データベースシステム 128－12
（２００２．７．１７）

例示操作に基づく XQuery 推論系

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

概要

既存の XML 問合せ言語や操作系では，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直接記述する．我々はこれらとは全く異なるアプローチの XML 操作系である XLearner の研究開発を行っている．XLearner 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 XQuery 問合せを推論する．構造化データである RDB 等と異なり，例示操作に基づく XQuery 問合せ推論は自明ではない．XLearner は XQuery 問合せに対するシステマティックな推論機構を提供する．XLearner を用いた問合せ推論過程では，ユーザとシステムのインタラクションが必要となる．本稿では，本推論機構の概要，および，問合せ推論に必要なインタラクション数に関する実験結果について 報告する．

A System for Learning XQuery Queries based on Example Operations

Akira Matsumoto †, Atsuyuki Morishima ††, Hiroyuki Kitagawa †††
† Doctoral Program in Sys. and Info. Eng. Univ. of Tsukuba
†† Dept. of Info. Sci. and Eng., Shibaura Inst. of Tech.
††† Institute of Info. Sci. and Elec., Univ. of Tsukuba

Abstract

Existing XML query languages are textual or graphical languages in which we can specify queries for XML manipulation. This paper explains XLearner, a different kind of manipulation framework for XML. XLearner infers XQuery queries based on operations of sample XML elements. Inferring queries for XML is a non-trivial task, because XML is a kind of semistructured data, in contrast to relational databases whose data structure is completely regular. XLearner gives a systematic way to infer XQuery queries. The algorithm requires interactions between the system and the users. The paper reports an overview of its inference mechanism and experimental results on the number of required interactions.

1 はじめに

XML はデータ交換のためのデファクトスタンダードとなり，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 [3]．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 操作系である，XLearner 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 XQuery 問合せを推論するものである．

我々は，特定のクラスの XQuery に対して，XLearner との単純な質問のやりとりだけで推論を行なうアルゴリズムを開発した．また，そのアルゴリズムを拡張した枠組みが，典型的な XQuery 問合せを表現出来るだけの expressive power を持つことを確認した．また，実際に Xmark[6] や XML Use Case[7] の問合せを推論する実験を行なった．本稿では XLearner の概要，および，これらの実験結果について説明する．

XQuery 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，XQuery の場合は DTD によって規定される木構造となる． 本稿の推論機構を用いると，ある特定のクラスに対して，XQuery 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 [4]．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 [1]．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．

本稿の構成は次の通りである．(以下変更) 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節で問合せの木構造表現を導入する．また，ある問合せのクラスを説明する．5節で推論メカニズムについて説明する．6節では，5節で説明したメカニズムの拡張を行う．7節で実験結果を示す．

2 XML操作例

本例ではXmark[6]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．

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．

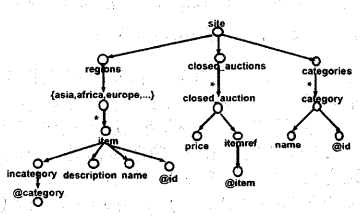

図 1: DTDの一部

3 XLearner

本節では，XLearnerにおけるXQuery問合せ推論方法について説明する．XLearnerの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．まず，結果として欲しいXMLのDTDをTemplate生成器に入力する．次に，ユーザは操作対象となるXML文書の中から，いくつか要素もしくは値 (ここでは*XML*ノードあるいは単にノードと呼ぶ) を選び，それらをテンプレートにドラッグアンドドロップする

```
<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>5000</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 Book</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)

ことによって，結果として欲しいXML文書の一部分を示す．テンプレートにドラッグアンドドロップされたXMLノードを*example*ノードと呼ぶ．テンプレート中の，XMLノードをドロップ可能な場所を，ドロップエリアと呼ぶ．

与えられたXML問合せ結果の断片に従って，推論エンジンは，ユーザが意図する問合せを推論するための質問をユーザに対して行う．そのため，ユーザをしばしば*teacher*と呼ぶ．最後に，システムはXQuery問合せを出力し，XQuery処理系に渡す．

具体例として2節の操作を説明する．まず，図3のDTDをTemplate生成器に入力すると，XLearnerは図6(b)のテンプレートを表示する．ユーザは，XMLブラウザ中のXMLの要素をいくつか選び，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のように埋めるとする．矢印はドラッグアンドドロップ操作を表している．Potterは300ドル以下で取引されたEuropeのitemの例である．

次にXLearnerは，ユーザの意図したXQuery問合せ (図4) を求めるために，ユーザに対して質問を行う．より具体的には，各exampleノードの*extent*を決定するための質問である．直感的にいうと，exampleノードeのextent (EXT_e) とは，eの*context*において，exampleが表しているノードの集合である．exampleのcontextとは，既にextentが決定されたexampleノードの集合である．例えば，extent

```
<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_auctions/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 (text() は省略)

が "book," "Potter," "Best Seller" の順番で決まったとする．この場合，$EXT_{\texttt{book}}$ の context は空であり，全ての category の名前の値の集合が $EXT_{\texttt{book}}$ となる．二番目の example の extent($EXT_{\texttt{Potter}}$) は，その context(すなわち，book カテゴリのもの) において，Africa もしくは Europe の item のうち 300 ドル以下で取引が成立した item の名前の集合となる．最後に，$EXT_{\texttt{BestSeller}}$ は，book カテゴリの Potter の description の集合 (この場合 Best Seller のみ) となる，extent を決める順番は，XLearner によって決められるため，ユーザが気にする必要はない．どのような順番で，XLearner が example ノード の extent を求めるための質問を行うかについては後述する．

質問には，*Membership Query* (以下 MQ) と *Equivalence Query* (以下 EQ) の二種類がある．

MQ:example e に対する MQ とは，XLearner は，あるXML ノード n を選び，ユーザに対して n が EXT_e に含まれるかどうかを聞く．ユーザは Y もしくは N で答える．図 7 (a) は MQ の例である．XLearner は，asia 要素の下の item の名前が，EXT_{Potter} に含まれるかを聞いている．この場合，Europe もしくは Africa の下の item の名前でないので，N と答えている．

EQ: XLearner は，XML ブラウザ中から，e の推測した extent である $E\hat{X}T_e$ の XML ノードをハイライト表示し，$E\hat{X}T_e = EXT_e$ であるか聞く．正しい場合，ユーザは [OK] ボタンを押すことで，XLearner に正しいことを伝える．正しくない場合，ユーザは反例 n を XLearner に与える．反例は，EXT_e と $E\hat{X}T_e$ の対称差に属するノードである．$n \in EXT_e - E\hat{X}T_e$ ($n \in E\hat{X}T_e - EXT_e$) の時，$n$ は正 (負) の反例という．図 7 (b) は EQ の例である．XML ブラウザ中の $E\hat{X}T_{Potter}$ がハイライト表示されている．Encyclopedia は 300 ドルより高い値で取引されたので，ユーザは負の反例であると答えている．

XLearner は，しばしばダイアログボックス (Condition Box と呼ぶ) を表示し，明示的な条件の入力を求める．どのタイミングで Condition Box が表示されるかは，XLearner によって (システマティックに) 決められる．本シナリオ中では，XLearner は EXT_{Potter} を決める過程で Condition Box を表示する．ユーザは Potter の値段の値 (closed_auction 要素の下にある) を Condition Box にドラッグアンドドロップし，選択条件 <=300 を入力する (図 8).

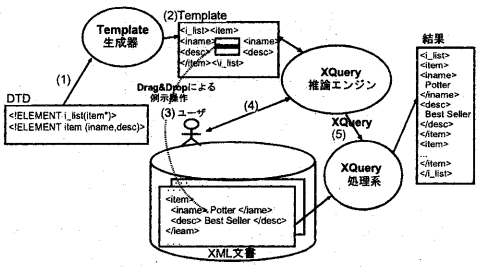

図 5: XLearner のアーキテクチャ

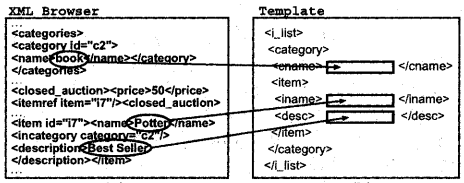

図 6: XML ブラウザとテンプレート

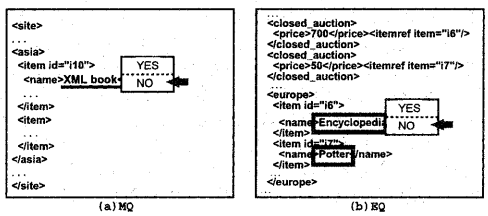

図 7: Learner の質問と答え

4 XQ-Tree

本節では，XQuery 問合せ表現の一つとして，XQ-Tree を導入する．図 4 の問合せに対応する XQ-Tree t_1 を図 9 に示す．XQ-Tree 中の各ノード n は Ni:-q_n の形式をしている．ここで，Ni はノード識別子であり，q_n は，問合せ断片 (flwr 式の形式を持つ) である．また，$q(n)$ は q_n を表すとする．

XQuery 問合せ Q が与えられた時，それに対応する XQ-Tree $XQT(Q)$ は次の手順で計算される:(1) 変数を，それに関連付けられた式で置き換えることによって，全ての let 節を取り除く，(2) 全てのノードシーケンスを返す式 (例えば，$c/name) を，等価な flwr 式 (for $cn in $c/name return $cn) に書き換える．(3) 各 flwr 式を，e を持つノードを参照しているノード識別子に置き換える．

XQ-Tree 表現は 2 つの重要な性質を持つ:(1)XML ノードのシーケンスを返す全ての式は，シーケンスのノードに変数を持っている．(2) ノード間の辺は XQ-Tree ノード間の問合せ依存関係を表している．XQ-Tree のノード n に対して，$q(n)$ を計算するためには，n の先祖の問合せが必要である．さらに，もし n が参照 (ノード識別子) を for 節もしくは where

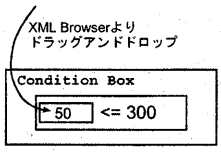

図 8: Condition Box

節に持つならば，$q(n)$ の計算には，参照している n の子孫の問合せが必要である．

意図する問合せ結果の DTD が与えられた時，XQ-Tree の各辺にラベルを割り当てることができる．辺のラベル '1', '?', '+', '*' はそれぞれ，1, 0 または 1, 1 以上, 0 以上の XML ノードが返されることを意味する．

もし，各 XML インスタンス $\mathbf{I}$ において，$q(\mathbf{I}) = Q'(\mathbf{I})$ であるならば，2 つの XQuery 問合せ Q と Q' は等価であるといい，$Q \equiv Q'$ で表す．もし，C を満たす各 XML インスタンス $\mathbf{I}$ において，$q(\mathbf{I}) = Q'(\mathbf{I})$ であるならば，2 つの XQuery 問合せ Q と Q' は C の制約の下で等価であるといい，$Q \equiv_C Q'$ で表す．同様に，XQ-Tree t と t' に対しても，$t \equiv t'$ と $t \equiv_C t'$ を用いる．ここで C は XLearner への入力に用いるインスタンスが持っている制約の集合であると仮定する．例えば，図 10 の XQ-Tree t_2 は $t_2 \equiv_C t_1$ であるような XQ-Tree である．ここで t_1 は図 9 の XQ-Tree である．t_2 においては，いくつかの `for` 節は `where` 節の `some` 式に記述されている．また，図 11 の t_3 は $t_3 \equiv_C t_2$ である XQ-Tree である．ここで，'1' でラベル付けされた辺でつながった全てのノードは，それらの問合せを結合することで，一つのノードにまとめられている (N1.1.1, N1.1.2.6, N1.1.2.7 がまとめられている)．

t 中のあるノード n に対して，$cq_t(n)$ を n の完全問合せと呼ぶ．ここで，$cq_t(n) = \oplus_{\forall m \in depends_t(n)} q(m)$ であり，$depends_t(n)$ は，t において n が依存している全ての XQ-Tree ノードの集合である．$q \oplus q'$ は問合せ q と q' の連結を表す．例えば，$cq_{t_2}(N1.1.1)$ =`for $c in /site/categories/category, $cn in $c/name return $cn`[1] である．t の完全問合せの集合は，$t' \equiv t$ であるような t' を再構成するのに十分な情報を持っている．

$q(n)$ が，`for` v `in` p `where` c `return` v の形をしている時，n は *simple* であるという．t 中の simple ノードの集合を N_t^{simple} で表す．$N_{t_2}^{simple} = \{N1.1.1, N1.1.2.1, N1.1.2.2\}$ である．もし，n が simple ならば，$cq(n)$ の `return` 節の出力と v に束縛されたノードは，1 対 1 の関係を持つ．

XQ-Tree において，変数 v に束縛された XML ノードは，常に `for` もしくは `some` 節で定義される．v が `for` v `in` p もしくは，`some` v `in` p によって定義されるとする．　この時，"v `in` p" を $expr_t(v)$ で表す．また，$domain(v)$ は式 p によって計算される XML ノードの集合を表す．$Expr_t^*(v)$ は $domain(v)$ を計算するために必要な，式の集合を表す．例えば，$Expr_{t_2}^*(v)$={`$c in /site/categories/category/, $cn in $c/name`} である．$expr_t^*(\mathrm{v})$ は，$expr_t^*(\mathrm{v})$=v `in` p とする．ここで，p は $Expr_t^*(v)$ 中の全ての式の連結である．例えば，$expr_{t_2}^*(\$\mathrm{cn})$ = `$cn in /site/categories/category/name` である．$relatedVar_t(v)$ を $Expr_t^*(v)$ 中に現れる変数の集合とする．`for` 節が v で定義されるノードを n_v とする．この時，$relatableVar_t(v)$ は n_v の先祖 (n 自身を含む) に現れる変数の集合である．$relatedVar_t(v) \subseteq relatableVar_t(v)$ である．また，もし $v' \notin relatableVar_t(v)$ ならば，$q(n_v)$ は v' と v の間の関連を指定できない．

```
N1:-        return <i_list>N1.1</i_list>
N1.1:-      for $c in /site/castegories/category
            return <category>
                      <cname>N1.1.1</cname>
                      N1.1.2
                   </category>
N1.1.1:-    for $cn in $c/name return $cn
N1.1.2:-    for $i in /site/regions/(europe|africa)/item,
            $o in /site/closed_auctions/closed_auction
            where N1.1.2.3=N1.1.2.4 and N1.1.2.1=N1.1.2.2
                and data(N1.1.2.5)<=300
            return <item>
               <iname>N1.1.2.6</iname><desc>N1.1.2.7</desc>
            </item>
N1.1.2.1:-  for $oi in $o/itemref/@item return $oi
N1.1.2.2:-  for $id in $i/@id return $id
N1.1.2.3:-  for $ic in $i/incategory/@category return $ic
N1.1.2.4:-  for $ci in $c/@id return $ci
N1.1.2.5:-  for $op in $o/price return $op
N1.1.2.6:-  for $in in $i/name return $in
N1.1.2.7:-  for $id in $i/description return $id
```

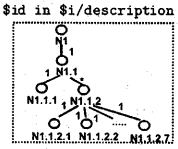

図 9: 図 4 の問合せの XQ-Tree 表現

```
N1:-      return <i_list>N1.1</i_list>
N1.1:-    for $c in /site/castegories/category
          return <category>N1.1.1 N1.1.2</category>
N1.1.1:-  for $cn in $c/name return <cname>$cn</cname>
N1.1.2:-  for $i in /site/regions/(europe|africa)/item
          where some $ic in $i/incategory/@category satisfies
          (some for $ci in $c/@id satisfies ($ic=$ci))
            and some $o in /site/closed_auctions/closed_auction,
            $oi in $o/itemref/@item, $op in $o/price,
            $id in $i/@id
            satisfies ($oi=$id and data($op)<=300))
          return <item>
             N1.1.2.6 N1.1.2.7
          </item>
N1.1.2.6:-for $in in $i/name return <iname>$in</iname>
N1.1.2.7:-for $id in $i/description return <desc>$id</desc>
```
図 10: t_2 s.t. $t_2 \equiv_C t_1$

4.1 クラス XQuery0

まず，XQ-Tree のクラス $\mathbf{XQT_0}$ を定義する．5 節で，$\mathbf{XQT_0}$ の XQ-Tree を推論するアルゴリズムの説明を行う．図 10 の t_2(四角で囲まれた部分を除く) は，$\mathbf{XQT_0}$ の XQ-Tree の例である．XQ-Tree t が，t 中の全てのノード n について，$q(n)$ (= `for` e_f `where` e_w `return` e_r) が次の 6 つの条件を満たす時，$t \in \mathbf{XQT_0}$ である．

[1] 説明の簡単化のため，タグを省略する．

```
N1:-    return <i_list>N1.1</i_list>
N1.1:-  for $c in /site/castegories/category,
        $cn in $c/name
        return <category><cname>$cn</cname>N1.1.2</category>
N1.1.2:-for $i in /site/regions/(europe|africa)/item,
        $in in $i/name,
        $id in $i/description,
        where some $ic in $i/incategory/@category satisfies
        (some for $ci in $c/@id satisfies ($ic=$ci))
          and some $o in /site/closed_auctions/closed_auction,
          $oi in $o/itemref/@item, $op in $o/price,
          $id in $i/@id
          satisfies ($oi=$id and data($op)<=300))
        return <item>
           <iname>$in</iname><desc>$id</desc>
        </item>
```

図 11: t_3 s.t. $t_3 \equiv_C t_2$

1. 全ての変数 v に対して, $expr_t^*(v)$ は, document 関数から始まるパス正規表現である.
2. 次の条件のうちいずれかを満たす:
   (a) $n \in N_t^{simple}$ または,
   (b) n は e_r 中に変数を持たず, かつ,
   - n は t のルートである. または,
   - n は '1' でラベル付けされた辺によって到着可能な子ノード m を持つ.
3. if then else 式を条件として持たない.
4. e_r は変数とタグ以外の表現を含まない.
5. sortby 式を持たない.
6. e_w は $\bigwedge_{v' \in relatableVar(v)} RS(\{v, v'\})$ である. ここで, $RS(\{v_1, v_2\})$ は v_1 と v_2 の間の関連を指定する次の表現のうちのひとつである:
   - $True$
   - $data(v_1) = data(v_2)$
   - some w in u/p satisfies $RS(w, v_2) \wedge RS(v_1, w)$. ここで u は document 関数もしくは変数であり, p は, child 軸のみで表される XPath 表現 (例えば a/b/c) である. w をこの関連の *relay node* と呼ぶ. もし, u が, $u \notin relatedVar(v_i)$ であるような変数ならば, $RS(w, v_i)$ は $True$ でない.

XQuery のクラス **XQUERY0** を次のように定義する. 次の条件が成立するときその時に限り, ある XQuery 問合せ Q が **XQUERY0** に属するという. すなわち, ある XQuery 問合せ Q' が存在し, $Q' \equiv_C Q$ かつ $XQT(Q') \in \mathbf{XQT_0}$. 図 4(Box の内側を除く) の問合せは **XQUERY0** に属する場合である.

5 XQuery 推論アルゴリズム

本節では, $\mathbf{XQT_0}$ の問合せを推論するアルゴリズムを示す. 4 節で触れたように, t の完全問合せの集合が与えられた時, XQ-Tree t を生成することができる. そのため, XQ-Tree t の推論は t 中の変数の完全問合せの集合の推論といいかえることができる. このアルゴリズムは, 2 つのステップから成る. まず, *XQ-Tree* スケルトン t_s を生成する. 次に, スケルトン t_s の全てのノード n の問合せ $q(n)$ を深さ優先で推論していく.

5.1 XQ-Tree スケルトン

与えられた DTD と example ノードから, 本アルゴリズムは XQ-Tree スケルトンを生成する. 図 12 に, 2 節の例に対する XQ-Tree スケルトンを示す. 各辺は, multiplicity を表すラベルを持つ. これは与えられた DTD に基づいて決定される. 例えば, 要素定義が A=(B,C) の時, ノード n_A は子ノードとして n_B と n_C を持ち, n_B と n_C にはそれぞれ '1' がラベル付けされる. 要素定義が A=B* の場合は, $n_A - n_B$ にはラベル '*' が付く. A=(B|C) の場合には, ノード n_A は子ノードとして, n_B と n_C を持ち, n_B と n_C にはそれぞれ '?' が付く.

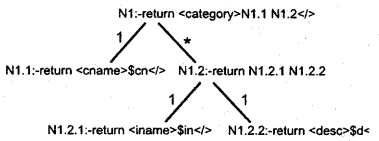

図 12: XQ-Tree スケルトン

5.2 最上位アルゴリズム

本アルゴリズムは, XQ-Tree スケルトンを深さ優先順で走査しながら, 各 *simple* ノードの問合せ $q(n_{e_i})$ を, e_i の extent を推論することで, 推論する (図 14(a)). 深さ優先で走査することで, XLearner は $cq(n_{e_i})$ と $cq(n_{e_i}$の親ノード) の "差分"(すなわち $q(n_{e_i})$) のみを推論すればよいことになる. $\mathbf{XQT_0}$ の性質から, simple でない全てのノードは, 子孫に '1' でラベル付けされた辺のみで到着できる simple ノード m を持っている. そのため, そのようなノードに対する問合せは return 節を除いて, m と同じ問合せになる. 図 13 は走査を行うための最上位アルゴリズムである. 10 行目の learnQuery(n) は, $q(n)$ を推論する関数である. 問合せは $\mathbf{XQT_0}$ に属すると仮定しているので, ドロップされた全ての example ノードは simple ノードである. 図 13 の 12-17 行では, n が simple ノードでない時の $q(n)$ を計算している. これは, n から 1 の辺のみで到着できるノード m の全ての問合せ $q(m)$ をまとめることで計算される (図 14(b)). 結果の XQ-Tree では, そのような全ての m に対して, $cq(n) = cq(m)$ (すなわち $q(m) =$'') となる.

5.3 Context と Extent

assignment を (v, n) の組の集合とする. ここで, v は変数であり, n_i は XML ノードであるとする. すると, XML テンプレートと XQ-Tree スケルトン t が与えられた時, $context_t(n)$ は assignment $\{(v_{e_i}, e_i)|$ ただし v_{e_i} は example node e_i に対応する変数であ

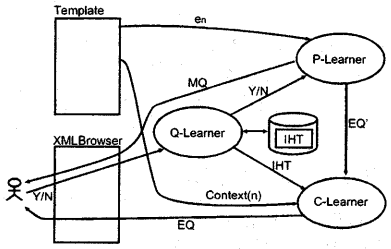

図 15: $q(n)$ の推論メカニズム

り，$q(n_{e_i})$ が既に推論済みである．} で表される．$q(n_{e_i})$ =**for** v_{e_i} **in** p **where** c **return** v_{e_i} とする．XLearner は，$context(n)$ 中のノードと，v_{e_i} に対する正の example ノード間の関連を一般化することによって c を推論する．

$context(n_e) = \{(v_{e_1}, e_1), \ldots, (v_{e_m}, e_m)\}$ とすると，$context(n)$ と EXT_n と $cq(n)$ の間の関連は，次のように表される．

EXT_{n_e}={a| a∈ $cq(n_e)$⊕ **where** v_{e_1}**is** e_1,..,v_{e_m}**is** e_m}

3 節のシナリオ中の $EXT_{n_{\texttt{Potter}}}$ は，カテゴリが "Book" であるという context における example ノード "Potter" の値の集合である．もし，context を取り除くと，$cq(n_{\texttt{Potter}})$ は，Africa もしくは Europe item のうち 300 ドル以下で取引が成立した全ての item の名前を返す．

5.4 $q(n)$ の推論メカニズム

simple ノード n の $q(n)$ を推論するためのアーキテクチャを図 15 に示す．$q(n)$ = **for** v **in** p **where** c **return** $\$v_{e_n}$[2] とする．まず，ノード e_n が P-Learner に渡される．P-Learner は MQ と EQ から，XPath p を推論するアルゴリズムを実装している．MQ は XML ブラウザを通してユーザに示される．それに対する答えは，Y か N の形式で，P-Learner に返される．Interaction recorder はその答えを *Interaction History Table (IHT)* に記録する．EQ を行なうためには条件 c が必要となるので，P-Leraner は EQ を直接 XML ブラウザに渡すことはしない．まず EQ は C-Learner に渡され，constraints c を推論する．C-Learner は EQ, IHT, $context(n)$ から，EQ EQ' を生成する．eq に対する反例は IHT に記録され，P-Learner もしくは C-Learner に渡される．これらのステップは，ユーザが EQ を受理するまで繰り返される．

5.5 P-Learner

P-Learner は MQ と EQ から，XPath p を推論する．入力 $path(e_n)$ から，P-Learner は MQ と EQ を行い，$expr^*_t(e_n)$，すなわち **document** 関数から始まる XPath を推論する．これは，Angluin のオートマトン導出アルゴリズム [1] を利用して推論を行うが，このアルゴリズムを XLearner にそのまま適用すると，ユーザとのインタラクション数が膨大になってしまう．そこで，XLearner では XML 固有の性質を利用して質問回数を削減する．特に，次に示す 2 つのルールを用いて，XLearner がユーザに対して質問する前に，自動的に棄却する．

[2] 説明の簡単化のため，タグを省略する．

```
A = {1, .., k}
while (true) {
   Let ĉ = ∧_{i∈A} x_i .
   Make an EQ with ĉ.
   If the answer of the EQ is ''OK'' {
      halt
   } else {
      let y_1, ..., y_n be the counterexample.
      A = A ∩ {i | y_i = true}
   }
}
```

図 16: k 変数単調単項式を EQ により推論するアルゴリズム

R1 DTD に矛盾するタグの並びの MQ は棄却する．

R2 正の exaple e が与えらたときに $path(e)$ の末端のタグが $t1$ だとする．この時，タグの並びの末端が $t1$ でない MQ は棄却する．これは，パターンマッチングにおいて，末端のタグが重要であるというヒューリスティクスに基づいている．もし，末端のタグが $t2(\neq t1)$ である正の example が与えられた場合には，XLearner は，正の example の末端のタグは $t1$ 及び $t2$ であるという新たな仮説を立て，もう一度最初から推論をやり直す．

5.6 C-Learner

C-Learner は **where** 節の条件 c を導出する．C-Learner は，与えられた正の example の集合に無矛盾な最も厳しい条件 c を出力する．クラス **XQuery0** 中の問合せにおいて，**where** 節の条件 c は，単調単項式と見なすことが出来る．k 変数の単調単項式の推論を，最大 k 回の EQ で推論するアルゴリズムが存在する (図 16)．

我々は本アルゴリズムの各変数に述語を対応させることにより，condition の推論を行なう．ここで問題は，c に含まれる述語の集合を求める事である．まず，e における positive assinment $PA(e)$ を $\{a_i | \exists o \in \pi_{Ex}(\sigma_{C=Y}(IHT))(a_i = \{(\$v_e, o)\} \cup Context(e))\}$ と定義する．このとき，positive assignment $a_i \in PA(e)$ に対する $cond(a_i)$ を，assignment a_i 中の example ノードに関して成立する全ての述語の集合として計算する．$cond(a_0)$ は最初の推測の述語の集合として用いる．$\{cond(a_i) | a_i \in PA(e) - \{a_0\}\}$ を，正の反例の候補の集合として用いる．以上の仕組みにより，C-Learner は全ての assignment に無矛盾な最も厳しい条件を出力する．

6 実用的な Expressive Power に向けての拡張

ユーザに次の 3 種類の明示的な指定を許すことにより，5 節で説明した仕組みの拡張を行う．

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 = learnQuery(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: 最上位アルゴリズム

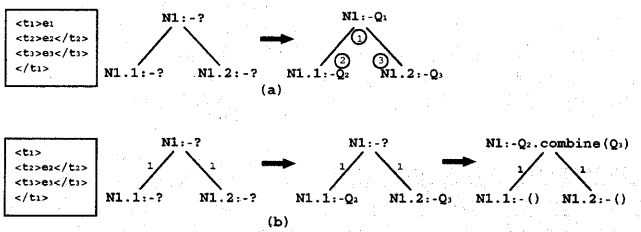

図 14: XQ-Tree における問合せの推論

ドロップエリアへの式の記述: ドロップエリアとは，ユーザが XML ブラウザ中から XML ノードをドラッグアンドドロップする場所であった．いま，ドロップエリアの概念を拡張し，式を入力することを認める．例えば，count(v) や distinct(v) のような関数を含む式を記述可能である．ここで，入れる型の概念を導入する．もしユーザが，パラメータを必要とする関数名を入力したとする．すると，XLearner は，各パラメータに対応する新たなドロップエリアを開く (図 17)．内部的には，XLearner はドロップエリアが開いたときに，テンプレート内のノードに対して，新しい子ノード[3]を生成する．distinct や union のような XML ノードのシーケンスを返す関数は，for 節に移動される．他の関数は，return 節に残る．その後，新たに作られたノードに対して，同様の推論処理が適用される．

ソートの指定: ソートのキーを指定するために，XLearner に *SortBy Box* と呼ばれるダイアログボックスを表示することを認める．SortBy Box は，ユーザがソートしたい要素の上で右クリックし，メニューを選択することにより出現する (図 18)．内部的には，XLearner は '1' でラベル付けされた辺を持つ新しい子を生成し，XQuery は，ソートのキーが新しい子ノードに一致する変数である sortby 式を持つ．

[3]辺のラベルはパラメータのタイプに従って生成される．

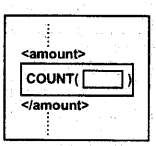

図 17: 入れ子型ドロップエリア

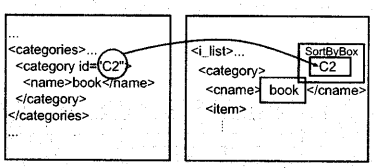

図 18: SortBy Box

明示的な選択/結合条件: これは，3節で述べた，*Condition Box* で指定する．Condition Box はユーザが負の反例を与えた時に表示される．Condition Box には Positive Condition Box (PCB) と Negative Condition Box (NCB) の二種類ある．PCB は，なぜその extent に正の example が含まれるのかを指定する

ために用いられる．NCBは，なぜそのextentに負の反例が含まれないのかを指定するために用いられる．ユーザはどちらを使用したいのか指定できる．3節ではPCBが使われている．もし，ユーザがNCBで，条件cを指定した場合，$\neg c$が条件として使用される．これは特に，ユーザが$empty(e)$関数を条件として使用したい時に有用である．なぜなら，eを満たす正のexampleはないからである．内部的には，XLearnerはCondition Boxにドロップされた XMLノードを通常のexampleノードと同様に扱う．そのため，ドロップされたオブジェクトと他のexampleの間の関連は，C-Learnerによって計算される．

このような，枠組みの自然な拡張によって，XLearnerは実用的なexpressive powerを持つようになる．XLearnerはXmarkの20の問合せ全てと，XML-Query Use Case "XMP" (Experiences and Exemplars)の12の問合せのうち11の問合せを推論が可能であることを確認した．"XMP"の問合せのうち推論できないものは，他の要素と1対1の関係を持たないような，空要素の出力を行なうものであった．

7 実験

問合せを推論する際に必要なインタラクション数を測定し，本機構が少ない回数で推論できるか調べた．データは，Xmarkの20の問合せと，XML Query Use Case "XMP" 中の12の問合せのうち，11の問合せを用いた．

	D&D(#t)	MQ	CE	CB(#t)	SB	Reduced(R1,R2,Both)
Q1	1(1)	5	1	1(3)	0	2434(2412,486,464)
Q2	1(1)	0	1	1(4)	0	2439(2416,486,463)
Q3	2(2)	0	1	1(13)	0	4878(4832,972,926)
Q4	1(1)	0	1	1(9)	0	1627(1608, 405,386)
Q5	1(2)	0	1	1(3)	0	1627(1612,405,390)
Q6	1(2)	5	0	0	0	1612(1590,405,383)
Q7	3(8)	10	0	0	0	7449(7382,1458,1391)
Q8	2(3)	0	0	0	0	2604(2573,729,698)
Q9	2(2)	0	5	0	0	4051(4023,881,853)
Q10	12(12)	0	3	0	0	26994(26756,5589,5351)
Q11	2(3)	0	1	1(5)	0	4066(4025,891,850)
Q12	2(3)	0	2	2(8)	0	4066(4025,891,850)
Q13	2(2)	10	0	0	0	4868(4822,972,926)
Q14	1(1)	5	2	1(3)	0	2426(2404,486,464)
Q15	1(1)	22	0	0	0	12637(12604,1053,1020)
Q16	1(1)	1	1	1(2)	0	2438(2422,486,470)
Q17	1(1)	0	1	1(2)	0	1177(1161,405,389)
Q18	1(2)	0	0	0	0	1627(1608,405,386)
Q19	2(2)	10	0	0	1	4848(4804,972,928)
Q20	4(8)	0	4	4(14)	0	9040(6420,2620,1532)

図 19: Xmarkの問合せの推論に必要なインタラクション数

	D&D(#t)	MQ	CE	CB(#t)	SB	Reduced (R1,R2,Both)
Q1	2(2)	0	1	1(3)	0	250(236,80,66)
Q2	2(2)	0	0	0	0	250(234,80,64)
Q3	2(2)	0	0	0	0	250(234,80,64)
Q4	2(3)	0	1	1(3)	0	250(234,80,64)
Q5	3(3)	0	1	1(3)	0	303(285,96,78)
Q6	-	-	-	-	-	-
Q7	2(2)	0	1	1(3)	1	250(236,80,66)
Q8	2(2)	0	1	1(3)	0	250(234,80,64)
Q9	1(1)	2	1	1(3)	0	26(23,8,5)
Q10	2(5)	0	0	0	0	106(98,32,24)
Q11	4(4)	0	2	2(6)	0	106(98,32,24)
Q12	2	0	1	1(10)	2	126(112,60,46)

図 20: XML Query Use Case "XMP" の問合せの推論に必要なインタラクション数

Xmarkの問合せを推論する際に必要であったインタラクション数を図19に示す．また，XML Query Use Case "XMP" の問合せを推論する際に必要であったインタラクション数を図20に示す．図において，**D&D**はドロップを行った回数である．ユーザは各ドロップにおいて，任意の式を記述することができる．そのため，その複雑さも測定する必要がある．複雑さとして，構文木表現における終点の数を用いた．終点には関数名と，数値，ドロップしたXMLノードがある．例えば，$multiply(plus(30,40),2)$における終点の数は5と計算する．括弧内の数字は終端記号の数である．**MQ**は，MQの回数であり，**CE**は，ユーザが与えた反例の回数である．ユーザは不必要な条件を導出しない適切なexampleを選ぶと想定している．そのため，**CE**は最小の反例の回数である．**CB**は，明示的な結合/選択条件を指定するために呼ばれたCondition Boxの回数である．括弧内の数字は終端記号の数である．**SB**は必要であった`SortBy` Boxの出現回数である．**Reduced**はXML固有のルールによって削減されたインタラクションの回数である．**R1**と**R2**によって削減された回数と，どちらのルールでも削減されるインタラクションの回数も示してある．

XML固有のルールによってインタラクションの回数が劇的に削減され，XLearnerが実用的なツールであるのに十分な少ないインタラクション数でXQuery問合せを推論できている．

8 おわりに

本稿では，例示操作に基づきXQuery問合せを推論する操作系XLearnerの概要および，Xmark, XQuery Use Case中の問合せに対する実験結果を説明した．実験から，実用の範囲内のインタラクション数で，問合せの推論を行うことができることを示した．今後の課題としては，推論効率の分析，GUI実装によるユーザビリティーの検証などがある．

参考文献

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

[2] P. Buneman, S. Davidson, M. Fernandez, and D. Suciu. Adding structure to unstructured data. *Proc. ICDT*, pp.336-350, 1997.

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

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

[5] J. McHugh, J. Widom, S. Abiteboul, Q. Luo, and A. Rajamaran. Indexing semistructured data. Technical report, Stanford University, Computer Science Department, 1998.

[6] 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)

[7] XML Query Use Cases. http://www.w3.org/TR/xmlquery-use-cases.