온톨로지 자동구축을 위한 OWL의 어휘와 구문 사용방법에 대한 이론적 연구

A Theoretical Study of Using Methods for OWL Vocabulary and Syntactics to Ontology Automatic Construction

서 휘(Whee Seo)*

〈 목 차 〉

초 록

본 연구에서는 선행 연구를 근거로 온톨로지에 대한 정의와 기능 그리고 종류에 대해 기술하였으며, 특히 온톨로지의 자동 구축을 위한 어휘인 OWL(Ontology Web Language)에 대하여 어휘와 구문 사용방법에 대하여 기술하였다. 또한 OWL의 구문 및 기능에 대한 사용법을 쉽게 익힐 수 있도록 클래스(Class), 속성(Property), 클래스 간의 관계, 속성 간의 관계 등에 대하여 각 구문에 대한 상세한 정의와 함께 기 구축된 Wine 온톨로지를 근거로 사용사례와 그 설명을 제시하였다.

주제어: 온톨로지, 웹온톨로지, 시맨틱웹

ABSTRACT

This paper deals with the definition, function and type of ontology based on precedent study. Particularly the paper describes a Using Methods for OWL vocabulary and syntactics to Ontology Automatic Construction. Also for easily learning the usage methods for OWL vocabulary and syntactics, it introduces a detailed definition for syntactics of Class, Property, Class relativeness, Property relativeness and presents a sample data and explanation based on Wine Ontology which have constructed.

Key Words: Ontology, OWL, Web Ontology, Semantic Web

* 창원전문대학 문헌정보과 조교수(drs733m@changwon-c.ac.kr)
• 접수일: 2006년 5월 29일 • 최초심사일: 2006년 6월 1일 • 최종심사일: 2006년 6월 22일

Ⅰ. 서 론

현재 인터넷은 차세대 웹인 시멘틱 웹(Semantic Web)으로 진입하는 단계에 와 있다. 그럼에도 불구하고 아직 인터넷의 어떠한 포털 검색엔진도 정보검색의 효율성을 보장하지 못하고 있다. 그 이유는 정보를 구축할 때 시간적 측면과 비용적 측면 때문에 검색의 실마리가 되는 키워드의 선정에 많은 노력을 기울일 수 없기 때문이다. 또한 정보를 탐색하는 과정에서도 일반 이용자들이 검색전략 구축의 미숙함을 드러내고 있으며, 특히 정보 구축 단계에서 사용한 키워드와 일치하는 탐색어휘를 선정하는데 많은 어려움을 겪고 있기 때문이다.

이와 같은 문제점 때문에 과거에도 시소러스와 같은 도구를 활용하여, 정보를 구축하는 과정에서의 키워드 선정과 정보를 탐색하는 과정에서의 탐색어휘 선정에 도움을 얻도록 했으나 그 노력에 비해 정보검색의 효율성은 미흡하다고 할 수 있다. 그 이유는 시소러스가 인간의 두뇌적 활동에 의해 용어간의 관계를 인식하는 기능을 가지고 있으나, 컴퓨터의 경우에는 용어간의 관계와 의미를 인식하여 추론의 기능까지 하는 기능을 갖지 못하고 있기 때문이다.

이와 같은 문제점은 시멘틱 웹의 근간이 되는 온톨로지(Ontology)를 도입하면 해결할 수 있을 것으로 판단되고 있다. 왜냐하면 온톨로지란 특정 용어에 대하여 인간의 두뇌적 활동 뿐만이 아니라 컴퓨터의 기계적 작용에 의해서 용어 간의 관계, 의미 그리고 차이점 등을 자동으로 인식하고 추론할 수 있는 능력을 갖추고 있기 때문이다. 따라서 본 연구에서는 선행논문을 근거로 온톨로지의 정의와 기능 그리고 종류에 대하여 기술할 것이다.

특히 온톨로지 자동 구축을 위해서 W3C에서 2004년에 권고안으로 채택한 OWL(Ontology Web Language)의 유형과 구문구조에 대하여 선행 연구를 근거로 제시할 것이다. 또한 OWL의 구문 및 기능에 대한 사용법을 쉽게 익힐 수 있도록 클래스(Class), 속성(Property), 클래스 간의 관계, 속성 간의 관계 등에 대하여 Wine 온톨로지를 근거로 상세히 정의하고, 사용사례와 그 설명을 제시할 것이다.

Ⅱ. 온톨로지의 정의

1. 온톨로지의 개념적 정의

온톨로지는 1967년 S. H. Mealy에 의해 전산학 분야에 도입되어 사용되었으며, 최근에는 인공지능(Artificial Intelligence) 분야에서 시멘틱 웹을 표현하는 중요한 언어로 각광받고 있다. 본래 온톨로지는 19세기 독일 철학자들에 의해 처음으로 사용된 단어로서 희랍어 'ontos(being)'와 'logos

(word)'에서 유래하며, "세상의 어떤 관점을 세계에 존재(being)하는 것들의 종류, 그 본성과 관계 특성 등을 설명하는 분류체계(Taxonomy)를 제공하는 것"이라 정의된다.[1][2]

온톨로지에 대하여 Sowa는 "온톨로지란 어떤 특정 영역(domain of interest)에서 존재하거나 존재할 수 있는 대상들의 범주(category)에 관한 연구"라고 정의한다.[3] 또한 Gruber는 "온톨로지는 도메인과 관련된 형식적(formal)이고 명시적(explicit)인 공유(Shared) 개념이다."라고 정의하고 있다.

이상과 같은 정의를 근거해 볼 때, 온톨로지는 특정 영역에서 어떤 대상들을 기계가 읽고 해석해서 처리할 수 있도록 하여 - 술어 논리와 같은 인공지능의 방법으로 - 지식의 형식적인(formal) 표현이 가능하다. 또한 온톨로지는 특정 대상들에 대한 개념 표현방법이 관련 분야 전문가들의 합의(shared)에 의한 것이며, 개념들 간의 유형 식별(명확한 분류)을 개념 사용에 대한 규칙을 근거로 - 제한적인 개념의 형태로 표현하므로 - 명시적인(explicit) 기술이 가능하다.

이와 같은 기능을 갖고 있는 온톨로지는 현재 효율적인 정보통합과 정보 검색 그리고 지식 관리 등의 기능이 필요한 전자학습, 디지털도서관, 지능형 에이젠트 그리고 정보검색 시스템 등의 분야에서 필요 정보에 대하여 연계적인 추출이 가능한 지능적인 추론 등에 활용된다. 그리고 최근에는 웹 정보에 대하여 인간과 응용 시스템과의 인터페이스 사이에서 역할을 할 수 있는 기능인 - 온톨로지가 본래부터 갖고 있는 - 공유된 지식과 공통된 해석의 수단이라는 차원에서 더욱 각광을 받고 있다.

즉 최근에 온톨로지가 이처럼 각광을 받고 있는 이유는 웹 기반의 지식 처리나 응용 프로그램 사이의 지식 공유, 재사용들을 가능하게 하는 핵심요소인 계층 분류(taxonomy)와 추론 규칙(inference rule)에 대한 정의가 온톨로지 내에 포함되어 있기 때문이다.[4] 계층 분류는 객체의 클래스와 하위 클래스와 그들 간의 관계를 정의하는 것을 의미하며, 추론 규칙은 프로그램이 새로운 사실을 인간의 도움이 없이 자동으로 추출하거나 제약 조건에 맞지 않는 오류를 찾아내는데 이용될 수 있는 규칙이다.

2. 온톨로지의 종류

앞의 장에서 온톨로지는 형식적 표현이 가능하며, 개념 사용 규칙을 근거로 명시적인 기술이 가

1) 日本情報處理開發協會, オントロジー工學に關する調査研究 : 大規模知識ベースに關する調査研究報告書 09-R 004. 동연구소, 1998. p.5.
2) 신효필, 지식기반(Knowledge Base)으로서의 온톨로지와 시맨틱 웹(Semantic Web), 정보처리 학회지, 제11권, 제2호(2004. 6), pp.111-112.
3) John F. Sowa. Semantic Networks. http://www.jfsowa.com/pubs/semnet.htm
4) 박사준, 시맨틱 웹에서 온톨로지를 이용한 전문가 지식 추출 모델, 중앙대학교대학원, 박사학위논문(2003), pp.7-8.

능하다고 하였다. 온톨로지가 형식적이란 의미는 규정된 용어들과 용어 사이의 관계를 기계가 이해할 수 있는 방법으로 표시한다는 의미이다. 이를 위해서 온톨로지에서의 지식은 클래스(Class), 관계(Relation), 함수(Function), 공리(Axiom), 인스턴스(Instance)의 다섯가지 요소를 이용하여 형식화된다.[5]

클래스는 일반적으로 개념어에 해당되며, 관계는 개념들을 규정하는 속성의 유형을 의미한다. 함수는 관계가 특정 값을 가질 때 성립되는 것이며, 공리는 논리의 전개나 추론의 근거가 되는 것으로 '참'으로 인정되는 문장을 의미한다. 그리고 인스턴스는 이와 같은 요소들이 결합되어진 실제의 값을 의미한다.

따라서 위의 다섯 가지 요소를 모두 갖춘 온톨로지를 형식적 온톨로지라고 하며, 또한 이와 같은 구성요소를 갖추어 형식화됨으로써 명시적인 온톨로지가 될 수 있는 것이다. 그러나 명시적 온톨로지의 형식성에는 정도에 차이가 있고, 형식성의 정도에 따라 규정된 의미도 다양하게 변화된다.

Uschold와 Gruninger는 온톨로지를 앞에 정의한 형식성에 기준하여 비형식적(highly informal) 온톨로지, 반비형식적(semi-informal) 온톨로지, 반형식적(semi-formal) 온톨로지, 형식적(rigorously formal) 온톨로지 등의 네가지 형태로 구분하고 있다.[6]

비형식적 온톨로지는 공리가 없는 용어의 집합(자연언어로 자유롭게 표현된 것)으로 용어집이나 시소러스가 해당된다. 반비형식적 온톨로지는 조직된 표현으로 모호성을 줄여 명확성을 상당히 높인 온톨로지이며, 반형식적 온톨로지는 조직된 표현을 인공적인 - 형식적 언어로 표현한 온톨로지를 의미한다. 형식적 온톨로지는 형식적 의미를 갖는 인공 언어로 속성의 법칙과 정리를 표현한 온톨로지를 의미한다.

또한 온톨로지는 일반화의 정도에 따라서 상위 수준(top-level) 온톨로지, 도메인(domain) 온톨로지, 과업(task) 온톨로지, 응용(application) 온톨로지로 구분된다.[7] 상위수준 온톨로지는 매우 일반적인 개념을 묘사하며, 도메인 온톨로지는 특정분야에 한정된 개념들을 제공하며 개념 규정 시 정제된 정의를 요구한다. 과업 온톨로지는 특정 과업 수행을 위한 개념을 기술하며 일반 또는 도메인 영역에서 사용하는 언어를 재사용하여 상위 수준 온톨로지에 도입해 사용한다. 응용 온톨로지는 특정 도메인과 과제 온톨로지 모두에 종속되는 개념을 묘사하는 것으로 온톨로지의 특수화라고 할 수 있다.

그 이외에 표현(Representation) 온톨로지, 일반(Generic) 온톨로지, 중개(Intermediate) 온톨로지 등이 있다. 표현 온톨로지는 개념화를 명시적으로 표현하며, 특정 영역에 관계없이 지식 표

5) Oscar Corcho, Mariano Fernandez-Lopez & Asuncion Gomez Perez, Onto Web Technical Road-map v1.0, IST Programme of the Commision of the European Communities as Project No. IST-2000-29243, pp.10-11.
6) Mike Uschold & Michael Gruninger, Ontologies : Principle, Methods and Applications, Edinburgh : Edinburgh University, 1996, pp.5-6.
7) 이현실, "온톨로지를 이용한 의학용어의 개념 모델링 사례분석 연구," 정보관리학회지, 제21권, 제3호(2004. 9), p.145.

현 형식의 기초가 되는 온톨로지로 프레임 온톨로지가 해당된다. 일반 온톨로지는 특정 영역과는 무관하게 일반적이고 기초적인 개념을 제공한다. 중개 온톨로지는 도메인의 일반적인 개념과 관계를 가지고 도메인 온톨로지와 일반 온톨로지 사이에서 인터페이스로 사용된다.

3. 온톨로지의 기능

온톨로지의 일반적 기능은 다음과 같다.[8] 첫째, 개념을 명확히 정의하고 있기 때문에 검색자 및 기타 사용자들에게 개념의 모호성을 줄여준다. 둘째, 색인 기능 제공으로 정보검색을 용이하게 해 줄 뿐만 아니라 자체 데이터베이스를 검색하는 노력을 줄여준다. 셋째, 자연언어로 질의를 해도 자동으로 적절한 용어를 인식하여 검색해주므로 정확성을 높여준다. 즉 사용자 질의어가 시스템 용어로 매핑이 가능하다. 온톨로지의 이와 같은 기능은 정보의 조직과 검색에 매우 중요한 역할을 담당하고 있기 때문에 정보검색 분야에서는 온톨로지를 시소러스(thesaurus)라고 부르기도 한다.

온톨로지에서의 지식 표현은 앞에서 설명한 바와 같이 클래스, 관계, 함수, 공리, 인스턴스의 5개 요소를 이용해 특정 분야의 전문가들의 관심, 목적, 관점을 형식화하는 방법으로 처리된다. 앞의 5개 요소에 의해 형식화된 온톨로지는 해당 분야에서 사용되는 어휘의 해석과 의미의 한계를 규정하므로, 인간들끼리의 이해의 공유와 커뮤니케이션, 기종이 다른 소프트웨어의 시스템 통합에 있어서의 상호운용성 제공, 특정 분야의 모델링과 목적에 따른 요구사항을 규격화하는 표준명세 규격 그리고 지식관리와 정보검색 분야에서 이용이 가능하다.[9][10][11]

본 논문에서는 지식관리와 정보검색 분야에서의 온톨로지의 적용 가능성에 대해서만 기술하기로 한다.

지식관리 분야에 적용이 가능한 온톨로지의 개념구조는 시멘틱 웹에서 데이터의 기계적 처리가 가능한 핵심적 역할을 수행하여 지식베이스의 일관성을 향상시킬 수 있는 방안으로 이용될 수 있다. 그 이유는 다음과 같다.

첫째, 온톨로지를 이용한 의미있는 주석은 지식의 식별, 수집, 개발에 있어서 전문가는 물론 일반인에게도 지식의 포착과 창조의 수단이 된다. 둘째, 온톨로지는 지식통합에 있어서 정보를 동일한 의미적 맥락에서 다른 정보와 연결시키는 개념적 일반화와 컴퓨터 메커니즘에 의해 정보에 접근하

8) 양재영 · 정현섭 · 최중민, "온톨로지를 이용한 상위레벨 웹 페이지 추천 에이전트," HCI 2002 학술대회, 정보과학회, pp.4-7.

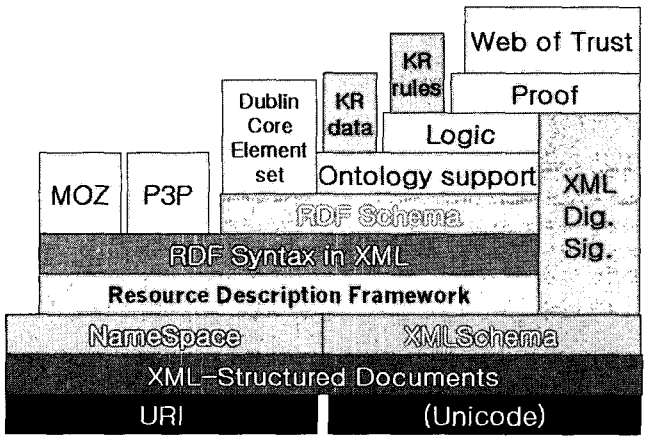
9) Fischer, Gerhard and Jonathan Ostwald, "Knowledge Management : Problems, Promises, Realities and Challenges," *IEEE Intelligent System*, January/February(2001) : pp.60-72.

10) Dieter Fensel et al. "On-To-Knowledge : Ontology-Based Tools for Knowledge Management," 2000.[cited 2004. 1. 29].(http://www.cs.vu.nl?~frankh/postsript/eBeW00.pdf)

11) Ying Ding, Dieter Fensel, Michel Klein and Borys Omelayenko, "The Semantic Web : Yet Another Hip ?," *Data & Knowledge Engineering*, 41(2002), pp.205-227.

고 해석할 수 있는 표현적 형식화를 제공한다. 셋째, 온톨로지를 이용해 개념화한 형식적 구조는 지식보급에 있어서 표준화되고 집중적인 접근체제를 갖출 수 있어 문제해결을 위한 실시간의 학습 환경을 제공할 수 있다. 따라서 온톨로지를 응용한 지식관리는 시멘틱 웹이나 자연언어의 기계적인 이해가 가능하도록 한다.

이상과 같은 이유 때문에 〈그림 1〉의 시멘틱 웹의 계층적 구조에서 나타난 바와 같이 온톨로지는 RDF 스키마(Schema)와 Logic 사이에서 'Ontology Support'의 역할을 하는 것으로 표현되는 것이다.



〈그림 1〉 시멘틱 웹의 계층적 구조[12]

또한 정보검색 분야에 있어서 온톨로지의 적용은 언어학적 개념체계를 갖는 용어학과 결합해 인간과 컴퓨터의 의사소통에 효과적이고 유연한 자원을 제공할 수 있으므로 키워드 매칭 방식에 의한 검색시스템보다 서비스의 질적 수준의 향상이란 효과를 얻을 수 있다. 왜냐하면 온톨로지가 갖고 있는 개념적 접근 및 개념간의 관계에 대한 정의의 명확성과 다양성이란 특성이 재현율과 정도율에 관계하는 질의 형성을 용이하게 할 수 있기 때문이다.

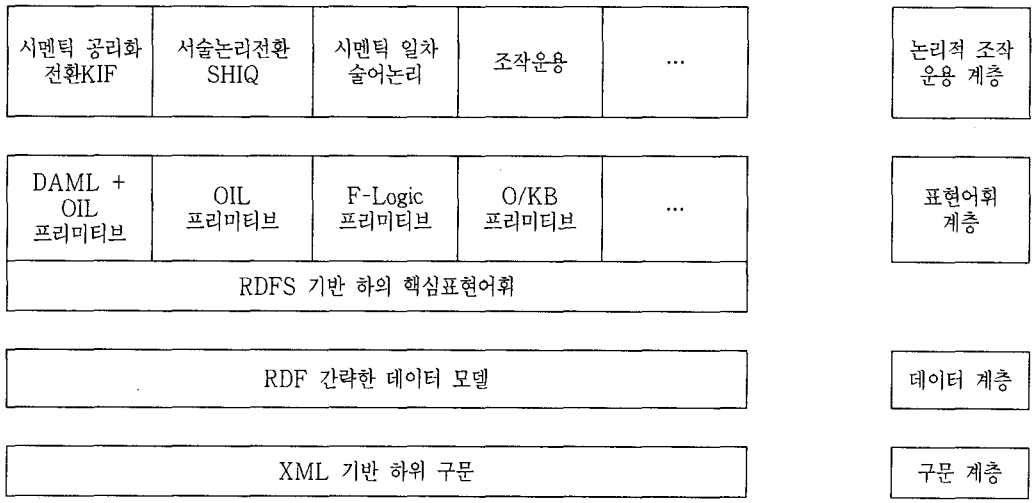
4. 온톨로지의 계층적 언어표현 구조

온톨로지의 종류에는 용어사전, 시소러스, 데이터베이스 스키마, 포털 검색사이트의 디렉토리 등이 포함된다고 할 수 있다. 그러나 이들과 온톨로지의 차이점은 단순한 개념의 분류와 단순한 클래스-하위 클래스 관계를 표시하는 것 이외에도 개념과 개념간의 다양한 관계와 개념의 속성과 제약

12) Robert Neches et. al., "Enabling Technology for Knowledge Sharing," *AI Magxine*, Winter(1991), pp.36-56.

조건 그리고 추론 규칙을 보유하고 있다는 점이다.[13]

온톨로지는 시멘틱 웹에서 다음의 〈그림 2〉와 같은 계층적 언어표현 구조로 표현하고 있다.



〈그림 2〉 시멘틱웹에서의 온톨로지 표현 구조[14]

시멘틱웹에서 온톨로지의 표현 구조는 구문(syntax) 계층, 데이터 계층, 표현 어휘 계층, 논리적 조작 운용 계층의 4개 구조로 이루어지고 있다고 정의하고 있다. 구문 계층에서 XML 문법을 사용하는 이유는 XML이 구조화된 문서의 생성을 유도할 수 있으며, 태그(tag) 이름을 사용자가 임의로 정의할 수 있기 때문에 의미정보를 쉽게 태그에 적용할 수 있기 때문이다.

구문 계층 위에는 데이터 계층이 존재한다. 데이터 계층에서 RDF(Resource Description Framework)의 역할은 컴퓨터가 처리할 수 있는 메타데이터로 표시가 가능한 형식적 데이터와 구조를 정의한다. 즉 RDF는 웹이나 문서 그리고 데이터베이스 등에 있는 어휘들에 대하여 그 의미를 규정할 수 있는 형태로 연결시켜 주는 역할을 한다. 그러나 RDF는 결국 확장이 가능한 형태의 시스템을 정의하고 있는 것으로 개념 계층, 속성들에 대한 영역 그리고 분야에 대한 제약 등을 정의하는 수단을 제공하고 있지만, 표현력(expressive power)의 부족이 문제로 제기되고 있다. 그래서 RDFS (Resource Description Framework Schema)가 요구되고 있는 것이다. RDFS는 상당한 표현력을 갖추고 있어 핵심적인 어휘들을 정의하고 있으나 이들 어휘 간의 부정, 이접, 연접 등의 논리 표현 기능들을 갖추고 있지 않기 때문에 온톨로지의 표현력을 제한하는 결과를 초래할 수가 있다.

13) 이재호, 시멘틱 웹의 온톨로지 언어, 정보과학회지, 제21권, 제3호(2003. 9), pp.4-10.
14) 박현근, OWL 시멘틱 웹 기반 온톨로지 상에서의 규칙-사실 생성에 관한 추론, 박사학위논문, 중앙대학교대학원, 2004. pp.29-32.

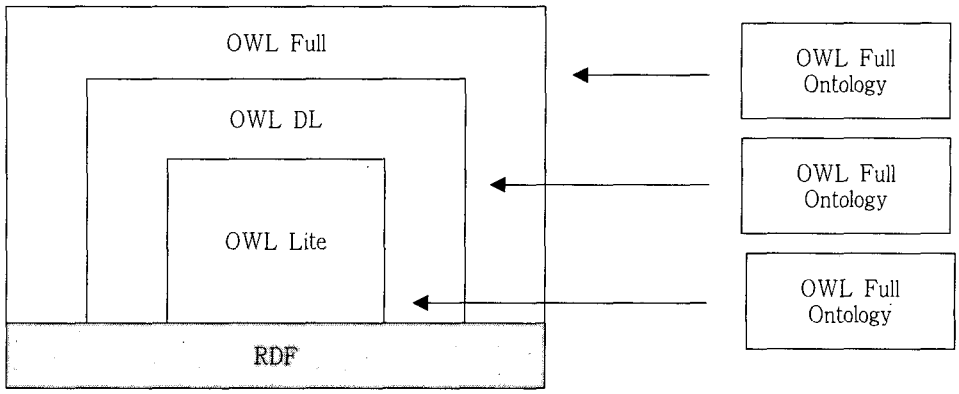
그래서 3단계의 표현어휘계층이 필요한 것이다. 표현어휘계층은 온톨로지를 위해서 주제가 다른 기초적인 어휘들을 제공하는데, 이를 RDFS에 의해 핵심적인 공통어휘들을 규정할 수 있다면 특정 지식표현이나 각기 다른 어플리케이션을 위한 추가적인 어휘들도 정의될 수 있다. 〈그림 2〉는 RDFS와 온톨로지 계층이 어떻게 상위계층인 논리적 조작 운용 계층과 관계되어 있는지를 나타내고 있다.

시멘틱 웹은 웹의 미래에 대한 비전이다. 시멘틱 웹에서 정보는 온톨로지를 통하여 명시적인 의미를 부여받게 되는데, 이를 통해 기계는 좀 더 쉽게 웹에 존재하는 정보들을 자동으로 처리하고 통합할 수 있다. 〈그림 1〉에 나타난 바와 같이 시멘틱 웹은 사용자 정의 태그 스키마를 정의할 수 있는 XML과 데이터를 유연하게 표현할 수 있는 RDF를 바탕으로 구축된다. 시멘틱 웹의 구현에 있어 RDF의 상위 계층에 필요한 것이 웹 문서에 포함된 용어의 의미를 형식적으로 기술할 수 있는 표현언어계층(온톨로지 언어)이다. 기계를 이용하여 웹 문서를 대상으로 유용한 추론 기능을 수행하려면 RDFS가 제공하는 기초적인 의미 표현력을 뛰어넘는 언어가 필요하다

〈그림 2〉에 기술되어 있는 표현어휘 계층은 결국 온톨로지를 표현하기 위해 스키마와 구문구조 등을 정의한 언어인 온톨로지 언어가 되는 것이다. 현재 온톨로지 언어에는 DAML+OIL, OWL, Ontolingua 등이 존재한다. 본 논문에서는 W3C에서 2004년에 공식적으로 제안한 OWL에 대해서만 기술하고자 한다.

Ⅲ. OWL의 정의

OWL은 시멘틱 웹에서 정보를 표현하는 방법으로 2004년에 W3C의 권고안으로 제정되었다. OWL은 풍부한 어휘(vocabulary)와 형식적 의미론(formal semantics)을 포함하고 있기 때문에 XML, RDF, RDFS 보다 더 많은 의미 표현 수단을 제공하므로, 웹 상에서 컴퓨터가 해석할 수 있는 정보를 작성하는데 있어 이들 언어보다 뛰어난 성능을 보인다. OWL언어는 그 표현 능력에 따라 〈그림 3〉과 같이 OWL Lite, OWL DL(Description Logic), OWL Full로 구분하며, 후자로 갈수록 표현력이 더 확장된다. 따라서 OWL을 이용하면 임의의 어휘를 구성하는 용어(term)의 의미와 용어들 간의 관계를 명시적으로 표현할 수 있는 온톨로지를 구축할 수 있다.



〈그림 3〉 OWL의 분류관계

1. OWL의 유형

가. OWL Lite

OWL Lite는 클래스의 분류 계층과 간단한 제약 사항 표현 기능을 요구하는 사용자들에게 적합한 언어이다. 따라서 다른 하위 언어들보다 표현력의 제약이 많다. 예를 들면, OWL Lite는 값의 개수 제약조건(cardinality constraints)의 표현을 지원하지만, 개수 제약조건의 값으로 0 또는 1만 사용할 수 있도록 제한한다. 따라서 OWL Lite를 지원하는 도구를 제작하는 것은 OWL DL이나 OWL Full 언어를 지원하는 도구를 제작하는 것 보다 상대적으로 용이하다. 그러나 OWL Lite는 이론적 복잡도가 낮기 때문에 유의어 사전이나 간단한 분류 체계 그리고 시소러스를 빠르고 손쉽게 OWL화하기 위한 용도로만 적합하다.[15]

나. OWL DL

OWL DL은 기술논리(Description Logic)에 기반한 언어이다. 기술논리는 OWL의 형식적 기반이 된 논리학의 한 분야이다. OWL DL은 계산적 완전성(Computational Completeness)과 결정 가능성(Decidability)을 유지하면서 최대의 표현력을 제공한다. 계산적 완전성은 모든 결론이 계산될 수 있다는 특성이며, 결정 가능성은 모든 계산이 유한한 시간 안에 끝난다는 것을 의미한다. OWL DL은 OWL에서 정의한 모든 어휘를 포함하고 있으나 어휘를 사용할 때에는 사전에 정해진 제약 사항을 준수해야 한다. OWL DL에서의 사전에 정의한 제약사항이라 함은 클래스에 대해서 - 초기에 클래스가 다른 클래스들의 하위 클래스가 될 수 있지만 다른 클래스의 인스턴스로

15) OWL Web Ontology Language Reference(W3C Recommendation 10 February 2004) http://www.w3.org/TR/owl-ref/#OWLLite

선언될 수는 없다고 - 선언한다면 이를 반드시 따라야 한다는 의미이다.[16]

다. OWL Full

OWL Full은 RDF의 모든 문법을 사용할 수 있으며, 최대의 표현력을 제공한다. 그러나 OWL DL처럼 계산적인 보장은 하지 않는다. OWL Full은 OWL DL과 OWL Lite의 모든 기능을 사용할 수 있으며, 유효성과 호환성이란 측면에서 OWL의 세가지 언어 중 가장 완벽하다고 할 수 있다.

그러나 OWL Full을 지원하는 추론 엔진을 구축하는 것은 매우 어려울 것으로 판단된다. 왜냐하면 OWL Full에서 클래스는 개체(Individual)의 집합인 동시에 그 자체가 하나의 개체가 될 수도 있기 때문이다. 뿐만 아니라 OWL Full은 사전 정의된(RDF 또는 OWL) 어휘의 의미를 확장하는 온톨로지를 새롭게 작성할 수 있도록 허용하고 있기 때문이다. 이와 같은 이유로 OWL Full의 모든 기능에 대하여 완벽한 추론 기능을 지원하는 추론 소프트웨어의 제작은 불가능한 것으로 판단된다.

2. OWL의 구문 구조

OWL의 구문 구조는 XML 문서 선언문, 문서의 형태 선언문, Namespaces, 온톨로지 헤더(Header), OWL의 다양한 구문 등의 순서로 이루어진다. OWL의 다양한 구문에는 클래스, 속성, 클래스 사례, 사례들 간의 관계 등을 표현하는 구문들이 포함된다.

가. Namespaces

OWL은 XML의 문법을 사용하므로 XML 문서임을 〈?xml version="1.0"?〉에 의해 선언 하고, 문서의 형태(DOCTYPE)가 rdf:RDF임을 선언한다. 〈표 1〉의 첫 번째 항목에서는 네개의 ENTITY를 제시하였다.

이상과 같이 문서의 형태를 선언한 다음에는 Namespace가 위치한다. OWL에서 Namespace는 온톨로지에 정의된 용어들이 정확한 것임을 확인하기 위해 〈표 1〉의 두 번째 항목에서와 같이 rdf:RDF 뒤에 그 근거를 표시한다. 즉 OWL에서는 XML 스키마, RDF, RDF 스키마 등에 대한 Namespace를 우선적으로 제시한다.

16) OWL Web Ontology Language Reference(W3C Recommendation 10 February 2004) http://www.w3.org/TR/owl-ref/#OWLLite

〈표 1〉 OWL의 구문 구조

```
<?xml version="1.0" ?>
  <!DOCTYPE rdf:RDF [
    <!ENTITY vin "http://www.w3.org/TR/2003/PR-owl-guide-20031209/wine#" >
    <!ENTITY food "http://www.w3.org/TR/2003/PR-owl-guide-20031209/food#" >
    <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
    <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
 ]>
<rdf:RDF
  xmlns = "http://www.w3c.org/TR/2003/PR-owl-guide-20031209/wine#"
  xmlns:vin = "http://www.w3c.org/TR/2003/PR-owl-guide-20031209/wine#"
  xml:base = "http://www.w3c.org/TR/2003/PR-owl-guide-20031209/wine#"
  xmlns:food = "http://www.w3c.org/TR/2003/PR-owl-guide-20031209/food#"
  xmlns:owl = "http://www.w3.org/2002/07/owl#"
  xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs = "http://www.w3.org/2000/01/rdf-schema#"
  xmlns:xsd = "http://www.w3.org/2001/XMLSchema#">
<owl:Ontology rdf:about="">
  <rdfs:comment>An example OWL ontology</rdfs:comment>
  <owl:priorVersion>
    <owl:Ontology "http://www.w3c.org/TR/2003/CR-owl-guide-20030818/wine"/>
  </owl:priorVersion>
  <owl:imports "http://www.w3c.org/TR/2003/PR-owl-guide-20031209/food"/>
  <rdfs:comment>Derived from the DAML Wine ontology at
    http://ontolongua.stanford.edu/doc/chimaera/ontologies/wine.daml
    Substantially changed, in particular the Region based relations.
  </rdfs:comment>
  <rdfs:label>Wine Ontology</rdfs:label>
</owl:Ontology>
  OWL의 다양한 구문
</rdf:RDF>
```

〈표 1〉의 두 번째 항목에서 첫 라인은 현 문서에 대한 Namespace이며, 두 번째 라인은 포도주의 프랑스어 표기법인 'vin'에 대한 Namespace이다. 세 번째 라인은 현 문서에 대한 URI를 나타내며, 네 번째 라인은 food 온톨로지의 지원을 받았음을 나타내며, 다섯 번째 라인은 OWL을 사용했음을 나타낸다. 여섯 번째 라인은 "http://www.w3.org/1999/02/22-rdf-syntax-ns#"의 Namespace를 사용했으며, 일곱 번째 라인과 여덟 번째 라인은 각각 RDF 스키마와 XML Schema datatype의 Namespace를 사용했음을 나타낸다.

나. Ontology Headers

헤더(Headers) 정보는 메타데이터의 역할을 수행한다. 헤더에는 해당 문서에 대한 주석, 판차사항, 다른 온톨로지와의 관계 여부, 해당 온톨로지의 이름 등이 제시된다. 헤더 정보는 〈표 1〉의 세 번째 항목과 같이 owl:Ontology요소와 rdf:about 속성으로 시작하며 복수의 하위요소를 사용해 메타데이터의 역할을 수행한다.

해당 온톨로지에 대한 명확한 설명을 위해서 rdfs:comment를, 판차 사항에 대한 통제 정보의 제공을

위해서 owl:versionInfo, owl:priorVersion, owl:backwardCompatibleWith, owl:incompatibleWith, owl:DeprecatedClass - owl:DeprecatedProperty 등을 사용한다. 또한 다른 온톨로지와의 관계 여부를 알려주기 위해서 owl:imports를 이용해 수집한 URI를 제시하며, 해당 온톨로지의 이름을 제시하기 위해서 rdfs:label을 사용한다.

다. OWL의 기본 요소(Elements)

헤더 정보 다음에 제시되는 요소(Element)들은 해당 온톨로지에서 사용하는 용어들에 대하여 계층, 속성, 용어들간의 관계, 속성들간의 관계 등을 근거로 정의하는 방법에 관한 것들이며, 이를 온톨로지 언어라고 한다. 즉 헤더 정보 다음에 제시되는 요소들은 온톨로지에서 사용하는 요소들인 클래스, 속성, 클래스 사례, 사례들 간의 관계 등을 기술하는 방법에 관련된 언어이다. OWL 문서는 해당 문서의 제일 마지막에 〈/rdf:RDF〉 태그를 제시함에 의해 종료된다.

OWL에서 특정 주제 분야의 용어들을 정의할 때 사용하는 구문과 기능에 대한 상세한 정의와 사용방법은 다음 장에서 포도주(Wine) 온톨로지를 근거로 제시하고자 한다.

Ⅳ. OWL의 어휘와 구문 사용법

OWL을 이용해 온톨로지를 구축하기 위해서는 OWL의 기본 어휘의 의미와 사용법을 이해할 수 있어야 한다. 본 장에서는 OWL의 기본 요소들을 클래스, 속성, 클래스 구성원(individual), 데이터의 형태(data type), 주석(Annotation)과 온톨로지 헤더 정보 표현방법의 순으로 제시할 것이다. 단 주석과 온톨로지 헤더 정보 표현방법은 앞에 제시하였으므로 생략하기로 한다.

다음에 제시할 OWL의 기본 어휘 중, 접두사로 rdf: 및 rdfs:가 부착되어 있는 어휘는 RDF 및 RDFS에 정의되어 있는 어휘이며, 접두사가 붙어있지 않은 단어는 OWL의 자체 어휘이다. 사용 사례는 포도주(Wine)와 관련된 온톨로지이다.[17]

1. 클래스 구문 정의(Class, Thing, Nothing)

가. 클래스 표현 방법 - owl:Class

클래스는 비슷한 특성을 갖는 정보자원들을 그룹화하도록 추출하는 메카니즘을 제공한다. OWL 클래스는 클래스 확장(class extension)과 클래스 인스턴스(instances of class)의 두가지 기능을 갖고 있는데, 클래스 확장은 각기 다른 개체들을 조합시켜 그룹화시키는 기능이며 인스턴스는 클래

17) http://www.w3.org/TR/2004/REC-owl-guide-20040210/wine.rdf

스가 확대됨에 따라 그에 속한 개체들을 의미한다.

OWL에서 클래스는 다음과 같이 여섯 가지 방법으로 표현된다. 클래스 식별자(identifier), 열거(enumeration), 속성 제약(property restriction), 교집합(intersection), 합집합(union), 여집합(complement) 등의 방법을 이용해 클래스를 표현한다. 속성 제한은 클래스가 공통적으로 갖고 있는 특성을 제시하며, 열거는 특성 제한에 해당하는 개체들을 제시하며, 교집합과 합집합과 여집합은 각기 부울 연산자(boolean operation)의 AND, OR, NOT의 기능을 수행하기 위해 존재한다. [18]

OWL DL과 OWL Full은 교집합(intersectionOf - AND), 합집합(unionOf - OR) 및 여집합(complementOf - NOT) 요소를 이용한 임의의 부울 조합을 통해 클래스 및 제약을 결합할 수 있도록 허용한다.

(1) 클래스 식별자(owl:Class)

클래스 식별자는 owl:Class로 시작하며 rdf:ID를 이용해 클래스의 이름을 제시한다. 클래스 식별자는 owl:Thing과 owl:Nothing을 사용한다. 클래스 확장이란 기능면에서 owl:Thing은 모든 구성원들이 모여진 집합체이며, owl:Nothing은 구성원들이 없다는 의미이다. 따라서 모든 OWL 클래스는 owl:Thing의 하위 클래스가 될 수 있으며, owl:Nothing은 모든 클래스의 하위 클래스라는 의미로 사용된다. <표 2>에서 용어의 접두에 '#'의 의미는 rdf:ID를 통해 정의되었다는 의미이며, 일단 정의된 용어는 rdf:resource나 rdf:about를 통해서 #기호를 부착해 표기된다. <표 2>의 첫 번째 항목에서는 WineColor라는 클래스를 정의하고 있다.

<표 2> owl:Class와 owl:oneOF, owl:unionOf의 사용법

```
<owl:Class rdf:ID="WineColor">
  <rdfs:subClassOf rdf:resource="#WineDescriptor" />
  <owl:oneOf rdf:parseType="Collection">
    <owl:Thing rdf:about="#White" />
    <owl:Thing rdf:about="#Rose" />
    <owl:Thing rdf:about="#Red" />
  </owl:oneOf>
</owl:Class>
```

```
<owl:Class rdf:ID="WineDescriptor">
  <rdfs:comment>
   Made WineDescriptor unionType of tastes and color
  </rdfs:comment>
  <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#WineTaste" />
    <owl:Class rdf:about="#WineColor" />
  </owl:unionOf>
</owl:Class>
```

18) OWL Web Ontology Language Reference(W3C Recommendation 10 February 2004). http://www.w3.org/TR/owl-ref/#Class

(2) 열거(owl:oneOF)

열거형 클래스는 owl:oneOF 특성을 이용해 정의하는데, 그 표현 방법은 rdf의 구문인 rdf:parse Type="Collection"을 빌어서 기술한다. 그리고 클래스의 인스턴스는 특성 값에 일치하는 개체들의 리스트가 제시된다.

즉 열거형 클래스는 클래스를 구성하는 개체들을 일일이 열거함에 의해 클래스를 정의할 때 사용한다. 개체 리스트는 owl:Thing로 시작하며 rdf:about를 이용하여 해당 개념(wine)의 collection에 속하는 개체들을 제시한다. 〈표 2〉의 첫 번째 항목의 의미는 WineColor는 WineDescriptor의 하위 클래스로써 WineTaste(맛)와 WineColor(색상)로 나뉘어지며, WineColor의 종류에는 White, Rose, Red 임을 나타내고 있다.

(3) 속성 제한(owl:Restriction)

속성 제한은 제한하는 조건과 일치하는 개체들을 모으는 역할을 하며, 그 종류에는 값의 제약조건(value constraints)과 값의 개수 제약조건(cardinality constraints)이 해당된다.

① 값의 제약조건(owl:onProperty)

값의 제약조건(value constraints)은 클래스에 관계하는 속성 조건과 속성 값의 일치 정도에 따라 완전조건(owl:allValuesFrom), 부분조건(owl:someValuesFrom), 속성값 제한조건(owl:hasValue) 등이 있다.

- 완전조건(owl:allValuesFrom)

값의 제약조건인 owl:allValuesFrom은 임의의 클래스에 대한 속성을 제한하는 역할을 한다. 화이트 와인을 만들 수 있는 포도의 품종은 다양하나 〈표 3〉의 첫 번째 항목과 같이 WhiteBordeaux를 생산하는 원료가 되는 포도는 SemillonGrape와 SauvignonBlancGrape 뿐임을 알 수 있다. 즉 완벽하게 일치하는 속성만을 owl:allValuesFrom으로 표시한다.

- 부분조건(owl:someValuesFrom)

값의 제약조건인 owl:someValuesFrom은 임의의 클래스에 대한 속성을 제약하는데, 앞의 owl:allValuesFrom과는 달리 제시된 속성 값 중 한 개 이상만 일치할 경우에 사용한다. 〈표 3〉의 두 번째 항목에서 Wine의 하위 계층은 지역에 관계하는 속성인 LocatedIn으로 식별이 가능한데, 지역의 일치 정도가 완전히 일치하지 않아도 상관없다는 의미이다.

- 속성값 제한조건(owl:hasValue)

값의 제한조건인 owl:hasValue는 클래스에 대한 속성을 제시하고 그 속성에 일치하는 값을 제시하여 클래스를 정의할 때 사용한다. 〈표 3〉의 세 번째 항목은 WhiteWine이란 클래스가 Wine으로써, 그 조건으로써 색상 속성(hasColor)을 사용하며, 값(owl:hasValue)이 White여야 한다는 의미이다.

〈표 3〉 owl:allValuesFrom, owl:someValuesFrom, owl:hasValue의 사용법

```
<owl:Class rdf:about="#WhiteBordeaux">
  <rdfs:subClassOf>
  <owl:Restriction>
    <owl:onProperty rdf:resource="#madeFromGrape" />
    <owl:allValuesFrom>
     <owl:Class>
      <owl:oneOf rdf:parseType="Collection">
       <owl:Thing rdf:about="#SemillonGrape" />
       <owl:Thing rdf:about="#SauvignonBlancGrape" />
      </owl:oneOf>
  생략
</owl:Class>
```

```
<owl:Class rdf:ID="Wine">
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:onProperty rdf:resource="#locatedIn" />
    <owl:someValuesFrom
     rdf:resource="http://www.w3.org/TR/2003/PR-owl-guide-20031209/wine#Region" />
   </owl:Restriction>
  </rdfs:subClassOf>
</owl:Class>
```

```
<owl:Class rdf:ID="WhiteWine">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Wine" />
    <owl:Restriction>
     <owl:onProperty rdf:resource="#hasColor" />
     <owl:hasValue rdf:resource="#White" />
    </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

② 값의 개수 제약조건(owl:onProperty)

값의 개수(個數) 제약 조건(cardinality constraints)은 특정 클래스의 속성을 구분하는 속성값의 숫자를 한정하기 위한 구문 표현 방법이다. 최대 관계 개수는 owl:maxCardinality을 최소 관계 개수는 owl:minCardinality을 사용한다.

- 최대 관계 개수(owl:maxCardinality)

최대 관계 개수는 특정 클래스의 특정 속성의 개수에 대해 기술한다. 모 클래스의 모 속성에 대해 maxCardinality를 1로 설정하면, 그 클래스의 모든 인스턴스는 해당 속성을 통해 최대 한 개의 개체와

연결된다. 최대 관계차수를 1로 설정한 제약은 함수(functional) 또는 유일(unique) 속성으로 불리기도 한다. 〈표 4〉의 첫 번째 항목에서 WhiteBurgundy를 만드는 포도의 품종은 ChardonnayGrape 밖에 없으므로, owl:maxCardinality의 값이 '1'이 되는 것이다. 만약 해당되는 품종이 없을 경우에는 '0'으로 처리한다. http://www.w3c.org/2001/XMLSh\chima#nonNegativeInteger

〈표 4〉 owl:maxCardinality, owl:minCardinality, 의 사용법

```
<owl:Class rdf:about="#WhiteBurgundy">
  <rdfs:subClassOf>
   <owl:Restriction>
    <owl:onProperty rdf:resource="#madeFromGrape" />
    <owl:hasValue rdf:resource="#ChardonnayGrape" />
   생략
 <rdfs:subClassOf>
   <owl:Restriction>
    <owl:onProperty rdf:resource="#madeFromGrape" />
    <owl:maxCardinality
   rdf:datatype="http://www.w3c.org/2001/XMLSchema#nonNegativeInteger">
     1
   생략
</owl:Class>
```

```
<owl:Class rdf:ID="Meritage">
  <owl:intersectionOf rdf:parseType="Collection">
   <owl:Classrdf:about="#Wine" />
   <owl:Restriction>
    <owl:onProperty rdf:resource="#madeFromGrape" />
    <owl:allValuesFrom>
      <owl:Class>
       <owl:oneOf rdf:parseType="Collection">
        <owl:Thing rdf:about="#CabernetSauvignonGrape" />
            생략(3개 포도 품종명)
        <owl:Thing rdf:about="#MerlotGrape" />
       </owl:oneOf>
        생략
   <owl:Restriction>
    <owl:onProperty rdf:resource="#madeFromGrape" />
    <owl:minCardinality
      rdf:datatype="http://www.w3.org/2001/XMLSchema#nonNegativeInteger">
      2
    </owl:minCardinality>
        생략
</owl:Class>
```

- 최소 관계 개수(owl:minCardinality)

최소 관계차수는 특정 클래스의 특정 속성에 대해 기술한다. 모 클래스의 모 속성에 대해 minCardinality를 1로 설정하면, 그 클래스의 모든 인스턴스는 해당 속성을 통해 최소 한 개 이상의 개체와 연결된다. 즉, 그 클래스의 모든 인스턴스는 반드시 해당 속성의 값을 한 개 이상 가져야 한다는 의미이다. 〈표 4〉의 두 번째 항목에서 Meritage란 포도주를 만들 수 있는 품종은 MerlotGrape

등 5개 품종이다. 그러나 Meritage 포도주는 반드시 2개 이상의 포도주를 혼합해 만들어지므로 owl:minCardinality의 값으로 '2'가 오게 되는 것이다.

(4)교집합(owl:intersectionOf)

교집합(intersection)은 특정 클래스(owl:Class)와 제약(owl:Restriction), 또는 클래스끼리 또는 제약끼리 2개 이상의 조건들을 동시에 만족해야 상위의 클래스를 정의할 수 있음을 의미한다. 〈표 5〉의 첫 번째 항목에서 WhiteBurgundy는 Burgundy에서만 생산되는 WhiteWine을 의미한다.

〈표 5〉 owl:intersectionOf, owl:unionOf, owl:complementOf의 사용법

```
<owl:Class rdf:ID="WhiteBurgundy">
  <owl:intersectionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#Burgundy" />
    <owl:Class rdf:about="#WhiteWine" />
  </owl:intersectionOf>
</owl:Class>
```

```
<owl:Class rdf:ID="WineDescriptor">
  <rdfs:comment>
   Made WineDescriptor unionType of tastes and color
  </rdfs:comment>
  <owl:unionOf rdf:parseType="Collection">
    <owl:Class rdf:about="#WineTaste" />
    <owl:Class rdf:about="#WineColor" />
  </owl:unionOf>
</owl:Class>
```

```
<owl:Class rdf:ID="NonFrenchWine">
  <owl:intersectionOf rdf:parseType="Collection">
   <owl:Class rdf:about="#Wine"/>
   <owl:Class>
    <owl:complementOf>
     <owl:Restriction>
      <owl:onProperty rdf:resource="#locatedIn" />
      <owl:hasValue rdf:resource="#FrenchRegion" />
     </owl:Restriction>
    </owl:complementOf>
   </owl:Class>
  </owl:intersectionOf>
</owl:Class>
```

(5) 합집합(owl:unionOf)

합집합(union)은 상위의 클래스를 정의하고자 할 때 하위에 속하는 복수의 클래스들 중 한개 이상의 조건만 만족시키면 정의할 수 있음을 의미한다. 〈표 5〉의 두 번째 항목에서 WineDescriptor는 WineTaste나 WineColor의 두가지 중에서 하나만 제시해도 가능함을 의미한다.

(6) 여집합(owl:complementOf)

여집합(complement)은 상위의 클래스가 하위 클래스에 정의한 속성들이 반드시 제외되어야 할 경우에 사용한다. 〈표 5〉의 세 번째 항목을 근거로 설명하면 NonFrenchWine이란 클래스는 두 개의 클래스가 결합되어야 하는데, 한 클래스의 속성은 반드시 Wine이어야 하고, 다른 한 클래스의 속성은 속성명이 locatedIn으로써 그 값이 FrenchRegion이 아닌 속성을 의미한다.

나. 클래스 공리(class axioms)

클래스 공리를 나타내는 요소는 하위클래스(rdfs:subClassOf), 동격 클래스(owl:equivalentClass), 배제 클래스(owl:disjointWith)와 열거(owl:oneOf), 교집합(owl:intersectionOf), 합집합(owl:unionOf), 여집합(owl:complementOf) 등이 해당된다. 열거, 교집합, 합집합, 여집합 등에 대한 정의와 구문 사용법은 앞의 항목에서 기술하였으므로 생략하기로 한다.

(1) 하위 클래스(rdfs:subClassOf)

하위 클래스는 특정 클래스가 다른 클래스의 하위에 속함을 표현하고자 할 때 사용한다. rdfs:subClassOf를 이용하면 특정 클래스에 대한 전반적인 계층 클래스를 구축할 수 있다. 〈표 6〉의 첫 번째 항목에서는 이 요소를 이용해 WineTaste가 WineDescriptor의 하위 클래스임을 표현하고 있다.

(2) 동격 클래스(owl:equivalentClass)

특정 클래스가 다른 클래스와 동등한 서열임을 나타내고자 할 때 사용한다. 영어의 wine과 프랑스어의 vin이 동격임을 나타내고자 할 때에는 〈표 6〉의 두 번째 항목과 같이 표현한다. 따라서 특정 클래스들이 vin 클래스에 속하면 이는 wine 클래스에도 동일하게 적용될 수 있음을 유추할 수 있다.

〈표 6〉 rdfs:subClassOf, owl:equivalentClass, owl:disjointWith의 사용법

```
<owl:Class rdf:ID="WineTaste">
  <rdfs:subClassOf rdf:resource="#WineDescriptor" />
</owl:Class>
```

```
<owl:Class rdf:ID="wine">
  <owl:equivalentClass rdf:resource="&vin;wine" />
</owl:Class>
```

```
<owl:Class rdf:ID="LateHarvest">
  <rdfs:subClassOf rdf:resource="#Wine" />
  <owl:disjointWith rdf:resource="#EarlyHarvest" />
</owl:Class>
```

(3) 배제 클래스(owl:disjointWith)

배제 클래스 요소는 두개의 클래스가 전혀 중첩되지 않음을 나타낼 때 사용한다. 〈표 6〉의 세

번째 구문은 LateHarvest는 Wine의 품종에 속하나 EarlyHarvest와는 전혀 관계없음을 나타내고 있는 구문이다.

2. 속성(owl:ObjectProperty, owl:DatatypeProperty)

OWL의 속성은 객체 간의 관계를 나타내는 객체형 속성(owl:ObjectProperty)과 객체와 데이터 값 사이의 관계를 표현하는 데이터 형태 속성(owl:DatatypeProperty)으로 나뉘어진다. 각각의 구문은 〈owl:ObjectProperty rdf:ID="hasColor"〉, 〈owl:DatatypeProperty rdf:ID="yearValue"〉 형태로 표현한다. 객체형 속성은 본 장에서 기술하며, 데이터 형태 속성은 '4. 데이터의 형태 속성'에서 기술한다.

가. RDF 스키마 속성

(1) 속성에 대한 하위속성(rdfs:subPropertyOf)

rdfs:subPropertyOf는 특정 속성에 대한 정의를 할 수 있도록 해당 속성의 하위 속성을 선언할 때 사용하는 요소로써 속성들의 계층 구조를 구축하는 기능을 갖는다. 〈표 7〉의 첫 번째 항목을 근거로 설명하면 hasColor 객체 속성은 hasWineDescriptor 객체 속성의 하위 속성임을 알 수 있다. hasWineDescriptor의 하위 속성은 hasColor, hasSugar, hasFlavor 등이 있으며, 그 값은 각기 White, Rose, Red 와 Sweet, OffDry, Dry 그리고 Delicate, Moderate, Strong 등이 해당된다.

따라서 하위 속성에 대한 선언 구문은 다양한 와인의 종류를 구분하기 위해 색상, 포도의 품종, 생산지 등의 다양한 속성을 부여하고 그 값을 부여할 수 있다. 〈표 7〉의 두 번째 항목에서 WhiteWine은 Wine이며 그 색상 속성(hasColor)의 값(hasValue)이 White임을 알 수 있다.

〈표 7〉 rdfs:subPropertyOf, rdfs:domain, rdfs:range의 사용법

```
<owl:ObjectProperty rdf:ID="hasColor">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty" />
  <rdfs:subPropertyOf rdf:resource="#hasWineDescriptor" />
  <rdfs:domain rdf:resource="#Wine" />
  <rdfs:range rdf:resource="#WineColor" />
</owl:ObjectProperty>
```

```
<owl:Class rdf:ID="WhiteWine">
  <owl:intersectionOf rdf:parseType="Collection">
   <owl:Class rdf:about="#Wine" />
   <owl:Restriction>
    <owl:onProperty rdf:resource="#hasColor" />
    <owl:hasValue rdf:resource="#White" />
   </owl:Restriction>
  </owl:intersectionOf>
</owl:Class>
```

(2) 속성의 정의 영역(rdfs:domain)

속성의 정의 영역은 객체의 속성값이 다른 용도로도 사용될 수 있으므로 현 객체값의 사용범주가 정확히 어떤 영역에서 사용되고 있는지를 한정하기 위해 사용한다. 〈표 7〉의 첫 번째 항목을 근거로 설명을 하면, owl:ObjectProperty에 의해 주어진 hasColor 객체 속성값이 다른 객체의 속성값에도 사용될 수 있으므로 그 사용범위를 정의하기 위해 rdfs:domain을 사용하여 Wine으로 규정하는 역할을 한다.

3) 속성값 제한(rdfs:range)

속성값 제한은 속성의 값으로 제시할 수 있는 값의 항목을 사전에 정의하고 그 범주 안에서 속성값을 주고자 할 때 사용한다. 〈표 7〉의 첫 번째 항목을 근거로 설명하면, hasColor 객체 속성은 hasWine Descriptor란 객체 속성의 하위 속성인데, hasWineDescriptor가 갖고있는 두가지 값(range)인 WineColor와 WineTaste 중 WineColor로 제한시킨 것임을 알 수 있다. 따라서 hasColor의 값은 White, Rose, Red의 세가지 색상 중에 하나로 지정해야 한다.

나. 속성 간의 관계 표현 속성

(1) 동격 속성(owl:equivalentProperty)

동격 속성은 상위 속성에 대한 특성을 정의하기 위해 다수의 하위 속성을 제시하는데, 이들 하위 속성들간의 관계가 대등하다는 의미를 표현하기 위해 사용한다. 〈표 8〉의 첫 번째 항목은 앞에 제시한 속성에 대한 하위속성에서 설명한 바와 같이 hasWineDescriptor의 하위 속성은 hasColor, hasSugar, hasFlavor 등이 있는데, 이들 간의 관계를 표현하고자 할 때 동격 속성을 사용한다. 즉 hasColor와 hasSugar는 계층 관계가 아닌 동격 속성 관계임을 나타낸다.

(2) 역관계 속성(owl:inverseOf)

역관계 속성은 두개의 속성들이 서로 역관계임을 밝혀주어 해당 속성에 대한 정확한 정의를 하고자 할 때 사용된다. 〈표 8〉의 두 번째 항목에서 객체 속성인 madeIntoWine과 madeFromGrape는 역관계임을 나타낸다. 이를 근거로 〈표 3〉에 제시되어 있는 WhiteBordeaux가 포도 품종 Semillon Grape와 SauvignonBlancGrape를 이용해 만들어진다면(madeFromGrape), 역으로 포도품종 SemillonGrape와 SauvignonBlancGrape은 WhiteBordeaux를 생산한다(madeIntoWine)는 역관계 속성 표현이 가능하다.

〈표 8〉 owl:equivalentProperty, owl:inverseOf의 사용법

```
<owl:ObjectProperty rdf:ID="hasColor">
  <owl:equivalentProperty rdf:resource="#hasSugar"/>
</owl:ObjectProperty>
```

```
<owl:ObjectProperty rdf:ID="madeIntoWine">
  <owl:inverseOf rdf:resource="#madeFromGrape" />
</owl:ObjectProperty>
<owl:ObjectProperty rdf:ID="madeFromGrape">
  <owl:inverseOf rdf:resource="#madeIntoWine" />
</owl:ObjectProperty>
```

다. 속성 관계 제한 속성(cardinality constraints)

(1) 기능적 속성(owl:FunctionalProperty)[19]

기능적 속성은 해당 속성의 기능과 관련된 속성을 표현할 때 사용한다. 해당 속성의 기능과 관련된 속성은 통일되어 사용하도록 유일한 형태를 가져야 한다. 〈표 9〉에서 hasSugar는 WineSugar를 나타내는 유일한 속성임을 나타내고 있다. 그리고 각 Wine 종류는 hasSugar를 이용해 그 값으로 Dry, OffDry, Sweet로 표현이 제한되어 있다.

〈표 9〉 owl:FunctionalProperty, owl:InverseFunctionalProperty 의 사용법

```
<owl:ObjectProperty rdf:ID="hasSugar">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty" />
  <rdfs:subPropertyOf rdf:resource="#hasWineDescriptor" />
  <rdfs:range rdf:resource="#WineSugar" />
</owl:ObjectProperty>
```

```
<owl:ObjectProperty rdf:ID="producesWine">
  <rdf:type rdf:resource="&owl;InverseFunctionalProperty" />
  <owl:inverseOf rdf:resource="#hasMaker" />
</owl:ObjectProperty>
```

(2) 역기능 속성(owl:InverseFunctionalProperty)

역기능 속성은 해당 기능 속성에 대한 반대의 기능이 무엇인지를 알려주는 역할을 한다. 예를 들어 역기능 속성의 사용 구문은 생산품에는 생산자가 있음을 알려주는 역할을 한다. 〈표 9〉에서 producesWine과 hasMaker는 역기능적인 속성 관계가 있음을 알려주는 구문이다.

라. 속성 간 논리적 특징 규정 속성

속성의 논리적 특징(characteristics)에는 이행적 속성과 대칭적 속성이 존재한다. 이행적 속성은 계층관계인 속성들끼리의 이행적 관계를 표현할 때 사용하며, 대칭적 속성은 속성 간의 관계가 대칭적일때 그 관계를 표현하기 위해 사용한다.

19) http://www.w3.org/TR/owl-guide/#owl_FunctionalProperty

(1) 이행 속성(owl:TransitiveProperty)

이행 속성은 최상위 속성과 중간 속성 그리고 최하위 속성간의 관계가 계층관계로 이루어지면 최하위 속성은 최상위 속성과 이행 관계임을 나타낼 때 사용한다. Wine은 hasWineDescriptor 속성에 의해 hasWineTaste와 hasWineColor 속성으로 나뉘어지며, hasWineTaste의 하위 속성계층에는 hasBody, hasFlavor, hasSugar의 계층이 있을 경우 hasWineDescriptor와 hasSugar는 서로 이행관계임을 나타낼 때 〈표 10〉의 첫 번째 항목과 같은 구문을 사용한다.

(2) 대칭 속성(owl:SymmetricProperty)

대칭 속성은 속성들 간의 관계가 대칭적인 경우를 표현할 때 사용한다. 〈표 10〉의 두 번째 항목에서 객체 속성 adjacentRegion의 데이터 형태가 대칭적 데이터임을 선언하면(owl#SymmetricProperty), 세 번째 항목에서 MendocinoRegion이 CaliforniaRegion 지역내에 소재하고 있으며, 대칭적 데이터 형태 속성의 성격을 갖고 있는 객체 속성인 adjacentRegion을 사용하여 SonomaRegion가 California Region 지역 내에 속함을 인지하도록 구문화하고 있다. 즉 MendocinoRegion과 SonomaRegion은 모두 CaliforniaRegion 지역 내에 속하므로 두지역이 대칭관계임을 나타내고 있다.

〈표 10〉 owl:TransitiveProperty, owl:SymmetricProperty의 사용법

```
<owl:TransitiveProperty rdf:ID="hasSugar">
  <rdfs:domain rdf:resource="#hasWineDescriptor"/>
  <rdfs:range rdf:resource="#hasWineTaste"/>
</owl:TransitiveProperty>
```

```
<owl:ObjectProperty rdf:ID="adjacentRegion">
  <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#SymmetricProperty" />
  <rdfs:domain rdf:resource="#Region" />
  <rdfs:range rdf:resource="#Region" />
</owl:ObjectProperty>
```

```
<Region rdf:ID="MendocinoRegion">
  <locatedIn rdf:resource="#CaliforniaRegion" />
  <adjacentRegion rdf:resource="#SonomaRegion" />
</Region>
```

3. 클래스 구성원 선언 구문

클래스 구성원 선언(individual)은 개체에 대한 공리를 의미한다. 그 종류는 클래스 멤버와 속성 값, 개체 증명의 두가지 종류가 해당된다.

가. 클래스 멤버와 속성 값

특정 개체를 다른 개체와 식별시키도록 하는 것을 클래스 구성원 선언(개별화)라고 정의하는데,

그 방법 중 하나가 해당 개체에 일치하는 다양한 속성들을 제시함에 의해 다른 개체와 구분하는 방법을 의미한다. 〈표 11〉의 첫 번째 항목은 CortonMontrachetWhiteBurgundy에 대해 hasMaker(생산자), hasSugar(당도), hasFlavor(향기), hasBody(농도) 등으로 그 특성을 개별화한 구문이다.

나. 개체 증명(identity- 실체)

- 동일(owl:sameAs)

클래스 구성원 선언 요소 중, owl:sameAs는 주로 형태가 다르지만 의미가 같은 이형동의어를 연결시키기 위해 사용한다. 〈표 11〉의 두 번째 항목에서는 MikesFavoriteWine와 StGenevieveTexas White은 동일한 와인임을 나타내고 있다.

- 차이(owl:differentFrom)

클래스 구성원 선언 요소 중, owl:differentFrom은 동격 관계인 속성 값들이 서로 다른 것임을 표현할 때 사용한다. 〈표 11〉의 세 번째 항목에서는 Sweet 와 Dry가 서로 다른 것임을 나타내는 구문이다. 인간의 경우 단어의 형태만 보고 서로 의미가 다른 것임을 알 수 있으나 컴퓨터는 이것만으로는 식별할 수 없으므로 자동으로 인식하기 위해서는 그 차이에 대한 정의를 구문화해야 한다.

또한 이 구문은 〈표 9〉의 owl:FunctionalProperty와 관련된 것으로 기능 속성의 값은 한개의 값만 가질 수 있으므로 Sweet와 Dry가 동시에 선언될 수가 없다. 따라서 owl:differentFrom을 사용하여 Sweet와 Dry가 서로 다른 것임을 기술해주어야 한다.

〈표 11〉 클래스 구성원 선언 구문의 사용법

```
<WhiteBurgundy rdf:ID="CortonMontrachetWhiteBurgundy">
  <hasMaker rdf:resource="#CortonMontrachet" />
  <hasSugar rdf:resource="#Dry" />
  <hasFlavor rdf:resource="#Strong" />
  <hasBody rdf:resource="#Full" />
</WhiteBurgundy>
```

```
<Wine rdf:ID="MikesFavoriteWine">
  <owl:sameAs rdf:resource="#StGenevieveTexasWhite" />
</Wine>
```

```
<WineSugar rdf:ID="Sweet">
  <owl:differentFrom rdf:resource="#Dry"/>
</WineSugar>
```

```
<owl:AllDifferent>
  <owl:distinctMembers rdf:parseType="Collection">
   <vin:WineColor rdf:about="#Red" />
   <vin:WineColor rdf:about="#White" />
   <vin:WineColor rdf:about="#Rose" />
  </owl:distinctMembers>
</owl:AllDifferent>
```

- 차이의 동시적 정의(owl:AllDifferent 와 owl:distinctMembers)

차이의 동시적 정의는 앞에 기술한 owl:differentFrom와 동일한 역할을 하나, 그 범주가 여러 개인 경우 각각을 정의하기에 번거로울 수 있으므로, owl:AllDifferent와 owl:distinctMembers를 사용하여 복수의 범주를 동시에 정의하기 위해서 사용하는 구문이다. 〈표 11〉의 네 번째 항목은 Red, White, Rose가 각기 다른 의미임을 한꺼번에 정의하고 있다.

4. 데이터 형태 구문

OWL의 속성은 앞에서 설명한 바와 같이 개체 간의 관계를 나타내는 객체형 속성(owl:Object Property)과 개체와 데이터 값 사이의 관계를 표현하는 데이터형태 속성(owl:DatatypeProperty)으로 나뉘어진다. 데이터형태 속성은 대부분 RDF의 데이터 값 처리 방식을 따른다. OWL 가이드의 데이터형태 관련 부분을 참조하면 대부분 XML 스키마 데이터형태로부터 차용된 OWL의 내장 데이터형태들에 대해 자세한 설명을 볼 수 있다.

데이터 값의 범위를 명세화하기 위해서 사용하는 데이터 범위에 대한 선언은 RDF 데이터 형태, rdfs:Literal 그리고 앞에서 기술한 owl:oneOf를 사용한 열거형 데이터 형태가 있다.

〈표 12〉 owl:DatatypeProperty의 사용법

```
<owl:Class rdf:ID="VintageYear" />
<owl:DatatypeProperty rdf:ID="yearValue">
   <rdfs:domain rdf:resource="#VintageYear" />
   <rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#positiveInteger" />
</owl:DatatypeProperty>
<VintageYear rdf:ID="Year1998">
   <yearValue rdf:datatype="http://www.w3.org/2001/XMLSchema#positiveInteger">
    1998
  </yearValue>
</VintageYear>
<owl:ObjectProperty rdf:ID="hasVintageYear">
   <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#FunctionalProperty" />
   <rdfs:domain rdf:resource="#Vintage" />
   <rdfs:range rdf:resource="#VintageYear" />
</owl:ObjectProperty>
```

〈표 12〉를 근거로 데이터 형태 속성의 사용방법을 설명하면 다음과 같다. 포도 수확연도를 의미하는 VintageYear는 데이터 형태의 속성을 yearValue로 하는데, 본 온톨로지에서 yearValue는 VintageYear(포도수확연도) 영역에 적용되며, yearValue에 대한 값의 종류는 데이터 형태의 속성을 표기하고 있는 XMLSchema에 따른다. 따라서 VintageYear가 Year1998이라 함은 앞에 정의한 XMLSchema에 따르면 1998이란 값을 갖고 있음을 의미한다. hasVintageYear란 객체 속

성은 OWL의 기능적 속성에 해당하며, 앞에 정의한(#) VintageYear의 속성값을 표현할 때 사용한다.

V. 결 론

본 연구에서는 차세대 웹인 시멘틱 웹을 구축하기 위해서 핵심 요소를 담당하는 온톨로지에 대하여 기능과 종류 그리고 적용 가능성에 대하여 살펴보았다. 또한 W3C에 의해 권고안으로 채택된 OWL에 대하여 그 종류와 구문구조에 대한 연구결과를 제시하였다.

또한 차세대 웹인 시멘틱 웹과 온톨로지의 자동 구축을 위하여 OWL의 어휘와 구문 사용방법에 대한 연구결과를 제시하였다. 특히 OWL의 어휘와 기능 그리고 구문에 대한 사용방법을 쉽게 익힐 수 있도록, 클래스 구문, 속성 구문, 클래스 구성원 선언 구문, 데이터 형태 구문 등에 대한 상세한 기능과 정의를 기술하였으며, 그 사용방법을 쉽게 익힐 수 있도록 Wine 온톨로지를 근거로 사용 사례와 설명을 제시하였다.

본 연구를 통해 얻을 수 있었던 결론은 다음과 같다.

첫째, 온톨로지를 OWL을 이용해 구축하려면, 그 어휘 수준은 OWL DL 수준 이상으로 함이 효율적이다.

둘째, 온톨로지에서 용어들간의 관계 표현은 계층적 분류표를 근거한 두뇌적 분류기호 부여방법과 매우 유사하다. 따라서 특정 주제분야에 대한 온톨로지를 구축하기 위해선 해당 분야의 분류표를 근거로 구축하는 것이 효율적이라 판단된다.

셋째, 시멘틱 웹의 계층적 구조에서 나타난 바와 같이 온톨로지는 RDF 스키마(Schema)와 Logic 사이에서 'Ontology Support'의 역할을 한다. 따라서 주제별 온톨로지는 인터넷 자원의 실존 유무에 관계없이 계층적 분류표 등을 이용해 구축하여도 정보검색의 효율성에 지대한 영향을 끼칠 것이다.

넷째, 온톨로지가 갖고 있는 개념적 접근 및 개념간의 관계에 대한 정의의 명확성과 다양성이란 특성이 재현율과 정도율에 관계하는 질의 형성을 용이하게 할 수 있으므로 키워드 매칭 방식에 의한 검색시스템보다 서비스의 질적 수준의 향상이란 효과를 얻을 수 있을 것이다.

다섯째, OWL을 이용해 온톨로지를 구축하면 정보검색식의 자동구축은 물론이고 검색식의 확대와 축소가 가능하다. 그 이유는 탐색어와 관계하는 용어들을 선택할 때 관련 용어를 표현하는 OWL 어휘와 구문을 이용해 AND, OR, NOT 등의 검색 연산자를 적용할 수 있기 때문이다. 단 이를 위해선 구축된 온톨로지를 근거로 관련 용어들을 제시할 수 있는 브라우저가 요구될 것이다.

여섯째, 온톨로지의 구축시 특정 주제에 대한 다국어 온톨로지를 구축하는 것이 정보검색의 효

율성을 향상할 수 있다는 측면에서 도입되어야 할 것이다.

일곱째, 현재 온톨로지 편집기는 해외의 Protege, OilEd, OntoEdit와 같이 우수한 온톨로지 편집기가 존재한다. 그러나 아직까지 용어의 선정, 용어 간의 관계, 용어의 의미를 한정하는 속성들을 모두 수작업으로 구축해야 하는 문제점이 있다. 따라서 특정 용어를 입력할 때 관련 용어를 제시해 줌에 의해 온톨로지를 쉽게 구축할 수 있는 편집기가 요구된다. 그와 같은 편집기는 계층분류표를 응용한 초기단계의 온톨로지를 브라우저로 제공함에 의해서 구축이 가능하다고 판단된다.

OWL의 어휘와 구문 사용방법에 대한 본 연구를 통하여 온톨로지는 지식의 식별, 수집, 개발에 있어서 전문가는 물론 일반인에게도 지식의 포착과 창조의 수단을 제공함을 알 수 있었다. 그리고 OWL을 이용한 온톨로지는 그 구조 표현방법이 계층분류표를 근거한 인간의 두뇌적 분류기호 부여방법과 매우 유사함을 알 수 있었다.

따라서 정보의 수집과 축적 그리고 검색의 효율성을 연구하는 학문인 문헌정보학에서 계층분류표를 근거로 한 다양한 주제의 온톨로지를 구축해서, 특정 주제분야에서 그 분야에 관한 상세한 온톨로지를 구축할 때 기본이 될 수 있는 기반구조를 제공해야 할 것이다.

또한 특정 주제에 대한 온톨로지를 자동으로 구축할 때에는 - 특히 정보검색의 효율을 높일 수 있는 온톨로지를 자동으로 구축하기 위해서는 - 자동색인 알고리즘, 클러스터링 알고리즘을 이용하여 용어들간의 밀착관계를 조사하는 것이 선행되어야 할 것이다.

〈참고문헌은 각주로 대신함〉