4N-1

正規木文法のための差分抽出アルゴリズム

堀江 和磨† 鈴木 伸崇‡

筑波大学 情報学群 知識情報・図書館学類† 筑波大学大学院 図書館情報メディア研究科‡

1 はじめに

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，時間の経過と共に格納すべきデータの構造や種類が変化し，それに応じてスキーマ定義が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する必要がある．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．本稿では，スキーマ定義言語として正規木文法を対象とし，正規木文法(regular tree grammar)のための差分抽出アルゴリズムを提案する．

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．正規木文法の差分抽出は，「更新前の正規木文法」を「更新後の正規木文法」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る．

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が，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[1] 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．

2 諸定義

文献[2]等に従って正規木文法を定義する．正規木文法は 4 次組 $G = (N, T, S, P)$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($S \subseteq N$)，P は生成規則の集合である．生成規則は $X \rightarrow a(r)$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．

"A difference extraction algorithm for regular tree grammar"

†Kazuma Horie

†College of Knowledge and Library Sciences, School of Informatics, University of Tsukuba

‡Nobutaka Suzuki

‡Graduate School of Library, Information and Media Studies, University of Tsukuba

正規木文法 $G = (N, T, S, P)$の例を以下に示す．

N = {*R, T, M, E, Pcdata*} //非終端記号(型)
T = {staffs, staff, name, email, pcdata} //終端記号
S = {*R*} //開始記号
P = {$R \rightarrow$ staffs(*T**), $T \rightarrow$ staff(*M,E*), //生成規則
M → name(*Pcdata*), *E* → email(*Pcdata*),
Pcdata → pcdata(ε)}

Gから生成される木の集合(言語)を $L(G)$と表す．

正規木文法における編集操作列とは以下の編集操作の系列である．

- 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
- 生成規則の右辺における，終端記号の変更および内容モデルの変更．生成規則の右辺を順序木として表し(図 1)，木編集操作列として表す．

各編集操作には非負のコストが付与される．

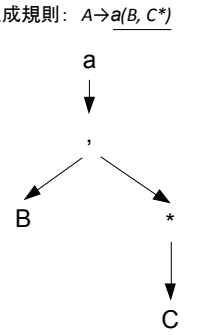

図 1 生成規則の右辺を木で表した例

3 計算複雑さ

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 G,G'に対して，G を G'と等価な文法 G''($L(G'') = L(G')$)に更新するためのコスト最小の編集操作列を発見することを考える．

定理 1 上記の差分抽出問題は EXPTIME 困難である．

証明：文献[3]から正規木文法の等価性判定問題は EXPTIME 完全であり，この問題からの帰着により容易に示せる． □

このように，言語の等価性を考慮すると差分抽出を効率よく行うのは困難である．このため，より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．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 2 構文的な等価性を用いた場合，正規木文法の差分抽出問題は NP 困難である．

証明：3-partition 問題からの帰着により示せる．詳細は紙面の都合上省略する． □

以下では，正規木文法の構文的な等価性を用いるものとする．

4 差分抽出アルゴリズム

定理 2 から，正規木文法の差分抽出を効率よく行うのは困難である．これを効率よく行うための十分条件について考える．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．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ので，この仮定を置いても一般性を失わない．

ここで，編集操作「生成規則の左辺の非終端記号の変更」は以下の2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．

上記2条件の下で，正規木文法の差分抽出問題は多項式時間で解くことができる．$G = (N, T, S, P)$と $G' = (N', T', S', P')$を正規木文法とする．本アルゴリズムは，以下の4つから成り立つ．

1. NとN'における差分抽出
2. TとT'における差分抽出
3. SとS'における差分抽出
4. PとP'における差分抽出

ここで，1.～3.は集合の比較から容易に得られる．4.の差分抽出は，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．上記2条件から，生成規則の対応は一意に定まる．対応する生成規則の組ごとに，「生成規則の右辺」を順序木とみなして木編集操作列を求めるアルゴリズムを用いて差分を求める．

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

5 評価実験

本アルゴリズムを Ruby で実装し(入力スキーマは RELAX NG 形式に対応)，評価実験を行った．実験では，2個の RELAX NG スキーマ(約 1.8kB)を作成し，以下の2つの方法で差分を抽出した．

A) 本アルゴリズムを用いて，2つのスキーマの差分を抽出する．出力は HTML 形式で2つのスキーマが並べて表示され，変更の生じた箇所は色を変えて示される(赤が削除，青が追加，緑が終端記号等の変更)．

B) 4名の被験者に，目視による比較や diff を用いて2つのスキーマの差分を抽出してもらう．2つのスキーマが印刷された紙を渡し，変更箇所に印をつける．

A)は約 30 秒で表示が得られるのに対し，B)の平均所要時間は 7 分 4 秒であった．この結果から，本アルゴリズムはスキーマの変更箇所を把握するのに有効であると考えられる．

6 むすび

本稿では，正規木文法の差分抽出問題の計算複雑さについて考察し，差分抽出が効率よく行えるための十分条件を求め，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．今後の課題として，編集操作に設けた制限の緩和が挙げられる．

謝辞

本研究の一部は筑波大学図書館情報メディア研究科プロジェクト研究による助成を受けたものである．

参考文献

[1] 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.

[2] M. Murata, D. Lee, M. Mani, and K. Kawaguchi, “Taxonomy of XML Schema Languages using Formal Language Theory”, ACM TOIT, vol.5, no.4, pp.660–704, Nov. 2005.

[3] H. Seidl, "Deciding Equivalence of Finite Tree Automata", SIAM J. Comput., vol.19, issue 3, pp.424-437, 1990.