ラベル付き有向グラフに対する Shape Expression Schema の抽出

筑波大学
図書館情報メディア研究科
2020 年 3 月
坪井　悠冬里

目次

第 1 章

はじめに

グラフとは，モノとモノとの関係をノード（点）とエッジ（線）で表現する表現方法である．現代社会において，様々な種類のデータがグラフで表現されるようになった．具体的には，人と人との関係を表現したソーシャルグラフや出版物と出版物の引用関係を表現した引用グラフが挙げられる．

多くの場合，グラフは非常に大規模であり数多くのノードやエッジを有する．従って，そのグラフの特徴を知るために，グラフの概形（構造）が得られれば有用である．スキーマとは，データの構造を表現したものである．そこで本研究では，グラフの構造を表現するためのスキーマである Shape Expression Schema (ShEx) に焦点を当て，グラフから ShEx を抽出することを考える．グラフから「適切な」スキーマを抽出できれば，そのスキーマはグラフの構造を簡潔に表現しているため，グラフデータの効率的な管理に役立つ．更に，スキーマはクエリの最適化や構造の参照やクエリの定式化をする際に利用できる．また，スキーマは ObjectRank スコアの計算にも使用される [1].

グラフデータからスキーマを抽出するアルゴリズムは数多く提案されている．これらのアルゴリズムは大規模なグラフから小さなグラフ（スキーマグラフ）を抽出する．主な先行研究として以下の 2 つの種類がある．1 つ目は，ノードが持つラベルの類似度を基にした研究である [2]．MDL 原理に基づき，以下で示すコストが最小になるようなスキーマを求める．このアルゴリズムにおいては，『与えられたグラフデータの概形 S』と『概形から元のグラフを再構成するのに必要なエッジ修正リスト C』を抽出する．コストを『$|E_s| + |C|$（E_s：概形に含まれるエッジの数，$|C|$：修正エッジの数)』と定義する．ラベルなしの無向グラフに対してスキーマ抽出を行っている．2 つ目は，グラフデータのパスを基にした研究である [3, 4]．同じラベルパスで到達できるノード同士を 1 つにまとめることで，スキーマ抽出を行う．しかし，サイクルを含むグラフデータに対してスキーマを抽出する際，スキーマが元のグラフデータより大きくなる可能性がある．これらのアルゴリズムは，大規模なグラフから小さいグラフを抽出するアルゴリズムを提案している．一方で，本研究はグラフから ShEx スキーマを抽出する研究である．

本研究では，ラベル付き有向グラフを対象にスキーマ抽出を行う．また，スキーマとして ShEx を考える．ShEx は，オントロジーセマンティクスというよりむしろ，RDF データの構造的特徴を取得するために設計されている．ShEx の仕様は，W3C Draft Community Group により開発されている [5]．ShEx は，ノードとその近傍に構造的制約を課す型の集合であり，それぞれのノードに型が割り当てられる．型はノードが持つ出力エッジと接続先のノードの型を規定する．

スキーマ抽出においては，類似した構造を持ったノードを 1 つにまとめることでグラフの概形を求めて

いる．更に，求めたスキーマの妥当性を確保するために，各ノードに適切に型を割り当てれば，どのノードも割り当てわれた型の定義を満たすようなスキーマを抽出する．

本研究では，ラベル付き有向グラフを対象として提案アルゴリズムの評価実験を行った．使用したデータは，RDF データのベンチマークツール SP2Bench を用いて生成した RDF データと LodPaddle プロジェクトによって提供されているナント市周辺の地域の情報を表した RDF データである．評価実験の結果，概ね適切にスキーマを抽出可能であることが分かった．

論文の構成は以下の通りである．第 2 章では Shape Expression Schema の定義を述べる．第 3 章では本研究で提案する手法について説明する．第 4 章では，スキーマ抽出の NP 完全性について述べる．第 5 章では，評価実験について述べる．第 6 章では，本研究のまとめを述べる．

第 2 章

諸定義

2.1 グラフ

本研究では主に，有限グラフに焦点を当てる．以下 Σ, T を有限集合とする．Σ の要素をエッジラベルと呼び，T の要素を型（またはノードラベル）と呼ぶ．

定義 1. 1. 有限グラフ G は，$N(G)$ で示されるノード集合と $E(G)$ で示されるエッジ集合で構成される．

2. $G = (N(G), E(G))$ は，$E(G)$ が $N(G) \times \Sigma \times N(G)$ の部分集合であるとき，Σ-エッジラベル付き有向グラフ（Σ-ラベル付き有向グラフ）と呼ばれる．

図 2.1 に Σ-ラベル付き有向グラフの例を示す．

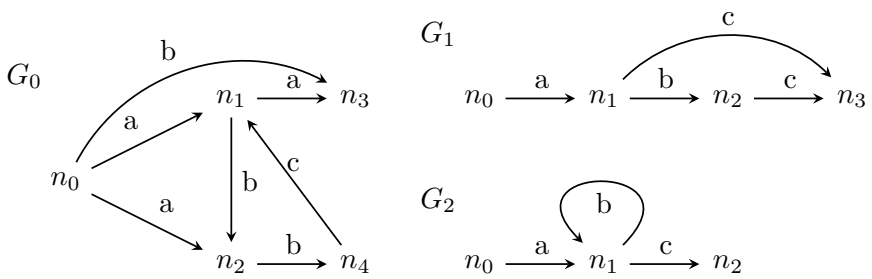


図 2.1 Σ-ラベル付き有向グラフ

本研究では，ノードの直近の外側近傍に基づいてグラフの構造を形成する．Σ-ラベル付き有向グラフ $G = (N(G), E(G))$ のノード n から出ているエッジの集合を

$$out\_lab\_node_G(n) = \{(a, m) \in \Sigma \times N(G) \mid (n, a, m) \in E(G)\}$$

と定義する．図 2.1 の G_0 において，$out\_lab\_node_{G_0}(n_0) = \{(a, n_1), (a, n_2), (b, n_3)\}$ である．場合によっては，ターゲットノードを無視して，出力ラベルのコレクションのみを使用する．しかし，ノードは同じラベルとその近傍を持った出力エッジを複数持っている可能性がある．例えば G_0 のノード n_0 は，ラベル a を持つエッジを 2 本，ラベル b を持つエッジを 1 本持っている．ここで，集合では要素の出現回

数を示すことはできないため，エッジの種類は示せてもエッジの数を示すことはできない．従って，本質的に全てのシンボルの出現回数を規定することができる多重集合を導入する．

2.2　シンボルの多重集合と Regular Bag Expression

定義 2. 関数

$$f : \Sigma \times \mathrm{T} \to \mathbb{N}$$

を直積集合 $\Sigma \times \mathrm{T}$ に対する多重集合（またはバッグ）と呼ぶ.

多重集合はシンボルをその出現回数に対応させる関数である．$f(\alpha) = 2$ は $\alpha(\in \Sigma \times \mathrm{T})$ が 2 回出現したことを意味する．

本研究においては，ノード（とエッジ）は線形に順序付けられていない非順序グラフを扱う．従って，順序付けられた連結演算子の代わりに，順序を無視した連結演算子 $\|$ を使用した多重集合に対する正規表現を使用する．Regular Bag Expression (RBE) は多重集合を表現する方法の 1 つである．$\Sigma \times \mathrm{T}$ に対する RBE は以下のように定義される．

$$E ::= \epsilon \mid \alpha \mid (E|E) \mid (E \parallel E) \mid E^* \ (\alpha \in \Sigma \times \mathrm{T}).$$

すなわち，RBE は次のように再帰的に定義される：

- ϵ は RBE
- α $(\in \Sigma \times T)$ は RBE
- もし E_1 と E_2 が RBE ならば
  - $E_1 \mid E_2$ は RBE
  - $E_1 \parallel E_2$ は RBE
  - E_1^* は RBE

更に，区間 $[n, m]$ に対して，$a^{[n,m]}(a \in \Sigma \times \mathrm{T})$ は論理和 $\mid$ と順序を無視した連結 $\|$ を使用して定義できる．例えば，$a^{[2,4]}$ は $a^{[2,4]} ::= (a \parallel a) \mid (a \parallel a \parallel a) \mid (a \parallel a \parallel a \parallel a)$ と定義できる．

L を RBE を $\Sigma \times \mathrm{T}$ に対する多重集合の集合に割り当てる関数とする．$\Sigma \times \mathrm{T}$ に対する RBE のセマンティクスは以下のように定義される：

- $L(\epsilon) = \{\bar{\epsilon}\}$，ただし $\bar{\epsilon}$ は全てのシンボルを 0 に対応させる関数（多重集合）である．
- 各 $\alpha \in \Sigma \times \mathrm{T}$ に対して，$L(\alpha) = \{f_\alpha\}$，ただし f_α は $f_\alpha(\alpha) = 1, f_\alpha(\beta) = 0(\beta \neq \alpha)$ となるような関数である．$(a, t) \in \Sigma \times \mathrm{T}$ を $a :: t$ と表記する．
- $L(E_1 \mid E_2) = L(E_1) \cup L(E_2)$
  E_1 に含まれる多重集合の集合と E_2 に含まれる多重集合の集合の和集合.
- $L(E_1 \parallel E_2) := \{f_1 \uplus f_2 \ : \ f_1 \in L(E_1), f_2 \in L(E_2)\}$, ただし $f_1 \uplus f_2$ は $(f_1 \uplus f_2)(\alpha) = f_1(\alpha) + f_2(\alpha)$.
- $L(E^*) := \{\bar{\epsilon}\} \cup \{f_0 \uplus \cdots \uplus f_n \ : \ n \geq 0, f_i \in L(E)\}$，ただし $\bar{\epsilon}$ は定数値 0 を持つ多重集合.
  E^* は E に含まれる多重集合の有限連結 $\|$ によって表現できる全ての多重集合の集合である.

$L(E)$ の直感的なイメージを得るために, 次のような多重集合の表記法が役立つ．f を $f(\alpha) = 2, f(\beta) = 3, f(x) = 0(x \neq \alpha, \beta)$ であるとすると，f は $\{\!|\alpha, \alpha, \beta, \beta, \beta|\!\}$ と表記できる．

例 3. $\Sigma \times \mathrm{T}$ に対する RBE E :

$$E = (\alpha_0 \mid \alpha_1) \parallel \alpha_2,$$

ただし $\alpha_0, \alpha_1, \alpha_2 \in \Sigma \times \mathrm{T}$．このとき，

$$L(E) = \{\{\!|\alpha_0, \alpha_2|\!\}, \{\!|\alpha_1, \alpha_2|\!\}\}.$$

シンボルに対する多重集合がベクトルとして表示される場合（ベクトルの i 番目の座標は，i 番目のシンボルの出現回数に対応する），RBE のクラスは，プレスバーガー算術で定義可能なベクトルのクラスと一致する [6, 7].

2.3　Shape Expression Schema

$\mathcal{E}$ を $\Sigma \times \mathrm{T}$ 上の全ての RBE のクラスとする．関数 $e : \mathrm{T} \to \mathcal{E}$ は型の規則関数と呼ばれる．このような e に対して，3 次組 (Σ, T, e) は Shape Expression Schema（ShEx）と呼ばれる．もし $e(t) = E$（t は型，E は RBE）であるならば，この状況を表現するために $t \to E$ と表記する．

例 4. ShEx の例 : $\Sigma = \{a\}, \mathrm{T} = \{t_0, t_1\}$ とし，e を次のように定義される型の規則関数とする．このとき，(Σ, T, e) は ShEx である．

$$t_0 \to a :: t_1, t_1 \to \epsilon$$

全てのノードにスキーマの型の定義を満たすように型を割り当てることができる場合，グラフはスキーマに対して妥当であるという．ノードに複数の型を割り当てることができるかどうかによって，single-type と multi-type の 2 つのセマンティックスが考えられる．本研究では single-type を扱うため，以下では single-type について説明する．

2.4　Single Type Semantics

定義 5. G を Σ-ラベル付き有向グラフとする．

1. ノードに型を割り当てる関数 $\tau : N(G) \to \mathrm{T}$ は，G 上の single-type typing（または簡単に s-typing）と呼ばれる．
   (G, τ) を (Σ, T)-ラベル付き有向グラフと呼ぶ．これは，ノードが T の中のシンボルでラベル付け（タイプ付け）されているグラフと見なすことができる．
2. (G, τ) を (Σ, T)-ラベル付き有向グラフとする．τ に関する n の出力近傍を，$\Sigma \times \mathrm{T}$ に対する多重集合として次のように定義する :

   $$out\_lab\_type^{\tau}_{G}(n) = \{\!|a :: \tau(n') \mid (n, a, n') \in E(G)|\!\}$$

   n の出力近傍が $\Sigma \times \mathrm{T}$ に対する関数 (多重集合) として表される時，(a, t) における値は，エッジ $(n, a, n')(\tau(n') = t)$ の数を意味する．

3. (G,τ) を (Σ,T)-ラベル付き有向グラフとする．3 次組 (Σ,T,e) を ShEx とする．もし全ての $n \in N(G)$ について
$$out\_lab\_type^{\tau}_{G}(n) \in L(e(\tau(n)))$$
が成り立つならば，(G,τ) は (Σ,T,e) を満たすという．この時，τ は (Σ,T,e) に対する G の妥当な s-typing であるという．

例 6. $G = (N(G), E(G))$ を以下によって定義される Σ-ラベル付き有向グラフとする．

1. $N(G) = \{n_0, n_1\}$
2. $E(G) = \{(n_0, a, n_1)\}$

以下の図がこのグラフの状況を示している：

$$n_0 \xrightarrow{\quad a \quad} n_1$$

ShEx (Σ,T,e) を考える．ただし，$\Sigma = \{a\}, \mathrm{T} = \{t_0, t_1\}$ かつ e は次のように定義される．

$$t_0 \to a :: t_1, t_1 \to \epsilon,$$

$\tau : N(G) \to \{t_0, t_1\}$ を次のような s-typing とする：

$$\tau(n_0) = t_0, \tau(n_1) = t_1.$$

n_0 は t_0，n_1 は t_1 でそれぞれラベル付けられており，n_0 と n_1 は以下のように割り当てられた型の定義を満たしているので，τ は G 上の妥当な (Σ,T,e) の s-typing である．

$out\_lab\_type^{\tau}_{G}(n_0) = \{\!|a :: t_1|\!\}$ かつ $L(e(t_0)) = \{\{\!|a :: t_1|\!\}\}$ である．従って，$out\_lab\_type^{\tau}_{G}(n_0)$ は $L(e(t_0))$ の要素である．$out\_lab\_type^{\tau}_{G}(n_1) = \bar{\epsilon}$ かつ $L(e(t_1)) = \{\bar{\epsilon}\}$ である．従って，$out\_lab\_type^{\tau}_{G}(n_1)$ は $L(e(t_1))$ の要素である．

例 7. Σ-ラベル付き有向グラフとして G_2（図 2.1）を考える．

ShEx (Σ,T,e) を考える．ただし $\Sigma = \{a,b,c\}, \mathrm{T} = \{t_0,t_1,t_2\}$ かつ，e は次のように定義される．

$$t_0 \to a :: t_1, t_1 \to b :: t_1 \parallel c :: t_2, t_2 \to \epsilon$$

$\tau_2 : N(G) \to \{t_0, t_1, t_2\}$ を次のような s-typing とする：

$$\tau_2(n_0) = t_0, \tau_2(n_1) = t_1, \tau_2(n_2) = t_2$$

n_0 は t_0 で n_1 は t_1 で n_2 は t_2 でそれぞれラベル付けられており，n_0, n_1, n_2 は以下のように割り当てられた型の定義を満たしている．従って，τ_2 は G_2 上の妥当な (Σ,T,e) の s-typing である．

$out\_lab\_type^{\tau_2}_{G_2}(n_0) = \{\!|a :: t_1|\!\}$ かつ $L(e(t_0)) = \{\{\!|a :: t_1|\!\}\}$ である．従って，$out\_lab\_type^{\tau_2}_{G_2}(n_0)$ は $L(e(t_0))$ の要素である．$out\_lab\_type^{\tau_2}_{G_2}(n_1) = \{\!|b :: t_1, c :: t_2|\!\}$ かつ $L(e(t_1)) = \{\{\!|b :: t_1, c :: t_2|\!\}\}$ である．従って，$out\_lab\_type^{\tau_2}_{G_2}(n_1)$ は $L(e(t_1))$ の要素である．$out\_lab\_type^{\tau_2}_{G_2}(n_2) = \bar{\epsilon}$ かつ $L(e(t_2)) = \{\bar{\epsilon}\}$ である．従って，$out\_lab\_type^{\tau_2}_{G_2}(n_2)$ は $L(e(t_2))$ の要素である．

ただし、G は (Σ, T, e) を満たす一意のグラフではない．たとえば，H が G の 2 つの互いに素なコピーの和集合である場合，H は e を満たす．H は連結（グラフ上の任意の 2 つのノード間に道が存在）でない．

定義 8. G を Σ-ラベル付き有向グラフ，$S = (\Sigma, \mathrm{T}, e)$ を ShEx とする．(G, τ) が S を満たすような s-typing $\tau : N(G) \to \mathrm{T}$ が存在する時，G は S を満たす（または G は S に対して妥当）と言う．

第 3 章

提案手法

グラフデータから ShEx スキーマを抽出するアルゴリズムを提案する．ShEx の定義では，型の規則関数としてノードの外側（出力）近傍のみを規定することができる．しかしながら，ノードの外側（出力）近傍だけでなく内側（入力）近傍も考慮に入れた上で ShEx を生成することで，より精度の高い ShEx を生成することができるようにする．

本研究のアルゴリズムは主に 2 つのステップに分けられる．

Step1　全てのノード $n \in N(G)$ に型 $t \in \mathrm{T}$ を割り当てる．
ノード $n \in N(G)$ の内・外側近傍を基に，類似したノードに同じ型を割り当てることで，全てのノード $n \in N(G)$ に型 $t \in \mathrm{T}$ を割り当てる関数 s-typing τ を得る．

Step2　Step1 で得られた s-typing τ を基に，G が妥当となるような ShEx を生成する．
G が ShEx (Σ, T, e) を満たすようにするために，次のようになるようにする．全てのノード $n \in N(G)$ に対して，$out\_lab\_type^{\tau}_{G}(n) \in L(e(\tau(n)))$.
型 t_i に対して，型 t_i が割り当てられている全てのノードの $out\_lab\_type^{\tau}_{G}(n)$ が $L(E)$ の要素となるような RBE E を生成し，$t_i \to E$ とする．

3.1　Step1

1. 関数 s-typing τ の初期値を得る．全てのノード $n \in N(G)$ に対して，異なる型 $\tau(n)$ を割り当てる（ただし，葉ノードは全て同じ型）．
2. 次に，τ に関するノード $n \in N(G)$ の内・外側近傍を考える．τ に関する n の内側（入力）近傍を，$\mathrm{T} \times \Sigma$ に対する多重集合として次のように定義する：

$$in\_lab\_type^{\tau}_{G}(n) = \{\!|\tau(n') :: a \mid (n', a, n) \in E(G)|\!\}$$

τ に関する n の外側（出力）近傍は，$\Sigma \times \mathrm{T}$ に対する多重集合として前章で次のように定義されている：

$$out\_lab\_type^{\tau}_{G}(n) = \{\!|a :: \tau(n') \mid (n, a, n') \in E(G)|\!\}$$

従って，(G, τ) において，τ に関するノード $n \in N(G)$ の入・出力近傍を，$(\mathrm{T} \times \Sigma) \cup (\Sigma \times \mathrm{T})$ に

対する多重集合として以下のように定義する．

$$neighborhood^{\tau}_{G}(n) = in\_lab\_type^{\tau}_{G}(n) \uplus out\_lab\_type^{\tau}_{G}(n)$$

例えば，例 7 の (G_2, τ_2) において，$neighnorhood^{\tau_2}_{G_2}(n_1) = \{\!|t_0 :: a|\!\} \uplus \{\!|b :: t_1, c :: t_2|\!\} = \{\!|t_0 :: a, b :: t_1, c :: t_2|\!\}$.

3. 型間の類似度を求めるため，多重集合をベクトルとして表現する．多重集合はシンボルをその出現回数に対応させる関数である．多重集合 f に対するベクトルを v_f とする．v_f の i 番目の要素は，i 番目のシンボルの出現回数を意味するものとする．従って，i 番目のシンボルが α である場合，i 番目の要素は α の出現回数 $f(\alpha)$ になるので，$v_f(i) = f(\alpha)$ となる．
   具体的には，$\Sigma = \{a, b\}, \mathrm{T} = \{t_0\}$ とし，$(\mathrm{T} \times \Sigma) \cup (\Sigma \times \mathrm{T})$ に対する多重集合 f を $\{\!|a :: t_0, a :: t_0, b :: t_0|\!\}$ であるとすると，

$$\begin{array}{rcl} & & \begin{array}{cccc} t_0 :: a & t_0 :: b & a :: t_0 & b :: t_0 \end{array} \\ v_f & = & \left(\begin{array}{cccc} f(t_0 :: a) & f(t_0 :: b) & f(a :: t_0) & f(b :: t_0) \end{array} \right) \\ & & \begin{array}{cccc} t_0 :: a & t_0 :: b & a :: t_0 & b :: t_0 \end{array} \\ & = & \left(\begin{array}{cccc} 0 & 0 & 2 & 1 \end{array} \right) \end{array}$$

   ここで，多重集合 f_1, f_2 の間の距離を以下のように定義する．

$$< f_1, f_2 >= \sqrt{\sum_{i \in \mathbb{N}} (df(i))^2},$$

   ただし

$$df(i) = \begin{cases} (v_{f_1}(i) - v_{f_2}(i))/2 & (\text{もし } v_{f_1}(i) \neq 0 \text{ かつ } v_{f_2}(i) \neq 0 \text{ ならば}) \\ v_{f_1}(i) - v_{f_2}(i) & (\text{そうでないならば}). \end{cases}$$

4. 任意の型の組 $(t_1, t_2) \in \mathrm{T} \times \mathrm{T}$ に対して，型 t_1 と型 t_2 の距離を以下のように定義する．

$$distance(t_1, t_2) = \frac{\sum_{n_1 \in n_{t_1}} \sum_{n_2 \in n_{t_2}} < neighborhood^{\tau}_{G}(n_1), neighborhood^{\tau}_{G}(n_2) >}{|n_{t_1}||n_{t_2}|}$$

   ただし $n_{t_i} = \{n \mid \tau(n) = t_i, n \in N(G)\}$.
   任意の型の組 $(t_1, t_2) \in \mathrm{T} \times \mathrm{T}$ の中から距離 $distance(t_1, t_2)$ が最も小さくなる組を求める．

$$(t_1, t_2) = \underset{(t_1, t_2) \in \mathrm{T} \times \mathrm{T}}{\arg\min} \; distance(t_1, t_2)$$

5. 実際に型 t_1, t_2 をマージする．

$$t_1 \leftarrow \{t_1, t_2\}$$

6. s-typing τ を変更する．τ が τ' に変化したとする．型 t_1 と型 t_2 がマージされて型 t_1 に集約されたので，型 t_2 が割り当てられているノードに対して，型 t_1 を割り当てるように変更する．全てのノード $n \in N(G)$ に対して，もし $\tau(n) = t_2$ ならば $\tau'(n) = t_1$ とする．
7. 型の定義を変更する．型 t_1 と型 t_2 がマージされて型 t_1 に集約されたとする．これにより，今まで型 t_2 が割り当てられていたノードに型 t_1 が割り当て割られるようになった．従って，それに合わせて全てのノード $n \in N(G)$ の入出力近傍も変化する．全ての $a \in \Sigma$ に対して，シンボルの出現回数は以下のように変化する．

- シンボル $a :: t_1$ の出現回数 $neighborhood_G^{\tau'}(n)(a :: t_1)$ は，シンボル $a :: t_1$ の出現回数 $neighborhood_G^{\tau}(n)(a :: t_1)$ とシンボル $a :: t_2$ の出現回数 $neighborhood_G^{\tau}(n)(a :: t_2)$ の合計となる．

$$neighborhood_G^{\tau'}(n)(a :: t_1) = neighborhood_G^{\tau}(n)(a :: t_1) + neighborhood_G^{\tau}(n)(a :: t_2)$$

- シンボル $a :: t_2$ の出現回数 $neighborhood_G^{\tau'}(n)(a :: t_2)$ は 0 になる．

$$neighborhood_G^{\tau'}(n)(a :: t_2) = 0$$

- シンボル $t_1 :: a$ の出現回数 $neighborhood_G^{\tau'}(n)(t_1 :: a)$ は，シンボル $a :: t_1$ の出現回数 $neighborhood_G^{\tau}(n)(t_1 :: a)$ とシンボル $t_2 :: a$ の出現回数 $neighborhood_G^{\tau}(n)(t_2 :: a)$ の合計となる．

$$neighborhood_G^{\tau'}(n)(t_1 :: a) = neighborhood_G^{\tau}(n)(t_1 :: a) + neighborhood_G^{\tau}(n)(t_2 :: a)$$

- シンボル $t_2 :: a$ の出現回数 $neighborhood_G^{\tau'}(n)(t_2 :: a)$ は 0 になる．

$$neighborhood_G^{\tau'}(n)(t_2 :: a) = 0$$

- それ以外のシンボルの出現回数は変化しない．

例えば，多重集合 $neighborhood_G^{\tau}(n) = \{\!|a :: t_1, a :: t_2, b :: t_2, t_1 :: a|\!\}$ とすると，多重集合 $neighborhood_G^{\tau'}(n) = \{\!|a :: t_1, a :: t_1, b :: t_1, t_1 :: a|\!\}$ となる．
以上の手順をアルゴリズムとして記述したものを Algorithm 1 に示す．

Algorithm 1 スキーマ抽出

Input: グラフ $G = (N(G), E(G))$，型の数 k

Output: s-typing τ

1: 各ノード $n \in N(G)$ に異なる型 $\tau(n)$ を割り当てる（ただし，葉ノードは全て同じ型）
2: 各ノード $n \in N(G)$ に対し，n の入出力近傍に対応した多重集合 $neighborhood_G^\tau(n)$ を得る．
3: **loop**
4:　　**for all** 任意の型の組 $(t_1, t_2) \in \mathrm{T} \times \mathrm{T}$ に対して **do**
5:　　型 t_1, t_2 の距離 $distance(t_1, t_2)$ を計算する
6:　　**end for**
7:　　距離 $distance(t_1, t_2)$ が一番小さい型の組 (t_1, t_2) を選ぶ
8:　　**if** もし型の数が k より大きければ **then**
9:　　/* s-typing τ を変更する */ 全ての $n \in N(G)$ について，$\tau(n) = t_2$ ならば $\tau'(n) = t_1$
10:　　/*ノード $n \in N(G)$ の入出力近傍が変更される*/ $neighborhood_G^\tau(n)$ が $neighborhood_G^{\tau'}(n)$ になる．
11:　　/* 実際に型 (t_1, t_2) をマージして型 t_2 を削除する*/ $\mathrm{T} \leftarrow \mathrm{T} \setminus \{t_2\}$
12:　　**else**
13:　　繰り返しの終了
14:　　**end if**
15: **end loop**
16: **return** s-typing τ

3.2　Step2

次に Step1 で得た s-typing τ を基に，型の規則関数 e を生成し，ShEx を生成する．s-typing τ に対して，$out\_lab\_type_G^\tau(n)(n \in N(G))$ が得られる．ここで，生成された ShEx (Σ, T, e) に G が妥当であるようにするために，全てのノード $n \in N(G)$ に対して，$out\_lab\_type_G^\tau(n)$ が $L(e(\tau(n)))$ の要素になるようにする．

各型 $t_i \in \mathrm{T}$ に対して，$B_G^\tau(t_i)$ を多重集合の集合として次のように定義する．

$$B_G^\tau(t_i) = \{out\_lab\_type_G^\tau(n) \mid \tau(n) = t_i, n \in N(G)\}$$

ここで，全ての $f_i \in B_G^\tau(t_i)$ に対して $f_i \in L(E)$ となるような RBE E を生成し，$t_i \to E$ とする．

多重集合同士の共通部分 $f_1 \cap f_2$ を次のように定義する．

$$(f_1 \cap f_2)(\alpha) = \min\{f_1(\alpha), f_2(\alpha)\}$$

また，多重集合同士の差部分 $f_1 \setminus f_2$ は次のように定義する．

$$(f_1 \setminus f_2)(\alpha) = \begin{cases} f_1(\alpha) - f_2(\alpha) & (\text{もし } f_1(\alpha) \geq f_2(\alpha) \text{ ならば}) \\ 0 & (\text{そうでないならば}) \end{cases}$$

と定義する.

ここで，多重集合の集合 M の共通集合 I_M と差集合 D_M を以下のように定義する.

$$I_M = \{i_M\},$$

ただし $i_M = \bigcap_{f_i \in M} f_i$.

$$D_M = \{f_i \setminus i_M \mid f_i \in M\}$$

多重集合 f_1 を $\{\!|\alpha, \alpha, \beta|\!\}$, 多重集合 f_2 を $\{\!|\alpha, \alpha, \gamma|\!\}$ とする．ここで $B^\tau_G(t_i) = \{f_1, f_2\}$ とする．$i_{B^\tau_G(t_i)} = f_1 \cap f_2 = \{\!|\alpha, \alpha|\!\}$ であるので，$I_{B^\tau_G(t_i)} = \{i_{B^\tau_G(t_i)}\} = \{\{\!|\alpha, \alpha|\!\}\}$．$f_1 \setminus i_{B^\tau_G(t_i)} = \{\!|\beta|\!\}, f_2 \setminus i_{B^\tau_G(t_i)} = \{\!|\gamma|\!\}$ であるので，$D_{B^\tau_G(t_i)} = \{\{\!|\beta|\!\}, \{\!|\gamma|\!\}\}$.

L^{-1} を，$\Sigma \times \mathrm{T}$ 上の多重集合の集合を RBE に対応させる関数とする．$\Sigma \times \mathrm{T}$ 上の多重集合の集合 M に対して，$L^{-1}(M)$ を次のように定義する.

$$L^{-1}(M) = \|_{f_i \in M} \|_{\alpha \in f_i}\ \alpha$$

例えば多重集合の集合 $M = \{\{\!|\alpha, \alpha, \beta|\!\}, \{\!|\alpha, \alpha, \gamma|\!\}\}$ とすると，$L^{-1}(M) = (\alpha \parallel \alpha \parallel \beta) \mid (\alpha \parallel \alpha \parallel \gamma)$.

更に，$RBE(M)$ を以下のように表現することとする：

$$RBE(M) := \begin{cases} \epsilon & (I_M = \{\bar{\epsilon}\} \text{ かつ } D_M = \{\bar{\epsilon}\}) \\ L^{-1}(I_M) & (I_M \neq \{\bar{\epsilon}\} \text{ かつ } D_M = \{\bar{\epsilon}\}) \\ L^{-1}(D_M) & (I_M = \{\bar{\epsilon}\} \text{ かつ } D_M \neq \{\bar{\epsilon}\}) \\ L^{-1}(I_M) \parallel L^{-1}(D_M) & (I_M \neq \{\bar{\epsilon}\} \text{ かつ } D_M \neq \{\bar{\epsilon}\}) \end{cases}$$

例えば $B^\tau_G(t_i) = \{\{\!|\alpha, \alpha, \beta|\!\}, \{\!|\alpha, \alpha, \gamma|\!\}\}$ は，$I_{B^\tau_G(t_i)} = \{\{\!|\alpha, \alpha|\!\}\}$ で $D_{B^\tau_G(t_i)} = \{\{\!|\beta|\!\}, \{\!|\gamma|\!\}\}$ であるので，$RBE(B^\tau_G) = \alpha \parallel \alpha \parallel (\beta \mid \gamma)$ となる．従って，$B^\tau_G(t_i) = \{\{\!|\alpha, \alpha, \beta|\!\}, \{\!|\alpha, \alpha, \gamma|\!\}\}$ である時，$t_i \to \alpha \parallel \alpha \parallel (\beta \mid \gamma)$ とする.

以上のように，各型 $t_i \in \mathrm{T}$ に対して，$RBE(B^\tau_G(t_i))$ を求め，$t_i \to RBE(B^\tau_G(t_i))$ とすることで型の規則関数 e を定める.

3.3　アルゴリズムの動作例

以下の Σ-ラベル付き有向グラフ（図 3.1）に対してアルゴリズムを適用することを考える.

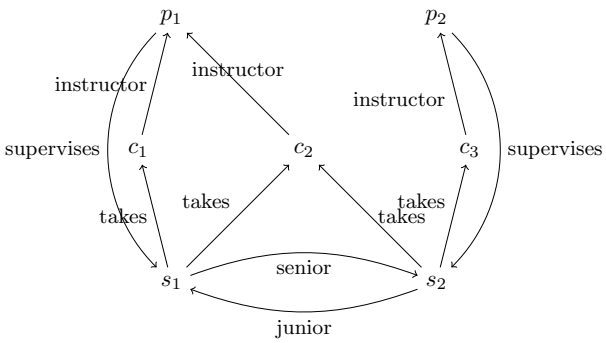
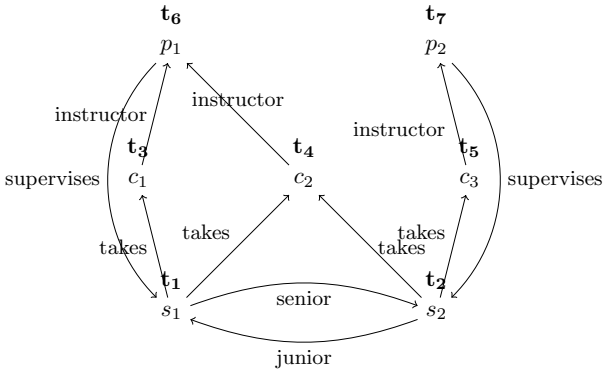


図 3.1　Σ-ラベル付き有向グラフ

3.3.1　Step1

- それぞれのノードに異なる型を割り当てる.
  従って s-typing τ は以下のようになる.

$$\tau(s_1) = t_1, \tau(s_2) = t_2, \tau(c_1) = t_3, \tau(c_2) = t_4, \tau(c_3) = t_5, \tau(p_1) = t_6, \tau(p_2) = t_7$$



- 入出力近傍を考える.
  - $neighborhood^{\tau}_{G}(s_1) = \{\!|t_2 :: junior, t_6 :: supervises|\!\} \uplus \{\!|takes :: t_3, takes :: t_4, senior :: t_2|\!\} = \{\!|t_2 :: junior, t_6 :: supervises, takes :: t_3, takes :: t_4, senior :: t_2|\!\}$
  - $neighborhood^{\tau}_{G}(s_2) = \{\!|t_1 :: senior, t_7 :: supervises|\!\} \uplus \{\!|takes :: t_4, takes :: t_5, junior :: t_1|\!\} = \{\!|t_1 :: senior, t_7 :: supervises, takes :: t_4, takes :: t_5, junior :: t_1|\!\}$
  - $neighborhood^{\tau}_{G}(c_1) = \{\!|t_1 :: takes|\!\} \uplus \{\!|instructor :: t_6|\!\} = \{\!|t_1 :: takes, instructor :: t_6|\!\}$
  - $neighborhood^{\tau}_{G}(c_2) = \{\!|t_1 :: takes, t_2 :: takes|\!\} \uplus \{\!|instructor :: t_6|\!\} = \{\!|t_1 :: takes, t_2 :: takes, instructor :: t_6|\!\}$
  - $neighborhood^{\tau}_{G}(c_3) = \{\!|t_2 :: takes|\!\} \uplus \{\!|instructor :: t_7|\!\} = \{\!|t_2 :: takes, instructor :: t_7|\!\}$
  - $neighborhood^{\tau}_{G}(p_1) = \{\!|t_3 :: instructor, t_4 :: instructor|\!\} \uplus \{\!|supervises :: t_1|\!\} = \{\!|t_3 :: instructor, t_4 :: instructor, supervises :: t_1|\!\}$

- $neighborhood^{\tau}_{G}(p_2) = \{\!|t_5 :: instructor|\!\} \uplus \{\!|supervises :: t_2|\!\} = \{\!|t_5 :: instructor, supervises :: t_2|\!\}$

- 型の類似度

  類似度を計算した結果を表 3.1 に示す.

表 3.1　距離

型の組	距離
t_1, t_3	2.6457513110645907
t_1, t_4	2.8284271247461903
t_1, t_2	2.8284271247461903
t_1, t_5	2.6457513110645907
t_1, t_6	2.8284271247461903
t_1, t_7	2.6457513110645907
t3,t4	**1.0**
t_3, t_2	2.6457513110645907
t_3, t_5	2.0
t_3, t_6	2.23606797749979
t_3, t_7	2.0
t_4, t_2	2.8284271247461903
t_4, t_5	1.7320508075688772
t_4, t_6	2.449489742783178
t_4, t_7	2.23606797749979
t_2, t_5	2.6457513110645907
t_2, t_6	2.8284271247461903
t_2, t_7	2.6457513110645907
t_5, t_6	2.23606797749979
t_5, t_7	2.0
t_6, t_7	2.23606797749979

  型と型の距離は t_3, t_4 が最も小さいので，実際に型 t_3 と型 t_4 をマージする.

- s-typing τ を変更する.

$$\tau(s_1) = t_1, \tau(s_2) = t_2, \tau(c_1) = t_3, \tau(c_2) = \mathbf{t_3}, \tau(c_3) = t_5, \tau(p_1) = t_6, \tau(p_2) = t_7$$

- 入出力近傍を考える.
  - $neighborhood^{\tau}_{G}(s_1) = \{\!|t_2 :: junior, t_6 :: supervises, takes :: t_3, takes :: \mathbf{t_3}, senior :: t_2|\!\}$
  - $neighborhood^{\tau}_{G}(s_2) = \{\!|t_1 :: senior, t_7 :: supervises, takes :: \mathbf{t_3}, takes :: t_5, junior :: t_1|\!\}$
  - $neighborhood^{\tau}_{G}(c_1) = \{\!|t_1 :: takes, instructor :: t_6|\!\}$
  - $neighborhood^{\tau}_{G}(c_2) = \{\!|t_1 :: takes, t_2 :: takes, instructor :: t_6|\!\}$
  - $neighborhood^{\tau}_{G}(c_3) = \{\!|t_2 :: takes, instructor :: t_7|\!\}$
  - $neighborhood^{\tau}_{G}(p_1) = \{\!|t_3 :: instructor, \mathbf{t_3} :: instructor, supervises :: t_1|\!\}$
  - $neighborhood^{\tau}_{G}(p_2) = \{\!|t_5 :: instructor, supervises :: t_2|\!\}$

- 距離を測る

距離を計算した結果を表 3.2 に示す.

表 3.2　距離

型の組	距離
t_1, t_3	3.08113883008419
t_1, t_2	2.692582403567252
t_1, t_5	3.0
t_1, t_6	3.4641016151377544
t_1, t_7	3.0
t_3, t_2	2.7370892179053907
t3,t5	**1.8660254037844386**
t_3, t_6	2.7370892179053907
t_3, t_7	2.118033988749895
t_2, t_5	2.6457513110645907
t_2, t_6	3.1622776601683795
t_2, t_7	2.6457513110645907
t_5, t_6	2.6457513110645907
t_5, t_7	2.0
t_6, t_7	2.6457513110645907

型と型の距離は t_3, t_5 が最も小さいので，実際に型 t_3 と型 t_5 をマージする.

- s-typing τ を変更する.

$$\tau(s_1) = t_1, \tau(s_2) = t_2, \tau(c_1) = t_3, \tau(c_2) = t_3, \tau(c_3) = \mathbf{t_3}, \tau(p_1) = t_6, \tau(p_2) = t_7$$

- 入出力近傍を考える.
  - $neighborhood_G^{\tau}(s_1) = \{\!|t_2 :: junior, t_6 :: supervises, takes :: t_3, takes :: t_3, senior :: t_2|\!\}$
  - $neighborhood_G^{\tau}(s_2) = \{\!|t_1 :: senior, t_7 :: supervises, takes :: t_3, takes :: \mathbf{t_3}, junior :: t_1|\!\}$
  - $neighborhood_G^{\tau}(c_1) = \{\!|t_1 :: takes, instructor :: t_6|\!\}$
  - $neighborhood_G^{\tau}(c_2) = \{\!|t_1 :: takes, t_2 :: takes, instructor :: t_6|\!\}$
  - $neighborhood_G^{\tau}(c_3) = \{\!|t_2 :: takes, instructor :: t_7|\!\}$
  - $neighborhood_G^{\tau}(p_1) = \{\!|t_3 :: instructor, t_3 :: instructor, supervises :: t_1|\!\}$
  - $neighborhood_G^{\tau}(p_2) = \{\!|\mathbf{t_3} :: instructor, supervises :: t_2|\!\}$
- 距離を測る

  距離を計算した結果を表 3.3 に示す.

  型と型の距離は t_6, t_7 が最も小さいので，実際に型 t_6 と型 t_7 をマージする.

表 3.3　距離

型の組	距離
t_1, t_3	3.05409255338946
t_1, t_2	2.449489742783178
t_1, t_6	3.4641016151377544
t_1, t_7	3.0
t_3, t_2	3.05409255338946
t_3, t_6	2.706643248958457
t_3, t_7	2.0786893258332633
t_2, t_6	3.4641016151377544
t_2, t_7	3.0
t6,t7	**1.5**

- s-typing τ を変更する.

$$\tau(s_1) = t_1, \tau(s_2) = t_2, \tau(c_1) = t_3, \tau(c_2) = t_3, \tau(c_3) = t_3, \tau(p_1) = t_6, \tau(p_2) = \mathbf{t_6}$$

- 入出力近傍を考える.
  - $neighborhood^{\tau}_{G}(s_1) = \{\!|t_2 :: junior, t_6 :: supervises, takes :: t_3, takes :: t_3, senior :: t_2|\!\}$
  - $neighborhood^{\tau}_{G}(s_2) = \{\!|t_1 :: senior, \mathbf{t_6} :: supervises, takes :: t_3, takes :: t_3, junior :: t_1|\!\}$
  - $neighborhood^{\tau}_{G}(c_1) = \{\!|t_1 :: takes, instructor :: t_6|\!\}$
  - $neighborhood^{\tau}_{G}(c_2) = \{\!|t_1 :: takes, t_2 :: takes, instructor :: t_6|\!\}$
  - $neighborhood^{\tau}_{G}(c_3) = \{\!|t_2 :: takes, instructor :: \mathbf{t_6}|\!\}$
  - $neighborhood^{\tau}_{G}(p_1) = \{\!|t_3 :: instructor, t_3 :: instructor, supervises :: t_1|\!\}$
  - $neighborhood^{\tau}_{G}(p_2) = \{\!|t_3 :: instructor, supervises :: t_2|\!\}$
- 距離を測る

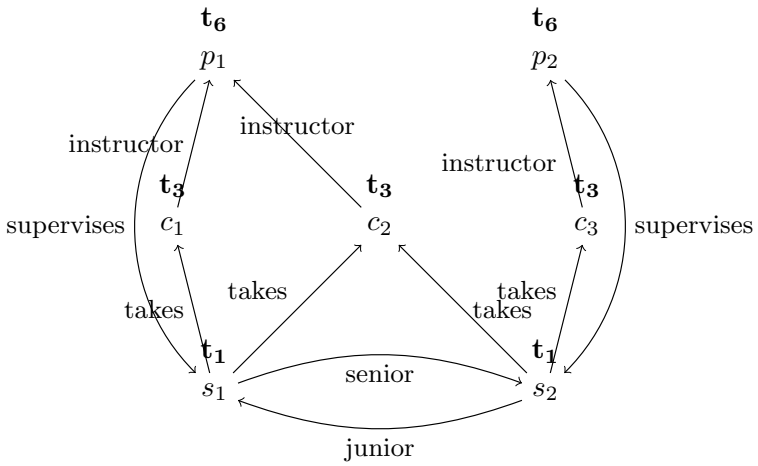
  距離を計算した結果を表 3.4 に示す.

表 3.4　距離

型の組	距離
t_1, t_3	3.05409255338946
t1,t2	**2.0**
t_1, t_6	3.232050807568877
t_3, t_2	3.05409255338946
t_3, t_6	2.39266628739586
t_2, t_6	3.232050807568877

  型と型の距離は t_1, t_2 が最も小さいので，実際に型 t_1 と型 t_2 をマージする.
- s-typing τ を変更する.

$$\tau(s_1) = t_1, \tau(s_2) = \mathbf{t_1}, \tau(c_1) = t_3, \tau(c_2) = t_3, \tau(c_3) = t_3, \tau(p_1) = t_6, \tau(p_2) = t_6$$



3.3.2　Step2

Step1 により次のような s-typing τ が得られた.

$$\tau(s_1) = t_1, \tau(s_2) = t_1, \tau(c_1) = t_3, \tau(c_2) = t_3, \tau(c_3) = t_3, \tau(p_1) = t_6, \tau(p_2) = t_6$$

ここで Step1 で得られた s-typing τ を基に，型の規則関数 e を生成し，ShEx を生成する.

- $B^{\tau}_{G}(t_1) = \{\{\!|takes :: t_3, takes :: t_3, senior :: t_1|\!\}, \{\!|takes :: t_3, takes :: t_3, junior :: t_1|\!\}\}$ である. 従って，$I_{B^{\tau}_{G}(t_1)} = \{\{\!|takes :: t_3, takes :: t_3|\!\}\}$ かつ $D_{B^{\tau}_{G}(t_1)} = \{\{\!|senior :: t_1|\!\}, \{\!|junior :: t_1|\!\}\}$ であるので，$RBE(B^{\tau}_{G}(t_1)) = takes :: t_3 \parallel takes :: t_3 \parallel (senior :: t_1 \mid junior :: t_1)$. 従って，$t_1 \to takes :: t_3 \parallel takes :: t_3 \parallel (senior :: t_1 \mid junior :: t_1)$.
- $B^{\tau}_{G}(t_3) = \{\{\!|instructor :: t_6|\!\}\}$ である. 従って，$I_{B^{\tau}_{G}(t_3)} = \{\{\!|instructor :: t_6|\!\}\}$ かつ $D_{B^{\tau}_{G}(t_3)} = \{\bar{\epsilon}\}$ であるので，$RBE(B^{\tau}_{G}(t_3)) = instructor :: t_6$. 従って，$t_3 \to instructor :: t_6$.
- $B^{\tau}_{G}(t_6) = \{\{\!|supervises :: t_1|\!\}\}$ である. 従って，$I_{B^{\tau}_{G}(t_6)} = \{\{\!|supervises :: t_1|\!\}\}$ かつ $D_{B^{\tau}_{G}(t_6)} = \{\bar{\epsilon}\}$ であるので，$RBE(B^{\tau}_{G}(t_1)) = supervises :: t_1$. 従って，$t_6 \to supervises :: t_1$.

よって，提案手法で抽出した ShEx (Σ, T, e) は以下のようになる.

$\Sigma = \{takes, instructor, supervises, senior, junior\}$，$\mathrm{T} = \{t_1, t_3, t_6\}$ であり，型の規則関数 e は以下のようになる.

$$t_1 \to takes :: t_3 \parallel takes :: t_3 \parallel (senior :: t_1 \mid junior :: t_1)$$
$$t_3 \to instructor :: t_6$$
$$t_6 \to supervises :: t_1$$

3.4　アルゴリズムの効率化

Step1 の Algorithm 1 は，まず最初に全てのノードに対して異なる型を割り当てているため，型の数が多くなり処理に時間を要することが予想される．そこで，初期段階として全てのノードに対して異なる型を割り当てるのではなく，各ノードの入出力ラベル集合を基にクラスタリングを行い，同じクラスタに分類されたノードに対しては同じ型を割り当てることにする．これ以降の処理は Step1，Step2 共に同様である．

第 4 章

スキーマ抽出の NP 完全性

グラフから k 個の型を抽出する場合，得られた型間の距離 $distance(t_i, t_j)$ ができるだけ大きくなるようにスキーマを抽出することが望ましい．しかし，この最適解を得る問題（ShEx スキーマ抽出問題）は NP 完全であることを示す．

定理 9. *ShEx スキーマ抽出問題は NP 完全である．*

証明．まず，ShEx スキーマ S が与えられた時に，S が ShEx スキーマ抽出問題の条件を満たすか否かは多項式時間て確かめることができる．よって，ShEx スキーマ抽出問題は NP に属する．

次に，ShEx スキーマ抽出問題の NP 困難性を示す．3-PARTITION 問題は NP 完全である．この問題を，ShEx スキーマ抽出問題に帰着する．ここで，3-PARTITION 問題は次のように定義される．

入力： $3k$ 個の整数 $a_1, a_2, \cdots, a_{3k}$．ただし，$\sum_{i=1}^{3k} a_i = kb$ かつ任意の i に対して $\frac{b}{4} < a_i < \frac{b}{2}$.

問題： この問題の解，すなわち，次の 2 条件を満たす k 個の集合 $S_1, S_2, \cdots, S_k$ が存在するかどうかを決定せよ．

- $S_1, S_2, \cdots, S_k$ は $\{a_1, a_2, \cdots, a_{3k}\}$ の分割である．すなわち，$\bigcup_{1 \le i \le k} S_i = \{a_1, a_2, \cdots, a_{3k}\}$，かつ，任意の $i \neq j$ に対して $S_i \cap S_j = \emptyset$
- 任意の S_i に対して $\sum_{a_j \in S_i} a_j \le b$

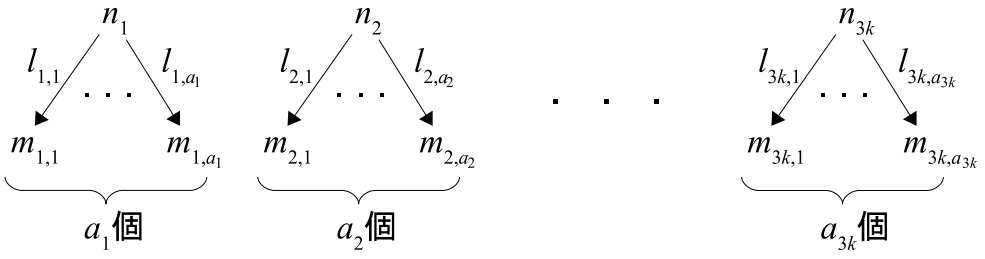


図 4.1　グラフ G

3-PARTITON 問題のインスタンスから，図 4.1 に示すグラフ $G = (V, E)$ を構成する．ここで，

$$V = N \cup \bigcup_{1 \leq i \leq 3k} M_i$$
$$N = \{n_1, n_2, \cdots, n_{3k}\}$$
$$M_i = \{m_{i,1}, m_{i,2}, \cdots, m_{i,a_i}\}$$

かつ

$$E = \bigcup_{1 \leq i \leq 3k} E_i$$
$$E_i = \{n_i \overset{l_{i,1}}{\rightarrow} m_{i,1}, n_i \overset{l_{i,2}}{\rightarrow} m_{i,2}, \cdots, n_i \overset{l_{i,a_i}}{\rightarrow} m_{i,a_i}\}$$

である．M_i に属する各ノードはテキストノードとする．さらに，$k' = k + 1$，$q = b^2/9$ とする．

以下，3-PARTITION 問題の解が存在することと，上記の条件を満たす ShEx スキーマ S が存在することが同値であることを示す．

⇒) 3-PARTITION 問題の解 $S_1, S_2, \cdots, S_k$ が存在すると仮定する．$S = (\Sigma, \mathrm{T}, e)$ を，以下の条件を満たす ShEx スキーマとする．

- $\mathrm{T} = \{t_1, t_2, \cdots, t_k\}$
- $a_j \in S_i \Leftrightarrow \tau(n_j) = t_i \quad (1 \leq i \leq k)$
- $S_i = \{a_{i_1}, a_{i_2}, a_{i_3}\}$ とすると，

$$\begin{aligned}\delta(t_i) \rightarrow\ & l_{i_1,1} :: t_{k+1} || l_{i_1,2} :: t_{k+1} || \cdots || l_{i_1,a_{i_1}} :: t_{k+1} | \\ & l_{i_2,1} :: t_{k+1} || l_{i_2,2} :: t_{k+1} || \cdots || l_{i_2,a_{i_2}} :: t_{k+1} | \\ & l_{i_3,1} :: t_{k+1} || l_{i_3,2} :: t_{k+1} || \cdots || l_{i_3,a_{i_3}} :: t_{k+1}\end{aligned}$$

  である．ここで，t_{k+1} はテキストノードの型を表す．

このとき，任意の型の組 $t_i, t_j \in \mathrm{T} \times \mathrm{T}$ に対して，$distance(t_i, t_j)$ は定義から次のようになる．

$$\begin{aligned}distance(t_i, t_j) =\ & [(a_{i_1} + a_{j_1}) + (a_{i_1} + a_{j_2}) + (a_{i_1} + a_{j_3}) \\ & +(a_{i_2} + a_{j_1}) + (a_{i_2} + a_{j_2}) + (a_{i_2} + a_{j_3}) \\ & +(a_{i_3} + a_{j_1}) + (a_{i_3} + a_{j_2}) + (a_{i_3} + a_{j_3})]/9 \\ =\ & ((a_{i_1} + a_{i_2} + a_{i_3})(a_{j_1} + a_{j_2} + a_{j_3}))/9 \\ =\ & b^2/9\end{aligned}$$

$|\Gamma| = k + 1 = k'$ かつ上式が成り立つので，S は条件を満たす ShEx スキーマである．

⇐) 3-PARTITION 問題の解が存在しないと仮定する．このとき，ある S_i に対して，S_i に属する数の和が $b' > b$ となる．よって，型の数が $k + 1$ 以下の ShEx スキーマ S を考えると，異なるラベルの数が $b' > b$ となるものが存在する．これを t_i とすると，$distance(t_i, t_j)$ は定義から次のようになる．

$$\begin{aligned}distance(t_i, t_j) &= ((a_{i_1} + a_{i_2} + a_{i_3})(a_{j_1} + a_{j_2} + a_{j_3}))/9 \\ &= b' \times b/9 > b^2/9\end{aligned}$$

よって，S は条件を満たさない. □

第 5 章

評価実験

本章では，提案アルゴリズムに対する評価実験の結果について述べる．評価実験を行った環境は以下の通りである．

CPU： Intel(R) Xeon(R) CPU E5-2623 v3 @ 3.00GHz
メモリ： 16GB RAM, 2TB SATA HDD
OS： Linux CentOS 7 64bit
使用言語： Ruby 2.4.1

評価実験において，提案手法のアルゴリズムにより抽出された ShEx の精度と提案アルゴリズムの実行時間を計測した．

5.1 評価実験に使用したデータセット

評価実験に以下の 2 つのデータセットを使用した．

SP2Bench グラフデータとして，RDF データのベンチマークツール SP2Bench[8] を用いて生成した RDF データを使用した．SP2Bench は，DBLP に沿った RDF 文書を生成する．SP2Bench は SPARQL に対する包括的なベンチマークツールである．任意のデータサイズの RDF データを生成することができる．SP2Bench を使用して生成したファイルの例を以下の図 5.2 に示す．
SP2Bench で生成されるファイルは，名前空間とトリプルで構成される．名前空間は，DBLP 固有の文書クラス（図 5.1 においては bench:Book, bench:Article 等）を定義する．接頭辞が@prefix である行は，名前空間を宣言していて，それ以外の行はトリプルを表している．トリプルを表す各行は，＜始点ノード＞＜エッジのラベル＞＜終点ノード＞＜．＞で構成されている．ノードには，一般ノード，匿名ノード，RDF のクラスを表すノードの 4 種類ある．一般ノードは＜＞で囲まれたノードで，匿名ノードは先頭に_:を付けて表記される．RDF のクラスを表すノードは，名前空間の接頭辞を付けて表記される．具体的には，Journal を表すクラスは，先頭に bench をつけて bench:Journal と表記する．
RDF データは，トリプルの集まりである．トリプルは，主語 (subject) と述語 (predicate) と目的

語 (object) の 3 つの要素からなる．トリプル（主語，述語，目的語）は，主語と目的語の間の二項関係（述語）をモデル化し，主語ノードから述語ノードへのエッジによって有向グラフを表すことができる．以下の図 5.1 は SP2Bench で生成した RDF 形式の DBLP の例である．
破線は rdf:type でラベル付けされたエッジを表していて， sc は rdfs:subClassOf の省略形である．具体的には，ノード Proceeding1 からノード:John Due への矢印はトリプル（Proceeding1,swrc:editor, :John Due）を表す.

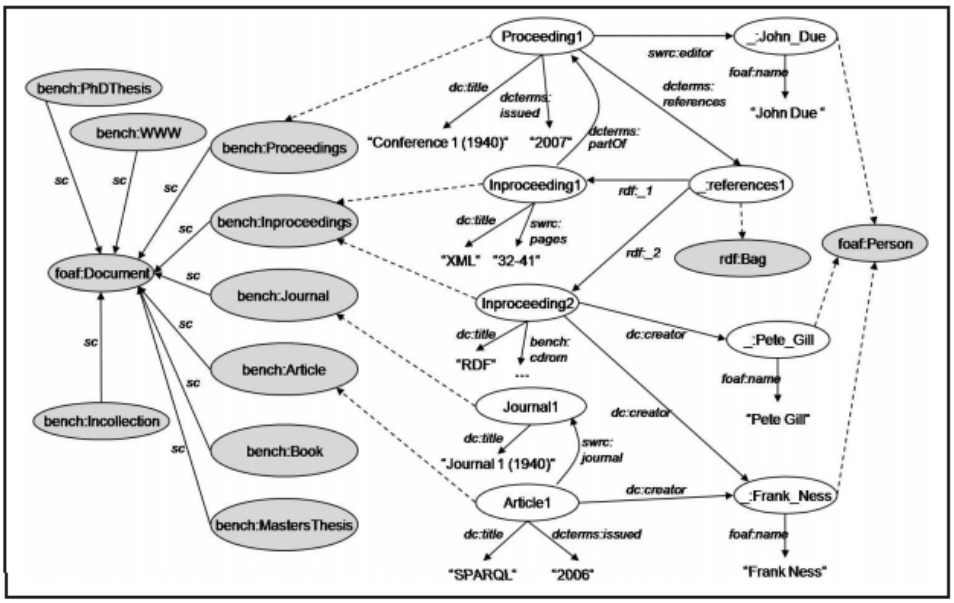


図 5.1　RDF 形式の DBLP の実例

```
@prefix dc: <http://purl.org/dc/elements/1.1/> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix swrc: <http://swrc.ontoware.org/ontology#> .
@prefix foaf: <http://xmlns.com/foaf/0.1/> .
@prefix bench: <http://localhost/vocabulary/bench/> .
@prefix xsd: <http://www.w3.org/2001/XMLSchema#> .
@prefix person: <http://localhost/persons/> .
bench:Journal rdfs:subClassOf foaf:Document.
bench:Proceedings rdfs:subClassOf foaf:Document.
bench:Inproceedings rdfs:subClassOf foaf:Document.
bench:Article rdfs:subClassOf foaf:Document.
bench:Www rdfs:subClassOf foaf:Document.
bench:MastersThesis rdfs:subClassOf foaf:Document.
bench:PhDThesis rdfs:subClassOf foaf:Document.
bench:Incollection rdfs:subClassOf foaf:Document.
bench:Book rdfs:subClassOf foaf:Document.
<http://localhost/persons/Paul_Erdoes> rdf:type foaf:Person.
<http://localhost/persons/Paul_Erdoes> foaf:name "Paul Erdoes"^^xsd:string.
<http://localhost/misc/UnknownDocument> rdf:type foaf:Document.
```

図 5.2　SP2Bench で生成されるファイルの例

LodPaddle　LodPaddle プロジェクト [9] は，地域のデータを Linked Open Data(LOD) と連動するセマンティクスデータとして公開することを目的としている．ペイドラロワール地域評議会，ロワールアトランティック，ナントメトロポール（人口 600,000 人の都市コミュニティであり，輸送，環境，エネルギー，経済開発，教育，研究を担当する 24 のコミュニティで構成されている），およびナント市は，350 を超えるオープンデータセットを持つ共有オープンデータポータルを立ち上げた．LodPaddle の目的には，全ての地域の生の構造化データセットを，リンクされたオープンデータと相互リンクされたセマンティックデータに変換および公開することが含まれる．

LodPaddle プロジェクトでは，(i) LOD として公開する価値のあるデータの選択すること，(ii) 継続的な変換とリンクされたオープンデータの公開を行う最も効率の良い方法を考えること，(iii) リンクされたオープンデータの公開するための最も良い方法を考えることの 3 つの問題を検討する必要がある．

このプロジェクトは，セマンティック Web のスキルを持つナント大学のコンピュータ科学研究所（LINA）の GDD チームによって行われている．

5.1.1　評価実験に使用するデータ

以下の 2 つの種類のデータを使用して，評価実験を行った

- SP2Bench を使用して生成したグラフデータ（表 5.1)

表 5.1　SP2Bench で生成したグラフデータ

指定したサイズ	実際のデータサイズ（KB）	ノードの数	エッジの数
50	49.956	343	461
100	101.576	687	964
150	166.544	1125	1631
200	203.462	1370	1997

- LodPaddle のデータ（表 5.2）
  ナント・メトロポールにある文化施設の場所，住所，URL，電話番号等の情報を Σ-ラベル付き有向グラフとして表現したデータである．

表 5.2　ナント・メトロポールの文化施設

実際のデータサイズ（KB）	ノードの数	エッジの数
127.647	2178	3252

5.2　抽出したスキーマの精度

以下では，抽出されたスキーマの精度について述べる．SP2Bench は，各ノードが RDF 型（記事，雑誌等）を持つように RDF グラフを生成する．また表 5.2 のナント・メトロポールのデータは，各のノードに複数の RDF 型が割り当てわれている．ここで，複数の RDF 型を 1 つの型にまとめることで，各ノードに 1 つの型が割り当てられるようにした．dbpedia：Hybrid library，sc：Library， dbpedia：Toy library の 3 つを sc：Library にまとめ，sc:LandmarksOrHistoricalBuildings，sc：Museum を sc：LandmarksOrHistoricalBuildings にまとめた．

このような RDF 型を「正解」とみなし，提案手法により抽出した各ノードの ShEx 型とその RDF 型を比較して精度を計算する．各ノードに RDF 型（正解）を割り当てる関数を r とする．各ノードに ShEx 型を割り当てる関数は τ である．

ここで，スキーマの精度を測る尺度として，Rand 尺度を導入する．以下を定義する．

- τ と r のどちらにおいても同じ型が割り当てられるノード対の数 a_s
  $a_s = |\{(n_i, n_j) \in N(G) \times N(G) \mid \tau(n_i) = \tau(n_j), r(n_i) = r(n_j), n_i \neq n_j\}|$

- τ と r のどちらにおいても違う型が割り当てられるノード対の数 a_d
  $a_d = |\{(n_i, n_j) \in N(G) \times N(G) \mid \tau(n_i) \neq \tau(n_j), r(n_i) \neq r(n_j), n_i \neq n_j\}|$

以上を基に，Rand 尺度は以下のように定義される．

$$Rand\_index := \frac{a_s + a_d}{M},$$

ただし $M = \frac{|N(G)|(|N(G)|-1)}{2}$．分母の M は，ノード対の総数を表す．

表 5.3 と表 5.4 に，SP2Bench で生成したデータ（表 5.1）とナント・メトロポールの文化施設（表 5.2）に対し評価実験を行った際の Rand 尺度を示す．Rand 尺度が最大となる時の型の数とその時の Rand 尺度の値を記載した．

表 5.3　表 5.1 のデータの Rand 尺度

指定したデータサイズ	型の数	Rand 尺度（効率化なし）
50	21	0.9659011474263891
100	35	0.9641318785780064
150	18	0.9698030842230131
200	34	0.9605647470315057

表 5.4　表 5.2 のデータの Rand 尺度

データサイズ	型の数	Rand 尺度（効率化なし）
127.647	10	0.9740099453633508

表 5.5 と表 5.6 に，3.4 節で述べた効率化を行った際の Rand 尺度を示す．指定した型の数の 2 倍の数のクラスタ数を指定してクラスタリングを行い，初期段階として型を割り当てた．効率化を行っても，抽出したスキーマの精度はあまり変わらないことが分かった．

表 5.5　表 5.1 のデータの Rand 尺度

指定したデータサイズ	型の数	Rand 尺度（効率化あり）
50	8	0.9450667485039128
100	8	0.978021651580158
150	8	0.975164887307236
200	8	0.965732352988222

表 5.6　表 5.2 のデータの Rand 尺度

データサイズ	型の数	Rand 尺度（効率化あり）
127.647	10	0.9251119791897342

5.3　実行時間

用意したデータセット（表 5.1）に対するスキーマを抽出を行うのにかかる時間を Mesure-Command コマンドレットを用いて計測した．計測した結果を図 5.3 に示す．

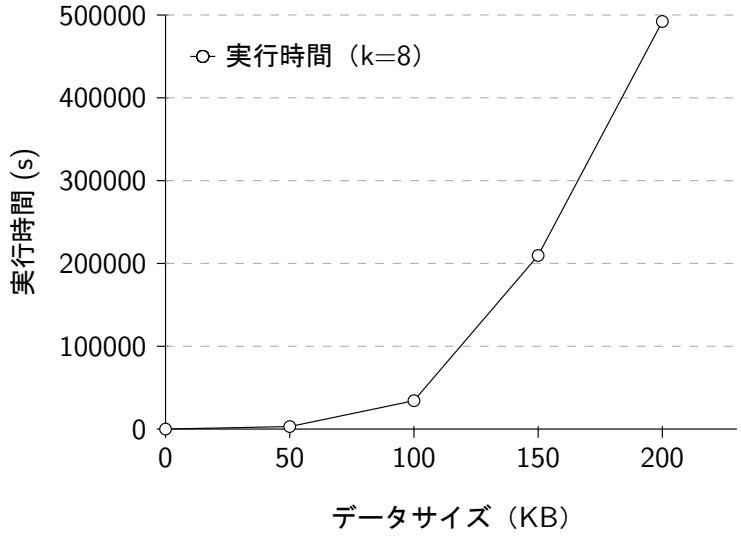


図 5.3　提案アルゴリズムの実行時間 (s)

次に，3.4 節で述べた効率化を用いた場合の結果を示す．効率化を行った場合の実行時間を測った結果を図 5.4 に示す．

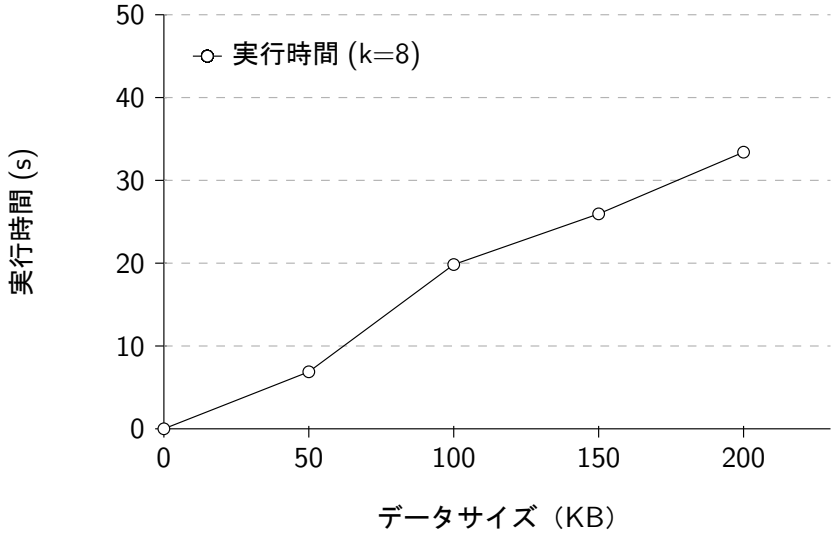


図 5.4　提案アルゴリズムの実行時間 (s)

第 6 章

むすび

本研究では，Σ-ラベル付き有向グラフに対する ShEx の抽出方法を提案した．ただし，厳密な最適解を求めるのは計算困難であるため，貪欲法に基づいてスキーマ抽出を行った．

評価実験の結果，概ね適切にスキーマを抽出を行えることが分かった．また，提案アルゴリズムの実行時間はデータサイズに対して線形であることが分かった．

今後の課題として，アルゴリズムの実行時間をさらに短縮する方法も考案すること，主記憶に収まらない大規模なグラフデータに対しても対応できるようにすることが挙げられる．グラフデータは年々増加傾向にありかつ大規模化している．このため，大規模なグラフデータを効率的に処理することが求められるようになっている．一方，本研究のスキーマ抽出アルゴリズムは，主記憶に収まらないような大規模グラフデータには対応していない．今後は，主記憶に収まらない大規模なグラフデータからでも効率よくスキーマを抽出可能なアルゴリズムを考案する予定である．更に，ノードの重要性（PageRank で取得したものなど）をアルゴリズムに組み込んで，ShEx スキーマを改善することも考えられる．

謝辞

本研究を進めるにあたって，多くの御指導と御助言を賜りました，指導教員の鈴木伸崇先生に深く感謝致します．また，御指導と御助言をくださった副指導教員の永森光晴先生に心から感謝致します．更に，共に研究に励んだ藤永さん，鈴木さん，そして同じ研究室に所属する皆様にも，様々な助言をして頂きました．改めて感謝致します．最後に，本研究に対してご助言を頂きました全ての皆様に感謝の意を表し，謝辞と致します．

参考文献

[1] Balmin, A., Hristidis, V., and Papakonstantinou, Y. ObjectRank: authoritybased keyword search in databases. In Proceedings of the Thirtieth International Conference on Very Large Data Bases (2004), pp. 564–575.

[2] Saket Navlakva, Rajeev Rastogi, Nisheeth Shrivastava. ” Graph Summarization with Bounded Error” , SIGMOD 2008. (accessed 2017-04-28).

[3] R. Goldman and J. Widom. ” DataGuides: Enabling query formulation and optimization in semistructured databases” , VLDB 1997, p. 436-445.

[4] R. Goldman, J. Widom. ”Approximate DataGuides,” Workshop on Query Processing for Semistructured Data and Non-standard Data Formats, 7p., 1999.

[5] Thomas Baker and Eric Prud'hommeaux eds. Shape Expressions (ShEx) Primer, http://shexspec.github.io/primer/.

[6] S. Ginsburg and Spanier E. H. Semigroups, presburger formulas, and languages. Pacific Journal of Mathematics, 16(2):285–296, December 1966.

[7] R. J. Parikh. On context-free languages. Journal of the ACM, 13(4):570–581, 1966.

[8] Michael Schmidt, Thomas Hornung, Georg Lausen, and Christoph Pinkel. SP2Bench: a SPARQL Performance Benchmark. In Data Engineering, 2009. ICDE' 09. IEEE 25th International Conference on, pp. 222–233. IEEE, 2009.

[9] ”DokuWiki”, LodPaddle, http://lodpaddle.univ-nantes.fr/doku.php, (accessed 2019-11-20).