Shape Expression Schemaの下でのパターン問合せ充足可能性判定アルゴリズム

松岡　栞†　鈴木　伸崇††

† 筑波大学図書館情報メディア研究科図書館情報メディア専攻　〒305–8550 茨城県つくば市春日 1-2
†† 筑波大学図書館情報メディア系　〒305–8550 茨城県つくば市春日 1-2
E-mail: †s1921645@s.tsukuba.ac.jp, ††nsuzuki@slis.tsukuba.ac.jp

あらまし　Shape Expression Schema（ShEx）はグラフデータのスキーマを記述するために新たに提案されたスキーマ言語である．一般に，グラフデータは莫大なデータ量をもつ．このようなデータに対してパターン問合せを行う場合，その莫大なデータの中からマッチするパターンを検索しなければならない．ここで，パターン P と ShEx S に対して，S に妥当かつ P を含むようなグラフデータ（RDF データ）が存在しない場合，P は S の下で充足不能であるという．充足不能なパターンで問合せを行うと，膨大なデータを検索したのちに解が存在しないという結果が返される．しかし，その検索は明らかに無駄である．そのため，充足不能なパターンを ShEx の定義から発見できることが望ましい．そこで，本稿では，ShEx の下でのパターン問合せ充足可能性問題を解くための手法を提案する．

キーワード　Shape Expression Schema，パターン問合せ，充足可能性判定

1 はじめに

近年，RDF データ/グラフデータは急速に増加し，様々なところで用いられている．これまで，RDF のスキーマ言語としては RDF Schema (RDFS) が用いられてきた．しかし，RDFS はスキーマ記述言語というよりもオントロジー記述言語としての性格が強く，スキーマの定義には必ずしも適していない [7]．そのため，新たなスキーマ言語として，Shape Expression Schema（以下 ShEx）が提案され，Shape Expressions Community Group で検討が進められている [1]．

パターン問合せは，問合せとしてグラフの断片（パターン）を指定してグラフからそのパターンにマッチする部分を検索するもので，グラフに対する最も基本的かつ重要な問合せである．パターン P と ShEx スキーマ S に対して，S に妥当かつ P を含むようなグラフデータ（RDF データ）が存在しない場合，P は S に関して充足不能であると言う．充足不能な問合せを実行した場合，膨大なデータを検索した後に解が存在しないという結果が返されることになるが，その検索は明らかに無駄である．一方，充足可能性はグラフデータを参照せず，スキーマのみで判定可能である．したがって，充足不能な問合せを実行前に効率よく検出することができれば有用であると考えられる．

そこで本研究では，ShEx スキーマの下でパターン問合せの充足可能性を判定するアルゴリズムを提案する．より具体的には，まず ShEx をエッジ毎に分解・表現し，内容モデルの選言や multitype 性（1 つのノードが複数の型を取り得ること）を考慮しつつ ShEx スキーマの型とパターンのノード全てがマッチするか否かを判定し，マッチする場合は充足可能として処理を終了する．

関連研究として，DTD の下での XPath 充足可能性問題に関する研究は数多く存在する（文献 [2], [3], [4] など）．また，DTD の下での CSS 規則の充足可能性問題が文献 [5] で考察されている．しかし，著者の知る限り，ShEx の下でのパターン問合せ充足可能性問題を考察した研究は存在しない．なお，本稿は文献 [6] に基づいてアルゴリズムの構成に修正を加え，証明や評価実験を追加したものである．

2 諸定義

Σ 上のラベル付き有向グラフ（以下，単にグラフ）を $G = (V, E)$ と表す．ここで，Σ はラベルの集合，V はノードの集合，$E \subseteq V \times \Sigma \times V$ はラベル付き有向辺の集合である．また，エッジを出力しているノードを出力ノード，エッジの向かう先である受け取り側のノードを入力ノードと呼称する．各ノードは出力ノードである場合，入力ノードである場合，さらに，出力ノードであり入力ノードである場合がある．

DTD や XML Schema 等，多くのスキーマでは型を定義する際に Regular Expression（正規表現）を用いる．一方，グラフデータでではノード間の順序が考慮されないことが多いため，ShEx スキーマでは Regular Expression ではなく Regular Bag Expression (RBE) を用いている．Regular Expression と異なり，RBE は連結 “||” において順序が無視される．RBE は以下のように定義される．

- ϵ と $a \in \Sigma$ 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，$E_1 \parallel E_2 \parallel \cdots \parallel E_k$ 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^* は RBE である．
- E が RBE であるならば，$E^{[n:m]}$ は RBE である．

このとき，ShEx スキーマは $S = (\Sigma, \Gamma, \delta)$ と表現される．ここで，Γ は型の集合，δ は Γ から $\Sigma \times \Gamma$ 上の RBE への関数である．ShEx スキーマには，1 つのノードが 1 つの型のみをもつ single type semantics と複数の型をもつことのできる multi type semantics とがある [7]．

本研究では，問合せとしてパターンを考える．ここで，パターンもグラフとして表される．パターン問合せでは，データグラフにおいてパターンと同型である（マッチする）部分グラフが解となる．パターン問合せ P と ShEx スキーマ S に対して，もし S に妥当かつ P の問合せ結果が空でないグラフデータが存在するならば，P は S の下で充足可能であるという．

3 提 案 手 法

本章では，ShEx の下でのパターン問合せ充足可能性判定アルゴリズムについて述べる．

3.1 前 処 理

パターングラフの候補ノード集合を求めるために，ShEx スキーマを修正 AND/OR グラフとして表す．従来の AND/OR グラフは同時に存在する「AND」のエッジに円弧を描くが，今回は OR 分岐に着目しているため，図 1 のように「OR」のエッジ間に円弧を描く．ShEx スキーマ $S = (\Sigma, \Gamma, \delta)$ に対して，まず，グラフ $G_S = (V_S, E_S)$ を構成する．ここで，

$$V_S = \Gamma$$
$$E_S = \{(t, l, t') \mid \delta(t) \text{ が } l :: t' \text{ を含む } \}$$

得られた G_S に対して，S の内容モデルにおいて選言で接続されており，かつ，$*$ や $+$ での繰り返しが指定されていないものに対して OR の円弧を付加する．例えば，ShEx スキーマ $S = (\{a, b, c\}, \Gamma, \delta)$ を考える．ここで，$\Gamma = \{t_0, t_1, t_2, t_3\}$ かつ δ は次のように定義される．

$$\delta(t_0) = a :: t_1 | a :: t_2 | b :: t_3$$
$$\delta(t_1) = b :: t_3$$
$$\delta(t_2) = c :: t_3$$
$$\delta(t_3) = \epsilon$$

このとき，S は図 1 の修正 AND/OR グラフで表される．ノード t_0 から伸びる 3 つのエッジは同時に使用不可能である．

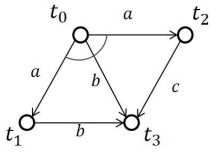


図 1 修正 AND/OR グラフ

single type semantics の場合はこのスキーマグラフを用いる．multi type semantics の場合を考える．この場合は複数の型をもつノードが存在し得るため，複数の型を組み合わせたノードの形成が必要となる．文献 [7] より，multi type semantics において，ノードに対する型集合の割り当て λ が妥当であるための条件は以下のように指定されている．

> ShEx S とグラフデータ G に対して，G のノードに対する（multi-type の）型の割り当て λ が以下の条件を満たすとき，λ は S の下で妥当であるという．
> 1. すべてのノード $n \in V$ について少なくとも 1 つの型が割り当てられている．すなわち．$\lambda(n) \neq \emptyset$ である．
> 2. 各ノード n に割り当てられたそれぞれの型 $t \in \lambda(n)$ に対して，n の出力エッジが $\delta(t)$ を満たす．

上記規則に基づいて，型 t_1 と t_2 を組み合わせた型を次の手順で生成する．

（1） 出力先の型を無視して，$\delta(t_1)$ と $\delta(t_2)$ の互いに共通する部分を求める．例えば，$\delta(t_1) = a :: t_3 | b :: t_4$，$\delta(t_2) = a :: t_2$ の場合，$\delta(t_1) = a :: t_3$，$\delta(t_2) = a :: t_2$ となる．共通する部分がない場合，t_1 と t_2 を組み合わせた型は生成しない．

（2） $\delta(t_1)$ と $\delta(t_2)$ において，対応する「ラベル::型」に対して，出力先の型を組み合わせる．上記の例では，t_1 の $a :: t_3$ と t_2 の $a :: t_2$ が対応するので，これを組み合わせて $a :: t_{23}$ が得られる．よって，$\delta(t_{12}) = a :: t_{23}$ となる．

型の組合せ及び平坦化の処理を行うと，以下の型が新しく生成される．

$$\delta(t_{01}) = b :: t_3$$

t_{01} 以外は空集合となるため，他の新しい型は生成されない．上記のように生成された型を元の ShEx スキーマに含めた上で，アルゴリズムを適用する．

3.2 アルゴリズム

「本稿のアルゴリズムは，パターングラフ P と ShEx スキーマのスキーマグラフ S に対して，P と S との間の部分グラフ同型問題を解くことを基礎とする．しかし，ShEx が選言を含む場合，部分グラフ同型問題をそのまま解くだけでは正しく判定を行うことができない．なぜならば，選言を含むエッジを同時に選択する必要のあるパターンで問合せを行った場合，同時に使用可能であるかを判定することができず，同時には存在し得ないエッジを使用してしまう場合があるからである．また，出力エッジ数の制限についても正しく判定する必要がある．このような制約を考慮して，パターングラフは ShEx の下で生成可能であるか判定することで，ShEx におけるパターン問合せの充足可能性判定ができるようにする．提案アルゴリズムを Algorithm 3.1（アルゴリズム本体）および Algorithm 3.2（Algorithm 3.1 から呼び出される関数）に示す．

Algorithm 3.1 において，1 行目の M はパターングラフとスキーマグラフ間でマッチしたノードの組を格納する集合である．2 行目の関数 MakeSchemaGraph は，修正 AND/OR グラフ g として ShEx スキーマ $S = (\Sigma, \Gamma, \delta)$ をグラフ化する関数である．3 行目の関数 ShExDecompose は，ShEx スキーマ $S = (\Sigma, \Gamma, \delta)$ をエッジ (ラベル) 毎に分解する関数である．こ

Algorithm 3.1 SATISFIABILITY DECISION

Input: パターングラフ $q = (V(q), E(q))$, スキーマ $S = (\Sigma, \Gamma, \delta)$
Output: 充足可能/充足不能

```
 1: M := ∅; SB := ∅; PB := ∅;
 2: g = MakeSchemaGraph(S);
 3: SB := ShExDecompose(S);
 4: PB := PatternDecompose(q);
 5: for each u ∈ V(q) do
 6:    C(u) := FilterCandidates(q, g, u);
       [[∀t ∈ C(u)((t ∈ V(g)) ∧ (L(u) ⊆ L(t)))]]
 7:    if C(u) = ∅ then
 8:       report 充足不能;
 9:    end if
10: end for
11: SubgraphSearch(q, S, g, M, SB, PB)
```

Algorithm 3.2 SUBGRAPH SEARCH

Input: パターングラフ q, スキーマ S, スキーマグラフ g, ノード組集合 M, S のエッジ集合 SB, q のエッジ集合 PB
Output: 充足可能/充足不能

```
 1: if |M| = |V(q)| then
 2:    report 充足可能;
 3: else
 4:    u := NextqueryVertex(q, PB, M);
 5:    for each t ∈ C such that v is not yet matched do
 6:       if IsOffalBranch(u, t, q, S, g, M, SB, PB) then
 7:          if IsUsableOneTime(u, t, q, S, g, M, SB, PB) then
 8:             if IsUsableNTimes(u, t, q, S, g, M, SB, PB) then
 9:                UpdateState(M, u, t);
                   [[(u, t) ∈ M]]
10:                SubgraphSearch(q, S, t, M, SB, PB);
11:                RestoreState(M, u, t);
                   [[(u, t) ∉ M]]
12:             end if
13:          end if
14:       end if
15:    end for
16:    report 充足不能;
17: end if
```

の際，型の定義について，選言「|」が存在する場合は同時に使用不可能なエッジ集合に追加する．0 または 1 回の出現を表現する「?」，連結「||」での接続は一度のみの出現のエッジ集合に追加する．さらに，$[n, m]$ については m 回までの出現として使用回数制限のあるエッジ集合に追加する．0 回以上の繰り返しを表現する「$*$」，1 回以上の繰り返しを表現する「+」については繰り返し使用に制限を設けないとする．選言「|」で接続されたノードが繰り返し処理の中に含まれていた場合は充足可能性には影響を及ぼさないため，この場合も繰り返し使用に制限を設けない．4 行目の関数 PatternDecompose はパターングラフ q をエッジ毎に分解する関数である．5〜10 行目で，パターングラフ q の各ノード u に対して，u にマッチし得る候補ノード集合 $C(u) \subseteq V(g)$ を求める．ここで，$C(u)$ は V に属するノードのうち，出力エッジラベルの集合が u と一致するものからなる．6 行目の関数 FilterCandidates は，前処理で作成されたスキーマグラフのノードのうち，入出力エッジのラベルが u と一致するものを求める関数である（6 行目の下の式は，$C(u)$ が満たすべき条件を記述している）．11 行目で SubgraphSearch により q を再帰的に探索し，パターン問合せを実行する．

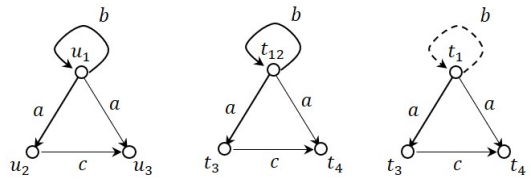
Algorithm 3.2 に SubgraphSearch を示す．1 行目の M は，パターングラフ q とスキーマグラフ g 間でマッチするノード組の集合である．1 行目でパターングラフのノード数と M のサイズが一致しているか判定する．一致していれば「充足可能」と出力し終了，不一致であれば 3 行目以降へと進む．4 行目の NextQueryVertex はパターングラフを分解したノード集合の次の（未選択）ノード u を取得する関数である．5〜8 行目でノード u についての各候補ノード $v \in C(u_{out})$ が u とマッチするかを判定する．選言等の条件を考慮した場合に，u と t が実際にマッチ可能か否かを以下のようにして判定する．

（1） 6 行目の関数 IsOffalBranch は，t が ShExDecompose で作成した条件のひとつである「選言分岐に含まれる」をもつか判定する関数である．条件に当たらない場合，いずれのエッジを選択しても充足可能性に影響しないため，true を返し次に進む．条件に当たる場合で，かつ M に属する（すなわち，マッチング済みの）エッジと t とが OR 分岐しているうちの二本 (あるいはそれ以上) あった場合，それらのエッジを同時に使用 してはならないため，t を u にマッチさせるのは不適切であり，false を返して 4 行目の次のノード候補へと戻る．

（2） 7 行目の関数 IsUsableOneTime は，t が ShExDecompose で作成した条件のひとつである「一度のみの出現」をもつか判定する関数である．条件に当たらない場合，いずれのエッジを選択しても充足可能性に影響しないため，true を返し次に進む．条件に当たる場合で，かつ M に属する（すなわち，マッチング済みの）エッジが t と同じエッジを使用している場合，それらのエッジを同時に使用してはならないため，t を u にマッチさせるのは不適切であり，false を返して 4 行目の次のノード候補へと戻る．

（3） 8 行目の関数 IsUsableNTimes は，t が ShExDecompose で作成した条件のひとつである「N 回までの出現」をもつか判定する関数である．条件に当たらない場合，いずれのエッジを選択しても充足可能性に影響しないため，true を返し次に進む．条件に当たる場合で，かつ t と同じエッジであり M に属する（すなわち，マッチング済みの）エッジの合計本数が N 回を上回った場合，それらのエッジを同時に使用してはならないため，t を u にマッチさせるのは不適切であり，false を返して 5 行目の次のノード候補に移る．

以上で u と t がマッチすると判定された場合，9 行目の関数 UpdateState によって組 (u, t) を M に追加する．10 行目で SubgraphSearch を再帰的に呼び出す．マッチするものが見つからず，かつ，マッチング中の階層での探索をすべて終えた場合は 11 行目の関数 RestoreState によって M を 9 行目の状態に戻し，次の候補にあたる．すべての候補ノードを探索し終えた場合，16 行目で「充足不能」と出力し終了する．



a. パターングラフ　b. マッチするグラフ　c. マッチしないグラフ

図 2　Single type semantics と multi type semantics の違い

$$\delta(t_1) = b :: t_2 || a :: t_3 || a :: t_4$$
$$\delta(t_2) = b :: t_1 || a :: t_3 || a :: t_4$$
$$\delta(t_3) = c :: t_4$$
$$\delta(t_4) = \epsilon$$

上記の ShEx スキーマに対して図 2a のパターングラフを用いて充足可能性を判定する．これらの型を組み合わせて新規生成可能なのは

$$\delta(t_{12}) = b :: t_1 2 || a :: t_3 || a :: t_4$$

であり，この型を用いると図 2b で示すようにスキーマグラフとマッチし，充足可能であるとされる．もし仮に multi type semantics を考慮しないとすると，図 2c に示すように，$t_1 \xrightarrow{b} t_1$ あるいは $t_2 \xrightarrow{b} t_2$ が存在しないことから，パターングラフの $u_1 \xrightarrow{b} u_1$ とマッチしないため，充足可能であるとの判定が得られない．

以下，アルゴリズムの正当性を示す．

定理 1：Algorithm 3.1 の出力が「充足可能」であることと，パターングラフ q が ShEx スキーマ S の下で充足可能であることは同値である．

証明の概要：q のノードを $u_1, u_2, \cdots, u_n$ とする．Algorithm 3.2 の 4 行目において，NextqueryVertex はこの順で q のノードを出力するとする．

($\Rightarrow$) Algorithm 3.1 が「充足可能」を出力し，その時点で M に n 個の組 $(u_1, t_1), (u_2, t_2), \cdots, (u_n, t_n)$ が格納されていたとする．v_i を型 t_i をもつノードとすると，$v_1, v_2, \cdots, v_n$ を含む S に妥当なグラフで，$v_1, v_2, \cdots, v_n$ から構成される部分グラフが q と同型であるようなものを構成できる．

($\Leftarrow$) q が S の下で充足可能であるとする．このとき，S に妥当なグラフ G で，q と同型な部分グラフを含むものが存在する．この部分グラフを $G(q)$ とする．$G(q)$ のノード集合を $\{v_1, v_2, \cdots, v_n\}$ とし，q のノード u_i と $G(q)$ のノード v_i が対応しているとする．まず，single-type semantics の場合を考える．このとき，G において v_i は 1 つの型（t_i と表す）をもつ．$G(q)$ と同型で，v_i を t_i に置き換えたグラフを G_t とし，列 $(u_1, t_1), (u_2, t_2), \cdots, (u_n, t_n)$ を考える．v_i は妥当なグラフ $G(q)$ に含まれておりかつ $G(q)$ と G_t は同型なので，Algorithm 3.2 の 5 行目で t_i が選択された時点で，t_i は Algorithm 3.2 における「選言分岐」「一度のみの出現」「N 回までの出現」の条件を満たす．次に，multi-type semantics の場合を考える．このとき，v_i は複数の型をもつ可能性がある．G における v_i の型集合を $\{t_{i1}, t_{i2}, \cdots, t_{ik}\}$ とし，$G(q)$ において v_i を $t_{i1i2\cdots ik}$ で置き換えたグラフを G_t とする．G は妥当なので，multi-type semantics の定義から v_i は $t_{i1}, t_{i2}, \cdots, t_{ik}$ をいずれも満たす．$t_{i1i2\cdots ik}$ の内容モデルは $t_{i1}, t_{i2}, \cdots, t_{ik}$ の内容モデルの共通部分から構成されるので，$t_{i1i2\cdots ik}$ が Algorithm 3.2 の 5 行目で選択された時点で，$t_{i1i2\cdots ik}$ は「選言分岐」「一度のみの出現」「N 回までの出現」の条件を満たす．　□

4　パターン問合せ充足可能性問題の計算複雑さ

本章では，パターン問合せ充足可能性問題の計算複雑さについて考察する．

定理 2:パターン問合せ充足可能性問題は，single type semantics に限定しても NP 困難である．

証明：本問題の NP 困難性を部分グラフ同型問題からの帰着により示す．部分グラフ同型問題の入力であるパターングラフ p とデータグラフ g に対して，スキーマグラフが g と同型となるように ShEx S を定義することができる．このとき，g が p と同型の部分グラフを含むことと，p が S の下で充足可能であることは同値である．　□

提案アルゴリズムは，パターングラフの各ノード u に対して，データグラフにおけるすべてのノード v に対して u と v がマッチし得るかどうかを判定しているため，その時間計算量は最悪で指数関数的になる．しかし，上記の結果から，これを多項式時間で行うのは困難である．しかし，スキーマのサイズは通常データグラフと比較しても非常に小さく，また，提案アルゴリズムにおいては，充足可能なマッチングが 1 つでも発見できればそこで探索を打ち切ることができる．そのため，ほとんどの場合において，本問題は効率よく解くことが可能であると考えられる．

5　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．評価実験では，まず，SP^{2}Bench [8] を用いて生成した RDF データ及び BSBM [9] を用いた．ここで，SP^{2}Bench は SPARQL の包括的なベンチマークソフトであり，DBLP に基づき指定したトリプル数の RDF データを生成する．実験に使用したデータは，トリプル数が 10291(1,087,517 byte) のものとトリプル数が 50168(5,400,376 byte) の 2 つの RDF データである．また，SP^{2}Bench には ShEx スキーマが定義されていないため，文献 [8] を基に著者が作成した (ノード数 11，エッジ数 69)．また，BSBM も同様に SPARQL の包括的なベンチマークソフトであり，データ構造は文献 [9] の p4，Figure1 に基づく．ShEx スキーマが定義されていないため，こちらも上記規則を基に著者が作成した (ノード数 10，エッジ数 71)．作成したそれぞれの ShEx スキーマに基づいて，充足不能なパターングラフを著者が作成した自動生成プログラムにより生成し（各 ShEx スキーマに基づき 50 個ずつ，平均エッジ数 : 5），問合せに要した時間を計測した．生成されるデータは single type semantics のものであるため，single type semantics 状況

表 1　SP2Bench 実行時間

エッジ数	3	4	5	6	7
充足不能性判定（提案手法）	0.00343	0.00357	0.00501	0.00463	0.00469
問合せ実行（10000）	43.2	34.1	34.1	36.0	38.9
問合せ実行（50000）	369	343	396	406	339
時間比率 (提案手法/10000)	0.0000793	0.000105	0.000147	0.000129	0.000120
時間比率 (提案手法/50000)	0.00000927	0.0000104	0.0000126	0.0000114	0.0000138

表 2　BSBM 実行時間

エッジ数	3	4	5	6	7
充足不能性判定（提案手法）	0.0105	0.00737	0.00888	0.0114	0.00762
問合せ実行（10000）	18.1	15.5	19.9	23.5	17.6
問合せ実行（50000）	224	216	236	230	238
時間比率 (提案手法/10000)	0.000574	0.000475	0. 000446	0.000486	0. 000433
時間比率 (提案手法/50000)	0.0000469	0.0000342	0. 0000377	0. 0000497	0.0000321

下での評価実験である．実験環境は，CPU: Intel(R) Core(TM) m3-7Y30 CPU 1.60GHz，RAM: 4.00GB， OS: Windows 10 Home であり，実装に使用した言語は Ruby 2.5.1 である．

上述の RDF データとパターングラフを用いてパターン問合せを行い，また，ShEx スキーマとパターングラフを用いて充足不能性の判定を行った．処理時間の計測には time コマンドを用いており，測定結果の単位は秒である．結果を表 1,2 に示す．値は有効数字 3 桁である．各エッジ数に対して 10 個のパターンを用いて計測し，平均値を記載してある．この結果から，充足可能性判定に要する時間は，パターン問合せに要する時間と比較して非常に小さく，ほぼ無視できる程度であることが分かる．このことから，問合せの実行前に提案アルゴリズムを用いて充足可能性の判定を行うことで，充足不能なパターン問合せに要する処理時間を大幅に短縮できることの可能性が示唆された．

6 む す び

本稿では，ShEx の下でのパターン問合せ充足可能性判定アルゴリズムを提案した．今後の課題として，より多くの ShEx スキーマやパターン問合せを用いて，特に multi type semantics に関する評価実験を行い，提案アルゴリズムの有効性を評価する必要がある．また，ShEx の仕様には否定など，本稿では議論していない機能が含まれている．今後はこのような機能についても考慮していく必要がある．

文　献

[1] World Wide Web Consortium (W3C). “Shape Expressions (ShEx) 2.1 Primer”. http://shex.io/shex-primer/. (accessed 2020-01-08).

[2] Benedikt, M., Fan, W. and Geerts, F.: XPath Satis-fiability in the Presence of DTDs, J. ACM, Vol.55, No.2, Article 8, pp.1-79 (2008)

[3] Geneves, P., Layada, N., Schmitt, A., Gesbert, N. and Knyttl, V.: XML Static Analysis and Type Checking: Online Web Solver, available from http://tyrex.inria.fr/ web-solver/.

[4] Ishihara, Y., Suzuki, N., Hashimoto, K., Shimizu, S. and Fujiwara, T.: XPath Satisfiability with Parent Axes or Qualifiers Is Tractable under Many of Real-World DTDs, Proc. DBPL (2013)

[5] Suzuki, N., Okada, T. and Kwon, Y.: Detecting Unsatisfiable CSS Rules in the Presence of DTDs, Proc. DBPL, pp.18–29 (2019)

[6] 松岡栞, 鈴木伸崇. ”Shape Expression Schema の下でのパターン問合せの充足可能性判定アルゴリズム. webDB Forum2019 論文集, pp49-52, 2019.

[7] Staworko, S., Boneva, I., Labra Gayo, J., Hym, S., Prud’hommeaux, E. and Sorbrig., H.: Complexity and Expressiveness of ShEx for RDF, Proc. ICDT, pp.195–211 (2015)

[8] Schmidt, M., Hornung, T., Lausen, G. and Pinkel, C.: SP2Bench: a SPARQL Performance Benchmark, Proc. ICDE, pp.371–393 (2008)

[9] Christian Bize and Andreas Schultz, The Berlin SPARQL Benchmark, International journal on Semantic Web and information systems 5(2):1-24 April 2009.