社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

半構造データからの効率よい無順序木パターン発見手法

浅井　達哉† 　有村　博紀† 　宇野　毅明†† 　中野　眞一†††

† 九州大学大学院システム情報科学府・研究院　〒 812–8581 福岡市東区箱崎 6–10–1
†† 国立情報学研究所　〒 101–8430 東京都千代田区一ツ橋 2–1–2
††† 群馬大学工学部　〒 376-8515 桐生市天神町 1–5–1
E-mail: †{t-asai,arim}@i.kyushu-u.ac.jp, ††uno@nii.ac.jp, †††nakano@msc.cs.gunma-u.ac.jp

あらまし　本稿では，XML 文書に代表される半構造データからのデータマイニング問題を考察する．我々は，半構造データのモデルとしてラベルつき無順序木を採用し，与えられた半構造データの集積から出現頻度の高い部分構造を発見するアルゴリズム UNOT を開発した．このアルゴリズムは，逆探索に基づいて無順序木パターンを高速に列挙し，各パターンの出現リストを漸増的に計算することにより，パターン 1 つあたり $O(kb^2m)$ 時間ですべての頻出無順序木パターン T を計算する．ここに，k は T の大きさであり，b はデータ木の最大枝分かれ数，m は T のデータ木への総出現数である．

キーワード　半構造データマイニング，グラフマイニング，頻出パターン発見，ラベルつき無順序木，逆探索

An Efficient Algorithm for Mining Frequent Unordered Trees from Semi-structured Data

Tatsuya ASAI†, Hiroki ARIMURA†, Takeaki UNO††, and Shin-ichi NAKANO†††

† Department of Informatics, Kyushu University　Fukuoka 812–8581, Japan
†† Natinal Institute of Informatics　Tokyo 101–8430, Japan
††† Faculty of Engineering, Gunma University　Gunma 376–8515, Japan
E-mail: †{t-asai,arim}@i.kyushu-u.ac.jp, ††uno@nii.ac.jp, †††nakano@msc.cs.gunma-u.ac.jp

Abstract　In this paper, we study a data mining problem of discovering frequent substructures in a large collection of semi-structured data, where both of the patterns and the data are modeled by labeled unordered trees. The keys of the algorithm are efficient enumerating all unordered trees and incrementally computation of the occurrences based on a powerful design technique known as the *reverse search*. We present an efficient algorithm called UNOT that computes all labeled unordered trees appearing in a collection of data trees with frequency above a user-specified threshold. We prove that the algorithm enumerates each frequent pattern T in $O(kb^2n)$ per pattern, where k is the size of T, b is the branching factor of the data tree, and n is the total number of occurrences of T in the data trees.

Key words　semi-structured data mining, graph mining, frequent pattern discovery, labeled unordered trees, reverse search

1. はじめに

データマイニング（Data mining）とは，データベースに蓄積された大量のデータから，自明でない規則性やパターンを半自動的にとりだす方法についての科学的研究である．データマイニングは，現在，ビジネス分野や科学技術分野などのさまざまな対象分野で，その適用が盛んに行われている [3].

一方，高速なネットワークと安価な大容量記憶装置の発達によって，ウェブページや XML 文書に代表される半構造データ (Semi-structured data) がネットワーク上に大量に蓄積されており [2], [21]，大規模半構造データを対象としたデータマイニング（半構造データマイニング）に対する要求が高まっている．しかし，半構造データは関係データベースとは違い，明示的な構造をもたないので，関係データベースを対象とした従来のデータマイニング手法を半構造データに直接適用することは困難である．そのため，大規模半構造データに対する高速なマイニング手法の開発が急務となっている．

また，XML 文書やウェブページのタグ構造だけでなく，インターネットのリンク構造や，ウェブサイト構造とそのアクセスログ，有機化合物，遺伝子ネットワークなど，従来の関係デー

タとして扱えない非定形・非正規形データも多い．これらの非定型データの多くは，個々のデータを表す頂点を，それらの関係を表す辺で結合したラベルつきの木やグラフとして自然にモデル化できる．そのため，これらの大規模グラフデータからのデータマイニングが関心を集めている．

半構造データマイニングは，Holder らの部分構造発見アルゴリズム SUBDUE [10] や吉田らの GBI [24]，Nestrov らのスキーマ抽出手法 [16]，Wang らの頻出パス列発見手法 [22] など，1990 年代に研究が始まった．その後，2000 年頃から，結合規則発見のグラフデータへの拡張として，頻出部分構造を発見する問題を中心に論文数が増え，盛んに研究が行われている [1], [6]〜[8], [11]〜[13], [18], [20], [22], [23], [25].

本稿では，与えられたラベルつき無順序木の集積に頻出するすべての部分構造パターンを，効率よく計算するアルゴリズム UNOT を提案する [8]．**ラベルつき無順序木**とは根つき閉路なし有向グラフである．また，根以外のすべての節点は唯一の親節点をもち，すべての節点にはラベルが割り振られる．ラベルつき無順序木は，半構造データマイニング [1], [6], [7], [13], [16], [22], [25] で盛んに扱われているラベルつき順序木の一般化とも，グラフマイニング [10]〜[12], [20], [23] で扱われる一般グラフ構造の制限ともみなせる．

木やグラフ構造のマイニングは，グラフ同型判定やパターンの出現位置の計算など，計算量的にハードな問題を含む．我々は，無順序木の正規形表現を導入し，すべての無順序木の正規形表現を重複せずに列挙することにより，無順序木パターンにおけるグラフ同型問題を回避する．これは，計算困難な列挙問題に対するアルゴリズム構成法として知られる**逆探索**（reverse search）[9], [19] の適用とみなせる．また，パターンの出現として埋め込み出現を定義し，これを漸増的に計算することにより，無順序木パターンの出現位置を効率よく計算する．

我々は，以上の手法を組み合わせて，与えられた無順序木の集積に頻出するすべての無順序木パターンを効率よく計算するアルゴリズム UNOT を開発した．このアルゴリズムは，パターン 1 つあたり $O(kb^2m)$ 時間ですべての頻出パターン T を計算する．ここに，k は T の大きさであり，b はデータ木の最大枝分かれ数，m は T のデータ木への総出現数である．

無順序木を対象としたデータマイニングの研究は，ほとんど行われていない．Termier ら [18] は，半構造データからの頻出無順序木パターン発見問題を考察し，これを解くアルゴリズム TreeFinder を開発した．しかし，TreeFinder はすべての頻出パターンを発見するとは限らず，重要なパターンを計算し損ねる可能性がある．それに対して，我々のアルゴリズム UNOT は，すべての頻出パターンをもれなく計算する点で，TreeFinder とは異なる．非常に最近，Nijssen ら [17] が同じ問題を考察し，我々とは独立に，UNOT とよく似た効率よいアルゴリズムを与えている．特に，無順序木パターンの列挙手法は，UNOT とまったく同一である．一方，パターンの出現の計算法は UNOT とは異なる．

本稿の構成は以下のとおりである．2 節で必要な定義と我々のデータマイニング問題を与える．3 節で無順序木の正規形を導入する．4 節で頻出無順序木パターン発見アルゴリズム UNOT を説明する．最後に，5 節で本稿をまとめる．

2. 準　　備

本節では，無順序木と順序木，照合写像等の基本的定義を与え，本稿のデータマイニング問題である頻出無順序木発見問題を導入する．集合 A に対して，A の大きさを $|A|$ であらわす．A 上の二項関係 $R \subseteq A^2$ に対して，R の反射推移閉包を R^* と表記する．

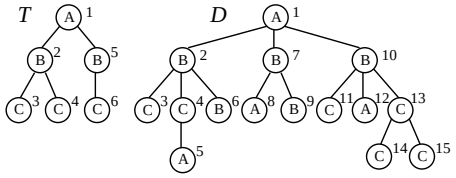

図 1　パターン木 T とデータ木 D

2.1　半構造データのモデル

本稿では，半構造データとパターンの形式的なモデルとして，ラベルつき無順序木を採用する．ここで，ラベル付き無順序木とは，閉路をもたない根付き有向グラフ (DAG) であり，根以外の節点がちょうど一つの親をもつようなものである [4]．形式的には，次のように定義する．

$\mathcal{L} = \{\ell, \ell_1, \ell_2, \ldots\}$ をラベルの可算集合とする．$\mathcal{L}$ 上には全順序 $\leq_{\mathcal{L}}$ が与えられていると仮定する．$\mathcal{L}$ 上の**ラベルつき無順序木**（labeled unordered tree）とは，次の条件を満たす四項組 $U = (V, E, r, label)$ である．$G = (V, E, r)$ は $r \in V$ を根とする木である．V は節点の集合をあらわす有限集合であり，二項関係 $E \subseteq V^2$ は G の親子関係を，$label : V \to \mathcal{L}$ はラベル関数をあらわす．$(u, v) \in E$ ならば，u は v の**親**（parent）である，または v は u の**子**（child）であるという．節点 $v \in V$ の**深さ**$dep(v)$ を，U の根 r から v への経路の長さ，すなわち経路上の辺の数と定義する．

次に，$\mathcal{L}$ 上の**ラベルつき順序木**（labeled ordered tree）とは，次の条件を満たす五項組 $T = (V, E, B, r, label)$ である．$V, E, r, label$ は，無順序木の場合と同様に定義される．$B \subseteq V^2$ は，T の兄弟関係をあらわす二項関係である [6].

$\mathcal{U}$ と $\mathcal{T}$ を，それぞれ $\mathcal{L}$ 上の無順序木と順序木のクラスと定義する．ラベルつき木 $T = (V, E, r, label)$ に対して，文脈から明らかな場合は，V および $E, r, label$ をそれぞれ $V_T, V_E, r_T, label_T$ と表記する．

T を任意の（順序または無順序）木とする．T の任意の節点 u, v に対して，$(u, v) \in E^*$ ならば，u は v の**先祖**（ancestor）である，または u は u の**子孫**（decendant）であるという．節点 v に対して，v を根とし，T における v のすべての子孫によって構成される無順序木を，U の v に関する**部分木**（subtree）と呼び，$T(v)$ と表記する．T の**大きさ**$|T|$ を，$|T| = |V|$ と定義する．**空木**$\perp$ を，大きさが 0 の木と定義する．

例 1　図 1 に，$\mathcal{L} = \{A, B, C\}$ 上のラベルつき無順序木 T と D の例を示す．T の根は節点 1 であり，ラベル A をもつ．節点 2 における部分木 $T(2)$ は，3 つの節点 $2, 3, 4$ から構成される．T の大きさは $|T| = 6$ である．

以降では，大きさ $k \geq 1$ のラベルつき順序木 $T = (V, E, B, r, label)$ に対して，その節点集合は $V = \{1, \ldots, k\}$ であり，それらはプリオーダーで番号付けられていると仮定する．このとき，T の根は $root(T) = 1$ であり，最右葉は $rml(T) = k$ であることに注意されたい．また，T の**最右枝**（rightmost branch）を，T の根から最右葉への経路 $RMB(T) = (r_0, \ldots, r_c)$ $(c \geq 0)$ と定義する．

2.2　パターンと照合写像と出現

無順序木パターン（または，パターン）とは，任意のラベルつき無順序木である．自然数 $k \geq 0$ に対して，大きさ k のパターンを k-**パターン**と呼ぶ．無順序木の有限集合 $\mathcal{D} = \{D_1, \ldots, D_n\} \subseteq \mathcal{U}$ を**データベース**と呼び，各 $D_i (i = 1, \ldots, n)$ を**データ木**と呼ぶ．$\mathcal{D}$ 中の全節点の集合を $V_{\mathcal{D}}$ と表記する．また，$||\mathcal{D}|| = |V_{\mathcal{D}}| = \sum_{D \in \mathcal{D}} |V_D|$ と定義する．

無順序木パターンの意味は，以下で定義される照合写像で与えられる．T と D を，それぞれ $\mathcal{L}$ 上のパターンとデータ木と

する．このとき，任意の $x, y \in V_T$ に対して，次の (1)–(3) を満たす写像 $\varphi : V_1 \to V_2$ が存在するならば，T は D に**出現する**という：

(1) φ は単射である．すなわち，任意の $x, y \in V_1$ に対して，$x \neq y$ ならば $\varphi(x) \neq \varphi(y)$ が成り立つ．

(2) φ は親子関係を保存する．すなわち，任意の $x, y \in V_1$ に対して，$(x, y) \in E_1 \Longleftrightarrow (\varphi(x), \varphi(y)) \in E_2$ が成り立つ．

(3) φ はラベル値を保存する．すなわち，任意の $x \in V_1$ に対して，$L_1(x) = L_2(\varphi(x))$ が成り立つ．

φ を T から D への**照合写像**（matching）と呼ぶ．データ木 D への照合写像 $\varphi : V_T \to V_D$ を，データベース $\mathcal{D}$ への照合写像 $\varphi : V_T \to 2^{V_\mathcal{D}}$ に自然に拡張する．また，パターン T からデータベース $\mathcal{D}$ へのすべての照合写像の集合を，$\mathcal{M}^\mathcal{D}(T)$ であらわす．

［定義 1］ 自然数 $k \geqq 1$ に対して，$T \in \mathcal{U}$ を任意の k パターンとし，$\mathcal{D}$ をデータベースとする．また，$\varphi : V_T \to V_\mathcal{D} \in \mathcal{M}^\mathcal{D}(T)$ を T から $\mathcal{D}$ への任意の照合写像とする．このとき，T の $\mathcal{D}$ への出現として，次の 4 つを定義する：

（1） T の**全出現**（total occurrence）とは，k 項組 $Toc(\varphi) = \langle \varphi(1), \ldots, \varphi(k) \rangle \in (V_\mathcal{D})^k$ である．

（2） T の**埋め込み出現**（embedding occurrence）とは，節点集合 $Eoc(\varphi) = \{\varphi(1), \ldots, \varphi(k)\} \subseteqq V_\mathcal{D}$ である．

（3） T の**根出現**（root occurrence）とは，$\mathcal{D}$ の節点 $Roc(\varphi) = \varphi(1) \in V_\mathcal{D}$ である．

（4） T の**文書出現**（document occurrence）とは，$Eoc(\varphi) \subseteqq V_{D_i}$ が成り立つような文書番号 $Doc(\varphi) = i$ $(1 \leqq i \leqq |\mathcal{D}|)$ である．

例 2 図 1 のパターン T からデータ木 D への全出現は，$\varphi_1 = \langle 1, 2, 3, 4, 10, 11 \rangle$，$\varphi_2 = \langle 1, 2, 4, 3, 10, 11 \rangle$，$\varphi_3 = \langle 1, 2, 3, 4, 10, 13 \rangle$，$\varphi_4 = \langle 1, 2, 4, 3, 10, 13 \rangle$，$\varphi_5 = \langle 1, 10, 11, 13, 2, 3 \rangle$，$\varphi_6 = \langle 1, 10, 13, 11, 2, 3 \rangle$，$\varphi_7 = \langle 1, 10, 11, 13, 2, 4 \rangle$，$\varphi_8 = \langle 1, 10, 13, 11, 2, 4 \rangle$ の 8 通り存在する．ここで，照合写像 φ_i と全出現 $Toc(\varphi_i)$ を同一視した．一方，T から D への埋め込み出現は $Eoc(\varphi_1) = Eoc(\varphi_2) = \{1, 2, 3, 4, 10, 11\}$，$Eoc(\varphi_3) = Eoc(\varphi_4) = \{1, 2, 3, 4, 10, 13\}$，$Eoc(\varphi_5) = Eoc(\varphi_6) = \{1, 2, 3, 10, 11, 13\}$，$Eoc(\varphi_7) = Eoc(\varphi_8) = \{1, 2, 4, 10, 11, 13\}$，の 4 通りであり，根出現は $\varphi_1(1) = \varphi_2(1) = \cdots = \varphi_8(1) = 1$ の 1 つしか存在しない．

任意の出現の種類 $\tau \in \{Toc, Eoc, Roc, Doc\}$ に対して，パターン T のデータベース $\mathcal{D}$ への τ-**出現数**を $|\tau^\mathcal{D}(U)|$ で定義する．また，T の $\mathcal{D}$ への τ-**出現頻度**（または，頻度）を，$\tau \in \{Toc, Eoc, Roc\}$ のときは $freq_\mathcal{D}(T) = |\tau^\mathcal{D}(T)|/||\mathcal{D}||$ と定義し，$\tau = Doc$ のときは $freq_\mathcal{D}(T) = |Doc^\mathcal{D}(T)|/|\mathcal{D}|$ と定義する．パターンの出現頻度の閾値として，ユーザは**最小指示度**（minimun support）と呼ばれる任意の実数 $0 \leqq \sigma \leqq 1$ を与える．以上にもとづき，本稿で考察するデータマイニング問題を導入する．

出現種類 τ に関する頻出無順序木パターン発見問題

与えられたデータベース $\mathcal{D} \subseteqq \mathcal{U}$ と最小指示度 $0 \leqq \sigma \leqq 1$，に対して，$\mathcal{D}$ への τ-出現頻度が σ 以上となるようなすべての無順序木パターン $T \in \mathcal{U}$ を，すべて見つけよ．

本稿では，Eoc に関する頻出無順序木パターン発見問題について議論する．この問題の Roc や Doc への拡張は容易である．

3. 無順序木の正規形表現

本節では，[15] にしたがい，無順序木の正規形表現を導入する．

3.1 ラベルつき順序木の深さラベル列

本稿では，任意のラベルつき無順序木をラベルつき順序木で表現する．すなわち，任意の無順序木 $U \in \mathcal{U}$ に対して，T から兄弟関係 B_T を無視して得られる木が U と一致するような順序木 $T \in \mathcal{T}$ が存在するので，これをこのような T を U の表現として用いる．2 つの順序木 $T_1, T_2 \in \mathcal{T}$ が同一の無順序木 $U \in \mathcal{U}$ を表現するとき，T_1 と T_2 は互いに**同値である**といい，$T_1 \equiv T_2$ と表記する．

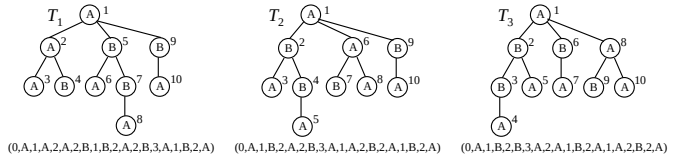

図 2 ラベルつき順序木の深さラベル列

大きさ k のラベルつき順序木を，以下のように符号化する [7], [14], [25]．T を大きさ k のラベルつき順序木とする．このとき，T の**深さラベル列**（depth-label sequence）とは，系列

$$C(T) = ((dep(v_1), label(v_1)), \ldots, (dep(v_k), label(v_k)) \in (\mathbf{N} \times \mathcal{L})^*$$

である．ただし，$(v_1, \ldots, v_k)$ は，T の節点をプリオーダー順に並べた系列であり，$(dep(v_i), label(v_i)) \in \mathbf{N} \times \mathcal{L}$ は，節点 $v_i \in V_T$ の**深さラベル対**（depth-label pair）である．$\mathcal{T}$ と $\mathbf{N} \times \mathcal{L}$ は一対一に対応するので，以降は順序木 T とその深さラベル列 $C(T)$ を同一視する．図 2 に深さラベル列の例を示す．

次に，深さラベル対上の全順序 $\geq \subseteqq (\mathbf{N} \times \mathcal{L})^2$ を導入する．任意の深さラベル対 $(d_i, \ell_i) \in \mathbf{N} \times \mathcal{L}$ $(i = 1, 2)$ に対して，$(d_1, \ell_1) > (d_2, \ell_2)$ であるのは，(i) $d_1 > d_2$ が成り立つとき，または (ii) $d_1 = d_2$ かつ $\ell_1 > \ell_2$ が成り立つときである．全順序 $\geq$ から生成される辞書式順序 $\geqq_{\rm lex}$ は，深さラベル列上の全順序を与える．すなわち，任意の順序木 T_1, T_2 に対して，$C(T_1) \geqq_{\rm lex} C(T_2)$ と $C(T_2) \geqq_{\rm lex} C(T_1)$ のいずれかが成り立つ．

以上の準備により，ラベルつき無順序木の正規形表現を定義する．

［定義 2］（[15]） T を任意のラベルつき順序木とする．T と同値な任意のラベルつき順序木 S に対して，$C(T) \geqq_{\rm lex} C(S)$ が成り立つとき，T を**正規形**と定義する．

ラベルつき無順序木 $U \in \mathcal{U}$ を表現する正規形順序木 $T \in \mathcal{T}$ を，U の**正規形表現**と呼び，$T = COT(U)$ と表記する．また，$\mathcal{L}$ 上のラベル付き無順序木の正規形表現全体を $\mathcal{C} \subseteqq \mathcal{T}$ と定義する．

次の補題は，無順序木の正規形表現に関する重要な特徴づけを与える．

［補題 1］（左荷重条件；Left-heavy condition [15]） ラベルつき順序木 T がある無順序木の正規形表現となることの必要十分条件は，T が**左荷重**（left-heavy）となること，すなわち，T の任意の節点 $v_1, v_2 \in V$ に対して，$(v_1, v_2) \in B$ ならば $C(T(v_1)) \geqq_{\rm lex} C(T(v_2))$ が成り立つことである．

例 3 図 2 に示される 3 つの相異なる順序木 T_1, T_2, T_3 は，同一の無順序木を表現する．ラベル間の順序として $A > B > C$ を仮定すると，T_1 は左荷重条件を満たすことから正規形であることが分かる．一方，T_2 と T_3 は左荷重条件を満たさないので，正規形ではない．

3.2 逆探索と最右拡張

逆探索（reverse search）とは，列挙問題を高速に解くための効率よいアルゴリズム構成法の 1 つである [9], [19]．逆探索では，列挙問題の解空間 $\mathcal{S}$ に対して，任意の解 $X \in \mathcal{S}$ が唯一の親 $P(X)$ をもつように，親子関係 $P \subseteqq \mathcal{S} \times \mathcal{S}$ を導入する．このとき，親子関係 P は解空間 $\mathcal{S}$ 上の探索木を構成するので，根からはじめて，順に子供を計算することにより，すべての解を重複せずに列挙することができる．

T を大きさ 2 以上のラベルつき順序木とする．T から最右葉

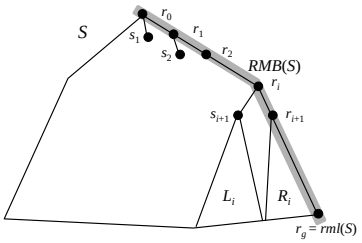

図 3　正規形表現に関する記法

Algorithm UNOT$(\mathcal{D}, \mathcal{L}, \sigma)$
入力：データベース $\mathcal{D} = \{D_1, \ldots, D_m\}$ $(m \geqq 0)$，ラベル集合 $\mathcal{L}$，最小指示度 $0 \leqq \sigma \leqq 1$.
出力：頻出無順序木パターンの集合 $\mathcal{F} \subseteqq \mathcal{C}$.
手法：
1. $\mathcal{F} := \emptyset$; $\alpha := \lceil |\mathcal{D}|\sigma \rceil$; /* 初期化 */
2. 任意のラベル $\ell \in \mathcal{L}$ について，以下をおこなう：
   $T_\ell := (0, \ell)$;
   Expand$(T_\ell, \mathcal{O}, 0, \alpha, \mathcal{F})$;
3. $\mathcal{F}$ を出力する; /* 頻出パターン集合 */

図 4　頻出無順序木パターン発見アルゴリズム UNOT

$rml(T)$ を除去して得られる順序木を $P(T)$ と定義する．逆探索の考え方にしたがい，$P(T)$ を T の親と呼び，T を $P(T)$ の子と呼ぶ．次の補題は，無順序木の正規形表現も逆探索手法で列挙可能なことを示している．

［補題 2］（[15]） 任意のラベルつき順序木 $T \in \mathcal{T}$ に対して，T が正規形ならば，その親 $P(T)$ も正規形である．すなわち，$T \in \mathcal{C}$ ならば $P(T) \in \mathcal{C}$ が成り立つ．

証明： 任意の正規形順序木 $T \in \mathcal{C}$ について，T から最右葉を除去する操作が左荷重条件を損ねることはない．よって，$P(T)$ も正規形である． □

［定義 3］（[6], [14], [25]） $S \in \mathcal{T}$ を $\mathcal{L}$ 上のラベルつき順序木とする．S の最右枝 $RMB(S)$ 上の節点に，新たな節点 v を一番右の子となるように付け加えて得られるラベルつき順序木 $T \in \mathcal{T}$ を，S の**最右拡張**（rightmost expansion）という．特に，$(dep(v), label(v)) = (d, \ell)$ のとき，T を S の (d, ℓ) **拡張**と呼ぶ．$\perp$ の $(0, \ell)$ 拡張を，ラベル ℓ をもつ大きさ 1 の木と定義する．

新しく付加される節点 v は，T のプリオーダーにおける最後の節点となることから，S の (d, ℓ) 拡張を $S{\cdot}(d, \ell)$ とも書く．

4. 頻出無順序木パターン発見

本節では，埋め込み出現に関する頻出無順序木パターン発見問題を効率よく解くアルゴリズム UNOT について述べる．

4.1 アルゴリズムの概要

図 4 に，我々のアルゴリズム UNOT を示す．このアルゴリズムは，与えられたデータベース $\mathcal{D}$ に頻出するすべての無順序木の正規形表現を効率よく発見する．

アルゴリズムの基本アイディアは，無順序木の正規形表現の高速な列挙と，パターンの埋め込み出現の漸増的な計算である．UNOT は，まず，図 4 の部分手続き FindAllChildren を用いて，パターン 1 つ当たり定数時間ですべての正規形表現を重複せずに列挙する．次に，図 8 の部分手続き UpdateOcc を用いて，パターン 1 つあたり $O(bk^2m)$ 時間ですべての埋め込み出現を計算する．ここに，k はパターン T の大きさであり，b はデータ木の最大枝分かれ数，m は T のデータ木への総出現数である．

本節の残りで，これらの手続きを細かく説明する．

4.2 無順序木の列挙

はじめに，いくつかの概念と記法を準備する（図 3）．T をラベルつき順序木とし，その最右枝を $RMB(T) = (r_0, r_1, \ldots, r_g)$ とおく．任意の自然数 $i = 0, 1, \ldots, g$ に対して，r_i が 2 つ以上の子供をもつとき，r_i の子供かつ r_{i+1} の直前の兄である節点を s_{i+1} と書く．すなわち，s_{i+1} は r_i の最後から 2 番目の子供である．また，$L_i = T(s_{i+1})$ を r_i の左木と，$R_i = T(r_{i+1})$ を r_i の右木と呼ぶ．r_i がちょうど 1 つの子供 r_{i+1} をもつとき，$L_i = \top_\infty$ と定義する．ここで，無限木 $\top_\infty$ は，任意の $S \in \mathcal{T}$ に対して $\top_\infty >_{\rm lex} S$ が成り立つ特別な木である．

補題 1 より，ある順序木が正規形であることの必要十分条件は，それが左荷重であることである．次の補題は，与えられた順序木 T が正規形かどうか調べるために，アルゴリズムは T の左木と右木のみをチェックすれば十分だと主張する．

［補題 3］（[15]） $S \in \mathcal{C}$ を任意の正規形表現とし，T を S の子供とする．T の最右枝を $(r_0, \ldots, r_g)$ とおく（$g \geqq 0$）．このとき，T が正規形表現であることの必要十分条件は，任意の $i = 0, \ldots, g-1$ について $L_i \geqq_{\rm lex} R_i$ が成り立つことである．

T を正規形順序木と仮定する．最右拡張を T まで繰り返す中で，右木 R_i は以下のように変化する．

（1） 新しい節点 v が r_i の最右の子として追加されたとき，新たな右木 R_i は v のみで構成される大きさ 1 の木である．このとき，$C(R_i) = \boldsymbol{v} = (dep(v), label(v))$ が成り立つ．

（2） 新たな節点 v の深さが $d = dep(v) > i$ のとき，右木 R_i は必ず大きくなる．その際，以下の 2 つの場合 (i),(ii) が考えられる：

(i) ある整数 $j \leqq \min(|C(L_i)|, |C(R_i)|)$ が存在し，$C(L_i)$ と $C(R_i)$ の j 番目の要素が異なると仮定する．このとき，$\boldsymbol{r}_{dep(v)} \geqq \boldsymbol{v}$ ならば，今回の最右拡張は T の左荷重条件を損なわない．ここで，$r_{dep(v)}$ は，新たな木における v の左隣の兄弟である．

(ii) それ以外のとき，すなわち $C(R_i)$ が $C(L_i)$ の接頭辞である場合，$m = |C(R_i)| < |C(L_i)|$ とし，$C(L_i)$ の m 番目の要素を $\boldsymbol{w}$ とおく．このとき，新たな節点 $\boldsymbol{v}$ に対して，$T{\cdot}\boldsymbol{v}$ が正規形となることの必要十分条件は，$\boldsymbol{w} \geqq \boldsymbol{v}$ と $\boldsymbol{r}_{dep(v)} \geqq \boldsymbol{v}$ が成り立つことである．

(iii) (i) と (ii) の場合において，$r_{dep(v)-1}$ が T の最右葉のときは，$\boldsymbol{r}_{dep(v)}$ が未定義である．この場合は，$\boldsymbol{r}_{dep(v)} = \top_\infty$ と定義する．

（3） 最後に，$C(L_i) = C(R_i)$ が成り立つまで R_i が成長したとする．このとき，R_i へのこれ以上の最右拡張はおこなわれない．

与えられた無順序木の正規形表現 T に対して，T のすべての右木 $R_0, \ldots, R_g$ が上の条件を満たすように拡張して得られた順序木は，正規形になる．

$RMB(T) = (r_0, r_1, \ldots, r_g)$ を T の最右枝とする．任意の $i = 0, 1, \ldots, g-1$ に対して，$C(R_i)$ が $C(L_i)$ の接頭辞のとき，節点 r_i を**アクティブ**と呼ぶ．T の**コピー深さ**（copy depth）とは，T のアクティブ節点の中でもっとも浅い節点の深さである．

Procedure Expand$(S, \mathcal{O}, c, \alpha, \mathcal{F})$
入力： 正規形表現 $S \in \mathcal{U}$，埋め込み出現 $\mathcal{O} = EO^{\mathcal{D}}(S)$，コピー深さ c，非負整数 α，頻出パターン集合 $\mathcal{F}$.
手法：
- $|\mathcal{O}| < \alpha$ ならば手続きを終了する; それ以外のとき，$\mathcal{F} := \mathcal{F} \cup \{S\}$ とする;
- 任意の $\langle S{\cdot}(i, \ell), c_{\rm new} \rangle \in$ FindAllChildren(S, c) について，以下をおこなう：
  - $T := S{\cdot}(i, \ell)$;
  - $\mathcal{P} :=$ UpdateOcc$(T, \mathcal{O}, (i, \ell))$;
  - Expand$(T, \mathcal{P}, c_{\rm new}, \alpha, \mathcal{F})$;

図 5　深さ優先手続き Expand

Procedure FindAllChildren(S, k)
入力： 無順序木の正規形表現 S と，S のコピー深さ k.
手法： S の正規最右拡張 T と，T のコピー深さ c の組 $\langle T, c\rangle$ をすべて出力する．T と c は，以下の場合分けにしたがって計算される：

Case I　$C(L_k) = C(R_k)$ のとき：

- S の正規最右拡張は，$S{\cdot}(1,\ell_1), \ldots, S{\cdot}(k+1,\ell_{k+1})$ である．ただし，任意の $i = 1, \ldots, k+1$ について，$label(r_i) \geqq \ell_i$ を満たすとする．一方，$S{\cdot}(k+2,\ell_{k+2}), \ldots, S{\cdot}(g+1,\ell_{g+1})$ は正規形でない．
- 任意の $i = 1, \ldots, k+1$ に対して，$S{\cdot}(i,\ell_i)$ のコピー深さは，$label(r_i) = \ell_i$ ならば $i-1$ であり，それ以外のときは i である．

Case II　$C(L_k) \neq C(R_k)$ のとき：

- $m = |C(R_k)| + 1$ とし，$\boldsymbol{w} = (d, \ell)$ を $C(L_k)$ の m 番目の要素とする．このとき，S の正規最右拡張は $S{\cdot}(1,\ell_1), \ldots, S{\cdot}(d,\ell_d)$ である．ただし，任意の $i = 1, \ldots, d-1$ について $label(r_i) \geqq \ell_i$ を満たし，$i = d$ のときは $\ell \geqq \ell_d$ を満たすとする．
- 任意の $i = 1, \ldots, d-1$ に対して，$S{\cdot}(i,\ell_i)$ のコピー深さは，$label(r_i) = \ell_i$ ならば $i-1$ であり，それ以外のときは i である．$S(d,\ell_d)$ のコピー深さは，$\boldsymbol{w} = \boldsymbol{v}$ ならば k であり，それ以外のときは d である．

図 6　すべての正規最右拡張を計算するアルゴリズム FindAllChildren

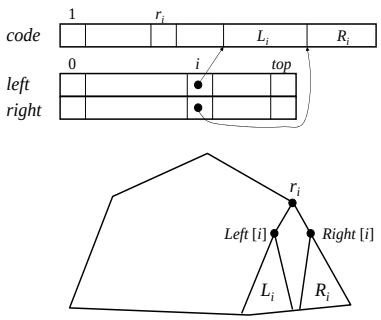

図 7　パターンのデータ構造

また，最右葉 r_g は常にアクティブであるとする．このとき，T が r_g 以外にアクティブ節点をもたないならば，コピー深さが g となることに注意されたい．

以降では，与えられた正規形表現 $T \in \mathcal{C}$ の子供（**正規最右拡張**と呼ぶ）をすべて生成するアルゴリズム FindAllChildren について述べる．図 6 に示されるこのアルゴリズムは，中野と宇野 [15] による，ラベルをもたない無順序木を列挙するアルゴリズムとほとんど同じである．ただし，ラベルが付くことにより，コピー深さの更新法が [15] とは若干異なる．

アルゴリズム FindAllChildren が，解 1 つあたり定数時間ですべての正規最右拡張を列挙できるよう，実装上の工夫としてパターンに図 7 のデータ構造を導入する．

- 深さラベル対の配列 $code : [1..size] \to (\mathbf{N} \times \mathcal{L})$．ここにはパターン T（大きさ $size \geqq 0$）の深さラベル列を格納する．
- 三項組 $(left, right, cmp)$ のスタック $RMB : [0..top] \to (\mathbf{N} \times \mathbf{N} \times \{=, \neq\})$．$(left, right, cmp) = RMB[i]$ に対して，$left$ と $right$ は，それぞれ $code$ における左木 L_i と右木 R_i の開始位置を指すポインタである．フラグ $cmp \in \{=, \neq\}$ は，$L_i = R_i$ が成り立つか否かを記録する．パターンの最右枝の長さを，$top \geqq 0$ であらわす．

このデータ構造を用いることにより，図 6 の FindAllChildren において，すべての演算が定数時間で動くように実装できる．ただし，解である正規最右拡張は，親の木との差分のみを出力すると仮定する．

次の補題は，ラベルの扱いを除けば，[15] と同様に証明できる．

［補題 4］（[15]）　任意の正規形表現 S と，そのコピー深さ $k \geqq 0$ に対して，図 6 のアルゴリズム FindAllChildren は，S のすべての正規最右拡張 T を，解 1 つあたり $O(1)$ 時間で計算する．ただし，T と S との差分のみを出力すると仮定する．

Algorithm UpdateOcc($T, \mathcal{O}, d, \ell$)
入力： S の正規最右拡張 T，全出現リスト $\mathcal{O} = TO^{\mathcal{D}}(S)$，
　　T の最右葉の深さラベル対 (d, ℓ).
出力： 更新された全出現リスト $\mathcal{P} = TO^{\mathcal{D}}(T)$.

- $\mathcal{P} := \emptyset$;
- 任意の $\varphi \in \mathcal{O}$ に対して，以下をおこなう：
  + $x := \varphi(r_{d-1})$;
  + x の任意の子節点 y に対して，以下をおこなう：
    - $label_D(y) = \ell$ かつ $y \notin E(\varphi)$ ならば，
      $\xi := \varphi{\cdot}y$ かつ $flag := true$ とする;
    - それ以外のとき，残りをスキップしてループを続ける;
    - ある $i = 1, \ldots, d-1$ が存在して，
      $C(L_i) = C(R_i)$ かつ $\xi(left_i) > \xi(right_i)$ ならば，
      $flag := false$ とした後，内側のループを抜ける;
    - $flag = true$ ならば $\mathcal{P} = \mathcal{P} \cup \{\xi\}$ とする;
- $\mathcal{P}$ を出力する;

図 8　パターンの埋め込み出現を更新するアルゴリズム UpdateOcc

［補題 1 に基づいて，アルゴリズム FindAllChildren を素直に構築すると，パターンの大きさ k に対して，解である正規最右拡張 1 つあたり $O(k^2)$ 時間の計算時間を要する．したがって，このアルゴリズムは，素朴なアルゴリズムに比べて非常に効率がよいといえる．

4.3　出現リストの更新

本節では，正規形表現 S の埋め込み出現 $EO^{\mathcal{D}}(S)$ から，その正規最右拡張 T の埋め込み出現 $EO^{\mathcal{D}}(T)$ を漸増的に計算する方法を与える．図 8 に，これを実現する部分手続き UpdateOcc を示す．

T を，大きさ k のラベルつき無順序木の正規形表現とする．また，T から $\mathcal{D}$ への照合写像を $\varphi \in \mathcal{M}^{\mathcal{D}}(T)$ とおく．T の φ に関する全出現と埋め込み出現は，それぞれ $TO(\varphi) = \langle \varphi(1), \ldots, \varphi(k)\rangle$ と $EO(\varphi) = \{\varphi(1), \ldots, \varphi(k)\}$ で与えられる．文脈から明らかな場合は φ と $TO(\varphi)$ を同一視する．

我々は，埋め込み出現 EO を，$EO = EO(\varphi)$ であるような全出現 φ の 1 つを用いて符号化する．しかし，ある埋め込み出現 EO に対応する全出現は，最悪で指数個存在する．そこで，3. 節で順序木の正規形を導入したように，全出現にも正規形を導入し，これを埋め込み出現をあらわす表現として用いる．

$EO(\varphi_1) = EO(\varphi_2)$ が成り立つとき，2 つの全出現 φ_1 と φ_2 は**同値である**という．φ_1 が，自然数列 $\mathbf{N}^*$ として辞書式順序で φ_2 より大きいことを，$\varphi_1 \geqq_{\rm lex} \varphi_2$ と表記する．埋め込み出現の正規形表現を以下で定義する．

［定義 4］　T をラベルつき無順序木の正規形表現とし，$EO \subseteq V_{\mathcal{D}}$ を T の $\mathcal{D}$ への埋め込み出現とする．このとき，EO の正規形表現 $CR(EO)$ を，同値類 $\{\, \varphi' \in \mathcal{M}^{\mathcal{D}}(T) \,|\, \varphi' \equiv \varphi \,\}$ の中で辞書式順序が最大となる全出現と定義する．

$\varphi = \langle \varphi(1), \ldots, \varphi(k)\rangle$ を T の全出現とする．φ から最後の要素 $\varphi(k)$ を除去して得られる全出現を，φ の**親出現**（parent occurrence）と呼び，$P(\varphi)$ と書く．T の節点 $v \in V_T$ に対して，部分木 $T(v)$ への φ の制限を $\varphi(T(v)) = \langle \varphi(i), \varphi(i+1), \ldots, \varphi(i+|T(v)|-1)\rangle$ とする．ここで，$\langle i, i+1, \ldots, i+|T(v)|-1\rangle$ は，プリオーダー順に並べた $T(v)$ の節点列である．

以上の準備に基づき，埋め込み出現の漸増的な計算法を与えよう．

［補題 5］　S を正規形順序木とし，φ を S の $\mathcal{D}$ への正規全出現とする．$T = S{\cdot}\boldsymbol{v}$ を S の正規最右拡張とし，その最右枝を $(r_0, \ldots, r_g)$ とおく．このとき，データ木の節点 $w \in V_{\mathcal{D}}$ に対して，写像 $\xi = \varphi{\cdot}\boldsymbol{w}$ が T の正規全出現になるための必要十分条

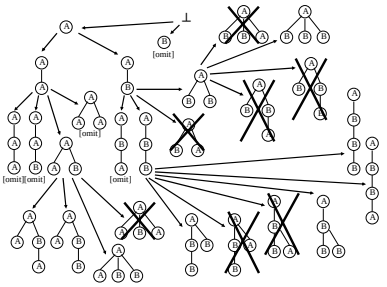

図 9　ラベルつき無順序木の探索木

件は，以下の条件 (1)–(4) が成り立つことである：

(1)　$label_D(w) = label_T(v)$.

(2)　任意の $i = 1, \ldots, k-1$ に対して，$w \neq \varphi(i)$ である．

(3)　w は $\varphi(r_{d-1})$ の子節点である．ここに，$d = dep(v)$ とする．

(4)　任意の $i = 0, \ldots, g-1$ に対して，$C(L_i) = C(R_i)$ ならば $\xi(root(L_i)) < \xi(root(R_i))$ が成立する．

証明：　任意の全出現 $\varphi \in \mathcal{M}^{\mathcal{P}}(T)$ に対して，φ が正規形ならば，その親出現 $P(\varphi)$ も正規形である．さらに，φ が正規形であることの必要十分条件は，任意の節点 $v_1, v_2 \in V_T$ について，$(v_1, v_2) \in B$ と $T(v_1) = T(v_2)$ が成り立つならば $\varphi(T(v_1)) \leqq_{\rm lex} \varphi(T(v_2))$ が成り立つことである．以上より，補題が示された．　□

補題 5 は，図 8 のアルゴリズム UpdateOcc の正当性を保障する．図 7 のデータ構造を用いることにより，このアルゴリズムにおける $C(L_i) = C(R_i)$ の判定を定数時間で行うことができる．また，すべての正規形順序木は，少なくとも 1 つの正規最右拡張をもつことに注意せよ．

以上より，本稿の主結果が導出される．

［定理 6］　$\mathcal{D}$ をデータベースとし，$0 \leqq \sigma \leqq 1$ を最小指示度とする．このとき，図 4 のアルゴリズム UNOT は，埋め込み出現に関するすべての頻出無順序木の正規形表現 T を，パターン 1 つあたり $O(kb^2m)$ 時間で計算する．ここで，k はパターンの最大サイズであり，b は入力データ木の最大枝分かれ数，m はパターン T の埋め込み出現数である．

証明：　S を正規形表現とし，T をその正規最右拡張とする．このとき，図 8 の手続き UpdateOcc は，$k' = |T|$，$m' = |EO(S)|$ とすると，T のすべての正規全出現を $O(k'bm')$ 時間で計算する．補題 4 と $|EO(T)| = O(b|EO(S)|)$ より，定理が成り立つ．　□

図 9 は，アルゴリズム UNOT が $\mathcal{L} = \{A, B\}$ 上のサイズ 4 以下のラベルつき無順序木を計算する様子を図示している．図中の矢印はラベルつき順序木の親子関係をあらわし，×印は正規形でない順序木をあらわす．

5. おわりに

本稿では，与えられた無順序木の集積に頻出するすべての無順序木パターンを効率よく計算するアルゴリズム UNOT を開発した．このアルゴリズムは，パターン 1 つあたり $O(kb^2m)$ 時間ですべての頻出パターン T を計算する．ここに，k は T の大きさであり，b はデータ木の最大枝分かれ数，m は T のデータ木への総出現数である．

研究会の発表では，UNOT の計算機実験についても報告する予定である．

文　献

[1] K. Abe, S. Kawasoe, T. Asai, H. Arimura, and S. Arikawa. Optimized Substructure Discovery for Semi-structured Data, In *Proc. PKDD'02*, 1–14, LNAI 2431, 2002.

[2] S. Abiteboul, P. Buneman, D. Suciu, *Data on the Web*, Morgan Kaufmann, 2000.

[3] R. Agrawal, R. Srikant, Fast Algorithms for Mining Assoiciation Rules, In *Proc. the 20th VLDB*, 487–499, 1994.

[4] Aho, A. V., Hopcroft, J. E., Ullman, J. D., *Data Structures and Algorithms*, Addison-Wesley, 1983.

[5] T. R. Amoth, P. Cull, and P. Tadepalli, Exact learning of unordered tree patterns from queries, In *Proc. COLT'99*, ACM Press, 323–332, 1999.

[6] T. Asai, K. Abe, S. Kawasoe, H. Arimura, H. Sakamoto, S. Arikawa, Efficient Substructure Discovery from Large Semi-structured Data, In *Proc. the 2nd SIAM Int'l Conf. on Data Mining (SDM2002)*, 158–174, 2002.

[7] T. Asai, H. Arimura, K. Abe, S. Kawasoe, S. Arikawa, Online Algorithms for Mining Semi-structured Data Stream, In *Proc. the 2002 IEEE Int'l Conf. on Data Mining (ICDM'02)*, 27–34, 2002.

[8] T. Asai, H. Arimura, T. Uno, S. Nakano, Discovering Frequent Substructures in Large Unordered Trees, In *Proc. the 6th Discovery Science (DS'03)*, LNAI, October 2003. (To appear)

[9] D. Avis, K. Fukuda, Reverse Search for Enumeration, Discrete Applied Mathematics, 65(1–3), 21–46, 1996.

[10] L. B. Holder, D. J. Cook, S. Djoko, Substructure Discovery in the SUBDUE System, In *Proc. KDD'94*, 169–180, 1994.

[11] A. Inokuchi, T. Washio, H. Motoda, An Apriori-Based Algorithm for Mining Frequent Substructures from Graph Data, In *Proc. PKDD 2000*, 13–23, LNAI, Springer, 2000.

[12] M. Kuramochi, G. Karypis, Frequent Subgraph Discovery, In *Proc. IEEE ICDM'01*, 2001.

[13] T. Miyahara, Y. Suzuki, T. Shoudai, T. Uchida, K. Takahashi, H. Ueda, Discovery of Frequent Tag Tree Patterns in Semistructured Web Documents, In *Proc. PAKDD-2002*, 341–355, 2002.

[14] S. Nakano, Efficient generation of plane trees, Information Processing Letters, 84, 167–172, 2002.

[15] S. Nakano, T. Uno, Efficient Generation of Rooted Trees, *NII Technical Report NII-2003-005E*, ISSN 1346-5597, Natinal Institute of Informatics, July 2003.

[16] S. Nestrov, S. Abiteboul, R. Motwani, Extracting Schema from Semistructured Data, In *Proc. SIGKDD'98*, 295–306, 1998.

[17] S. Nijssen, J. N. Kok, Effcient Discovery of Frequent Unordered Trees, In *Proc. MGTS'03*, September 2003.

[18] A. Termier, M. Rousset, M. Sebug, TreeFinder: a First Step towards XML Data Mining, In *Proc. IEEE ICDM'02*, 450–457, 2002.

[19] 宇野毅明，“ 効率的な列挙アルゴリズムの構築と利用（3）–困難な列挙問題と逆探索法– ”，人工知能学会誌，18(5)，586–591，2003.

[20] N. Vanetik, E. Gudes, E. Shimony, Computing Frequent Graph Patterns from Semistructured Data, In *Proc. IEEE ICDM'02*, 458–465, 2002.

[21] W3C, Extensive Markup Language (XML) 1.0 (Second Edition), *W3C Recommendation*, 06 October 2000. `http://www.w3.org/TR/REC-xml`

[22] K. Wang, H. Liu, Schema Discovery from Semistructured Data, In *Proc. KDD'97*, 271–274, 1997.

[23] X. Yan, J. Han, gSpan: Graph-Based Substructure Pattern Mining, In *Proc. IEEE ICDM'02*, 721–724, 2002.

[24] 吉田健一，元田浩，“ 推論過程からの概念学習（1）–類型的推論過程の抽出– ”，人工知能学会誌，7(4)，119–129，1992.

[25] M. J. Zaki. Efficiently Mining Frequent Trees in a Forest, In *Proc. SIGKDD 2002*, ACM, 2002.