A5-4　例示操作に基づくXML問合せ推論機構

松本 明[†]　森嶋 厚行[††]　北川 博之[†††]

[†] 筑波大学 システム情報工学研究科　[††] 芝浦工業大学 工学部情報工学科　[†††] 筑波大学 電子・情報工学系

1　はじめに

XML は既に標準データフォーマットの一つとしての地位を確立しており，XML を対象とした問合せ言語や操作系が数多く提案されてきた [4]．これらでは一般に，ユーザはテキストエディタや視覚的操作系を用いて問合せを記述する．我々はこれらとは全く異なるアプローチの XML 操作系の研究開発を行っている．これは，サンプルの XML 要素に対するユーザの例示操作からシステムが問合せを推論するものである．本稿では本操作系および推論機構について説明する．

XML 問合せの推論は，次の理由により自明ではない: (1) XML は半構造データである．RDB で問合せ対象となるリレーションは完全に規則的な構造を持つが，半構造データにおいてはそうとは限らない．したがって，QBE のような単純な例示操作だけでは推論が出来ない．(2)XML は構造が複雑である．問合せの結果を比べても，RDB の場合は常にフラットなテーブルであるのに対し，XML 問合せの場合は DTD によって規定される木構造となる．DTD は場合によっては数百もの要素から構成される [7]. 本稿の推論機構を用いると，ある特定のクラスに対して，XML 問合せをシステマティックに推論可能である．

この推論は自明でないだけでなく，計算量的に困難である．XML 問合せ言語では，半構造性に対応するため一般にパス正規表現 (もしくは相当物) が使われる．ここで，XML 要素のパスを文字列を見なすと，パス正規表現を推論することは，パスの集合が表す言語を受理する有限オートマトンを求めることと言い換えることが出来る．ある言語が与えられた時，それを受理する最小の有限オートマトンを見つけることは NP-complete である [6]．一方，Active Learning を利用すれば，多項式時間で発見する事が可能であることが知られている [2]．ここで Active Learning とは，ユーザがシステムに例を与えるだけでなく，システム側からユーザに質問可能な枠組みである．本システムは，Active Learning の枠組みを利用して，パス正規表現の推論を行う．

さらに，多項式時間の推論が可能というだけでは，実用システムとしては不十分である．すなわち，本システムは推論の過程でユーザに質問を行うため「ユーザとのインタラクション回数」が決定的に重要である．簡単な問合せの推論に数百回の質問が必要なシステムは全く実用的でない．本システムでは次の手段を用いて必要な質問の回数を大幅に削減する: (1)XML 操作固有の制約を利用する．(2) 明示的な指定が容易な部分は，ユーザからの直接入力を求める．

本システムが出力する問合せは XQuery 問合せである．既に様々な XQuery 処理系 [5][9][10] が開発されているため，出力された問合せをそのまま実行可能である．本システムは次のようなアプリケーションで特に有効であると考えられる．

(A1) 容易な XML 操作系: 例示操作に基づいて XML 操作を推論するため，利用者は XML 問合せに特有の概念 (例えばパス正規表現など) を覚える必要が無い．したがって，XQuery などでは敷居が高いと感じるエンドユーザでも利用可能である．HTML を直接書くエンドユーザが多数存在することからも推測されるように，個人レベルで XML データを管理するエンドユーザが増える可能性も高い．したがって，彼らの XML 操作系として利用できる．また，ソフトウエア開発組織などにおいても，XML 操作が可能な人材の範囲を広げることが出来る．

(A2)XQuery 学習支援ツール: 利用者は，自分の意図した XML 操作を行う XQuery 問合せをすぐにチェック出来る．これは，XQuery の学習に役立つ．

(A3) 問合せ生成ウィザード: 本システムは，推論に必要な利用者とのインタラクション回数を極力減らすように設計されているため，比較的数少ないやりとりで XQuery 問合せが生成可能である．したがって，XQuery に精通している技術者に対しても，ウィザードとして利用可能である．

本稿の構成は次の通りである．2 節で XML 操作の例を示す．3 節で本操作系の説明をする．4 節で本操作系の expressive power について述べる．5 節で推論メカニズムについて説明する．6 節で実験結果を示す．7 節で結論を述べる．

2　XML 操作例

本例では The XML Benchmark Project[8] のデータを利用する．これは，インターネットオークションサイトのデータの XML 表現である．図 1 に DTD の一部を図示する．item 要素はオークションの対象となる item を表す．category は，item が属するカテゴリを示す．それぞれの item が実際にどのカテゴリに属するかは，item の子要素である incategory の cagtegory 属性に，category 要素への参照 (IDREF) を持つことにより表す．closed_auction 要素は，取引が成立した要素への参照 (itemref 要素の item 属性) と，取引の成立した値段 (price 要素) を持つ．図 2 に，データの一部を示す．この時，次の問合せを行いたいとする．

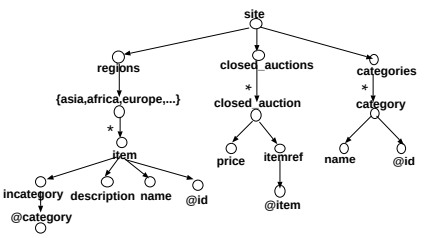

図 1: DTD の一部

問:各カテゴリごとに，Africa もしくは Europe の item のうち 300 ドル以下で取引が成立したものの名前および説明を示せ．結果は図 3 の DTD に従うものとする．

これは XQuery で図 4 のように記述される．

Inference of XML Queries based on Examples

Akira Matsumoto[†], Atsuyuki Morishima[††], Hiroyuki Kitagawa[†††]

[†]Graduate School of Sys. and Info. Eng., Univ. of Tsukuba

[††]Dept. of Info. Sci. and Eng., Shibaura Inst. of Tech.

[†††]Inst. of Info. Sci. and Elec., Univ. of Tsukuba

```
<categories>
 <category id="c1"><name>computer</name></category>
 <category id="c2"><name>book</name></category> ...
</categories>
<closed_auctions>
 <closed_auction><price>2000</price>
 <itemref item="i1"/></closed_auction> ...
 <closed_auction><price>700</price>
 <itemref item="i6"/></closed_auction>
 <closed_auction><price>50</price>
 <itemref item="i7"/></closed_auction> ...
 <closed_auction><price>100</price>
 <itemref item="i10"/></closed_auction> ...
</closed_auctions>
<regions>
 <africa>
  <item id="i1"><name>MZ70X</name>
  <incategory category="c1"/>
  <description>8bit CPU</description></item>
  ...
 </africa>
 <europe>
  <item id="i6"><name>Encyclopedia</name>
  <incategory category="c2"/>
  <description>New edition</description></item>
  <item id="i7"><name>Potter</name>
  <incategory category="c2"/>
  <description>Best Seller</description></item>
  ...
 </europe>
 <asia>
  <item id="i10"><name>XML Bookt</name>
  <incategory category="c2"/>
  <description>how-to book</description></item>
  ...
 </asia>
</regions>
```

図 2: データ例

```
<!ELEMENT i_list (category*)>
<!ELEMENT category (cname, item*)>
<!ELEMENT item (iname, desc)>
```

図 3: 結果の DTD(#PCDATA は省略)

3 例示操作を用いた XML 操作系

本節では例示操作を用いた XML 操作系について説明する．アーキテクチャを図 5 に示す．操作の流れは次のようになる．(1) ユーザが，結果として欲しい XML の DTD をシステムに入力する．(2)DTD に基づき，システムが結果のテンプレートを作る．(3) ユーザは，問合せ対象となる XML 文書の中から，いくつか要素を例として選び，それらをテンプレートにドラッグアンドドロップすることによって，結果として欲しい XML 文書の一部分を示す．(4) システムが，ユーザに質問を行いながら問合せを推論する．(5) システムは最後に問合せを出力する．

具体例として 2 節の操作を説明する．まず，図 3 の DTD を入力する．すると画面に，操作対象となる XML 文書を表示するブラウザと，結果のテンプレートが現れる (図 6)．ユーザはブラウザ中の XML の要素をドラッグアンドドロップし，テンプレートを埋める．出力したい XML 文書と矛盾していなければ何でも良い．ここでは，図 6 の矢印のように埋めるとする．Potter は 300 ドル以下の Europe の item の例である．

次に，システムがユーザに対していくつかの質問を行う．これらの質問は，テンプレートに埋められた各要素 (<cname>, <iname>, <desc>) の値の，“TargetSet” を決めるために行われる．ここで TargetSet とは，結果の XML 中の要素 (もしくはその値) として，元の XML 文書のどの部分が入るかを示したものである．

```
<i_list> {
 FOR $c IN /site/categories/category
 RETURN <category>
  <cname>{$c/name}</cname> {
   FOR $i IN /site/regions/(europe|africa)/item
   FOR $o IN /site/closed_auction
   WHERE $o/itemref/@item = $i/@id
    and $i/incategory/@category = $c/@id
    and data($o/price) <= 300
   RETURN <item>
       <iname>{$i/name}</iname>
       <desc>{$i/description}</desc>
      </item>
  }</category>
}</i_list>
```

図 4: XQuery による問合せ (text() は省略)

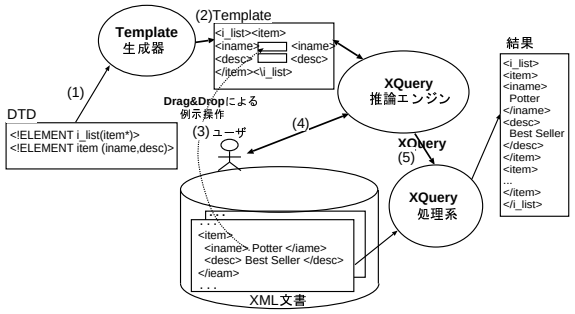

図 5: 本操作系のアーキテクチャ

この例では，<cname>要素の値の TargetSet は，元の XML 文書の category 要素の下の name 要素の値の集合である．また，<iname>要素の値の TargetSet は，africa もしくは europe 要素の下の item のうち，300 ドル以下で取引が成立した “book” カテゴリのものの名前の集合である．

TargetSet を決めるためにシステムが行う質問は，Membership Query (以下 MQ) と Equivalence Query (以下 EQ) の二種類がある．(1)MQ: 元の XML 中のある要素 (もしくは値) が，TargetSet に含まれるかどうかを聞く．ユーザは Yes もしくは No で答える．No の場合は，理由として次のいずれかを指定する．すなわち，(n1) 要素や値の位置が違うか，(n2) 位置は正しいが他の理由によって No であるか，である．図 7(a) は MQ の例である．システムは，asia の下の item の名前が，iname の TargetSet に含まれるかを尋ねている．この場合，欲しいのは europe もしくは africa の下の item の名前だけなので，ユーザは「No(位置が違う)」と答えている．(2)EQ: システムはブラウザ中の要素をハイライト表示することにより，TargetSet T の現在の推測 $\hat{T}$ をユーザに提示する．ユーザは，次のいずれかの返答をする:(1) $\hat{T} = T$ の時，[ok] ボタンを押す．(2)$\hat{T} \neq T$ の時，反例を与える．反例には正の反例 ($T - \hat{T}$ 中の要素) もしくは負の反例 ($\hat{T} - T$ 中の要素) がある．負の反例を与えるときには，MQ の No の場合と同様に，その理由を与える．図 7(b) は EQ の例である．iname の TargetSet の推測 $\hat{T}$ が表示されている．Encyclopedia は europe の下の item 要素ではあるが，300 ドルより高い値で取り引きされたので，ユーザは「NO(位置は正しい)」と答えている．

各要素の TargetSet に関する質問は，深さ優先で行われる (理由は後述する)．2 節の例では次のように，cname, iname, desc の順で行われる．

(1)cname の TargetSet を推測するための質問を行う．

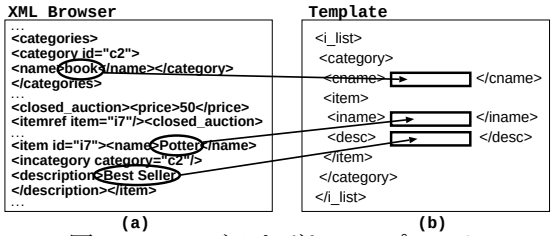

図 6: XML ブラウザとテンプレート

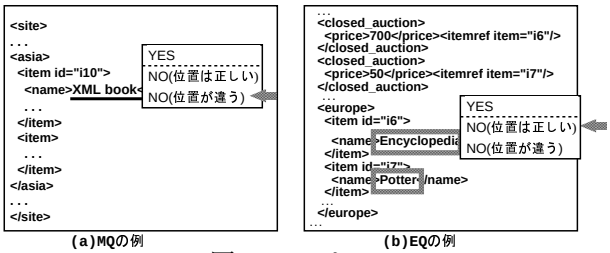

図 7: MQ と EQ

この TargetSet は純粋に要素の位置 (category の下の name 要素か否か) によって決まるので，No もしくは負の反例の理由は必ず「位置が違う」である．
(2)iname の TargetSet の推測を行う．europe もしくは africa の item のうち，300 ドルよりも高く取り引きされたものが存在すれば,「No(位置は正しい)」を指定する．すると「条件を指定せよ」という指示と共に，Condition Box が開く．ユーザは closed_auction の下に存在する Potter の price の値をそこにドラッグアンドドロップし，Condition Box に選択条件 <= 300 を入力する (図 8)．
(3)desc の TargetSet の推測を行う．

以上のように，簡単なインタラクションで図 4 の XQuery 問合せが生成されるが，その裏では複雑なメカニズムが動いている．次節以降でそれを説明する．

4 Expressive Power

我々は，XQuery を木構造の抽象表現で表す．ここではそれを XTree と呼ぶ．これは文献 [3] における ViewTree と本質的に同じであるが，次の点が異なる．すなわち，(1) 各ノードに付随する問合せが，conjunctive query でなく，複数の FOR 節と一つの WHERE 節からなる XQuery の問合せ断片であること，(2) パス変数 [1] を用いてパスの共有を表現すること，(3) 各ノードには，親要素の問合せとの差分のみを記述すること，である．パス変数の先頭には&を付け，$で始まる通常の変数 (ここでは XML ノード[1]変数と呼ぶ．以下では特に断らない限り，変数とは XML ノード変数を指すものとする) と区別する．パス変数を導入する理由は，FOR 節 (FOR v_i IN ...) で定義される XML ノード変数 v_i 間の関連を，すべて WHERE 節の述語として表現できるからである．図 4 の問合せに対応する XTree を図 9 に示す．各要素を計算するための問合せは，ルートからその要素ノードにいたるパス上の問合せ差分をまとめたものである．例えば，N1.2.2 の <desc>要素を計算する問合せは，N1, N1.2, N1.2.2 の問合せ差分の組合せである．これを N1.2.2 の完全問合せ，あるいは単に問合せと呼ぶ．一般的には，各ノードは次の情報を持つ．

$tag(output\text{-}expr)$:- FOR v_1 IN $p_1, \ldots,$ FOR v_m IN p_m WHERE $pred_1, \ldots, pred_n$

ここで FOR ... WHERE ... $Pred_n$ は完全問合せである．p_i は XPath 式である．$pred_i$ は述語である．tag は，タグ名もしくは#let である．#let は次の場合に用いられる．すなわち，あるノード n の $output\text{-}expr$ もしくは完全問合せに，そのノードの FOR 節で定義されない変数 $\$x$ が現れる時，n は子ノードに #let($\$x$) : $-Q$ を持つ (図 10)[2]．tag がタグ名であるようなノードを**タグノード**と呼び，#let であるノードを#let **ノード**と呼ぶ．

$output\text{-}expr$ は出力する値を指定する式である．タグノードの場合はオプションである．評価結果が XML のノードとなる任意の式であり，XML ノード変数を含むことが出来る．

以下では，V^n_{out} はノード n の $output\text{-}expr$ で定義される変数の集合，V^n_{for} は n の完全問合せの FOR 節に現れる変数の集合であるとする．また，V^{n*}_{for} を，n の祖先ノードの FOR 節で定義する変数の集合であるとする．ノード n の $output\text{-}expr$ がただ一つの変数 v で構成されるとき，n を simple ノードと呼び，そうでなければ，complex ノードと呼ぶ．XTree を構成するノード集合を $N = N^{simple} \oplus N^{complex}$ で表す．N^{simple} は simple ノードの集合，$N^{complex}$ は complex ノードの集合である．

```
N1:category():- FOR $cn IN /site/castegories/category{&R}/name
N1.1:cname($cn):-
N1.2:item():-
  FOR $ci IN /site/categories/category{&S}/@id
  FOR $id IN /site/regions/(europe|africa)/item{&T}/@id
  FOR $in IN /site/regions/(europe|africa)/item{&U}/name
  FOR $d  IN /site/regions/(europe|africa)/item{&V}/description
  FOR $ic IN /site/regions/(europe|africa)/item{&W}
                                        /incategory/@category
  FOR $oi IN /site/closed_auction{&X}/itemref/@item
  FOR $op IN /site/closed_auction{&Y}/price
  WHERE &R = &S and &T = &U and &T = &V and &T = &W and
        &X =&Y and $oi = $id and $ic = $ci and data($op) <= 300
N1.2.1:iname($in):-
N1.2.2:desc($d):-
```

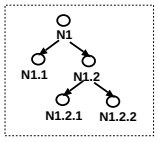

図 9: 図 4 の問合せの XTree 表現

本操作系で生成可能な問合せは，XTree で表現したとき次の制約を満たすものである．

1. パス正規表現の範囲の XPath を扱う．
2. $\forall n \in N^{simple}(V^n_{for} = V^{n*}_{for} \cup V^n_{out})$
3. $\forall n \in N^{complex}(\exists m \in N(m$ is adjacent to $n \wedge query(m) = query(n)))$．ただし，$query(n)$ は n

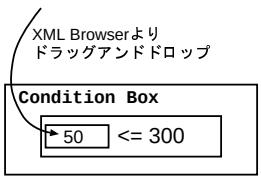

図 8: Condition Box

[1]ノードとは，document, element, attribute, text, namespace, processing instruction, comment の事である [12]．ここでは，XTree のノードと区別するため XML ノードと呼ぶ．

[2]XQuery における let 節や aggregation function の引数とは変数の使い方が異なる．

```
LET $c:=FOR $p IN /site//closed_auction/price
       WHERE $p<300 RETURN $p
RETURN <ans>count($c)</ans>


N1:ans(count($p)):-
N1.1:#let($p):- FOR $p IN /site//closed_auction/price
                  WHERE $p<300
```

図 10: `#let` が必要な問合せ例 (上) と XTree 表現 (下)

の完全問合せである．

図 4 の問合せは本操作系で生成可能である．なぜなら，図 9 の XTree を書き換えた図 11 が，以上の条件を満たすからである．

```
N1:category():- FOR $cn IN /site/castegories/category{&R}/name
N1.1:cname($cn):-
N1.2:item():-
  FOR $in IN /site/regions/(europe|africa)/item{&U}/name
  FOR $d  IN /site/regions/(europe|africa)/item{&V}/description
  WHERE &R = &S and &T = &U and &T = &V and &T = &W and
        &X =&Y and $oi = $id and $ic = $ci and data($op) <= 300
N1.2.1:iname($in):-
N1.2.2:desc($d):-
N1.2.3:#let($ci):-FOR $ci IN /site/categories/category{&S}/@id
N1.2.4:#let($id):-FOR $id IN /site/regions/(europe|africa)/
                                              item{&T}/@id
N1.2.5:#let($ic):-FOR $ic IN /site/regions/(europe|africa)
                            /item{&W}/incategory/@category
N1.2.6:#let($oi):-FOR $oi IN /site/closed_auction{&X}
                                            /itemref/@item
N1.2.7:#let($op):-FOR $op IN /site/closed_auction{&Y}/price
```

図 11: 図 9 の変形

この条件を The XML Benchmark Project の 20 の問合せに当てはめると 17 の問合せがこの操作系で生成可能である．生成不可能な 3 つの内訳は次の通りである．(Q4)BEFORE を利用している (条件 1. に違反) (Q6) 要素をグルーピングするための値が，明示的に結果に現れない (条件 2. に違反)．(Q18) ユーザ定義関数を用いている．

5 推論アルゴリズム

本推論アルゴリズムでは，次の手順で XQuery 問合せの推論を行う: (1) 生成される XQuery 問合せのベースとなる，XTree スケルトンを生成する．(2) ユーザとのやり取りを基に，各ノードの問合せを生成する．

5.1 XTree スケルトンの生成

システムは，与えられた DTD と指定された例示要素から XTree スケルトンを生成する．図 12 に，図 6 から生成された XTree スケルトンを示す．例示要素 e_i をもつ要素に対応するノードには，*output-expr* に対応する変数 v_{e_i} が入る．また，XML テンプレートに集約関数などの関数が入力された場合には，その要素のノードの下に`#let` ノード (`#let`(v_{e_i}):- ?) を追加する．ここで e_i は，関数の引数として XML テンプレートにドラッグアンドドロップされた例示要素である．それ以外の`#let` ノードは必要に応じて推論過程で生成される．各辺は，要素の multiplicity を表すラベルを持つ．これは DTD に基づき決定される．例えば要素定義が A=(B,C) の時，A は子ノードとして B,C を持ち，A-B, A-C にはそれぞれ 1 がラベル付けされる．A=B*の場合は，A-B には*が付く．A=(B|C) の場合には A は B,C を子ノードとして持ち,A-B, A-C にはそれぞれ?が付く．

1 でラベル付けされた辺のみで到達可能なノードの集合をクラスタと呼ぶ．図 12 では，C1={N1,N1.1}, C2={N1.2, N1.2.1, N1.2.2} がそれぞれクラスタである．ノード間と同様に，クラスタ間にも親子関係を定義する．C1 は C2 の親である．

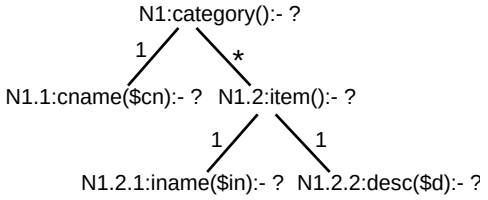

図 12: XTree スケルトンの例

TargetSet の定義 $n \in N^{simple}$ であるようなノード n があり，n は $tag(v_e) = Q(v_e)$ によって計算されるとする．また，n が所属するクラスタの先祖クラスタに属する全てのノードの例示要素の集合を E_n とする．この時，例示要素 e の TargetSet $TS(e)$ を次のように定義する．

$$TS(e) = \{v_e | Q(v_e) \wedge \forall e_m \in E_n (v_{e_m} = e_m)\}$$

すなわち，ある例示要素 e の TargetSet とは，先祖クラスタで与えられた例示要素群の文脈において，v_e が取りうる値の集合である．例えば，2 節の例の N1.2.1 では，category が book という文脈のもとで，iname が取りうる値の集合である．

5.2 問合せ推論の最上位アルゴリズム

XQuery 問合せの推論は，XTree スケルトンにおける各ノードの完全問合せの推論に帰着できる．本推論アルゴリズムでは，基本的に，深さ優先で XTree のノードを走査し，simple ノードの output 変数 v_{e_i} の TargetSet の推論を行いながら，順番に問合せを求めていく (図 14(a))．TargetSet の質問が深さ優先で行われる理由は，常に親要素の完全問合せが既知であるので，差分のみを求めれば良いからである．

本推論アルゴリズムの最上位アルゴリズムを図 13 に示す．10 行目の getTargetSet(n) は，XTree ノード n の完全問合せを求める関数である．一般的には，complex ノード n の問合せを直接求めることはできない (図 14(b)) が，n と同じクラスタに属する simple ノードがあれば，n の問合せをを求めることができる (図 14(c))．その理由は，同一のクラスタに属するノードの問合せは，全て同じものであるとみなせるからである．図 13 の 12-17 行ではその処理を行っている．図 11 では N1, N1.2 の問合せがこれによって計算される．

5.3 各ノードの問合せの推論

各完全問合せを求めるためには次の 4 つを求める必要がある．ここでは図 11 の N1.2.2 の推論を例に説明する．

(1)パス正規表現: N1.2.2 の問合せでは，まず FOR $d IN /site/regions/(europe|africa)/item/description が必要であることを推論する必要がある．

(2)他の要素との構造による関連: N1.2.2 の問合せの $d と，N1.2.1 の問合せの FOR $in IN /site/regions/

learnXTree: XTree×QUEUE→ void:	XTree のノードの内，QUEUE に入っているものの問合せを順次求める．
getTargetSet: XTree→QUERY:	ノード n の問合せを求める．
head: QUEUE→XTree:	QUEUE の先頭を得る．
drop: QUEUE×XTree→QUEUE:	QUEUE から XTree ノードを除く
query: XTree→QUERY:	n の完全問合せを求める．
combine: QUERY×QUERY→QUERY:	問合せを組み合わせる．

```
 1. queue = [ n | n is a node in XTree, in the depth-first order]
 2. XTree root = queue.head();
 3. learnXTree(root, queue);
 4.
 5. void learnXTree(XTree n, queue) {
 6.  while (queue!=[]) {
 7.   queue.drop(n);
10.   if (n.isSimple()) n.query = getTargetSet(n) else n.query=();
11.
12.   N=[n_i| n_i is a 1-edge child of n]
13.   for each n_i in N do {
14.        learnXTree(n_i,queue);
15.        n.query= n.query.combine(n_i.query);
16.        n_i.query =();
17.   }
18.   n = queue.head();
19.  }
20. }
```

図 13: 推論アルゴリズムの最上位ルーチン

(europe|africa)/item/name で計算される$in との関連は，同じ item をシェアしている事であることを推論するする必要がある．すなわち，FOR $in IN /site/regions/(europe|africa)/item{&U}/name, FOR $d IN /site/regions/ (europe|africa)/ item{&V}/description, WHERE &U = &V であることを求める必要がある．
(3)値による関連: FOR $oi IN /site/closed_auction{&X}/itemref/@item で定義される値$op と，FOR $id IN /site/regions/(europe|africa)/item/{&T}/@id で定義される値には，$oi=$id という関連があることを推論する必要がある．
(4)要素の選択条件: data($op)≤300 であることを知る必要がある．本稿の枠組みでは値による選択条件は明示的に与えられるため，推論する必要はない．

図 15 に，各ノードの問合せを推論する機構のアーキテクチャを示す．あるノード n の $tag(v_{e_i}) : -Q$ における問合せ Q を推論することを考える．上記の (1) の推論を XPath Learner(以下 X-Learner) が担当し，(2)(3) を Relationship Learner(以下 R-Learner) が担当する．より具体的にいうと，X-Learner と R-Learner が担当する部分は図 16 の通りである．X-Learner は，v_{e_i} を定義する FOR 節の XPath 式を推論する．また，R-Learner は，$pred_i$，および，最初に与えられる XTree スケルトンに存在しない#let ノードを求める．これら#let ノードと $pred_i$ は併用されて，V^n_{for} 中の変数間の関連付けを行うために利用される．

両 Learner は，推測を行うたびにユーザに対して EQ を行う．X-Learner はそれに加えて MQ も行う．ユーザの答えは次のうちのいずれかである．
(a)**Yes:** この要素 (もしくは値) は TargetSet に含まれる．
(b)**No(位置は正しい):** 位置は正しいが他の理由により TargetSet に含まれない．本操作系において例示要素 e_i の TargetSet を推測する際に,「位置が正しい」とは次のことである．すなわち，文書のルートからその指定された要素 (もしくは値) に至るパスが，X-Learner が推論しなくてはならない FOR $\$v_{e_i}$ IN P_m(図 16(b)) の XPath P_m と矛盾しないということである．
(c)**No(位置が違う):** 位置が違うため TargetSet に含まれない．

利用者からの答えが Yes の場合，両 Learner に Y が送られる．利用者からの答えが No(位置は正しい) の場合，X-Learner には Y，R-Learner には N(この要素を含まない) との情報が送られる．利用者からの答えが NO(位置が違う) の場合，X-Learner に N が送られるが，R-Learner には何も送られない．その理由は，必ずしも R-Learner の推測が間違っているとは限らないからである．

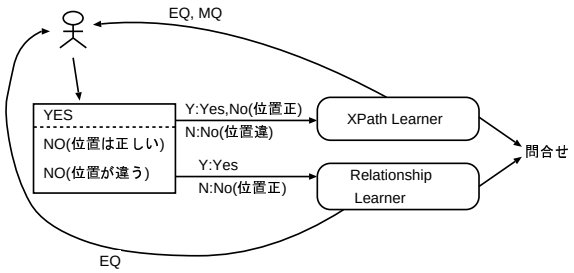

図 15: ノード問合せ推論機構のアーキテクチャ

5.4 XPath Learner

X-Learner は，最初に与えられる XTree スケルトンで，Simple ノードの *output-expr* として現れる変数のための XPath(図 16(b)) を推論する．Angluin の有限

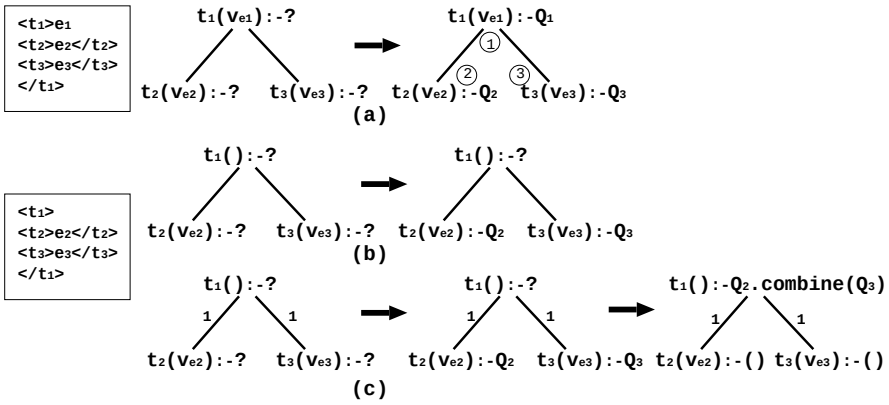

図 14: 問合せの推論順序

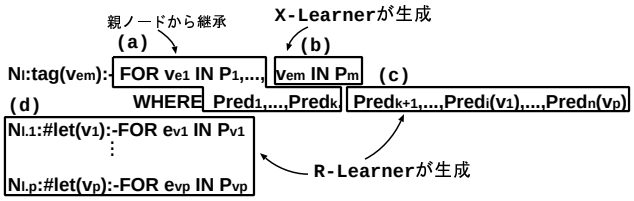

図 16: X-Learner と R-Learner の担当

オートマトン学習アルゴリズム [2] を応用した推論を行う．このアルゴリズムは，MQ と EQ に対するユーザからの返答を用いてオートマトンを導出する． 図 17 は/site/regions/asia に対応するオートマトン (図 17(d)) の導出過程である．アルゴリズムは，まず MQ を何回か行い，オートマトンの推測 (図 17(a)) を生成する．次に，これに受理される要素を XML ブラウザでハイライトし，EQ を行う．この場合は，正の反例 <site><regions><asia> を得たとする．反例をもとに推測を繰り返す．EQ の答えが OK ならば終了する．詳細は文献 [2] にある．

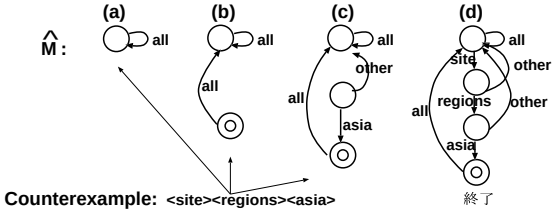

図 17: オートマトン導出過程

このアルゴリズムは多項式時間で有限オートマトンを推論出来るが，これを本操作系にそのまま適用することは実用的ではない．質問回数が膨大になるからである．そこで，質問回数を削減するために，本操作系では次のような XML 固有の性質を利用する．(A) 文字列の並びにパターンがある．したがって，例えば，DTD を用いることで，DTD に違反する MQ はユーザに聞かずに棄却できる．(B) 末尾の要素名が重要なパターンだというヒューリスティクスが利用できる．具体的には，例示要素から末尾の要素名 z を求め，末尾が z でない MQ は新たにユーザに聞かずに棄却する．なお，導出の過程において末尾の要素名 $z'(\neq z)$ である正の反例が与えられた場合には，末尾が z 及び z' でないものはユーザに聞かずに棄却するとの新たな仮説を立て，もう一度最初から推論をやり直す．

5.5 Relationship Learner

まず構造による関連と値による関連を生成するための，基本となるアイデアを示す．

構造による関連 e_1 と e_2 を Template 上の 2 つの例示要素とする．$p(e_i)$ はルート要素から e_i への要素のパスとする．e_1 と e_2 のための FOR 節として FOR $\$v_{e_1}$ IN $P(e_1)$, FOR $\$v_{e_2}$ IN $P(e_2)$ (ただし $P(e_i)$ はパス正規表現) が与えられた時，これらの「構造による関連」を求めることは，次を行うことである．すなわち，パス変数&R，&S を導入して変形し，FOR $\$v_{e_1}$ IN $P_1\{\&R\}P'(e_1)$, FOR $\$v_{e_2}$ IN $P_1\{\&S\}P'(e_2)$ WHERE $\&R = \&S$ を求めることである (ここで $P_1P'(e_1) = P(e_1)$，$P_1P'(e_2) = P(e_2)$ とする)．これは次のように求められる．まず，$P(e)$ の正規表現を，部分式の列 $r_1r_2 \ldots r_n$ で表現したもののうち，n が最大のものとする．例えば，e の TargetSet を表すパス正規表現が $abde|abdf|acde|acdf$ の時，$P(e) = r_1r_2r_3r_4 = a(b|c)d(e|f)$ である．また，ある二つの列 x_1, x_2 の最長共通 prefix を $cp(x_1, x_2)$ と表記する．この時 P_1 は次のように定義される．すなわち，$cp(P(e_1), P(e_2))$ の prefix の一つであって，次の 3 条件を満たすものである:(1)$cp(p(e_1), p(e_2))$ のある prefix q を受理する．(2)q より長い $cp(p(e_1), p(e_2))$ の prefix を受理する，$cp(P(e_1), P(e_2))$ の prefix は存在しない．(3)q を受理する，より長い $cp(P(e_1), P(e_2))$ の prefix は存在しない．

値による関連 e_1 と e_2 のための FOR 節として FOR $\$v_{e_1}$ IN $P(e_1)$，FOR $\$v_{e_2}$ IN $P(e_2)$ が与えられた時，これらの「値による関連」は次のように求められる．すなわち，e_1 および e_2 が同じ値を持てば，$\$v_{e_1}$ と $\$v_{e_2}$ の間に値による関連があるとする．この場合，FOR $\$v_{e_1}$ IN $P(e_1)$，FOR $\$v_{e_2}$ IN $P(e_2)$, WHERE $\$v_{e_1} = \$v_{e_2}$ を導出する．

本推論機構では，間接的な「値による関連」の推論も行う．ここで間接的とは，間に「構造による関連」が存在するという事である．例えば，図 4 の問合せにおける，closed_auction での値段 FOR $op IN /site/closed_auction/price と，item の名前の要素 FOR $i IN /site/regions/ (europe |africa)/item/name 間の

関連がある．これらの要素を結びつけるのは，これら自身ではなく，他の要素 (item と itemref) の属性である (図 18，c_1 と c_2)．この時，問合せ変形の結果は，FOR $id IN /site/regions/ (europe|africa)/item{&T} /@id, FOR $in IN /site/regions/ (europe|africa)/item {&U} /name, FOR $oi IN /site/closed_auction{&X} /itemref/@item, FOR $op IN /site/closed_auction{&Y} /price, FOR $i IN /site/regions/ (europe|africa)/item, WHERE &T=&U and &X=&Y and $oi=$id となる．c_1, c_2 は，内部的にはテンプレートに明示的に渡された例示要素と同様に扱われる．したがって，e_i と c_i の構造による関連は，先の方法によってシステムが自動的に導出する．

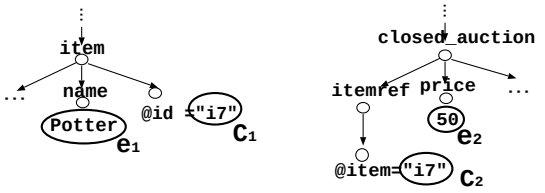

図 18: 値による関連の例

R-Learner の仕組み　R-Learner は，以上の考え方を基に，より一般的な場合に対応した方法で問合せの推論を行う．まずいくつかの定義を行う．

Data Graph: 操作対象となる XML 文書群の要素階層構造を木構造で表現し，さらに，同じ値を持つノード同士を辺（ラベルとして'=' を持つ) でつないだグラフ．

Assignment: ある変数集合 W の Assignment a^W を次のように定義する．すなわち，W に属する各変数に対して Data Graph のノードを割り当てたものである．

ノード n の正例: ある Assignment $a^{V^n_{for}}$ が，n の問合せの結果に含まれるとき，$a^{V^n_{for}}$ をノード n の正例であると呼ぶ．例えば，N1.2.1 の正例 $a^{V^{N1.2.1}_{for}}$ としては {($cn, "book"), ($in, "Potter"),($d, "BestSeller")} がある．

Participants: Data Graph g における，ある正例 $a^{V^n_{for}}$ の participants を次のように定義する．
$Participants\,(g, a^{V^n_{for}}) = E \oplus C$ ここで，E は，$a^{V^n_{for}}$ の各変数に割り当てられたノードの集合である．また C は，g において E に属するノード間を結ぶ道上に存在し，かつ，道上の辺=と隣接する全てのノード集合である (図 19)．

Known Domains: 一般に，問合せ中のパス正規表現は，何らかの意味のあるドメインを表現していると考えられる．V^n_{for} を構成するパス正規表現の集合を D とする．この時，ノード n における Known Domains KD_n を次のように定義する．

$KD_n = \{r | P \in 2^{D'}, L(r) = \cap\{L(p) | p \in P\}\}$ ただし，$D' = \{q | \exists p \in D, q$ is a subsequence of $p\}$，$L(r)$ は正規表現 r が表現する言語である．

また，正規表現 r で表現されるドメインの**派生ドメイン**を次のように定義する．すなわち，r の直後にワイルドカードなどを全く含まない単純なパス式を追加したものである．

R-Learner は，次のような順に入力を得る．まず，Template に与えられた例から，正例 $a_1^{V^n_{for}}$ を得ることができる．R-Learner は，この正例に基づいた推測

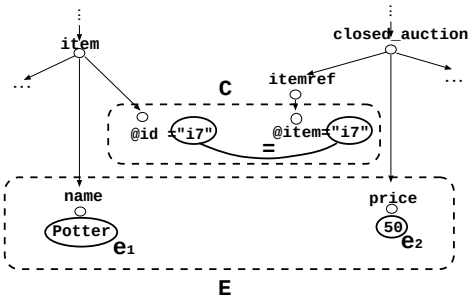

図 19: Participants

に関する EQ を行う．その後，ユーザの操作に応じて，次のどちらかの入力を得る．
(1)「Y」と共に，Condition Box に入力された，ノード n の正例 $a_i^{V^n_{for}}$ を得る．
(2)「N」と共に，明示的な選択もしくは結合条件を得る．

(1) の入力は，EQ で示された推測に対する正の反例となる．例えば，最初に与えた正例の中に，たまたま (意図しない) 値による関連が含まれていた場合，意図したものよりも厳しい制約 (関連) を持つ問合せが推測される．これに対する反例を与えることにより，意図した関連だけを生成させることができる．

(2) の入力によって，R-Learner が自動的には生成できない条件を指定することが出来る．

これらの入力は，EQ を通じて，最終的にユーザが推測された TargetSet に対して [OK] ボタンを押すまで繰り返される．

R-Learner は，これらの入力を元に，次の性質を満たす問合せ q を出力する．

- 与えられた全ての正例 $a_i^{V^n_{for}}$ と矛盾しない．
- 次の構成要素のみを用いて表現される
  - 与えられた正例から得られる Participants (C に属するもの) に対応する変数 (図 16(c) に出現，(d) で定義)
  - Known Domains もしくはその派生ドメインを表す XPath(図 16(d) での変数定義に利用する)
  - WHERE 節における次の形式の述語 (図 16(c))
    - $\$x = \$y$
    - $\&R = \&T$
    - SOME $\$v$ IN $\$x$ SATISFIES $\$v = \$y$
  - 明示的に与えられた上記以外の形式の論理式 (図 16(c))
- 上記の条件を満たす他のいかなる問合せ q_i に，q は含まれる (q is contained in any q_i's)

R-Learner のポイントは，Known Domains の定義にある．いかなる Domain の組み合わせをとっても，それらの intersection である Domain が存在することが保証されているので，生成候補となる問合せ間の包含関係は束を構成する．したがって，常にすべての正例と無矛盾な，最小の結果を持つ問合せが存在し，それを効率よく発見することが出来る．

6 実験

本操作系を用いたパス正規表現の推論に必要なインタラクション数を調べる実験を行った．データは The XML Benchmark Project のものを用いた．図 20 は，いくつかの問合せの推論に必要であった質問数である．比較のために，単純な Angluin のアルゴリズムに比べて削減された MQ の数も示す．大幅に MQ の質問数が削減されていることがわかる．

問合せ	**EQ**	**MQ**	**削減数**
/site/categories/category	1	0	725
/site//item	1	5	1192
/site//asia/item/(name|description)	2	5	1816
/site/regions//keyword	3	10	5337

図 20: MQ と EQ の回数，削減された MQ の数

7 おわりに

本稿では，例示操作に基づき XQuery 問合せを推論する機構を説明した．さらに，本機構が，XML 操作固有の性質を利用し，少ないインタラクション数での推論を可能とすることを示した．今後の課題としては，さらに広い expressive power のサポート，本アプローチの限界および推論効率の分析などがある．

参考文献

[1] S. Abiteboul, D. Quass, J. McHugh, J. Widom, and J. Wiener. The Lorel Query Language for Semistructured Data. International Journal on Digital Libraries, 1(1):68-88, April 1997.

[2] D. Angluin. Learning regular sets from queries and counterexamples. Information and Conputation, 75(2):87-106, 1987

[3] M. Fernandez, A. Morishima, D. Suciu, W. Tan. Publishing Relational Data in XML: the SilkRoute Approach. IEEE Data Engineering Bulletin 24(2): 12-19 (2001)

[4] M. Fernandez, J. Siméon and P. Wadler. XML Query Languages: Experiences and Exemplars. http://www.w3.org/1999/09/ql/docs/xquery.html.

[5] Fatdog Software. XML Query Engine. http://www.fatdog.com/.

[6] E.M. Gold. Complexity of automaton identification from given data. Information and Control 37:302-320, 1978

[7] A. Sahuguet. Everything You Ever Wanted to Know About DTDs, But Were Afraid to Ask (Extended Abstract). WebDB(Selected Papers) 2000: 171-183

[8] A. Schmidt, F. Waas, M. Kersten, D. Florescu, M. Carey, I. Manolescu, R. Busse. Why And How To Benchmark XML Databases. SIGMOD Record 30(3): 27-32 (2001)

[9] J. Siméon, B. Choi, M. Fernandez, P. Wadler. Galax. http://www-db.research.bell-labs.com/galax/.

[10] Software AG. Quip. http://www.softwareag.com/tamino/.

[11] W3C. XQuery: A Query Language for XML. X3C Working Draft, http://www.w3.org/TR/xquery, 2001.

[12] W3C. XQuery 1.0 and XPath 2.0 Data Model http://www.w3.org/TR/query-datamodel/