# Shape Expression Schemaのスキーマ進化に対するProperty Path式修正アルゴリズム

筑波大学
図書館情報メディア研究科

2021年3月

赤澤　豪樹

# 目 次

# 第1章　序章

RDF/グラフデータ（以下，グラフデータ）が広く普及し，このようなデータに対するスキーマ言語（RDF Schema，SHACL，Shape Expression Schema 等）が提案されている．グラフデータの活用が進むにつれて，スキーマの利用もより拡大していくと考えられる．ここでのスキーマとは，グラフデータの構造に関する規則を表したものである．スキーマを定義することで，問合せ式の記述に役立つことやその効率性を向上させる等の利点がある．加えて，グラフデータは急速に増加しており，またそのサイズも増大の一途を辿っているため，様々な処理においてより効率性が求められる．そのため，上記の利点をもつスキーマの重要性も増していると考えられる．スキーマは，関係データベースや XML の分野では広く活用されており，グラフデータにおいても一層の活用が期待されている．

RDF データはグラフデータの一種とみなすことができる．これまで，RDF データではスキーマが定義されることが比較的多かったが，スキーマは通常 RDF Schema[1] というスキーマ言語を用いて定義されていた．しかし RDF Schema はオントロジ記述言語としての側面が強く，スキーマ言語としての形式的なセマンティクスが定義されていない．このため，データ構造を厳密に定義していると言い難く，妥当性検証を行うための記述言語として必ずしも適さない．そこで提案されたのが Shape Expression Schema (以下 ShEx)[2][3] である．ShEx は，型と呼ばれる小規模のスキーマの集合であり，各型は Regular Bag Expression という規則に基づいて定義されている．ShEx がグラフデータに対して妥当であるとき，グラフの各ノードには，その定義を満たしている妥当な型を割り当てることができる．ShEx では，各ノードに複数の型を割り当てることができる点も特徴の一つである．

本研究では，特にスキーマの更新に着目する．時間の経過と共にデータの利用状況等も変化するため，それに応じてスキーマも更新されるのが一般的である．その場合，スキーマ下にあるデータの構造も変化し，更新前の問合せ式が使用できなくなることがある．このような場合，問合せ式の修正が必要になるが，スキーマの構造を正確に把握し，適切な修正を行うのは容易ではない．そこで本研究では，スキーマが更新された際に，その更新に応じて問合せ式を自動修正する手法について考える．問合せ式として，本研究では Property Path[4] を対象とする．Property Path は，RDF に対する問合せ言語である SPARQL 1.1 において定義されている．Property Path は，経路問合せとして一般的な Regular Path Query の機能に加え，いくつかの拡張した機能が備わった問合せ言語である．

本論文では，ShEx スキーマが更新された際，その更新に応じて Property Path 式を修正する手法を提案する．修正においては，可能な限り元の Property Path 式と同じ解を返す式が得られるようにする．まず，スキーマ更新に必要と考えられるそれぞれの操作において，Property Path 式の修正が必要となるかどうかを考察する．次に，ShEx スキーマの更新に応じて Property Path 式を修正するアルゴリズムを提案する．このアルゴリズムは，まず ShEx スキーマをグラフ化し，そのスキーマグラフ上で更新操作を適用する．更新操作の適用後，経路に欠落が生じた場合，Property Path 式の経路を補完した上で，Property Path 式を復元する．

評価実験においては，修正前の Property Path 式の解を正解として扱い，提案アルゴリズムによって修正された Property Path 式の解の一致度を求めた．具体的には，それぞれの

再現率，適合率，F 値を算出するという方法であり，計 10 回の実験結果を示している．スキーマ上の定義で「*」や「?」が付随する「オプション」のエッジ等要因から，すべての実験で完全一致が得られた訳ではないが，結果的に，F 値の平均が 0.87 となり，概ね良好であると考えられる．

関連研究として，XML に関しては，DTD が更新された際に XPath 式を修正するアルゴリズム [5] やスキーマ更新に応じて XPath 式の充足可能性判定を行うシステム [6]，XSLT スタイルシートの修正を行う手法 [7] など，数多くの研究がなされている．しかし，グラフデータにおいて，スキーマ更新に応じて問合せ式を修正する手法は著者の知る限り提案されていない．XML は木構造でデータを記述するのに対し，RDF ではグラフ構造で記述する．木構造では任意の 2 頂点間の経路はただ一つとなるため，もしその経路が遮断された場合は元々の終点には到達不可能となる．よって [5] の研究等，XML の問合せ修正において経路を補完するという過程はない．それに対し，グラフ構造（RDF）における問合せでは任意の 2 頂点間の経路が複数存在する可能性があるため，特定の経路が遮断されたとしても，経路を補完すれば同一の終点に到達できる場合がある．さらに本研究においては経路補完の際に，エッジを逆向きに辿るという Property Path 特有の機能にも対応させている．また，Property Path にはテキストノード以外にも適用可能な選言や要素の出現回数を指定するオペレータ等，XPath にはない機能が存在する．それらの機能が一因となり，本研究において問合せを変換したオートマトンには閉路が含まれる可能性がある．そのため，XPath に関する [5] の研究では問合せを木構造の有限オートマトンに変換しているが，本研究では木構造でない有限オートマトンを採用している．これらの相違から，XML の問合せに関する関連研究を本研究に応用することは困難であると言える．

本論文の構成は以下の通りである．2 章では，グラフ，ShEx，および Property Path に関する定義を行う．3 章では，スキーマにおける更新操作を定義した後，Property Path 式修正アルゴリズムについて説明する．4 章では評価実験について述べる．5 章ではまとめと今後の課題について述べる．

# 第2章　諸定義

本章では，グラフ，ShEx，および，Property Path に関する定義を行う．

## 2.1　グラフ

本研究では，ラベル付き有向グラフ（以下，グラフ）を対象とする．グラフは $G = (V, E)$ と表され，ここで $V$ はノード集合，$E$ はエッジ集合である．例えば，図 2.1（左図）のグラフ $G$ の場合，$G = (V, E)$ と表され，ここで

$$\begin{aligned} V &= \{v_0, v_1, v_2, v_3, v_4\} \\ E &= \{(v_0, a, v_1), (v_0, b, v_3), (v_0, c, v_2), (v_1, c, v_4), (v_2, c, v_3), (v_4, a, v_3)\} \end{aligned}$$

である．

## 2.2　Shape Expression Schema

ShEx はグラフデータに対するスキーマ言語である．XML データとは異なり，グラフデータにおいては兄弟ノード間の順序関係を考慮しないことが多い．そのため，内容モデルの記述には Regular Expression の代わりに Regular Bag Expression（以下 RBE）を用いる．RBE の大きな特徴は連結「||」において順序が無視されることである．RBE は以下のように定義される．

- ε および任意の $a \in \Sigma$ は RBE である
- $E_1, E_2, \cdots, E_k$ が RBE であるならば，$E_1|E_2|\cdots|E_k$ は RBE である．ここで，|は選言を表す．
- $E_1, E_2, \cdots, E_k$ が RBE であるならば，$E_1||E_2||\cdots||E_k$ は RBE である．ここで，||は順序を無視した連結を表す．
- $E$ が RBE であるならば，$E^{[n,m]}$ は RBE である．ここで，$[n, m]$ は $n$ 回以上 $m$ 回以下の繰り返しを表す．

ShEx スキーマは $S = (\Sigma, \Gamma, \delta)$ と表現される．ここで，$\Gamma$ はラベルの有限集合，$\delta$ は型の集合，$\delta$ は $\Gamma$ から $\Sigma \times \Gamma$ 上の RBE への関数であり，型の内容モデルを定義する．ShEx スキーマではそれぞれのノードに型を付与するが，全てのノードが 1 つの型を持つ場合を single-type，2 つ以上の型を取り得る場合を multi-type と呼ぶ [8]．本研究では single-type の場合を考える．

スキーマの更新を考えるにあたって，本研究では ShEx スキーマをグラフとして表現する．例えば，ShEx スキーマ $S = (\Sigma, \Gamma, \delta)$ を考える．ここで，$\Sigma = \{a, b, c\}$, $\Gamma = \{t_0, t_1, t_2, t_3, t_4\}$

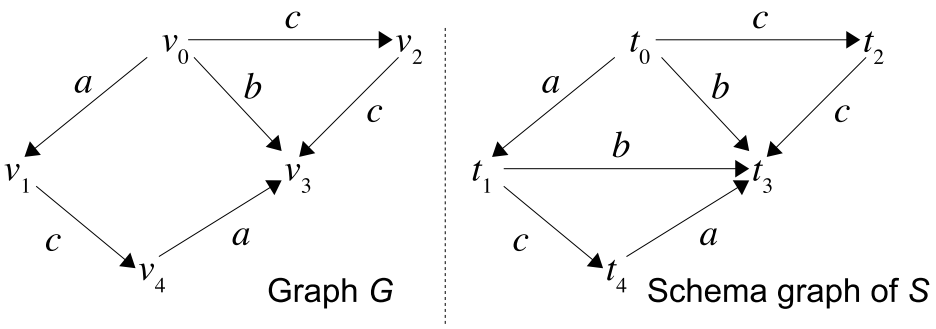


図 2.1: $S$ に対して妥当なグラフ $G$ と $S$ のスキーマグラフ

かつ δ は次のように定義される．

$$\begin{aligned}
\delta(t_0) &= a :: t_1 \parallel b :: t_3 \parallel (c :: t_2)^*, \\
\delta(t_1) &= b :: t_3 | c :: t_4, \\
\delta(t_2) &= c :: t_3, \\
\delta(t_3) &= \epsilon, \\
\delta(t_4) &= a :: t_3.
\end{aligned}$$

この時，S のスキーマグラフは図 2.1（右図）で表される．すなわち，型 $t$ に対して，$\delta(t)$ の中に $a :: t'$ が含まれるときかつそのときのみ，$t$ から $t'$ に向けてラベル $a$ のエッジを設ける．

## 2.3 Property Path

Property Path は RDF データに対する問合せ式の一種であり，探索経路を指定する．Property Path は Regular Path Query と類似しているが，幾つかの拡張された表現を持つ．主な拡張としては以下が挙げられる．

- エッジを逆向きに辿る探索が可能である
  - $a^{-1}$ と指定すると，$a$ ラベルのエッジを逆向きに進む
- 特定のラベルを辿らない（否定）探索が可能である
  - $!\{a, b\}$と指定 r すると，a ラベルと b ラベル以外のエッジを進む

より形式的には，$\Sigma$ 上の Property Path 式は以下のように定義される．

- $\epsilon$ および任意の $a \in \Sigma$ は Property Path 式である．
- $*$ は Property Path 式である．
- $\{a_1, a_2, \cdots, a_k\}$ をラベル集合としたとき，$!\{a_1, a_2, \cdots, a_k\}$ は Property Path 式である．これは，$\{a_1, a_2, \cdots, a_k\}$ 以外のラベルにマッチする．
- 任意の $a \in \Sigma$ に対して，$a^{-1}$ は Property Path 式である．これは，ラベル $a$ のエッジを逆向きに辿る．

- $q_1, q_2, \cdots, q_k$ を Property Path 式とする時，$q_1.q_2.\cdots.q_k$ および $q_1|q_2|\cdots|q_k$ は Propery Path 式である．
- $q$ を Property Path 式とするとき，$q^*$ は Property Path 式である．

本研究では，探索を開始する始点のノードが 1 つ与えられるものと仮定する．そのため，始点のノードから Property Path を経て到達したノードの集合が解となる．例えば，図 2.1（左図）のグラフ $G$ の例を考える．Property Path 式と始点に対する解（終点ノードの集合）の例を示す．

- Property Path 式 $a.c$ および始点ノード $v_0$ → 解：$\{v_4\}$
- Property Path 式 $c^{-1}.c^{-1}$ および始点ノード $v_3$ → 解：$\{v_0\}$
- Property Path 式 $b.(a^{-1}|c^{-1})$ および始点ノード $v_0$ → 解：$\{v_2, v_4\}$

# 第3章　提案手法

本章では，まず ShEx の更新操作について述べる．次に，Property Path 式に対する修正手法に関して述べる．

## 3.1　型の更新操作

本節では，スキーマ更新を表すために，ShEx の型に対する更新操作を示す（更新操作は [9] に基づく）．更新操作の定義のため，型を木構造で表現する．ここでは各ノードの位置を識別するために，デューイの順序に基づく ID を各ノードに与える．例えば，以下のような型の定義があるとする．

$$\delta(t_0) = a :: t_1^* \parallel (b :: t_2 | c :: t_3).$$

$t_0$ の木構造表現を図 3.1 に示す．各ノードに関連付けられた ID は，ノードの位置を意味している．

以下，ShEx スキーマの型に対して次の 8 つの更新操作を定義する．

- 「ラベル::型」に関する処理：
  - $add\_lt(t, i, l' :: t')$: $t$ の内容モデルに「ラベル::型」を追加する処理である．これは，スキーマグラフにおいてはエッジの追加に相当する．ここで，$t$ は更新対象の型，$i$ は $\delta(t)$ における追加位置，$l' :: t'$ は追加する「ラベル::型」である．
  - $del\_lt(t, i)$: $t$ の内容モデルから「ラベル::型」を削除する処理である．これは，スキーマグラフにおいてはエッジの削除に相当する．ここで，$t$ は更新する型，$i$ は $\delta(t)$ における削除位置である．
  - $change\_lt(t, i, l' :: t')$: 「ラベル::型」を変更する処理である．ここで，$t$ は更新する型，$i$ は $\delta(t)$ における変更位置，$l' :: t'$ は変更後の「ラベル::型」である．

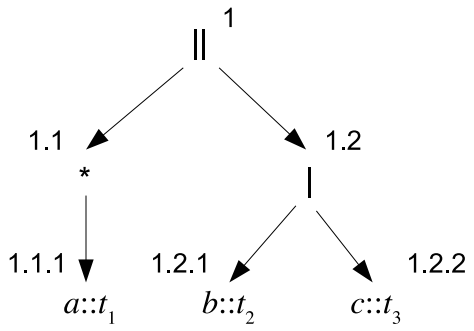


図 3.1: $t_0$ の木構造表現

- オペレータに関する処理：
  ここで言うオペレータとは，選言"|"，連接"||"，繰り返しの回数 $[n, m]$ である．
  - $add\_opr(t, i, op)$: オペレータ $op$ を追加する．ここで，$t$ は更新する型，$i$ は $\delta(t)$ における追加位置，$op$ は追加するオペレータである．
  - $del\_opr(t, i)$: 指定された位置のオペレータを削除する．ここで，$t$ は更新する型，$i$ は $\delta(t)$ における削除位置である．
  - $change\_opr(t, i, op)$:指定された位置のオペレータを $op$ に変更する． ここで，$t$ は更新する型，$i$ は $\delta(t)$ における変更位置，$op$ は変更後のオペレータである．
- 型の追加・削除:
  - $add\_type(t)$: 新たな型を追加する処理である．ここで，$t$ は追加する型であり，$\delta(t) = \varepsilon$ とする．
  - $del\_type(t)$: 既存の型を削除する処理である．ここで，$t$ は削除する型である．

更新操作は $s = op_1 op_2 \cdots op_n$ のような系列として表現される．例えば， 図 3.1 における $t_0$ に対して，以下の更新操作が行われたとする．

$$s = change\_lt(t_0, 1.2, \|)\, add\_lt(t_0, 1.1, d :: t_3)$$

この場合，$t_0$ の定義は

$$\delta(t_0) = d :: t_3 \parallel a :: t_1^* \parallel (b :: t_2 \parallel c :: t_3)$$

と更新される．

上記 8 種類の更新操作のうち，追加処理全般とオペレータへの処理全般は，探索経路に影響はないため，更新前に使用していた Property Path 式が妥当でなくなることはない．それに対し，「ラベル::型」の削除，変更処理，型の削除処理は，更新前の探索に用いていた経路がなくなる，または，ラベル名が変わるといった状況が生じ得るため，更新前に使用していた Property Path 式が妥当ではなくなる可能性があり，その場合は修正が必要になる．

そのため，以下に示すアルゴリズムはスキーマ $S$ に $del\_lt()$, $change\_lt()$, または $del\_type()$ が適用された場合を想定している．

また，ShEx スキーマ $S$ が更新された場合，スキーマの更新に従って $S$ の下のデータも更新される．これは既存のアルゴリズムを用いて行なうものとする [9].

## 3.2 アルゴリズム

本研究のアルゴリズムは，3 つの部分から構成される．そのうち，Algorithm 1 が主要部分となる．ShEx スキーマ $S$ に対する更新操作 $s = op_1.op_2.\cdots.op_n$ および探索開始の型 $t_s$ が与えられ，Property Path 式 $q$ を $s$ に応じて変換する．$G_S$ を S のスキーマグラフとし，$G_S(q, t_s)$ を $t_s$ から始まる $q$ の経路領域とする（1 行目と 2 行目）．まず，$H_S$ に $G_S(q, t_s)$，$G'_S$ に $G_S$ をそれぞれコピーする（3 行目）．次に，$s$ の各操作 $op_i$ に従って $H_S$ を変更し（4 行目から 27 行目），$H_S$ をクエリ $q'$ に変換する（28 行目）．4 行目から 27 行目のループは次のように進行する．$op_i$ が $H_S$ に影響を与えない場合，アルゴリズムの処理は何も行われない（5 行目から 7 行目），それ以外の場合，$H_S$（および $G'_S$）は，8 行目から 26 行目において $op_i$ に従って変更される．

- 8 行目から 14 行目は,「ラベル::型」の変更処理（$change\_lt(t, i, l' :: t')$）を扱う．この操作により，位置 $i$ の $\delta(t)$ の label :: type pair$l_i :: t_i$ が $l' :: t'$ に変更される． これによると，$H_S$ と $G_S$ においてエッジ $(t, l_i, t_i)$ を $(t, l', t')$ に置き換えることとなる．もし $t_i = t'$ の場合，ラベル名の変更のみで処理は完了する．しかし $t_i = t'$ でない場合は，$t_i$ が $t'$ に変更されるため，$t_s$ から $t_i$ を介した入力ノードへの経路が切断される可能性がある．これを修復するために，FindPaths（Algorithm 2）によって，$G'_S$ における $t'$ から $t_i$ までの単純経路 $P$ の集合を見つけ，各経路 $p \in P$ を $H_S$ に追加して $t_i$ と $t'$ を接続する．

  FindPaths は，与えられたタイプ $t, t'$ に対して，$G'_S$ 上で $t$ から $t'$ までの単純経路 $p$ の集合 $P$ を見つけるためのメソッドである．その探索においては，エッジを逆向きに辿ることも許可される．ただし，すべての単純経路 $p$ の長さが指定された閾値を超えると，FindPaths は，より短い単純経路を探索し，見つかった場合はその経路を出力する．詳細は後述する．

- 15 行目から 19 行目は,「ラベル::型」の変更処理（$del\_lt(t, i)$）を扱う．この操作により，$del\_lt(t, i)$ の位置 $i$ にある label :: type ペア $l_i :: t_i$ が削除される．これに従って，$H_S$ と $G'_S$ からエッジ $(t, l_i, t_i)$ を削除する．その際，$t$ と $t_i$ の経路が切断された場合は，FindPaths によって $G'_S$ 上で $t$ から $t_i$ への経路を見つけ，$H_S$ に経路を追加する．

- 20 行目から 26 行目は，型の削除（$del\_type(t)$）を扱う．この操作により，型 $t$ が $S$ から削除される．したがって，$t$ および $t$ に付随するすべてのエッジは，$H_S$ および $G'_S$ から削除されることとなる．これを修復するには，$t$ に出力するノードの集合 $T_s$ と $t$ から出力されるノードの集合 $T_g$ を特定し，$T_s$ から $T_g$ への経路を探索する．そして発見した新たな経路を $H_S$ に追加する．

28 行目で，ConstructPropertyPath（Algorithm 3）は $H_S$ を新しい Property Path 式 $q'$ に変換する．これは，$H_S$ を開始状態 $t_s$ と受け入れ状態の集合 $Ans(G_S(q, t_s))$ を持つ NFA$M$ と見なし（2 行目），$M$ と同等の DFA$M'$ を構築する（3 行目）そして $M'$ を Property Path 式 $q'$ に変換する（4 行目）．$M'$ から $q'$ への変換は，DFA の状態除去方法の拡張に基づいて行われる．

以下，FindPaths（Algorithm 2）の詳細を述べる．スキーマグラフ $G_S$ のコピーである $G'_S$，その間の経路修復が必要なノード 2 つ（経路開始地点である型 $t_s$ と終着地点 $t_g$）が与えられる．

- まず 1〜2 行目で，修正 Property Path 式候補を記録するための変数 $path$，最終的な修正 Property Path 式候補を保存するための集合 $P$ を初期化する．3〜4 行目で全てのノードに集合 $Prepaths$ を付与する．これはそのノードまでたどり着く Property Path 式の経路として既に使用されたものを保存するための配列である．5 行目で，遷移中のノードを保存しておく変数 $current$ に初期値として $t_s$ を代入する．6 行目では遷移する可能性のある隣接ノードを保存しておく変数 $next$ を示している．

- そして $t_s$ から $t_g$ に至る単純経路上のノードで未訪問のものがなくなるまで（8 行目），9〜37 行目の処理が行われる．10〜18 行目では $next$ のノードにおける $Prepaths$ が空である，または，$current$ が $next$ のノードにおける $Prepaths$ に含まれない場合，かつ， $next$ のノードがその時点の $path$ における式の経路に含まれない場合の処理を示している．

  – 11 行目で $path$ にその隣接ノード間のエッジラベルを追加し，12 行目では $current$

と $next$ 間のエッジ（逆向きを含む）を $next$ のノードにおける $Prepaths$ に追加する．

- 13 行目で $current$ に $next$ のノードを代入し，14 行目では $next$ には，また次に遷移可能性のある隣接ノードを代入する．この時点で $current$ が $t_g$ のノードである場合（15 行目），16 行目で $P$ にその時点の $path$ を保存する．
- 17 行目で $path$ の末尾のラベルを削除する．
- 19〜22 行目では，$next$ のノードがその時点の $path$ における式の経路に含まれる場合や遷移先ノードが葉ノードである場合等の処理を示している． $path$ の末尾のラベルを削除して一つ前のノードに戻る，という手順である．
- $next$ のノードにおける $Prepaths$ が空である，または，$current$ が $next$ のノードにおける $Prepaths$ に含まれない場合，かつ， $next$ のノードがその時点の $path$ における式の経路に含まれてしまっている場合（24〜27 行目），19〜22 行目と同様に 1 つ前のノードに戻る．

**Algorithm 1** Query Transformation

**Input:** ShEx schema $S = (\Sigma, \Gamma, \delta)$, update script $s = op_1 op_2 \cdots op_n$ to $S$, Property Path $q$, type $t_s \in \Gamma$

**Output:** query $q'$

construct the schema graph $G_S$ of $S$
construct the traverse area $G_S(q, t_s)$ of $q$ from $t_s$ on $G_S$
$H_S \leftarrow G_S(q, t_s)$; $G'_S \leftarrow G_S$
**for** $i = 1, 2, \cdots, n$ **do**
  **if** $op_i$ does not affect $H_S$ **then**
    **continue**
  **end if**
  **if** $op_i = change\_lt(t, i, l' :: t')$ **then**
    let $l_i :: t_i$ be the label::type pair at position $i$ of $\delta(t)$
    replace $(t, l_i, t_i)$ with $(t, l', t')$ in $H_S$ and $G'_S$
    **if** $t_i \neq t'$ **then**
      $P \leftarrow FindPaths(G'_S, t', t_i)$
      add all $p \in P$ to $H_S$
    **end if**
  **else if** $op_i = del\_lt(t, i)$ **then**
    let $l_i :: t_i$ be the label::type pair at position $i$ of $\delta(t)$
    delete $(t, l_i, t_i)$ from $H_S$ and $G'_S$
    $P \leftarrow FindPaths(G'_S, t, t_i)$
    add all $p \in P$ to $H_S$
  **else if** $op_i = del\_type(t)$ **then**
    $T_s \leftarrow \{t_1 \mid (t_1,l,t)$ is an edge from $t_1$ to $t$ in $G'_S\}$
    $T_g \leftarrow \{t_2 \mid (t,l,t_2)$ is an edge from $t$ to $t_2$ in $G'_S\}$
    delete $t$ and every edge adjacent to $t$ from $H_S$ and $G'_S$
    $P \leftarrow \{p \mid p \in FindPaths(G'_S, t_1, t_2), t_1 \in T_s, t_2 \in T_g\}$
    add all $p \in P$ to $H_S$
  **end if**
**end for**
$q' \leftarrow$ ConstructPropertyPath($H_S$,$t_s$, $ans(G_S(q,t_s))$)
**return** $q'$

**Algorithm 2** FindpPaths

**Input:** $G'_S$, start type $t_s$, goal type $t_g$

**Output:** set of paths $P$

$path \leftarrow$ ""
$P \leftarrow \emptyset$
**for** $u \in V'_s$ **do**
  $Prepaths(u) \leftarrow \emptyset$
  $current \leftarrow t_s$
  $next \leftarrow$ an unvisited neighbor of $current$
**end for**
**while** there are unvisited types on the simple route from $t_s$ to $t_g$ **do**
  **if** $Prepaths(next) = \emptyset$ or $current$ (part of the route in $Prepaths(current)$) is not included in any route in $Prepaths(next)$ **then**
    **if** $next$ is not included in $path$ then **then**
      append the label of the edge between $current$ and $next$ to $path$
      add the edge (including inverse direction) between $current$ and $next$ to $Prepaths(next)$
      $current \leftarrow next$
      $next \leftarrow$ an unvisited neighbor of $current$
      **if** $current = t_g$ **then**
        add $path$ to $P$
        delete the last label of $path$
      **end if**
    **else**
      delete the last label of $path$
      $current \leftarrow$ the previous type of $current$
      $next \leftarrow$ an unvisited neighbor of $current$
    **end if**
  **else**
    delete the last label of $path$
    $current \leftarrow$ the previous type of $current$
    $next \leftarrow$ an unvisited neighbor of $current$
  **end if**
**end while**
return $P$

**Algorithm 3** ConstructPropertyPath

**Input:** traversal area $H_S$, start type $t_s$, set of types $Ans$

**Output:** Property Path $q'$

let $V$ and $E$ be the sets of nodes and edges of $H_S$, respectively
construct an NFA $M = (Q, \Sigma, \delta, t_s, Ans)$, where $Q = V$ and $\delta$ is a transition function s.t. $\delta(t, a) = t'$ iff $(t, a, t') \in E$
construct a DFA $M'$ equivalent to $M$
construct a Property Path $q'$ from $M'$
**return** $q'$

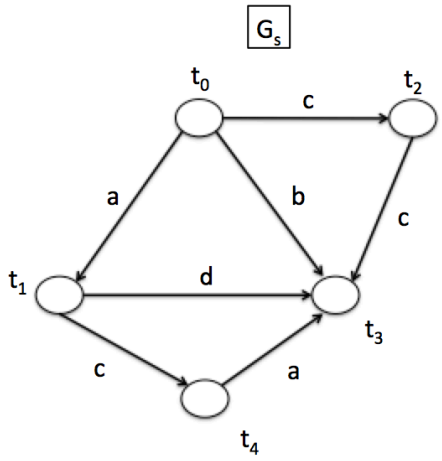


図 3.2: $S$ のスキーマグラフ

### 3.2.1 アルゴリズム適用例

アルゴリズムの適用例を示す．以下の ShEx スキーマ $S = (\Sigma, \Gamma, \delta)$ を考える．ここで，$\Sigma = \{a, b, c\}$, $\Gamma = \{t_0, t_1, t_2, t_3, t_4\}$ かつ $\delta$ は次のように定義される．

$$
\begin{aligned}
\delta(t_0) &= a :: t_1 \parallel b :: t_3 \parallel (c :: t_2)^*, \\
\delta(t_1) &= b :: t_3 | c :: t_4, \\
\delta(t_2) &= c :: t_3, \\
\delta(t_3) &= \epsilon, \\
\delta(t_4) &= a :: t_3.
\end{aligned}
$$

このとき，$S$ のスキーマグラフ $G_S$ は図 3.2 で表される．

また，Property Path 式 $q$ を $a.c$，$t_s$ を $t_0$ とすると，$t_s$ から始まる $q$ の経路領域 $G_S(q, t_s)$ は図 3.3 となる（ここで $G_S$，$G_S(q, t_s)$ のそれぞれのコピー $G'_S$，$H_S$ をとる）．$G_S$ 上で $t_0$ の $a :: t_1$ の削除（$s = del\_(t_0, 1)$）という更新操作が行われたとすると，該当部分があるため $G_S(q, t_s)$ のコピーである $H_S$ 上でも同様の更新操作が行われる．$p$ における「.」区切りの要素（ラベル）のどれに影響があるか判別する．今回の更新操作の場合，第一要素「$a$」部分に影響があり（修正の必要あり），第二要素「$c$」部分には影響がない（修正の必要なし）ことが分かる．また，更新操作が「ラベル::型」の削除であるため，Algorithm 1 において 15〜19 行目の処理となる．$H_S$ における更新操作対象部分の始点ノード $t$ は $t_0$，終点ノード $t_i$ は $t_1$ となり FindPaths を呼び出す．(Algorithm 1 の 18 行目)

ここで FindPaths による経路修復が行われる（図 3.4）．

- $current$ には $t_0$ が代入されているが，その後 $t_3$ が代入され，$t_0$ から $t_3$ に遷移する形となる．
  - $Prepaths(t_3) = [[(t_0, b, t_1]]$
  - $path = b$

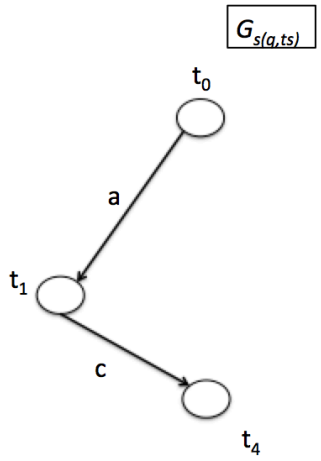


図 3.3: $t_s$ から始まる $q$ の経路領域 $G_S(q, t_s)$

  - $current = t_3$
  - $next = t_1$

- 次に，$current$ は $t_3$ から $t_1$ へと遷移する．
  - $Prepaths(t_1) = [[(t_0, b, t_3), (t_3, d^{-1}, t_1)]]$
  - $path = b.d^{-1}$
  - $current = t_1$
  - $next = t_3$
- このとき，$current$ が $t_1$（終点ノード）となる．
  - $P$ に $path$ を追加 $\rightarrow P = [b.d^{-1}]$
  - $path = b$
- ここで，その後どのように遷移しても，FindPaths の 8,9 行目の条件から，$t_0$ まで戻ることとなる．

続けての実行の様子を図 3.5 に示す．

- $t_0$ から $t_2$，$t_3$ へと進む．
  - $Prepaths(t_3) = [[(t_0, b, t_1)]]$ であり，$[(t_0, c, t_2)]$ ($Prepaths(t_2)$ 内の経路) は含まれず，$path = c$) の経路上に $t_3$ もないため，問題なく進める．
  - $Prepaths(t_3) = [[(t_0, b, t_3)], [(t_0, c, t_2), (t_2, c, t_3)]]$ となる．
- 次に $t_3$ から $t_1$ へと進む．

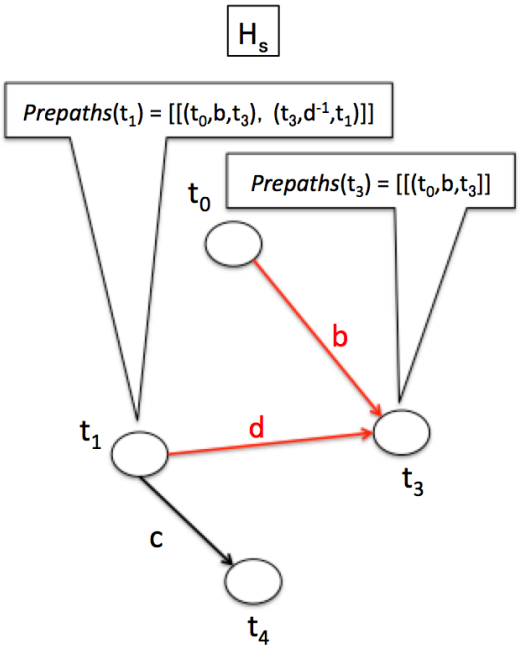


図 3.4: $H_s$ における探索 1

  - $Prepaths(t_1)$ は $[[(t_0, b, t_3)],\ [(t_3, d^{-1}, t_1)]]$ であり，$[(t_0, c, t_2), (t_2, c, t_3)]$ ($Prepaths(t_3)$ 内の経路の一つ) は含まれず，$path(= c.c)$ の経路上に $t_1$ もないため，問題なく遷移できる．
  - $Prepaths(t_1) = [[(t_0, b, t_3), (t_3, d^{-1}, t_1)], [(t_0, c, t_2), (t_2, c, t_3), (t_3, d^{-1}, t_1)]]$ となる．
- ここで，$current$ が $t_1$（終点ノード）となる．
  - $P$ に $path$ を追加 $\rightarrow$ $paths = [b.d^{-1}, c.c.d^{-1}]$
  - $path = c.c$
- これで，$t_0$（始点ノード）から $t_1$（終点ノード）に至る単純経路は全て辿ったこととなる．
- FindPaths の処理は終了し，Algorithm 1 の 19 行目に進む．
- ConstructPropertyPath を呼び出し，Property Path 式に変換する．最終的に以下の修正候補が出力される．
  - $b.d^{-1}.c$
  - $c.c.d^{-1}.c$

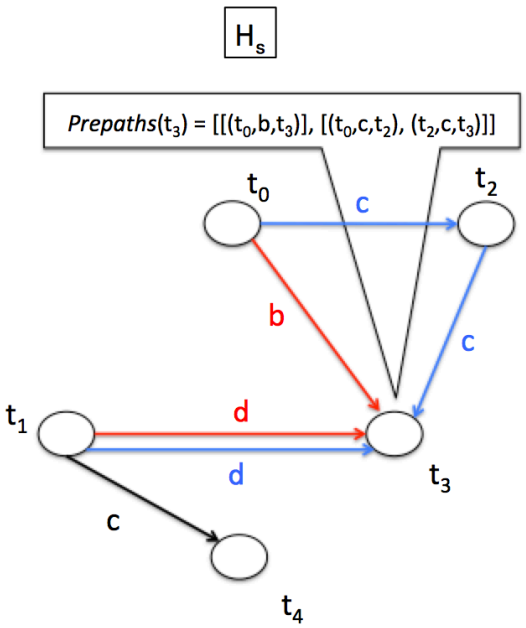


図 3.5: $H_s$ における探索 2

# 第4章　評価実験

本章では，提案アルゴリズムに関する評価実験について述べる．

実験データとして教科書 LOD[10][11] を使用した．教科書 LOD は，NIER 教育図書館と教科書研究センター図書館によって長年にわたって構築された教科書のコレクションを扱う RDF データである．教科書 LOD のデータ構造を図 4.1 に示す．Turtle 形式の 286,149 トリプルで構成されており， データサイズは 13.5MB となっている．教科書 LOD のデータの一部を図 4.2 に示す．

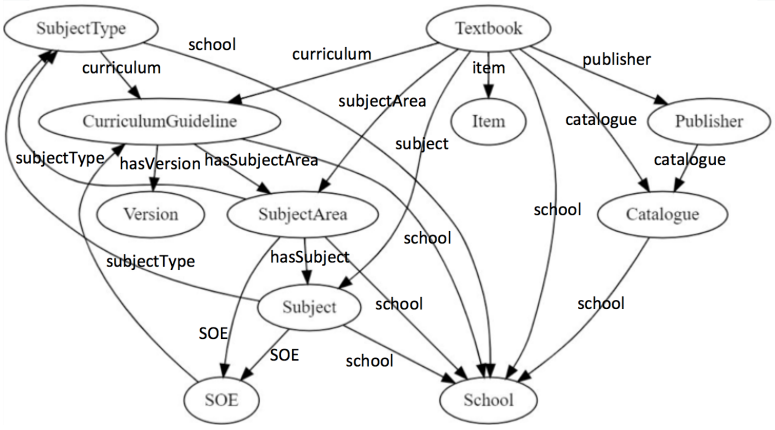


図 4.1: 教科書 LOD のデータ構造

本評価実験では，まず表 4.1 に示す 10 個の Property PAth 式とスキーマの更新操作を手動で作成した．次に，アルゴリズムによって各 Propertry Path 式を変換した．スキーマの更新によって RDF データの変換も必要となるため，文献 [9] の手法を用いて RDF データも変換した．元の Property Path 式と修正後の Property Path 式をそれぞれ元のデータと更新されたデータに対して実行し，再現率，適合率，F 値を計算した．ここで，$q$ を元の Propeerty Path 式，$q'$ を修正後の Property Path 式，$Ans(q)$ を $q$ の解の集合とすると，$q'$ の $q$ に対する再現率は次のように定義される．

$$recall(q, q') = \frac{|Ans(q) \cap Ans(q')|}{|Ans(q)|}$$

同様に，$q'$ の $q$ に対する適合率は次のように定義される．

$$precision(q, q') = \frac{|Ans(q) \cap Ans(q')|}{|Ans(q')|}$$

本実験における問合せ結果の指標として再現率，適合率を用いる理由としては，本研究では，変換後の Property Path 式は元の Property Path 式の解をできるだけ保存しているかどうかを評価するためである．例えば，変換後の Property Path 式の解の集合に不必要な

```
@prefix bf: <http://id.loc.gov/ontologies/bibframe/>.
@prefix curriculum: <https://w3id.org/jp-textbook/curriculum/>.
@prefix dct: <http://purl.org/dc/terms/>.
@prefix nier: <http://dl.nier.go.jp/library/vocab/>.
@prefix owl: <http://www.w3.org/2002/07/owl#>.
@prefix qb: <http://purl.org/linked-data/cube#>.
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#>.
@prefix schema: <http://schema.org/>.
@prefix sh: <http://www.w3.org/ns/shacl#>.
@prefix skos: <http://www.w3.org/2004/02/skos/core#>.
@prefix textbook: <https://w3id.org/jp-textbook/>.
@prefix textbook-rc: <http://dl.nier.go.jp/library/vocab/textbook-rc/>.
@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.
<https://w3id.org/jp-textbook/中学校/1992/保体/701> a textbook:Textbook;
  schema:name "新しい保健体育(中学校全)";
  schema:editor "秋山 房雄, 笠井 恵雄, 斎藤 キ能, ほか１４名";
  schema:publisher <https://w3id.org/jp-textbook/publisher/1992/東書>;
  textbook:item [
  nier:callNumber "K260.49||T8N||92/93";
  nier:recordID "EB10012620"
  ];
  textbook:catalogue <https://w3id.org/jp-textbook/catalogue/中学校/1992>, <https://w3id.org/jp-textbook/catalogue/中学校/1993>,
<https://w3id.org/jp-textbook/catalogue/中学校/1994>, <https://w3id.org/jp-textbook/catalogue/中学校/1995>;
  textbook:school <https://w3id.org/jp-textbook/school/中学校>;
  textbook:subjectArea <https://w3id.org/jp-textbook/curriculum/中学校/1993/保健体育>;
  textbook:subject <https://w3id.org/jp-textbook/curriculum/中学校/1993/保健体育/保健体育>;
  textbook:grade 1, 2, 3;
  textbook:curriculum <https://w3id.org/jp-textbook/curriculum/中学校/1993>;
  textbook:authorizedYear "1992";
  textbook:usageYear "1993-1996";
  bf:extent "160";
  bf:dimensions "B5";
  textbook:textbookSymbol "保体";
  textbook:textbookNumber "701".
```

図 4.2: 教科書 LOD のデータ（一部）

（元の Property Path 式の解とは異なる）情報が多く含まれていれば，それは適切な変換とは言えないと考えられる．そのため，本評価実験では，元の Property Path 式の解をどれだけ含むか（上記の再現率）だけでなく，元の Property Path 式の解に対して，どれだけ不必要な情報を含まないか（上記の適合率）も含めて算出している．

教科書 LOD における ShEx スキーマ $S = (\Sigma, \Gamma, \delta)$ を以下に示す．可読性を考慮し，型名を以下のように略記する．また，ラベル名の接頭辞も省略している．

- Literal $\rightarrow l_1$
- IRI $\rightarrow l_2$
- School $\rightarrow t_1$
- Catalogue $\rightarrow t_2$
- Publisher $\rightarrow t_3$
- SubjectType $\rightarrow t_4$
- SubjectArea $\rightarrow t_5$
- Subject $\rightarrow t_6$
- SOE $\rightarrow t_7$
- CurriculumGuideline $\rightarrow t_8$
- Version $\rightarrow t_9$
- Textbook $\rightarrow t_{10}$

- Item $\rightarrow t_{11}$

$\Sigma$ と $\Gamma$ は以下の通りである．

$$
\begin{aligned}
\Sigma &= \{name, sameAs, callNumber, dataPublished, itemID, recordID, school, \\
&\quad seeAlso, url, usageYear, catalogue, catalogueYear, note, publisherAbbreviation, \\
&\quad publisherNumber, citation, curriculum, hasSubject, order, soe, subjetType, \\
&\quad hasSubjectArea, hasVersion, startDate, isbn, authorizedYear, bookEdition, \\
&\quad dimensions, editor, extent, gradeitempublisher, subject, subjectArea, \\
&\quad textbookNumber, textbookSymbol, rc : callNumber, rc : recordID\} \\
\Gamma &= \{l_1, l_2, t_0, t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8, t_9, t_{10}, t_{11}\}
\end{aligned}
$$

また，$\delta$ は次のように定義される．

$$
\begin{aligned}
\delta(l_1) &= \epsilon \\
\delta(l_2) &= \epsilon \\
\delta(t_1) &= \text{name} :: {l_1}^{[0,3]} \parallel \text{sameAs} :: {l_2}^{?} \\
\delta(t_2) &= \text{callNumber} :: {l_1}^{?} \parallel \text{datePublished} :: l_1 \parallel \text{itemID} :: {l_1}^{?} \parallel \text{name} :: {l_1}^{[1,3]} \\
&\quad \parallel \text{recordID} :: {l_1}^{?} \parallel \text{school} :: t_1 \parallel \text{seeAlso} :: {l_2}^{*} \parallel \text{url} :: {l_2}^{?} \parallel \text{usageYear} :: l_1 \\
\delta(t_3) &= \text{catalogue} :: {t_2}^{+} \parallel \text{catalogueYear} :: l_1 \parallel \text{name} :: {l_1}^{[1,3]} \parallel \text{note} :: {l_1}^{?} \\
&\quad \parallel \text{publisherAbbreviation} :: l_1 \parallel \text{publisherNumber} :: {l_1}^{+} \parallel \text{seeAlso} :: {l_2}^{*} \\
\delta(t_4) &= \text{citation} :: l_1 \parallel \text{curriculum} :: t_8 \parallel \text{name} :: {l_1}^{[1,2]} \parallel \text{school} :: t_1 \\
\delta(t_5) &= \text{hasSubject} :: {t_6}^{*} \parallel \text{name} :: {l_1}^{[1,3]} \parallel \text{order} :: {l_1}^{+} \parallel \text{school} :: t_1 \parallel \text{soe} :: {t_7}^{*} \\
&\quad \parallel \text{subjectType} :: {t_4}^{*} \\
\delta(t_6) &= \text{citation} :: {l_1}^{?} \parallel \text{name} :: {l_1}^{[1,3]} \parallel \text{order} :: l_1 \parallel \text{school} :: t_1 \parallel \text{seeAlso} :: {l_2}^{*} \parallel \text{soe} :: {t_7}^{?} \\
&\quad \parallel \text{subjectType} :: {t_4}^{*}, \\
\delta(t_7) &= \text{curriculum} :: {t_8}^{?} \parallel \text{name} :: {l_1}^{[1,2]} \parallel \text{seeAlso} :: {l_2}^{*} \\
\delta(t_8) &= \text{datePublished} :: l_1 \parallel \text{hasSubjectArea} :: {t_5}^{+} \parallel \text{hasVersion} :: {t_9}^{*} \parallel \text{name} :: {l_1}^{[1,3]} \\
&\quad \parallel \text{school} :: t_1 \parallel \text{startDate} :: l_1 \parallel \text{url} :: {l_2}^{*} \\
\delta(t_9) &= \text{callNumber} :: {l_1}^{?} \parallel \text{citation} :: {l_1}^{[1,2]} \parallel \text{datePublished} :: l_1 \parallel \text{isbn} :: {l_1}^{*} \\
&\quad \parallel \text{itemID} :: {l_1}^{?} \parallel \text{name} :: {l_1}^{[1,2]} \parallel \text{recordID} :: {l_1}^{?} \parallel \text{seeAlso} :: {l_2}^{*} \parallel \text{url} :: {l_2}^{*} \\
\delta(t_{10}) &= \text{authorizedYear} :: {l_1}^{*} \parallel \text{bookEdition} :: {l_1}^{*} \parallel \text{catalogue} :: t2^{+} \parallel \text{curriculum} :: t_8 \\
&\quad \parallel \text{dimensions} :: {l_1}^{?} \parallel \text{editor} :: l_1 \parallel \text{extent} :: {l_1}^{?} \parallel \text{grade} :: {l_1}^{[0,6]} \parallel \text{isbn} :: {l_1}^{*} \\
&\quad \parallel \text{item} :: {t_{11}}^{*} \parallel \text{name} :: l_1 \parallel \text{note} :: {l_1}^{*} \parallel \text{publisher} :: {t_3}^{+} \parallel \text{school} :: t_1 \\
&\quad \parallel \text{seeAlso} :: {l_2}^{*} \parallel \text{subject} :: {t_6}^{?} \parallel \text{subjectArea} :: t_5 \parallel \text{textbookNumber} :: l_1 \\
&\quad \parallel \text{textbookSymbol} :: l_1 \parallel \text{usageYear} :: l_1 \\
\delta(t_{11}) &= (\text{callNumber} :: l_1 \parallel \text{recordID} :: l_1) \mid (\text{rc:callNumber} :: l_1 \parallel \text{rc:recordID} :: l_1)
\end{aligned}
$$

以下，評価実験で用いた Property Path 式および更新操作，得られた結果について述べる．まず，Property Path 式とそれに対する更新操作は 10 組あり，それぞれの内容と修復箇所は以下の通りである（実験 1〜実験 10 と称する）．始点，終点とは，Property Path 式の経路全体におけるものである．

- 実験 1
  - Property Path 式:catalogue.school
  - 始点:Textbook 型 終点:Catalogue 型
  - 更新操作 1:Textbook 型から Catalog 型に出力されるエッジを削除
  - 更新操作 2:extbook 型から SubjectType 型に出力されるエッジを追加
  - 修復箇所:Textbook 型から Catalogue 型への経路
- 実験 2
  - Property Path 式:$\text{catalogue}^{-1}.\text{publisher}^{-1}$.curriculum.hasSubjectArea.hasSubject
  - 始点:Catalogue 終点:Subject
  - 更新操作 1:Publisher 型の削除
  - 更新操作 2:Catalogue 型から School 型へのエッジを削除
  - 修復箇所:Catalogue 型から Textbook 型への経路
- 実験 3
  - Property Path 式:$\text{curriculum}^{-1}$.school
  - 始点:CurriculumGuideline 型 終点:School 型
  - 更新操作 1:Textbook 型から School 型に出力されるエッジを削除
  - 更新操作 2:SubjectArea 型の削除
  - 更新操作 3:Subject 型の削除
  - 修復箇所:Textbook 型から School 型への経路
- 実験 4
  - Property Path 式:(catalogue | subjectArea).school
  - 始点:Textbook 型 終点:School 型
  - 更新操作 1:Textbook 型から Catalogue 型に出力されるエッジを削除
  - 更新操作 2:SubjectType 型に Subject 型へのエッジを追加
  - 修復箇所:Textbook 型から Catalogue 型への経路
- 実験 5
  - Property Path 式:$\text{subjectArea}^{-1}$.curriculum.hasSubjectArea.hasSubject.school
  - 始点:SubjectArea 型 終点:School 型
  - 更新操作 1:Subject 型の削除
  - 更新操作 2:CurriculumGuideline 型から Version 型へのエッジのラベル名「version」に変更
  - 修復箇所:SubjectArea 型から School 型への経路
- 実験 6
  - Property Path 式:$\text{curriculum}^{-1}$.catalogue.school

  - 始点:CurriculumGuideline 型 終点:School 型
  - 更新操作 1:Textbook 型から CurriculumGuideline 型へのエッジのラベル名を「curriculumguideline」に変更
  - 更新操作 2:Textbook 型から Catalogue 型へのエッジを削除
  - 修復箇所 1:Property Path 式における「curriculum」を「curriculumguideline」に変更
  - 修復箇所 2:Textbook 型から Catalogue 型への経路
- 実験 7
  - Property Path 式:curriculum.hasSubjectArea.hasSubject
  - 始点:Textbook 型 終点:Subject 型
  - 更新操作 1:CurriculumGuideline 型から SubjectArea 型へのエッジのラベル名を「subjectArea」に変更
  - 更新操作 2:SubjectArea 型から Subject 型へのエッジのラベル名を「subject」に変更
  - 修復箇所 1:Property Path 式における「hasSubjectArea」を「subjectArea」に変更
  - 修復箇所 2:Property Path 式における「hasSubject」を「subject」に変更
- 実験 8
  - Property Path 式:subjectArea.hasSubject.school
  - 始点:Textbook 型 終点:Subject 型
  - 更新操作 1:Textbook 型から SubjectArea 型へのエッジを削除
  - 更新操作 2:CurriculumGuideline 型から SubjectArea 型へのエッジのラベル名を「hasArea」に変更
  - 更新操作 3:Subject 型の削除
  - 修復箇所:Textbook 型から SubjectArea 型への経路
- 実験 9
  - Property Path 式:$\text{curriculum}^{-1}.(\text{catalogue} \mid \text{publisher.catalogue})^{*}.\text{school}$
  - 始点:CurriculumGuideline 型 終点:School 型
  - 更新操作 1:Textbook 型から Catalogue 型へのエッジを削除
  - 更新操作 2:Textbook 型から School 型へのエッジを削除
  - 更新操作 3:Subject 型の削除
  - 修復箇所:Textbook 型から School 型への経路
- 実験 10
  - Property Path 式:$\text{catalogue}^{-1}.(\text{publisher}^{-1})^{*}.(\text{subjectArea.hasSubject}|\text{subject}).\text{school}$
  - 始点:Catalogue 型 終点:School 型
  - 更新操作 1:Textbook 型から Catalogue 型へのエッジのラベル名を「list」に変更

- 更新操作 2:Publisher 型から Catalogue 型へのエッジのラベル名を「list」に変更
- 更新操作 3:CurriculumGuideline 型から SubjectArea 型へのエッジのラベル名を「subjectArea」に変更
- 更新操作 4:SubjectArea 型から Subject 型へのエッジのラベル名を「subject」に変更
- 更新操作 5:Textbook 型から SubjectArea 型へのエッジを削除
- 更新操作 6:Subject 型の削除
- 修復箇所 1:Property Path 式における「catalogue」を「list」に変更
- 修復箇所 2:Property Path 式における「hasSubject」を「subject」に変更
- 修復箇所 3:Textbook 型から SubjectArea 型への経路

表 4.1 に修正前の Property Path 式および更新操作の一覧を示す．また，表 4.2 には，修正後の Property Path 式，およびそれらの再現率，適合率，F 値の一覧を示す．2 つの表における番号は対応している．

結果として，10 個の実験における平均 F 値は 0.87 であるため，修正された Property Path 式は全体的に概ね良好に動作したと考えられる．ただし，3 番目，5 番目，および 7 番目の実験以外では，すべての正解を得られたわけではないことが分かる．

その理由としては次のようなことが考えられる．教科書 LOD のスキーマには,「*」または「?」等のオペレータが付随するエッジが含まれている． そのようなエッジは「オプション」であり，対応するエッジが実際の RDF データに表示されない場合がある．したがって，変換された Property Path 式にそのような「オプション」のエッジのラベルが含まれている場合，修正された Property Path 式によって得られる解は，元の Property Path 式の解と一致しない可能性がある． 例えば，実験 2 の Property Path 式の「hasSubject」にはそのような「オプション」としての性質がある．

表 4.1: 入力 Property Path 式および更新操作

| No. | (a) Property Path 式 (b) 更新操作 |
|---|---|
| 1 | (a) catalogue.school |
| | (b) $del\_lt(Textbook, 6)$ $add\_lt(Textbook, 1, subjectType :: SubjectType)$ |
| 2 | (a) catalogue$^{-1}$.publisher$^{-1}$.curriculum.hasSubjectArea.hasSubject |
| | (b) $del\_type(Publisher)$ $del\_lt(Catalogue, 1)$ |
| 3 | (a) curriculum$^{-1}$.school |
| | (b) $del\_lt(Textbook, 5)$ $del\_type(SubjectArea)$ $del\_type(Subject)$ |
| 4 | (a) (catalogue \| subjectArea).school |
| | (b) $del\_lt(Textbook, 6)$ $add\_lt(SubjectType, 3, hasSubject :: Subject)$ |
| 5 | (a) subjectArea$^{-1}$.curriculum.hasSubjectArea.hasSubject.school |
| | (b) $del\_type(Subject)$ $change\_lt(CurriculumGuideline, 1, version :: Version)$ |
| 6 | (a) curriculum$^{-1}$.catalogue.school |
| | (b) $change\_lt(Textbook, curriclum, CurriculumGuideline, Textbook, curriclumguideline,$ $CurriculumGuideline)$ $del\_lt(Textbook, catalogue, Catalogue)$ |
| 7 | (a) curriculum.hasSubjectArea.hasSubject |
| | (b)$change\_lt(CurriculumGuideline, hasSubjectArea, SubjectArea, CurriculumGuideline,$ $subjectArea, SubjectArea)$ $change\_lt(SubjectArea, hasSubject, Subject, SubjectArea, subject, Subject)$ |
| 8 | (a) subjectArea.hasSubject.school |
| | (b) $del\_lt(Textbook, subjectArea, SubjectArea)$ $change\_lt(CurriculumGuideline, hasSubjectArea, SubjectArea, CurriculumGuideline,$ $hasArea, SubjectArea)$ $del\_type(SubjectType)$ |
| 9 | (a) curriculum$^{-1}$.(catalogue \| publisher.catalogue)$^*$.school |
| | (b) $del\_lt(Textbook, school, School)$ $del\_lt(Textbook, catalogue, Catalogue)$ $del\_type(SubjectType)$ |
| 10 | (a) catalogue$^{-1}$.(publisher$^{-1}$ )$^*$.(subjectArea.hasSubject\|subject).school |
| | (b) $change\_lt(Textbook, catalogue, List, Textbook, list, List)$ $change\_lt(Publisher, catalogue, List, Publisher, list, List)$ $change\_lt(CurriculumGuideline, hasSubjectArea, SubjectArea, CurriculumGuideline,$ $subjectArea, SubjectArea)$ $change\_lt(SubjectArea, hasSubject, Subject, SubjectArea, subject, Subject)$ $del\_lt(Textbook, subjectArea, SubjectArea)$ $del_itype(SubjectType)$ |

表 4.2: 修正 Property Path 式の再現率，適合率，F 値

| No. | 修正 Property Path 式 | 再現率 | 適合率 | F 値 |
|---|---|---|---|---|
| 1 | publisher.catalogue.school | 0.88 | 0.99 | 0.93 |
| 2 | catalogue$^{-1}$.curriculum.hasSubjectArea.hasSubject | 0.70 | 0.50 | 0.59 |
| 3 | curriculum$^{-1}$.publisher$^*$.catalogue.school | 1.00 | 1.00 | 1.00 |
| 4 | (publisher.catalogue \| subjectArea).school | 0.88 | 0.77 | 0.82 |
| 5 | (subjectArea$^{-1}$.curriculum.hasSubjectArea)$^*$.subjectType$^*$.school | 1.00 | 1.00 | 1.00 |
| 6 | curriculumguideline$^{-1}$.publisher.catalogue.school | 0.87 | 0.98 | 0.90 |
| 7 | curriculum.subjectArea.subject | 1.00 | 1.00 | 1.00 |
| 8 | curriculum.hasArea.hasSubject.school | 0.97 | 0.78 | 0.87 |
| 9 | curriculum$^{-1}$.(publisher.catalogue \| subjectArea.hasSubject$^*$\|subject). school | 0.87 | 0.78 | 0.83 |
| 10 | list$^{-1}$.(publisher$^{-1}$)$^*$.(curriculum.subjectArea)$^*$.subject.school | 0.90 | 0.74 | 0.82 |
| 平均 | | 0.90 | 0.85 | 0.87 |

# 第5章　まとめ

本論文では，ShEx スキーマの更新に従って Property Path 式を修正するためのアルゴリズムを提案した．評価実験では，アルゴリズムによって導いた修正 Property Path 式の再現率，適合率，F 値を算出した．その結果，概ね良好な結果が得られた．

しかし，本研究にはいくつかの課題は残っている．まずは，より様々なデータセットで実験を行うことである．現時点では限られたデータセットのため，今後は多種多様なデータ構造に対して実験を行い，どのような結果が表れるかを見ていく必要があると考えられる．加えて，今回の実験に用いた更新操作は著者が作成したものに限られているため，実際の更新操作を用いて実験を行なう必要がある．また，現時点の研究では否定表現等，ShEx における定義において，まだ対応できていない要素があるため，そのための拡張が必要である．さらには，問合せ式修正の際に，補完する経路がその時点（スキーマ更新後）で存在するかどうかを事前に判断する機能を加える必要もあると考えられる．

# 謝辞

本研究を進めるに当たって，多くのや励ましやご指導・ご助言をいただいた鈴木伸崇先生に深く感謝いたします．先生のご指導・ご協力がなければ，本論文の執筆は可能ではありませんでした．併せて，多くのご指導・ご助言をいただいた阪口哲男先生，高久雅生先生，森嶋厚行先生に深く感謝いたします．また，ともに切磋琢磨した鈴木伸崇研究室の皆様，本当にありがとうございました．

# 参考文献


[1] World Wide Web Consortium (W3C). RDF Schema 1.1. https://www.w3.org/TR/rdfschema/. Accessed: 2020-12-28.

[2] World Wide Web Consortium (W3C). Shape Expressions (ShEx) 2.1 Primer. http://shex.io/shex-primer/. Accessed: 2020-12-28.

[3] World Wide Web Consortium (W3C). Shape Expressions Language 2.1. http://shex.io/shex-semantics/. Accessed: 2019-12-28.

[4] World Wide Web Consortium (W3C).“SPARQL 1.1 Property Paths”. https://www.w3.org/TR/sparql11-property-paths/. Accessed: 2020-12-28.

[5] K.Hasegawa，K.Ikeda，and N.Suzuki.“An Algorithm for Transforming Xpath Expressions According to Schema Evolution”，Proceeding of the First International Workshop on Document Changes: Modeling Detection，Storage and Visualization，p.8，2013.

[6] R. Oliveira , P. Genevès , and N. Layaïda.“Toward automated schema-directed code revision,”Proc. ACM DocEng, 2012.

[7] Y. Wu and N. Suzuki,“An algorithm for correcting XSLT rules according to DTD updates,”Proc. DChanges, 2016.

[8] S. Staworko, I. Boneva, J. Labra Gayo, S. Hym, E. Prud'hommeaux and H. Sorbrig.,“Complexity and Expressiveness of ShEx for RDF,”Proc. ICDT, 2015, pp.195-211.

[9] 松原尚利．Shape Expression Schema の更新操作とそれに伴うグラフデータの修正．筑波大学，卒業論文，2019.

[10] 江草由佳, 高久雅生. 教科書 Linked Open Data (LOD) の構築と公開. 情報の科学と技術, Vol. 68, No. 7, pp. 361–367, 2018.

[11] 江草由佳, 高久雅生. 教科書 Linked Open Data (LOD). https://jp-textbook.github. io/. Accessed: 2019-12-27.