어휘의 공리화를 이용한 Web Ontology 추론 시스템의 설계 및 구현

하영국[0] 손주찬 함호상
한국전자통신연구원 인터넷컴퓨팅연구부
(ygha[0], jcsohn, hsham)@etri.re.kr

Design and Implementation of Web Ontology Inference System Using Axiomatisation

Young-Guk Ha[0] Joo-Chan Sohn Ho-Sang Ham
Internet Computing Department, ETRI

요 약

최근 차세대 Web 기술로서 Semantic Web이 주목 받고 있다. Semantic Web에서는 Web상에 존재하는 문서에 Web Resource들에 대한 Ontology를 기반으로 Semantic Annotation을 하고 Ontology 추론 Agent를 통하여 의미 기반으로 Web을 검색할 수 있도록 해준다. 이와 같은 Semantic Web 기술의 핵심 요소는 Web Ontology이며 W3C에서는 이를 표현할 수 있는 표준 언어로서 RDF기반의 OWL(Web Ontology Language) 명세를 제정하고 있다. 따라서 표준 Web Ontology 언어인 OWL을 위한 추론 시스템은 Semantic Web 검색 Agent의 구현을 위한 필수적인 기반 기술이라 할 수 있으나 아직 그 개발이 미비한 상태이다. OWL 추론 시스템을 구현하기 위해서는 OWL의 이론적인 기반을 제공하는 DL(Description Logic)을 추론할 수 있는 엔진을 사용하는 것이 한가지 방법이 될 수 있으나 OWL이 Rule과 같은 DL의 범주를 벗어나는 Vocabulary를 지원하는 언어로 확장되는 경우에 이를 처리하기가 어렵다. 또 다른 방법으로서 Logic Programming을 통하여 OWL 언어의 Semantic을 기술하고 정리 증명(Theorem Proving)을 통하여 Ontology를 추론하는 공리화(Axiomatisation) 기법이 있는데 이러한 방법의 장점은 기반이 되는 Logic의 범주 내에서 새로운 언어를 위한 Vocabulary의 확장이 용이하다는 점이다. 본 논문에서는 Axiomatisation 방법을 이용하여 OWL로 기술된 Ontology를 추론할 수 있는 시스템의 설계 및 구현에 대해 설명하기로 한다.

1. 서론

최근 들어 차세대 Web 기술로서 의미 기반의 검색을 제공하는 Semantic Web이 주목 받고 있다. 기존의 HTML기반의 Web 기술이 인간이 이해할 수 있는 내용을 통하여 사용자와 대화하기 위한 Web이었다면 Semantic Web은 인간이 아닌 컴퓨터가 이해할 수 있는 의미를 기술하여 사용자에게 보다 자동화된 검색 및 서비스를 제공하는 기술이다. Semantic Web에서는 Web상에 존재하는 문서에 Ontology를 이용하여 Semantic Tagging을 하고 Ontology를 추론할 수 있는 지능형 Agent를 통하여 의미 기반의 질의를 입력 받아 효과적으로 Web을 검색할 수 있도록 한다. 이러한 Semantic 검색을 가능하도록 하는 핵심 기술은 Web Ontology로서 W3C의 RDF(Resource Description Framework)[6]를 기반으로 정의된 Ontology 언어로써 표현된다. 현재 표준 Ontology 언어로는 DAML(DARPA Agent Markup Language)과 OIL(Ontology Interface Layer)이 합쳐진 DAML+OIL 및 W3C의 OWL(Web Ontology Language)[6] 등이 있으며 Semantic Web Agent를 구현하기 위해서는 이러한 Web Ontology 언어를 추론할 수 있는 엔진이 필요하게 된다. 현재 제정되어 있는 표준 Web Ontology 언어들은 기본적으로 DL(Description Logic)[1]을 그 이론적인 기반으로 하고 있다. DL은 명확한 Semantic을 제공하는 전통적인 Logic 기반의 지식 표현 방법과 개념간의 Network 형태로 다양한 표현력을 제공하는 Frame이나 Semantic Network의 장점을 혼합한 지식 표현 방법으로 추론을 위하여 기본적으로 Tree기반의 추론 기법인 Tableaux 알고리즘을 사용한다. 그러나 Tableaux 알고리즘과 같은 DL 추론 방법의 문제점은 Web Ontology 언어가 DL의 범주를 벗어나는 표현력을 제공하게 되는 경우(예를 들어 일반적인 Rule을 표현할 수 있는 새로운 어휘로 확장되는 경우) 이를 처리할 수 없다는 한계를 가지고 있다. Web Ontology를 추론하기 위한 또 다른 기법으로는 Axiomatisation[3][5]을 이용하는 방법이 있다. Axiomatisation 방법은 FOL(First Order Logic)[1][3]과 같은 Rule을 표현할 수 있는 Logic을 이용하여 Web Ontology 언어의 어휘에 대한 Formal Semantics를 공리(Axiom)로 기술하고 정리 증명(Theorem Proving) 기법을 사용하여 Ontology를 추론하는 방법이다. 이러한 방법의 주된 장점은 기반이 되는 Logic의 범주 내에서 새로운 어휘에 대한 Logic 프로그래밍을 통해 Ontology 언어의 확장이 용이하게 이루어질 수 있으므로 추론 엔진 자체를 변경할 필요가 없다는 것이다.

본 논문에서는 우선 Semantic Web과 Web Ontology의 추론 기술 전반에 대하여 살펴보고, Axiomatisation 기반의 OWL Ontology 추론 시스템의 구조 및 그 구현 내용에 대하여 설명하기로 한다.

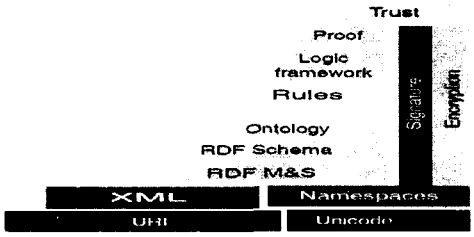
2. 관련 기술

Semantic Web의 핵심 구성 요소는 Web Resource들간의 의미와 상관 관계를 기술하는 Web Ontology[6]이다. Web

Ontology는 표준화된 RDF기반의 Ontology 언어로 기술되며 Resource의 의미와 그들간의 관계를 추론하기 위해서는 적합한 추론 방법이 필요하게 된다. 본 장에서는 OWL 추론 시스템을 구현하기 위한 관련 기술에 대해서 설명하기로 한다.

2.1 Semantic Web과 Web Ontology

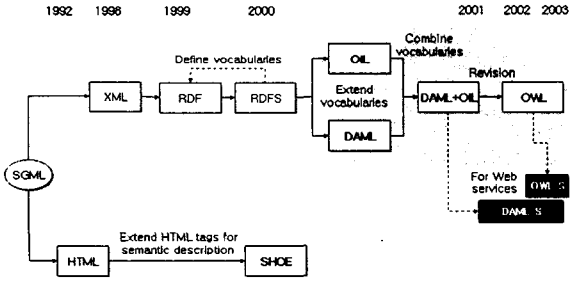
Ontology는 특정 개념에 대한 의미를 표현하기 위해 개념과 개념들 간의 관계를 이용한 지식 표현 방식이다. Web Ontology란 Web상에 존재하는 Resource들에 대한 Ontology로서 W3C 표준인 RDF를 기반으로 정의된 Ontology 언어를 통해 기술된다. <그림 1>은 Semantic Web의 계층 구조를 나타낸다. 현재는 표준화 작업을 통해 Ontology 계층에 대한 명세가 진행되고 있으며 Rule을 표현하기 위한 표준 작업이 시작되고 있는 단계이다. Semantic Web은 다양한 지능형 Agent를 이용하여 Web Ontology를 기반으로 Web 문서상에 포함된 Semantic Annotation에 대한 추론을 통해 사용자가 원하는 정보만을 보다 정확하게 찾아내는 것을 가능하게 해준다.



<그림 1> Semantic Web 계층 구조

2.2 Web Ontology 언어

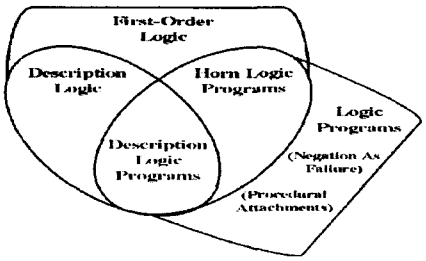
잘 알려진 Web Ontology 언어로는 DARPA 프로젝트의 DAML, 유럽의 OntoKnowledge 프로젝트의 산물인 OIL, HTML에 Semantic Tag을 확장한 SHOE(Simple HTML Ontology Extension) 등이 있으며 DAML과 OIL의 기능을 합쳐서 만들어진 DAML+OIL이 있다. 현재는 W3C에 의해 DAML+OIL을 Revision한 OWL에 대한 표준화가 진행되고 있다(Web Ontology 언어에서는 일반적으로 개념을 Class로 관계를 Property로 표현함). 또한 Web Service에 대한 Ontology를 기술하기 위한 DAML-S(DAML Service)와 최근 발표된 OWL-S(OWL Service) 등이 있다. <그림 2>는 Web 표준 언어와 Ontology 언어들 간의 상관 관계를 보여준다.



<그림 2> Web Ontology 언어 계통도

2.3 Logic Program과 Ontology 추론

<그림 3>은 Logic 기반 지식 표현 방법들 간의 상호 관계를 나타내는 그림이다. 그림과 같이 DL은 FOL의 Decidable 한 부분 집합임을 알 수 있으며, 다시 말해서 모든 DL은 FOL을 이용해서 표현이 가능함을 알 수 있다. 즉 FOL을 이용하여 DL을 기반으로 하는 Web Ontology 언어의 어휘에 대한 Logic Programming[1]을 하고 Theorem Proving[5]을 통해 Ontology를 추론할 수 있다는 것이다. 또한 DL의 범주를 벗어나는 어휘에 대한 확장 및 추론도 FOL이 표현 가능한 범위 내에서 가능하게 된다.



<그림 3> Logic기반 지식 표현 방법

3. OWL Axiomatisation

FOL을 이용하여 DL의 Operator에 대한 의미를 표현하면 <표 1>과 같다. 이러한 DL Semantic[1]은 OWL의 어휘에 대한 Logic Programming을 하기 위하여 기본적인 구성 요소로서 이용된다.

DL	FOL
$a : C$	$C(a)$
$\langle a, b \rangle : P$	$P(a, b)$
$C \sqsubseteq D$	$\forall x.C(x) \rightarrow D(x)$
$P^+ \sqsubseteq P$	$\forall x, y, z.(P(x, y) \wedge P(y, z)) \rightarrow P(x, z)$
$\top \sqsubseteq \leqslant 1\, P$	$\forall x, y, z.(P(x, y) \wedge P(x, z)) \rightarrow y = z$
$P \equiv Q$	$\forall x, y.P(x, y) \iff Q(y, x)$
$C_1 \sqcap \ldots \sqcap C_n$	$C_1(x) \wedge \ldots \wedge C_n(x)$
$C_1 \sqcup \ldots \sqcap C_n$	$C_1(x) \vee \ldots \vee C_n(x)$
$\neg C$	$\neg C(x)$
$\{a_1, \ldots, a_n\}$	$x = a_1 \vee \ldots \vee x = a_n$
$\exists P.C$	$\exists y.(P(x, y) \wedge C(y))$
$\forall P.C$	$\forall y..(P(x, y) \rightarrow C(y))$
$\geqslant n\, P.C$	$\exists y_1, \ldots, y_n. \bigwedge_{1 \leqslant i \leqslant n} (P(x, y_i) \wedge C(y_i)) \wedge \bigwedge_{1 \leqslant i < n, i < j \leqslant n} y_i \neq y_j$
$\leqslant (n-1)\, P.C$	$\forall y_1, \ldots, y_n.(\bigwedge_{1 \leqslant i \leqslant n} (P(x, y_i) \wedge C(y_i))) \rightarrow (\bigvee_{1 \leqslant i < n, i < j \leqslant n} y_i = y_j)$

<표 1> FOL기반의 DL 표현

<표 2>와 <표 3>은 각각 OWL의 Class Constructor 어휘와 Axiom 어휘에 대한 DL기반의 Semantic을 표현하는데 각각의 DL Operator는 최종적으로 <표 1>과 같이 FOL 표현으로 Axiomatise되어야 한다. 다음의 예는 "subClassOf" 어휘에 대한 FOL 표현을 보여준다. 이를 위해 FOL 표현 언어인 KIF(Knowledge Interchange Format)[5]을 사용하였다.

FOL Description for CSUPER ⊆ CSUB :
∀x.CSUPER(x) → CSUB(x)

KIF Description :
(=> (PropertyValue rdfs:subClassOf ?CSUB ?CSUPER)
(and (Type ?CSUB owl:Class) (Type ?CSUPER owl:Class)
(forall (?X) (=> (Type ?X ?CSUB) (Type ?X ?CSUPER)))))

예제의 subClassOf(CSUPER ⊆ CSUB)에 대한 FOL 표현의 의미는 Class CSUPER에 포함된 모든 Property x는 Class

CSUB에도 포함되어 있다는 것이다(Preorder 표기법을 사용하는 KIF에서도 동일한 의미임).

Constructor	DL Syntax	Example	(Modal Syntax)
intersectionOf	$C_1 \sqcap \ldots \sqcap C_n$	Human ⊓ Male	$C_1 \wedge \ldots \wedge C_n$
unionOf	$C_1 \sqcup \ldots \sqcup C_n$	Doctor ⊔ Lawyer	$C_1 \vee \ldots \vee C_n$
complementOf	$\neg C$	¬Male	$\neg C$
oneOf	$\{x_1 \ldots x_n\}$	{john, mary}	$x_1 \vee \ldots \vee x_n$
allValuesFrom	$\forall P.C$	∀hasChild.Doctor	$[P]C$
someValuesFrom	$\exists P.C$	∃hasChild.Lawyer	$\langle P\rangle C$
maxCardinality	$\leqslant nP$	⩽1hasChild	$[P]_{n+1}$
minCardinality	$\geqslant nP$	⩾2hasChild	$\langle P\rangle_n$

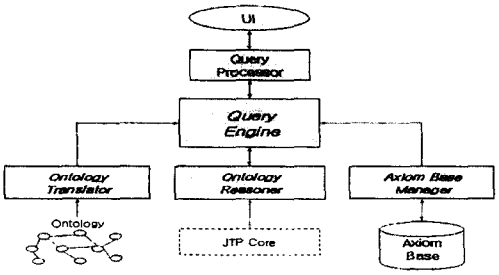
<표 2> DL기반의 OWL Class Constructor 표현

Axiom	DL Syntax	Example
subClassOf	$C_1 \sqsubset C_2$	Human ⊏ Animal ⊓ Biped
equivalentClass	$C_1 = C_2$	Man ≡ Human ⊓ Male
disjointWith	$C_1 \sqsubset \neg C_2$	Male ⊏ ¬Female
sameIndividualAs	$\{x_1\} = \{x_2\}$	{President_Bush} = {G_W_Bush}
differentFrom	$\{x_1\} \sqsubset \neg\{x_2\}$	{john} ⊏ ¬{peter}
subPropertyOf	$P_1 \sqsubset P_2$	hasDaughter ⊏ hasChild
equivalentProperty	$P_1 = P_2$	cost ≡ price
inverseOf	$P_1 \equiv P_2^-$	hasChild ≡ hasParent⁻
transitiveProperty	$P^+ \sqsubset P$	ancestor⁺ ⊏ ancestor
functionalProperty	$\top \sqsubset \leqslant 1P$	⊤ ⊏ ⩽1hasMother
inverseFunctionalProperty	$\top \sqsubset \leqslant 1P^-$	⊤ ⊏ ⩽1hasSSN

<표 3> DL기반의 OWL Axiom 표현

4. OWL 추론 시스템

OWL 추론 시스템은 <그림 4>와 같이 크게 Query Engine, Axiom Base Manager, Ontology Translator, Ontology Reasoner, Query Processor 및 사용자 인터페이스로 이루어져 있다. 본 장에서는 각각의 구성 요소에 대해 설명하기로 한다.



<그림 4> OWL 추론 시스템 구조

4.1 Query Engine

Query Engine은 OWL 추론 시스템의 주 모듈로서 각각의 구성 모듈을 호출하여 전체적인 추론 과정을 수행하는 기능을 제공한다.

4.2 Axiom Base와 Ontology Translation

Axiom Base는 각각의 OWL 언어의 어휘에 대한 Logic Program을 저장하고 있는 Knowledge Base로서 XML 또는 RDB 형태의 저장소를 이용하여 구현한다. Axiom Base Manager는 Axiom Base를 접근하기 위한 모듈로서 Query Engine으로부터 호출되어 필요한 Axiom Base를 읽고 새로운 Axiom을 추가하거나 또는 기존의 Axiom을 삭제하는 등의 관리 기능을 수행한다.

Ontology Translator는 Web Ontology를 입력으로 받아 정리 증명이 가능한 정규형(Conjunctive Normal Form)으로 변환하여 주는 역할을 수행한다.

4.2 Ontology Reasoning

Ontology Reasoner는 Axiom Base 및 CNF로 정규화된 Ontology를 이용하여 주어진 Query를 입력 받아 추론을 수행하는 모듈이다. 실제 추론 과정은 OWL 언어에 대한 Axiom Base와 정규화된 Ontology에 대한 Theorem Proving을 통하여 이루어 지는데 본 시스템의 구현에서는 Stanford 대학의 Knowledge System Lab에서 개발한 JTP(Java Theorem Prover)[4]를 사용하였다.

4.3 Ontology Query Processing

Ontology Query Processor는 사용자로부터 특정 문법의 Query를 입력 받아 Ontology Reasoner가 처리할 수 있는 정규화된 문법으로 변환해주는 역할을 수행한다. 일반적으로 사용자 Query 문법의 경우 Syntactic Sugar로서 기억하기 쉽고 사용이 편리한 문법을 채용하고 있는데 본 시스템에서는 RDQL(RDF Data Query Language)[2]을 이용한다. RDQL은 RDF Data를 추론하기 위한 Query 언어로서 기본적으로 "Select", "From", "Where" 절을 포함하는 SQL의 형태를 취하고 있다. 다음은 단순한 RDQL 문장의 예로서 "http://localhost/ontology/owl_ex" Ontology상의 모든 18세 이상의 성인을 추론하는 Query를 보여준다.

```
SELECT x, ?y
FROM   <http://localhost/ontology/owl_ex>
WHERE  (?x, <rdf:type>, <owl_ex:Person>),
       (?x, <owl_ex:age>, ?y)
AND    ?y >= 18
USING  ont_ex FOR <http://localhost/ontology/owl_ex#>
```

5. 결론

본 논문에서는 Semantic Web의 핵심 기술인 Web Ontology를 추론하기 위하여 Axiomatisation을 이용한 시스템의 설계 및 구현에 대하여 살펴보았다. 이러한 방법이 표준 Web Ontology 언어의 확장에 대한 유연성을 제공한다는 장점을 제공하기는 하지만 어휘의 Semantic에 대한 Logic Programming이 올바르다는 것을 증명하기 어렵고 Undecidable한 FOL의 부분 집합을 사용함으로써 발생하는 계산상의 Overhead 등의 문제점을 가지고 있는 것도 사실이다. 따라서 추후 효과적인 Web Ontology에 대한 추론을 위하여 다양한 최적화 방법을 연구 개발하여 적용할 계획이다.

참고 문헌

[1] Benjamin N. Grosof, Ian Horrocks, et al., "Description logic programs: Combining logic programs with description logic," WWW 2003, Budapest, May. 21. 2003.

[2] HP Labs, "RDQL - RDF Data Query Language," http://www.hpl.hp.com/semweb/rdql.htm.

[3] Ian Horrocks, Peter F. Patel-Schneider, "Three Theses of Representation in the Semantic Web," WWW2003, Budapest, May. 21. 2003.

[4] KSL Stanford Univ., "JTP: An Object-Oriented Modular Reasoning System," http://www.ksl.stanford.edu/software/JTP/.

[5] Richard Fikes, Deborah McGuinness, "An Axiomatic Semantics for RDF, RDF-S, and DAML+OIL," Dec. 18. 2001. http://www.w3.org/TR/daml+oil-axioms.

[6] Web-Ontology (WebOnt) Working Group, "WebOnt Tech Reports," http://www.w3.org/2001/sw/WebOnt/.