人工知能学会研究会資料 SIG-SWO-048-03

SPARQLにおける集約の形式化とクエリ書き換え

Formalization of Aggregation in SPARQL and Query Rewriting

平山 健太 [1*]　　兼岩 憲 [1]
Kenta Hirayama[1]　　Ken Kaneiwa[1]

[1] 電気通信大学大学院 情報理工学研究科 情報・ネットワーク工学専攻
[1] Department of Computer and Network Engineering, Graduate School of Informatics and Engineering, The University of Electro-Communications

Abstract: RDF データは Web 上に存在するリソースに関するメタデータのオントロジーを記述する．その RDF データの検索に利用されるクエリ言語として，SPARQL がある．SPARQL のクエリ (SELECT 文など) において DISTINCT キーワードを用いると，検索結果から重複を排除できるが，重複排除処理の対象となるデータの件数が多いほど実行に時間がかかるという課題がある．本論文では，SPARQL の検索操作を形式化し，DISTINCT キーワードを用いたクエリの書き換えによって解決時間を短縮する手法を提案する．提案手法によって，実験を行ったクエリの一部について，既存手法に比べて実行時間が短縮された．

1　はじめに

従来の Web ページは単純なテキストとして扱われてきたが，セマンティック Web では Web ページの意味を機械的に取り扱うことができる．セマンティック Web は Web ページに対しメタデータやオントロジー (Web ページ自身の持つ意味に関する情報) を付与することで実現される．

セマンティック Web の技術要素のひとつに，複数の概念の間の関係を記述する RDF (Resource Description Framework) がある．RDF データは，URI(Uniform Resource Identifier) によって表現されるリソースの三つ組み (RDF トリプル) の集合である．RDF データはその機械可読性から，機械学習への応用が期待されている．RDF クエリはデータを取り扱う RDF ストアに対する問い合わせ文である．代表的な RDF クエリ言語に SPARQL (SPARQL Protocol and RDF Query Language) などがある．

SPARQL に準拠した RDF ストアの実装に，Apache Jena[1]などの例がある. これらの RDF ストアは，高速な検索を実現するためのインデックスを RDF ストア内部に構築する．さらに，RDF クエリの実行にかかる時間を定量的に見積もることで最適なクエリ解決の計画を作成する手法も提案されている．しかし，これらの研究成果のもとにおいてもなお，RDF データを用いた機械学習など，タスクの内容によって実行に多大な時間を要する場合があり，より大きな RDF データの活用のため，さらなる検索処理の効率化が求められている．

DISTINCT キーワードを用いている SPARQL クエリの解決では，重複排除処理の際に中間結果をすべて参照して重複検査を行わなければならず，動作が低速になるという課題がある．本研究では，SPARQL の DISTINCT キーワードによる結果の重複の排除に着目し，集約関数を用いた等価なクエリへの書き換えによるクエリ解決の効率化を試みる．このとき，クエリの等価性と効率性を示すために，SPARQL 代数によるクエリの評価が必要となる．したがって，本論文の前半部で多重集合に基づく SPARQL の意味論を定義し，後半部でクエリ書き換えの手法及び評価実験について述べる．評価は，ベンチマーク用の RDF データである LUBM に対しクエリ書き換えの有無によるクエリ解決時間の変化をみることにより行う．これにより，重複排除が必要な RDF クエリの解決が効率的になる．その結果，大規模な RDF データをより多くのタスクに利用することができる．

本論文の構成は次のとおりである．2 章では RDF データと SPARQL の構文を定義し，3 章に SPARQL の意味論について記述する．4 章にクエリ書き換えの手法について記述し，5 章に評価実験について記述する．6 章には本論文のまとめを記述する．

*連絡先：電気通信大学大学院 情報理工学研究科
情報・ネットワーク工学専攻
東京都調布市調布ヶ丘 1-5-1
E-mail: hirayama@uec.ac.jp

[1]https://jena.apache.org/

2 SPARQL の構文

本章では，RDF データと SPARQL クエリの構文を定義する．

2.1 RDF データ

RDF データは Web リソース間の関係を表現したデータの集合である．RDF データの最小単位は RDF トリプルである．RDF トリプルはリソースの URI(Uniform Resource Identifier) の三つ組 (s, p, o) によって表され，各語彙 s, p, o は順に主語，述語，目的語と呼ばれる．RDF データの例を図 1 に示す．図 1 の 1 行目に示した RDF トリプルは主語`<hydrogen>`，述語`rdf:type`，目的語`<atom>`から構成される．

空白ノード集合 B, リテラル集合 L, URI 集合 U に対し，RDF トリプルは $(B \cup U) \times U \times (B \cup L \cup U)$ の元である．RDF データは RDF トリプルの有限集合 $D \subseteq (B \cup U) \times U \times (B \cup L \cup U)$ から構成される．

```
<hydrogen> rdf:type <atom> .
<hydrogen> <atomic_weight> "1" .
<CH4> <contains> <hydrogen> .
<CH4> <contains> <carbon> .
<CH3COOH> <contains> <hydrogen> .
<CH3COOH> <contains> <carbon> .
<CH3COOH> <contains> <oxygen> .
...
```

図 1: RDF データの例

2.2 SPARQL クエリ

RDF データはクエリ言語 SPARQL を用いて検索される．そのとき，SPARQL クエリに条件文を付与することで制約付きの検索が可能になる．最初に，SPARQL クエリとその構成要素について定義する．

定義 2.1 (トリプルパターン)．トリプルパターンは RDF トリプルの成分のうち 0 個以上を変数に置き換えた表現である．URI 集合 U，リテラル集合 L，変数集合 V を用いると，トリプルパターンは $(U \cup V) \times (U \cup V) \times (L \cup U \cup V)$ の要素である．

本論文では，トリプルパターン内の変数を $?x, ?y$ のように表し，それ以外のものはリソース定数 ($L \cup U$ の要素) であるとする．

定義 2.2 (項)．変数集合 V，リソース集合 $B \cup L \cup U$ に対し，変数 $?x \in V$ とリソース $e \in B \cup L \cup U$ は項である．また，n 項関数 f と n 個の項 $t_1, \ldots, t_n$ に対し，$f(t_1, \ldots, t_n)$ は項である．

定義 2.3 (条件式)．項 t_1, t_2，二項述語 r に対し，$r(t_1, t_2)$ は (原子) 条件式である．また，R_1, R_2 が条件式であるとき，$\neg R_1, R_1 \wedge R_2, R_1 \vee R_2$ も条件式である．二項述語には $<, \leq, >, \geq, =, \neq$ などがある．

定義 2.4 (集約表現)．集約関数 $\mathbf{f}$，項 t に対して，$\mathbf{f}(t)$ は集約表現である．SPARQL 1.1[4] では，集約関数として`COUNT`, `SUM`, `MIN`, `MAX`, `AVG`, `SAMPLE`, `GROUP_CONCAT` の 7 種類が定義されている．

定義 2.5 (集約に関する条件式)．集約表現 $\mathbf{a}$，項 t，二項述語 r に対し，$r(\mathbf{a}, t)$ は集約に関する (原子) 条件式である．

次に SPARQL クエリに含まれる表現を定義する．

定義 2.6 (SPARQL 表現)．トリプルパターン tp，条件式 R，SPARQL 表現 Q_1, Q_2 に対し，$tp, Q_1 \textsc{And} Q_2$, $Q_1 \textsc{Union} Q_2, Q_1 \textsc{Filter} R$ は SPARQL 表現である．

定義 2.7 (SPARQL クエリ)．SPARQL 表現 Q，変数 $?v_1, \ldots, ?v_k$，変数集合 $S \subseteq V$，集約に関する条件式 $\mathbf{R}$ および集約表現 $\mathbf{a}_1, \ldots, \mathbf{a}_k$ に対し，以下は SPARQL クエリである．

(1). Q.
(2). $\textsc{Select}_S(Q)$.
(3). $\textsc{Select Distinct}(Q)$.
(4). $\textsc{Select}_{\mathbf{a}_1 \textsc{ as } ?v_1, \ldots, \mathbf{a}_k \textsc{ as } ?v_k}(Q) \textsc{ Group by } S \textsc{ Having } \mathbf{R}$.

S が空である場合，$\mathbf{R}$ が恒真のときそれぞれ $\textsc{Group by} S, \textsc{Having} \mathbf{R}$ を省略できる．

3 SPARQL の意味論

本章では SPARQL 代数により RDF データにおける SPARQL クエリの評価を定義する．

3.1 代入と集約関数

SPARQL クエリに対して，トリプルパターンに含まれる変数への代入は 1 つのクエリ解決を意味する．

定義 3.1 (代入)．変数集合 V からリソース集合 $B \cup L \cup U$ への部分関数 $\mu : V \to B \cup L \cup U$ を代入という．

本論文では，代入 μ を部分集合 $\mu \subseteq V \times (B \cup L \cup U)$ としても扱う．代入の全体多重集合を Sub と表し，代入 μ の定義域を $dom(\mu)$ と表す．

定義 3.2 (代入の拡張)．項とトリプルパターンに対し代入 μ の拡張 $\mu(E)$ を次のように定義する．ここで，$t_1, \ldots, t_n$ は項，f は n 項関数である．

(1). $e \in R \Rightarrow \mu(e) = e$.
(2). $\mu(f(t_1, \ldots, t_n)) = f(\mu(t_1), \ldots, \mu(t_n))$.
(3). $\mu((s, p, o)) = (\mu(s), \mu(p), \mu(o))$.

この定義により，トリプルパターン tp に代入 μ を適用した結果を $\mu(tp)$ と表す．

定義 3.3 (条件式と充足可能関係)．ある代入 μ により条件式 R が真となることを $\mu \models R$ と表し，関係 $\models$ を充足可能関係と呼ぶ．任意の条件式に対し，この関係を帰納的に次のように定義する．

(1). $\mu \models r(t_1, t_2) \Leftrightarrow r(\mu(t_1), \mu(t_2))$.
(2). $\mu \models \neg R_1 \Leftrightarrow \mu \not\models R_1$.
(3). $\mu \models R_1 \wedge R_2 \Leftrightarrow \mu \models R_1 \wedge \mu \models R_2$.
(4). $\mu \models R_1 \vee R_2 \Leftrightarrow \mu \models R_1 \vee \mu \models R_2$.

定義 3.4 (互換な代入)．代入 μ_1, μ_2 に対し，すべての $?v \in dom(\mu_1) \cap dom(\mu_2)$ が $\mu_1(?v) = \mu_2(?v)$ を満たすとき，μ_1 と μ_2 が互換であるといい，$\mu_1 \sim \mu_2$ と表す．$\mu_1 \sim \mu_2$ のとき，代入の結合 $\mu_1 \cup \mu_2$ が定義される．代入の結合もまた代入である (右一意性に注意)．μ_1 と μ_2 の定義域から与えられる共通部分集合 $V \subseteq dom(\mu_1) \cap dom(\mu_2)$ について，$\forall ?v \in V, \mu_1(?v) = \mu_2(?v)$ であるとき，$\mu_1 \sim_V \mu_2$ と表す．二項関係 $\sim$ および $\sim_V$ は常に同値関係である．特に，$\sim_\emptyset$ は自明な関係となる．

図 2 は関係 $\sim, \sim_V$ の視覚的なイメージを表している．例えば，2 つの代入 μ_1, μ_3 の間には $\mu_1(?y) = \mu_3(?y)$ (= <2>) が成り立つので，$\mu_1 \sim_{\{?y\}} \mu_3$ が成り立つ．

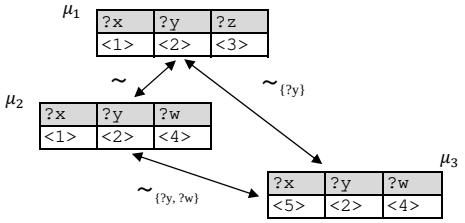

図 2: 関係 $\sim, \sim_V$ のイメージ

SPARQL クエリは RDF データ上のすべての解決 (代入) を集合として返す．例えば SPARQL クエリ Q のすべての解決が 3 つの代入 $\mu_1, \mu_2, \mu_3 \in Sub$ ならば，それらの代入をまとめて集合 $\{\mu_1, \mu_2, \mu_3\}$ (代入集合と呼ぶ) で表す．本論文における代入集合には要素の重複を許すものとする．集合を内包的あるいは外延的に表す場合，その集合の要素に重複を許さないとき $\{\cdot\}$，それ以外のとき $[\cdot]$ でくくって表す．

集約関数 $\mathbf{f}$ は SPARQL クエリの代入集合 Ω を集約しリソース r $(\in R)$ を求める関数である．集約関数 `COUNT`, `SUM`, `MIN`, `MAX`, `AVG`, `SAMPLE`, `GROUP_CONCAT` の解釈は次のように定義される．

定義 3.5 (集約関数への代入集合の適用)．SPARQL 1.1[4] に定義されている集約関数を以下のように定義する．ここで，t は項である．また，$\bullet$ は文字列連結演算子とする．

(1). $\Omega(\textsc{Count}_*) = |\Omega|$
(2). $\Omega(\textsc{Count}_t) = |[\mu(t) | \mu \in \Omega]|$.
(3). $\Omega(\textsc{Sum}_t) = \sum_{\mu \in \Omega} \mu(t)$.
(4). $\Omega(\textsc{Min}_t) = \min_{\mu \in \Omega} \mu(t)$.
(5). $\Omega(\textsc{Max}_t) = \max_{\mu \in \Omega} \mu(t)$.
(6). $\Omega(\textsc{Avg}_t) = \textsc{Sum}_t(\Omega) / \textsc{Count}_t(\Omega)$.
(7). $\Omega(\textsc{Sample}_t) = \mu(t)$ $(\exists \mu \in \Omega)$.
(8). $\Omega(\textsc{Group\_concat}_t) = \bullet_{\mu \in \Omega} \mu(t)$.

定義 3.6 (代入集合に関する拡張)．代入集合 Ω と変数 $?v$，リソース e に対し，拡張 $\Omega(?v), \Omega(e)$ を次のように定義する．

(1). $\Omega(?v) = \left(\bigcap_{\mu \in \Omega} \mu\right)(?v)$.
(2). $\Omega(e) = e$.

定義 3.7 (集約に関する条件式)．以下のように，代入集合 Ω が集約に関する条件式 $\mathbf{R}$ を満たすことを $\Omega \models \mathbf{R}$ と表す．

$$\Omega \models r(\mathbf{a}, t) \Leftrightarrow r(\Omega(\mathbf{a}), \Omega(t)).$$

例えば，$\Omega \models (\textsc{Count}_* = 3) \Leftrightarrow \Omega(\textsc{Count}_*) = 3 \Leftrightarrow |\Omega| = 3$ となる．

3.2 SPARQL 代数

本章では，SPARQL クエリの検索処理を SPARQL 代数として数学的に定義する．

定義 3.8 (SPARQL 代数)．代入集合 $\Omega, \Omega_l, \Omega_r$ に対し，結合 $(\bowtie)$, 直和 $(\sqcup)$, 射影 (π_S), 選択 (σ_R), 集約 $(\gamma_{S;\{(v_1,\mathbf{a}_1),\ldots,(v_k,\mathbf{a}_k)\};R})$ を次のように定義する．これらの操作を SPARQL 代数と総称する．これらの操作のうち，$\bowtie, \sqcup$ はともに結合的かつ可換な操作である．

(1). $\Omega_l \bowtie \Omega_r = [\mu_l \cup \mu_r | \mu_l \in \Omega_l, \mu_r \in \Omega_r : \mu_l \sim \mu_r]$.
(2). $\Omega_l \sqcup \Omega_r = [\mu | \mu \in \Omega_l \vee \mu \in \Omega_r]$.
(3). $\pi_S(\Omega) = [\mu_1 | \exists \mu_2 : \mu_1 \cup \mu_2 \in \Omega \wedge dom(\mu_1) \subseteq S \wedge dom(\mu_2) \cap S = \emptyset]$.
(4). $\sigma_R(\Omega) = [\mu \in \Omega | \mu \models R]$.
(5). $\gamma_{S;\{(v_1,\mathbf{a}_1),\ldots,(v_k,\mathbf{a}_k)\};\mathbf{R}}(\Omega) = \{\{(v_1, \Omega'(\mathbf{a}_1)), \ldots, (v_k, \Omega'(\mathbf{a}_k))\} | \Omega' \in (\Omega / \sim_S), \Omega' \models \mathbf{R}\}$.

ただし，$v_1, \ldots, v_k$ は互いに相異なる変数，S は変数集合，$\mathbf{a}_1, \ldots, \mathbf{a}_k$ は集約表現である．また，R は条件式，$\mathbf{R}$ は集約に関する条件式である．

例えば，$\Omega = \{\mu_1, \mu_2, \mu_3\}, \mu_1 = \{(?x, 1), (?y, 1)\}$, $\mu_2 = \{(?x, 1), (?y, 2)\}, \mu_3 = \{(?x, 2), (?y, 4)\}$ が与えられたとする．このとき，定義 3.8 の *(5)* によって $\gamma_{\{?x\};\{(?x,?x),(?s,\mathrm{SUM}_{?y})\};\mathrm{SUM}_{?y}\geq 4}$ を計算することを考える．$\mu_1 \sim_{\{?x\}} \mu_2, \mu_2 \not\sim_{\{?x\}} \mu_3, \mu_3 \not\sim_{\{?x\}} \mu_1$ だから，$\Omega/\sim_{\{?x\}} = \{\{\mu_1, \mu_2\}, \{\mu_3\}\}$ となる．これらから，

$$\begin{aligned} \{\mu_1, \mu_2\}(?x) &= (\mu_1 \cup \mu_2)(?x) = 1, \\ \{\mu_1, \mu_2\}(\mathrm{SUM}_{?y}) &= \mu_1(?y) + \mu_2(?y) = 3, \\ \{\mu_3\}(?x) &= \mu_3(?x) = 2, \\ \{\mu_3\}(\mathrm{SUM}_{?y}) &= \mu_3(?y) = 4 \end{aligned}$$

が得られる．このうち，$\Omega \models \mathrm{SUM}_{?y} \geq 4$ を満たす代入集合 Ω は $\{\mu_3\}$ だから，$\gamma_{\{?x\};\{(?x,?x),(?s,\mathrm{SUM}_{?y})\};\mathrm{SUM}_{?y}\geq 4}$ の値は $\{\{(?x, 2), (?s, 4)\}\}$ と計算される．

定義 3.9 (SPARQL クエリの評価)．SPARQL クエリの評価を SPARQL 代数によって次のように定義する．SPARQL クエリ Q に対し，それを RDF データ D において評価した結果を $[\![Q]\!]_D^+$ と表す．また，$[\![Q]\!]_D^+$ から重複を排除した集合を $[\![Q]\!]_D = \{\mu | \mu \in [\![Q]\!]_D^+\}$ と表す．

(1). $[\![tp]\!]_D^+ = \{\mu | dom(\mu) = Vars(tp) \wedge \mu(tp) \in D\}$.
(2). $[\![Q_1 \,\textsc{And}\, Q_2]\!]_D^+ = [\![Q_1]\!]_D^+ \bowtie [\![Q_2]\!]_D^+$.
(3). $[\![Q_1 \,\textsc{Union}\, Q_2]\!]_D^+ = [\![Q_1]\!]_D^+ \sqcup [\![Q_2]\!]_D^+$.
(4). $[\![Q \,\textsc{Filter}\, R]\!]_D^+ = \sigma_R([\![Q]\!]_D^+)$.
(5). $[\![\textsc{Select}_S(Q)]\!]_D^+ = \pi_S([\![Q]\!]_D^+)$.
(6). $[\![\textsc{Select Distinct}_S(Q)]\!]_D^+ = [\![\textsc{Select}_S(Q)]\!]_D$.
(7). $[\![\textsc{Select}_{\mathbf{a}_1 \,\textsc{as}\, v_1, \ldots, \mathbf{a}_k \,\textsc{as}\, v_k}(Q) \,\textsc{Group by}\, S \,\textsc{Having}\, \mathbf{R}]\!]_D^+ = \gamma_{S;\{(v_1,\mathbf{a}_1),\ldots,(v_k,\mathbf{a}_k)\};\mathbf{R}}([\![Q]\!]_D^+)$.

誤解を招くおそれのないときは，RDF データを表す D を省略して $[\![Q]\!]^+$ のように表す．

4 SPARQL クエリ書き換え

本章では，DISTINCT キーワードを用いたクエリに対し，等価なクエリへの書き換えを行う．

4.1 クエリ書き換えの手続き

クエリ書き換えでは，DISTINCT キーワードが使われ，選択が 1 変数で，クエリパターンがトリプルパターンのみで構成されるクエリを対象とする．選択の対象となっている唯一の変数を**主変数**と呼び，それ以外の変数を総称して**副変数**と呼ぶ．図 3 に書き換え対象のクエリの例を示す．図 3 のクエリの主変数は`?p` である．また，副変数は`?x1`，`?x2`，`?x3` である．

```
SELECT DISTINCT ?p WHERE
{
  ?x1 ?p <atom>.
  ?x2 ?p <atom>.
  ?x2 <contains> ?x3.
}
```

図 3: 書き換え対象のクエリの例

n 個のトリプルパターンからなるクエリ

$$Q_o = \textsc{Select Distinct}\{?v_p\}(tp_1 \,\textsc{And}\, tp_2 \,\textsc{And} \ldots \,\textsc{And}\, tp_n). \quad (1)$$

の書き換えを考える．Q の変数集合を S とし，トリプルパターンの集合を $TP = \{tp_1, tp_2, \ldots, tp_n\}$ とする．また，クエリ Q の主変数を $?v_p$ とする．

副変数 $?v \in S \setminus \{?v_p\}$ について，$tp \in TP, ?v \in Var(t)$ のとき tp から $?v$ への有向辺を作成する．このようにしてクエリ Q に対して作成される有向辺の集合を $A = \{(tp, ?v) | ?v \in V \setminus \{?v_p\}, tp \in TP, ?v \in Var(tp)\}$ として依存関係グラフ $G = (TP \cup V \setminus \{?v_p\}, A)$ を構成する．

G に含まれる各弱連結成分を $G_1, G_2, \ldots, G_k$ とする．G の i 個目の弱連結成分に含まれるトリプルパターンの集合を TP_i, 変数の集合を V_i, 有向辺の集合を A_i とする $(1 \leq i \leq k)$.

G に含まれるすべての弱連結成分に，主変数を含むトリプルパターンが存在している場合 (すなわち，$\forall i \in \{1, 2, \ldots, k\}, \exists tp \in TP_i, ?v_p \in Var(tp)$ を満たす場合)，G に含まれる各弱連結成分に対応するサブクエリ $Q_1, Q_2, \ldots, Q_k$ を作成することで，クエリ書き換えを適用することが可能である．ただし，作成する i 個目のサブクエリ Q_i について，$c_i = |TP_i|, TP_i = \{tp_{i1}, \ldots, tp_{ic_i}\}$ とおき，$Q_i = \textsc{Select}_{\{?p\}}(tp_{i1} \,\textsc{And} \ldots \,\textsc{And}\, tp_{ic_i})$ とする．例えば，図 3 にに示したクエリからは図 4 に示すような依存関係グラフが構成できる．

代入 μ が k 個のサブクエリの解決の集合 $[\![Q_1]\!], [\![Q_2]\!], \ldots [\![Q_k]\!]$ のすべてに含まれるとき，かつその場合に限り，μ はもとのクエリ Q の解決結果 $[\![Q]\!]$ にも含まれる．よって，解決の集合 $[\![Q_1]\!], [\![Q_2]\!], \ldots, [\![Q_k]\!]$ の共通部分を計算するクエリを作成する．ただし，SPARQL には複数のクエリの結果の共通部分を直接計算するための構文は存在しないので，集約関数 (COUNT 関数) によって共通部分を求めるようクエリを書き換える．クエリ Q_o に対しクエリ書き換え適用後のクエリ Q_r は次のようになる．

$$\begin{aligned} Q_r = {} & \textsc{Select}_{?v_p \,\textsc{as}\, ?v_p} \\ & ((\textsc{Select Distinct}_{\{?v_p\}}(Q_1)) \,\textsc{Union} \\ & \ldots \,\textsc{Union}(\textsc{Select Distinct}_{\{?v_p\}}(Q_k))) \\ & \textsc{Group by}\{?v_p\} \,\textsc{Having}(\textsc{Count}(?v_p) = k). \end{aligned} \quad (2)$$

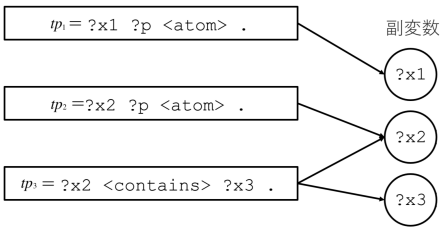

図 4: 図 3 のクエリに対する依存関係グラフ

```
SELECT ?p WHERE
{
    {  SELECT DISTINCT ?p WHERE
       { ?x1 ?p <atom>. }
  }
  UNION
  {  SELECT DISTINCT ?p WHERE
     { ?x2 ?p <atom>.
       ?x2 <contains> ?x3. }
  }
}
GROUP BY (?p) HAVING (COUNT(?p) = 2)
```

図 5: 書き換え後のクエリの例

3 章の意味論により，クエリ Q_o と Q_r の等価性が以下のように成り立つ．図 3 のクエリを Q_o とすると，クエリ書き換え適用後のクエリは図 5 のようになる．

定理 4.1. $[\![Q_r]\!]^+ = [\![Q_o]\!]^+$.

4.2 実行時間の短縮

書き換え前のクエリ Q_o について，トリプルパターンの連言 tp_1 AND tp_2 AND $\ldots tp_n$ の解決時間と結果の個数の上界は $O(|D|^n), |D|^n$ で与えられる．tp_1 AND tp_2 AND $\ldots tp_n$ の解決結果をソートし，重複を排除することを考える．ソートにかかる計算時間は比較ソートを用いる場合 $O(|D|^n \log(|D|^n)) = O(n|D|^n \log|D|)$，ソート後の処理にかかる計算時間は $O(|D|^n)$ で与えられる．したがって，全体では $O(n|D|^n \log|D|)$ 時間で処理ができる．

一方で，書き換え後のクエリ Q_r の解決時間について考える．サブクエリ Q_i に含まれるトリプルパターンの連言 tp_{i1} AND tp_2 AND $\ldots tp_{ic_i}$ の解決時間と結果の個数は $O(|D|^{c_i}), |D|^{c_i}$ で与えられる．したがって，書き換え前のクエリ Q_o の場合と同様にして，SELECT DISTINCT $_{\{?p\}}(Q_i)$ の解決時間と解決結果の個数の上界を $O(c_i |D|^{c_i} \log|D|), |D|^{c_i}$ と計算できる．

したがって，SELECT DISTINCT$_{\{?p\}}(Q_1)$ UNION $\ldots$ UNION SELECT DISTINCT$_{\{?p\}}(Q_k)$ の解決時間と結果の個数の上界は $\max_{1 \le i \le k} c_i = c$ として $O(c^2|D|^c \log|D|)$, $c|D|^c$ と計算できる．これらの計算結果について COUNT 関数による計数と HAVING 句による選択はいずれも $O(c|D|^c)$ 時間で処理できるので，全体として $O(c^2|D|^c log|D|)$ 時間で解決できる．

ここで，c はクエリ書き換えを行った後のサブクエリに含まれるトリプルパターンの最大数だから，$c \le n$ すなわち $c^2 \le n^2$ が成り立つ．さらに，一般的に用いられている RDF データでは，$n \ll |D|$ より $|D|/n \gg 1$ である．これらのことから，次のことが言える．

$$\begin{aligned} T_r &= O(c^2|D|^c \log|D|) \\ &\le O(n^2|D|^c \log|D|) \\ &\ll O(n|D|^{c+1} \log|D|) \\ &\le O(n|D|^n \log|D|) = T_o. \end{aligned}$$

以上から $T_r \ll T_o$ となり，解決時間の上界が小さくなることがわかる．

5 評価実験

クエリ書き換えの有効性を確認するため，重複排除を必要とする 10 個のクエリを実行し，その実行時間を測定した．実験に用いたデータセットには RDF ベンチマークデータである LUBM[2]を用いた．また，データセット生成時のパラメータである大学の数は 10 とした (LUBM10 と呼ぶ)．

また，RDF ストアは FROST[3] 1.1, Jena 3.1.1, Virtuoso[4] 7.1.0 の 3 種を用いる．ただし，FROST は DISTINCT，集約関数の機能を追加実装している．実験はクエリ書き換えなし，クエリ書き換えありの 2 通りの条件で行った．実験環境は，OS: Windows 10 Education 64bit, CPU:Intel®Core™i7-6800K @ 3.40 GHz 3.40 GHz, 実装メモリ: 64.00 GB　と設定した．また，クエリの実行時間が 600000 ms を超えた場合はタイムアウトとする．LUBM10 に対する DISTINCT キーワードを含む 10 個のクエリを実行した結果，書き換えなしのとき FROST で Q_{10}，Jena の場合で Q_3, Q_9, Q_{10}，Virtuoso の場合で Q_{10} がタイムアウトとなった．

FROST, Jena, Virtuoso それぞれの RDF ストアによる 10 個のクエリの実行時間をそれぞれ図 6，図 7，図 8 に示す．ただし，タイムアウトはグラフ中の実行時間を 600000 ms と表示し，さらに $*$ をつけて示す．これら結果の比較から，重複排除に対してクエリ書き換えが有効な場合とそうでない場合が存在する．Q_1 と Q_{10} は FROST, Jena, Virtuoso のすべての場合において，LUBM10 で 200 倍以上の高速化をクエリ書き換えで達成した．Q_1 と Q_{10} の内容を図 9, 図 10 に示す．特に，クエリ書き換えなしの場合にどの RDF ストアでも 10 分以上を要した Q_{10} の解決がクエリ書き換えによって

[2]http://swat.cse.lehigh.edu/projects/lubm/
[3]http://www.sw.cei.uec.ac.jp/frost/index-j.html
[4]https://virtuoso.openlinksw.com/

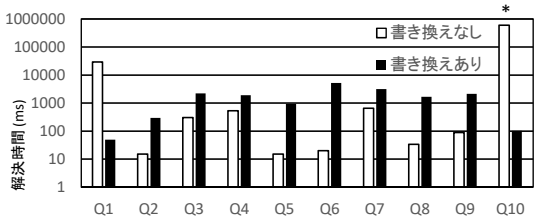

図 6: FROST による LUBM クエリの解決時間

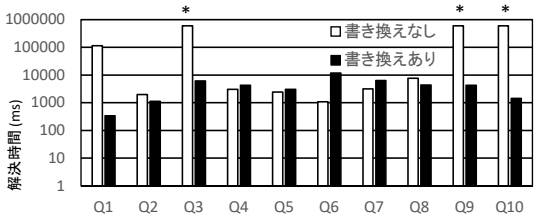

図 7: Jena による LUBM クエリの解決時間

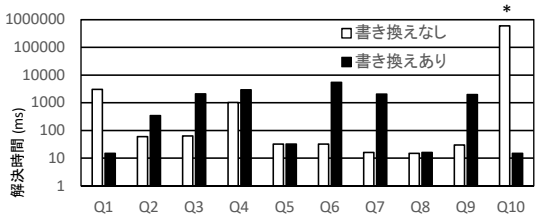

図 8: Virtuoso による LUBM クエリの解決時間

現実的な時間内に解決できた．これらのクエリはいずれも主変数を述語とするトリプルパターンが複数含まれるという点で共通している．

一方で，FROST を用いた場合の Q_2, Q_5, Q_8 など，クエリ書き換えにが実行時間を長くするケースもある．その理由として，クエリが複数のサブクエリに分割されたことで，各サブクエリの実行結果に対して課される制約が元のクエリよりも少なくなり，各サブクエリの解決が増加したことが考えられる．これらのことから，すべてのクエリを書き換えず，クエリの形状，RDF データの分布などから解決時間の短縮を推定する手法が必要である．

6 おわりに

本研究では，重複排除 (DISTINCT キーワード) を必要とする RDF クエリの高速化のための SPARQL クエリの書き換え手法を提案した．特に，SPARQL クエリの意味論を SPARQL 代数によって形式化し，集約関数を含むクエリの等価性と効率性を示した．実験により，提案した手法によって重複排除を必要とするクエリの一部を高速化できることが確認できた．

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-
     syntax-ns#>
PREFIX ub: <http://swat.cse.lehigh.edu/onto/univ-
     bench.owl#>
SELECT DISTINCT ?Y
WHERE
{
  ?X ?Y ub:FullProfessor.
  ?Z ?Y ub:AssociateProfessor.
}
```

図 9: 書き換え前のクエリ Q_1

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-
     syntax-ns#>
PREFIX ub: <http://swat.cse.lehigh.edu/onto/univ-
     bench.owl#>
SELECT DISTINCT ?P
WHERE
{
  ?X ?P ub:GraduateStudent .
  ?Y ?P ub:University .
  ?Z ?P ub:Department .
}
```

図 10: 書き換え前のクエリ Q_{10}

今後の課題として，より広範な SPARQL クエリの処理に対応した処理の定式化，本研究の手法によって解決時間の短縮ができるかどうかを判断する手法の開発，より多様なクエリに対する書き換え手法の開発が挙げられる．

参考文献

[1] 藤原 浩司, 兼岩 憲. 大規模 RDF グラフに対する高速検索とデータ圧縮の両立, 人工知能学会セマンティックウェブとオントロジー研究会資料, SIG-SWO-A1402-08, 2014.

[2] 藤原 浩司, 兼岩 憲. 大規模 RDF グラフのための効率的なクエリ解決, 人工知能学会論文誌, Vol. 29, No. 4, pp. 364-374, 2014.

[3] Thomas Neumann and Gerhard Weikum. *The RDF-3X engine for scalable management of RDF data*, The VLDB Journal Vol. 19, Issue 1, pp.91-113, 2010.

[4] Eric Prud'hommeaux and Andy Seaborne. *SPARQL 1.1 Query Language*, 2013. https://www.w3.org/TR/2013/REC-sparql11-query-20130321/

[5] S. Groppe. *Data Management and Query Processing in Semantic Web Databases*, Berlin Heidelberg: Springer-Verlag, 2011.

[6] S. Harris. *SPARQL query processing with conventional relational database systems.* SSWS, 2005.

[7] M. Vardi. *The Complexity of Relational Query Languages (Extended Abstract).* STOC 1982, pages 137-146, 1982.

[8] D. Marin. *RDF Formalization.* Santiago de Chile, Tech. Report Univ. Chile, TR/DCC-2006-8, 2004.

[9] P. Haase, et al. *A Comparison of RDF Query Languages.* ISWC 2004, pages 502-517, 2004.

[10] Jorge P'erez, Marcelo Arenas, and Claudio Gutierrez. *Semantics and Complexity of SPARQL.* Lecture Notes in Computer Science 4273, pp.30-43, 2006.