セマンティックWebにおけるクエリ変換を用いた分散知識からの知識獲得

A Query Mapping Method for Knowledge Retrieval from Distributed Knowledge on the Semantic Web

深谷　崇元 † 徳田 雄洋 †

Takayuki Fukatani Takehiro Tokuda

† 東京工業大学大学院情報理工学研究科
計算工学専攻
Department of Computer Science, Tokyo Institute of Technology
{fukatani, tokuda}@tt.cs.titech.ac.jp

近年, W3C によりセマンティック Web 関連技術群が勧告され, Web 上でオントロジを用いた機械可読の形式による知識の記述および統合が可能になった. しかし独立に定められた複数のオントロジに対して, これらを用いた知識の統合を行った場合, オントロジ間の意味の不整合やクエリ処理時の計算量の増加が問題となる. これらの問題に対し本研究では一つのオントロジを用いたクエリ文を他のオントロジを用いたクエリ文へ変換し, それぞれのオントロジに対してクエリ処理を行う方法を提案する. 提案手法によりクエリ処理の計算時間が減ることを実験により確認した. また, 提案手法の応用により RDF データベースや関係データベースに蓄えられた知識も直接統合できることを確認した.

1 はじめに

近年, Web の発達により Web 上の莫大な知識や資源をいかに効率良く利用するかが課題となっている. 解決策として Google[10] に代表される自然言語処理による知識や資源の検索が主流であるが, それとは別に機械可読な形式を用いた知識自体の記述や資源のメタデータとしての知識の記述を行い, 知識や資源の自動処理を可能とするセマンティック Web への期待が高まっている. そのような流れの中, RDF/OWL[2, 8] が W3C によって勧告され, インターネットやイントラネット上で独自に作成したスキーマ (オントロジ) を用い知識を記述することが理論上は可能となった. 各団体が独自に作成, 変更したオントロジが複数存在する場合, それぞれのオントロジを用いた知識 (分散知識と呼ぶ) を統合し, 統合した知識から必要な知識を獲得することが重要になる.

分散知識を統合する方法として, 知識を一箇所に集め各オントロジが持つクラスやプロパティ, インスタンスなどの要素間の関係を定義する手法が考えられる. しかし, この方法では

- オントロジの巨大化
- 各オントロジ間の意味の不整合
- 誤った知識による誤った推論
- 関係データベースや RDF データベースの統合の非効率

といった問題が存在する. これらの問題を解決するために本研究では各分散知識の用いるオントロジに対してより一般的なクラス, プロパティを定義するインターフェースオントロジの導入および, クエリを変換し, 推論を各分散知識ごとに行う分散推論を提案する.

提案手法では音楽や大学といった特定のドメインに属する一定個数の分散知識からの知識獲得を実現する. 各分散知識ごとに推論を行うため, 知識全体を知ることにより新たな推論が行えるような場合には適切ではない. しかし, Web においては分散知識は独立であることが多いと予想されるため, 有用であると考えられる.

2 セマンティック Web と知識統合

2.1 RDF/OWL

RDF により知識の記述を, OWL により RDF で用いられるオントロジの定義を行うことができる. OWL のうち OWL DL/Lite は Description Logic(DL)[1] を基にした理論が与えられており, 計

算決定性, 計算完全性を持つことが証明されている. 本論文では OWL DL/Lite を扱い, OWL と表記した, DL/Lite を指すものとする.

RDF ではクラス-インスタンス関係, プロパティを用いたインスタンス間の 2 項関係, インスタンスとデータリテラル間の 2 項関係を記述することができる. OWL では RDF にて使われるクラスやプロパティの階層関係や定義を記述する. また RDF/OWL で記述された知識に対し DL の推論エンジンを応用できることが知られている [6]. RDF/OWL ではクラス, プロパティ, インスタンスが URI で表されるため, 他の文書から参照することができる. そのため既にあるオントロジを新たなオントロジで再利用することや, 異なるオントロジ間のクラスやプロパティに対して外部から等価関係, 親子関係などを記述することができ, インスタンスに対しては等価関係や非等価関係を記述できる [2].

2.2 集中推論による知識統合と課題

DL の推論エンジンを用いて RDF/OWL 文書から知識獲得をする場合, クエリとして与えられたある概念に属するインスタンスの集合を推論する Instance Retrieval が重要になる. DL の推論アルゴリズムとして主にタブローアルゴリズムの研究がなされており, RDF/OWL に対しても処理系として Pellet が提供されている [13]. しかし, タブローアルゴリズムでは効率のよい Instance Retrieval が行えないという欠点が指摘されており, Description Logic Programs[5] や Disjunctive Logic Program を応用した KAON2[11] も提案されている.

これらの推論エンジンは基本的に一箇所に集められた知識を対象としており, 分散知識の統合を行うためには分散知識を一箇所に集め, それらが用いるオントロジの要素間の関係を外部から定義する必要がある. しかしこの手法を用いた, 以下の問題が生じる.

オントロジの巨大化 分散知識を一箇所に集めた場合, オントロジも一箇所に集めることになり, オントロジが巨大化する. OWL の計算量は Lite は EXPTIME, DL では NEXPTIME であり, 計算量が大きくなってしまう.

各オントロジ間の意味の不整合 同じ意味を持っていながら定義の異なるクラスに対して直接関係を定義した場合, 意味の不整合が発生する可能性がある. 例えば, 2 つの大学のオントロジ A と B があり, オントロジ A で以下のように修士学生のクラス a:Master が定義されているとする.

```
<owl:Class rdf:ID='Master'>
 <owl:subClassOf>
  <owl:Restriction>
   <owl:onProperty rdf:resource="#belongTo"/>
   <owl:someValuesFrom rdf:resource="#Lab"/>
  </owl:Restriction>
 </owl:subClassOf>
</owl:Class>
```

オントロジ B にも修士学生のクラス $b : Master$ があり, $a : Master \equiv b : Master$ という関係を記述したとする. しかし, もしオントロジ B では修士学生が必ずしも研究室 (Lab) に所属する必要がない場合でも, 修士学生は必ず研究室に所属することになってしまい, 意味の不整合が生じる.

誤った知識による誤った推論 知識を一箇所に集めることにより推論時に各知識が相互に影響しあう. そのため誤った知識が統合されれば, その他の知識に対して誤った推論を行う可能性がある. 例えば, 知識 A では $hasWife(Taro, Hanako)$ という正しい 2 項関係, 知識 B では $hasWife(Taro, Sachiko)$ という誤った 2 項関係が記述されている場合, もしプロパティ $hasWife$ が関数型プロパティであると定義されているならば $Hanako \equiv Sachiko$ と誤って推論されてしまう.

関係データベースや RDF データベースの統合の非効率 RDF/OWL で記述された知識しか直接処理することができず, 他のストレージシステムに蓄えられた知識を統合するためには RDF/OWL 形式で出力する必要がある.

本研究ではこれら 4 つの課題に対して, 解決策を与える手法を提案する.

3 アプローチ

3.1 概要

提案手法 (図 1) では以下の 2 つの方法を組み合わせることにより分散知識を統合し知識獲得を実現する.

インターフェースオントロジ 提案手法では要素間の直接の関係定義を避けるために新たに抽象的なク

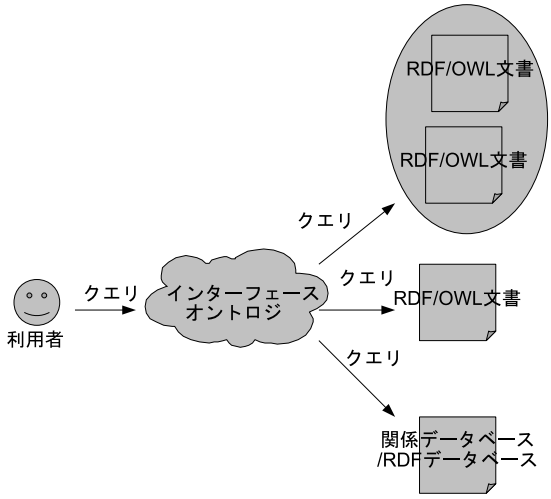

図 1: クエリ変換を用いた知識獲得法の概要

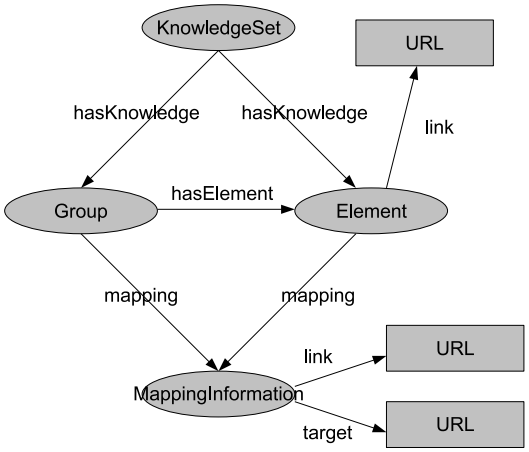

図 2: 統合知識オントロジ

ラス, プロパティを定義し, その部分集合として各オントロジのクラスやプロパティを定義する. 各オントロジの要素は他のオントロジとは直接関係を定義する必要がなく, 意味の不整合は発生しない. 例えば, 前節の修士学生の例において抽象的なクラス $c : Master$ を定義し, $a : Master$, $b : Master$ をその部分集合とする. こうすることにより, $a : Master$ と $b : Master$ の関係を定義することなく修士学生クラスを統合することができる. 本論文ではこのように抽象的なクラスやプロパティを定義したオントロジをインターフェースオントロジと呼ぶものとする.

提案手法では分散知識もしくは関連の強い分散知識の集合が用いる複数のオントロジに対し, 全体で一つのインターフェースオントロジを, 知識を統合する情報提供者が記述する. またインターフェースオントロジと各オントロジの要素間の関係の記述をマッピング情報と呼び, 同様に情報提供者が記述するものとする.

クエリ変換による分散推論 統合した知識から知識獲得をしようとする利用者は, インターフェースオントロジを用いたクエリ文 (RDQL[14]) を記述する. 与えられたクエリ文はマッピング情報を利用して各分散知識やその集合で用いられるオントロジを用いたクエリ文へと変換される. クエリ処理に必要な推論は各分散知識やその集合ごとに行われるため, オントロジの大きさは変わらず, 誤った知識が他に影響を与えることはない.

これら二つの方法を応用することにより関係データベースや RDF データベースから直接知識を獲得することができる. 詳しくは次節で述べる.

3.2 統合知識の記述

どの知識を統合するかは情報提供者が統合知識オントロジを用いて記述する. ここで統合知識オントロジを図 2 に示す. 統合する知識全体は KnowledgeSet クラス, 統合する分散知識の集合は Group クラス, 分散知識は Element クラスで表される. KnowledgeSet のインスタンスは hasKnowledge プロパティにより, Group か Element のインスタンスを持ち, Group のインスタンスは hasElement プロパティにより複数の Element のインスタンスを持つ. Element のインスタンスは一つの RDF/OWL 文書に対応し link プロパティによって実際の URL が指定されている. Group または Group に属さない Element のインスタンスはマッピング情報として MappingInformation のインスタンスを持つ. MappingInformation のインスタンスは link プロパティとしてマッピング情報を定義した RDF/OWL 文書の URL を, target プロパティとして分散知識やその集合が用いるオントロジの URL を持つ.

3.3 マッピング情報の記述

マッピング情報は以下の変換規則を用いて記述される. 変換規則は 5 つあり, インターフェースオントロジのクラス $C1$ に対し, 変換先のクラス $C2$, プロパティ $P1$ から $P2$, インスタンス $I1$ から $I2$ への変換を表している. $X1, \cdots, X4$, $Y1, \cdots, Y4$ は変換先の

適当な要素名を表す．変換規則 2, 3 では $C1$ の部分集合として変換先のクラスもしくは owl:Restriction の積, 和を定義する．そのため $C2$ は無名クラスとなり URI を持たない．

規則 *1*
C2 owl:subClassOf C1

規則 *2*
C2 owl:subClassOf C1
owl:interSectionOf(
クラス名;*
owl:Restriction(owl:hasValue X1;
owl:onProperty Y1);*
owl:Restriction(owl:someValuesFrom X2;
owl:onProperty Y2);*
)

規則 *3*
C2 owl:subClassOf C1
owl:unionOf (
クラス名;*
owl:Restriction(owl:hasValue X1;
owl:onProperty Y1);*
owl:Restriction(owl:someValuesFrom X2;
owl:onProperty Y2);*
owl:interSectionOf (
クラス名;*
owl:Restriction(owl:hasValue X3;
owl:onProperty Y3);*
owl:Restriction(owl:someValuesFrom X4;
owl:onProperty Y4);*
)*)

規則 *4*
P2 owl:subPropertyOf P1

規則 *5*
I1 owl:sameAs I2

3.4 クエリ変換

インターフェースオントロジを用いたクエリ文を, 分散知識やその集合に用いられるオントロジへのクエリ文へと, マッピング情報を用いて変換する．クエリ変換は以下の処理を行う．

1. クエリ文中のクラス名に対して, マッピング情報中の変換規則 1-3 まで適用可能かどうか調べ, 適用可能な場合, クエリ文の該当部分を置き換える．
2. クエリ文中のプロパティ名に対して, 変換規則 4 に対して同様に調べる．
3. クエリ文中のインスタンス名に対して, 変換規則 5 に対して同様に調べる．

クエリ文中にマッピング情報または変換先のオントロジ中に定義されていないクラス名もしくはプロパティ名が出てきた場合は, その変換先でのクエリ処理を行わない．また変換規則 3 によりクエリ文中に or 条件がでてくる場合がある．しかし今回利用した RDQL では or 条件を表現できないため, 実装上では複数のクエリを作成, クエリ処理を複数回行うことで対処した．

4 他のストレージシステムへの応用

4.1 RDF データベース

RDF データを格納する RDF データベースとして Sesame[12] がある．RDF データベースは RDF データへのクエリを高速に処理することができる．しかし, 分散知識を一箇所に集める手法ではデータベースに格納された RDF データを直接統合することができず, 効率的ではない．提案手法を応用することによって, RDF データベースに対してマッピング情報を記述することにより, RDF/OWL 文書の場合と同様にクエリ変換を行うことができ, RDF データベース上で高速なクエリ処理を行うことができる．

RDF データベースにアクセスするためには新たな情報が必要となるため, 統合知識オントロジに新たなクラスを定義する．図 2 の Element クラスの子クラスとして RDFDBElement を, MappingInformation の子クラスとして RDFDBMappingInformation をそれぞれ定義する．RDFDBElement のインスタンスは RDFDBMappingInformation のインスタンスを持ち, RDFDBMappingInformation のインスタンスはプロパティとしてマッピング情報, サーバーの URL, コネクションドライバ, ユーザー名, パスワードを持つ．

4.2 関係データベース

関係データベースから知識獲得するために RDQL から SQL への変換を行う必要がある．変換では SQL のテーブル, 属性, 列に対して RDF/OWL のクラス, プロパティ, インスタンスに対応させる．外部キーとして指定された列はオブジェクトプロパティ, それ以外はデータタイププロパティと対応する．また, 関係データベースでは概念を URI で表すという考え方がない．そのため, { 関係データベースの URL}+{ テーブル名 }, { コラム名 } それぞれをクラスとプロパティの URI とし, インスタンス名は { 関係データ

	大学 A	大学 B	大学 C	大学 D
クラス	30	29	35	9
プロパティ	3	2	2	0
インスタンス	11686	7205	4749	5068

表 1: 実験データ

ベースの URL}+{ テーブル名 }+{ 主キーの値 } をもって URI とする.

XML スキーマデータ型 [3] と変換規則を組み合わせることにより WHERE 節の探索条件中に不等号などの複雑な条件を記述することができる.

RDF データベースと同様に統合知識オントロジに新たなクラスを定義する. 図 2 の Element クラスの子クラスとして RDBElement を, MappingInformation の子クラスとして RDBMappingInformation を導入する. RDBMappingInformation のインスタンスはプロパティとしてサーバーの URL, コネクションドライバ, マッピング情報, ユーザー名, パスワードを持つ.

5 実験

本節では提案手法を用いることによりクエリ処理に必要な計算時間がどれだけ削減されるか, 実験にて確かめた.

5.1 実験環境

実験環境は以下の通りである.

- 実験マシン: Pentium4 3GHz, 1MB, Linux 2.4.27, Java VM 1.5.0_04.
- 実験データ: 東京 4 大学連合に参加する各大学の職員, 学生に関する情報を各 HP から抽出し, それをもとにオントロジを作成, 知識の生成を行った. また 4 大学の知識に対して一つのインターフェースオントロジと各知識の用いたオントロジに対するマッピング情報を準備した. 4 つの大学の知識の大きさを表 1 に示す.

また, 計算量を計るため実験では並行処理は行わず, 単一プロセスで行うものとする.

5.2 実験

学生や職員に関する知識獲得のための 10 個のクエリを用意し, それらの処理にかかる時間の平均をもって計測時間とした. 提案手法を用いた場合のデータ読み込み (読み込み), クエリ変換 (変換), 推論のそれぞれの計算時間と, インターフェースオントロジ, 知識, マッピング情報を一箇所に集めてクエリ処理を行った場合 (集中処理) のデータ読み込みと推論それぞれの計算時間を計測した. また, 実験では推論エンジンがデータ読み取り後, 最初のクエリ処理に必要な時間を計測し, 推論結果の再利用などによる推論の最適化など [7] を考慮しないものとする. 実験は Pellet および KAON2 の二つの推論エンジンそれぞれを利用して行った. ただし, KAON2 では今回用いたオントロジの一部がサポートされていなかったためその部分を取り除いて実験を行っている. 実験データとして用いた知識は独立に記述され, 知識間で同一の URI は用いていないため, 提案手法と集中処理で同じ結果が得られた.

	読み込み	変換	推論	合計
大学 A	2959	34	113766	116759
大学 B	3123	46	1442	4611
大学 C	1009	46	698	1753
大学 D	598	12	730	1347
提案手法	7689	138	116636	117246
集中推論	6250		1150723	1156973

表 2: Pellet を用いた実験結果 (単位: ms)

	読み込み	変換	推論	合計
大学 A	1355	29	111	1495
大学 B	953	29	144	1126
大学 C	621	32	55	708
大学 D	384	12	15	411
提案手法	3313	102	325	3740
集中推論	5648		747	6395

表 3: KAON2 を用いた実験結果 (単位: ms)

Pellet, KAON2 それぞれの実験結果を表 2, 表 3 に示す. 提案手法を用いることによりクエリ処理全体としてそれぞれ約 90%, 40% の計算時間が削減されていることが確認できる. 各クエリごとに見た場合, Pellet を用いた場合の特徴としてクエリ処理にかかる計算時間が大きくなるほど削減率が大きくなり, KAON2 を用いた場合では平均して計算時間が削減されていることが確認された.

6 関連研究

P2P 環境にて, クエリ変換を用いそれぞれ独立なオントロジを持つピアから知識を獲得する方法が [9] にて提案されている. [9] では提案手法と異なり多数の独立したピアを接続することを目的にしている. そのため OWL よりも記述力の低い独自言語を用いており, より単純なクエリ変換しか行わない. またピアに対するグループ分けなどもなく, 各ピアが相手のピアそれぞれへのクエリ変換を行うといった違いがある.

提案手法ではマッピング情報の記述法については特に規定していないが, 自動オントロジマッピングの手法 [4] と組み合わせることにより, 簡略化できると予想される.

7 まとめ

本論文ではセマンティック Web 上で分散知識や RDF データベース, 関係データベースに蓄えられた知識を統合し知識獲得する方法を提案した. 提案手法ではインターフェースオントロジとクエリ変換を用いた分散推論により, 分散知識を一箇所に集める手法の課題を解決することができる. 提案手法は分散知識の全てを知ることにより新たな知識が推論できる場合, 適切ではない. しかし Web においては分散知識同士のつながりが疎であることが多くなることが予想されるため, 有用であると考えられる. また, 提案手法を用いることによりクエリ処理に必要な計算量が減ることを実験を通して確認した.

今後の課題としてマッピング情報の自動生成などが挙げられる.

参考文献

[1] F. Baader, D. Calvanese, D. McGuinness, D. Nardi, and P. Patel-Schneider, editors. *The Description Logic Handbook*. Cambridge University Press, 2002.

[2] S. Bechhofer, F. Harmelen, J. Hendler, I. Horrocks, D. McGuinness, P. Patel-Schneider, et al. OWL Web Ontology Language Reference, W3C Recommendation, http://www.w3.org/TR/owl-ref/.

[3] J. J. Carroll, J. Z. Pan, XML Schema Datatypes in RDF and OWL, W3C Draft, http://www.w3.org/2001/sw/BestPractices/XSCH/xsch-sw/.

[4] M. Ehrig and S. Staab, QOM: Quick Ontology Mapping, *Proc. of the International Semantic Web Conference 2004*,pp. 683-697, Hiroshima, Japan, 2004.

[5] B. N. Grosof, I. Horrocks, R. Volz and S. Decker, "Description logic programs: Combining logic programs with description logics," *Proc. of the World Wide Web Conference 2003*,pp. 48-57, Budapest, Hungary, 2003.

[6] I. Horrocks and P. F. Patel-Schneider. Reducing OWL entailment to description logic satisfiability. *Proc. of the Internaitonal Semantic Web Conference 2003*, Florida, USA, 2003, 17-29.

[7] V. Haarslev and R. Moller, Optimization strategies for instance retrieval, *Proc. of the International Workshop on Description Logics 2002*,pp. 83-90, Toulouse, France, 2002.

[8] F. Manola, and E. Miller, Editors. RDF Primer, W3C Recommendation, http://www.w3.org/TR/rdf-primer/.

[9] M. C. Rousset, Small Can Be Beautiful in the Semantic Web, *Proc. of the nternaitonal Semantic Web Conference 2004*,pp. 6-16, Hiroshima, Japan, 2004.

[10] Google, http://www.google.co.jp/.

[11] KAON2, http://kaon2.semanticweb.org/.

[12] openRDF.org, http://www.openrdf.org/.

[13] Pellet OWL Reasoner, http://www.mindswap.org/2003/pellet/index.shtml.

[14] RDQL - A Query Language for RDF, http://www.w3.org/Submission/2004/SUBM-RDQL-20040109/.