異種情報源統合のための例示による学習を用いた情報抽出システム

Information Extraction via Learning to Integrate Heterogeneous Data Sources

筒井　淳平†　　有村　博紀†

† 北海道大学大学院情報科学研究科　〒 060–0814 北海道札幌市北区北 14 条西 9 丁目
E-mail: †{tsutsui,arim}@ist.hokudai.ac.jp

あらまし　近年，ウェブページや XML データベースなどの半構造データが急増している．これらからの情報抽出および抽出された情報の統合は，利用者が問い合わせ言語を用いて記述するなど，煩雑な作業を必要とする．本稿では，半構造データからの情報抽出規則を例示からの学習によって獲得する手法を拡張し，異なる複数の情報源からの情報抽出と，これらの統合を行うシステムを提案する．

キーワード　XML, HTML, 半構造データ統合，XQuery, 例からの学習

1. はじめに

HTML ページや XML 文書などの増大を背景として，これらの半構造データを対象とした情報検索および情報統合技術が重要性を増している．XQuery に代表される半構造データ処理言語は，構造パターンと制御構造を備え，半構造データ処理の多様な場面で活用されている．しかし，その一方で構文は複雑であり，利用には高いスキルが必要である．

これに関して，2000 年前後から機械学習や半構造データ照合の技術を応用した情報検索・抽出規則の自動生成の研究が盛んになっており [3], [8], [10]〜[12]，従来の頂点抽出問合せ等 [8], [10]，情報抽出抽出と再構成問合せ等，新しい研究の動きがある [4], [9], [11], [13]．その一方で，効率よく半自動構成可能な XQuery 問合せのクラスの同定と，その実現方法はまだ明らかになっておらず，XQuery 等の複雑で強力な問合せ能力をもつ言語に対する半自動的な合成は未解決の問題である [4]．

そこで本稿では，Composition-free CoreXQuery [7] の部分クラスに対応し，構造照合，複数情報源，木の再構成を同時にもつような変換問合せのクラス [13] を考察し，これを Join を許した複数情報源に拡張する．これは，XQuery 問合せを，入力文書とのパターン照合を行う入力パターン木集合 $P = \{P_1, \ldots, P_m\}$ と，出力文書生成を行う出力パターン木 Q に分解した上で，例示を用いて入力パターン木を学習・編集し，出力パターン木の視覚的構成を支援するものである．また，質問学習の枠組みによる理論的解析を行った．

上記の情報抽出・統合システムのプロトタイプ XQUBE2 を javascript 1.5 上で実装し，ウェブブラウザ Firefox 3 の拡張機能（アドオンパッケージ）xqube.xpi として，第一著者のホームページで公開している(注1)．

(注1)：Junpei Tsutsui HP: http://www-ikn.ist.hokudai.ac.jp/~tsutsui/. アドオンパッケージは，ダウンロードしたファイル xqube.xpi を，Firefox ブラウザのウインドウにドラッグ&ドロップすることでインストールできる．詳しい使用方法は，Web ページにある README ファイルを参照されたい．

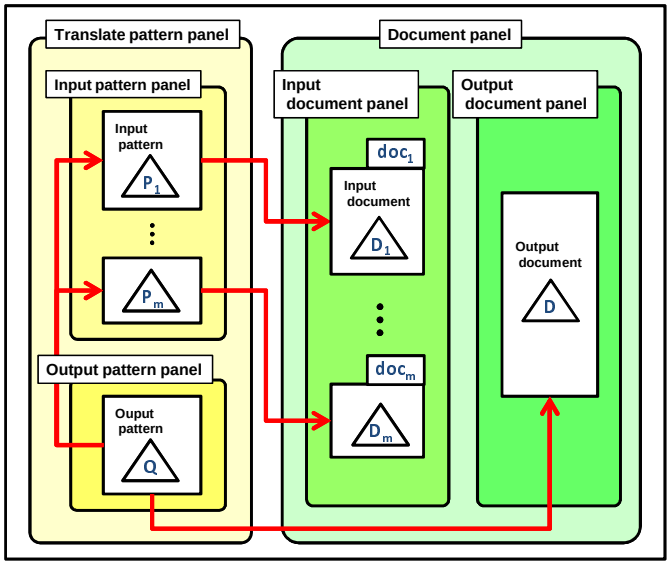


図 1　システムの概要

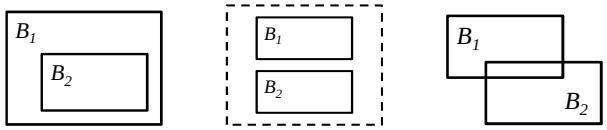
2. 情報抽出システム

2.1 システムの概要

図 1 は本稿で提案する情報抽出システムの概要を表している．**文書パネル**は入力文書パネルと出力文書パネルからなり，変換パターンパネルは入力パターンパネルと出力パターンパネルからなる．

入力パターンパネルは、内部に任意個の入力パターンを持つ．**入力パターン**はそれぞれ一つの文書プレースホルダ（図の doc_1，$\ldots$，doc_m）に対応しており，情報抽出において，対応するプレースホルダが保持する入力文書の，どの箇所を抽出するかを示している．入力パターンの抽出規則は，利用者が指定した入力文書の一部分を一般化して学習される．

出力パターンは入力パターンのそれぞれが抽出した，入力文書の一部を受け取って再構成し，出力文書とする．また，シス



(a) 包含　(b) 並列　(c) 重なり

図 2　箱の制約

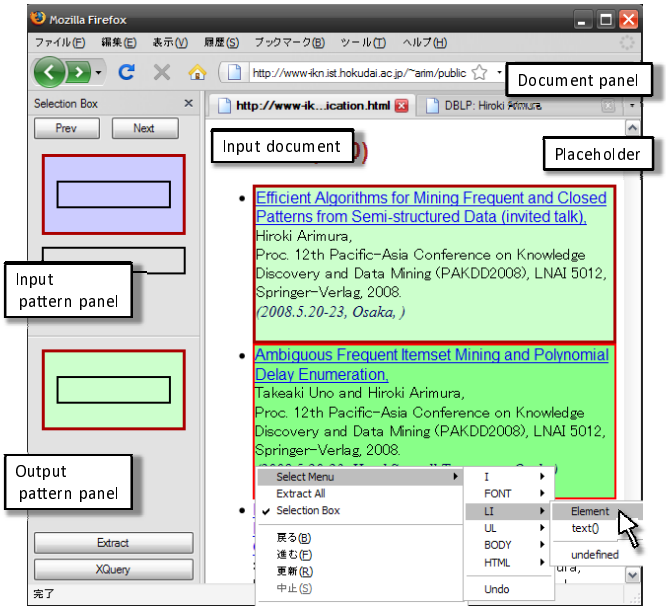


図 3　実　　装

テムは出力文書を**出力パネル**に表示する．出力パターンの構築は人手であり，学習アルゴリズムは用いていない．

2.2　木の箱表現

本システムで用いられるパターンと文書は，箱表現によって，利用者に示される．

木構造はその入れ子構造から，直接等価な平面上の箱（長方形）の組み合わせによって表現できる．作業パネル上の任意の箱を B_1, B_2 とおく．B_1 と B_2 は，包含関係または並列関係を満たし，重なり関係を許さない．

木の一つの頂点を，一つの箱で表し，木の子孫関係を，箱の包含関係によって表す．また，兄弟は並列関係によって表し，兄弟の順序は箱の位置によって表す．ここでは，箱はすべて縦に並ぶものとし，平面の上部にあるものが兄と仮定する．

2.3　実　　装

図 3 に，システムの実際のインタフェース画面を示す．左側のサイドバー上部は入力パターンパネルであり，サイドバー下部は出力パターンパネルである．また，ブラウザの一つのタブは，一つの文書を保持するプレースホルダに対応する．

利用者は，入力パターンの箱を選択したり，ブラウザの画面上で入力文書の抽出箇所を選択したりすることで，入力パターンの集合を構築する．また，入力パターンの箱を出力パターンパネルに移すことで，出力パターンを構築する．

利用者はサイドバー最下部のボタンにより，構築された出力パターンの出力文書を，新しいタブに表示することができる．

3.　準　　備

3.1　基本的な定義

自然数全体の集合を $N = \{0, 1, \ldots\}$ で表す．Σ を記号の集合とする．Σ の要素の有限列全体の集合を Σ^* と書く．二つの集合 A, B に対して，A と B の対称差 $A \triangle B = (A - B) \cup (B - A)$ と定義する．

3.2　ラベル付き木

本稿では，データやパターン等のさまざまな対象をラベル付きの根付き木で表現する．**根付き木**（rooted tree），または**木**（tree）は，根と呼ばれる親をもたない特別な頂点 $root$ を持ち，根以外のすべての頂点が唯一つの親をもつような連結な有向グラフ T である．形式的には，根付き木は，三つ組み $T = (V, E, root)$ であり，ここに，$V = \{1, \ldots, n\}$ $(n \geqq 0)$ は頂点集合，$E \subseteqq V^2$ は有向辺の集合である．もし $(x, y) \in E$ ならば，x は y の親である，または y は x の子であるという．頂点 x の子の集合を $Child(x)$ と書く．**根** $root \in V$ は，親を持たない唯一の頂点である．グラフ T は連結であり，根以外のすべての頂点 $x \in V$ は唯一の親 $parent(x) \in V$ をもつと仮定する．以後，根付き木 D の頂点集合を $V(D)$ と書き，頂点集合の要素数を $|P| = |V(D)|$ と書く．

記号のアルファベット Σ に対して，Σ 上の**頂点ラベル付きの根付き木**，またはラベル付き木（labeled tree）は，頂点ラベルをもつ根付き木 $S = (T, lab)$ である．ここに，T は根付き木であり，$lab : V(T) \to \Sigma$ は，各頂点 x にラベル $lab(x) \in \Sigma$ を割り当てるラベル付け関数である．拡張として，複数の種類のラベル付け関数を用いる場合もある．辺にラベル付けした木は，各辺 $e = (x, y) \in E$ の子 y にラベル付けすることで，頂点ラベル付き木で表せる．以後，根付き木の全体を $\mathcal{T}$ で表し，Σ 上のラベル付き根付き木の全体を $\mathcal{T}(\Sigma)$ で表す．しかし，誤解のおそれがない場合は $\mathcal{T}(\Sigma)$ を単に $\mathcal{T}$ で表すこともある．

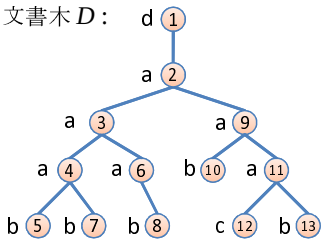
3.3　文　書　木

タグの全体集合を $\mathcal{TAG}$ で表す．HTML 文書や XML 文書などの半構造データを**文書**（document）とよび，$\mathcal{TAG}$ の文書の全体を $\mathcal{DOC}(\mathcal{TAG})$ で表す．HTML 文書や XML 文書などの半構造データ $d \in \mathcal{DOC}(\mathcal{TAG})$ は，入れ子構造の構文解析を行うことで，文書木（document tree）と呼ばれるタグを頂点ラベルにもつラベル付き木 $D = (\mathcal{T}, tag) \in \mathcal{T}(\mathcal{TAG})$ で表すことができる．ここに，$tag : V(\mathcal{T}) \to \mathcal{TAG}$ は，頂点のラベル付け関数である．HTML の DOM 木（DOM tree）は，文書木の一例である．以後，文書とその文書木を同一視する．

HTML や XML においては，**タグ名**と，**属性名**，**属性値**の全体集合 $\mathcal{N}$ と $\mathcal{A}$，$\mathcal{V}$ を仮定する．属性は属性名をタグと考えて符号化し，テキスト頂点はタグ `#text` で符号化する．形式的には，$\mathcal{TAG} = \mathcal{N} \cup \mathcal{A}$ である．**属性値対集合**は，関数を表すような属性値対の集合 $AS = \{\langle a_1, v_1\rangle, \ldots, \langle a_n, v_n\rangle\} \subseteqq \mathcal{A} \times \mathcal{V}$ $(n \geqq 0)$ である．任意の文書頂点 $x \in V(D)$ において，対応する属性値対の集合を $AS(x)$ で表す．

［例 1］　図 4 は文書木の例である．各頂点の左に，対応するタ

文書木 D：



$AS(5) = \{\langle \text{want}, \text{true}\rangle\}$
$AS(7) = \{\langle \text{want}, \text{false}\rangle\}$
$AS(8) = \{\langle \text{want}, \text{true}\rangle\}$
$AS(10) = \{\langle \text{want}, \text{true}\rangle\}$
$AS(13) = \{\langle \text{want}, \text{true}\rangle\}$

図 4　文書木の例

グ名を記している．また，文書木の右には，属性値集合を記した．

3.4　パ ス 表 現

[文献: 村上，坂本，有村 1999] に従って，ラベル木に対するパス表現を定義し，その照合演算と併合演算を導入する．**要素位置**は任意の自然数 $pos \in \mathbf{N}$ である．

要素ステップは三つ組 $\sigma = (tag, pos, AS) \in \mathcal{N} \times (\mathbf{N} \cup \{*\}) \times 2^{\mathcal{A}\times\mathcal{V}}$ であり，これを $tag[pos][AS]$ と書く．ここに，$AS \subseteqq \mathcal{A}\times\mathcal{V}$ $(n \geqq 0)$ は，属性値対の集合である．要素ステップの全体集合を $\mathcal{S}$ で表す．$pos = *, AS = \emptyset$ である場合は，それぞれ省略して要素ステップを表記する．

基本パス表現は，子供軸 / だけからなる要素ステップの有限列 $\pi = \sigma_1/\cdots/\sigma_n \in \mathcal{S}^*$ $(n \geqq 0)$ であり，**伸張パス表現**は，先頭だけに子孫軸 // をもつ基本パス表現 $\pi = //\sigma_1/\cdots/\sigma_n \in \mathcal{S}^*$ $(n \geqq 0)$ である．基本パス表現と伸張パス表現の全体集合を，$\mathcal{PATH}$ および $\mathcal{EPATH}$ で表す．要素ステップ σ が要素ステップ σ' にマッチすることを，$\sigma \sqsubseteq \sigma'$ で表す．(定義は付録を参照.)

[定義 1]（パスパターンのマッチング） 文書頂点 x に対して，基本パス表現 $\pi = \sigma_1/\cdots/\sigma_n \in \mathcal{S}^*$ $(n \geqq 0)$ で x から到達可能な文書頂点の集合を $\mathcal{P}[\pi](x)$ で表す．(定義は付録を参照.)

パスパターンのマッチングは，パス表現の間での定義可能である．伸張パス表現 π と伸張パス表現 π' に対して，π によって，π' の上端の頂点から下端の頂点に到達可能なとき，単に π が π' マッチするといい，$\pi \sqsubseteq \pi'$ で表す．

[例 2] パスパターンを $\pi = $ //a/a[1]/b[{⟨want, true⟩}] とする．このとき，図 4 の文書木について，$\mathcal{P}[\pi](1) = \{5, 8, 13\}$ である．

4.　入力パターン

入力パターンは，いわゆる木パターン [3] であり，変換パターンにおいて，入力文書木 $D \in \mathcal{T}(\mathcal{TAG})$ から一組の頂点をパス照合によって抽出するための構造制約を実現する．

4.1　入　力　木

[定義 2] 変数集合 $\mathcal{VAR}$ 上の**入力パターン木**（input pattern tree）は，(i) すべての頂点 x に相異なる変数 $var(x) \in \mathcal{VAR}$ がラベル付けされ，同時に (ii) すべての辺 $e = (x, y)$ にパス表現 $path(e) = path(y) \in \mathcal{PATH}$ が関連付けされたラベル付き根付き木 $P = (P, lab, path)$ である．

π_1 ＝ a[1]/a[1][{⟨type, tv⟩}]/a[1]/b[1][{⟨want, true⟩}]
π_2 ＝ a[2][{⟨type, radio⟩}]/a[1]/b[3][{⟨want, true⟩, ⟨rank, 1⟩}]

$\pi_1 \oplus \pi_2 = $ //a/a[1]/b[{⟨want, true⟩}]

図 5　パスの一般化の例

パスのクラス $\Pi \subseteqq \mathcal{PATH}$ と変数集合 $\mathcal{VAR}$ 上の入力木の全体集合を $\mathcal{P}(\mathcal{VAR}, \Pi)$ で表す．

文書木 $D \in \mathcal{T}(\mathcal{TAG})$ に対する入力パターン木 $P = (P, lab, path)$ のマッチングを次のように定義する．

[定義 3]（$Match(v, X)$） 任意の文書頂点集合 $X \subseteqq V(D)$ に対して，パターン頂点 $v \in V(P)\backslash\{root(P)\}$ がマッチする頂点すべての集合を $Match(v, X) = \bigcup_{w\in X} \mathcal{P}[\pi](w)$ で定義する．ただし，$\pi = path(parent(v), v)$ は頂点 v の入辺のラベルであるパス表現である．

4.2　パス表現の一般化

提案システムでは，入力パターン木のパスを例からの学習を用いて構成する．二つの要素ステップ $\sigma_i = \sigma_i[pos_i][AS_i] \in \mathcal{S}$ $(i = 1, 2)$ の一般化演算を $\sigma_1 \oplus \sigma_2$ と書く．(定義は付録を参照.)

[定義 4]（伸張パス表現の一般化） 伸張パス表現 $\pi = //\sigma^i_{n_i}/\cdots/\sigma^i_1$ $(i = 1, 2, n_i \geqq 0)$ に対して，一般化演算 $\pi_1 \oplus \pi_2$ を以下のように定義する [11]．任意の $1 \leqq j \leqq k$ に対して，$\sigma^1_j \oplus \sigma^2_j \neq *$ となる最大の非負整数を k とおく．

- $\pi_1 \oplus \pi_2 = (\sigma^1_k \oplus \sigma^2_k)/\cdots/(\sigma^1_1 \oplus \sigma^2_1)$ $(k = n_1 = n_2$ のとき)
- $\pi_1 \oplus \pi_2 = //(\sigma^1_k \oplus \sigma^2_k)/\cdots/(\sigma^1_1 \oplus \sigma^2_1)$ (それ以外のとき).

[補題 1]（Sakamoto *et al.* [10]） 任意のパス $\pi_1, \pi_2 \in \mathcal{EPATH}$ に対して，伸張パス表現の一般化 $\pi_1 \oplus \pi_2$ は，クラス $\mathcal{EPATH}$ における π_1 と π_2 の一意な**最小汎化**（LGG, least general generalization）である．すなわち，$\pi_1 \oplus \pi_2$ は，π_1 と π_2 の両方にマッチするパス表現の中で，最も具体的なパス表現となっている．

図 5 にパスの一般化の例を示す．

4.3　システムにおける入力木集合の構築

提案の情報抽出・統合システム XQUBE2 は，図 1 に示されるように，入力パターンパネルと文書パネルに，それぞれ，同数の入力パターン木と入力文書をもっており，それぞれの根が互いに対応付けられている．図 6 に，XQUBE2 システムにおける入力パターン木の学習を用いた構築手順を示す．ここに，(U) と (S) は，それぞれ，利用者とシステムの応答を表す．

文書木 D と k 頂点のパターン木 P に対して，**マッチング**とはパターン頂点を文書頂点に対応付ける関数 $\mu : V(P) \to V(D)$ ですべての $e = (x, y) \in E(P)$ に対して，$path(x, y) \sqsubseteq \pi$ が成立することを言う．ただし，π は $\mu(x)$ から $\mu(y)$ までの D 中のパスである．$\mathcal{M}(P, D)$ で，D における P の**マッチング全体の集合**を表す．関数 $f : A \to B$ の**グラフ**とは，二項関係 $G(f) = \{\,(x, f(x)) \,|\, x \in A\,\}$ をいう．関数 $f, g : A \to B$ に対し

学習手順

前提: 利用者が意図する未知の入力パターンを $P_* \in \mathcal{P}$ とする.

(1) **初期状態:** 入力パターン木集合 PIN は空である.

(2) **途中の状態:** この時点で入力パターン木集合のパネル全体に含まれている入力パターン木の集合を $PIN = \{P_1, \ldots, P_{m-1}\}$ $(m \geqq 0)$ で表す.

(a) **プレースホルダーの生成 (INIT):**
- (S) パネル PIN に根だけからなる木 P を新たに追加する.

(a) **新しい文書の読み込み (DOC):**
- 前提：初期状態または，現在対応している文書 $D = doc(P)$ 上ではパターン P による反例がなくなった場合.
- (U) パネル PIN の木 P を選択する.
- (U) URL を指定して，パネル DIN にその文書木 D を読み込み，P に D を対応付けて，$doc(P) = D$ とする.

(b) **枝の追加 (ADD):**
- (U) 枝を追加したい P のパターン頂点 v を選択する.
- (U) v がマッチする任意の文書頂点 $x = \mu(v)$ を，枝の始点として選択する．次に x の任意の子孫である文書頂点 y を枝の終点として選択する.
- (S) ここで，$\gamma = path(x, y)$ を x から y への枝のコピーとする．パターン頂点 v の子供として新しいパターン頂点 w を P に追加し，そのラベルを $path(v, w) = \gamma$ として設定する.

(c) **枝の一般化 (GEN):**
- 前提：P_* に対して入力パターン木 P が弱すぎるとき.
- (U) 学習したい入力木 P のパターン枝 $e = (v, w)$ を選択する.
- (U) 可能なマッチング μ から，v がマッチする任意の文書頂点 $x = \mu(v)$ を文書枝の始点として選択し，その任意の子孫 y を文書枝の終点として選択する.
- (S) ここで，$\gamma = path(x, y)$ を x から y への枝のコピーとする．パス表現の一般化 $path(v, w) = path(v, w) \oplus path(x, y)$ を用いて，ラベル $path(e)$ を置き換える.

図 6　入力パターン木の学習を用いた構築手順

て，$f \subseteqq g$ とは $G(f) \subseteqq G(g)$ をいう.

[定義 5]（反例） 未知の k 頂点のパターン木を P_* に対して，k 頂点のパターン木 P が**弱すぎる**とは，$\mathcal{M}(P_*, D) \backslash \mathcal{M}(P, D) \neq \emptyset$ をいい，P が**強すぎる**とは，$\mathcal{M}(P, D) \backslash \mathcal{M}(P_*, D) \neq \emptyset$ をいう．それぞれの場合に，$\mu \in \mathcal{M}(P_*, D)$ を正の反例といい，$\mu \in \mathcal{M}(P, D)$ を負の反例という.

構築手順における「利用者の意図に適合した選択」を定式化する．$X = \{v_1, \ldots, v_k\} \subseteqq V(P)$ に対して，(部分) 代入 $\mu : X \to V(D)$ と，データ頂点の k 項組 $\mu = (\mu(v_1), \ldots, \mu(v_k)) \in V(D)^*$ を同一視する．(部分) 代入 $\mu : X \to V(D)$ が，P_* に関して**保守的である**とは，$\exists \mu_* \in \mathcal{M}(P_*, D), \mu \subseteqq \mu_*$ が成立することをいう．演算 ADD と GEN の適用が P_* に関して**保守的である**とは，適用における拡張 $\mu \cup \{(w, y)\} \subseteqq \mu_*$ が保守的であることをいう．ここに，μ と $(w, y) \in V(P) \times V(D)$ は，演算の適用で選択したマッチングと頂点対である.

[補題 2]（演算 ADD と GEN の正当性） パターン木 P に操作 $op \in \{ADD, GEN\}$ を行って得られたパターン木を Q とする．このとき，$\mu \cup \{(w, y)\} \subseteqq \mu_*$ が P_* に関して保守的ならば，次が成立する.

（1） 演算 $op = ADD$ に対して，$P \sqsubseteq Q$ かつ $|P| + 1 = |Q|$.

（2） 演算 $op = GEN$ に対して，$P \sqsupset Q$ かつ $|P| = |Q|$.

（3） 演算 $op \in \{ADD, GEN\}$ に対して，$P \sqsupseteq P_*^{(k)}$ が成立する．ここに，$k = |P|$ かつ，$P_*^{(k)}$ は P_* を P に対応する頂点集合に制限したものである.

次の定理は，利用者が意図したとおりに正しく頂点を選択していったならば，入力パターン木の学習による構築が可能であることを主張する.

[定理 1] 図 6 の手順において，P_* に関して保守的な演算 ADD または GEN の適用が存在するならばそれを見つけられると仮定する．このとき，そのような適用が可能な限り選択し続けるならば，最終的に P が未知パターン P_* に一致して停止する.（ただし，マッチングは一対一であるもののみを考える.）

次の二つの質問を用いて，保守的な演算 ADD または GEN の適用が見つけられる.

[定義 6] P_* に対する拡張性質問は，ある $k \leqq |P_*|$ に対して，データ頂点の組 $\mu = (x_1, \ldots, x_k) \in V(D)^*$ を提示する．もし μ が保守的ならば"'Yes' を，それ以外ならば"No" を受け取る.

[定義 7] P_* に対する等価性質問は，現在のパターン P_* を提示する．もし任意の D に対して $\mathcal{M}(P, D) = \mathcal{M}(P_*, D)$ ならば，"Yes" を受け取り，それ以外ならば，$\mathcal{M}(P, D') \neq \mathcal{M}(P_*, D')$ となるある D' および，反例 $\mu \in \mathcal{M}(P, D') \triangle \mathcal{M}(P_*, D')$ を受け取る.

質問学習モデルにおけるアルゴリズムの正当性の詳細は，別の機会に譲る.

5. 出力パターン

5.1 出力パターン木

出力パターン木は根付き木 $Q = (Q, I)$ である．ここに，Q は根付き木であり，$I = \{var, cond, tag\}$ は，頂点のラベル付け関数の集合である．各頂点 $v \in V(Q)$ は，$cmd(v) = \tau$ のとき，τ **型の頂点**と呼ぶ.

出力パターン木 Q は根 $v \in V(Q)$ の種類によって，次のように分類される.

- `for` 頂点 v は，変数 $x = var(v) \in \mathcal{VAR}$ をラベルにもち，1 個の子 Q_1 をもつ．木全体を $Q =$ `for` x Q_1 で表す.
- `let` 頂点 v は，変数 $x = var(v) \in \mathcal{VAR}$ をラベルにもち，1 個の子 Q_1 をもつ．木全体を $Q =$ `let` x Q_1 で表す.
- `where` 頂点 v は，一つまたは二つの変数を含む $x = c$ または $x = y$ の形をした条件式 $\phi = cond(v)$ をラベルにもち，1 個の子 Q_1 をもつ．木全体を $Q =$ `where` cond Q_1 で表す.

Procedure Eval$[Q](E)$
input: Q: 出力パターン木, E: 環境;

```
1: Eval[for x Q](E) =
2:     foreach v ∈ P[path(x)](E(parent(x))) do
3:         S := S ∪ Eval[Q](E ∪ {(x, v)});
4:     return S;

5: Eval[where cond Q](E) =
6:     if Eval[cond](E) == ()
7:     then return Eval[Q](E);
8:     else return ();

9: Eval[value x](E) =
10:    return E(x);

11: Eval[const tag Q](E) =
12:    return tag(Eval[Q](E));

13: Eval[seq Q1 ··· Qn](E) =
14:    return Eval[Q1](E) + ··· + Eval[Qn](E);
```

図 7　出力パターン木 Q と，環境 E，入力パターン木集合 P から，出力文書木を計算する再帰手続き Eval

- `value` 頂点 v は, 変数 $x = var(v) \in \mathcal{VAR}$ をラベルにもち，0 個の子をもつ．木全体を $Q =$ `value` x で表す.
- `const` 頂点 v は, タグ $t = tag(v) \in \mathcal{TAG}$ をラベルにもち，1 個の子 Q_1 をもつ．木全体を $Q =$ `const` tag Q_1 で表す.
- `seq` 頂点 v は, 任意個の子 $Q_1, \ldots, Q_n$ をもつ．木全体を $Q =$ `seq` $Q_1 \cdots Q_n$ で表す.

木の自由変数を，`for` と `let` を限量子（quantifier）と考えて，通常の論理式と同様に定義する [6]．トップレベルの環境 E では，出力パターン木のすべての自由変数は入力文書木いずれかの根に束縛されるとする．

提案の問い合わせ木 $F = (P, Q)$ は，任意個数の入力パターンの集合 $P = \{P_1, \ldots, P_m\}$ を用いて，複数のソース文書を同時に扱える．この機構を用いて，XQuery における結合 (Join) を実現できる．

5.2　木問合せの意味

正整数を m とおく．このとき，入力文書木集合 $D = \{D_1, \ldots, D_m\}$ に対して，木問合せ $F = (P, Q)$ が計算する出力文書木 F を定義する．$V(F) = \cup_i V(P_i)$ を P に出現する変数全体の集合とする．

環境は変数名に文書頂点を割り当てる関数 $E : \mathcal{V} \to V(P)$ である．

図 7 に，出力パターン木 Q と，環境 E，入力パターン木集合 P を受け取り，出力文書木を計算する再帰手続き Eval を示す．トップレベルでは，Eval は，すべての入力パターンに対して，その根と対応する入力文書木の根を対応付けた環境を受け取る．ここで，() で木の空列を表し，$T_1 + T_2$ で木の列 T_1 と T_2 の連結を表す．P 中のパターン木の頂点と変数を区別しない．したがって，$parent(x)$ は変数 x の親である変数を表し，$\pi = path(x)$ で辺 $(parent(x), x)$ のラベルであるパス表現を表す.

前提： 入力パターン木集合 $P = \{P_1, \ldots, P_m\} \subseteqq \mathcal{P}$ は既に構築されている.

(1) **初期状態：** 出力パターン木 Q は，根のみからなる木である.

(2) **途中の状態：** この時点で構築されている出力パターン木を Q とする.

(a) `for` **型または** `value` **型頂点の追加:**

- (U) 追加するコマンド型 $\tau \in \{$`for`, `value`$\}$ を選択し，子頂点を追加したい出力パターン木 Q の頂点 v を選択する．システムに P の頂点 x を候補として提示させる.
- (S) 入力文書上のマッチングを列挙しながら，x の候補を提示する．このとき，入力パターン中の親変数 $y = parent_P(x)$ が，v で束縛されているかを検査する.
- (U) 提示された適当な入力パターンの頂点 x を選択する．もし適当な頂点がなければ，入力パターンパネルに戻り，前節で述べた方法で頂点 x と，それに関連するパス表現 $path(x)$ を新しく生成する.
- (S) Q に，v の子供として新しい τ 型頂点 w を追加し，その変数を $var(w) = x$ に設定する.
- v の部分木の上に `for` 型頂点を挿入する場合は，下記 (c) の (S) と同様に部分木を移動する.

(b) `where` **型頂点の追加:**

- (U) 枝を追加したい出力木 Q の頂点 v を選択する.
- (S) Q に，v の子供として新しい `where` 型頂点 w を追加する.
- (U) 条件式が"x *op* y"型の場合には，二項演算子 *op* と，v で束縛された二つの入力変数 x, y を選択し，条件式 $cond(v) = (xopy)$ を設定する．条件式が"x *op* c" 型の場合には，二項演算子 *op* と，v で束縛された入力変数 x を選択し，条件式 $cond(v) = (xopc)$ を設定する.
- v の部分木の上に `where` 型頂点を挿入する場合は，下記 (c) の (S) と同様に部分木を移動する.

(c) `const` **型頂点の追加:**

- (U) 枝を追加したい出力パターン木 Q の頂点 v とその部分木の一つ以上 R を選択する．タグ名 `tag` $\in \mathcal{TAG}$ を選択する.
- (S) 出力パターン Q において，v の子供として新しい `const` 型頂点 w を追加し，タグを $tag(v) =$ `tag` に設定する．部分木 R を v の下から w の直下に移動する.

図 8　出力パターン木の直接操作による構築手順

5.3　出力パターン木の構築手順

図 8 に，システムにおける出力パターン木の構築手順を示す．(U) と (S) は，それぞれ，利用者とシステムの応答を表す.

構築において導入されるすべての変数は束縛されている必要がある．これは，図 7 のアルゴリズム Eval が出力文書木を正しく計算するための条件である．出力パターン木 Q の任意の頂点を v とおく．入力パターン中の変数 x が，v で**束縛されている**とは，Q において，x をラベルにもつような `for` 型または `let` 型頂点が，v の祖先に存在することをいう.

6.　おわりに

本稿では，半構造データの変換問合せの例からの構築システムについて考察した．入力パターン木集合と出力パターン木の

組からなる Composition-free CoreXQuery [7] の部分クラスに対応する変換問合せ [13] を考察し，例示による学習と視覚化環境を用いた問合せ構築支援のプロトタイプシステムを開発した．

機械学習では，データと反例を一般化し，隠された未知パターンの構造や機能を推定する．今後の課題として，視覚や対話からの情報抽出や情報統合の理論的モデルを導入して，データ表現やプロトコルのより詳細な理論的解析を行いたい [3]．特に，帰納論理プログラミングにおける逆導出法や飽和法，能動質問による枝刈り法等は有望だと考えられる [1], [2], [5] ．今後の課題としたい．

文　　献

[1] Marta Arias, Roni Khardon, Learning Closed Horn Expressions, Inf. Comput., 178(1), 214–240, 2002.

[2] Hiroki Arimura, Hiroshi Sakamoto, Setsuo Arikawa, Efficient Learning of Semi-structured Data from Queries,, Proc. the 12th International Conference on Algorithmic Learning Theory (ALT'01), LNAI 2225, 315–331, 2001.

[3] H. Arimura, H. Sakamoto, S. Arikawa, Efficient Learning of Semi-structured Data from Queries, Proc. ALT'01, LNAI 2225, 315-331, 2001.

[4] D. Braga, A. Campi, and S. Ceri, XQBE (XQuery By Example): A Visual Interface to the Standard XML Query Languages, ACM TODS, 30(2), 398–443, 2005.

[5] C. David Page Jr., Alan M. Frisch, Learning Constrained Atoms. Machine Learning, 427-431, 1991.

[6] Christoph Koch, On the complexity of nonrecursive XQuery and functional query langauges on complex values, Proc. ACM PODS'05, 2005.

[7] Christoph Koch, On the role of composition in XQuery, Proc. the Eighth International Workshop on the Web and Databases (WebDB'05), June 16–17, Baltimore, 2005.

[8] N. Kushmerick, Wrapper induction: Efficiency and expressiveness, Artif. Intell., 118(1-2): 15-68, 2000.

[9] Atsuyuki Morishima, Hiroyuki Kitagawa, Akira Matsumoto, A Machine Learning Approach to Rapid Development of XML Mapping Queries, ICDE 2004, 276-287.

[10] H. Sakamoto, Y. Murakami, H. Arimura, S. Arikawa, Extracting partial structures from HTML documents, Proc. 14th Int'l FLAIRS Conf., 264-268, AAAI, 2001.

[11] 筒井, 伊藤, 有村, リンク情報と構造情報を用いた HTML 群からのテキスト自動切り出しアルゴリズムの実装, DEW'07, 2007.

[12] 筒井, 有村, ラベル付きグラフからのウォークの多項式時間学習, 電子情報通信学会, コンピュテーション研究会, COMP, 2008.

[13] 筒井，有村, XQUBE: 具体例と演示からの XQuery 問合せ構築のための視覚言語, iDB フォーラム 2008, 情処 DBS 研，2008.

[14] W3C, XQuery 1.0: An XML Query Language, W3C Recommendation 23 January 2007.
http://www.w3.org/TR/2007/REC-xquery-20070123/

付　　録

1. 基本的な定義

1.1 パス表現

[文献: 村上，坂本，有村 1999] に従って，ラベル木に対するパス表現を定義し，その照合演算と併合演算を導入する．

[定義 8]（要素ステップのマッチング） 要素ステップ $\sigma_i = tag_i[pos_i][AS_i] \in \mathcal{S}$ $(i = 1, 2)$ に対して，σ_1 が σ_2 に**マッチする**（$\sigma_1 \sqsubseteq \sigma_2$）とは，次が成立することをいう：(i) $tag_1 = *$ または $tag_1 = tag_2$ が成立する．(ii) $pos_1 = *$ または $pos_1 = pos_2$ が成立する．(iii) $AS_1 \subseteqq AS_2$ が成立する．

[定義 9]（要素ステップと文書頂点のマッチング） 要素ステップ $\sigma \in \mathcal{S}$ が，文書頂点 $x \in V(D)$ にマッチする（$\sigma \sqsubseteq x$）とは，$\sigma \sqsubseteq tag(x)[pos(x, tag)][AS(x)]$ を言う．ここに，$pos(x, tag)$ は，タグが tag に適合する兄弟の内での x の先頭からの順位である．

パス表現 π に対して，その照合関数 $\mathcal{P}[\pi] : V(D) \to 2^{V(D)}$ を，次のように導入する．

[定義 10]（ステップのマッチング） 文書 D の頂点 x において要素ステップ $\sigma \in \mathcal{S}$ で到達可能な頂点集合 $\mathcal{P}[\sigma](x) \subseteqq V(D)$ を，(i) $y \in Child(x)$, かつ (ii) ステップ σ は頂点 x にマッチすることを満たす文書頂点 $y \in V(D)$ のなす集合と定義する．

集合値関数 $f : A \to 2^A$ と集合 $X \subseteqq A$ に対して，$f(X) = \bigcup_{x \in X} f(x)$ と定義する．

[定義 11]（パスパターンのマッチング） 文書頂点 x に対して，基本パス表現 $\pi = \sigma_1/\cdots/\sigma_n \in \mathcal{S}^*$ $(n \geqq 0)$ で x から到達可能な文書頂点の集合を $\mathcal{P}[\pi](x) = \mathcal{P}[\sigma_n](\cdots(\mathcal{P}[\sigma_1](x))\cdots) \subseteqq V(D)$ と定義する．これは，要素ステップによる照合関数の合成である．

1.2 パス表現の一般化

[定義 12] $\sigma_i = (tag_i, pos_i, AS_i)$ $(i = 1, 2)$ について，一般化演算 $\sigma_1 \oplus \sigma_2$ を次のように定義する:

（1） $tag_1 = tag_2$ のときは $tag_1 \oplus tag_2 = tag_1$ とし，それ以外は $tag_1 \oplus tag_2 = \bot$ とする．

（2） $pos_1 = pos_2$ のときは $pos_1 \oplus pos_2 = pos_1$ とし，それ以外は $pos_1 \oplus pos_2 = *$ とする．

（3） $AS_1 \oplus AS_2$ は，次を満たす属性値対集合：任意の $a \in \mathcal{A}$ に対して，$AS_1(a) = AS_2(a)$ ならば $(AS_1 \oplus AS_2)(a) = AS_1(a)$ とし，それ以外は $(AS_1 \oplus AS_2)(a) = *$ とする．

（4） $tag_1 \oplus tag_2$ が定義されるとき，$\sigma_1 \oplus \sigma_2 = (tag_1 \oplus tag_2, pos_1 \oplus pos_2)$ とし，それ以外は $\sigma_1 \oplus \sigma_2 = \bot$ とする．