社団法人 情報処理学会 研究報告
IPSJ SIG Technical Reports

2007－DBS－143 (18)
2007／7／2

木の埋め込み関係に基づくXMLスキーマ進化に応じた文書変換器の生成法

吉田 昌起† 橋本 健二† 石原 靖哲† 藤原 融†

† 大阪大学大学院情報科学研究科 〒565-0871 大阪府吹田市山田丘1-5
E-mail: †{masaki-y,k-hasimt,ishihara,fujiwara}@ist.osaka-u.ac.jp

あらまし XMLスキーマの進化が木の埋め込み関係に基づくスキーマ更新操作列により表されているときに，既存の文書を新しいスキーマに従うように変換するための文書変換器の生成法を示す．まず文書変換器がもつべき能力を検討し，それら必要な能力をもつ非決定性木変換器のクラスを定義する．その上でスキーマ更新操作列から木変換器を生成するアルゴリズムを示す．さらに，この枠組みにおいて文書の変換が一般に非決定性になる要因を明らかにする．
キーワード XML文書変換，スキーマ進化，木変換器

Generating XML document transformers according to XML schema evolution based on the tree-embedding relation

Masaki YOSHIDA†, Kenji HASHIMOTO†, Yasunori ISHIHARA†, and Toru FUJIWARA†

† Graduate School of Information Science and Technology, Osaka University
Yamadaoka 1-5, Suita-shi, Osaka, 565-0871 Japan
E-mail: †{masaki-y,k-hasimt,ishihara,fujiwara}@ist.osaka-u.ac.jp

Abstract Given an update operation sequence for schema evolution based on the tree-embedding relation, the method of generating XML document transformers that transform each document valid against the old schema into a document valid against the new schema is proposed. First of all, we discuss how powerful the document transducers should be, and define a class of nondeterministic tree transducers that meet the requirements for the transformation power. Then, we present an algorithm of generating an appropriate tree transducer. In addition, we address the cause of that the transducer is nondeterministic in our settings.
Key words XML document transformers, schema evolution, tree transducers.

1. まえがき

近年，XML(Extensible Markup Language) は急速に広まり，様々な種類の情報を表現するために用いられている．このXML文書の構造を指定するために，さまざまなスキーマが定義されている．XMLはスキーマレスでも利用できるように設計されているが，スキーマはXML文書の統合や変換，問い合わせの最適化などに有用である．しかし，システムの新たな機能の導入やユーザからの要求の変化などにより，システムが扱うべき情報の構造や種類に変化が生じた場合には，それに応じてスキーマも進化させることが必要となる．ここで，進化先スキーマは，元のスキーマに従った既存のXML文書の情報を何らかの意味で失うことなく表現できる能力をもつことが要求される．さもなければ，もともと利用できていた情報が，スキーマ進化後には利用できなくなってしまうからである．

文献[1],[2]では，元のスキーマのどの部分をどのように変更して進化先スキーマが得られるかを表す道具だてとしてスキーマ更新操作群を提案している．つまり，スキーマ進化を，元のスキーマから新しいスキーマを得るための更新操作群の系列で表す．文献[1]では，XMLスキーマ間の表現能力保存の関係を木の埋め込み関係に基づいて定義し，ある制約のもとでこの更新操作群が以下の性質を満たすことを示している．

- 更新操作群の適用によって得られたスキーマが元のスキーマの表現能力を保存している (健全性).
- 元のスキーマの表現能力を保存しているどんなスキーマも，元のスキーマに基本操作群を適用して得られる (完全性)

本稿では，スキーマ進化が文献[1]で提案されてるスキーマ更新操作列により表されているときに，既存の文書を新しいスキーマに従うように変換するための文書変換器の生成法を提案する．まず，この文書変換器がもつべき能力を検討し，それら必要な能力をもつ非決定性木変換器を定義する．この木変換器は，入力木をトップダウンに走査しながら，非決定的に子のラ

ベルの系列に対する正規表現のマッチングに応じて状態を遷移させる能力がある．次に，既存の文書を新しいスキーマのどの文書に変換するべきかを定義する．その際に，文書の変換が一般に非決定性となる要因を明らかにする．その上で，この定義に従う文書変換を行うような木変換器の生成法を示す．この生成法は，文献[1]で提案されたスキーマ更新操作群の基本操作である等価変換の能力を制限し，その上でスキーマの更新に合わせた文書変換器を生成する手法である．

以下，2節では各種定義や本稿で用いる非決定性木変換器の定義を与える．3節ではスキーマ更新操作列が与えられた時に，既存の文書を新しいスキーマのどの文書に変換するべきかを定義する．4節では3節で示した定義に従った文書変換を行うような木変換器の生成法を示す．5節では4節の方法で生成される木変換器が3節の文書変換の定義に関する健全性と完全性を満たすことを証明する．最後に，6節でまとめと今後の課題を述べる．

2. 定　　義

2.1 木

本稿では，XML文書のモデルとしてラベル付き順序木を扱う．以降，ラベル付き順序木を単に木と呼ぶ．記号集合Σに対して，各頂点がΣ中の記号でラベル付けされた木をΣ上の木といい，Σ上の木の集合をT_Σと書く．また，記号集合N,Σに対して，N中の記号は葉のラベルとしてのみ現れるような，$N\cup\Sigma$上の木の集合を$T_\Sigma(N)$と書く．

2.2 正規木文法

本稿では，XML文書のスキーマ記述言語のモデルとして正規木文法を用いる．以下，文献[3]等にしたがって正規木文法に関する定義を行う．

[定義1] 正規木文法は以下のような四つ組$G=(N,\Sigma,S,P)$である:

- Nは非終端記号の有限集合，
- Σは終端記号の有限集合，
- $S\subseteq N$は開始記号の集合，
- Pは生成規則の有限集合．

ただし，$X\in N$，$a\in\Sigma$であり，rはN上の正規表現とすると，生成規則は$X\to a(r)$の形式である．X，$a(r)$をそれぞれこの規則の左辺，右辺と呼ぶ．rをこの規則の内容モデルと呼ぶ．　□

次に，正規木文法$G=(N,\Sigma,S,P)$に対して，$T_\Sigma(N)$上の導出関係を定義する．以下，正規表現rによって表現される文字列言語を$L(r)$と書く．$t\in T_\Sigma(N)$のある葉uのラベルが$X\in N$であるとする．$t'\in T_\Sigma(N)$を，tのuを$a(\gamma)$に置き換えた木とする．ここでγはある生成規則$p=(X\to a(r))\in P$に対して$\gamma\in L(r)$を満たす任意の非終端記号系列である．このとき，Gにおいてtからt'が導出されるといい，$t\Rightarrow t'$と書く．そして，Gにおいて，ある$A\in S$から生成規則の0回以上の適用により導出される，すべての$t'\in T_\Sigma$からなる集合を，Gによって生成される木言語と呼び，$TL(G)$と書く．正規木文法によって生成される木言語を正規木言語と呼ぶ．開始記号からある木$t\in T_\Sigma$が導出されるとき，tの各ノードについて，そのノードが生成されるのに用いられた規則の左辺に現れる非終端記号と右辺に現れる終端記号のペアをラベル付けしたものをtの解釈と呼び，tの解釈の集合を$I_G(t)$と書く．

2.3 非決定性木変換器

この木変換器は，入力木をトップダウンに走査しながら，非決定的に子のラベルの系列に対する正規表現のマッチングに応じて状態を遷移させ，葉に新たなノードを追加したり，ある部分木の系列とその親ノードの間に新たなノードを追加したりする能力がある．

[定義2] 非決定性木変換器は次の五つ組$T=(Q,\Sigma,\Delta,S_T,P_T)$である:

- Qは状態の有限集合，
- Σは入力木及び出力木に用いられる記号の集合，
- Δは出力木のみに用いられる記号の集合，
- $S_T\subseteq Q$は初期状態の集合，
- P_Tは$(q,a(r))\to a(r')$の形をした変換規則の有限集合．

ただし，$q\in Q$，$a\in\Sigma\cup\Delta$である．また，r_ΣをΣ上の正規表現とすると，rはr_Σの任意の部分式αを$[\![\alpha]\!]^i$に置き換え，それ以外の部分に含まれる各記号$s\in\Sigma$を$[s]^i$に置き換えた形をしている．ここで$\mathbb{N}$を自然数の集合とすると，$i\in\mathbb{N}$であり，左から数えてk番目の$[s]^i$または$[\![\alpha]\!]^i$の肩付き文字をkとする．また，r_Δを$\Sigma\cup\Delta$上の正規表現とすると，r'はr_Δの全ての記号$u\in\Delta$を$[\![u]\!]^i_p$または$[u]_p$に置き換え，それ以外の各記号sを$[s]^i_p$に置き換えた形をしている．ただし$p\in Q$であり，rに含まれる各$[s]^j$についてr'は$[s]^j_p$を含み，またrに含まれる各$[\![\alpha]\!]^j$についてr'は肩付き文字が同じ$[u]^j_p$を含む．rとr'に含まれる記号の肩付き文字に現れる自然数の系列は等しい．　□

以下，木変換器の動作について述べる．木tに状態qを割り当てた時の$T=(Q,\Sigma,\Delta,S_T,P_T)$で定義される変換を$T^q(t)$と表し，これを次のように再帰的に定義する．

$t=a(t_1\cdots t_n)$であり（aの子のラベルの系列を$a_1a_2\cdots a_n$とする），変換規則$(q,a(r))\to a(r')$が存在し，ここでrに含まれる$[s]^i$をsに置き換え，$[\![\alpha]\!]^i$をαに置き換えた（αはΣ上の正規表現）正規表現をr_aとすると，$a_1a_2\cdots a_n\in L(r_a)$である時，この変換規則を$t$に適用する．

まず，$\beta\in L(r)$を満たす系列βを非決定的に選択する．ただしβに含まれる記号の$[\,]^i$と$[\![\,]\!]^i$を取り除いた系列は$a_1a_2\cdots a_n$と等しい．この時$[s]^i$とa_jの対応(i,j)を取る．次に，以下の条件を満たす$\beta'\in L(r')$を生成する．

- β'に含まれる$[s]^i_q$を$[s]^i$に置換し，$[s]_q$を取り除き，$[\![u]\!]^i_q$をそれぞれβに含まれる肩付き文字が等しい$[\![\gamma]\!]^i$に置換した系列はβと等しい．

$T^q(t)$は，$a(\beta')$に以下の置換を行って得られる木を含む．

- β'に含まれる記号$[s]^i_q$について，先ほど取った対応(i,j)を見て$T^q(t_j)$に置換する．
- β'に含まれる$[s]_q$を$T^q(s)$に置換する．
- β'に含まれる$[\![u]\!]^i_q$について，それぞれβは$[\![\gamma]\!]^i_q$を含む．$[\![u]\!]^i_q$を$T^q(u(\gamma))$に置換する．

[例 1] $T=(Q,\Sigma,\Delta,\{p\},P)$，ここで $Q=\{p,q\}$，$\Sigma=\{a,b\}$，$\Delta=\{a,b,c\}$，P は以下の規則を含むとする．

$(p, a([[b*]]^1[a]^2*)) \rightarrow a([[c]]^1_q[a]^2_p*)$
$(p, a(\epsilon)) \rightarrow a([c]_q)$
$(q, c([b]^1*)) \rightarrow c([b]^1_p*)$
$(q, c(\epsilon)) \rightarrow c(\epsilon)$
$(p, b(\epsilon)) \rightarrow b(\epsilon)$

T は図 1 のような変換を行う変換器である．

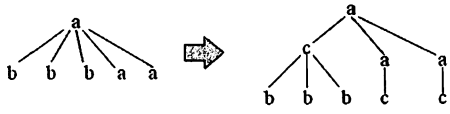

図 1　木の変換の例

2.4　スキーマ更新操作群

本稿では，スキーマ進化を表すモデルとして，スキーマ更新操作群を用いる．以下，文献 [1] に従ってスキーマ更新操作群を以下のように定義する．

[定義 3] $G_k=(N_k,\Sigma\cup\Delta,S_k,P_k)$ に対してくくり出し，挿入，拡張，等価変換の四つの基本操作を適用すると，それぞれ以下のような正規木文法 $G_{k+1}=(N_{k+1},\Sigma\cup\Delta,S_{k+1},P_{k+1})$ が得られる．

(くくり出し)：$p=(A\rightarrow a(r))\in P_k$ とする．

- $N_{k+1}=N_k\cup\{X\}$，ただし $X\notin N_k$，
- $S_{k+1}=S_k$，
- $P_{k+1}=(P_k-\{p\})\cup\{A\rightarrow a(r_B), X\rightarrow x(r_x)\}$．ただし，$x\in\Delta$ であり，r_B は N_{k+1} 上の正規表現である．また r_x は $\epsilon\notin L(r_x)$ を満たす r の部分式であり，r_B 中のすべての X を r_x に置換すると r となる．

(挿入)：$p=(A\rightarrow a(r))\in P_1$ とする．

- $N_{k+1}=N_k\cup\{X\}$，ただし $X\notin N_k$，
- $S_{k+1}=S_k$，
- $P_{k+1}=(P_k-\{p\})\cup\{A\rightarrow a(r_S), X\rightarrow x(\epsilon)\}$．ただし，$x\in\Delta$ である．また，r_S は r の部分式 $r'\cdot r''$ (r'，r'' は N_1 上の正規表現) を $r'\cdot X\cdot r''$ で置換して得られる正規表現である．

(拡張)：$x\in\Sigma\cup\Delta$ とする．

- $N_{k+1}=N_k\cup\{X\}$，ただし $X\in N_k$ でもよい，
- $S_{k+1}=S_k$ あるいは $S_{k+1}=S_k\cup\{X\}$ のいずれでもよい，
- $P_{k+1}=P_k\cup\{X\rightarrow x(r_E)\}$．ただし r_E は N_{k+1} 上の正規表現である．

(等価変換)：G_{k+1} は $TL(G_{k+1})=TL(G_k)$ を満たす．　□

3. 文書変換

文書 t_2 が文書 t_1 に要素をいくつか追加して得られるとき，すなわち，t_1 が t_2 に埋め込み可能であるとき，t_2 は t_1 の情報を保存すると考える．この節では，スキーマ更新操作列が与えられた時に，既存の文書に対して，新しいスキーマに従い情報を保存した変換先を定義する．

[定義 4] スキーマ更新操作を k 回適用してスキーマを G_1 から G_k に更新した際に，$TL(G_1)$ から $TL(G_k)$ への非決定性変換 T_{k+1} を次のように再帰的に定義する．

$$T_{k+1}=T'_{k+1}\circ T_k$$

ここで，T'_{k+1} は k 番目に適用されるスキーマ更新操作の種類に応じて次のように定義する．

(くくり出し，挿入)：任意の $t\in TL(G_k)$，$t'\in TL(G_{k+1})$ について，ある $s\in I_{G_k}(t)$，$s'\in I_{G_{k+1}}(t')$ が存在して，s' は s を埋め込み可能ならば $t'\in T'_{k+1}(t)$ である．

(拡張)：拡張で新たに追加された規則を適用して導出された木は変換先としない．よって任意の $t\in TL(G_k)$ について，$T'_{k+1}(t)=\{t\}$ である．

(等価変換)：任意の $t\in TL(G_k)$ について，$T'_{k+1}(t)=\{t\}$ である．　□

文書の変換が一般に非決定性になるのは，スキーマ更新操作は生成規則に対して更新を行うが，文書の各ノードと生成規則の内容モデルに含まれる非終端記号との対応付けが一意に決まらないからである．対応付けが一意に決まらない要因は以下の二つである．

- 文書 $t\in TL(G_1)$ を導出するための導出過程が複数ある．
- 非終端記号系列と内容モデルの正規表現のマッチングの仕方が複数ある．

本稿では，文書の各ノードと生成規則の内容モデルに含まれる非終端記号との対応が非決定性となるように，非決定性木変換器を用いることにした．

4. スキーマ更新操作群に従う変換を行う木変換器の生成法

本節では，スキーマ更新操作群に従う木変換器の生成法を示す．4.1 節で提案する生成法で用いるスキーマ更新操作群に対する制約を述べる．次に 4.2 節で設計方針を述べ，4.3 節，4.4 節で設計方針に従った生成法を述べる．

4.1　スキーマ更新操作群の制約

2 節で定義したスキーマ更新操作群の等価変換の能力を以下の定義のように制限する．

[定義 5] $G_k=(N_k,\Sigma\cup\Delta,S_k,P_k)$ に対して，8 個に分けたいずれかの等価変換を適用すると，それぞれ以下のような正規木文法 $G_{k+1}=(N_{k+1},\Sigma\cup\Delta,S_{k+1},P_{k+1})$ が得られる．

(等価変換)：$A, A_1, A_2\in N_k$，r_1, r_2 を N_k 上の正規表現とする．

(1)　$p_1=(A\rightarrow a(r_1))\in P_k$ とする．

- $N_{k+1}=N_k$，
- $S_{k+1}=S_k$，
- $P_{k+1}=(P_k-\{p_1\})\cup\{A\rightarrow a(r_2)\}$．

ただし r_2 は，r_1 と等価な正規表現である．

(2)　$p_1=(A\rightarrow a(r_1|r_2))\in P_k$ とする．

- $N_{k+1}=N_k$，
- $S_{k+1}=S_k$，
- $P_{k+1}=(P_k-\{p_1\})\cup\{A\rightarrow a(r_1), A\rightarrow a(r_2)\}$．

(3)　$p_1=(A\rightarrow a(r_1)), p_2=(A\rightarrow a(r_2))\in P_k$ とする．

p_1, p_2 はどちらも拡張によって追加された規則ではなく，かつ拡張によって追加された規則にくくり出し・挿入・等価変換を行うことによって追加された規則でもない場合に，

- $N_{k+1} = N_k$,
- $S_{k+1} = S_k$,
- $P_{k+1} = (P_k - \{p_1, p_2\}) \cup \{A \to a(r_1|r_2)\}$.

(4) $p_1 = (A \to a(r_1|r_2)) \in P_k$ とする．

- $N_{k+1} = N_k \cup \{A'\}$, ただし $A' \notin N_k$,
- $S_{k+1} = \begin{cases} S_k & \text{if } A \notin S_k, \\ S_k \cup \{A'\} & \text{otherwise,} \end{cases}$
- $P_{k+1} = \{p' \mid p \in (P_k - \{p_1\}), p'$は$p$の内容モデルに表れる$A$を$(A|A')$に置換したもの$\} \cup \{A \to a(r_1), A' \to a(r_2)\}$.

(5) P_k に属し，左辺が A_2 である規則の集合を P_{A_2} とする．P_k の全ての規則の内容モデルに現れる A_1 または A_2 は $(A_1|A_2)$ の形であり，$A_2 \notin S_k$ または $A_1, A_2 \in S_k$ である場合に，

- $N_{k+1} = N_k - \{A_2\}$,
- $S_{k+1} = S_k - \{A_2\}$,
- $P_{k+1} = \{p' \mid p \in (P_k - P_{A_2}), p'$は$p$の内容モデルに表れる$(A_1|A_2)$を$A_1$に置換したもの$\} \cup \{p' \mid p \in P_{A_2}, p'$は$p$の左辺を$A_1$に置換したもの$\}$.

(6) $p_1 = (A \to a(r_1)) \in P_1$ とする．

- $N_{k+1} = N_k \cup \{A'\}$, ただし $A' \notin N_k$,
- $S_{k+1} = S_k$,
- $P_{k+1} = P_k \cup \{A' \to a(r_1)\}$.

(7) P_k に属し，左辺が A_1 である規則の集合を P_{A_1} とする．A_1 は，P_k のどの規則の内容モデルにも現れず，$A_1 \notin S_k$ である場合に，

- $N_{k+1} = N_k - \{A_1\}$,
- $S_{k+1} = S_k$,
- $P_{k+1} = P_k - P_{A_1}$.

(8) A_1 を左辺に持つ生成規則の集合を P_{A_1} とし，A_2 を左辺に持つ生成規則の集合を P_{A_2} とし，P_{A_1} に属する全ての規則について，左辺を A_2 に置換した規則の集合を P'_{A_1} とすると，$P'_{A_1} = P_{A_2}$ である場合に，

- $N_{k+1} = N_k$,
- $S_{k+1} = S_k$,
- $P_{k+1} = P'_k$. ただし P'_k は，P_k に属する全ての規則について，内容モデルに現れる A_1 を $(A_1|A_2)$ に置換したものである． □

4.2 設計方針

提案手法は，3節で示したような木変換器の合成は行わず，各スキーマ更新操作に対して木変換器を更新することで行う．提案手法で生成した木変換器は，生成規則と変換規則は一対一に対応し，状態数は，高々スキーマの非終端記号の数である．

$G_1 = (N_1, \Sigma, S_1, P_1)$ に対してスキーマ更新操作を $n-1$ 回 $(n \geq 1)$ 適用することにより $G_n = (N_n, \Sigma \cup \Delta, S_n, P_n)$ が得られるとすると，生成する木変換器 T_n は $t \in TL(G_1)$ を入力として，t の情報を保存している $t' \in TL(G_n)$ を出力するような木変換器である．以降，スキーマ更新操作を $k-1$ 回適用した後の木変換器を $T_k = (Q_k, \Sigma, \Delta, S_{Tk}, P_{Tk})$ と表し，入力木 t に対する出力木の集合を $T_k(t)$ と表す．

3節の文書変換の定義に従い，拡張で新たに追加された規則を適用して導出された木を変換先としない．よって，拡張で追加された規則，またはその規則に対して挿入・くくり出し・等価変換を行って得られた規則の集合を区別する必要があり，G_k 中のそのような規則の集合を Γ_k とする．また，各非終端記号 B について，T_k の変換規則の左辺の B に対応する部分を $L_k(B)$，右辺の B に対応する部分を $R_k(B)$ と表し，それらの定義域と値域を以下のように定義する．

- $L_k : N_k \to \Omega_L$, ただし Ω_L は $R_\Sigma = \{[s]^i \mid s \in \Sigma, i \in \mathbb{N}\}$, $\hat{R}_\Sigma = \{[\![\alpha]\!]^i \mid \alpha \in \Sigma$上の正規表現$, i \in \mathbb{N}\}$ に対して $\Omega_L = \{(u_1|u_2|\cdots|u_m) \mid u_j \in R_\Sigma \cup \hat{R}_\Sigma \cup \{\epsilon\}, 1 \leqq j \leqq m, m \geq 1\}$ である．
- $R_k : N_k \to \Omega_R$, ただし Ω_R は $R_\Delta = \{[s]^i_{qX} \mid s \in \Sigma \cup \Delta, i \in \mathbb{N}, X \in N_k\}$, $R_S = \{[s]_{qX} \mid s \in \Delta, X \in N\}$, $\hat{R}_\Delta = \{[\![s]\!]^i_{qX} \mid s \in \Delta, i \in \mathbb{N}, X \in N_k\}$ に対して $\Omega_R = \{(u_1|u_2|\cdots|u_m) \mid u_j \in R_\Delta \cup R_S \cup \hat{R}_\Delta, 1 \leqq j \leqq m, m \geq 1\}$ である．

また，G_k の各生成規則の内容モデルと，T_k の変換規則の左辺との対応を表すために f_{L_k}，右辺との対応を表すために f_{R_k} を定義する．

- $f_{L_k} : R_N \to \Psi_L$, ただし R_N は N 上の正規表現であり，Ψ_L は Ω_L 上の正規表現である．$f_{L_k}(r_N) = \psi_L$, ただし $\psi_L \in \Psi_L$ は，$r_N \in R_N$ に含まれる非終端記号 B を $L_k(B)$ に置換し，肩付き文字がついている記号について，左から i 番目の記号の肩付き文字を i に変換したものである．
- $f_{R_k} : R_N \to \Psi_R$, ただし R_N は N 上の正規表現であり，Ψ_R は Ω_R 上の正規表現である．$f_{R_k}(r_N) = \psi_R$, ただし $\psi_R \in \Psi_R$ は，$r_N \in R_N$ に含まれる非終端記号 B を $R_k(B)$ に置換し，肩付き文字がついている記号について，左から i 番目の記号の肩付き文字を i に変換したものである．

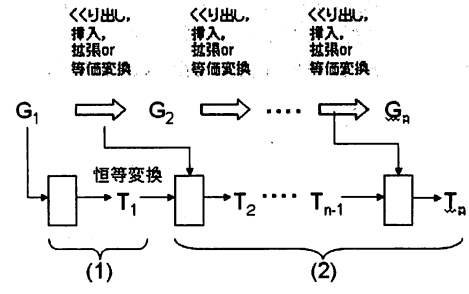

図2 スキーマと木変換器との対応

木変換器 T_n は，k 回目に適用されるスキーマ更新操作に応じて，Γ_{k+1}, $L_{k+1}(B)$, $R_{k+1}(B)$ を生成しながら，以下のようなステップで生成する．

(1) スキーマ G_1 から，$t \in TL(G_1)$ を入力すると t を出力するような恒等変換を行う木変換器 T_1 を生成し，Γ_1 と各非終端記号について L_1, R_1 を生成する．

(2) 各 $k=1,...,n-1$ について順に，木変換器 T_k，L_k，R_k，Γ_k，スキーマ更新操作から，木変換器 T_{k+1}，L_{k+1}，R_{k+1}，Γ_{k+1} を生成する．

4.3 恒等変換を行う木変換器

本節では，恒等変換を行う木変換器の生成法を示す．スキーマ $G_1=(N_1,\Sigma,S_1,P_1)$ から以下のように L_1，R_1，Γ_1，T_1 を順に生成する．

(L_1,R_1 の生成)：各非終端記号 $B\in N_1$ について，B を左辺にもつ生成規則の右辺に現れる終端記号を $s_1,...,s_m$ とすると，$L_1(B)=([s_1]^1|[s_2]^2|\cdots|[s_m]^m)$，$R_1(B)=([s_1]^1_{q_B}|[s_2]^2_{q_B}|\cdots|[s_m]^m_{q_B})$ とする．

(Γ_1 の生成)：$\Gamma_1=\phi$.

(木変換器の生成)：恒等変換を行う以下の木変換器 $T_1=(Q_1,\Sigma,\Sigma,S_{T1},P_{T1})$ を生成する．

- $Q_1=\{q_X\mid X\in N_1\}$,
- $S_{T1}=\{q_X\mid X\in S_1\}$,
- $P_{T1}=\{(q_A,a(f_{L_1}(r)))\rightarrow a(f_{R_1}(r))\mid A\rightarrow a(r)\in(P_1)\}$.

[例 2] $N=\{A,B,C\}$，$\Sigma=\{a,b_1,b_2,c\}$，$S=\{A\}$，$P=\{A\rightarrow a(B^*|C^*),B\rightarrow b_1(CC),B\rightarrow b_2(C),C\rightarrow c(\epsilon)\}$ として，スキーマ $G=(N,\Sigma,S,P)$ の恒等変換を行う木変換器 $T=(Q,\Sigma,\Delta,\{q_A\},P_T)$ は次のとおりである．

- $Q=\{q_A,q_B,q_C\}$,
- $\Sigma=\{a,b_1,b_2,c\}$,
- $\Delta=\phi$,
- P_T は以下の規則を含む．

$$
\begin{array}{lcl}
(q_A,a(([b_1]^1|[b_2]^2)^*|[c]^{3*}) & \rightarrow & a(([b_1]^1_{q_B}|[b_2]^2_{q_B})^*|[c]^{3*}_{q_C})\\
(q_B,b_1([c]^1[c]^2)) & \rightarrow & b_1([c]^1_{q_C}[c]^2_{q_C})\\
(q_B,b_2([c]^1)) & \rightarrow & b_2([c]^1_{q_C})\\
(q_C,c(\epsilon)) & \rightarrow & c(\epsilon)
\end{array}
$$

4.4 木変換器の更新

$G_k=(N_k,\Sigma\cup\Delta,S_k,P_k)$ に対して，定義 4 で示したくくり出し，挿入，拡張，または定義 5 で示した等価変換の操作を適用し $G_{k+1}=(N_{k+1},\Sigma\cup\Delta,S_{k+1},P_{k+1})$ が得られたとする．それに従って木変換器 T_k を T_{k+1} に更新する．

木変換器 $T_{k+1}=(Q_{k+1},\Sigma,\Delta,S_{k+1},P_{k+1})$ は，等価変換の (4) の操作以外については以下のように生成する．

- $Q_{k+1}=\{q_X\mid X\in N_{k+1}\}$,
- $S_{Tk+1}=S_{Tk}\cap Q_{k+1}$,
- $P_{Tk+1}=\{(q_A,a(f_{L_{k+1}}(r)))\rightarrow a(f_{R_{k+1}}(r))\mid A\rightarrow a(r)\in(P_{k+1}-\Gamma_{k+1})\}$.

以下，各スキーマ更新操作に対する木変換器の更新を示す．

4.4.1 くくり出しによる変換器の更新

(L_{k+1},R_{k+1} の生成)：$B\in N_{k+1}$ それぞれについて，

$$
L_{k+1}(B)=\begin{cases}[\![R_x]\!]^1_{qX} & \text{if } B=X,\\ L_k(B) & \text{otherwise,}\end{cases}
$$

$$
R_{k+1}(B)=\begin{cases}[\![x]\!]^1_{qX} & \text{if } B=X,\\ R_k(B) & \text{otherwise.}\end{cases}
$$

ここで R_x は，r_x に含まれる各非終端記号 B について，B を左辺にもつ生成規則の右辺に現れる終端記号を $s_1,...,s_m$ とすると，B を $(s_1|s_2|\cdots|s_m)$ に置換して得られる系列である．

(Γ_{k+1} の生成)：もし $p\in\Gamma_k$ であるなら，$\Gamma_{k+1}=(\Gamma_k-\{p_1\})\cup\{A\rightarrow a(r_B),X\rightarrow x(r_x)\}$ とする．もし $p\notin\Gamma_k$ であるなら，$\Gamma_{k+1}=\Gamma_k$.

4.4.2 挿入による変換器の更新

(L_{k+1},R_{k+1} の生成)：$B\in N_{k+1}$ それぞれについて，

$$
L_{k+1}(B)=\begin{cases}\epsilon & \text{if } B=X,\\ L_k(B) & \text{otherwise.}\end{cases}
$$

$$
R_{k+1}(B)=\begin{cases}[\![x]\!]_{qX} & \text{if } B=X,\\ L_k(B) & \text{otherwise.}\end{cases}
$$

(Γ_{k+1} の生成)：もし $p\in\Gamma_k$ であるなら，Γ_{k+1} を $(\Gamma_k-\{p_1\})\cup\{A\rightarrow a(r_S),X\rightarrow x(\epsilon)\}$ とする．もし $p\notin\Gamma_k$ であるなら，$\Gamma_{k+1}=\Gamma_k$.

4.4.3 拡張による変換器の更新

(L_{k+1},R_{k+1} の生成)：$B\in N_{k+1}$ それぞれについて，

$L_{k+1}(B)=L_k(B)$，$R_{k+1}(B)=R_k(B)$.

(Γ_{k+1} の生成)：$\Gamma_{k+1}=\Gamma_k\cup\{X\rightarrow x(r_E)\}$.

4.4.4 等価変換による変換器の更新

(1)〜(8) はそれぞれ 4 節で示した等価変換の 8 個の操作に対応する．

(1)

(L_{k+1},R_{k+1} の生成)：$B\in N_{k+1}$ それぞれについて，

$L_{k+1}(B)=L_k(B)$，$R_{k+1}(B)=R_k(B)$.

(Γ_{k+1} の生成)：$\Gamma_{k+1}=\Gamma_k$.

(2)

(L_{k+1},R_{k+1} の生成)：$B\in N_{k+1}$ それぞれについて，

$L_{k+1}(B)=L_k(B)$，$R_{k+1}(B)=R_k(B)$.

(Γ_{k+1} の生成)：もし $p_1\in\Gamma_k$ であるなら，$\Gamma_{k+1}=(\Gamma_k-\{p_1\})\cup\{A\rightarrow a(r_1),A\rightarrow a(r_2)\}$ とする．もし $p_1\notin\Gamma_k$ であるなら，$\Gamma_{k+1}=\Gamma_k$.

(3)

(L_{k+1},R_{k+1} の生成)：$B\in N_{k+1}$ それぞれについて，

$L_{k+1}(B)=L_k(B)$，$R_{k+1}(B)=R_k(B)$.

(Γ_{k+1} の生成)：$\Gamma_{k+1}=\Gamma_k$.

(4)

(L_{k+1},R_{k+1} の生成)：$B\in N_{k+1}$ それぞれについて，

$L_{k+1}(B)=L_k(B)$，$R_{k+1}(B)=R_k(B)$.

(Γ_{k+1} の生成)：もし $p_1\in\Gamma_k$ であるなら，Γ_k を $(\Gamma_k-\{p_1\})\cup\{A\rightarrow a(r_1),A'\rightarrow a(r_2)\}$ とする．もし $p\notin\Gamma_k$ であるなら，$\Gamma_{k+1}=\Gamma_k$.

(木変換器の更新)：この更新操作に限り，S_{Tk+1} を以下のように生成する．

- $S_{Tk+1}=\begin{cases}S_{Tk} & \text{if } q_A\notin S_{Tk},\\ S_{Tk}\cup\{q_{A'}\} & \text{otherwise.}\end{cases}$

(5)

(L_{k+1},R_{k+1} の生成)：$B\in N_{k+1}$ それぞれについて，

$$
L_{k+1}(B)=\begin{cases}(L_k(A_1)|L_k(A_2)) & \text{if } B=A_1,\\ L_k(B) & \text{otherwise,}\end{cases}
$$

$$R_{k+1}(B) = \begin{cases} (R_k(A_1)|R_k(A_2)) & \text{if } B = A_1, \\ L_k(B) & \text{otherwise.} \end{cases}$$

(Γ_{k+1} の生成)：$\Gamma_{k+1} = (\Gamma_k - P_{A_2}) \cup \{p' \mid p \in P_{A_2} \cap \Gamma_k, p'$は p の左辺を A_1に置換したもの $\}$.

（6）

(L_{k+1}, R_{k+1} の生成)：$B \in N_{k+1}$ それぞれについて，

$$L_{k+1}(B) = \begin{cases} L_k(A) & \text{if } B = A', \\ L_k(B) & \text{otherwise,} \end{cases}$$

$$R_{k+1}(B) = \begin{cases} R_k(A) & \text{if } B = A', \\ L_k(B) & \text{otherwise.} \end{cases}$$

(Γ_{k+1} の生成)：もし $p_1 \in \Gamma$ であるなら，Γ を $(\Gamma - \{p_1\}) \cup \{A' \to a(r_1)\}$ とする．もし $p \notin \Gamma$ であるなら，$\Gamma_{k+1} = \Gamma_k$.

（7）

(L_{k+1}, R_{k+1} の生成)：$B \in (N_{k+1} - \{A_1\})$ それぞれについて，

$L_{k+1}(B) = L_k(B)$，$R_{k+1}(B) = R_k(B)$.

(Γ_{k+1} の生成)：$\Gamma_{k+1} = \Gamma_k - P_{A_1}$.

（8）

(L_{k+1}, R_{k+1} の生成)：$B \in N_{k+1}$ それぞれについて，

$L_{k+1}(B) = L_k(B)$，$R_{k+1}(B) = R_k(B)$.

(Γ_{k+1} の生成)：$\Gamma_{k+1} = \Gamma_k$.

5. 木変換器の表現能力保存性

この節では，4 節で述べたように等価変換の能力を制限したスキーマ更新操作群に対して 4 節の方法で生成される木変換器が 3 節の文書変換の定義に関する健全性と完全性を満たすことを証明する．

[補題 1] 任意の $t_1 \in TL(G_1)$，$t_k \in T_k(t_1)$ について，t_k の各ノードについて，そのノードが $p_{Ti} \in P_{Tk}$ を適用して変換されているならば，$G_k = (N_k, \Sigma, S_k, P_k)$ において t_k を導出する際に，そのノードは生成規則 $p_i \in P_k$ を適用することで得られる．ただし p_i は p_{Ti} に対応する生成規則である．

(証明) スキーマ更新操作群の適用回数に関する帰納法で証明を行う．基底段階では，4.3 節に従って構成された恒等変換を行う木変換器 T_1 については，各変換規則の生成方法より明らかである．帰納段階では，各 k 番目のスキーマ更新操作について，G_k の生成規則と T_k の変換規則とが 1 対 1 に対応づくように，L_{k+1} と R_{k+1} を生成していることを示すことによって証明する．詳細は，紙面の都合により省略する． □

[定理 1] 4 節の方法で生成される木変換器は 3 節の文書変換の定義に関して健全である．すなわち，任意の $t_1 \in TL(G_1)$，$t_{k+1} \in T_{k+1}(t_1)$ について，ある $t_k \in T_k(t_1)$ が存在して，3 節で定義した変換 T'_{k+1} に対して $t_{k+1} \in T'_{k+1}(t_k)$ である．

(証明) 各スキーマ更新操作群について場合分けを行って証明できるが，ここでは紙面の都合上，挿入のみについて証明する．

挿入によって，P_k に属する規則 $p_1 = (A \to a(r_a))$ を，P_{k+1} に属する規則 $p_2 = (A \to a(r_S))$ と $p_3 = (X \to x(\epsilon))$ に変換したとする．ただし，r_S は r_a の部分式 $r' \cdot r''$ を $r' \cdot X \cdot r''$ に置換して得られる正規表現である．

p_{T1}，p_{T2}，p_{T3} をそれぞれ p_1，p_2，p_3 に対応する変換規則とする．$t_1 \in TL(G_1)$ について，$t_{k+1} \in T_{k+1}(t_1)$ が t_1 からの変換で p_{T2}，p_{T3} を適用して得られるならば，p_{T2}，p_{T3} の代わりに p_{T1} を適用して得られる木を $t_k \in T_k(t_1)$ とおく．

補題 1 より，t_k は G_k において p_1 を適用する導出が存在する．t_{k+1} は G_k における t_k の導出において p_1 を適用するところで，p_2，p_3 を適用するような導出がある．従って，s_k，s_{k+1} をそれぞれ上のような導出に対応する t_k，t_{k+1} の解釈とすると，s_k は s_{k+1} に埋め込み可能である．

よって，$t_{k+1} \in T'_{k+1}(t_k)$ である． □

[定理 2] 4 節の方法で生成される木変換器は 3 節の文書変換の定義に関して完全である．すなわち，任意の $t_1 \in TL(G_1)$，$t_k \in T_k(t_1)$ について，3 節で定義した変換 T'_{k+1} に対して $t_{k+1} \in T'_{k+1}(t_k)$ を満たす任意の t_{k+1} について，$t_{k+1} \in T_{k+1}(t_1)$ を満たす．

(証明) 紙面の都合により証明は省略する． □

6. あとがき

本稿では，スキーマ進化がスキーマ更新操作列により表されているときに，既存の文書を新しいスキーマに従うように変換するための文書変換器の生成法を提案した．

提案手法は，等価変換の能力を制限したスキーマ更新操作群に従う木変換器の生成法であるため一部の等価変換によるスキーマ更新操作に対応できていない．また，拡張で新たに追加された規則を適用して導出された木は変換先としないことを方針としているため，拡張で新たに追加された規則を適用して導出された木に埋め込み可能である場合でも，その木に変換することはできない．よって今後はそのような問題を解決することを考えている．

また木変換器の実装についての課題もある．実装は，XSLT スタイルシートを出力することや，Java 言語などを用いて既存の XML 文書を入力として，新しいスキーマに従う XML 文書を出力するプログラムを出力することが考えられるが，以下のような問題がある．

- 木変換器は非決定性であるため，変換先が一意に決まらない．
- XSLT では，子のラベルの系列に対する正規表現のマッチングに応じた状態の遷移が表現可能かどうか明らかではない．変換先を決めるために，スキーマ更新操作群に制約を加えることや，ユーザに何らかの情報を入力してもらうことなどが考えられる．

文　献

[1] 橋本 健二，石原 靖哲，藤原 融，“XML データベースにおけるスキーマ進化のための更新操作群とそれらのスキーマ表現能力保存に関する性質，” 電子情報通信学会論文誌 (D)，Vol. J90-D, No. 4, pp. 990-1004, 2007.

[2] G. Guerrini, and D. Rossi, “Impact of XML Schema evolution on valid documents,” Proc. 7th ACM International Workshop on Web Information and Data Management, pp.39-44, 2005.

[3] M. Murata, D. Lee, M. Mani, and K. Kawaguchi, “Taxonomy of XML schema languages using formal language theory,” ACM Trans. Internet Technology, vol.5, no.4, pp. 660-704, 2005.