなくなる．そこで，一定間隔毎に，木構造中のエレメントとコンテナ中のテキストデータとの対応を表すポインタを入れることにする．これにより，木構造を大きくスキップした場合，スキップした先の

一つ手前のポインタの個所からのみ，解凍を再開すれば，エレメントとテキストデータの対応を取ることができる．

6. 実験

現在, structure 部分の符号化のみが実装されている．実験では，XML データの特徴であるタグの繰り返しパターンに文法に基づく圧縮がどのように作用するかを明らかにすることを目標とする．実験データは，XMill に添付されている 6 種類の XML データのうち Apache のログを XML 化した weblog を対象とした．添付されている weblog では，3KB 中で，タグのサイズは 1．33KB で，44%のサイズを占めている．実験環境は Sun　Blade 1500 で，言語は，C++を使用した．タグ部分に対する効果をみるために，weblog は，タグ部分を抽出し，エレメント部分を符号に置き換え，それを複数回複製した．weblog のあとの数字が複製した回数である．

	元サイズ (B)	文法数	サイズ (B)	比率 (%)	時間 (s)
weblog1	49	5	107	218	0．03
weblog10	459	9	192	41	0．11
weblog20	911	10	213	23	0．07
weblog30	1490	11	231	15	0．08
weblog210	10400	13	293	2．8	0．56
Weblog 1050	51980	16	359	0．69	5:57
Weblog 2100	103956	25	600	0．57	22:00

Fig．4 weblog の元データと文法化後の比較

Fig．4 は，weblog の元データと文法化後の比較である．比率は，文法化後のサイズを元データで割ったものである．文法数は，文法化した後の右辺の変数の数である．Weblog1 では，オーバーヘッドために文法化したときはサイズが逆に増加している．weblog10 で元データより文法化後のサイズが減少している．繰り返しパターンのために，文法化後のサイズは，減少する．

Fig．5，Fig．6，Fig．7 では，圧縮結果を示す．圧縮結果は，gzip，bzip2 と比較した．算術符号には特許があるが，Alistair　Moffat の研究用にのみ使用可能な適応型算術符号を使用した[27]．この適応型算術符号には，3 種類あり，0 次の char ベースの適応型算術符号，0 次の bit ベースの適応型算術符号，0 次の word ベースの適応型算術符号がある．実験結果により，0 次の word ベースの適応型算術符号が最も圧縮率が良かったので，その結果のみを示す．

文法化とは，文法の形式での出力のサイズである．実験では，Fig．4 の weblog210，weblog1050，weblog2100 の正準化された実験値を用いている．実験結果から，50KB 以上のサイズでは，文法による圧縮が gzip，bzip2 の圧縮率を上回っている．XML データには，タグの繰り返しパターンが多いために，このような文法による圧縮が効果的であることが認められる．Fig．4 より 100KB の weblog に対して，22 分ほどかかかっている．これに関しては，実装の上でのプログラミング技術などの影響もあり，実用化には，プログラムの最適化が必要である．

weblog210	10400(B)
gzip	113
bzip2	154
文法化した時のサイズ	151
文法化後算術符号化 (word ベース)	100

Fig．5 weblog210 の他ツールとの比較

weblog1050	51980(B)
gzip	230
bzip2	359
文法化した時のサイズ	177
文法化後算術符号化 (word ベース)	111

Fig．6 weblog1050 の他ツールとの比較

weblog2100	103956(B)
gzip	394
bzip2	586
文法化した時のサイズ	222
文法化後算術符号化 (word ベース)	153

Fig．7 weblog2100 の他ツールとの比較

7. まとめと今後の課題

現在圧縮アルゴリズム部分の実装のみであるが，問い合わせの実装が今後の課題である．今回提案した方式では, XML 文書から最頻 digram 統合法を用いて文法を生成する手法を用いているが, 今後の課題としては, 最頻 N-gram の実装と実験がある．また, XML 用のスキーマ言語である DTD, RELAX などによるスキーマが与えられている場合に, これらのスキーマ記述を利用して許容可能な文法を生成することによって, 圧縮処理の計算時間を短縮する方法について検討したいと考えている．スキーマを用いた XML データの圧縮の研究では, DTD に関しては文献[6]があり, これは, ツリーオートマトンの符号化を用いている．また,

決め打ちしたhedge automatonを利用する研究では[7]がある.

謝辞:査読者の方の懇切丁寧なご指導に感謝致します。

参考文献

[1] Hartmut Liefke and Dan Suciu, “XMill: an efficient compressor for XML Data”, In Proc. of the 2000 ACM SIGMOD, 2000.

[2] Pankaji M. Tolani and Jayant R. Haritosa, “XGrind: A query-friendly XML compressor”, In Proc. of ICDE, 2002.

[3] Jun-Ki Min, Myung-Jae Park and Chin-Wan Chung, “XPRESS: A Queriable Compression for XML Data”, In Proc. of the 2003 ACM SIGMOD, 2003.

[4] James Cheney, “Compressing XML with Multiplexed Hierarchical PPM Models”, In Proc. of Data Compression Conference (DCC2001), pp. 163-172, 2001.

[5] Wai Yeung Lam, Wilfred Ng, Peter Wood and Mark Levene, “XCQ: XML Compression and Querying System”, WWW conference, poster, 2003.

[6] Mark Levene and Peter Wood, “XML Structure Compression”, Technical Report BBKCS-02-05, School of Computer Science and Information Systems, Birkbeck College, University of London, 2002.

[7] Aditya Nori, Priti Shankar and Helmut Seidl, “Using Hardcoded Hedge Automata for Compressing Structure Documents”, Technical Report No. IISc-CSA-02-08, Department of Computer Science and Automation, IISc, July 2002.

[8] 河野正太郎, 有村博紀, 有川節夫, “半構造データ系列のオンライン予測とXMLデータ圧縮への応用”, DEWS, 2003.

[9] 山本博資, 植松友彦, 横尾英俊, 古賀弘樹, 星守, “データ圧縮における最新アルゴリズム[Ⅰ]～[Ⅴ]”, 電子通信情報処理学会誌, vol86, No. 2～7, 2003.

[10] 植松友彦, “文書データ圧縮アルゴリズム入門”, CQ出版, 1995.

[11] 韓太舜, 小林欣悟, “情報と符号化の数理”, 培風館, 1999.

[12] Craig. G. Nevill - Manning and I. H. Witten, “Compression and explanation using hierarchical grammars”, Computer Journal 40(2/3), pp. 103-116, 1997.

[13] Craig. G. Nevill-Manning, “Inferring Sequential Structure”, PhD thesis, Waikato. Univ., 1996.

[14] John C. Kieffer and En-hui Yang, “Grammar-Based Codes: A New Class of Universal Lossless Source Codes”, IEEE Trans. Inf. Theory, vol IT-46, pp. 737-754, 2000.

[15] En-hui Yang and John C. Kieffer: “Efficient Universal Lossless Data Compression Algorithms Based on a Greedy Sequential Grammar transform - Part One: Without Context Models”, IEEE Trans. Inf. Theory, vol IT-46, pp. 755-777, 2000.

[16] Eric Lehman, “Approximation Algorithms for Grammar based Data Compression”, PhD thesis, MIT, 2002.

[17] Abhi shelat, “Evaluting Grammar-Based Data Compression Algorithms”, PhD thesis, MIT, 2001.

[18] Wojciech Rytter, “Application of Lempel-Ziv factorization to the approximation of grammar-based compression”, In Proc. of 13th Ann. Sympo. Combinatorial Pattern Matching, pp. 20-31, 2002.

[19] Hiroshi Sakamoto, “A Fully Linear-Time Approximation Algorithm for Grammar-Based Compression”, In Proc. of 14th Ann. Sympo. Combinatorial Pattern Matching (CPM 2003), pp. 348-360, Springer-Verlag.

[20] 竹川 視野, 山本 博資, “逐次符号可能な改良MPM符号の漸近的圧縮性能”, 信学技法, vol. 101, No. 725, pp. 121-126, Mar 2002.

[21] 神田 勝, 石田 崇, 小林 学, 平澤 茂一, “最頻Digram統合に基づくデータ圧縮法”, 信学技法, vol. 102, No. 198, pp. 31-36, July 2002.

[22] 大石 昭久, 石田 崇, 平澤 茂一, “最頻N-gramを考慮した文法生成法に基づくデータ圧縮法”, 信学技法, vol103, No. 214, pp. 13-18, July 2003.

[23] 中澤 真, 松嶋 敏泰, 平澤 茂一, “形式言語と圧縮に関する一考察”, 信学技法, vol. 101, No. 214, pp. 19-24, Nov 2001.

[24] Todd J. Green, Gerome Miklau, Makoto Onizuka, Dan Suciu, “Processing XML Streams with Deterministic Automata and Stream Indexs”, http://www.cs.washington.edu/homes/suciu/XMLTK/planx.pdf, PlanX 2002.

[25] Andrei Arion, Angela Bonifati, Gianni Costa, Sandra d'Aguanno, Ioana ManoLescu and Andrea Pugliese, “XQueC:Pushing queries to compressed XML data”, VLDB 2003.

[26] Andrei Arion, Angela Bonifati, Gianni Costa,

Sandra d'Aguanno and Ioana ManoLescu, "Efficient query evalution over compressed XML data", accepted for publication in EDBT 2004.

[27] Alistair Moffat, adaptive artithmetric encoding, ftp://munnari. oz. au/pub/arith_coder/

[28] Roy Goldman and Jennifer Widom, "DataGuides: Enabling Query Formulation and Optimization in Semistructured Databases", In Proc. of VLDB, pp. 436-445, 1997.