有向グラフに対する Shape Expression Schema の妥当性検証及び修正手法の提案

筑波大学
図書館情報メディア研究科

2020年3月

藤永　健

目 次

第1章　序章

グラフデータの構造はスキーマによって表現することができる．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いる．一方，グラフデータについては，標準的なスキーマ言語が提案されていなかったこともあり，明示的なスキーマが付与されたグラフデータは必ずしも多くない．しかし近年，Shape Expression Schema [1] や SHACL [2] など，グラフデータのスキーマ定義に適したスキーマ言語制定の動きが広まっており，今後はグラフデータに対するスキーマの活用も進展するものと期待される．このような状況では，スキーマが先に設計されてからデータが作成されるという通常のプロセスの他に，スキーマを持たない既存のデータに後からスキーマが付与される場合も少なからず生じると思われるが，その際にはより慎重になる必要がある．というのもグラフデータはスキーマの定義を満たしている必要があり，誤りが存在していると，恩恵を得られるどころか混乱を招いていしまう恐れがあるためである．そのため，スキーマに対してグラフが妥当かどうかを確認しなければならず，妥当性が確かめられて初めてスキーマの利点を活かすことができる．もちろんこの妥当性の検証は，スキーマを先に設計してからデータを記述した場合にも行うことが求められる．

データが主語，述語，目的語の文から構成される RDF グラフでは，スキーマが定義されることが比較的多い．もともと RDF は，データの記述方法が非常に制約的であり，RDF Schema[3] というスキーマが定義されていることがほとんどである．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)[1][4] 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．

本論文では，グラフデータに対して ShEx が妥当か否かを検証するアルゴリムを示す．ここでいうグラフデータとは，RDF グラフに限らずラベル付き有向グラフも含める．本アルゴリズムでは，大規模グラフデータに対応するために，グラフデータファイルをシーケンシャルに読み込んで検証を行う．さらに，データの並びによっては検証時間が大きく増加する可能性もあるため，ShEx の型に階層を導入し，効率的に検証を行えるアルゴリズムも提案する．また，どの型の定義も満たさないノードが存在する場合，通常は妥当でないという結果のみを出力して妥当性検証が終了する．しかし，ここで単に妥当でないという結果だけでなく，どのようにノードを修正すればよいかを提示できれば有用である．そこで，グラフデータにデータ構造上の誤りを含む場合，そのノードに最適な型とその修正方法を提案する．評価実験の結果，提案アルゴリムによって妥当性検証が効率的に行える点と，誤

りを含む場合に，機械的に最適な型とその修正方法を導くことができる点を確認した．

グラフデータに対する ShEx の妥当性検証は，Regular Bag Expression に一定の制限を置くことで，多項式時間で検証できることが Stowarko[5] らによって証明されている．また，RDF データについて，SHACL や ShEx の妥当性検証を行っている研究 [6] も存在する．本論文では，Regular Bag Expression に特に制限を置かず，効率のよい妥当性検証アルゴリズムを提案している点や，大規模グラフデータへの対応も考慮している点で違いがある．一方，グラフデータの修正手法の提案については，RDF データの構文エラーを修正する研究 [7] が存在しているが，本論文のようなスキーマレベルの修正ではない．また ShEx 以外のスキーマでは，XML のスキーマである DTD を参照して妥当でない XPath 式を修正する研究 [8] も存在する．著者の知る限り，本論文のように ShEx を対象としたスキーマレベルでのグラフデータを修正するアルゴリズムはまだ提案されていない．

本論文の構成は以下の通りである．2 章では，グラフ，ShEx，そして妥当条件に関する定義を行う．3 章では，妥当性検証アルゴリズムについて説明する．4 章では修正手法について説明する．5 章では評価実験について述べる．6 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

本章では本論文で扱うグラフの概要と，Regular Bag Expression，Shape Expression Schema 及び妥当性条件の定義について述べる．

2.1　グラフの概要

本論文では RDF グラフと同様な形式であるラベル付き有向グラフを用いる．ラベルの集合を Σ，Σ 上のグラフ (以下，単にグラフ) を $G = (V, E)$ とする．ここで，V はノードの集合であり，$E \subseteq V \times \Sigma \times V$ はラベル付き有向辺 (以下エッジ) の集合である．簡単な例として，図 2.1 にグラフ G_1 を示す．

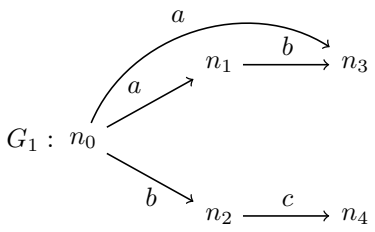


図 2.1: ラベル付き有向グラフ

このとき，

$$\begin{aligned} \Sigma &= \{a, b, c\}, \\ V_1 &= \{n_0, n_1, n_2, n_3, n_4\}, \\ E_1 &= \{(n_0\, a\, n_1), (n_0\, b\, n_2), (n_0\, a\, n_3), (n_1\, b\, n_3), (n_2\, c\, n_4)\} \end{aligned}$$

である．また，本論文ではエッジを出力しているノードを出力ノード，そのエッジに付いているラベルを出力ラベル，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．グラフの各ノードは出力ノードでもあり，入力ノードでもあり得る．ここで，グラフ $G = (V, E)$ におけるノード n から出ているエッジの集合，すなわち出力ラベルとその入力ノードの集合を，

$$\textit{out-lab-node}_G(n) = \{(a, m) \in \Sigma \times V \mid (n, a, m) \in E\}$$

と定義する．例えば，図 2.1 の G_1 において，$\textit{out-lab-node}_{G_1}(n_0) = \{(a, n_1), (b, n_2), (a, n_3)\}$ である．ただし，ノードによっては同じ出力ラベルや近傍入力ノードから成るエッジを複数持つ可能性もある．その場合，重複を許さない集合ではグラフを定義することは出来ない．同様に，ノード n の出力ラベルに関しても，集合では各ラベルの出現回数を示すことは不可能である．実際 G_1 における，n_0 の出力ラベルの集合は $\{a, b\}$ であるが，ラベル a が 2 回

出現していることまでは示されていない．このように集合では表現に限界があるため，シンボルの出現回数を規定する bag を導入する．

シンボルの集合を Δ とする．Δ に対する bag とは，シンボルをその出現回数に対応させる関数 $w : \Delta \to \mathbb{N}$ のことである．集合が $\{a..\}$ で表現されるのに対して，bag は $\{|a..|\}$ で表現される．例えば，n_0 の出力ラベルは集合では $\{a, b\}$ となるが，bag では $\{|a, a, b|\}$ と表現できる．この bag をグラフに適用すると，グラフ $G = (V, E)$ におけるノード n についての出力ラベルは，

$$\mathit{out\text{-}lab}_G(n) = \{|a \mid (n, a, m) \in E|\}$$

と定義できる．

2.2 Regular Bag Expression (RBE)

XML のスキーマである DTD や XML Schema を始めとし，多くのスキーマでは Regular Expression を用いて定義付けをしている．一方，グラフではノード間の順序を考慮しないため，bag の言語を表現できる．そこで，Regular Bag Expression (以下，RBE) という概念を導入する．RBE は，論理和 | と順序を無視した連結 || と $*$ を用いて bag を定義する．Δ に対する RBE は以下の文法 E で定義される．

$$E ::= \epsilon | a | E^* | (E\text{"|"}E) \quad (a \in \Delta)$$

また，RBE E の言語は $L(E)$ と表され，次のように定義される．

$$\begin{aligned}
L(E_1|E_2) &= L(E_1) \cup L(E_2) && : E_1 \text{または} E_2 \text{に一致} \\
L(E_1 \parallel E_2) &= (E_1) \uplus L(E_2) && : E_1 \text{と} E_2 \text{の順序を無視して連結} \\
L(E^*) &= \bigcup_{i \geq 0} L(E)^i && : E \text{が} 0 \text{回以上出現} \\
L(E^?) &= (\epsilon|E) && : E \text{が} 0 \text{回か} 1 \text{回出現} \\
L(E^+) &= (E \parallel E^*) && : E \text{が} 1 \text{回以上出現} \\
L(E^{[n,m]}) &= \bigcup_{n \leq i \leq m} E^i && : E \text{が} n \text{回以上} m \text{回以下出現}
\end{aligned}$$

2.3 Shape Expression Schema (ShEx)

本節では Shape Expression Schema (ShEx) を定義する．ShEx S は，$S = (\Sigma, \Gamma, \delta)$ であり，ここで Σ はラベルの集合，Γ は型の集合，δ は型を定義する関数であり，次を満たす．

$$\delta : \Gamma \to \Sigma \times \Gamma\text{に対する bag の集合}$$

この δ で bag の集合を定義する際に，前節で定義した RBE を用いる．以下に ShEx の例を示す．ただし，$(a, t) \in (\Sigma, \delta)$ を以下 $a :: t$ と表記する．次節で詳しく述べるが，グラフが ShEx に妥当であるとき，グラフの全てのノードに型の定義を満たすような型を割り当てることができる．このとき，ノードに割り当てられる型は 1 つだけという制限はなく，複数でも許されている．ここで，全てのノードに型を 1 つだけ割り当てることができる場合は single-type，2 つ以上割り当てることができるノードが存在している場合は multi-type と呼称する．

例 1. ShEx $S_1 = (\Sigma, \Gamma, \delta)$

$$
\begin{aligned}
\Sigma = &\ \{\text{a, b, c}\},\\
\Gamma = &\ \{t_0, t_1, t_2, t_3\},\\
\delta(t_0) \to &\ \epsilon,\\
\delta(t_1) \to &\ (a :: t_1 | a :: t_2)^+ \parallel b :: t_3,\\
\delta(t_2) \to &\ b :: t_0,\\
\delta(t_1) \to &\ c :: t_0
\end{aligned}
$$

2.4 妥当性条件

グラフ $G = (V, E)$ について，ShEx $S = (\Sigma, \Gamma, \delta)$ の妥当性を考える．そのために，いくつかの関数及び概念を導入する．まず，ノードと型を結びつける関数 λ を以下のように定義する．

$$
\begin{aligned}
\text{single-type} &: V \to \Gamma\\
\text{multi-type} &: V \to 2^{\Gamma}
\end{aligned}
$$

次に，2.1 節で導入した出力ノードと入力ノードの集合 $\mathit{out\text{-}lab\text{-}node}_G(n)$ について，λ を用いて拡張し，出力ラベルと入力ノードの型の bag，

$$\mathit{out\text{-}lab\text{-}type}^{\lambda}_G(n) = \{|a :: \lambda(m) \mid (n, a, m) \in E|\}$$

と定義する．ただし，multi-type の場合は $\lambda(m)$ の要素を複数もつ可能性があるため，$\mathit{out\text{-}lab\text{-}type}^{\lambda}_G(n)$ を平坦化した $\mathit{fl\text{-}out\text{-}lab\text{-}type}^{\lambda}_G(n)$ を導入する．そのために，bag $w(\Sigma \times 2^{\Gamma})$ の平坦化関数 $Flatten$ を，

$$Flatten(w) = \|_{a::T \Subset w} (|_{t \in T}\ a :: t)$$

と定義する．ここで，$a :: T \Subset w$ はシンボル $a :: T$ が $w(a :: T)$ 回出現することを意味している．例えば，$\text{bag}\{|a :: \{t_0, t_1\}, b :: t_2|\}$ では，

$$Flatten(\{|a :: \{t_0, t_1\}, b :: t_2|\}) = (a :: t_0 | a :: t_1) \parallel (b :: t_2)$$

である．この関数 $Flatten$ を用いることで，

$$\mathit{fl\text{-}out\text{-}lab\text{-}type}^{\lambda}_G(n) = L(Flatten(\mathit{out\text{-}lab\text{-}type}^{\lambda}_G(n)))$$

と定義することができる．ここまでで定義した概念を用いて，グラフが ShEx に妥当であるための条件を以下に示す．

single-type

$$\forall n \in V \mid \mathit{out\text{-}lab\text{-}type}^{\lambda}_G(n) \in \delta(\lambda(n))$$

multi-type

$$\forall n \in V, \forall t \in \lambda(n) \mid \mathit{fl\text{-}out\text{-}lab\text{-}type}^{\lambda}_G(n) \cap \delta(t) \neq \emptyset$$

ただし，single-type と multi-type の両方とも $\lambda(n) \neq \emptyset$ でない，すなわちあるノードに対して妥当な型が必ず存在していなければならない．以下の例 2 で single-type，例 3 で multi-type の場合の例をそれぞれ示す．ただし，ShEx は Σ, Γ を省略する．

例 2. singe-type

$$G_2 : n_0 \xrightarrow{a} n_1 \xrightarrow{b} n_2$$

$$\begin{aligned} S_2 : \delta(t_0) &\to a :: t_1 \\ \delta(t_1) &\to b :: t_2 \\ \delta(t_2) &\to \epsilon \end{aligned}$$

このとき，次のように定義される λ を考える．

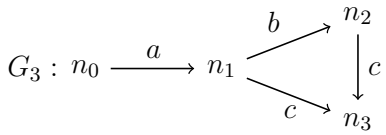
$$\lambda(n_0) = t_0, \quad \lambda(n_1) = t_1, \quad \lambda(n_2) = t_2$$

ここで，n_0 について，

$$\mathit{out\text{-}lab\text{-}type}^{\lambda}_{G_2}(n_0) = \{|a :: t_1|\}$$

となり，$\delta(t_0)$ を満たしているためノード n_0 は型 t_0 に妥当である．同様に他のノードについても考えていくと，上記の条件を満たすため，グラフ G_2 は ShEx S_2 に妥当であるといえる．

例 3. multi-type



$$\begin{aligned} S_3 : \delta(t_0) &\to a :: t_1 \\ \delta(t_1) &\to b :: t_2 \parallel c :: t_3 \\ \delta(t_2) &\to c :: t_3{}^{?} \\ \delta(t_3) &\to \epsilon \end{aligned}$$

このとき，次のように定義される λ を考える．

$$\lambda(n_0) = \{t_0\}, \quad \lambda(n_1) = \{t_1\}, \quad \lambda(n_2) = \{t_2\}, \quad \lambda(n_3) = \{t_2, t_3\}$$

ここで，n_1 について，

$$\mathit{fl\text{-}out\text{-}lab\text{-}type}^{\lambda}_{G_3}(n_1) = \{|b :: t_2, c :: t_2|, |b :: t_2, c :: t_3|\}$$

となり，

$$\mathit{fl\text{-}out\text{-}lab\text{-}type}^{\lambda}_{G_3}(n_1) \cap \delta(t_1) = \{|b :: t_2, c :: t_3|\} \neq \emptyset$$

であるため，ノード n_1 は型 t_1 に妥当である．同様に他のノードについても考えていくと，上記の条件を満たすため，グラフ G_3 は ShEx S_3 に妥当であるといえる．

第3章　妥当性検証

本章では，妥当性検証の提案アルゴリズムについて述べる．提案アルゴリズムは，single-type と multi-type の両方に対応している．

3.1　入力グラフデータ

本論文で扱うグラフデータは，RDF データの形式である N-triples 形式に準じた形でファイルに格納されていることを前提とする．すなわち，グラフデータの各行はエッジであり，出力ノード，出力ラベル，入力ノードの順で構成されている．以下にグラフデータの例を示す．

出力ノード	出力ラベル	入力ノード
$n1$	a	$n2$
$n1$	a	$n3$
$n1$	b	$n4$
$n2$	c	$n5$
$n2$	d	$n6$
$n3$	c	$n5$
$n3$	d	$n7$
$n4$	d	$n6$
$n4$	d	$n8$

妥当性検証を行っていく際は，グラフデータが大規模である場合を考慮し，グラフデータ全体をメモリ上に置くのではなく，出力ノード毎に出力エッジを読み込み検証を行っていく．例では，最初に読み込むのは出力ノードが $n1$ である 3 行目までである．ただし，グラフデータによっては同一出力ノード毎にエッジが並んでいないため，あらかじめ外部ソートで並び替えを行っておく．というのも，出力ノードのエッジを全て読み込むことができないと，妥当条件の一部である $\mathit{fl\text{-}out\text{-}lab\text{-}type}^{\lambda}_{G}(n) \cap \delta(t) \neq \emptyset$ かどうかを正しく検証できないためである．

3.2　λ の初期設定

入力グラフデータに対して λ の初期定義を行う．既存手法 [5] では λ の初期値として各ノードに ShEx の全ての型を割り当てているが，これでは $\mathit{fl\text{-}out\text{-}lab\text{-}type}^{\lambda}_{G}(n)$ のサイズが膨大になり，妥当性検証の時間およびメモリ消費量も大きく増加してしまう．そこで，ノードの出力ラベル集合 $\mathit{out\text{-}lab}_G(n)$ と型のラベル集合 $\Sigma(t)$ に注目する．ここで，$\Sigma(t)$ は t に出現するラベルの集合である．型の bag 及び RBE によっては出力ラベルが 0 回の出現もあり得るため，$\mathit{out\text{-}lab}_G(n)$ が $\Sigma(t)$ の部分集合である場合，$\lambda(n)$ に型を追加する．しかし，妥

当でないグラフデータの場合 $\lambda(n)$ が妥当性検証前に空となる可能性がある．その場合，妥当性検証の際に判別しやすくするため，エラーであることを示す型を $\lambda(n)$ に付与する．また，出力エッジをもたないノードは，$\delta(t) = \epsilon$ である型を付与し，リテラルノードには対応するリテラル型を付与する．ただし，これらのノードは簡単に区別できるため，$\lambda(n)$ としてノード毎には定義しない．これらのことを踏まえて，λ の初期定義を行うアルゴリズム initialization を以下に示す．

Algorithm 1 initialization

Input: $G = (V, E)$, $S = (\Sigma, \Gamma, \delta)$
Output: λ

```
1: λ ← ∅
2: for each n ∈ V do
3:     for each t ∈ Γ do
4:         if out-lab_G(n) ⊆ Σ(t) then
5:             λ(n) ← λ(n) ∪ {t}
6:         end if
7:     end for
8:     if λ(n) = ∅ then
9:         λ(n) ← {t_error}
10:    end if
11: end for
```

次に，アルゴリズムの各行の処理を説明する．2〜11 行目までの繰り返しは，グラフデータの各出力ノードに対して 3 行目以降の処理を行う．その 3〜7 行目では，ShEx の各型について，出力ラベル集合が型のラベル集合の部分集合である場合，$\lambda(n)$ にその型を追加している．最後の 8〜10 行目では，$\lambda(n)$ が空である場合，妥当でないことを意味する型 t_{error} を追加する．

3.3 妥当性検証アルゴリズム

ここまでの節で，グラフデータを格納するファイル (以下，グラフファイル) と λ の初期定義を準備することができた．本節では，これらと ShEx を用いて妥当性検証を行うアルゴリズムについて説明する．まず，以下に妥当性検証アルゴリズム validation を示す．

1 行目では，λ が変化していないか判別する目的で，2 行目以降の繰り返し内の処理を行う前の λ を格納する λ_{old} を用意している．2〜18 行目までの繰り返しでは，λ が変化している間 3 行目以降の処理を行う．4〜17 行目までは，グラフファイルが EOF に到達するまで繰り返しを行っている．5 行目では，ファイルを同一出力ノード間読み込み，その出力ノードの出力ラベルと入力ノードの bag $\mathit{out\text{-}lab\text{-}node}_G(n)$ を取得している．6 行目で，取得した $\mathit{out\text{-}lab\text{-}node}_G(n)$ に対して，λ を参照し $\mathit{out\text{-}lab\text{-}type}^{\lambda}_G(n)$ を得た後，平坦化することで $\mathit{fl\text{-}out\text{-}lab\text{-}type}^{\lambda}_G(n)$ を取得している．8〜12 行目の繰り返しでは，$\lambda(n)$ の各型について，$\mathit{fl\text{-}out\text{-}lab\text{-}type}^{\lambda}_G(n) \cap \delta(t) \neq \emptyset$ である場合，集合 F にその型を付与する処理を行っている．なお，$\mathit{fl\text{-}out\text{-}lab\text{-}type}^{\lambda}_G(n) \cap \delta(t) \neq \emptyset$ か否かの判定は，後述の Refine により行う．13〜15 行目では，$F = \emptyset$ の場合，すなわち $\lambda(n)$ の全ての型が $\mathit{fl\text{-}out\text{-}lab\text{-}type}^{\lambda}_G(n) \cap \delta(t) = \emptyset$ である場合，妥当でないという結果になるため，妥当性判定 false を返してアルゴリズムを途中で終了する．16 行目では，$\lambda(n)$ と F の共通集合を新しい $\lambda(n)$ として更新する．このとき共通

集合の要素は全て，現時点でノード n に妥当である型である．最後に，妥当条件を満たしている λ 及び，妥当性判定 true を返してアルゴリズムを終了する．

Algorithm 2 validation

Input: グラフファイル $(G = (V, E))$, ShEx $S = (\Sigma, \Gamma, \delta)$, λ
Output: 妥当性判定 (true or false)，λ

```
1: λ_old ← ∅
2: while λ ≠ λ_old do
3:     λ_old ← λ
4:     while グラフファイルが EOF に到達していない do
5:         同一出力ノード n ∈ V である行まで読み込み，out-lab-node_G(n) を取得する
6:         λ を参照し，fl-out-lab-type^λ_G(n) を取得する．
7:         F ← ∅
8:         for each t ∈ λ(n) do
9:             if fl-out-lab-type^λ_G(n) ∩ δ(t) ≠ ∅ then
10:                F ← F ∪ {t}
11:            end if
12:        end for
13:        if F = ∅ then
14:            return false
15:        end if
16:        λ(n) ← λ(n) ∩ F
17:    end while
18: end while
19: return true, λ
```

ここで，$\mathit{fl\text{-}out\text{-}lab\text{-}type}^{\lambda}_{G}(n) \cap \delta(t) \neq \emptyset$ であるか否かを判定するアルゴリズム refine を示す．ただし，$\delta(t)$ が一般の RBE である場合，refine を効率良く行うことは困難である [5]．そのため，以下では $\delta(t)$ が以下の形に限定されると仮定する．

$$\delta(t) = E_1 \parallel E_2 \parallel ... \parallel E_n$$

ここで，E_i は $a :: t$, $a :: t^+$, $a :: t^*$, $a :: t^{[n,m]}$, $(a_{i_1} :: t_{i_1} \mid a_{i_2} :: t_{i_2} \mid ... \mid a_{i_k} :: t_{i_k})$ のいずれかである．また，$\delta(t)$ の要素集合を $E(t) = \{E_1, E_2, ..., E_n\}$ と定義する．以下に示す Refine は，E_i が選言 | を含まない場合のものである (| を含む場合の処理は後述する)．また，E の出現回数の最小値及び最大値を示す関数について，それぞれ $min(E), max(E)$ と定義する．例えば，$E_1 = a :: t^+$ のとき，$min(E_1) = 1$，$max(E_1) = \infty$ である．

一方 $\mathit{fl\text{-}out\text{-}lab\text{-}type}^{\lambda}_{G}(n)$ の要素は bag $\mathit{out\text{-}lab\text{-}node}_G(n)$ である．この bag のシンボルは $a :: t$ のようなラベルと型の組合わせである．2 章で Δ をシンボルの集合と定義したが，bag のシンボルの集合を意味する関数を d とすると，$d(\mathit{out\text{-}lab\text{-}node}_G(n)) = \Delta$ と表現できる．例えば，$d(|a :: t_1, a :: t_1, b :: t_2|) = \{a :: t_1, b :: t_2\}$ である．

Algorithm 3 refine

Input: *fl-out-lab-type*$^{\lambda}_{G}(n)$, $\delta(t)$

```
1: for each g ∈ fl-out-lab-type^λ_G(n) do
2:     Sym ← d(g)
3:     error ← flase
4:     for each E ∈ E(t) do
5:         n ← 0
6:         for each lab-type ∈ Sym do
7:             if lab-type = E then
8:                 n ← n + w(lab-type)
9:                 Sym ← Sym \ {lab-type}
10:                break
11:            end if
12:        end for
13:        if n < min(E) or n > max(E) then
14:            erorr ← true
15:            break
16:        end if
17:    end for
18:    if error = true then
19:        next
20:    end if
21:    if Sym = ∅ then
22:        return true
23:    end if
24: end for
25: return false
```

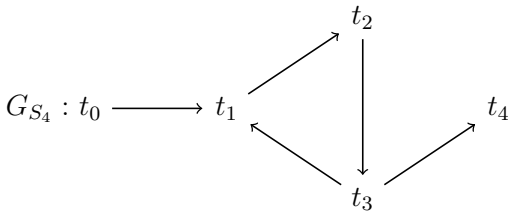
各行の処理を説明する．1〜25 行目まで，*fl-out-lab-type*$^{\lambda}_{G}(n)$ の要素 g についてそれぞれ処理を行っていく．2 行目で，g に関するシンボルの集合を Sym に格納する．4〜18 行目まで，$\delta(t)$ の各要素 E について処理を行っていく．Algorithm 3 では，E が | を含まない場合を記述している．E が | を含む場合の処理は後述する．5 行目では，E と一致した g のシンボルの出現回数を格納する，n を用意する．6〜12 行目では，Sym の要素 *lab-type* (g のシンボル) それぞれに対して 7 行目の条件を見ていく．*lab-type* が E と一致している場合，その出現回数を n に加える．一度一致した *lab-type* は，他の E と一致する可能性がないため，Sym から削除して 6 行目からの繰り返しを終了する．13〜16 行目では，E が回数の定義を満たしているかを確認している．満たしていない場合，繰り返しを終了して次の g に移る．そのために，$error$ に true を代入することで，18 行目で条件を満たし，以降の処理を行わず次の g に移ることができる．21〜23 行目では，Sym が空である場合，つまり出力ノード n が t に妥当であるとき，true を返しアルゴリズムを終了する．最後に，アルゴリズムが途中で終了せず，全ての繰り返しを終えた場合，false を返す．つまり，*fl-out-lab-type*$^{\lambda}_{G}(n)$ の要素のうち，$\delta(t)$ を満たしているものは存在しない．

最後に，4 行目で E が | を含む場合の処理について述べる．もし 4 行目で $E = (a_{i_1} :: t_{i_1} \mid a_{i_2} :: t_{i_2} \mid ... \mid a_{i_k} :: t_{i_k})$ の場合，k 個の要素それぞれに対して 5 行目以下の処理を行い，そのうち 1 つが $error$ = false であれば E は成立するとし，$\delta(t)$ の次の要素に移る．

3.4 ShEx の階層化と妥当性検証アルゴリズム

前節の妥当性検証アルゴリズムでは，*fl-out-lab-type*$^{\lambda}_{G}(n)$ の要素が多いほど処理の時間が増加する．またその生成方法である，*out-lab-type*$^{\lambda}_{G}(n)$ の平坦化にかかる時間も当然大きく増加する．これは入力ノードの λ によって結び付けられる型が多いほど顕著に現れる．initialization アルゴリズムでは，型のラベル状態によって，ほぼ全型が $\lambda(n)$ に付与されるおそれもあるが，実際のグラフファイルでは，このノードがそのまま入力ノードで現れる可能性が十分にある．そこで，型同士の依存関係に着目し，ShEx の階層化を行う．先に入力ノード型が確定していれば，その型に依存している，すなわちエッジをもつ型が付与されているノードの検証時，*fl-out-lab-type*$^{\lambda}_{G}(n)$ の要素を最小に抑えることができる．階層を求めるため，最初に ShEx からスキーマグラフ G_S を抽出する．各型において，リテラル型以外の定義部分からグラフを作成する．例えば，次の ShEx S_4 から抽出されるグラフは以下の通りである．

$$
\begin{aligned}
S_4 : \delta(t_0) &\to a :: t_1 \\
\delta(t_1) &\to b :: t_2 \\
\delta(t_2) &\to c :: t_3 \\
\delta(t_3) &\to a :: t_1 \parallel d :: t_4 \\
\delta(t_4) &\to \epsilon
\end{aligned}
$$



このグラフから階層，すなわち依存関係を得るには，強連結成分分解を行う．この例では，以下の 3 階層を得ることができる．ここで，$\{t_4\}$ が最も下の階層であり，$\{t_0\}$ が最も上の階層である．

$$R = [\{t_4\}, \{t_1, t_2, t_3\}, \{t_0\}]$$

そして，この得られた階層毎に妥当性検証を行っていく．具体的には，$\lambda(n)$ に該当する階層の要素を含んでいる場合に限り，refine を行うことで，入力ノードの型がほぼ確定された状態で検証することができる．次の階層に移った際，再度ファイルを先頭から読み込んでいく．これを全ての階層について行う．以下に，階層化妥当性検証アルゴリズム hierarchical validation を示す．

前節から追加した部分について簡単に説明する．3 行目からは各階層毎に妥当性検証の処理を行うことを意味している．ここで，R から r を取り出す順番は任意ではなく，階層の下から上に向けて取り出す必要がある．7〜9 行目では，現在読み込んだ出力ノードが該当する型かどうかを確認している．そうでない場合は，次の出力ノードを読み込む．

Algorithm 4 hierarchical validation

Input: グラフファイル ($G = (V, E)$), ShEx $S = (\Sigma, \Gamma, \delta)$, λ, R

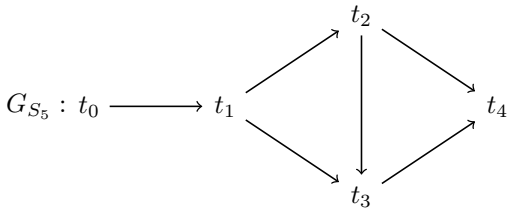
Output: 妥当性判定 (true or false)，λ

$\lambda_{old} \leftarrow \emptyset$
while $\lambda \neq \lambda_{old}$ **do**
  for each $r \in R$ **do**
    $\lambda_{old} \leftarrow \lambda$
    while グラフファイルが EOF に到達していない **do**
      同一出力ノード $n \in V$ である行まで読み込み，*out-lab-node*$_G(n)$ を取得する
      if $\lambda(n) \cap \{r\} = \emptyset$ **then**
        next
      end if
      λ を参照し，*fl-out-lab-type*$^{\lambda}_G(n)$ を取得する．
      $F \leftarrow \emptyset$
      for each $t \in \lambda(n)$ **do**
        if *fl-out-lab-type*$^{\lambda}_G(n) \cap \delta(t) \neq \emptyset$ **then**
          $F \leftarrow F \cup \{t\}$
        end if
      end for
      if $F = \emptyset$ **then**
        return false
      end if
      $\lambda(n) \leftarrow \lambda(n) \cap F$
    end while
  end for
end while
return true, λ

3.5 アルゴリズムの動作例

この節では，これまでのアルゴリズム等をふまえて，階層化を用いた妥当性検証の簡単な動作例を示す．始めに入力する ShEx S_5 及びそのスキーマグラフ G_{S_5} を以下に示す．

$$
\begin{array}{llll}
S_5 & \delta(t_0) & \rightarrow & a :: {t_1}^{+} \\
 & \delta(t_1) & \rightarrow & b :: {t_2}^{?} \parallel c :: t_3 \\
 & \delta(t_2) & \rightarrow & c :: {t_3}^{[2,5]} \parallel d :: t_4 \\
 & \delta(t_3) & \rightarrow & d :: t_4 \\
 & \delta(t_4) & \rightarrow & \epsilon
\end{array}
$$



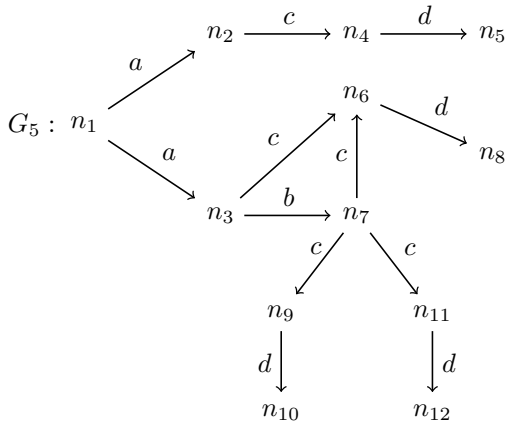
ここで，S_5 における各型のラベル集合は，

$$\Sigma(t_0) = \{a\}, \Sigma(t_1) = \{b, c\}, \Sigma(t_2) = \{c, d\}, \Sigma(t_3) = \{d\}, \Sigma(t_4) = \emptyset$$

である．また階層は，

$$R = [\{t_3\}, \{t_2\}, \{t_1\}, \{t_0\}]$$

の 4 つの階層を得ることができる．ここで，$\delta(t_4)$ は出力エッジをもたない型であるため，階層には含めない．次に，入力するグラフ G_5 及びそのグラフファイルを以下に示す．



n_1	a	n_2
n_1	a	n_3
n_{11}	d	n_{12}
n_2	c	n_4
n_3	b	n_7
n_3	c	n_6
n_4	d	n_5
n_6	d	n_8
n_7	c	n_{11}
n_7	c	n_6
n_7	c	n_9
n_9	d	n_{10}

最初に，initialization アルゴリズムを用いて λ の初期定義をする．ファイルの先頭から同一出力ノードの行を読み込む．この場合，初めに読み込むのは n_1 を出力ノードとする 2 行である．その結果，$\mathit{out\text{-}lab}_{G_5}(n_1) = \{a\}$ とわかるが，各型のラベル集合と比較して，部分集合となるのは，t_0 である．この結果から $\lambda(n_1) = \{t_1\}$ を初期値として得る．この処理をファイルの EOF まで繰り返し，最終的に出力される $\lambda(n)$ は以下の通りである．ただし，n_5, n_8, n_{10}, n_{12} は t_4 に妥当なのは明らかであるため，特に明記しない．

$$\begin{aligned}
\lambda(n_1) &= \{t_1\}, \\
\lambda(n_4), \lambda(n_6), \lambda(n_9), \lambda(n_{11}) &= \{t_2, t_3\}, \\
\lambda(n_2), \lambda(n_7) &= \{t_1, t_2\}, \\
\lambda(n_3) &= \{t_1\}
\end{aligned}$$

この λ に加えて，R と ShEx，グラフファイルを入力し，妥当性検証を行っていく．グラフの先頭から読み込んでいくが，最も依存関係が低い階層の型は t_3 であり，$\lambda(n)$ に含んむ出力ノードの行を読み込む．よって n_1 の次の n_{11} の行から，

$$\mathit{fl\text{-}out\text{-}lab\text{-}type}^{\lambda}_{G}(n_{11}) = \{|d :: t_4|\}$$

を得る．次に，$\lambda(n_{11})$ の要素 t_2, t_3 それぞれについて refine を行う．t_2 では $E = c :: t_3$ と $d :: t_4$ が一致しないため，n が 0 となる。これは，$min(E) = 2$ よりも小さいため，そのままアルゴリズムが最後まで実行されて false が返される．一方で，t_3 では，$E = d :: t_4$ と $d :: t_4$ が一致し，n が 1 となる．さらに，$min(E) = 1$，$max(E) = 1$ より出現回数の条件も満たす．よって $d :: t_4$ が格納された Sym も空となるため，true が返される．この refine の結果から，

$$\lambda(n_{11}) = \{t_3\}$$

と更新される．その後，ファイルを続きから読み込んでいき，型 t_3 を $\lambda(n)$ に含む出力ノードについて，同様に refine を行っていく．ファイルの EOF に到達した時点での λ は以下の

通りである．

$$\begin{aligned} \lambda(n_1) &= \{t_1\}, \\ \lambda(n_4), \lambda(n_6), \lambda(n_9), \lambda(n_{11}) &= \{t_3\}, \\ \lambda(n_2), \lambda(n_7) &= \{t_1, t_2\}, \\ \lambda(n_3) &= \{t_1\} \end{aligned}$$

再度ファイルを先頭から読み込んいくが，同時に型の階層も 1 つ上に移行し，今度は t_2 を $\lambda(n)$ に含むノードが対象となる．したがって，今回最初に読み込むのは n_2 の行となり，

$$\mathit{flt\text{-}out\text{-}lab\text{-}type}^{\lambda}_{G}(n_2) = \{|c :: t_3|\}$$

を得る．まず，t_1 について refine を行う．$E_1 = b :: t_2$ と一致しないが，$min(E_1) = 0$ であるため，繰り返しを終了せず次の E_2 に移る．$E_2 = c :: t_3$ と一致し，回数の条件も満たため，Sym も空となり true が返される．一方で，t_2 では，$c :: t_3$ の最低出現回数を満たさないため false が返される．以上より，

$$\lambda(n_2) = \{t_1\}$$

と更新される．残りのノードや階層の型についてみていくと，最終的に妥当性判定結果として true が返され，以下を出力する．

$$\begin{aligned} \lambda(n_1) &= \{t_1\}, \\ \lambda(n_4), \lambda(n_6), \lambda(n_9), \lambda(n_{11}) &= \{t_3\}, \\ \lambda(n_2), \lambda(n_3) &= \{t_1\}, \\ \lambda(n_7) &= \{t_2\} \end{aligned}$$

したがってグラフ G_5 は ShEx S_5 に妥当である．

もし，通常の妥当性検証であれば，ファイルの先頭のノードから refine を行う．すると，最初に読み込む n_1 では，

$$\mathit{flt\text{-}out\text{-}lab\text{-}type}^{\lambda}_{G}(n_1 1) = \{|a :: t_1, a :: t_1|, |a :: t_1, a :: t_2|\}$$

となり，それぞれの要素について $\delta(t)$ と比較していかなければならない．n_4 ではこの要素数が 4 つとなり，グラフによっては膨大な数になる恐れもある．先ほどのように階層毎に検証を行うことで，この要素を最小数に抑えることが可能なのである．また，スキーマのサイズ数はグラフデータの大きさに比べて非常に小さいため，グラフデータをノード間の依存関係で並び替えるよりも，効率がよいと考えられる．

第4章　修正手法の提案

前章の妥当性検証において，妥当でないグラフデータの場合検証が途中で終了する．その際，妥当でない出力ノードのエッジを修正するなどした後，再度はじめから妥当性検証をしなければならない．そこで，妥当性検証が途中で終了した場合，その出力ノードに対して，データ構造的に最適だと思われる型を推定し，その型に妥当となるような修正方法を導く．ここでの修正とは，データ構造的な修正であり，出力ラベルや入力ノードの型を別の型に変更することや，エッジの追加及び削除が該当する．このアルゴリズムによって推定された型を $\lambda(n)$ に付与し，妥当性検証を再開するという用途を想定している．妥当性検証が終了して，グラフデータが ShEx に妥当となった後，修正する出力ノードの $\mathit{out\text{-}lab\text{-}node}_G(n)$ 及び，推定した型とその修正方法を示す．また，multi-type の場合，本論文で提案する手法では，全ての型の組み合わせを考える必要があり，膨大な時間がかかる可能性もある．そのため，本論文が対象としているのは single-type である．single-type の場合，最大でも ShEx の型の数だけ考慮すればよい．

関連研究として，Regular Expression に対してマッチしない文字列を，マッチするように修正する方法を求める手法 [9] が存在する．この手法では，

Input: Regular Expression r, 文字列 s

Output: スタートからゴールまでのどの経路も s を r を満たすように修正した文字列を表し，経路の重みが修正コストと一致するような有向グラフ

という入力と出力からなる．本論文では，この手法を利用して妥当でなかった出力ノードの $\mathit{out\text{-}lab\text{-}type}^{\lambda}_G(n)$ を型 $\delta(t)$ に修正する方法を求める．ただし，Regular Bag Expression を Regular Expression に，$\mathit{out\text{-}lab\text{-}type}^{\lambda}_G(n)$ を文字列に対応付ける必要がある．そこで，本論文での入力と出力，及び入力を既存手法に対応付ける方法を以下に示す．

Input: $\delta(t)$(Regular Bag Expression), $\mathit{out\text{-}lab\text{-}type}^{\lambda}_G(n)$

Output: スタートからゴールに到達すると，どの経路でも $\delta(t)$ を満たす $\mathit{out\text{-}lab\text{-}type}^{\lambda}_G(n)$ となる有向グラフ

1. $\delta(t)$ に順序を付与し，Regular Expression にする．順序は特に制約を設けない．
2. 得られた Regular Expression の順序に合わせて， $\mathit{out\text{-}lab\text{-}type}^{\lambda}_G(n)$ を並び替えて文字列にする．$a :: t$ のような，出力ラベルと型の組が 1 つの文字である．
3. 上記 1 と 2 で得られた Regular Expression と文字列から，[9] の手法を用いて有向グラフを生成する．

例えば，$\delta(t) \to (a :: t_1|b :: t_1) \parallel a :: t_2{}^{+}$ のとき，順序を付与して $(a :: t_1|b :: t_1)a :: t_2{}^{+}$ という Regular Expression にする．一方で，$\mathit{out\text{-}lab\text{-}type}^{\lambda}_G(n) = |b :: t_1, a :: t_1|$ であれば，文字列 $(a :: t_1)(b :: t_1)$ に置き換える．このとき生成されるグラフを以下の図 4.1 に示す．

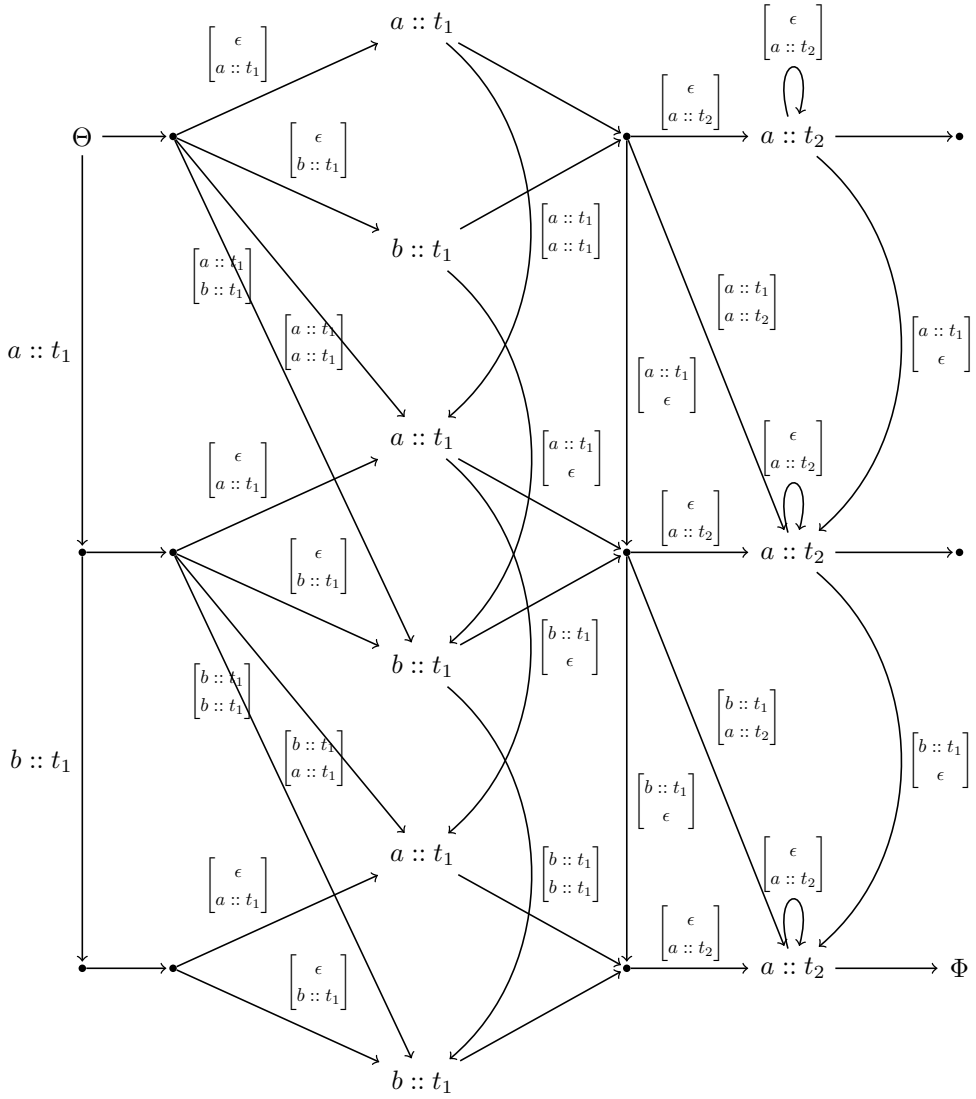


図 4.1: 修正方法発見グラフ (no cost version)

生成されたグラフについて説明する．Θ がスタート，Φ がゴールを表す．横方向のエッジはラベル型，すなわち新たなエッジの追加を意味している．$\begin{bmatrix} \epsilon \\ a :: t_1 \end{bmatrix}$ では出力ラベルが a で，型が t_1 の入力ノードであるエッジを追加する．縦方向のエッジは削除を意味する．例えば，$\begin{bmatrix} a :: t_1 \\ \epsilon \end{bmatrix}$ では，入力グラフ $(\mathit{out\text{-}lab\text{-}type}^{\lambda}_{G}(n))$ の $a :: t_1$ であるエッジを削除する．ただし，縦方向のエッジは場合によって置換にもなり得る．斜め方向のエッジはラベルまたは入力ノードの型，あるいはその両方の変更を意味する．$\begin{bmatrix} a :: t_1 \\ a :: t_2 \end{bmatrix}$ では，入力ノードの型 t_1 を型 t_2 に変更する．この斜め方向のエッジでも置換となる可能性がある．

さて，生成したグラフから最適な修正方法を導くには，エッジにコストを付与し，最小コストかつ最短となる経路を求める必要がある．まず，エッジに付与するコストを以下のように設定する．

追加	1.0
削除	2.0
ラベルのみの変更	1.5
型のみの変更	2.5
ラベルと型の両方を変更	3.0
置換	0.0

ここでの「置換」はグラフに対して特に修正を加えない (同一ラベル，同一型への修正) 処理のため，コストが 0.0 となる．次に「追加」を 1.0,「削除」を 2.0，これらの組み合わせである「ラベルと型の両方を変更」は，合計値である 3.0 と設定した．ここで，削除よりも追加のコストを低く設定したのは，削除の方が低いと E が少ない型へ優先的に修正してしまうのに加えて，既にあるデータを消すよりも追加して妥当となるならばその方が有効的だと考えたからである．また「ラベルのみの変更」は，最も修正対象であるグラフに対してデータ構造の変更を伴わないため，1.5 と低く設定した．一方で「型のみの変更」は，入力ノードの型を変更するため，変更する入力ノードが出力ノードでもあった場合，そのデータ構造にも大きな影響をもたらす可能性が高い．よって，入力ノードの型を変更するよりも，エッジを変更したい型に妥当なノードに張り替える，あるいはその型に妥当なノードを追加する方がよいと考えている．以上の理由から，ラベルと型の両方の変更よりは低いコストである 2.5 と設定した．

先ほど生成したグラフのエッジに対して，ラベルを基にコストを付与する．そして，得られた重み付きグラフ上でスタートからゴールまでの最短経路を求め，そのコストを算出する．図 4.1 の場合，その経路は図 4.2 の通りである．

得られた経路から，$a :: t_1$ はそのままで ($a :: t_1$ に置換)，次の $b :: t_1$ をラベルと型の両方とも変更して $a :: t_2$ に修正する方法が提案される．実際には ShEx の各型と妥当でない部分グラフについてそれぞれ修正グラフを作成し，得られた修正コストのうち，最小となる型とその経路を提案する．ただし，最小となる型が複数存在する場合，全て提案する．

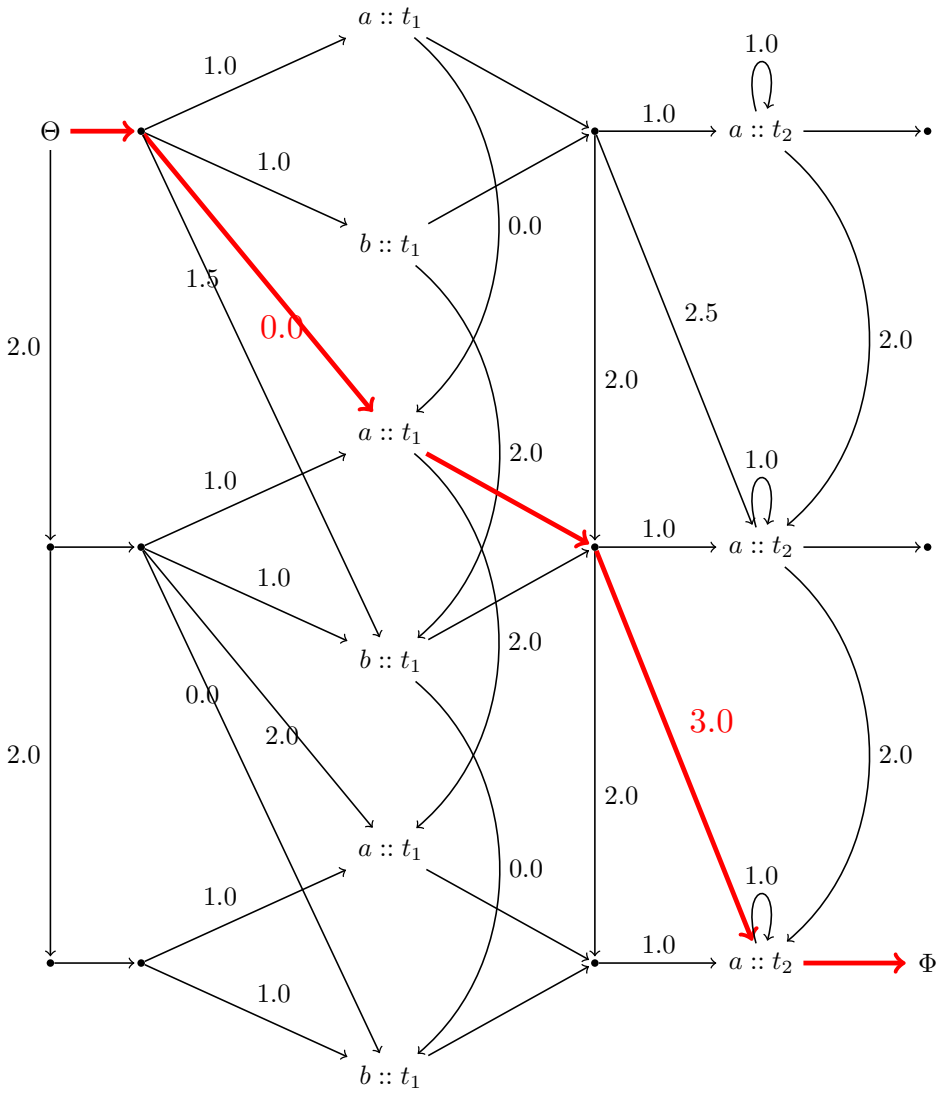


図 4.2: 修正経路

第5章　評価実験

本章では妥当性検証アルゴリズム及び修正手法の評価実験について述べる．実装には，Ruby 2.6.5 を用いた．また全ての実験を，Intel Core i3-6100U @ 2.30GHz 2.30GHz CPU，8GB RAM，Windows 10 Home 64bit の環境で行った．

5.1　妥当性検証の実験

妥当性検証アルゴリズムを評価するために，以下の 2 つのデータセットを用いた．

SP^{2}Bench[10]　コンピュータ科学に関する書誌情報データベースである DBLP に基づき，RDF データを生成するツールである．生成された RDF データは， Nortaion 3 形式で，Turtle 形式の表記を使用して，N-triple 形式にした記述方法である．SP^{2}Bench によって生成されるグラフの全体像は図 5.1 の通りである．本実験では，ShEx を用意するため，RDF Schema の型である RDF type ノードについては特に妥当性検証をしない．他のノードも，RDF type ノードに type ラベルで出力しているが，同様の理由でそのエッジは検証をしない．また，rdf:_1 や rdf:_2 というラベルが存在するが，この数字部分は常に変動する．しかし，数字自体は重要ではなく，区別をつける必要がないため， rdf:_ という 1 つのラベルとして扱う．

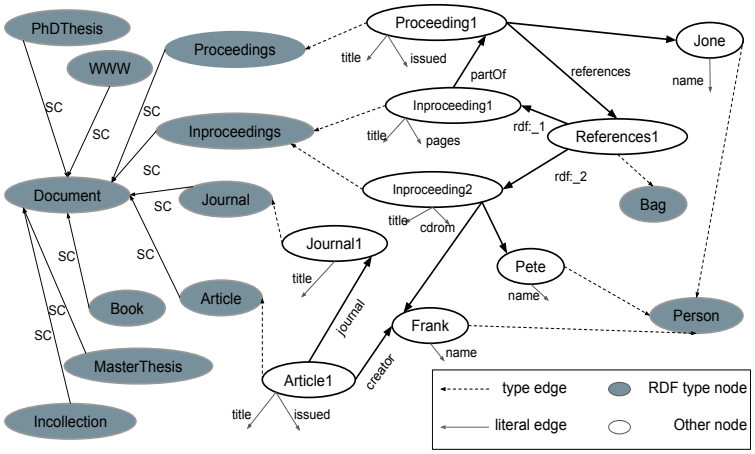


図 5.1: SP^{2}Bench で生成されるグラフの概観

このグラフの全体像及びリテラルノードへの出力ラベルから以下の ShEx を生成した．

BSBM[11]　電子商取引の形態に基づき，RDF データを生成するツールである．生成される RDF データは Turtle または N-triple 形式となっている．本実験では，Tutrle 形式のデータを生成し，SP^{2}Bench と同様の記述形式に変換して使用する．生成されるグラフの全体像は図 5.2 の通りである．SP^{2}Bench と同様に，RDF Type ノード及び type エッジは検

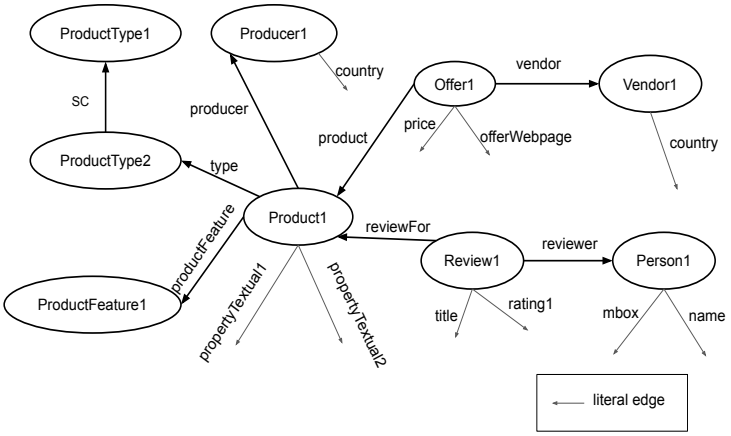


図 5.2: BSBM で生成されるグラフの概観

証を行わない．ただし，Product クラスから ProductType へのエッジについては，データ構造上重要な意味をもつため，ShEx で定義して検証を行う．また，productFeature ラベルのエッジの本数は生成するグラフデータのサイズによって変動するため，1 回以上とする．さらに，propertyTexutual1 や propertyTextual2 ラベルの末尾に付く数字も上限がないため，数字を削除して同一のラベルである propertyTextual とする．

次に，それぞれのデータセットによって生成するエッジ数 (トリプル数)，λ に格納される出力ノード数，データサイズを以下に示す．

表 5.1: SP^{2}Bench のデータサイズ

$\|E\|$	$\|V*\|$	size(MB)
95,768	18,584	10.1
922,241	173,154	100.9
9,053,244	1,571,136	1,008.9
18,135,502	3,152,062	2,017.8

表 5.2: BSBM のデータサイズ

$\|E\|$	$\|V*\|$	size(MB)
374,911	36,433	33.9
1,809,874	168,555	162.1
8,873,389	808,154	794.8
17,686,178	1,601,677	1,584.0

ここで，SP^{2}Bench 及び BSBM の ShEx の型を示す．ただし各ラベルの接頭辞 (foaf や dc) は略記する．また，SP^{2}Bench の ShEx は multi type，BSBM の ShEx は single type となっている．

SP^{2}Bench の型 (multi type)

$$
\begin{aligned}
< \mathrm{String}(l_1) > &\rightarrow \epsilon \\
< \mathrm{Integer}(l_2) > &\rightarrow \epsilon \\
< \mathrm{Iri}(l_3) > &\rightarrow \epsilon \\
< \mathrm{Person}(t_1) > &\rightarrow \mathrm{name} :: l_1 \\
< \mathrm{Journal}(t_2) > &\rightarrow \mathrm{title} :: l_1 \parallel \mathrm{issued} :: l_2 \parallel \mathrm{editor} :: {l_3}^* \parallel \mathrm{editor} :: {t_1}^* \\
&\quad \parallel \mathrm{number} :: {l_2}^? \parallel \mathrm{volume} :: {l_2}^? \\
< \mathrm{Article}(t_3) > &\rightarrow \mathrm{abstract} :: {l_1}^? \parallel \mathrm{booktitle} :: {l_1}^? \parallel \mathrm{cdrom} :: {l_1}? \parallel \mathrm{publisher} :: {l_1}^* \parallel \mathrm{title} :: l_1 \\
&\quad \parallel (\mathrm{creator} :: {l_3}^* \mid \mathit{mboxcreator} :: {t_1}^*) \parallel \mathrm{references} :: {t_4}^* \parallel \mathrm{homepage} :: {l_1}^? \\
&\quad \parallel \mathrm{seeAlso} :: {l_1}^* \parallel \mathrm{journal} :: t_2 \parallel \mathrm{month} :: {l_2}^? \parallel \mathrm{note} :: {l_1}^? \parallel \mathrm{pages} :: {l_2}^? \\
< \mathrm{Reference}(t_4) > &\rightarrow (\mathrm{rdf{:}\_} :: l_3 \mid \mathrm{rdf{:}\_} :: t_3 \mid \mathrm{rdf{:}\_} :: t_7)^+ \\
< \mathrm{Incollection}(t_5) > &\rightarrow \mathrm{booktitle} :: l_1 \parallel \mathrm{cdrom} :: {l_1}^? \parallel \mathrm{creator} :: {t_1}^+ \parallel \mathrm{publisher} :: {l_1}^* \parallel \mathrm{title} :: l_1 \\
&\quad \parallel \mathrm{issued} :: l_2 \parallel \mathrm{references} :: {t_4}^* \parallel \mathrm{homepage} :: {l_1}^? \parallel \mathrm{seeAlso} :: {l_1}^* \\
&\quad \parallel \mathrm{chapter} :: {l_2}^? \parallel \mathrm{isbn} :: {l_1}^? \parallel \mathrm{pages} :: {l_2}^? \\
< \mathrm{Proceeding}(t_6) > &\rightarrow \mathrm{booktitle} :: {l_1}^? \parallel \mathrm{publisher} :: {l_1}^* \parallel \mathrm{title} :: l_1 \parallel \mathrm{issued} :: l_2 \parallel \mathrm{homepage} :: {l_1}^? \\
&\quad \parallel \mathrm{seeAlso} :: {l_1}^* \parallel \mathrm{address} :: {l_1}^? \parallel (\mathrm{editor} :: l_3 \mid \mathrm{editor} :: t_1)^+ \parallel \mathrm{isbn} :: {l_1}^? \\
&\quad \parallel \mathrm{month} :: {l_2}^? \parallel \mathrm{note} :: {l_1}^? \parallel \mathrm{number} :: {l_2}^? \parallel \mathrm{series} :: {l_2}^? \parallel \mathrm{volume} :: {l_2}^? \\
< \mathrm{Inproceeding}(t_7) > &\rightarrow \mathrm{abstract} :: {l_1}^? \parallel \mathrm{booktitle} :: l_1 \parallel \mathrm{cdrom} :: {l_1}^? \parallel (\mathrm{creator} :: l_3 \mid \mathrm{creator} :: t_1)^+ \\
&\quad \parallel \mathrm{title} :: l_1 \parallel \mathrm{issued} :: l_2 \parallel \mathrm{partOf} :: {t_6}^? \parallel \mathrm{references} :: {t_4}^* \parallel \mathrm{homepage} :: l_1 \\
&\quad \parallel \mathrm{seeAlso} :: {l_1}^* \parallel \mathrm{number} :: {l_2}^? \parallel \mathrm{pages} :: {l_2}^? \\
< \mathrm{Www}(t_8) > &\rightarrow \mathrm{creator} :: {t_1}^+ \parallel \mathrm{title} :: l_1 \parallel \mathrm{homepage} :: {l_1}^? \parallel \mathrm{note} :: {l_1}^? \\
< \mathrm{MstersThesis}(t_9) > &\rightarrow \mathrm{creator} :: {t_1}^+ \parallel \mathrm{publisher} :: l_1 \parallel \mathrm{title} :: l_1 \parallel \mathrm{issued} :: l_2 \parallel \mathrm{homepage} :: {l_1}^? \\
< \mathrm{PhDThesis}(t_{10}) > &\rightarrow \mathrm{creator} :: {t_1}^+ \parallel \mathrm{publisher} :: {l_1}^+ \parallel \mathrm{title} :: l_1 \parallel \mathrm{issued} :: l_2 \parallel \mathrm{homepage} :: {l_1}^? \\
&\quad \parallel \mathrm{seeAlso} :: {l_1}^* \parallel \mathrm{isbn} :: {l_1}^? \parallel \mathrm{month} :: {l_2}^? \parallel \mathrm{number} :: {l_2}^? \\
&\quad \parallel \mathrm{series} :: {l_2}^? \parallel \mathrm{volume} :: {l_2}^? \\
< \mathrm{Book}(t_{11}) > &\rightarrow \mathrm{booktitle} :: {l_1}^? \parallel \mathrm{cdrom} :: {l_1}^? \parallel \mathrm{creator} :: {t_1}^+ \parallel \mathrm{publisher} :: {l_1}^? \parallel \mathrm{title} :: l_1 \\
&\quad \parallel \mathrm{issued} :: l_2 \parallel \mathrm{references} :: {t_4}^* \parallel \mathrm{homepage} :: {l_1}^? \parallel \mathrm{seeAlso} :: {l_1}^* \parallel \mathrm{isbn} :: {l_1}^? \\
&\quad \parallel (\mathrm{editor} :: l_3 \mid \mathrm{editor} :: t_1)^+ \parallel \mathrm{month} :: {l_2}^? \parallel \mathrm{series} :: {l_2}^? \parallel \mathrm{volume} :: {l_2}^?
\end{aligned}
$$

BSBM の型 (single type)

$$
\begin{aligned}
< \mathrm{String}(l_1) > &\rightarrow \epsilon \\
< \mathrm{Integer}(l_2) > &\rightarrow \epsilon \\
< \mathrm{Date}(l_3) > &\rightarrow \epsilon \\
< \mathrm{DateTime}(l_4) > &\rightarrow \epsilon \\
< \mathrm{Iri}(l_5) > &\rightarrow \epsilon \\
< \mathrm{Usd}(l_6) > &\rightarrow \epsilon \\
< \mathrm{ProductType}(t_1) > &\rightarrow \mathrm{label} :: l_1 \parallel \mathrm{comment} :: l_1 \parallel \mathrm{subClassOf} :: {t_1}^{?} \parallel \mathrm{publisher} :: l_5 \\
&\quad \parallel \mathrm{date} :: l_3 \\
< \mathrm{ProductFeature}(t_2) > &\rightarrow \mathrm{label} :: l_1 \parallel \mathrm{comment} :: l_1 \parallel \mathrm{publisher} :: l_5 \parallel \mathrm{date} :: l_3 \\
< \mathrm{Producer}(t_3) > &\rightarrow \mathrm{label} :: l_1 \parallel \mathrm{comment} :: l_1 \parallel \mathrm{homepage} :: l_5 \parallel \mathrm{country} :: l_5 \\
&\quad \parallel \mathrm{publisher} :: t_3 \parallel \mathrm{date} :: l_3 \\
< \mathrm{Product}(t_4) > &\rightarrow \mathrm{label} :: l_1 \parallel \mathrm{comment} :: l_1 \parallel \mathrm{producer} :: t_3 \parallel \mathrm{ProductFeature} :: {t_2}^{+} \\
&\quad \parallel \mathrm{productPropertyTextual} :: {l_1}^{[3,5]} \parallel \mathrm{productPropertyNumeric} :: {l_2}^{[3,5]} \\
&\quad \parallel \mathrm{type} :: {t_1}^{+} \parallel \mathrm{publisher} :: l_5 \parallel \mathrm{date} :: l_3 \\
< \mathrm{Vendor}(t_5) > &\rightarrow \mathrm{label} :: l_1 \parallel \mathrm{comment} :: l_1 \parallel \mathrm{homepage} :: l_5 \parallel \mathrm{country} :: l_5 \\
&\quad \parallel \mathrm{publisher} :: t_5 \parallel \mathrm{date} :: l_3 \\
< \mathrm{Offer}(t_6) > &\rightarrow \mathrm{product} :: t_4 \parallel \mathrm{vendor} :: t_5 \parallel \mathrm{price} :: l_6 \parallel \mathrm{validFrom} :: l_4 \\
&\quad \parallel \mathrm{validTo} :: l_4 \parallel \mathrm{deliveryDays} :: l_2 \parallel \mathrm{offerWebpage} :: l_5 \parallel \mathrm{publisher} :: t_5 \\
&\quad \parallel \mathrm{date} :: l_3 \\
< \mathrm{Person}(t_7) > &\rightarrow \mathrm{name} :: l_1 \parallel \mathrm{mbox\_sha1sum} :: l_1 \parallel \mathrm{country} :: l_5 \parallel \mathrm{publisher} :: l_5 \\
&\quad \parallel \mathrm{date} :: l_3 \\
< \mathrm{Review}(t_8) > &\rightarrow \mathrm{reviewFor} :: t_4 \parallel \mathrm{reviewer} :: t_7 \parallel \mathrm{reviewDate} :: l_4 \parallel \mathrm{title} :: l_1 \\
&\quad \parallel \mathrm{text} :: l_1 \parallel \mathrm{rating1} :: {l_2}^{?} \parallel \mathrm{rating2} :: {l_2}^{?} \parallel \mathrm{rating3} :: {l_2}^{?} \parallel \mathrm{rating4} :: {l_2}^{?} \\
&\quad \parallel \mathrm{publisher} :: l_5 \parallel \mathrm{date} :: l_3
\end{aligned}
$$

まず，グラフデータのデータサイズを変化させた場合に，アルゴリズムの実行時間がどのように変化するか計測した．比較対象は，3 章で示したアルゴリズム validation 及び hierarchical validation である．さらに，λ の初期定義の際に initialization アルゴリズムを用いず，λ の初期値として各出力ノードに全ての型を割り当てる場合の実行時間も計測する．なお，実行時間の計測は，ruby の benchmark ライブラリを利用した．結果を図 5.3 と図 5.4 に示す．

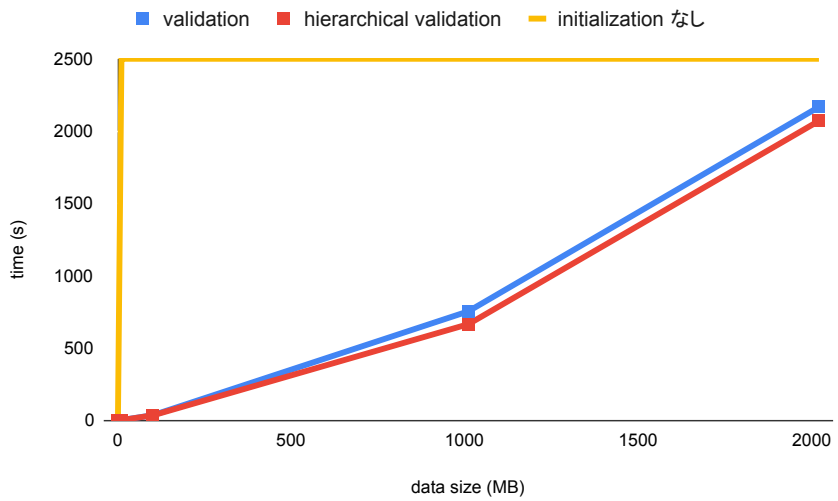


図 5.3: SP^{2}Bench に対する実行時間

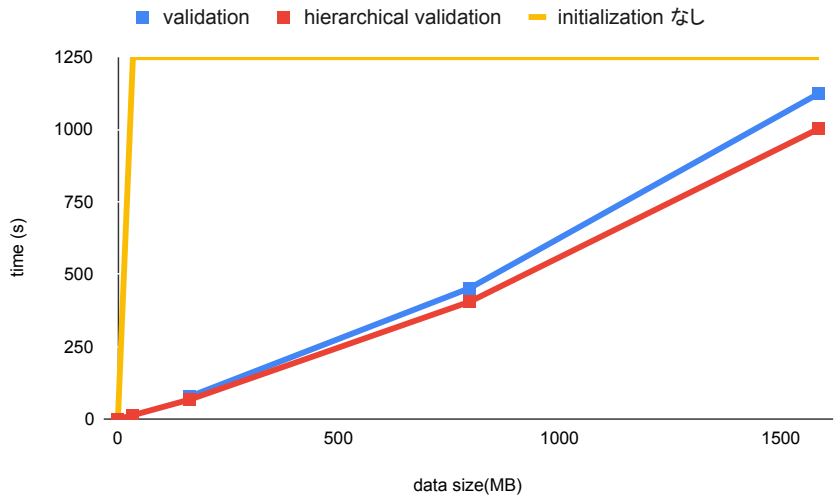


図 5.4: BSBM に対する実行時間

initialization アルゴリズムを用いない場合，どちらのデータセットでも実行時間は計測不能だった．よって本アルゴリズムでは，最初に出力ノードと結びつける型をある程度絞っておく必要があるといえる．一方で，本論文で提案する 2 つのアルゴリズムは，両方とも概ね線形時間で処理を完了している．また，実行時間は僅かに短い程度という結果だった．hierarchical validation の場合ファイルの読み込み回数が増えているが，$\mathit{fl\text{-}out\text{-}lab\text{-}type}^{\lambda}_{G}(n)$ の要素数が少ないため，データサイズが大きくなると，実行時間にもう少し差が出ると予

想していた．この結果は，出力ノードが検証している階層の型を割り当てられているかを毎回調べているため，λが大きくなると，その時間も増えることが原因と考えている．

次に，グラフデータのデータサイズを変化させた場合に，メモリの消費量がどのように変化するか計測した．計測には，ruby の Objectspace を利用した．結果を図 5.5 と図 5.6 に示す．

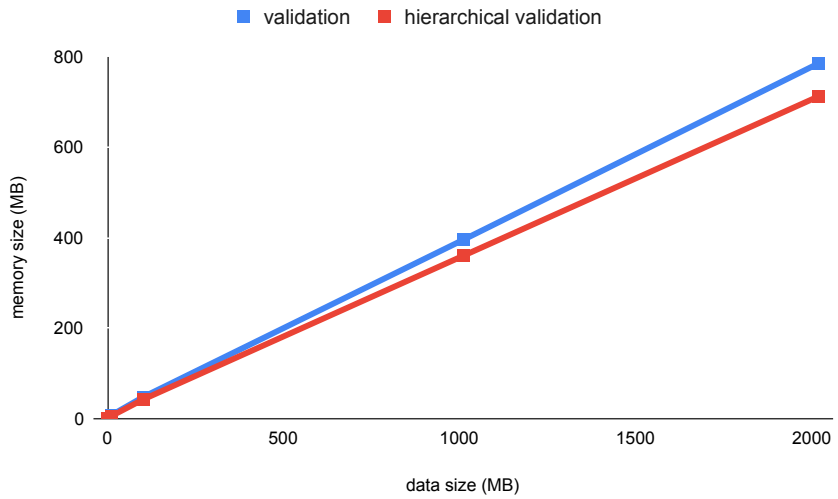


図 5.5: SP^{2}Bench に対するメモリ消費量

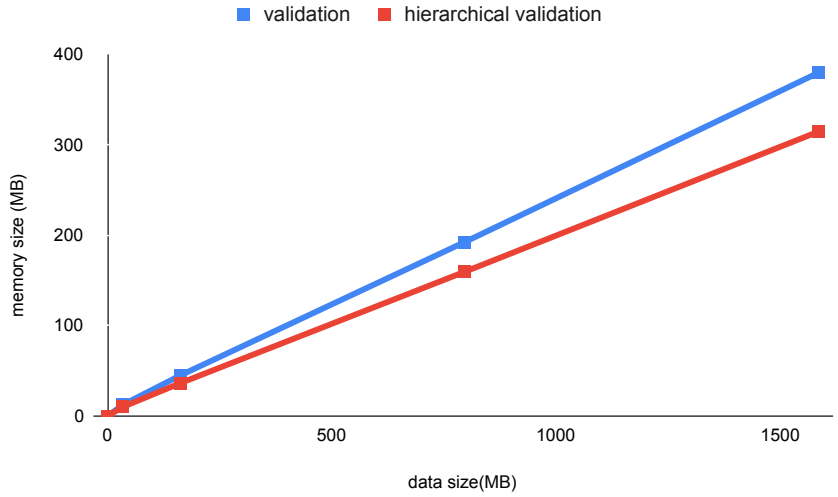


図 5.6: BSBM に対するメモリ消費量

両方のデータセットともデータサイズに対して 50%以下にメモリ消費量を抑えることができた．一方，2 つのアルゴリズム間では，実行時間に比べて差はあったといえる．よって hierarchical validation の方が省メモリであり，実行時間もほぼ一緒であるため，サイズの

大きいグラフに適していると考えられる．なお，グラフデータに対してスキーマは非常に小さくグラフサイズに影響されないので，階層を求める時間を考慮する必要はない．

5.2 修正手法の実験

本節では，実データに対して誤りを加えることで妥当性検証を中断し，正しく誤り部分の型を推定及びその修正方法を提示できるかを確かめる．実データとして以下のデータセットを用意した．

教科書 LOD[12][13] 1992 年施行の学習指導要領以降の検定教科書を対象として、書誌事項と教科等の関連情報を LOD 化している．この LOD を SP^{2}Bench と同様な形式のグラフデータに変更し，評価実験を行う．ただし，空白ノードの部分は自動的にランダムなノードを作成することで，エッジが (n, a, m) の形式となるようにしている．教科書 LOD グラフの全体像を図 5.5 に示す．なお，グラフのエッジ数，出力ノード数，データサイズは表 5.3 の通りである．

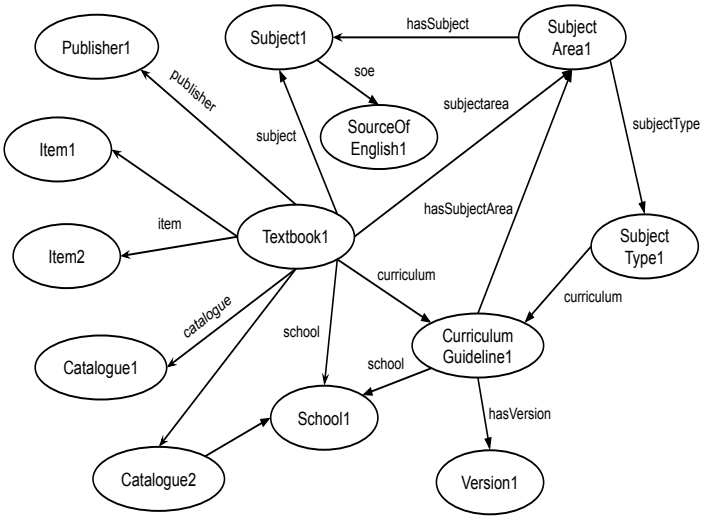


図 5.7: 教科書 LOD グラフの概観

表 5.3: 教科書 LOD のデータサイズ

$\|E\|$	$\|V*\|$	size(MB)
219,018	21,258	11.56

また，このグラフデータに妥当な ShEx の型も以下に示す．この ShEx は single type であり，グラフに対して妥当であることは確認済みである．

教科書 LOD の型 (single type)

$$
\begin{aligned}
< \text{Literal}(l_1) > &\rightarrow \epsilon \\
< \text{IRI}(l_2) > &\rightarrow \epsilon \\
< \text{School}(t_1) > &\rightarrow \text{name} :: {l_1}^{[0,3]} \parallel \text{sameAs} :: {l_2}^{?} \\
< \text{Catalogue}(t_2) > &\rightarrow \text{callNumber} :: {l_1}^{?} \parallel \text{datePublished} :: l_1 \parallel \text{itemID} :: {l_1}^{?} \\
&\quad \parallel \text{name} :: {l_1}^{[1,3]} \parallel \text{recordID} :: {l_1}^{?} \parallel \text{school} :: t_1 \parallel \text{seeAlso} :: {l_2}^{*} \\
&\quad \parallel \text{url} :: {l_2}^{?} \parallel \text{usageYear} :: l_1 \\
< \text{Publisher}(t_3) > &\rightarrow \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}^{*} \\
< \text{SubjectType}(t_4) > &\rightarrow \text{citation} :: l_1 \parallel \text{curriculum} :: t_8 \parallel \text{name} :: {l_1}^{[1,2]} \parallel \text{school} :: t_1 \\
< \text{SubjectArea}(t_5) > &\rightarrow \text{hasSubject} :: {t_6}^{*} \parallel \text{name} :: {l_1}^{[1,3]} \parallel \text{order} :: {l_1}^{+} \parallel \text{school} :: t_1 \\
&\quad \parallel \text{soe} :: {t_7}^{*} \parallel \text{subjectType} :: {t_4}^{*} \\
< \text{Subject}(t_6) > &\rightarrow \text{citation} :: {l_1}^{?} \parallel \text{name} :: {l_1}^{[1,3]} \parallel \text{order} :: l_1 \parallel \text{school} :: t_1 \\
&\quad \parallel \text{seeAlso} :: {l_2}^{*} \parallel \text{soe} :: {t_7}^{?} \parallel \text{subjectType} :: {t_4}^{*} \\
< \text{SOE}(t_7) > &\rightarrow \text{curriculum} :: {t_8}^{?} \parallel \text{name} :: {l_1}^{[1,2]} \parallel \text{seeAlso} :: {l_2}^{*} \\
< \text{CurriculumGuideline}(t_8) > &\rightarrow \text{datePublished} :: l_1 \parallel \text{hasSubjectArea} :: {t_5}^{+} \parallel \text{hasVersion} :: {t_9}^{*} \\
&\quad \parallel \text{name} :: {l_1}^{[1,3]} \parallel \text{school} :: t_1 \parallel \text{startDate} :: l_1 \parallel \text{url} :: {l_2}^{*} \\
< \text{Version}(t_9) > &\rightarrow \text{callNumber} :: {l_1}^{?} \parallel \text{citation} :: {l_1}^{[1,2]} \parallel \text{datePublished} :: l_1 \\
&\quad \parallel \text{isbn} :: {l_1}^{*} \parallel \text{itemID} :: {l_1}^{?} \parallel \text{name} :: {l_1}^{[1,2]} \parallel \text{recordID} :: {l_1}^{?} \\
&\quad \parallel \text{seeAlso} :: {l_2}^{*} \parallel \text{url} :: {l_2}^{*} \\
< \text{Textbook}(t_{10}) > &\rightarrow \text{authorizedYear} :: {l_1}^{*} \parallel \text{bookEdition} :: {l_1}^{*} \parallel \text{catalogue} :: t2^{+} \\
&\quad \parallel \text{curriculum} :: t_8 \parallel \text{dimensions} :: {l_1}^{?} \parallel \text{editor} :: l_1 \parallel \text{extent} :: {l_1}^{?} \\
&\quad \parallel \text{grade} :: {l_1}^{[0,6]} \parallel \text{isbn} :: {l_1}^{*} \parallel \text{item} :: {t_{11}}^{*} \parallel \text{name} :: l_1 \parallel \text{note} :: {l_1}^{*} \\
&\quad \parallel \text{publisher} :: {t_3}^{+} \parallel \text{school} :: t_1 \parallel \text{seeAlso} :: {l_2}^{*} \parallel \text{subject} :: {t_6}^{?} \\
&\quad \parallel \text{subjectArea} :: t_5 \parallel \text{textbookNumber} :: l_1 \parallel \text{textbookSymbol} :: l_1 \\
&\quad \parallel \text{usageYear} :: l_1 \\
< \text{Item}(t_{11}) > &\rightarrow (\text{callNumber} :: l_1 \parallel \text{recordID} :: l_1) \mid \\
&\quad (\text{rc:callNumber} :: l_1 \parallel \text{rc:recordID} :: l_1)
\end{aligned}
$$

評価実験のために，ある型に妥当なノードに対して，誤ったラベルに変更，入力型を誤った型に変更 (エッジの張りミス)，不要なエッジの追加，エッジの削除のいずれかを行い，妥当ではない状態にする．以下に実際の動作例を示す．ここでは，t_2 に妥当な出力ノード『<https://w3id.org/jp-textbook/catalogue/高等学校/1992>』について，エッジ『<https://w3id.org/jp-textbook/catalogue/高等学校/1992> textbook:usageYear 1993』を削除した．

例 4. 実行例

- ノード：<https://w3id.org/jp-textbook/catalogue/高等学校/1992>
- 修正対象：[callNumber::l1,datePublished::l1,itemID::l1, name::l1, name::l1, name::l1, recordID::l1, school::t1, seeAlso::l2, seeAlso::l2]
- 推定型：<Catalogue>(t2)
- 修正コスト：1.0
- 修正方法：
  callNumber::l1 をそのまま
  datePublished::l1 をそのまま
  itemID::l1 をそのまま
  name::l1 をそのまま
  name::l1 をそのまま
  name::l1 をそのまま
  recordID::l1 をそのまま
  school::t1 をそのまま
  seeAlso::l1 をそのまま
  seeAlso::l2 をそのまま
  usageYear::l1 を追加
  終わり
- 妥当なグラフ：[callNumber::l1,datePublished::l1,itemID::l1, name::l1, name::l1, name::l1, recordID::l1, school::t1, seeAlso::l2, seeAlso::l2, usageYear::l1]

この結果より，全ての型の中で最もこのノードに妥当な可能性が高い型は t_2 と推定され，usageYear :: l_1 を追加するという修正方法を提案することができた．これは誤りを加える前に妥当であった型と同じであり，正しく型を推定及び修正できたといえる．

同様にして，各型に妥当であるノード 5 個に誤りを加え，正しく修正できるかを検証していく．誤りの数を増やしていき，どこまでであれば正しく修正できるかを調べる．ここで，元々妥当だった型に修正でき，なおかつその修正方法が求めているもの (加えた誤りを直す) であれば点数を 1.0 とする．型は正しく推定できたが，その修正方法が求めているものと違う場合は 0.5 点とする．全く違う型を推定した場合は 0.0 点とする．その平均値を誤りの数に対する精度と考える．表 5.4 に結果を示す．

表 5.4: 誤りの数と精度

誤りの数	1	2	3	4	5
精度	0.99	0.97	0.86	0.71	0.57

概ね型は正しく推定でき，修正方法も誤りの数が 1,2 個程度であれば，著者が求めている修正方法を提案することができていた．一方で，最低出現回数が 0 回の *out-lab-type* に関しては，ひとつ前が繰り返しを許す *out-lab-type* の場合こちらを追加してしまう傾向が多々見られた．そのため，この手法ではある程度妥当な型を推定することは可能だが，適切な修正方法の提案に課題があるといえる．

また，誤りを加える際にエッジの削除のみに限定すると，最終的に t_1 や t_{11} といった型のサイズが小さいものを提案し始めることがわかっている．そのような極端な誤りの修正は難しいため，誤りの数や妥当でないノードの数に基準を設け，その基準を超えるようであれば修正せずにそのまま妥当性検証を終了する等，使い方も考慮すべきである．

第6章　まとめ

本論文では，有向グラフに対する ShEx の妥当性検証アルゴリズムを提案した．本アルゴリズムはグラフデータをメモリに載せずに検証を行い，階層の概念も導入した．評価実験の結果，提案アルゴリズムの効率性及び省メモリであることを確認することができた．一方で，階層を用いた妥当性検証では，省メモリではあるが実行時間の差がほぼ無く，課題を残す結果となった．

また，グラフデータが妥当でない場合，データ構造上誤りのあるノードに対して，最適な型とその型への修正手法も提案した．評価実験の結果，誤りが少数であれば概ね正しく提案できることを確認した．

今後の課題としては，階層化妥当性検証の実行時間を早くすることが挙げられる．現状では，グラフファイルに λ を記述し，検証している階層の型を含むか瞬時に判別できるようにする方法を考えている．この方法では，λ が主記憶に格納することができない場合でも，妥当性検証が行える可能性がある．修正手法に関しては，より型の修正方法の精度を上げること，否定等の表現にも対応できるよう拡張することを検討している．

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・ご助言がなければ，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永森光晴先生と阪口哲男先生に深く感謝いたします．また，多くのご助言をくださっ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). Shape Expressions (ShEx) 2.1 Primer. `http://shex.io/shex-primer/`. Accessed: 2019-12-22.

[2] World Wide Web Consortium (W3C). Shapes Constraint Language (SHACL). `https://www.w3.org/TR/shacl/`. Accessed: 2019-12-22.

[3] World Wide Web Consortium (W3C). RDF Schema 1.1. `https://www.w3.org/TR/rdfschema/`. Accessed: 2019-12-22.

[4] World Wide Web Consortium (W3C). Shape Expressions Language 2.1. `http://shex.io/shex-semantics/`. Accessed: 2019-12-22.

[5] Slawek Staworko, Iovka Boneva, Jose Labra Gayo, Samuel Hym, Eric Prud'hommeaux, and Harold Solbrig. Complexity and expressiveness of ShEx for RDF. In *18th International Conference on Database Theory, ICDT 2015*, Vol. 31, pp. 195–211, 2015.

[6] Jose E. Labra Gayo, EricPrud ' hommeaux, Iovka Boneva, Dimitris Kontokostas. *Validating RDF Data*. Synthesis Lectures on the Semantic Web: Theory and Technology. Morgan & Claypool, 2018.

[7] Ahmad Hemid, Lavdim Halilaj, Abderrahmane Khiat, and Steffen Lohmann. RDF Doctor: A Holistic Approach for Syntax Error Detection and Correction of RDF Data. In *11th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (IC3K 2019)*, pp. 508–516, 2019.

[8] K. Ikeda and N. Suzuki. An Algorithm for Finding top-K Valid XPath Queries. *IPSJ Transactions on Databases*, Vol. 7, No. 2, pp. 70–82, 2014.

[9] Eugene W. Myers and Warner A. Miller. Approximate matching of regular expressions. *Bulletin of mathematical biology*, Vol. 51, No. 1, pp. 5–37, 1989.

[10] Michael Schmidt, Thomas Hornung, Georg Lausen, and Christoph Pinkel. SP^2Bench: a SPARQL Performance Benchmark. In *Proceedings of the 25th International Conference on Data Engineering (ICDE 2009)*, pp. 222–233, 2009.

[11] Christian Bizer and Andreas Schultz. The Berlin SPARQL Benchmark. *International Journal on Semantic Web and Information Systems*, Vol. 5, pp. 1–24, 2009.

[12] 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.

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