Vol.2012-DBS-154 No.12
Vol.2012-IFAT-107 No.12
2012/8/2

正規木文法間の差分抽出アルゴリズムの提案

堀江 和磨[1]　鈴木 伸崇[2]

概要：XML は Web 上の標準的なデータ記述フォーマットとして広く普及している．XML データをデータベース等で継続的に蓄積・管理する場合，格納すべきデータの構造をスキーマで定義しておき，それに沿った構造のデータを作成・格納することが一般的である．利用状況の変化により格納すべきデータの構造や種類が変化するため，それに応じてスキーマ定義も更新されることが多い．その場合，スキーマの更新履歴の管理やスキーマの更新に応じた XML データの修正等が必要となるため，スキーマ間の差分抽出アルゴリズムが有用である．そこで本稿では，正規木文法のための差分抽出問題について考察し，同問題が計算困難であること，および，同問題が効率良く解けるための十分条件を示す．また，その十分条件の下で動作する多項式アルゴリズムを求め，このアルゴリズムに関する評価実験を行う．

1. はじめに

XML は Web 上の標準的なデータ記述フォーマットとして広く普及している．XML データをデータベース等で継続的に蓄積・管理する場合，格納すべきデータの構造をスキーマで定義しておき，それに沿った構造のデータを作成・格納することが一般的である．また，利用状況の変化により格納すべきデータの構造や種類が変化するため，それに応じてスキーマ定義も更新されることが多い．このような状況では，スキーマの更新履歴の管理，スキーマの更新に応じた XML データの修正等が必要となるため，スキーマの更新内容を適切に把握しておく必要がある．特に，管理者が複数で更新内容の共有が必要な場合や，スキーマが複雑で更新内容が多岐にわたる場合等は，スキーマの更新内容を把握することがより重要となる．スキーマの更新内容を把握するには更新前後のスキーマ間で差分抽出を行う必要があるが，これを適切に行える手法はこれまでほとんど提案されていない．本研究は，XML のスキーマ定義言語として広く用いられている正規木文法を対象とし，正規木文法のための差分抽出アルゴリズムの開発を行う．

これまで，文字列間の差分 (編集操作列) に関しては基本的なアルゴリズムが確立しており，順序木や XML データ間の差分抽出に関しても，順序木の編集操作列を求めるアルゴリズムがいくつか提案されている．しかし，これら既存のアルゴリズムは木文法の意味を解することができないため，正規木文法の適切な差分抽出を行うことは困難である．実際，文字列間や順序木間の差分抽出は多項式時間可解であるが，正規木文法の差分抽出は文法の表す意味も絡み，より複雑な問題である．また，文献 [4] において，スキーマを DTD に限定した場合でも，XML データの差分抽出アルゴリズムではスキーマの差分抽出が適切に行えないとの指摘もされている．以上から，本研究では正規木文法の差分集出問題について考察し，以下の結果を示した．

(1) 正規木文法の差分抽出問題の計算複雑さについて考察し，同問題が計算困難であることを示した
(2) 正規木文法の差分抽出が効率よく行えるための十分条件を求めた
(3) 上記十分条件の下で，正規木文法の差分抽出を行う効率の良いアルゴリズムを構成した．更に，このアルゴリズムを実装し評価実験を行った

正規木文法は終端記号 (要素名) の集合，非終端記号 (要素の型) の集合，開始記号，および生成規則の集合から構成される (図 1)．正規木文法の差分抽出は，2 つの正規木文法 G と G' が与えられた時に，G を G' へ更新するために必要なコスト最小の編集操作列を求めることをいう．ここで，編集操作列とは「生成規則の追加・削除」等の編集操作の系列であり，各編集操作にはコストが付与される．なお，2 つの正規木文法 G と G' が与えられた時に，G の内容をすべて削除して G' の内容を追加すれば G を G' に更新するための編集操作列が得られるが，そのような差分を抽出するのは無意味である．そこで，本研究ではコスト最小の編集操作列を差分として抽出する．

[1] 筑波大学大学院 図書館情報メディア研究科
Graduate School of Library, Information and Media Studies, University of Tsukuba
[2] 筑波大学 図書館情報メディア系
Faculty of Library, Information and Media Studies, University of Tsukuba

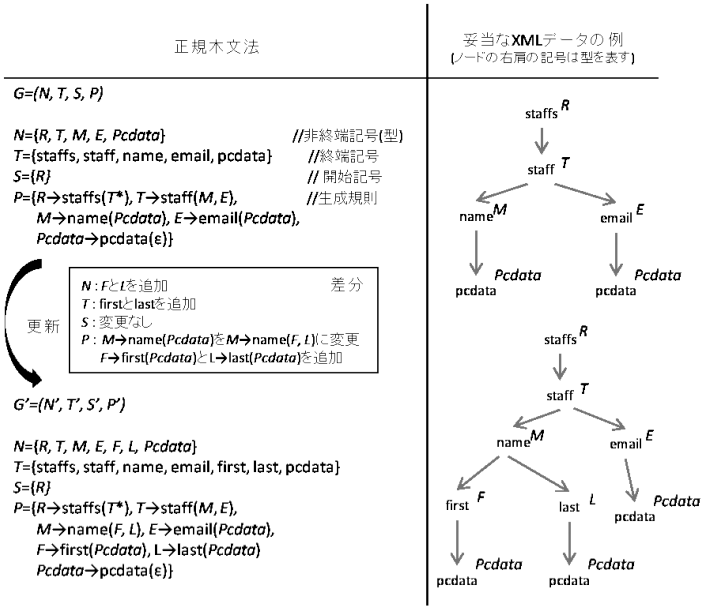

図 1 正規木文法の例

提案アルゴリズムを Ruby を用いて実装し評価実験を行った．その結果，本アルゴリズムを用いた場合，DiffMk[11] や X-Diff[12] を用いた場合と比較して，より適切にスキーマ間の更新内容が把握できるという結果が得られた．これまで，文字列や順序木間の編集操作列を求めるアルゴリズムは多数提案されている (文献 [1], [6] 等)．しかし，上述のように，これらのアルゴリズムを用いて正規木文法の差分抽出を適切に行うのは困難である．文献 [4] では DTD の差分抽出を行うアルゴリズムが提案されているが，これはヒューリスティックに基づく手法であり，最適解 (コスト最小の編集操作列) が得られるとは限らない．また，正規木文法の表現力は DTD より真に高いため，このアルゴリズムを正規木文法の差分抽出に適用することは不可能である．文献 [8] では，更新前後のスキーマの差分が得られているという仮定の下で，その差分から適切な XML 変換操作列，すなわち，スキーマの更新により妥当でなくなった XML データを妥当なものに変換する操作列を推測・生成するアルゴリズムが提案されている．しかし，同文献では差分抽出の方法については議論されていない．

本稿の構成は以下の通りである．2 章では，正規木文法に関する定義を行う．3 章では，正規木文法の差分抽出問題に関する計算複雑さについて考察する．4 章では，正規木文法の差分集出問題が効率良く解けるための十分条件を示す．5 章では，4 章で得られた十分条件の下で多項式時間で動作する差分抽出アルゴリズムを示す．5 章では，評価実験について述べる．6 章では，まとめと今後の課題を述べる．

```
G=(N, T, S, P)

N={Doc, Para1, Para2, Pcdata}                    //非終端記号(型)
T={doc, para, pcdata}                            //終端記号
S={Doc}                                          // 開始記号
P={Doc→doc(Para1, Para2*), Para1→para(Pcdata),   //生成規則
   Para2→para(Pcdata), Pcdata→pcdata(ε)}
```

図 2 正規木文法 G

2. 諸定義

2.1 正規木文法

文献 [5] に倣い，正規木文法を定義する．正規木文法 (regular tree grammar) は 4 つ組 $G = (N, T, S, P)$ と表される．ここで，

- N は非終端記号の集合
- T は終端記号の集合
- S は開始記号で $S \subseteq N$
- P は生成規則の集合

である．

終端記号は，木のノードのラベルとなる記号で，XML データの要素名に相当する．非終端記号は，木を生成する過程で中間的に用いられる記号で，XML データ内にそのまま出現することはない．以下，非終端記号の先頭は大文字，終端記号の先頭は小文字を用いて区別することにする．また，生成規則は $X \to a(r)$ という形をしており，X は非終端記号，a は終端記号，r は N 上の正規表現である．正規木文法 $G = (N, T, S, P)$ の例を示す (図 2)．

2.2 解釈

正規木文法 G と木 t が与えられた時に，t が G から生成

G'=(N', T', S', P')

N'={*A, B, C, D*} //非終端記号(型)
T'={doc, para, pcdata} //終端記号
S'={*A*} // 開始記号
P'={*A*→doc(B, *C**), *B*→para*(D)*, //生成規則
C→para(*D*), *D*→pcdata(ε)}

図 3 正規木文法 G'

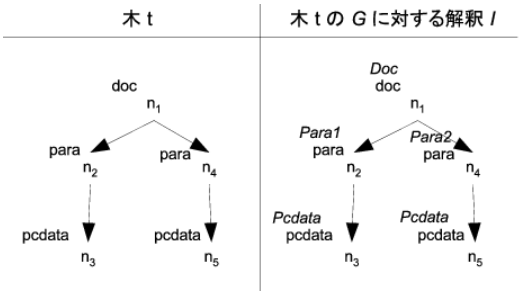

図 4 木 t および木 t の G に対する解釈 I

可能なものである場合，t は G に関して妥当 (valid) であるという．このことをより形式的に定義する．まず，木の「解釈」を定義する．$G = (N, T, S, P)$ を正規木文法，t を木とする．もし t の各ノードに対する非終端記号の割り当て I が次の条件を満たすならば，I は t の G に対する解釈 (interpretation) という．

- t のルートノード n に対して，$I(n) \in S$，すなわち，$I(n)$ は G の開始記号である
- t の各ノード n に対して，P は次の条件を満たす生成規則 $X \to a(r)$ を含む
  n の子ノードを $n_1, ..., n_k$ とする
  - $I(n) = X$
  - n のラベルが a と一致する
  - $I(n_1)...I(n_k) \in L(r)$，すなわち，非終端記号列 $I(n_1)...I(n_k)$ が r にマッチする

木 t と，図 2 の正規木文法 $G = (N, T, S, P)$ を考える．木 t の G に対する解釈 I を図 4 に示す．これを式で表すと，$I(n_1) = Doc$, $I(n_2) = Para1$, $I(n_3) = Pcdata$, $I(n_4) = Para2$, $I(n_5) = Pcdata$ となる．もし木 t の G に対する解釈が存在するならば，t は G に関して妥当 (valid) であるという．$L(G)$ を，G に関して妥当な木の集合と定義する．

2.3 単一型木文法と局所木文法

正規木文法では，妥当な木に対する解釈が複数存在することがある．解釈を一意に決めるために，正規木文法に制限を加えた木文法も提案されている．そのように制限を加えられた木文法として単一型木文法が挙げられる．2 つの異なる非終端記号 A と B に対して，次の条件が成り立つとき，A と B は競合するという．

- 左辺が A である生成規則と，左辺が B である生成規則が存在し，それらの右辺の終端記号が同一である．

G=(N, T, S, P)

N={*Doc1, Doc2, Para1, Para2, Pcdata*} //非終端記号(型)
T={doc1, doc2, para, pcdata} //終端記号
S={*Doc1, Doc2*} // 開始記号
P={*Doc1*→doc1(*Para1*), *Doc2*→*doc2(Para2*)*, //生成規則
Para1→para*(Pcdata)*, *Para2*→para(*Pcdata*), *Pcdata*→pcdata(ε)}

図 5 単一型木文法

図 5 は $Para1$ と $Para2$ の右辺の終端記号は共に $para$ であり，定義から競合していることがわかる．正規木文法のうち，次の 2 つの条件を満たすものを単一型木文法 (single-type tree grammar) という．

- 各生成規則に対して，その右辺の生成規則に出現する非終端記号は競合しない
- 開始記号は競合しない

例えば，図 5 の木文法は単一型木文法である．一方，図 2 の文法は単一型木文法ではない．

また，競合する非終端記号をもたない正規木文法を局所木文法 (local tree grammar) という．

2.4 編集操作列

正規木文法における，編集操作列とは以下のような編集操作の系列である．形式的な定義は省略する．

- 生成規則の追加・削除
- 生成規則の左辺の非終端記号の変更
- 生成規則の右辺の終端記号の変更
- 内容モデルにおける，非終端記号や演算子 (*,?など) の追加・削除・変更．内容モデルを順序木として表し，順序木に対する編集操作列として定義する．

3. 正規木文法の差分抽出問題の計算複雑さ

本章では，正規木文法の差分抽出問題の計算複雑さについて考える．まず，正規木文法の差分抽出問題 (文法の意味を考慮した場合) を次のように定義する．

- 正規木文法 G,G' およびコストの上限 B が与えられたときに，G を G' と等価，すなわち，$L(G) = L(G')$ を満たす文法に変換するコスト B 以下の編集操作列が存在するか否かを決定せよ

次の定理に示すように，この問題は EXPTIME 困難である．

定理 1 文法の意味を考慮した場合，正規木文法の差分抽出問題は EXPTIME 困難である．

証明：正規木文法の等価性問題は EXPTIME 完全であることが文献 [7] の結果から示せる．この問題を正規木文法の差分抽出問題に帰着する．まず，正規木文法の等価性問題は次のように定義される．

- 正規木文法 G と G' が与えられたときに，G と G' が等価か否か，すなわち $L(G) = L(G')$ か否かを決定せよ

次に，正規木文法の等価性問題を以下のようにして上記の

差分抽出問題に帰着する．

- 入力：正規木文法 G と G', およびコストの上限 $B(B=1$ とする)
- すべての編集操作のコストを 2 と設定

このとき，$L(G)=L(G')$ であることと，G を G' と等価な文法に変換するコスト B 以下の編集操作列が存在することは同値である． □

局所木文法に限定した場合の計算困難性も同様に示せる (証明は省略する)．

定理 2 文法の意味を考慮した場合，局所木文法の差分抽出問題は PSPACE 困難である． □

上記の結果から，文法の意味を考慮すると正規木文法の差分抽出問題は極めて複雑な問題となる．そこで，より限定的で判定の容易な等価性の概念を導入する．これは正規木文法の構文的な等価性であり，正規木文法 $G=(N,T,S.P)$ と $G'=(N',T',S'.P')$ に対して，$N=N'$，$T=T'$，$S=S'$，かつ，$P=P'$(すべての生成規則が一致する) であるとき，G と G' は構文的に等価であるという．例えば，図 2 の正規木文法 G と図 3 の正規木文法 G' は，$L(G)=L(G')$ であるが構文的には等価ではない．

以下に示すように，構文的な等価性を用いた場合，複雑さの下限は下がるが，正規木文法の差分抽出問題は依然として計算困難である．

定理 3 文法間の等価性を構文的なものに限定しても，正規木文法の差分抽出問題は NP 困難である．

証明：正規木文法の差分抽出問題の NP 困難性を，3-partition 問題からの帰着により示す．3-partition 問題は次のように定義される NP 完全問題である．

入力： $|S|=3m$ なる整数集合 $S=\{i_1,\cdots,i_{3m}\}$ と正整数 B．ただし，任意の $1\le j\le 3m$ に対して $B/4<i_j<B/2$ と仮定する．

問題： S を，次の条件を満たす m 個の集合 $S_1,\cdots,S_m$ に分割できるか否か．

- $S_1,\cdots,S_m$ の要素数はそれぞれ 3 である．
- 任意の $1\le k\le m$ に対して $\sum_{i\in S_k} i=B$．すなわち，$S_i,\cdots,S_m$ に属する整数の和がいずれも B となる．

3-partition 問題から正規木文法の差分抽出問題への帰着を行う．上記 3-partition 問題から，正規木文法の差分抽出問題のインスタンスを構成する．

2 つの正規木文法 G と G' を構成する．まず，$G=(N,T,S,P)$ を次のように定義する．

$$
\begin{aligned}
N &= \{X_r\}\cup\{X_{k,j}\mid 1\le k\le 3m, 1\le j\le i_k\}\\
T &= \{r,n,pcdata\}\\
S &= \{X_r\}\\
P &= \{X_r\to r(X_{1,1}\cdots X_{3m,1})\}\cup P_1\cup\cdots\cup P_{3m}\cup P_c
\end{aligned}
$$

ここで，$P_j(1\le j\le 3m)$ は次のような i_j-1 個の生成規則からなる集合である．

$$
\begin{aligned}
P_j = \{&X_{j,1}\to n(X_{j,2}), X_{j,2}\to n(X_{j,3}),\cdots,\\
&X_{j,i_j-1}\to n(X_{j,i_j})\}
\end{aligned}
$$

また，P_c は次のような生成規則の集合である．

$$P_c=\{X_{1,i_1}\to pcdata(\epsilon),\cdots,X_{3m,i_{3m}}\to pcdata(\epsilon)\}$$

G に妥当な木を図 6(左) に示す．

次に，$G'=(N',T',S',P')$ を次のように定義する．

$$
\begin{aligned}
N' &= \{Y_r\}\cup\{Y_{k,j}\mid 1\le k\le m, 1\le j\le B\}\\
T' &= \{r,n,pcdata\}\\
S' &= \{Y_r\}\\
P' &= \{Y_r\to r(Y_{1,1}\cdots Y_{m,1})\}\cup P'_1\cup\cdots\cup P'_m\cup P'_c
\end{aligned}
$$

ここで，$P'_j(1\le j\le m)$ は次のような $B-1$ 個の生成規則からなる集合である．

$$
\begin{aligned}
P'_j = \{&Y_{j,1}\to n(Y_{j,2}), Y_{j,2}\to n(Y_{j,3}),\cdots,\\
&Y_{j,B-1}\to n(Y_{j,B})\}
\end{aligned}
$$

また，P'_c は次のような生成規則の集合である．

$$P'_c=\{Y_{1,B}\to pcdata(\epsilon),\cdots,Y_{m,B}\to pcdata(\epsilon)\}$$

G' に妥当な木を図 6(右) に示す．

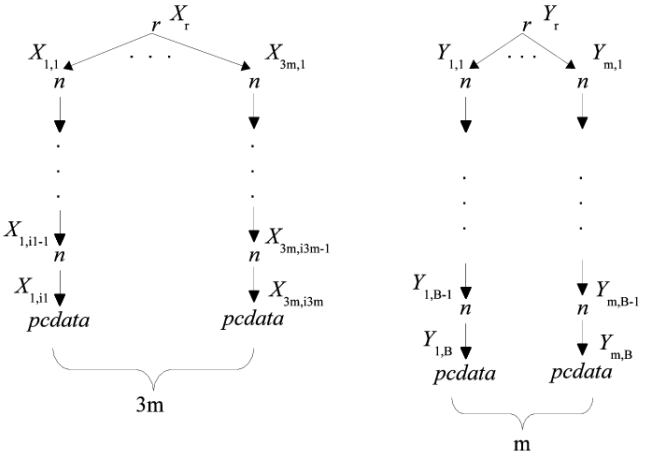

図 6 G に妥当な木 (左) と G' に妥当な木 (右)

編集操作に対するコストに関して，非終端記号の置換はコスト 1 とする．また，内容モデルの更新は，生成規則 $X_r\to r(X_{1,1}\cdots X_{3m,1})$ に対してのみ許し，その (各編集操作に対する) コストを 1 とする．それ以外のコストは ∞ とする．

以下，S が 3-partition の条件を満たす $S_1,\cdots,S_m$ に分割できることと，G から G' への差分抽出コストが $(5+B)m$ 以下であることが同値であることを示す．

まず，S が 3-partition の条件を満たす $S_1,\cdots,S_m$ に分割できると仮定する．このとき，任意の $1\le j\le m$ に対し

て $S_j = \{i_{j_1}, i_{j_2}, i_{j_3}\}$ と表すことができ，3-partition の条件から $i_{j_1} + i_{j_2} + i_{j_3} = B$ である．G から G' への差分抽出コストが $(5+B)m$ であることを示す．G に対して，次のような更新操作を適用する．

(1) 生成規則 $X_r \rightarrow r(X_{1,1} \cdots X_{3m,1})$ の内容モデルを $Y_{1,1} \cdots Y_{m,1}$ に更新する

(2) 各 $1 \leq j \leq m$ に対して，以下を適用する
- 非終端記号 $X_{j_1,1}$ を $Y_{j,1}$ に置換する
- $2 \leq k \leq i_{j_1}$ に対して，非終端記号 $X_{j_1,k}$ を $Y_{j,k}$ に置換する
- 非終端記号 $X_{j_2,1}$ を $Y_{j,i_{j_1}+1}$ に置換する
- $2 \leq k \leq i_{j_2}$ に対して，非終端記号 $X_{j_2,k}$ を $Y_{j,i_{j_1}+k}$ に置換する
- 非終端記号 $X_{j_3,1}$ を $Y_{j,i_{j_1}+i_{j_2}+1}$ に置換する
- $2 \leq k \leq i_{j_3}$ に対して，非終端記号 $X_{j_3,k}$ を $Y_{j,i_{j_1}+i_{j_2}+k}$ に置換する

(3) P_c の生成規則のうち，不要なもの (開始記号から到達不可能なもの) を削除する．このような生成規則は $3m - m = 2m$ 個存在する．

上記の編集操作で得られた文法が G' と一致することは容易に示せる．上記の編集コストを考えると，1 は $3m$，2 は Bm，3 は $2m$ である．したがって，コストの合計は $(5+B)m$ である．

逆に，S が上記条件を満たす $S_1, \cdots, S_m$ に分割できないと仮定する．このとき，生成規則の追加・削除／内容モデルの変更が行えない限り，G' と等価な文法を得ることは不可能である．生成規則の追加のコストは ∞ であるため，この場合のコストは ∞ となる． □

次に，単一型木文法についても同様にして計算複雑さを考える．

定理 4 文法間の等価性を構文的なものに限定しても，単一型木文法の差分抽出問題は NP 困難である．

証明：単一型木文法の差分抽出問題の NP 困難性を，3-partition 問題からの帰着により示す．3-partition 問題は次のように定義される NP 完全問題である．

入力： $|S| = 3m$ なる整数集合 $S = \{i_1, \cdots, i_{3m}\}$ と正整数 B．ただし，任意の $1 \leq j \leq 3m$ に対して $B/4 < i_j < B/2$ と仮定する．

問題： S を，次の条件を満たす m 個の集合 $S_1, \cdots, S_m$ に分割できるか否か．
- $S_1, \cdots, S_m$ の要素数はそれぞれ 3 である．
- 任意の $1 \leq k \leq m$ に対して $\sum_{i \in S_k} i = B$．すなわち，$S_i, \cdots, S_m$ に属する整数の和がいずれも B となる．

3-partition 問題から単一型木文法の差分抽出問題への帰着を行う．上記 3-partition 問題から，単一型木文法の差分抽出問題のインスタンスを構成する．

2 つの単一型木文法 G と G' を構成する．まず，$G = (N, T, S, P)$ を次のように定義する．

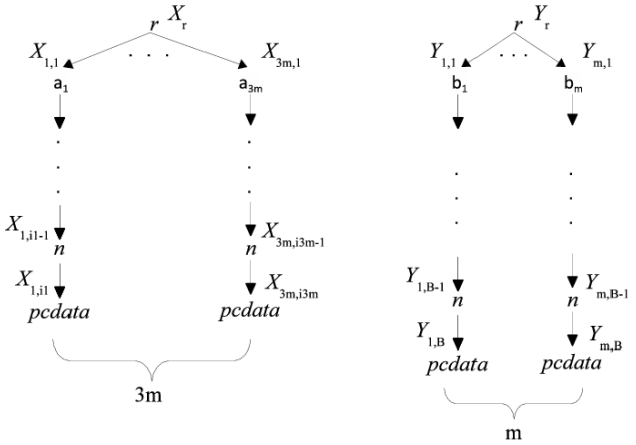

図 7 G に妥当な木 (左) と G' に妥当な木 (右)

$$
\begin{aligned}
N &= \{X_r\} \cup \{X_{k,j} \mid 1 \leq k \leq 3m, 1 \leq j \leq i_k\} \\
T &= \{r, n, pcdata\} \\
S &= \{X_r\} \\
P &= \{X_r \rightarrow r(X_{1,1} \cdots X_{3m,1})\} \cup P_1 \cup \cdots \cup P_{3m} \cup P_c
\end{aligned}
$$

ここで，$P_j (1 \leq j \leq 3m)$ は次のような $i_j - 1$ 個の生成規則からなる集合である．

$$
\begin{aligned}
P_j = \{&X_{j,1} \rightarrow a_j(X_{j,2}), X_{j,2} \rightarrow n(X_{j,3}), \cdots, \\
&X_{j,i_j-1} \rightarrow n(X_{j,i_j})\}
\end{aligned}
$$

また，P_c は次のような生成規則の集合である．

$$P_c = \{X_{1,i_1} \rightarrow pcdata(\epsilon), \cdots, X_{3m,i_{3m}} \rightarrow pcdata(\epsilon)\}$$

G に妥当な木を図 7(左) に示す．

次に，$G' = (N', T', S', P')$ を次のように定義する．

$$
\begin{aligned}
N' &= \{Y_r\} \cup \{Y_{k,j} \mid 1 \leq k \leq m, 1 \leq j \leq B\} \\
T' &= \{r, n, pcdata\} \\
S' &= \{Y_r\} \\
P' &= \{Y_r \rightarrow r(Y_{1,1} \cdots Y_{m,1})\} \cup P'_1 \cup \cdots \cup P'_m \cup P'_c
\end{aligned}
$$

ここで，$P'_j (1 \leq j \leq m)$ は次のような $B - 1$ 個の生成規則からなる集合である．

$$
\begin{aligned}
P'_j = \{&Y_{j,1} \rightarrow b_j(Y_{j,2}), Y_{j,2} \rightarrow n(Y_{j,3}), \cdots, \\
&Y_{j,B-1} \rightarrow n(Y_{j,B})\}
\end{aligned}
$$

また，P'_c は次のような生成規則の集合である．

$$P'_c = \{Y_{1,B} \rightarrow pcdata(\epsilon), \cdots, Y_{m,B} \rightarrow pcdata(\epsilon)\}$$

G' に妥当な木を図 7(右) に示す．

編集操作に対するコストに関して，非終端記号の置換はコスト 1 とする．終端記号の置換は，a_{j_1} を b_j，a_{j_2} を n，a_{j_3} を n に対してのみ許し，それぞれコストは 1 とする．また，内容モデルの更新は，生成規則 $X_r \rightarrow r(X_{1,1} \cdots X_{3m,1})$

表 1 差分抽出問題の計算複雑さ

文法	文法間の等価性	
	意味	構文
正規木文法	EXPTIME 困難	NP 困難
単一型木文法	PSPACE 困難	NP 困難
局所木文法	PSPACE 困難	PTIME

に対してのみ許し，その (各編集操作に対する) コストを 1 とする．それ以外のコストは ∞ とする．

以下，S が 3-partition の条件を満たす $S_1, \cdots, S_m$ に分割できることと，G から G' への差分抽出コストが $(B+8)m$ 以下であることが同値であることを示す．

まず，S が 3-partition の条件を満たす $S_1, \cdots, S_m$ に分割できると仮定する．このとき，任意の $1 \leq j \leq m$ に対して $S_j = \{i_{j_1}, i_{j_2}, i_{j_3}\}$ と表すことができ，3-partition の条件から $i_{j_1} + i_{j_2} + i_{j_3} = B$ である．G から G' への差分抽出コストが $(B+5)m$ であることを示す．G に対して，次のような更新操作を適用する．

(1) 生成規則 $X_r \rightarrow r(X_{1,1} \cdots X_{3m,1})$ の内容モデルを $Y_{1,1} \cdots Y_{m,1}$ に更新する

(2) 各 $1 \leq j \leq m$ に対して，以下を適用する
- 非終端記号 $X_{j_1,1}$ を $Y_{j,1}$ に置換する
- 終端記号 a_{j_1} を b_j に置換する
- $2 \leq k \leq i_{j_1}$ に対して，非終端記号 $X_{j_1,k}$ を $Y_{j,k}$ に置換する
- 非終端記号 $X_{j_2,1}$ を $Y_{j,i_{j_1}+1}$ に置換する
- 終端記号 a_{j_2} を n に置換する
- $2 \leq k \leq i_{j_2}$ に対して，非終端記号 $X_{j_2,k}$ を $Y_{j,i_{j_1}+k}$ に置換する
- 非終端記号 $X_{j_3,1}$ を $Y_{j,i_{j_1}+i_{j_2}+1}$ に置換する
- 終端記号 a_{j_3} を n に置換する
- $2 \leq k \leq i_{j_3}$ に対して, 非終端記号 $X_{j_3,k}$ を $Y_{j,i_{j_1}+i_{j_2}+k}$ に置換する

(3) P_c の生成規則のうち，不要なもの (開始記号から到達不可能なもの) を削除する．このような生成規則は $3m - m = 2m$ 個存在する．

上記の編集操作で得られた文法が G' と一致することは容易に示せる．上記の編集コストを考えると，1 は $3m$，2 は $(B+3)m$，3 は $2m$ である．したがって，コストの合計は $(B+8)m$ である．

逆に，S が上記条件を満たす $S_1, \cdots, S_m$ に分割できないと仮定する．このとき，生成規則の追加・削除／内容モデルの変更が行えない限り，G' と等価な文法を得ることは不可能である．生成規則の追加のコストは ∞ であるため，この場合のコストは ∞ となる． □

これまでの結果をまとめたものを表 1 に示す．

以下では，正規木文法の構文的な等価性を用いるものと仮定する．

G=(N, T, S, P)

N={A, B, C, D}
T={a, c, pcdata}
S={A}
P={*A*→a(B, *C, E*), *B*→pcdata(ε), *C*→c(*D, B**),
E→*c(D, B*), D*→pcdata(ε)}

等価 ： この仮定を置いても一般性は失わない.

G'=(N', T', S', P')

N'={A, B, C}
T'={a, c, pcdata}
S'={A}
P'={*A*→a(B, *C, C*), *B*→pcdata(ε), *C*→c(*D, B**),
D→pcdata(ε)}

図 8 正規木文法に対する仮定

4. 差分抽出が効率よく行える十分条件

本章では，正規木文法の差分抽出が効率よく行える十分条件を示す．

まず，生成規則の集合が「左辺の非終端記号が異なり，かつ右辺が同一」である生成規則を含まないと仮定する．図 8 に示すように，この仮定を満たさない正規木文法はすべてこの仮定を満たすものに変換できる．したがって，この仮定を置いても一般性を失わない．

更に，編集操作「生成規則の左辺の非終端記号の変更」は以下の条件の下でのみ許される，という制限を設ける．
- 生成規則の左辺の非終端記号を変更する場合，その規則の右辺は変更されない
- 既に使われている非終端記号への変更は行わない

前述の仮定と上記 2 つの条件が成り立つ場合，「どの生成規則の非終端記号が変更されたか」は一意に特定できる．したがって，更新前後の生成規則を比較して，右辺が一致しかつ左辺が一致しないものは，非終端記号が変更されたとみなせる．なお，上記 2 条件は正規木文法に対する制限ではなく，編集操作に対する制限であることに注意．

以上の制限により，正規木文法の差分抽出問題は多項式時間で効率よく解くことができる．そのためのアルゴリズムを次章で示す．

5. 正規木文法の差分抽出アルゴリズム

本章では，前章で示した十分条件の下で正規木文法の差分抽出を行う多項式時間アルゴリズムを示す．

2 つの正規木文法 $G = (N, T, S, P)$ と $G' = (N', T', S', P')$ の差分抽出を行うことを考える．本アルゴリズムは，大きく分けて以下の四つの差分抽出から成り立つ．
- N と N' における差分抽出
- T と T' における差分抽出

- S と S' における差分抽出
- P と P' における差分抽出

ここで，N と N'，T と T'，S と S' における差分抽出はそれぞれの集合を比較することで容易に可能である．一方，P と P' における差分抽出は，P における生成規則とそれに対応する P' における生成規則の組ごとに，それぞれの生成規則の右辺の差分をとる必要がある．なお，前節で示した十分条件から，対応する生成規則は一意に決まる．本アルゴリズムでは，この「生成規則の右辺」を順序木とみなし (図 9)，木編集距離を求めるアルゴリズム [1] を応用して差分を求める．

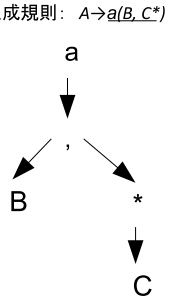

図 9 生成規則の右辺を木で表した例

時間計算量

$G = (N, T, S, P)$ と $G' = (N', T', S', P')$ に対して，上記アルゴリズムを用いて G と G' の差分抽出を行うために要する時間計算量を考える．まず，N と N' の差は $O(|N|+|N'|)$ で得られる．同様に，T と T' の差は $O(|T|+|T'|)$，S と S' の差は $O(|S|+|S'|)$ で得られる．最後に，P と P' の差分について考える．まず，P と P' の間の生成規則の対応を求めるのに $O(|P|\cdot|P'|\cdot|r_{max}|)$ 要する．ここで，$|P|$ は P に含まれる生成規則の数，r_{max} は P と P' における生成規則のうち右辺のサイズが最大のものを表し，$|r_{max}|$ はそのサイズである．また，対応する生成規則の対 (r, r') に対して，r と r' の右辺の木表現をそれぞれ T，T' とし，$n = \max(|T|, |T'|)$ とすると，上記の $\delta(T, T')$ を求めるには $O(n^3)$ 要する．更に，G に関して $|N| \leq |P|$，$|T| \leq |P|$，かつ $|S| \leq |P|$ と仮定して一般性を失わない (G' も同様)．以上から，本アルゴリズムの時間計算量は次の通りである．

$$O(|P|\cdot|P'|\cdot|r_{max}| + |P \cap P'|\cdot(|r_{max}|)^3)$$

ここで，$|P \cap P'|$ は P と P' との間で対応する生成規則の数である．

上記のように，アルゴリズムの時間計算量の最大次数は 3 であるが，r_{max} は個々の生成規則の右辺のサイズである．これは，通常，正規木文法全体のサイズと比べても相当小さいため，実用上の問題はほとんど生じないと考えられる．

6. 評価実験

提案アルゴリズムを Ruby で実装し，評価実験を行った．実験環境は以下の通りである．

- CPU：IntelCore2 Quad Q8400 2.66Ghz
- メモリ：4.00GB
- OS バージョン：Windows 7 Home Premium
- 使用言語：Ruby 1.9.1p378

実験の具体的な手順は以下の通りである．

(1) RELAX NG スキーマを用意する．今回は，relaxng.rng[9] と VoiceXML10_full[10] を用いた．
(2) それぞれの RELAX NG スキーマに対してランダムに生成した編集操作列を適用し，更新後のスキーマを作成する．これにより，「更新前の RELAX NG スキーマと更新後の RELAX NG スキーマ」の組が得られる
(3) 2 で得られた RELAX NG の組に対して本アルゴリズム，X-Diff[12]，Diffmk[11] をそれぞれ適用し，差分を抽出する．

使用したスキーマと適用した編集操作のサイズは以下に示す．

- パターン 1a：
  - 使用スキーマ：relaxng.rng (要素数 44)
  - 編集操作数 (差分長)：13
- パターン 1b：
  - 使用スキーマ：relaxng.rng (要素数 44)
  - 編集操作数 (差分長)：22
- パターン 2a：
  - 使用スキーマ：VoiceXML10_full (要素数 69)
  - 編集操作数 (差分長)：21
- パターン 2b：
  - 使用スキーマ：VoiceXML10_full (要素数 69)
  - 編集操作数 (差分長)：35

表 2 に結果を示す．本アルゴリズムでは，上記いずれの場合でも適切に差分抽出が可能であった．一方，X-Diff で RELAX NG の差分を抽出した場合，要素によっては要素定義全体を削除/追加という差分抽出が行われ差分がかなり冗長となる場合がある．差分の内容を調べた結果，２つの RELAX NG スキーマの差分を X-Diff で抽出しようとすると，以下の理由のために適切な差分抽出が困難であることが分かった．

- 要素定義の出現順序の変更に対応できない
- 内容モデルの差分抽出において，本来交換すべきでない，違う種類のノード同士 (正規表現のオペレータと非終端記号など) の交換がおきてしまう

また，本アルゴリズムと Diffmk に関して，得られる差分長にあまり大きな差は見受けられないが，Diffmk は削除された要素を検出しない (差分に現れない) ため，削除された要素は目視で確認するしかなく，実際に正しい差分を得

表 2 実験結果

パターン	適用差分長	本アルゴリズム	X-Diff	Diffmk
1a	13	12	24	12
1b	22	22	187	26
2a	21	21	176	25
2b	35	34	278	35

るのは極めて手間のかかる作業であった[*1]．また，Diffmk において，削除がほとんど含まない編集操作列が適用された場合，適切な差分抽出は難しいと考えられる．これについては，別途評価実験を行う予定である．

7. むすび

本稿ではまず，正規木文法の差分抽出問題の計算複雑さについて考察し，差分抽出が効率よく行えるための十分条件を求めた．更に，その十分条件の下で，正規木文法の差分抽出を行う効率の良いアルゴリズムを構成した．評価実験の結果，本アルゴリズムは更新されたスキーマの変更内容を把握するには有用であると考えられる．

今後の課題として，DTD や XML Schema など，RELAX NG 以外の形式で記述されたスキーマにもそのまま適用できるよう，実装を改良することが挙げられる．また，本稿では使用していないが，XML データの差分を抽出するツールとして XyDiff[2] がある．このツールに関しても，評価実験を行う必要がある．

謝辞

本研究の一部は，科研費・基盤研究 (C)(23500110) の補助を受けている．

参考文献

[1] P. Bille, “A Survey on Tree Edit Distance and Related Problems”, Theoretical Computer Science, vol.337, pp.217–239, 2005.
[2] G. Cobena, S. Abiteboul, A. Marian, “Detecting Changes in XML Documents”, Proc. ICDE, pp.41–52, 2002.
[3] E. Demaine, S. Mozes, B. Rossman, and O. Weimann, ”An Optimal Decomposition Algorithm for Tree Edit Distance”, Proc. the 34th International Colloquium on Automata, Languages and Programming, pp.146-157, 2007.
[4] E. Leonardi, T. T. Hoai, S. S Bhowmick, and S. Madria, “DTD-Diff: A Change Detection Algorithm for DTDs”, Data & Knowledge Engineering, vol.61, no.2, pp.384–402, 2007.
[5] M. Murata, D. Lee, M. Mani, and K. Kawaguchi, “Taxonomy of XML Schema Languages using Formal Language Theory”, ACM Transactions on Internet Technology, vol.5, no.4, pp.660–704, Nov. 2005.
[6] D. Sankoff and J. Kruskal (eds.), “Time Warps, String Edits, and Macromolecules: The Theory and Practice of Sequence Comparison”, Addison-Wesley, 1983.
[7] H. Seidl, “Deciding Equivalence of Finite Tree Automata”, SIAM J. Comput. vol.19, issue 3, pp.424–437, 1990.
[8] N. Suzuki, “An Algorithm for Inferring K Optimum Transformations of XML Document from Update Script to DTD”, IEICE Transactions on Information and Systems, vol.E93-D, no.8, pp.2198–2212, 2010.
[9] RELAX NG 1.0, http://relaxng.org/relaxng.rng
[10] VoiceXML in RELAX NG, http://www.kohsuke.org/relaxng/voicexml.html
[11] N. Walsh, DiffMk, http://diffmk.sourceforge.net/
[12] Y. Wang, D. J. DeWitt, J. Cai: ”X-Diff: An Effective Change Detection Algorithm for XML Documents”, Proc. ICDE, pp.519-530, 2003.

[*1] 表 2 の Diffmk に関する値について，削除された要素は出力結果に現れないので，削除された要素数に関してはスキーマを手作業で調べて算出している．