시맨틱웹(SemanticWeb) 기술과 정보서비스

오삼균(성균관대학교 문헌정보학과)

1. 시맨틱웹의 등장 배경

웹이 제안된 본 목적은 사람 간의 커뮤니케이션 뿐만 아니라 컴퓨터 시스템 간의 커뮤니케이션 또한 원활하게 하려는 것이었다. 그러나, 이러한 시스템 커뮤미케이션의 발전은 인터넷 정보의 기하급수적 증가와 다량의 정보분석을 위한 하부구조의 부재로 아직 실현되지 못하고 있다. 상당한 가능성을 제시하고 있는 지식표현 (Knowledge Representation) 분야의 연구도 광범위 적용에 관한 해법의 도출에는 한계를 드러내고 있다고 봐야 할 것이다.

이러한 상황에서 현재 W3C가 추진하고 있는 시맨틱웹은 모든 자원의 포괄적인 표현과 자원 간의 관계를 의미적으로 연결하기 위한 논리적 규칙을 도모하는 확실한 대안으로 부각된다. 시맨틱웹은 자원의 의미를 체계적으로정의하여 컴퓨터와 사람의 협력적 운용을 유도하는 현 웹의 확장으로 간주할 수 있다. 다시 말해서 시맨틱웹은 웹 정보의 명확한 정의와 연결에 기반하여 효율적 검색, 자동화, 통합, 재사용을 시도하는 새로운 기술이라 할 것이다.

2. 연구의 목적

본 연구의 목적은 시맨틱웹 기술을 적용했을 경우 나타날 수 있는 메타데이터의 구문적, 의미적 호환성 증가와 이로 인한 정보서비스의 변화를 알아보려는 것이다. 이를 위해 시맨틱웹의 핵심 개념(URI , XML 네임스페이스) 과 이 개념들에 기반한 시맨틱웹의 기반구조(RDF, RDF 스키마, DAML+OIL, OWL 마크업 언어)를 예제 중심으로 살핀 후, 시맨틱웹을 사용한 정보서비스의 형태와 질을 논의하고자 한다.

3. 핵심 개념

3.1 Uniform Resource Identifier (URI)

URI는 기존의 URL과 달리 모든 성격의 자원에 불변의 고유식별자를 할당하여 자원의 효율적 관리를 도모하는 개념이다. 이러한 URI의 불변성은 웹을 하나로 엮어 나갈 중요한 초석이 되는 특징으로서, 그 할당 대상의 예로는 다음과 같은 여러 자원들이 있다[1].

- 네트워크를 통한 접근이 가능한 것 : 전자문서, 이미지, 정보서비스 (예 : 오늘의 세계 날씨에 대한 정보), 자원 컬렉션.
- 네트워크를 통한 접근이 가능치 않은 것: 사람, 회사, 도서관의 일반장서.
- 추상적 개념 : '생성자', '주제', '표제'.

3.2 XML 네임스페이스 (Namespace)

URI로 식별되는 이름들의 집합체인 XML 네임스페이스는 XML 문서의 요소명(element types)이나 속성명(attribute names)의 식별에 주로 사용되며, 다양한 기관에서 정의하는 요소명 또는 속성명 사이에 잠재한 충돌가능성을 방지할 목적으로 제정되었다[2].

더블린코어에서는 자체 정의한 메타데이터 요소들을 다른 기관의 요소들과 구별하기 위해 3개의 네임스페이스를 사용한다[3]. 1) 주요소의 식별을 위한 네임스페이스 (http://purl.org/dc/elements/1.1/), 2) 한정어에 적용되는 네임스페이스 (http://purl.org/dc/terms/), 3) 자료유형(Type)의 식별을 위한 네임스페이스 (http://purl.org/dc/dcmitype) 가 그것인데, 각 경우의 구체적인 예를 들면 다음과 같다.

1) 주요소의 네임스페이스에 각 요소명을 합하여 고유 URI 생성
   - 더블린코어 '표제'의 URI: http://purl.org/dc/elements/1.1/title
   - 더블린코어 생성자의 URI: http://purl.org/dc/elements/1.1/creator
   - 더블린코어 주제의 URI: http://purl.org/dc/elements/1.1/subject

2) 한정어의 네임스페이스에 각 한정어명을 합하여 고유 URI 생성
   - 더블린코어 '대체표제'의 URI: http://purl.org/dc/terms/alternative

---

1) RDF Primer. http://www.w3.org/TR/rdf-primer/
2) Namespaces in XML. http://www.w3.org/TR/REC-xml-names/
3) Namespace Policy for the Dublin Core Metadata Initiative (DCMI). http://dublincore.org/documents/2001/10/26/dcmi-namespace/

- 더블린코어 '갱신일'의 URI:
- http://purl.org/dc/terms/modified
- 더블린코어 '초록'의 URI:
- http://purl.org/dc/terms/abstract

3) 자료유형의 네임스페이스에 각 자료유형의 통제어를 합하여 고유 URI 생성.
   - 자료유형 '이미지'의 URI: http://purl.org/dc/dcmitype/Image
   - 자료유형 '컬렉션'의 URI: http://purl.org/dc/dcmitype/Collection
   - 자료유형 '소프트웨어'의 URI: http://purl.org/dc/dcmitype/Software

이것은 요소명에 URI를 할당하는 것과는 다르다. 각 통제어에 URI를 할당하는 점이 주목을 요한다. 이전에 구축된 시소러스에 URI 개념을 접목시키면 '온톨로지'를 생성할 수 있게 된다고 볼 수 있다. 인공지능 분야에서 '온톨로지'는 주로 '개념의 표시'로 정의되었고, 기계가독형으로 정의된 어휘 및 어휘들 간의 관계를 정의하는 것으로 간주되어 왔다.

4. 시맨틱웹의 핵심 기반구조

시맨틱웹은 현 웹의 연장으로 보아야 하며, 정보의 의미가 보다 명확하게 정의된 환경, 즉 모든 자원 기술에 사용된 요소명과 요소명들 간의 관계, 속성명, 통제어와 통제어들 간의 관계 규명으로 컴퓨터와 사람의 협력이 용이해진 환경을 말한다[4].

4.1 XML 스키마[5]

XML DTD를 보완할 목적으로 제정된 XML 스키마는 시맨틱웹의 구성에 필요한 기반구조를 구현하는 역할을 한다고 볼 수 있다. 이하는 DTD로 정의된 간단한 서적 관련 메타데이터 스키마<표 1>를 XML 스키마<표 2>로 변환시킨 후 그 이점을 간단히 비교해 본 것이다.

---

4) The Semantic Web Community Portal. http://www.semanticweb.org
5) XML Schema. http://www.w3.org/XML/Schema

BookStore DTD[6]

```
<!ELEMENT BookStore (Book)*>
<!ELEMENT Book (Title, Author, Date, ISBN, Publisher)>
<!ELEMENT Title (#PCDATA)>
<!ELEMENT Author (#PCDATA)>
<!ELEMENT Date (#PCDATA)>
<!ELEMENT ISBN (#PCDATA)>
<!ELEMENT Publisher (#PCDATA)>
```

〈표 1〉

BookStore XML 스키마[7]

```
<?xml version="1.0"?>
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema"
   targetNamespace=http://www.books.org
   xmlns="http://www.books.org">
    <xsd:simpleType name="ISBNType">
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="\d{1}-\d{5}-\d{3}-\d{1}"/>
            <xsd:pattern value="\d{1}-\d{3}-\d{5}-\d{1}"/>
            <xsd:pattern value="\d{1}-\d{2}-\d{6}-\d{1}"/>
        </xsd:restriction>
    </xsd:simpleType>
    <xsd:element name="BookStore">
        <xsd:complexType>
            <xsd:sequence>
                <xsd:element name="Book"  maxOccurs="unbounded">
                    <xsd:complexType>
                        <xsd:sequence>
                            <xsd:element name="Title" type="xsd:string"/>
                            <xsd:element name="Author" type="xsd:string"/>
                            <xsd:element name="Date" type="xsd:gYear"/>
                            <xsd:element name="ISBN" type="ISBNType"/>
                            <xsd:element name="Publisher" type="xsd:string"/>
                        </xsd:sequence>
                    </xsd:complexType>
                </xsd:element>
            </xsd:sequence>
        </xsd:complexType>
    </xsd:element>
</xsd:schema>
```

〈표 2〉

6) XML Schema Tutorial. http://www.xfront.com/#schema
7) XML Schema Tutorial. http://www.xfront.com/#schema

DTD의 데이터 유형 지원이 약한 관계로 거의 모든 요소가 문자열인 #PCDATA로 정의되었고 카디랄니티 조절도 0,1,무한 등 3가지 중 선택해야 한다. 그러나 이것을 XML 스키마로 표현하면 〈표 2〉와 같은 결과가 나온다.

XML 스키마에서는 새로운 네임스페이스를 지정할 수 있기 때문에 targetNamespace를 적용. 스키마 지정 문서에서 정의하고 있는 요소들이 http://www.books.org 라는 네임스페이스에 연관되어 저장되도록 하고 있다. 〈표 2〉에서의 ISBN과 Date 요소 정의는 또한 보다 정교해지고, 카디날리티 조절에 있어서도 minOccurs와 maxOccurs 속성에 의지해 정수의 값 중에서 임의선택이 가능해졌다.

XML 스키마의 장점은 이처럼 네임스페이스 개념을 완벽하게 지원하고, 거의 모든 데이터 유형에 적용되며, 또한 카디날리티(cardinality) 조절이 용이하다는 것이다.

4.2 Resource Description Framework (RDF)

상이한 메타데이터 도메인 간의 의미적 매핑(mapping) 지원이 상당히 미흡한 XML 스키마 만으로는 메타데이터 요소들의 의미와 다른 요소들과의 관계를 기계가독형으로 표현하는 일은 불가능하다.

RDF는 웹자원의 기술을 위한 일반용도의 언어로써 제정된 것으로, 그 장점 중 하나는 의미의 손상 없이 응용프로그램 간의 정보가 교환되도록 해당 정보를 표현하는 프래임워크를 제시한다는 점이다.

RDF는 메타데이터를 인코딩, 교환, 재사용할 수 있는 기반을 제공한다. 또한 메타데이터 의미의 상호 충돌 없는 표현에 필요한 구문구조를 갖추었고, XML 네임스페이스 개념을 완벽하게 지원한다. 표준화된 메타데이터들의 일관된 인코딩과 교환을 가능케 하는 RDF의 이러한 구조는 상이한 메타데이터 간의 호환성과 의미적 모듈화 (semantic modularity)를 제공하는 기반이라 할 수 있다[8]. XML 스키마와 RDF 스키마는 그러므로 상호 보완적인 기술로 이해되어야 할 것이다. 차후에는 이 두 스키마 간의 긴밀한 협조도 예상되며 결국은 병합될 가능성 또한 배제할 수 없다.

예를 들어, '홍길동' 이라는 이름의 교수가 변량분석(ANOVA)에 대한 강의안을 작성하였고 한국의 어떤 URI기관에서 이 교수에게 부여한 URI가 'http://uri.kr/professor/70032' 라고 가정했을 때, 이 사실을 RDF로 표현한다면 다음과 같이 정리될 수 있다.

---

8) An introduction to the Resource Description Framework, Miller, E.
URL: http://www.dlib.org/dlib/may98/miller/05miller.html

- Subject: http://www.skku.ac.kr/statistics/anova.xml
- Predicate: http://purl.org/dc/elements/1.1/creator
- Object: 홍길동 혹은 http://uri.kr/professor/70032

RDF에서는 자원을 '주어(subject)'로, 속성을 '서술어(predicate)'로, 속성의 값을 '목적어(object)'로 간주하여 표현하기도 한다. 이 경우 '주어'와 '술어'의 값은 반드시 URI를 사용해야 하지만, '목적어'의 값은 문자열 또는 URI를 사용할 수 있다. 따라서 이 예에 주어, 술어, 목적어로 표기된 사항은 'http://www.skku.ac.kr/statistics/anova.xml자원의 생성자는 홍길동 이다' 라는 문장으로 표현될 수 있다. 여기서 주목할 점은 '자원 생성자'의 속성 표기에 더블린코어 요소인 creator의 URI를 사용한다는 것, 그리고 효율적 인적자원관리를 위해 홍길동 교수에게 부여된 URI를 사용함으로써 식별에서의 혼동을 완벽하게 방지한다는 점이다.

또 만일 이 자원이 통계학강의 (http://www.skku.ac.kr/statistics/main.xml)의 한 부분이라면 더블린코어의 'isPartOf' 요소를 사용하여 그 관계를 다음과 같이 명시할 수도 있다.

- Subject: http://www.skku.ac.kr/statistics/anova.xml
- Predicate: http://purl.org/dc/terms/isPartOf
- Object: http://www.skku.ac.kr/statistics/main.xml

RDF는 레코드를 하나의 기술단위로 취급해온 기존 방식과는 달리 자원, 속성, 속성값을 하나의 단위로 취급하는 'TRIPLE' 개념을 그 핵심으로 삼는다. 이러한 자원속성 표현의 세분화로 자원 기술이 정교해지며, 자원들 간의 관계설정이 속성(predicate)에 의해 무한으로 가능해진다. RDF는 1) 각 자원이 URI로 고유의 식별자를 가진다는 점, 또한 2) 자원을 기술하는 속성명이 고유한 URI로 표현되어 각 네임스페이스에서 정의된 속성을 사용함으로써 의미충돌을 방지할 수 있다는 점, 3) 속성의 값으로 다른 URI가 지칭될 수 있고 속성의 값으로 지정된 자원은 다시 기술 대상이 되며 따라서 그 자원에 대한 속성과 속성의 값을 새로 부과할 수 있다는 점이 그 강력한 특징이다.

4.3 RDF 스키마

RDF는 자원의 속성, 자원들 간의 관계 기술에는 우수하지만 속성(properties)과 클래스(classes)의 정의, 클래스와 클래스 간의 관계, 속성과 속성 간의 관계 등을 정의하는 방법은 제공하지 않는다. 반면 'RDF 스키마'는 기술된 자원의 유형(types)및 클래스(classes)를

규명하고, 이러한 기술을 위해 필요한 속성(properties)을 명시하는 기능을 지원한다. 따라서 'RDF 스키마'를 사용하면 사람이 이해하는 동시에 기계 처리가 가능한 형태로 메타데이터 속성과 클래스 간의 관계를 표현할 수 있다. 이러한 체계적 표현의 핵심은 클래스와 속성의 명확한 정의, 속성들 간의 관계와 클래스들 간의 관계 정립에 있으며, 이와 같이 체계화된 어휘는 이종 메타데이터의 구문적, 의미적 호환성을 촉진하는 기반이 될 것이다.

4.3.1 클래스(class)의 정의

RDF 스키마에서의 클래스는 일반적인 Type이나 Category 개념과 유사하며, 그 예로는 웹페이지, 사람, 문서, 데이터베이스, 추상적 개념 등을 들 수 있다. 〈표 3〉은 클래스의 정의, 클래스와 클래스들 간의 상하위 관계를 RDF 스키마로 표현한 것으로서, RDF 스키마에서의 온톨로지 체계 확립 방법을 보여주는 한 예이다.

```
〈rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"〉

〈rdf:Description rdf:ID="MotorVehicle"〉
  〈rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/〉
  〈rdfs:subClassOf rdf:resource="http://www.w3.org/2000/01/rdf-schema#Resource"/〉
〈/rdf:Description〉

〈rdf:Description rdf:ID="PassengerVehicle"〉
  〈rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/〉
  〈rdfs:subClassOf rdf:resource="#MotorVehicle"/〉
〈/rdf:Description〉

〈rdf:Description rdf:ID="Truck"〉
  〈rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/〉
  〈rdfs:subClassOf rdf:resource="#MotorVehicle"/〉
〈/rdf:Description〉

〈rdf:Description rdf:ID="Van"〉
  〈rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/〉
  〈rdfs:subClassOf rdf:resource="#MotorVehicle"/〉
〈/rdf:Description〉

〈rdf:Description rdf:ID="MiniVan"〉
  〈rdf:type rdf:resource="http://www.w3.org/2000/01/rdf-schema#Class"/〉
  〈rdfs:subClassOf rdf:resource="#Van"/〉
  〈rdfs:subClassOf rdf:resource="#PassengerVehicle"/〉
〈/rdf:Description〉
〈/rdf:RDF〉
```

〈표 3〉

이 예의 경우RDF와 RDF 스키마에 관한 네임스페이스만 선언된 점으로 볼 때 RDF와 RDF 네임스페이스에 규정된 것을 활용하여 새로운 class 또는 property를 정의하려는 것이다. 우선 rdf:Description은 새로운 class나 property의 기술하겠다는 뜻이고, rdf:ID에 정의할 class나 property를 지정한다. rdf:ID의 값으로 지정된 'MotorVehicle'이 class에 속하는 것은 rdf:type으로 표기하였고, rdfs:subClassOf를 사용, 'MotorVehicle'을 resource의 하위 class로 정의하였다. Truck, Van, MinVan 등은 모두 class로 (rdf:type 적용), PassengerVehicle, Truck 및 Van은 MotorVehicle의 하위 class, MinVan은 Van과 PassengerVehicle의 하위 class로 정의하였다.

4.3.2 속성(property)의 정의

RDF 스키마에서 속성(property)에 관한 정의는 rdf:Property, rdfs:domain, rdfs:range, rdfs:subPropertyOf 등을 통해 기술한다. 새로운 property를 정의할 경우, 먼저 rdf:type으로 정의하고자 하는 것을 property로 선언하고, property가 적용되는 class의 영역은 rdfs:domain으로, property가 취하는 값의 범위는 rdfs:range로 각각 제한할 수 있다. 표 4.는 이의 간단한 예이다.

```
〈rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"〉
〈rdf:Description rdf:ID="registeredTo"〉
  〈rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/〉
  〈rdfs:domain rdf:resource="#MotorVehicle"/〉
  〈rdfs:range rdf:resource="http://www.example.org/classes#Person"/〉
〈/rdf:Description〉
〈rdf:Description rdf:ID="rearSeatLegRoom"〉
  〈rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/〉
  〈rdfs:domain rdf:resource="#PassengerVehicle"/〉
  〈rdfs:range rdf:resource="http://www.w3.org/2001/XMLSchema#integer"/〉
〈/rdf:Description〉
〈/rdf:RDF〉
```

〈표 4〉

'registeredTo'는 rdf:type에 의해서 property로 정의되었고, 이 속성의 도메인은 rdfs:domain에 의해서 'MotorVehicle' class에 속하는 것에만 적용할 수 있다고 한정되었고, 'person' class 에 속한 것만이 이 속성의 값으로 올 수 있다는 점은 rdfs:range로 규정지었다. 'rearSearLegRoom'의 경우도 먼저 property로 선언한 후, 이 속성은 'PassengerVehicle' class에 속하는 것에만 적용될 수 있고, 이 속성의 값으로는 정수(integer)만 가능하다고 기

술하고 있다.

RDF 스키마에서는 또한 'rdfs:subPropertyOf'를 사용, 한 property를 다른 property의 하위속성으로 표시할 수도 있는데, 이 때 property에 적용되는 domain과 range는 그 하위속성에도 그대로 적용된다. 〈표 5〉는 이러한 property 간의 상.하위 관계를 정의한 예이다.

```
〈rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"〉
〈rdf:Description rdf:ID="biologicalParent"〉
〈rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/〉
〈/rdf:Description〉
〈rdf:Description rdf:ID="biologicalFather"〉
  〈rdf:type rdf:resource="http://www.w3.org/1999/02/22-rdf-syntax-ns#Property"/〉
〈rdfs:subPropertyOf rdf:resource="#biologicalParent"/〉
〈/rdf:Description〉
〈/rdf:RDF〉
```

〈표 5〉

예에서는 'biologicalParent'를 property로 우선 정의한 후, 'biologicalFather'는 property이자 이미 정의된 'biologicalParent'의 하위 속성이라고 규정하고 있다. 〈표 6〉은 더블린코어를 RDF 스키마로 정의하는 방법의 한 예를 보인 것이다[9].

```
〈rdf:RDF
  xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
xmlns:rdfs=http://www.w3.org/2000/01/rdf-schema#
xmlns:dc="http://purl.org/dc/elements/1.1/"
xmlns:dcterms=http://purl.org/dc/terms/
xmlns:dctype="http://purl.org/dc/dcmitype/"〉
〈rdf:Property rdf:about="http://purl.org/dc/elements/1.1/relation"〉
 〈rdfs:isDefinedBy rdf:resource="http://purl.org/dc/elements/1.1/" /〉
〈/rdf:Property〉
〈rdf:Property rdf:about="http://purl.org/dc/terms/isReferencedBy"〉
 〈rdfs:subPropertyOf rdf:resource="http://purl.org/dc/elements/1.1/relation" /〉
 〈rdfs:isDefinedBy rdf:resource="http://purl.org/dc/terms/" /〉
〈/rdf:Property〉
〈/rdf:RDF〉
```

〈표 6〉

9) Dublin Core Qualiers in RDF. http://purl.org/dc/terms/

예에서는 'relation'이 property에 속한다는 사실을 rdf:type 아닌 rdf:Property로 정의하였고 (축약의 형태), 이 요소가 정의되어 있는 URI를 언급하였다. 또한 'isReferencedBy'도 property에 속하며, 이것이 'relation'의 하위속성이라는 점을 'rdfs:subPropertyOf'에 의해 정의하고 있다.

4.3.3 RDF 스키마와 상호운용성

RDF 스키마가 메타데이터 스키마 설계와 정의의 국제적인 관례로 합의된다면, 모든 메타데이터와 연관된 class와 property는 기계가독형으로 정의되고, 모든 property의 적용영역과 범위는 domain과 range로 명확히 제한될 것이다. 이럴 경우 메타데이터 스키마를 정의하는 각 정보 기관들이 1) 가능한 한 기존의 보편적 표준 요소들을 채택하고 2) 결여된 필요 요소들은 새로이 정의하되 기존 요소들과의 관계설정이 가능한 것은 RDF 스키마를 적용하여 정의한다면 메타데이터 스키마들 간의 상호운용성은 현재보다 크게 향상될 수 있을 것이다.

4.4 DAML+OIL

시맨틱웹에 본격적인 웹 온톨로지 언어가 필요한 이유는 메타데이터 스키마 정의의 근간이 되는 RDF 스키마에는 서로 동일한 의미의 요소, 역관계, union, intersection 등 메타데이터의 중요한 관계를 지원하는 능력이 결여되어 있기 때문이다.

RDF와 RDF 스키마를 기반으로 이 두 언어에 부족한 모델링 요소를 확장, 강화하여, 개발된 DAML+OIL[10]은 웹자원을 기술하기 위한 시맨틱웹 마크업 언어이다. DAML+OIL의 네임스페이스 선언은 〈표 7〉과 같은 형태를 취한다.

```
<rdf:RDF
  xmlns:rdf  ="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
  xmlns:daml="http://www.w3.org/2001/10/daml+oil#"
  xmlns:xsd  ="http://www.w3.org/2000/10/XMLSchema#"
  xmlns:dex  ="http://www.w3.org/TR/2001/NOTE-daml+oil-walkthru-20011218/daml+oil-ex#"
  xmlns:exd  ="http://www.w3.org/TR/2001/NOTE-daml+oil-walkthru-20011218/daml+oil-ex-dt#"
  xmlns      ="http://www.w3.org/TR/2001/NOTE-daml+oil-walkthru-20011218/daml+oil-ex#">
```

〈표 7〉

10) Annotated DAML+OIL Markup. http://www.w3.org/TR/daml+oil-walkthru/

이하 DAML+OIL에서 강화된 class 및 property 정의의 주요 골자를 각각 살펴본다.

4.4.1 DAML+OIL에서의 class 정의

〈표 8〉의 예는 동물 온톨로지 개발과 관련한 DAML+OIL에서의 class 정의 구문 구조이다.

```
〈daml:Class rdf:ID="Animal"〉
  〈rdfs:label〉Animal〈/rdfs:label〉
  〈rdfs:comment〉
    This class of animals is illustrative of a number of ontological idioms.
  〈/rdfs:comment〉
〈/daml:Class〉

〈daml:Class rdf:ID="Male"〉
  〈rdfs:subClassOf rdf:resource="#Animal"/〉
〈/daml:Class〉

〈daml:Class rdf:ID="Female"〉
  〈rdfs:subClassOf rdf:resource="#Animal"/〉
  〈daml:disjointWith rdf:resource="#Male"/〉
〈/daml:Class〉
```

〈표 8〉

DAML+OIL은 세상에 존재하는 모든 것을 객체(objects)로 분리해 생각한다. 이 객체는 DAML class, property, datatype으로 구분되며, datatype은 XML 스키마에서 이미 정의된 것들로 class의 정의에 사용된다. 예에서는 'Animal'을 DAML class로 우선 정의하고, 'Male'과 'Female'을 'Animal'의 하위 class로 정의하면서 'Female'은 'Male'과 'disjoint'관계라는 것을 명시하고 있다. 즉 이 부분은 어느 누구도 'Male'인 동시에 'Female'이 될 수는 없다는 사실을 규정한다.

4.4.2 DAML+OIL에서의 property 정의

DAML+OIL의 property는 객체들 간의 관계를 기술하는 property와 객체를 데이터유형의 값과 연결시키는 property로 나눠진다. 전자는 daml:ObjectProperty에 속하고, 후자는 daml:DatatypeProperty에 속하는데, 〈표 9〉는 property를 정의하는 방법에 관한 DAML+OIL의 구문 예이다.

```
<daml:ObjectProperty rdf:ID="hasParent">
  <rdfs:domain rdf:resource="#Animal"/>
  <rdfs:range rdf:resource="#Animal"/>
</daml:ObjectProperty>

<daml:ObjectProperty rdf:ID="hasFather">
  <rdfs:subPropertyOf rdf:resource="#hasParent"/>
  <rdfs:range rdf:resource="#Male"/>
</daml:ObjectProperty>

<daml:DatatypeProperty rdf:ID="shoesize">
<rdf:type rdf:resource="http://www.w3.org/2001/10/daml+oil#UniqueProperty"/>
  <rdfs:range rdf:resource="http://www.w3.org/2000/10/XMLSchema#decimal"/>
</daml:DatatypeProperty>

<daml:DatatypeProperty rdf:ID="age">
  <rdf:type rdf:resource="http://www.w3.org/2001/10/daml+oil#UniqueProperty"/>
  <rdfs:range rdf:resource="http://www.w3.org/2000/10/XMLSchema#nonNegativeInteger"/>
</daml:DatatypeProperty>
```

〈표 9〉

이 예에서는 'hasParent'와 'hasFather'가 객체들 간의 관계를 기술하는 property로 정의되었고, 'hasFather'는 'hasParent'의 하위 속성(subPropertyOf)에 속한다는 것도 함께 나타냈다. 또 'hasParent'가 적용될 수 있는 class는 'Animal'이며, 이 속성의 값 역시 'Animal' 중에서 취할 수 있다고 명시하고 있다. 이와는 약간 다른 속성의 정의로 'shoesize'와 'age'를 기술하고 있는데, 둘 다 'UniqueProperty'라 한 것은 'shoesize'와 'age'는 유일해야 한다는 뜻으로서, 'shoesize'는 소수를, 'age'는 정수를 그 값으로 취한다.

4.4.3 DAML+OIL에서의 property 제한 표기 방법

어떤 class에 관한 제한 부여는 'daml:Restriction', 'onProperty', 'toClass' 등으로 한다. 일례로 'Person'이라는 class를 정의하면서 이를 'Animal'의 하위 class로 규정한 후, 'Person'의 'hasParent' 속성은 'Animal' 에서 그 값을 취하지 않고 반드시 'Person' class에서만 값을 취해야 하는 사실을 명시하고자 한다면 〈표 10〉의 예와 같이 할 수 있다.

예에서는 'Person'이라는 class의 'hasParent'의 속성값이 'Person' 클래스로부터 와야 한다는 것을 표현하고 있다. 또 사람은 아버지가 한 사람(cardinality가 1)이어야 하고, 신발 크기는 적어도 하나 (minCardinality)를 가져야 한다고 제한하고 있다.

```
<daml:Class rdf:ID="Person">
  <rdfs:subClassOf rdf:resource="#Animal"/>
  <rdfs:subClassOf>
    <daml:Restriction>
      <daml:onProperty rdf:resource="#hasParent"/>
      <daml:toClass rdf:resource="#Person"/>
    </daml:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <daml:Restriction daml:cardinality="1">
      <daml:onProperty rdf:resource="#hasFather"/>
    </daml:Restriction>
  </rdfs:subClassOf>
  <rdfs:subClassOf>
    <daml:Restriction>
      <daml:onProperty rdf:resource="#shoesize"/>
      <daml:minCardinality>1</daml:minCardinality>
    </daml:Restriction>
  </rdfs:subClassOf>
</daml:Class>
```

〈표 10〉

이상의 예들 외에도 DAML+OIL은 RDF와 RDF 스키마에 기타의 다양한 데이터 모델링 표현력을 부가했다[11]. 현재의 추세로는 특히 'wine'[12]과 'food'[13] 분야에서 DAML+OIL 언어를 사용한 온톨로지의 개발이 활발하다. 이 언어는 또한 Web Ontology Languge (OWL)로 흡수되어 발전을 계속하고 있다.

4.5 Web Ontology Language (OWL)

OWL은 DAML+OIL과 RDF(S) 기반 위에 class와 property들 간의 관계를 보다 명료하게 정의할 수 있도록 표현력을 더욱 강화시킨 것이다. 다시 말하자면 OWL은 웹 문서와 응용프로그램에 내재해 있는 모든 class와 class들, property와 property 간의 관계를 기술할 수 있는 언어이다.

만일 일례로 현존의 웹 에이전트(agents)에게 '식사 메뉴의 리스트를 제공하고 각 메뉴에 적합한 술을 추천하되, Sauterne은 제외시키라'는 요구사항을 준다면 이것은 아주 힘든

11) DAML+OIL Reference Descritpion. http://www.w3.org/daml+oil-reference
12) Wine Ontology. http://www.w3.org/TR/2002/WD-owl-guide-20021104/wine.owl
13) Food Ontology. http://www.w3.org/TR/2002/WD-owl-guide-20021104/food.owl

질의가 될 것이다. 키워드 검색을 넘어서는 이런 기능의 지원은 웹 자원의 의미 기술을 요한다. OWL은 웹 온톨로지에 연관된 지식창고(knowledge bases)를 정의하는 언어이다. 온톨로지는 철학에서 빌려온 어휘로서, 세상에 존재하는 여러 종류의 개체(entities)들과 그들의 연결관계를 규명하는 학문분야라고 볼 수 있다. OWL 온톨로지는 클래스(classes)와 속성(propertities), 그리고 클래스와 속성에 적용할 수 있는 갖가지 제약사항(constraints)들의 집합이며, 다음과 같은 요소들을 포함한다.

- 클래스들 사이의 텍사노미 관계
- 데이터의 속성, 즉 클래스의 요소가 되는 속성의 값에 관한 기술
- 객체의 속성, 즉 클래스의 요소간의 관계에 관한 기술
- 클래스들의 인스턴스
- 속성들의 인스턴스

OWL의 논리적 명제(assertions)들이 추론시스템에 축적될 경우, 이것은 지식창고로 간주된다. 명제는 클래스의 멤버(members)들에 관한 사실들 뿐 아니라 온톨로지에 구문적으로 명확히 정의되지는 않았지만 논리적으로 유추 가능한 사실들도 포함한다. 이런 명제들은 하나의 온톨로지에 근거할 수도 있고, 다수의 분산형 온톨로지를 OWL에 명시된 방식에 따라 수집한 온톨로지에 근거한 것일 수도 있다.

XML과 XML 스키마를 사용하는 경우와 비교한 OWL의 구체적 이점은 현재 계속 논의되고 있는 사항이다. 예를 들면 XML 태그와 내용에 관해 운영적 합의를 도출한다면 OWL에서 성취하려는 결과와 과연 무엇이 다를 것인가 하는 질문이 그것인데, 이에 대해 OWL이 제시하는 답변은 다음과 같다[14].

- 온톨로지는 지식표현(knowledge representation)이지 메시지 전달을 목적으로 한 포맷이 아니라는 점에서 근본적인 차이가 있다. 대부분의 웹기반 산업표준은 메시지 포맷과 프로토콜에 관한 기술이기 때문에 상거래 이외의 상황에서는 추론 기능을 제공하지 못한다.
- OWL 온톨로지의 장점은 추후 온톨로지에 관한 추론을 가능케 하는 많은 도구가 개발될 것이라는 점이다. 즉 특정 도메인에 국한되지 않는 포괄적 적용이 가능한 온톨로지를 형성할 수 있다는 것이다. 다양한 분야에서의 온톨로지 구축이 기대되고 있으며, 이 과정에서 OWL 의 공식 속성들에 의거하여 이미 개발된 도구들 중 우수한 부분이 재활용 될 수 있다. 강력하고 유용한 온톨로지 구축은 결코 용이한 작업이 아니지만, 단계적이고 점진적인 구축은 충분히 실현 가능하다.

14) Web Ontology Language (OWL) Guide Version 1.0 http://www.w3.org/TR/owl-guide/

5. 시맨틱웹과 정보서비스

RDF, RDF 스키마, 그리고 OWL을 사용한 시맨틱웹의 형성은 미래의 정보서비스를 크게 발전시킬 것이다. 이하 시맨틱웹의 구축으로 가능해질 새로운 검색기능을 요약해본다.

- 고유의 URI로 표기된 자원들 간의 관계가 속성으로 기술되므로 다양하게 정의된 속성의 온톨로지 중 필요한 관계에 놓여 있는 자료의 수집이 용이해진다. 예를 들자면, 현 자원의 다른 버전들 (hasVersion), 딸림자료들 (hasPart), 대체자료 (isReplacedBy), 같은 내용이지만 다른 포맷들 (hasFormat), 참고자료들 (references) 등을 검색할 수 있게 될 것이다.
- OWL의 'owl:samePropertyAs'를 활용하면 속성명은 달라도 의미가 동일하기 때문에 정확성을 유지한 확장검색이 가능하다.
- 검색결과 자료가 지나치게 방대할 경우, 검색대상으로 취한 속성의 하위 속성 온톨로지를 참조한 협의의 검색을 쉽게 시도할 수 있다.
- 예를 들어 'James Hendler'라는 이름의 '연구원'에 대한 정보 검색이 가능해진다. 흔히 현재의 웹 검색엔진으로 이 검색이 된다고 생각하기 쉽지만, '연구원' 신분에 대한 여과 없이 진행할 경우에는 '연구원' 아닌 'James Hendler'가 다수 검색될 가능성이 크다[15].

- 'James Hendler'가 공저한 'SHOE'에 관한 논문을 참조한 다른 논문들의 검색 또한 SHOE는 shoes와 다르다는 온톨로지 설정을 통해 용이해진다.
- 나아가 서지정보와 날짜의 구조를 알아야 가능한 'James Hendler' 의 'SHOE'에 관한 최근 논문의 검색 같은 질의가 가능해진다.

이하는 DAML을 사용한 프로젝트에서 현재 시도하고 있는 질의의 유형들이다.

'이력서'와 관련된 DAML 프로젝트

- 입사 지원자와 같이 일해 본 경험이 있는 사람을 검색하라.
- 지식표현 (Knowledge Representation) 분야에서 Java를 사용한 경력이 있는 사람을 검색하라.

---

15) Grit Denker, et al. Accessing Information and Services on the DAML-Enabled Web. Semantic Web Workshop 2001. Hongkong, China.

Wall Street Journal DAML 프로젝트

- 회사에 관한 기사는 즉각 '보통사항'으로 통보하고, 만약 좋지 않은 측면에서 기사에 쓰였으면 '긴급사항'으로 통보하라.
- 표준기업 분류표에 근거하여 경쟁사에 관한 기사가 나올 경우 즉각 통보하라.

이 논문에서 살펴본대로 시맨틱웹 관련기술은 정보검색과 정보서비스에 다각도의 강력한 가능성을 시사한다. 무엇보다도 우리나라에서 개발되는 메타데이터 스키마들의 네임스페이스 관리가 있어야 하고, 국제 메타데이터 네임스페이스들과의 연계관계를 성립하여 국제적 호환성을 증진시킬 필요가 있다. 이러한 효율적인 메타데이터 스키마 설계를 위해 XML 스키마와 RDF 스키마를 적극적으로 활용해야 할 것이다.

나아가 메타데이터 요소 간의 상호운영성을 넘어서는 의미적 상호운영성을 이룩하기 위해 시맨틱웹이 해결해야 할 급선무를 주요 서비스 분야의 온톨로지 개발로 본다면, 시소러스 구축에 관한 문헌정보학적 연구는 이의 효과적 형성과 발전에 매우 필요한 핵심 지식을 제공할만한 것이다. OWL 에서 시소러스에 사용되는 기본관계를 더 확장하긴 하였으나 시소러스 구축의 핵심지식은 온톨로지 구축에 상당한 영향력을 행사할 것으로 기대된다. 이러한 시맨틱웹 연구에 동참하기 위해서는 XML/RDF 스키마의 구문구조와 OWL에 대한 이해의 폭을 넓히는 노력이 따라야 할 것이다.