XMLスキーマで定義された型とXPath式との対応の解析およびその複雑度

大野　敦司† 　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 吹田市山田丘 1–5
†† 奈良先端科学技術大学院大学情報科学研究科　〒630–0192 生駒市高山町 8916–5
E-mail: †{at-ohno,ishihara,fujiwara}@ist.osaka-u.ac.jp, ††k-hasimt@is.naist.jp

あらまし　筆者らはこれまでに，XML スキーマで定義された型と XPath 部分式との対応が与えられたときに，指定された対応を満たしかつ XPath 式を充足するような XML 文書が存在するかという問題の判定法を提案した．本稿では，さまざまな問合せクラスとスキーマの組合せの下で，この問題の複雑度と充足可能性判定問題の複雑度が一致することを示す．また，各 XPath 部分式によって指定され得る XML 文書の要素が，スキーマで定義されたどの型をもつかを求める手法を新たに提案する．

キーワード　XML，XPath，有限木オートマトン，充足可能性判定

Analysis of the Correspondence Between Types Defined by an XML Schema and XPath Subexpressions and its Complexity

Atsushi OHNO†, 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
†† Graduate School of Information Science, Nara Institute of Science and Technology
Takayama-cho 8916–5, Ikoma-shi, Nara 630–0192, Japan
E-mail: †{at-ohno,ishihara,fujiwara}@ist.osaka-u.ac.jp, ††k-hasimt@is.naist.jp

Abstract　We have proposed a method of deciding, given the correspondence between types defined by an XML schema and an XPath expression, whether or not there exists an XML document such that the document satisfies the XPath expression and the correspondence exists. In this paper, we show that the complexity of this problem corresponds with the complexity of the satisfiability problem for various XPath fragments under the different XML schemas. Furthermore, we propose a method of computing the types of the elements of an XML document specified by each XPath subexpression.

Key words　XML, XPath, finite tree automaton, satisfiability analysis

1. まえがき

XML データベースのデータ構造の指定には XML スキーマが，XML 文書の特定の要素を指定する問合せ言語としては XPath [1] が広く使われており，XML スキーマや XPath は XML データベースにおける重要な研究課題となっている．

XPath 式の解析に関する研究としては，与えられたスキーマのもとでの XPath 式の充足可能性問題 [2] や包含問題 [3] を扱う研究が既に存在する．前者は，XPath 問合せ式 p とスキーマ S が与えられた場合に，XML 文書 t が S に従いかつ，t における p の答えが空でないような t が存在するかどうかを決定する問題であり，後者は，XPath 問合せ式 p_1 と p_2 が与えられた場合に，任意の (文書) 木 t について，p_1 の答えが p_2 の答えを含むかどうかを決定する問題である．XPath 部分式 p' によって指定され得る XML 文書 t の要素が，スキーマ S の型 s によって定義されているとき，p' と s が対応するという．筆者らが知る限りでは，XML スキーマで定義された型と XPath 式との対応の解析に関する研究は存在しなかった．対応を解析する問題は充足可能性判定問題を一般化した問題であり，充足可能性に加えて XML スキーマで定義された型と XPath 式との対応が分かれば，スキーマ進化 [4] に伴う XPath 式の書き換えに有用である．例えば，進化前のスキー

マを $s = \{list \to$ リスト $(stu^*tea^*), stu \to$ 学生 $(sID), tea \to$ 教師 $(tID), sID \to$ 番号 $(\epsilon), tID \to$ 番号 $(\epsilon)\}$，XPath 式を $p_1 =$↓:: リスト/ ↓:: 学生/ ↓:: 番号，$p_2 =$↓:: リスト/ ↓:: 教師/ ↓:: 番号 とする．番号という要素は s の 2 つの型 sID と tID で定義されているが，p_1 の部分式 ↓:: 番号 に対応するのは sID，p_2 の部分式 ↓:: 番号 に対応するのは tID である．s の進化により $sID \to$ 学生番号, $tID \to$ 教師番号 のように規則が変化したとき，p_1 の ↓:: 番号 に対応するのは sID，p_2 の ↓:: 番号 に対応するのは tID であることから，p_1 を ↓:: リスト/ ↓:: 学生/ ↓:: 学生番号，p_2 を ↓:: リスト/ ↓:: 教師/ ↓:: 教師番号 と書き換えなければならない．

そこで，XML スキーマで定義された型と XPath 式との対応の解析手法を提案した [5]．まず，XPath 式を有限木オートマトンに変換する手法を与えた．各部分式に対する有限木オートマトンへの変換は，軸別に変換方法を与え, XPath の構文定義に従って積オートマトンを構成していくことで，最終的に XPath 式全体を有限木オートマトンに変換する．そして，得られた XPath 式の有限木オートマトンと XML スキーマを表す有限木オートマトンの積オートマトンを構成する．完成した積オートマトンは，XPath 式を充足しかつ XML スキーマに従う木を受理する．この積オートマトンを解析することで，XML スキーマで定義された型と XPath 部分式との対応が与えられたときに，指定された対応を満たしかつ XPath 式を充足するような XML 文書が存在するかという問題の判定を行う．しかし，この判定問題の複雑度については調査を行っていなかった．

本稿では，この判定問題の複雑度について調査する．さまざまな XPath 問合せクラスとスキーマの組合せの下でこの判定問題から充足可能性判定問題への帰着を与える．それにより，判定問題と充足可能性判定問題の複雑度が一致することを示す．

判定問題を解くことで各 XPath 部分式とスキーマで定義されたある型の間の対応の有無を知ることができるが，各部分式にスキーマで定義されたどの型が対応するかを直接求めることができればより有用である．判定問題の結果から，XML スキーマで定義された型と XPath 部分式との対応を求めることは可能である．しかし，そのためには判定問題への入力を変えながら結果を確かめることを繰り返し行わなくてはならない．そこで，そのような対応を知るための手法を提案する．XPath 式の有限木オートマトンと XML スキーマを表す有限木オートマトンの積オートマトンを構成し，さらにそこから「部分式とスキーマの正しい対応情報が入力文字列として与えられたときにそれを受理するような有限オートマトン」を構成する．これにより，各部分式にスキーマで定義されたどの型が対応するかを求める．

2. 諸 定 義

2.1 XML 文書

ここでは，文献 [6] を参考に，本稿で考える XML 文書を定義する．

［定義 1］（XML 文書） XML 文書はラベル付き順序木で表される．ノード v のラベルを $\lambda(v)$ と書く．ラベルは要素名に相当する．また，λ をノードの系列上の関数に拡張する．つまり，ノードの系列 $v_1 \cdots v_n$ について，$\lambda(v_1 \cdots v_n) = \lambda(v_1) \cdots \lambda(v_n)$ である．

以下では, ラベル付き順序木を単に木と呼ぶ.

2.2 XPath

ここでは，文献 [2], [5], [6] を参考に，本稿で対象とする XPath の構文および意味論を与える．

［定義 2］（XPath 構文） Σ をアルファベットとする．XPath 式 p を以下のように定義する:

$$
\begin{aligned}
p &::= \chi :: l \mid p/p \mid p \cup p \mid p[q],\\
\chi &::= \cdot \mid \downarrow \mid \uparrow \mid \downarrow^+ \mid \uparrow^+ \mid \downarrow^* \mid \uparrow^* \mid \rightarrow^+ \mid \leftarrow^+ \mid \\
&\qquad \searrow \mid \swarrow,\\
q &::= p \mid q \wedge q \mid q \vee q
\end{aligned}
$$

ただし，$l \in \Sigma$ である．各 $\chi \in \{\cdot, \downarrow, \uparrow, \downarrow^+, \uparrow^+, \downarrow^*, \uparrow^*, \rightarrow^+, \leftarrow^+, \searrow, \swarrow\}$ を軸，q を述語という．

次に，XPath 式の位置を定義する．

［定義 3］（XPath 式の位置 [5]） XPath 式 p の位置 α とは正整数の有限系列であり，p の部分式 $p|_\alpha$ を指定する．$p|_\alpha$ は以下のように定義される:

- $p|_\epsilon = p$
- $p|_\alpha = p'$ とすると，
  - $p' = p'_1/p'_2$ のとき，$p|_{\alpha\cdot 1} = p'_1$, $p|_{\alpha\cdot 2} = p'_2$
  - $p' = p'_1 \cup p'_2$ のとき，$p|_{\alpha\cdot 1} = p'_1$, $p|_{\alpha\cdot 2} = p'_2$
  - $p' = p''[q]$ のとき，$p|_{\alpha\cdot 1} = p''$, $p|_{\alpha\cdot 2} = q$
  - $p' = q_1 \wedge q_2$ のとき，$p|_{\alpha\cdot 1} = q_1$, $p|_{\alpha\cdot 2} = q_2$
  - $p' = q_1 \vee q_2$ のとき，$p|_{\alpha\cdot 1} = q_1$, $p|_{\alpha\cdot 2} = q_2$

［定義 4］（XPath 意味論） 木 t が XPath 式 p の部分式 $p|_\alpha$ を充足するとは，ある θ, ρ について $(t, \theta, \rho, \alpha) \models p$ が成立することであると定義する．ここで θ, ρ は XPath 式の位置から，木 t のノード集合への写像であり，関係 $\models$ は以下のように定義される．直観的には，部分式がノードを指定する基準点となるノードをコンテクストノードといい，θ は XPath 式の位置から部分式のコンテクストノードへの写像，ρ は XPath 式の位置から部分式が指定するノードへの写像である．

- $(t, \theta, \rho, \alpha) \models (\cdot :: l)$

$\theta(\alpha) = \rho(\alpha)$ かつ $\lambda(\rho(\alpha)) = l$.

- $(t, \theta, \rho, \alpha) \models (\downarrow :: l)$

$\rho(\alpha)$ は $\theta(\alpha)$ の子であり，かつ $\lambda(\rho(\alpha)) = l$.

- $(t, \theta, \rho, \alpha) \models (\uparrow :: l)$

$\rho(\alpha)$ は $\theta(\alpha)$ の親であり，かつ $\lambda(\rho(\alpha)) = l$.

- $(t, \theta, \rho, \alpha) \models (\downarrow^+ :: l)$

$\rho(\alpha)$ は $\theta(\alpha)$ の子孫であり，かつ $\lambda(\rho(\alpha)) = l$.

- $(t, \theta, \rho, \alpha) \models (\uparrow^+ :: l)$

$\rho(\alpha)$ は $\theta(\alpha)$ の祖先であり，かつ $\lambda(\rho(\alpha)) = l$.

- $(t, \theta, \rho, \alpha) \models (\downarrow^* :: l)$

$\rho(\alpha)$ は $\theta(\alpha)$ 自身またはその子孫であり，かつ $\lambda(\rho(\alpha)) = l$.

- $(t, \theta, \rho, \alpha) \models (\uparrow^* :: l)$

$\rho(\alpha)$ は $\theta(\alpha)$ 自身またはその祖先であり，かつ $\lambda(\rho(\alpha)) = l$.

- $(t,\theta,\rho,\alpha) \models (\rightarrow^{+}:: l)$

$\rho(\alpha)$ は $\theta(\alpha)$ の弟であり，かつ $\lambda(\rho(\alpha)) = l$.

- $(t,\theta,\rho,\alpha) \models (\leftarrow^{+}:: l)$

$\rho(\alpha)$ は $\theta(\alpha)$ の兄であり，かつ $\lambda(\rho(\alpha)) = l$.

- $(t,\theta,\rho,\alpha) \models (\searrow:: l)$

$\rho(\alpha)$ は $\theta(\alpha)$ のある祖先 n の弟 n' 自身またはその子孫であり，かつ $\lambda(\rho(\alpha)) = l$.

- $(t,\theta,\rho,\alpha) \models (\swarrow:: l)$

$\rho(\alpha)$ は $\theta(\alpha)$ のある祖先 n の兄 n' 自身またはその子孫であり，かつ $\lambda(\rho(\alpha)) = l$.

- $(t,\theta,\rho,\alpha) \models p_1/p_2$

$(t,\theta,\rho,\alpha\cdot 1) \models p_1$ かつ $(t,\theta,\rho,\alpha\cdot 2) \models p_2$ かつ $\theta(\alpha) = \theta(\alpha\cdot 1)$ かつ $\rho(\alpha) = \rho(\alpha\cdot 2)$ かつ $\rho(\alpha\cdot 1) = \theta(\alpha\cdot 2)$.

- $(t,\theta,\rho,\alpha) \models (p_1 \cup p_2)$

$(t,\theta,\rho,\alpha\cdot 1) \models p_1$ かつ $\theta(\alpha) = \theta(\alpha\cdot 1)$ かつ $\rho(\alpha) = \rho(\alpha\cdot 1)$，または $(t,\theta,\rho,\alpha\cdot 2) \models p_2$ かつ $\theta(\alpha) = \theta(\alpha\cdot 2)$ かつ $\rho(\alpha) = \rho(\alpha\cdot 2)$.

- $(t,\theta,\rho,\alpha) \models (p[q])$

$(t,\theta,\rho,\alpha\cdot 1) \models p$ かつ $(t,\theta,\rho,\alpha\cdot 2) \models q$ かつ $\theta(\alpha) = \theta(\alpha\cdot 1)$ かつ $\rho(\alpha) = \rho(\alpha\cdot 1) = \theta(\alpha\cdot 2)$.

- $(t,\theta,\rho,\alpha) \models (q_1 \wedge q_2)$

$(t,\theta,\rho,\alpha\cdot 1) \models q_1$ かつ $(t,\theta,\rho,\alpha\cdot 2) \models q_2$ かつ $\theta(\alpha) = \theta(\alpha\cdot 1) = \theta(\alpha\cdot 2)$.

- $(t,\theta,\rho,\alpha) \models (q_1 \vee q_2)$

$(t,\theta,\rho,\alpha\cdot 1) \models q_1$ かつ $\theta(\alpha) = \theta(\alpha\cdot 1)$，または $(t,\theta,\rho,\alpha\cdot 2) \models q_2$ かつ $\theta(\alpha) = \theta(\alpha\cdot 2)$.

2.3 有限木オートマトン

本稿で考える有限木オートマトン（正規木文法），局所木オートマトン（局所木文法），解釈，受理の各定義を文献 [7] に従って与える．

［定義 5］（有限木オートマトン） 有限木オートマトン $TA = (N, \Sigma, s, P)$ を定義する．

- N は状態の有限集合，
- Σ は要素名の有限集合，
- $s \in N$ は初期状態，
- P は $X \rightarrow a(reg)$ または $X \rightarrow Y$ の形をした遷移規則の有限集合．

ただし，$X, Y \in N$，$a \in \Sigma$ であり，reg は N 上の正規表現である．X を規則の左辺，Y, $a(reg)$ を右辺といい，a，reg をそれぞれこの規則のラベル，内容モデルという．

局所木オートマトンは有限木オートマトンに次の性質を加えたものである．

- $\forall X_i \rightarrow a_i(reg_i), X_j \rightarrow a_j(reg_j) \in P, i \neq j \Rightarrow a_i \neq a_j$

ただし，$X_i, X_j \in N$，$a_i, a_j \in \Sigma$ であり，reg_i, reg_j は N 上の正規表現である．$X_i \rightarrow a_i(reg_i), X_j \rightarrow a_j(reg_j) \in P, i \neq j$，かつ $a_i = a_j$ であるとき，X_i と X_j は競合するという．

本稿ではスキーマを有限木オートマトン TA_s で表現する．ただし，TA_s には ϵ 遷移は含まれない．スキーマの型は有限木オートマトンの状態で表現される．

［定義 6］（解釈） 有限木オートマトン TA に対する木 t の解釈 I は，t 中の各ノード v から状態 $I(v)$ への以下を満たす写像である:

- v が t の根である場合，$I(v)$ は初期状態である．
- 各ノード v とその子ノード $v_1, \ldots, v_n$ について，TA 中の遷移規則 $X_1 \rightarrow X_2,\ X_2 \rightarrow X_3, \ldots,\ X_k \rightarrow a(reg)$ が存在し，以下を満たす:
  - $I(v) = X_1 X_2 \cdots X_k$,
  - $\lambda(v) = a$,
  - $first(I(v_1)) \ldots first(I(v_n))$ が reg にマッチする．ただし，$first$ は $first(X_1 \cdots X_k) = X_1$ を満たす関数である．

有限木オートマトン TA によって受理される木の集合を $TL(TA)$ と表す．有限木オートマトン TA が木 t を受理するとは，TA に対する t の解釈が存在することである．

2.4 問題定義と対応成立の条件式

ここでは対応の判定問題 $\mathbf{SAT}^{+}$，対応を求める問題と対応成立の条件式を定義する．スキーマを TA_s，XPath 式を p，p の位置の系列を $\alpha_1, \ldots, \alpha_k$，$TA_s$ 中の型の系列を $n_{s_1}, \ldots, n_{s_k}$ とする．

［定義 7］（対応成立の条件式） 対応成立の条件式は以下のとおり:

- $\exists t \in TL(TA_s),\ \exists I,\ \exists \theta,\ \exists \rho$
  - $(t,\theta,\rho,\epsilon) \models p$ かつ，
  - $\forall i (1 \leqq i \leqq k),\ (t,\theta,\rho,\alpha_i) \models p|_{\alpha_i} \ \wedge\ I(\rho(\alpha_i)) = n_{s_i}$

ただし，I は t の解釈である．

［定義 8］（対応の判定問題 $\mathbf{SAT}^{+}$） 対応の判定問題 $\mathbf{SAT}^{+}$ とは，対応成立の条件式が成り立つかどうかを判定する問題である．

［定義 9］（対応を求める問題） 対応を求める問題とは，対応成立の条件式を満たす系列 $n_{s_1}, \ldots, n_{s_k}$ すべてを求める問題である．

3. $\mathbf{SAT}^{+}$ の複雑度

ここでは，$\mathbf{SAT}^{+}$ の複雑度の結果と $\mathbf{SAT}^{+}$ の充足可能性判定問題への帰着を示す．

3.1 $\mathbf{SAT}^{+}$ の複雑度

［定義 10］（disjunction 無し XML スキーマ） disjunction 無し XML スキーマは以下の性質を満たすスキーマである:

- あらゆる内容モデル（遷移規則の右辺の正規表現）内に disjunction が現れない．
- 一つの状態に対応する要素名はちょうど 1 つである

$\mathbf{SAT}^{+}$ の複雑度を表 1 に示す．各問合せクラスと disjunction をもたないスキーマ，disjunction をもつスキーマの下での複雑度を示している．

3.2 充足可能性判定問題への帰着

充足可能性判定問題への帰着を簡単に説明する．まず，使用する定理（文献 [4]）を示す．局所木文法における充足可能性判定問題を $\mathbf{SAT}_L(\mathcal{X})$，正規木文法における充足可能性判定問題を $\mathbf{SAT}_R(\mathcal{X})$ とする．

［定理 1］ 任意の問合せクラス $\mathcal{X}$ において，$\mathbf{SAT}_L(\mathcal{X})$ と $\mathbf{SAT}_R(\mathcal{X})$ は相互に多項式時間で帰着可能．

［証明 ］ 定理 1 の証明を行う．スキーマクラスの包含関係

表 1　$\mathbf{SAT}^+$ の複雑度

	disjunction 無し	disjunction 有り
$\mathcal{X}(\downarrow,\downarrow^*,\cup)$	P	P
$\mathcal{X}(\cup,[\,])$	P	NP 完全
$\mathcal{X}(\downarrow,\uparrow)$	P	NP 完全
$\mathcal{X}(\downarrow,[\,])$	P	NP 完全
$\mathcal{X}(\downarrow,\downarrow^*,\cup,[\,])$	P	NP 完全
$\mathcal{X}(\downarrow,\uparrow,\cup,[\,])$	NP 完全	NP 完全
$\mathcal{X}(\downarrow,\downarrow^*,\uparrow,\uparrow^*,\cup,[\,])$	NP 完全	NP 完全

から，$\mathbf{SAT}_L(\mathcal{X})$ は $\mathbf{SAT}_R(\mathcal{X})$ に明らかに帰着可能である．$\mathbf{SAT}_R(\mathcal{X})$ から $\mathbf{SAT}_L(\mathcal{X})$ への帰着は以下の通りである．

正規木文法 G_R において，終端記号 a について k 個のスキーマ生成規則が競合していると仮定する．競合を解決するために，$A_1 \to a(r_1), A_2 \to a(r_2), ..., A_k \to a(r_k)$ を $A_1 \to a_1(r_1), A_2 \to a_2(r_2), ..., A_k \to a_k(r_k)$ に書き換える．このように，G_R 中のすべての競合しているスキーマ生成規則の終端記号を書き換えることで，G_R を局所木文法 G_L に変換することができる．

$\mathbf{SAT}_R(\mathcal{X})$ の入力問合せ p_1 の $\chi :: a$ を，競合の解決に基づき $(\chi :: a_1 \cup \cdots \cup \chi :: a_k)$ に書き換え，問合せ p_2 を構成する．このとき，G_R における p_1 の充足可能性と G_L における p_2 の充足可能性は一致する．スキーマと問合せの書き換えに要する計算量は $O(|G_R||p|)$ である．□

［定理 2］ $\mathbf{SAT}^+$ の $\mathbf{SAT}$ への帰着は次に示す手順で可能．

まず，定理 1 によって，$\mathbf{SAT}_R(\mathcal{X})$ を $\mathbf{SAT}_L(\mathcal{X})$ に帰着する．そして，XPath 式中の競合解決を行った各部分式 $(\chi :: a_1 \cup \cdots \cup \chi :: a_k)$ において，その部分式に対して指定された型に属する要素 a_j 含む部分のみを残し $\chi :: a_j$ とすることで $\mathbf{SAT}^+(\mathcal{X})$ を $\mathbf{SAT}_L(\mathcal{X})$ に帰着可能である．

［証明 ］ 定理 2 の証明を行う．帰着前のスキーマ（正規木文法）を G_R，XPath 式を p，p の位置の系列を $\alpha_1, \ldots, \alpha_k$，$G_R$ 中の型の系列を $X_1, \ldots, X_m$ とする．p の部分式 $p|_{\alpha_i} = \chi :: a$ と終端記号 a を定義している G_R の型 X_i が対応する場合を考える．$p = p_1/\chi :: a/p_2$ であり，終端記号 a について k 個のスキーマ生成規則が競合していると仮定する．

競合を解決するために，G_R のスキーマ生成規則 $A_1 \to a(r_1), A_2 \to a(r_2), ..., A_k \to a(r_k)$ を $A_1 \to a_1(r_1), A_2 \to a_2(r_2), ..., A_k \to a_k(r_k)$ に書き換える．ただし，$X_i = A_j (1 \leqq j \leqq k)$ とする．

p の $\chi :: a$ を，競合の解決に基づき $(\chi :: a_1 \cup \cdots \cup \chi :: a_k)$ に書き換えた XPath 式を $p' = p_1/(\chi :: a_1 \cup \cdots \cup \chi :: a_k)/p_2$ とする．XPath の意味定義より p' は XPath 式 $p'' = (p_1/\chi :: a_1/p_2) \cup \cdots \cup (p_1/\chi :: a_k/p_2)$ と等価である．

p'' の部分式 $p_1/\chi :: a_1/p_2$，$\cdots$，$p_1/\chi :: a_k/p_2$ のうちの少なくとも 1 つの部分式 $p_1/\chi :: a_n/p_2 (1 \leqq n \leqq k)$ が充足可能であれば p'' は充足可能であり，定理 1 より p も充足可能である．またこのとき，$p_1/\chi :: a_n/p_2 (1 \leqq n \leqq k)$ の $\chi :: a_n$ はスキーマ生成規則 A_n で定義されていることから p の部分式 $\chi :: a$ に A_n が対応する．

以上より，p の $\chi :: a$ を，競合の解決に基づいて書き換えるときに p の部分式 $\chi :: a$ に対して指定された型に属する要素 a_j を含む部分のみを残し，p を $p_{cut} = p_1/\chi :: a_j/p_2$ と書き換えると，p_{cut} が充足可能であれば p も充足可能であり，$\chi :: a_j$ はスキーマ生成規則 $A_j = X_i$ で定義されていることから p の部分式 $\chi :: a$ に X_i が対応するので，p_{cut} が充足可能であるならば帰着元の $\mathbf{SAT}^+$ は真と判定され，p_{cut} が充足不能であるならば帰着元の $\mathbf{SAT}^+$ は偽と判定される．□

4. XPath 式の有限木オートマトンの構成と対応の解析

ここでは XPath 式に対する有限木オートマトンを構成する手法（文献 [5]）と，スキーマで定義された型と XPath 部分式との対応の解析手法を与える．

文献 [5] で提案した変換手法では，定義 2 の XPath 式の構文定義に従って以下のように変換を進める．まず $\chi :: l$ を有限木オートマトンに変換する．この変換は各軸 χ に対して用意する有限木オートマトンを用いて行う．用意された有限木オートマトンには，入力状態と出力状態があり，それぞれ $\chi :: l$ に対するコンテクストノード，$\chi :: l$ が指定するノードを出力する状態である．

次に XPath 式の構文に従って，ボトムアップ的に有限木オートマトンを構成していく．この際，有限木オートマトンの出力状態と入力状態を考慮する必要がある．

4.1 各軸に対する有限木オートマトン

ここでは $\chi :: l$ という形の部分式に対する有限木オートマトンを与える．部分式に出現する軸によって，各部分式に対する有限木オートマトンが異なる．以下では，$A \to \sigma(\cdots)$ という記述は $\{A \to \sigma(\cdots) \mid \sigma \in \Sigma\}$ を表すものとする．

- $\cdot :: l$ に対する有限木オートマトン
  - 初期状態：B
  - $A \to \sigma(A^*)$
  - $B \to \sigma(A^*\ (B|C)\ A^*)$
  - $B \to C$
  - $C \to D$
  - $D \to l(A^*)$
- $\downarrow :: l$ に対する有限木オートマトン
  - 初期状態：B
  - $A \to \sigma(A^*)$
  - $B \to \sigma(A^*\ (B|C)\ A^*)$
  - $B \to C$
  - $C \to \sigma(A^*\ D\ A^*)$
  - $D \to l(A^*)$
- $\downarrow^+ :: l$ に対する有限木オートマトン
  - 初期状態：B_1
  - $A \to \sigma(A^*)$
  - $B_1 \to \sigma(A^*\ (B_1|C)\ A^*)$
  - $B_1 \to C$
  - $C \to \sigma(A^*\ (B_2|D)\ A^*)$
  - $B_2 \to \sigma(A^*\ (B_2|D)\ A^*)$

– $D \rightarrow l(A^*)$
- $\downarrow^*::l$ に対する有限木オートマトン
– 初期状態：B_1
– $A \rightarrow \sigma(A^*)$
– $B_1 \rightarrow \sigma(A^*\ (B_1|C)\ A^*)$
– $B_1 \rightarrow C$
– $C \rightarrow \sigma(A^*\ (B_2|D)\ A^*)$
– $C \rightarrow D$
– $B_2 \rightarrow \sigma(A^*\ (B_2|D)\ A^*)$
– $D \rightarrow l(A^*)$
- $\rightarrow^+::l$ に対する有限木オートマトン
– 初期状態：B_1
– $A \rightarrow \sigma(A^*)$
– $B_1 \rightarrow \sigma(A^*\ (B_1|B_2)\ A^*)$
– $B_1 \rightarrow B_2$
– $B_2 \rightarrow \sigma(A^*\ C\ A^*\ D\ A^*)$
– $C \rightarrow \sigma(A^*)$
– $D \rightarrow l(A^*)$
- $\searrow::l$ に対する有限木オートマトン
– 初期状態：B_1
– $A \rightarrow \sigma(A^*)$
– $B_1 \rightarrow \sigma(A^*\ (B_1|B_2)\ A^*)$
– $B_1 \rightarrow B_2$
– $B_2 \rightarrow \sigma(A^*\ (B_3|C)\ A^*\ (B_4|D)\ A^*)$
– $B_3 \rightarrow \sigma(A^*\ (B_3|C)\ A^*)$
– $B_4 \rightarrow \sigma(A^*\ (B_4|D)\ A^*)$
– $C \rightarrow \sigma(A^*)$
– $D \rightarrow l(A^*)$

各軸に対する有限木オートマトンの状態のうち，C は部分式のコンテクストノードに対応する状態であり，D は部分式が指定するノードに対応する状態である．これらの有限オートマトンの入力状態集合は $\{C\}$ であり，出力状態集合は $\{D\}$ である．

XPath の軸の中で，↑，$\uparrow^+$，$\uparrow^*$，←，↙ のことを reverse step と呼び，それぞれ ↓，$\downarrow^+$，$\downarrow^*$，→，↘ とは逆向きにノードを辿る軸である．reverse step を含む部分式を有限木オートマトンに変換するには，ここで与えた各軸に対する有限木オートマトンの規則のうち，C と D の規則の終端記号をそれぞれ l と σ に書き換え，次に全規則中の C を D に，D を C に書き換えることで対応可能である．

4.2 XPath 式の有限木オートマトンの構成法

ここでは XPath 式の有限木オートマトンの構成法を与える．

p_1/p_2，$p[q]$，$p_1 \cup p_2$，$q_1 \wedge q_2$，$q_1 \vee q_2$ それぞれの場合について XPath 式の有限木オートマトンの構成法が存在する．p の有限木オートマトン TA_p の状態集合を N_p，入力状態集合を $NI_p \subset N_p$，出力状態集合を $NO_p \subset N_p$ と表す．

4.3 p_1/p_2 の場合

p_1，p_2 はそれぞれ有限木オートマトン TA_{p_1}，TA_{p_2} に変換済みであるとする．TA_{p_1} の出力状態と，TA_{p_2} の入力状態以外の状態のペアや，TA_{p_1} の出力状態以外の状態と，TA_{p_2} の入力状態のペアが，積を取ったオートマトンの状態に現れないように積を取る．すなわち，積を取ったオートマトン TA_{p_1/p_2} の状態集合は，

$$N_{p_1/p_2} = (NO_{p_1} \times NI_{p_2}) \cup ((N_{p_1} - NO_{p_1}) \times (N_{p_2} - NI_{p_2}))$$

となる．また，入力状態集合は，

$$NI_{p_1/p_2} = NI_{p_1} \times N_{p_2}$$

出力状態集合は，

$$NO_{p_1/p_2} = N_{p_1} \times NO_{p_2}$$

となる．

4.4 $p[q]$ の場合

p，q はそれぞれ有限木オートマトン TA_p，TA_q に変換済みであるとする．TA_p の出力状態と，TA_q の入力状態以外の状態のペアや，TA_p の出力状態以外の状態と，TA_q の入力状態のペアが，積を取ったオートマトンの状態に現れないように積を取る．

$$N_{p[q]} = (NO_p \times NI_q) \cup ((N_p - NO_p) \times (N_q - NI_q))$$

となる．また，入力状態集合は，

$$NI_{p[q]} = NI_p \times N_q$$

出力状態集合は，

$$NO_{p[q]} = NO_p \times NI_q$$

となる．

4.5 $p_1 \cup p_2$ の場合

p_1，p_2 はそれぞれ有限木オートマトン TA_{p_1}，TA_{p_2} に変換済みであるとする．TA_{p_1}，TA_{p_2} の和オートマトン $TA_{p_1 \cup p_2}$ を構成する．新しい初期状態を n_{ini} とする．ただし，TA_{p_1} の状態名と TA_{p_2} の状態名が重複しないようにするため，状態に対してユニークな整数をインデックスとして付加する（例：$[A[DC]]_1$）．

和を取ったオートマトン $TA_{p_1 \cup p_2}$ の状態集合は，

$$N_{p_1 \cup p_2} = N_{p_1} \cup N_{p_2} \cup \{n_{ini}\}$$

となる．また，入力状態集合は，

$$NI_{p_1 \cup p_2} = \{n_{ini}\}$$

出力状態集合は，

$$NO_{p_1 \cup p_2} = NO_{p_1} \ \cup \ NO_{p_2}$$

となる．

4.6 $q_1 \wedge q_2$ の場合

q_1, q_2 はそれぞれ有限木オートマトン TA_{q_1}, TA_{q_2} に変換済みであるとする．TA_{q_1}, TA_{q_2} の積オートマトン $TA_{q_1 \wedge q_2}$ を構成する．$TA_{q_1 \wedge q_2}$ の状態集合は

$$N_{q_1 \wedge q_2} = N_{q_1} \times N_{q_2}$$

となる．また，入力状態集合は，

$$NI_{q_1 \wedge q_2} = NI_{q_1} \times NI_{q_2}$$

出力状態集合は，

$$NO_{q_1 \wedge q_2} = (NO_{q_1} \times N_{q_2}) \cup (N_{q_1} \times NO_{q_2})$$

となる．

4.7 $q_1 \vee q_2$ の場合

q_1, q_2 はそれぞれ有限木オートマトン TA_{q_1}, TA_{q_2} に変換済みであるとする．TA_{q_1}, TA_{q_2} の和オートマトン $TA_{q_1 \cup q_2}$ を構成する．新しい初期状態を n_{ini} とする．$p_1 \cup p_2$ の場合と同じように，TA_{q_1} の状態名と TA_{q_2} の状態名が重複しないようにするため，状態に対してユニークな整数をインデックスとして付加する．

和を取ったオートマトン $TA_{q_1 \cup q_2}$ の状態集合は，

$$N_{q_1 \cup q_2} = N_{q_1} \cup N_{q_2} \cup \{n_{ini}\}$$

となる．また，入力状態集合は，

$$NI_{q_1 \cup q_2} = \{n_{ini}\}$$

出力状態集合は，

$$NO_{q_1 \cup q_2} = NO_{q_1} \cup NO_{q_2}$$

となる．

5. スキーマで定義された型と XPath 部分式との対応の解析

ここでは，スキーマで定義された型と XPath 部分式との対応の解析手法を与える．まず，XPath 式 p から得た有限木オートマトン TA_p と，スキーマを表す有限木オートマトン TA_s の積オートマトン $TA_{p \cap s}$ を構成する．次に $TA_{p \cap s}$ を解析することで，定義 7 の対応成立の条件式の真偽を判定できる．以下の条件を満たすときかつそのときのみ対応成立の条件式は真と判定される．

- ある $t \in TL(TA_{p \cap s})$, ある I, t のあるノード $v_1, \ldots, v_k$ について
  - $I(v_i)(1 \leqq i \leqq k)$ が $p|_{\alpha_i}$ に対して構成された有限木オートマトンのある出力状態および n_{s_i} の両方を成分として含む

ただし，I は $TA_{p \cap s}$ に対する t の解釈である．上記の条件の判定は以下のようにして行える．$TA_{p \cap s}$ の各状態 $n_{p \cap s}$ から同時に到達可能な状態の極大集合をすべて求める．得た状態の極大集合に，各部分式の出力状態と指定された型に対応する状態が対となっている状態のすべてが含まれているかどうかを判定する．このような極大集合が存在すれば，上記の条件は真と判定できる．

この判定を行うために $TA_{p \cap s}$ から有限文字列オートマトン A_c を作成する．A は入力文字列として $TA_{p \cap s}$ の遷移規則を深さ優先的に辿った場合に経る状態の系列から出力状態以外の状態を取り除いた系列を与えられた場合にそれを受理するようなオートマトンである．

A_c が $I(v_1) \cdots I(v_k)$ を入力文字列として受理，すなわち上記の条件式の ある $t \in TL(TA_{p \cap s})$, ある I, t のあるノード $v_1, \ldots, v_k$ について，$I(v_i)(1 \leqq i \leqq k)$ が $p|_{\alpha_i}$ に対して構成された有限木オートマトンのある出力状態および n_{s_i} の両方を成分として含むとき，上記の判定条件式は真と判定できる．

上記の条件の判定を行うために $TA_{p \cap s}$ から文字列有限オートマトン A_c を構成する．

5.1 文字列オートマトンの構成

ここでは，入力として与えられた $TA_{p \cap s}$ から文字列有限オートマトン A_c を構成するアルゴリズムを与える．アルゴリズムは以下のとおり:

(1) $TA_{p \cap s}$ の各遷移規則から終端記号を取り除く．

(1) によって $TA_{p \cap s}$ の遷移規則は $[X_n]S_n \rightarrow REG_n$ の形になる．ただし，$[X_n]$ は TA_p の状態由来の成分，S_n は TA_s の状態由来の成分，REG_n は $TA_{p \cap s}$ の状態集合上の正規表現である．

(2) 左辺の状態 $[X_n]S_n$ の $[X_n]$ に A のみが現れるような遷移規則を削除する．他の遷移規則 $[X_m]S_m \rightarrow REG_m$ において，$[X_n]S_n \in REG_m$ ならば REG_m 中のすべての $[X_n]S_n$ を ϵ で置換する．処理対象となる遷移規則が無くなるまでこの処理を繰り返す．

遷移規則 $[X_n]S_n \rightarrow REG_n$ の展開とは，他のすべての遷移規則 $[X_m]S_m \rightarrow REG_m$ 中に現れる $[X_n]S_n$ を，$[X_n]S_n$ が出力状態ならば $([X_n]S_n(REG_n))$，$[X_n]S_n$ が出力状態でないなら (REG_n) で置換することである．置換により得られた遷移規則 $[X_m]S_m \rightarrow REG'_m$ において，$[X_m]S_m \in REG'_m$ ならば，REG'_m 中の $[X_m]S_m$ を ϵ で置換する（再帰の解消）．置換後，遷移規則 $[X_n]S_n \rightarrow REG_n$ を削除する．

(3) 右辺が ϵ である遷移規則 $[X_n]S_n \rightarrow \epsilon$ を展開する．処理対象となる遷移規則が無くなるまでこの処理を繰り返す．ただし，初期状態を含む遷移規則は除く．

(4) 右辺に出力状態をもつ遷移規則 $[X_n]S_n \rightarrow REG_n$ を展開する．処理対象となる遷移規則が無くなるまでこの処理を繰り返す．ただし，初期状態を含む遷移規則は除く．

(5) 初期状態を含む遷移規則 $[X_s]S_s \rightarrow REG_s$ で，REG_s に存在する出力状態以外の状態を ϵ で置換する．$[X_s]S_s$ が出力状態ならば $[X_s]S_s(REG_s)$ が表す文字列集合を認識する文字列有限オートマトン A_c を，そうでなければ REG_s が表す文字列集合を認識する文字列有限オートマトン A_c を出力する．

5.2 対応を解析する文字列オートマトンの構成の正しさ

$TA_{p \cap s}$ の状態集合を N，出力状態集合を N_{out}，出力状態以外の集合を $\overline{N_{out}} = N - N_{out}$ とする．$\overline{N_{out}} = N_{allA} \cup N_{unReach} \cup N_{unReachLoop} \cup N_{Reach}$ である．た

だし，

- N_{allA} は状態 $[X_n]S_n$ の $[X_n]$ に A のみが現れるような状態の集合，
- $N_{unReachLoop}$ は出力状態に到達不能な出力状態以外の状態（N_{allA} に属する状態を除く）で再帰に含まれる，もしくは再帰に含まれる状態に到達するものの集合，
- $N_{unReach}$ は出力状態に到達不能な出力状態以外の状態（N_{allA} に属する状態を除く）で，$N_{unReachLoop}$ に属さないものの集合，
- N_{Reach} は出力状態に到達可能な出力状態以外の状態（N_{allA} に属する状態を除く）の集合，

N_{allA}，$N_{unReachLoop}$，$N_{unReach}$，N_{Reach} は互いに素である．

t に対する $TA_{p\cap s}$ の実行を $I_t^{p\cap s}$，f は以下のような関数とする．

- f : 有限木オートマトンの実行 $\rightarrow$ 状態名の系列
- $f(I_t^{p\cap s}) = I_t^{p\cap s}(v_1)\cdots I_t^{p\cap s}(v_i)$, ただし，$v_1 \ldots v_i \in t$ は前順.

$TA_{p\cap s}$ の各遷移規則から終端記号を取り除き，すべての遷移規則を展開すると，言語 $L = \{f(I_t^{p\cap s}) \mid t \in TL(TA_{p\cap s})\}$ が得られる．L の出力状態（文字）以外を空文字に置換した言語 $L[n \leftarrow \epsilon \mid n \notin N_{out}]$ を文字列オートマトン A_c が認識すれば，A_c を正しく構成できたと言える．構成の各ステップについて，正しいことを確認する．

(1) $TA_{p\cap s}$ の各遷移規則から終端記号を取り除く．

この時点ですべての遷移規則を展開すると，言語 L を受理する文字列オートマトンが得られる．

(2) 左辺の状態 $[X_n]S_n$ の $[X_n]$ に A のみが現れるような遷移規則を削除する．他の遷移規則 $[X_m]S_m \rightarrow REG_m$ において，$[X_n]S_n \in REG_m$ ならば REG_m 中のすべての $[X_n]S_n$ を ϵ で置換する．処理対象となる遷移規則が無くなるまでこの処理を繰り返す.

テンプレートオートマトンと XPath 式に対する有限木オートマトンの構成方法より，左辺に A のみがあらわれる規則は出力状態ではなく，必ず A のみがあらわれる状態に遷移する．この時点ですべての遷移規則を展開すると，言語 $L[n \leftarrow \epsilon \mid n \in N_{allA}]$ を受理する文字列オートマトンが得られる．

(3) 右辺が ϵ である遷移規則 $[X_n]S_n \rightarrow \epsilon$ を展開する．処理対象となる遷移規則が無くなるまでこの処理を繰り返す．ただし，初期状態を含む遷移規則は除く．

右辺が ϵ である遷移規則から他の遷移規則へ遷移することはあり得ないので，この遷移規則を削除する．また，遷移規則の左辺 N_{left} が出力状態でないならその状態は $N_{unReach}$ に属する．よって，この時点ですべての遷移規則を展開すると，言語 $L[n \leftarrow \epsilon \mid n \in N_{allA} \cup N_{unReach}]$ を受理する文字列オートマトンが得られる．

(4) 右辺に出力状態をもつ遷移規則 $[X_n]S_n \rightarrow REG_n$ を展開する．処理対象となる遷移規則が無くなるまでこの処理を繰り返す．ただし，初期状態を含む遷移規則は除く．

ここで展開される遷移規則の左辺の状態が出力状態でないなら必ず N_{Reach} に属する．この時点ですべての遷移規則を展開すると，言語 $L[n \leftarrow \epsilon \mid n \in N_{allA} \cup N_{unReach} \cup N_{Reach}]$ を受理する文字列オートマトンが得られる．

(5) 初期状態を含む遷移規則 $[X_s]S_s \rightarrow REG_s$ で，REG_s に存在する出力状態以外の状態を ϵ で置換する．$[X_s]S_s$ が出力状態ならば $[X_s]S_s(REG_s)$ が表す文字列集合を認識する文字列有限オートマトン A_c を，そうでなければ REG_s が表す文字列集合を認識する文字列有限オートマトン A_c を出力する．

削除される状態は $N_{unReachLoop}$ に属する状態である．

よって，文字列オートマトン A_c は $L[n \leftarrow \epsilon \mid n \in N_{allA} \cup N_{unReach} \cup N_{Reach} \cup N_{unReachLoop}]$ を認識する．$N_{allA} \cup N_{unReach} \cup N_{Reach} \cup N_{unReachLoop} = \overline{N_{out}}$ であるから，A_c は $L[n \leftarrow \epsilon \mid n \notin N_{out}]$ を認識する．

6. あとがき

本論文ではスキーマで定義された型と XPath 式との対応の解析手法を二つ提案した．一つ目は指定した対応を満たしかつ XPath 式が充足可能かを判定する問題を既存の充足可能性判定問題に帰着し，考え得る対応のすべての組合せについて判定問題を繰り返し解くことでしらみつぶしに対応を求める手法である．二つ目は有限木オートマトンを用いた専用のアルゴリズムであり，可能な対応の組合せを精密に解析することで解を求める手法である．

また，対応の判定問題を充足可能性判定問題に帰着し，さまざまな問合せクラスとスキーマの組合せの下で，対応の判定問題の複雑度と充足可能性判定問題の複雑度が一致することを示した．

今後の課題としては，提案した二つの解析手法を実装し，実世界においてどちらの手法が効率的であるのかを検証することが挙げられる．検証方法としては，一つ目の手法を既存の SAT ソルバを用いて実装し，二つ目の手法を並行研究で作成されたプログラムを利用して実装することで，それぞれの空間計算量や時間計算量の計測と比較を行うことが考えられる．

文　献

[1] James Clark, Steve DeRose, et al. XML Path Language (XPath) Version 1.0. *Available on: http://www.w3.org/TR/xpath*, 1999.

[2] Michael Benedikt, Wenfei Fan, and Floris Geerts. XPath Satisfiability in the Presence of DTDs. *Journal of the ACM*, Vol. 55, No. 2, pp. 25–36, 2008.

[3] Gerome Miklau and Dan Suciu. Containment and Equivalence for a Fragment of XPath. *Journal of the ACM*, Vol. 51, No. 1, pp. 2–45, 2004.

[4] 森本卓爾, 橋本健二, 石原靖哲, 藤原融. スキーマ進化に応じた XPath 問合せ変換手法における入力問合せクラスの拡張と問合せ簡単化処理の改良. 電子情報通信学会第 19 回データ工学ワークショップ, pp. C4–5, 2008.

[5] 大野敦司, 石原靖哲, 藤原融. XML スキーマで定義された型と XPath 式との対応の解析手法. 情報処理学会研究報告. 2009-MPS-75(20).

[6] Yasunori Ishihara, Takuji Morimoto, Shougo Shimizu, Kenji Hashimoto, and Toru Fujiwara. A Tractable Subclass of DTDs for XPath Satisfiability with Sibling Axes. In *In: Proceedings of the 12th International Symposium on Database Programming Languages*, pp. 68–83. Lecture Notes in Computer Science 5708, 2009.

[7] Makoto Murata, Dongwon Lee, Murali Mani, and Kohsuke

Kawaguchi. Taxonomy of XML Schema Languages Using Formal Language Theory. *ACM Transactions on Internet Technology*, Vol. 5, No. 4, pp. 660–704, 2005.